Copyright is owned by the Author of the thesis. Permission is given for a copy to be downloaded by an individual for the purpose of research and private study only. The thesis may not be reproduced elsewhere without the permission of the Author.

Design and Development of a Hybrid Flexible Manufacturing System

A thesis presented in fulfillment of the requirements for the degree of Master of Technology at Massey University

Volume 1

By

Matthew J Jolly

1999

MASSEY

University

Abstract

The ability of a manufacturing environment to be able to modify itself and to incorporate a wide variety of heterogeneous multi-vendor devices is becoming a matter of increasing importance in the modern manufacturing enterprise. Many companies in the past have been forced to procure devices which are compatible with existing systems but are not as suitable as other less compatible devices. The inability to be able to integrate new devices into an existing company has made such enterprises dependent on one vendor and has decreased their ability to be able to respond to changes in the market. It is said that typically 60% of orders received in a company are new orders. Therefore the ability of a company to be able to reconfigure itself and respond to such demands and reintegrate itself with new equipment requirements is of paramount importance.

In the past much effort has been made towards the integration of shop floor devices in industry whereby such devices can communicate with each other so that certain tasks are able to be achieved in a single environment. Up until recently however much of this was carried out in a very much improvised fashion with no real structure existing within the factory. This meant that once the factory was set up it became a hard-wired entity and extensibility and modifiability were difficult indeed. When formalised Computer Integrated Manufacturing (CIM) system architectures were developed it was found that although they solved many existing shortcomings there were inherent problems associated with these as well. What became apparent was that a fresh approach was required that took the advantages of existing architectures and combined them into an new architecture that not only capitalised on these advantages but also nullified the weaknesses of the existing systems.

This thesis outlines the design of a new FMS architecture and its implementation in a factory environment on a PC based system.

Table of Contents

VOLUME 1:

Abstract ii

Table of contents iii

List of Tables vii

List of Figures viii

Acknowledgements x

Glossary xii

CHAPTER 1 – LITERATURE REVIEW 1

1.1 Introduction 2

1.2 Types of Manufacturing Systems 2

1.2.1 Project Shop 2

1.2.2 Job Shop 2

1.2.3 Flow Shop 3

1.2.4 Continuous System 3

1.2.5 Summary 3

1.3 CIM-OSA Computer Integrated Manufacturing 4

1.3.1 Introduction 4

1.3.2 Features a Quality CIM System should exhibit 4

1.3.3 CIM-OSA Integration Requirements 5

1.3.4 Building a CIM-OSA Model 7

1.3.5 The Integrating Infrastucture 14

1.4 CIM Control Architectures 17

1.4.1 The Centralised Form 18

1.4.2 The Proper Hierarchical Form 19

1.4.3 The Modified Hierarchical Form 20
1.4.4 The Heterarchical Form 20
1.4.5 Discussion 23
1.5 Centralised and Distributed Scheduling Systems 26
1.5.1 Overview 26
1.5.2 Centralised Scheduling and Distributed Scheduling: A Definition 26
1.5.3 Centralised Scheduling and Distributed Scheduling: Performance Evaluation 28
1.6 Contract Net Protocol – A Distributed Problem Solver 30
1.6.1 Introduction 30
1.6.2 Advantages of the Negotiation Approach 33
1.6.3 Discussion 34
1.7 Summary 35

CHAPTER 2 – DEVELOPMENT OF THE HYBRID CONTROL SYSTEM FOR FMS 36
2.1 Introduction 37
2.2 Purpose of the Research 38
2.3 System Architecture 38
2.4 Hybrid Architecture Overview 40
2.4.1 Introduction 40
2.4.2 Features of the Hybrid System 41
2.5 The Hybrid Model - An intimate look 46
2.5.1 The Hybrid Architecture - General Services 49
2.5.2 The Hybrid Architecture – Factory Controller Based Services 59
2.5.3 The Hybrid Architecture – Cell Controller Services 69
2.6 Summary 86

CHAPTER 3 – IMPLEMENTATION OF THE HYBRID MODEL – GENERAL SERVICES 87
3.1 Introduction 88
3.2 Overview – Intellution FIXDMACS GUI 89

3.2.1 I/O Drivers 89

3.2.2 Database 90

3.2.3 Scan, Alarm and Control Program 91

3.2.4 Man-Machine Interface 92

3.2.5 Handling Multiple Views of system wide Equipment 93

3.2.6 Limitations of FIXDMACS Software 94

3.3 Implementation of the Hybrid FMS – General Services 96

3.3.1 Implentation of the UMS System Model 96

3.3.2 Implementation of the Node Polling Application 96

3.3.3 Implementing the Task Scheduling Service 98

3.3.4 Implementing the Resource Scheduling Service 116

3.4 Summary 124

CHAPTER 4 – IMPLEMENTATION OF THE HYBRID MODEL – FACTORY CONTROLLER BASED SERVICES 125

4.1 Implementation of the Hybrid FMS – Factory Based Services 126

4.1.1 Implementation of the Factory Configuration Service 126

4.1.2 Implementation of the Factory Controller Parameter Setup Service 138

4.1.3 Implementation of the Resource Configuration Service 140

4.1.4 Implementation of the Job Definition Service 153

4.2 Summary 161

CHAPTER 5 – IMPLEMENTATION OF THE HYBRID MODEL – CELL CONTROLLER BASED SERVICES 162

5.1 Implementing Cell Controller Based Services 163

5.1.1 Implementation of the Address Philosophy 163

5.1.2 Implementation of the Manufacturing Process Services 168

5.2 Summary 185

CHAPTER 6 – TESTING THE HYBRID FMS SYSTEM 186

6.1 Introduction 187

6.2 Hybrid System Configuration 188

6.2.1 Modelling the VMD Behaviour 190

6.3 Experiment 1 – Standard System 193

6.3.1 Results of Experiment 1 193

6.4 Experiment 2 – Process Failure on Cell 1 196

6.4.1 Results of Experiment 2 196

6.5 Experiment 3 – Buffer Management 198

6.5.1 Results of Experiment 3 198

6.6 Experiment 4 – Standard test with variable Intercellular Transit times 200

6.6.1 Results 201

6.7 Conclusions 204

CHAPTER 7 – MANUFACTURING ORGANISATION IMPLICATIONS AND RECOMMENDATIONS 205

7.1 Hybrid System Strengths and Limitations 206

7.2 Implications of Hybrid FMS for Industry 210

7.3 Future Work 212

CHAPTER 8 – SUMMARY 214

8.1 Summary 215

REFERENCES 217

APPENDIX A HYBRID FMS ALGORITHMS

APPENDIX B FIXDMACS FACTORY DATABASE BLOCKS

APPENDIX C FIXDMACS CELL CONTROLLER DATABASE BLOCKS

APPENDIX D EXPERIMENTAL DATA – TESTING THE HYBRID FMS SYSTEM

APPENDIX E EXPERIMENTAL JOB AND TASK DETAILS

VOLUME 2:

INTRODUCTION

APPENDIX A CELL CONTROLLER SERVICES – VISUAL BASIC IMPLEMENTATION

A.1 IMPLEMENTATION OF THE QUEUING APPLICATION

A.2 IMPLEMENTATION OF THE AUCTION MANAGER APPLICATION

A.3 IMPLEMENTATION OF THE HYBRID PROGRAM DOWNLOAD APPLICATION

A.4 IMPLEMENTATION OF THE HYBRID PROCESS CONTROL APPLICATION

A.5 IMPLEMENTATION OF THE BID FORMULATION APPLICATION

A.6 IMPLEMENTATION OF THE NODE POLLING APPLICATION

APPENDIX B FACTORY CONTROLLER SERVICES – VISUAL BASIC IMPLEMENTATION

List of Tables

Table 1.1 Advantages and Disadvantages of Centralised Architectures 21
Table 1.2 Advantages and Disadvantages of Proper Hierarchical Architectures 22
Table 1.3 Advantages and Disadvantages of Modified Hierarchical Architectures 22
Table 1.4 Advantages and Disadvantages of Heterarchical Architectures 23
Table 1.5 Comparison of Performance between Heterarchical and Hierarchical Architectures 25
Table 1.6 Features of Distributed Scheduling: Performance Evaluation 28
Table 1.7 Results of Simulation Based Assessment of a Distributed and Centralised Scheduling Mechanism 30

Table 3.1 Hybrid Architecture bidding objects 103

Table 7.1 Strengths and Weaknesses of the Hybrid System 207

List of Figures

Figure 1.1 Steps towards the integrated enterprise 6
Figure 1.2 Levels of Modelling under the CIM-OSA Paradigm 7
Figure 1.3 Business Process Derivation 8
Figure 1.4 Business Process Relationship Model 9
Figure 1.5 Enterprise Activity Input Output Model 9
Figure 1.6 Mapping CIM-OSA system onto Real World Environment 13
Figure 1.7 Integration of Applications through the services of the Integrating Infrastructure 15
Figure 1.8 Common CIM System Architectures 18
Figure 1.9a Centralised Scheduling Architecture 26
Figure 1.9b Distributed Manufacturing System Scheduling and Control 27
Figure 1.10 Information Flow Model of the Bidding Process 33

Figure 2.1 The Hybrid Architecture 40
Figure 2.2 Node Polling Model 45
Figure 2.3 Basic Model of the Bidding Algorithm 46
Figure 2.4 A Definitive look at the Hybrid Model 48
Figure 2.5 Model of the UMS Communication Protocol 49
Figure 2.6 Lowest Level Representation of the Hybrid FMS Model 50
Figure 2.7 Model of the Factory Based Task Allocation 53
Figure 2.8 Model of Resource Scheduling Service 58
Figure 2.9 Process Configuration Model 62
Figure 2.10 Tool Configuration Model 65
Figure 2.11 Hybrid Job Definition Model 67
Figure 2.12 Interaction of the Front End Addresses with the SFE 70
Figure 2.13 MFE – Lowest level of Representation 71
Figure 2.14 High Level Relationship of the Manufacturing Process Services with the Front End 74
Figure 2.15 Manufacturing Process Integrating Infrastructure 76
Figure 2.16 Role of Manufacturing Support Services in the Cell 82
Figure 2.17 Cell Configuration Data Storage Address Model 85

Figure 3.1 Basic Architecture of FIXDMACS 89
Figure 3.2 Example of a Database Chain 90
Figure 3.3 GUI Representation for a lathe 92
Figure 3.4 Typical Hierarchy of views – Factory Level 94
Figure 3.5 The FIXDMACS role in the Hybrid Architecture 95
Figure 3.6 Node Polling Application Window 97
Figure 3.7 Factory based Task Scheduling Application 99
Figure 3.8 Cell Controller Bid Formulation Program 104
Figure 3.9 Auction Manager Application 108
Figure 3.10 Queuing Application – In going and Out going Queue Status 110
Figure 3.11 Results Collection Form – Factory Controller 114
Figure 3.12 Manufacturing Process Event form 115
Figure 3.13 Main Factory Resource Scheduling Window 117
Figure 3.14 Cell Controller Bid Formulation Program, Tool Bidding Aspect 120
Figure 3.15 Role of Intercellular Traveltime in the Resource Allocation Process 123

Figure 4.1 Main Window for Factory Controller 127
Figure 4.2 Main Configure Factory Window 128
Figure 4.3 Main Node Definition Window 129
Figure 4.4 Process Configuration Form 130
Figure 4.5 Cell Layout Definition Form 136
Figure 4.6 Configure System Parameters Form 139
Figure 4.7 Front End for Process Definition 141
Figure 4.8 Define Capacity Bitmap form Showing Lathes Limiting Aspect 144

Figure 4.9 Define Maximum Limiting Factor window Showing Lathes Limiting Aspect 145
Figure 4.10 Controller Integration Algorithm 147
Figure 4.11 Define Tools Form 149
Figure 4.12 Statetools Form 150
Figure 4.13 Toollists Form 152
Figure 4.14 Def Job Form 153
Figure 4.15 CNC Drill Form 156
Figure 4.16 Cad Cam Form 158
Figure 4.17 Mastercam Package Main Window Showing Dimensioned Drawing 159

Figure 5.1 Model Showing Link between Virtual and Real World 163
Figure 5.2 Example of a Process Behaviour Algorithm – Mill 165
Figure 5.3 Model of the Generic NC Download Program 166
Figure 5.4 FIXDMACS Database Builder Showing Code for DOWNLDCNCI Program Block 167
Figure 5.5 Replication of the Manufacturing Process Integrating Infrastructure Model 169
Figure 5.6 The Hybrid Program Download Application 170
Figure 5.7 Queuing Application Main Window 173
Figure 5.8 Queue Status Form 174
Figure 5.9 The Results form is used to log events in the system that are important to the user 175
Figure 5.10 Results form showing the results of a Simulation Run 180
Figure 5.11 Process Control Application 181
Figure 5.12 Example of Capability Implementation using FIXDMACS Program Block 184

Figure 6.1 Interaction of the Virtual and Real Worlds – The Real System 191
Figure 6.2 The Simulated System 191
Figure 6.3 Queuecost Versus Time for the NC Milling Machines in each Cell 193
Figure 6.4 Queuecost Versus Time for the NC Grinding Machines in each Cell 194
Figure 6.5 Queuecost Versus Time for the NC Lathe Equipment in each Cell 195
Figure 6.6 Average Queuecost Versus Time for the NC Milling Machines in each Cell 195
Figure 6.7 Queuecost Versus Time for the NC Milling Machines in each Cell 196
Figure 6.8 Average Queuecost Versus Time for the NC Milling Machines in each Cell – Cell 1 Mill Failure 197
Figure 6.9 Queuecost Versus Time for the NC Milling Machines in each Cell – Variable Buffer Sizes 199
Figure 6.10 Intercellular Transit Times for Test 1 201
Figure 6.11 Intercellular Transit Times for Test 4 202
Figure 6.12 Queuecost Versus Time for the NC Milling Machines in each Cell – Cell 1 Isolated 203
Figure 6.13 Average Queuecost Versus Time for the NC Milling Machines in each Cell – Cell 1 Isolated 203

Figure 7.1 Norton System Watch as Used during Testing 210

Acknowledgements

I would like to thank all the people who have been involved in the research and development of the Hybrid FMS system over the last few years.

In particular I would like to thank my two supervisors Professor Don Barnes and Dr Saeid Nahavandi for their invaluable support and their vision in the project guidance.
I would like to thank Saeid Nahavandi for the many hours of assistance he gave to the project and also the excellent guidance and morale support given in times of crisis.

The contribution given to the project by Mr Len Chisholm (Computer Systems Engineer) cannot be ignored. Len spent many hours of his own time voluntarily assisting with setting up the required computer network systems and also assisting in the many hours of system debugging, without his contribution the success of the project could not be guaranteed.

I would like to thank Jill Bracegirdle for her help in formatting and printing this document.

Lastly, but certainly not least, I would like to thank my wife for the inspiration and support that she provided and also for the burden she sustained due to the many lonely nights endured whilst I completed my studies.

An old man going along a lone highway came at the evening, cold and grey, to a chasm - vast, deep and wide.
The old man crossed in the twilight dim.
The depths below held no fear for him.
And he stopped when safe on the other side,
And built a bridge to span the tide.
"Old Man!" said a fellow pilgrim near,
"You are wasting your time with building here.
Your journey will end with the close of day,
And never again will you pass this way.
You've crossed the chasm both deep and wide,
Why build this bridge on the eventide?"
The old man lifted his grey old head.
"Good friend in the path I have come", he said
"There followeth after me today,
A youth whose feet must pass this way.
This chasm which now is as naught to me,
To that unlearned youth a pitfall may be.
He, too, must cross in the twilight dim.
Good friend ..... I am building this bridge for him".....

"The Builder" (Author Unknown)

Glossary

AGV	Automated Guided Vehicle
Capability	A short item of logic used to carry out a given item of function
Centralised Control	A system in which decisions are made at one location within the FMS/CIM System.
CIM	Computer Integrated Manufacturing
DFMS	Distributed Flexible Manufacturing System
Distributed Control	A system in which decisions are made by a global decisonmaking process involving all entities (or some of the entities) in the FMS/CIM system
DIT	Data Image Table
EFT	Earliest Finishing Time
Fault Tolerance	The ability of a software based system to cope with faults and/or unexpected events
FMS	Flexible Manufacturing System
GUI	Graphical User Interface
HFE	Human Functional Entity
Loose Coupling	The situation whereby units of modular software have minimal interaction in terms of data transfer
MFE	Machine Functional Entity
NC	Numeric Control
PIL	Program Invocation Loop.
Proprietary Controller	The controller used to control the actual physical behaviour of the manufacturing process using a Numeric Control script
SCADA	Supervisory Control and Data Acquisition
SFE	Specific Functional Entity
SPT	Shortest Processing Time
VMD	Virtual Manufacturing Device. A software model that generically emulates the behaviour of a real world device and can be mapped onto it thereby giving real control.

Chapter 1

Literature Review

1.1 Introduction 2
1.2 Types of Manufacturing Systems 2
1.3 CIM-OSA Computer Integrated Manufacturing 4
1.4 CIM Control Architectures 17
1.5 Centralised and Distributed Scheduling Systems 26
1.6 Contract Net Protocol - A Distributed Problem Solver 30
1.7 Summary 35

1.1 Introduction

As indicated in the abstract the research being carried out is concerned with the design and development of a Hybrid Control System for a Flexible Manufacturing System (FMS). This chapter gives a review of all important subject matter relevant to the development of such an FMS. Areas that will be covered include the types of manufacturing systems currently being implemented, an overview of cellular manufacturing and flexible manufacturing, the architectures that are currently being implemented, an overview of the CIM-OSA project as well as current scheduling methods that are being implemented.

1.2 Types of Manufacturing Systems

Generally speaking a manufacturing system can be broadly classified as a series of processes each with a given sized queue of jobs required to be completed in front of them. The dynamics of the system are stochastic in nature[2] with the various features of the system such as job interarrival times, processing times and set up times being distributed about a mean value and this tends to be the case irrelevant of the system type.

There are four main classical types of manufacturing systems, these are

- Project shop
- Job Shop
- Flow Shop
- Continuous System

1.2.1 Project Shop

In a project shop the product being manufactured remains fixed and the parts that constitute the finished product are brought to it, such is the case in the ship building industry and aircraft industry.

1.2.2 Job Shop

A job is a work part or component to be manufactured, such manufacturing includes activities or operations such as turning, milling, grinding and forging. The job therefore consists of an operation or

number of operations to be executed on a number of distinct processes, when all operations are complete the job is complete.[1]

A job shop is characterised as one in which each one of a number of job's operations visit a sequence of machines in order that each job is completed fully.[2]
Typically a job shop exists in two forms:[2]

- **Open Shop** - one in which the job orders received are from outside concerns (make to order)
- **Closed Shop** - one in which the job orders are received internally (make to stock)

1.2.3 Flow Shop

The flow shop is one in which the processes are ordered in the sequence that the parts passing through them are to be processed, the idea being that this type of shop fosters high throughput. Typical examples of such a manufacturing system is the motor vehicle assembly line and in the meat processing industry where carcasses are transferred along a chain requiring various operations to be performed at various stations.

1.2.4 Continuous System

In contrast to the other types of flow shop which work on or produce discrete parts the continuous system involves the production of gases, liquids or slurries. As in the flow shop processes are arranged in a given order in which raw materials are fed in one end to produce the finished product at the other end.

1.2.5 Summary

The five systems as mentioned above are the classical types of manufacturing systems however they all have significant weaknesses associated with them. New developments in manufacturing systems have resulted in the development of cellular manufacturing systems that have benefits in the areas of improved product quality, reduced work-in-process, reduced lead times, reduced handling costs and improved operator efficiency and flexibility.[3]
ise the goals of the company.

1.3 CIM-OSA Computer Integrated Manufacturing

1.3.1 Introduction

Today worldwide competition between manufacturing enterprises has significantly increased. In order to survive, to grow and to maintain the margins of profit they are required to be able to compete in terms of quality, price and delivery time.[4] One of the weapons a manufacturing enterprise can use to consolidate its competitive position is the implementation of Computer Integrated Manufacturing (CIM).

The CIM-OSA is a total enterprise CIM design framework developed by a research project set up by the Commission of the European Communities in the framework of the Esprit program.

CIM-OSA provides a total enterprise modelling framework - in the framework the overall business requirements of the enterprise are captured and from this the physical system is derived. The system is the result of the recognition for the necessity of a structured top down approach from the derivation of business requirements to the design for those requirements and to the actual physical implementation of the system.[4] Other research carried out at Purdue University, USA, by the CIM Reference Committee has created an overall model of the enterprise using similar types of generic framework.[5]

1.3.2 Features a Quality CIM system should exhibit

- An optimal use of all the resources in an enterprise require that they be integrated into a CIM system, and that the result of the CIM integration is such that the flow of control and information is structured to model some proven heuristic.

  Structured control gives the following benefits.[4]

- Provides decision makers with up to date information allowing more accurate dynamic decisions to be made.
- Higher planning flexibility.

- Reduced work in progress and inventory.

Structured information flow results in a greater ability to access data and more predictable structured data results. This results in the following advantages:[4]

- Improved ability to exchange data between the stages involved in a product's life cycle allows better integration of each stage of the cycle, resulting in shorter time span between product design and marketing.

Having a structured information flow and control together results in the following benefits

- Improves product quality due to avoidance of errors
- Increased production flexibility
- Increased system flexibility

1.3.3 CIM-OSA Integration Requirements

There are two aspects of CIM related features that have to be considered in respect of CIM-OSA.

1.3.3.1 CIM Integration Aspects

CIM-OSA outlines a number of integration aspects that must be addressed in an enterprise wide CIM implementation in order for the developed system to reflect the integration in a comprehensive fashion. These aspects include:[4]

- Physical System Integration - Physical System Integration level is the lowest level and involves the physical interconnection of systems such as robots, CNC devices and computers and is one of the first aspects of integration that must be addressed as without the lowest level of integration successfully addressed, higher levels cannot be achieved.
- Application Integration - the Application Integration level deals with the information exchange between applications, the portability of applications between different devices and the distribution of applications and the use of standardised interfaces between applications. Layer 7 of the ISO

(International Standards Organisation) seven layer model deals with standards regarding the design and implementation of an application interface.[6-7]

- Business Integration - This is the highest level of all the aspects involved and addresses the integration of all the business processes at the corporate level, the business processes including concerns such as design, finance, accounting, marketing and production planning. It is the business level that is the driving force behind the requirements of the CIM design and implementation as it dictates the type of applications and physical systems that must be integrated.

1.3.3.2 CIM-OSA System Evolution

CIM-OSA treats the Business Integration needs as the motivation for the integration of the lower levels of the system to implement those needs. Therefore the CIM system is required to be updateable so that as the enterprise's business requirements change the system can change. This is achieved through the implementation of a generic structure and generic building blocks.

The combination of the three integration features are represented in the model[4] as shown in Figure 1.1

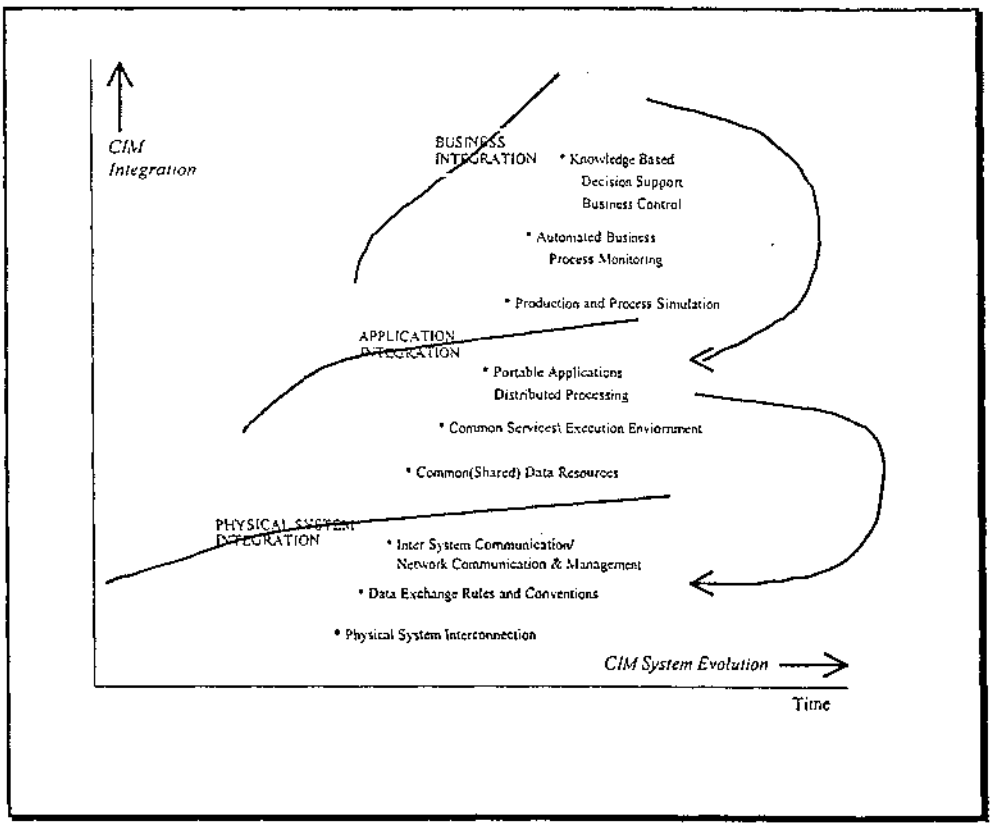


Figure 1.1 - Steps towards the integrated enterprise

The arrows in the model are representative of the derivation of the requirements of the lower level from the one above - this follows a logical derivation of the business requirements of the system, to the functional application requirements at the middle level to the derivation of the physical requirements at the lowest level.

1.3.4 Building a CIM-OSA Model

In section 1.4.3 we discuss the aspects which are required to be integrated in a CIM environment, however every business enterprise is different. Therefore we are required to construct a CIM-OSA implementation model for each enterprise considered and it is from the ensuing implementation model that the implementer finds solutions regarding what aspects are required to achieve the enterprise goals and how they are to be achieved.

The CIM-OSA model implements a top down approach[4,8-9] to CIM system design and implementation as shown in Figure 1.2. As can be seen from Figure 1.2 there are three levels of system modelling involved for the successful implementation of CIM for an enterprise under the CIM-OSA modelling paradigm.

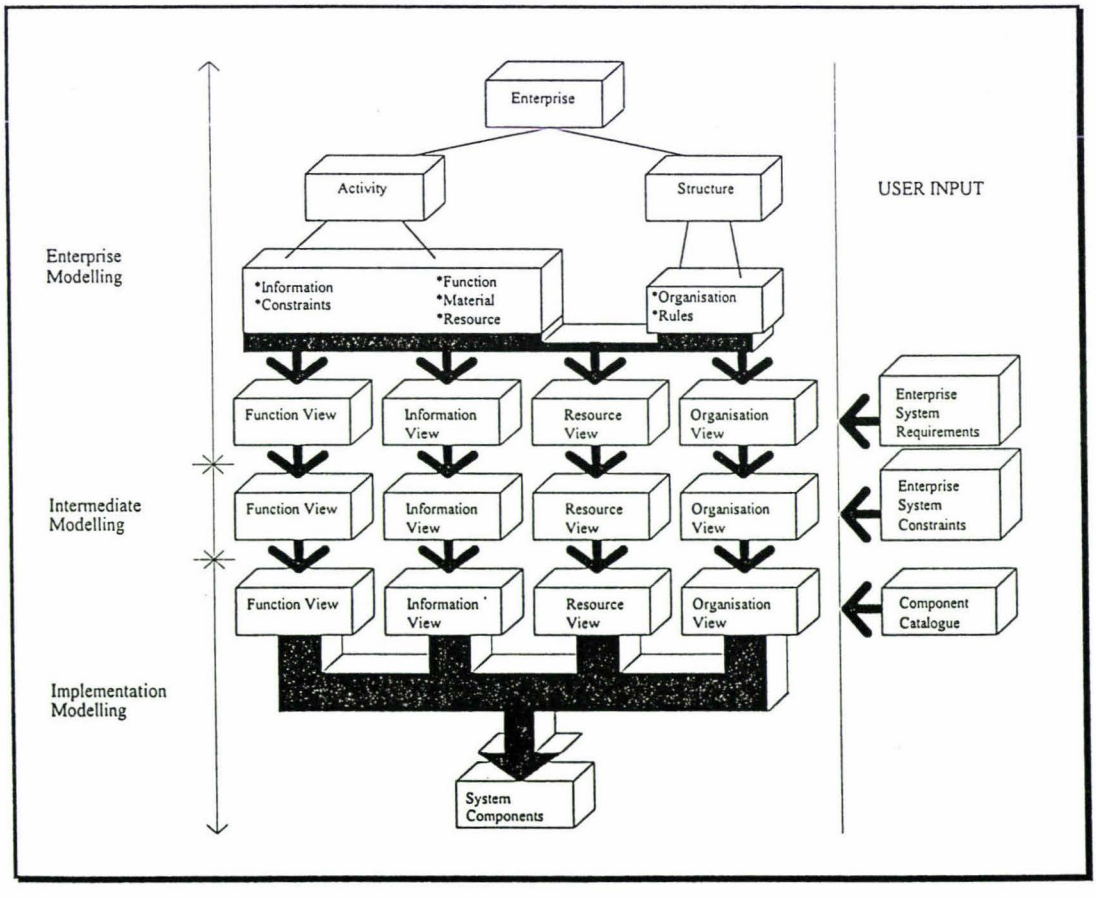


Figure 1.2 - Levels of Modelling under the CIM-OSA Paradigm

The system takes a top down approach to overall model formulation by modelling the goals of the enterprise at the top level (Enterprise Modelling) and then by using a system of derivation arriving at the implementation model at the bottom level which details how the specific implementation will be carried out.

Details of what is involved at each level of the modelling process is as follows:

1.3.4.1 Enterprise Modelling Level

The enterprise modelling level describes in a business sense and terminology what needs to be done within the enterprise. This is achieved in the following manner

- Identify all the business processes required to achieve the enterprise objectives and break each business process into a hierarchy of business processes required to achieve each main business process. Each business process is expanded into a hierarchy of spawned business processes required to produce results for the higher level business process, this is shown in Figure 1.3 as below

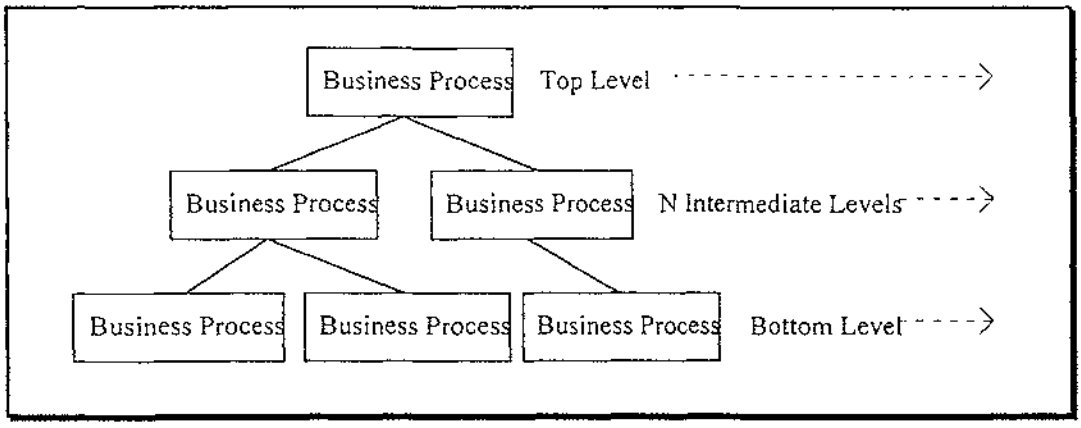


Figure 1.3 - Business Process Derivation

Therefore for each business process required to fulfil the goals of the enterprise we break the business process into a hierarchy of business processes. When the all the business processes has been derived we have all the business processes at the different levels of the hierarchy required to achieve the goals of the enterprise.

Each Business Process at each level is made up of a number of Enterprise Activities, each Enterprise Activity (EA) describing a given functionality of the enterprise, these EA's are not part of a given business

process but are used by it (as a function is), meaning that any business process can use any EA from the pool of EA's available.

Therefore looking at the lowest level of Figure 1.3 we have the situation as regards the relationship between two business processes as in Figure 1.4

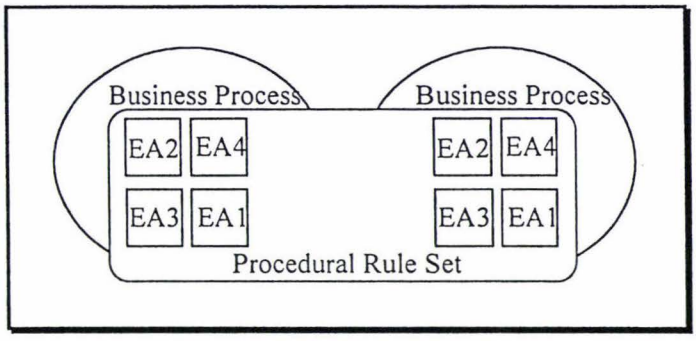


Figure 1.4 - Business Process Relationship Model

In Figure 1.4 the concept of the procedural rule set is introduced. Each business process has its own procedural rule set, the function of the procedural rule set is to define the flow of control between the EA's within a Business Process and also the flow of control between the Business Process housing the procedural rule set and the Business Process it is chained to.

This separation of functionality (EA's) and behaviour (Procedural Rule Set) means that it is easy to revise the overall CIM system design as depicted in Figure 1.1 because as the system is required to be changed to meet changing circumstances it is simply a matter of modifying the procedural rule set and constituent EA's for each Business Process.

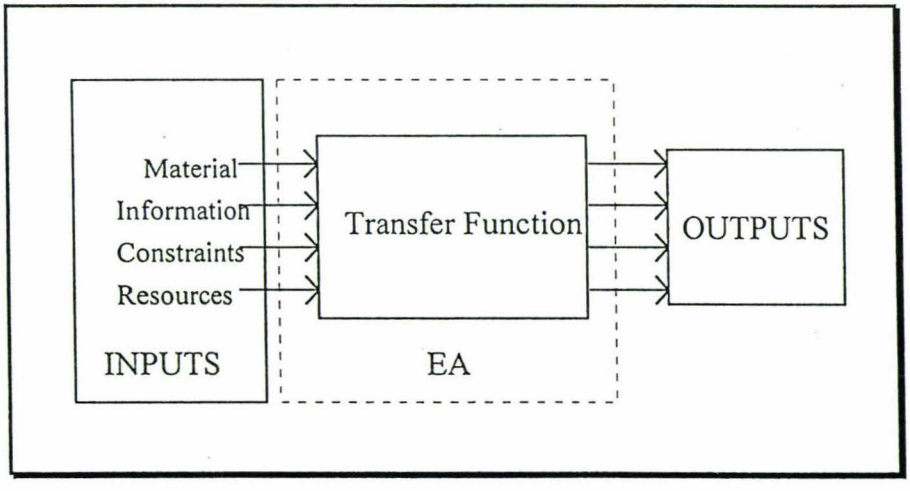


Figure 1.5 - Enterprise Activity Input Output Model

The enterprise model is finalised by breaking each EA down into the inputs required for each EA, the transfer function and the required outputs. The required types of inputs, the outputs and transfer function are shown in Figure 1.5.

The input types are:

- Material - types of materials required to input into the EA.
- Information - what type of information is required to be input to the EA
- Constraints - these are the types of constraints on the transfer function such as process definitions, work instructions and product information.
- Resources - these are the means required to carry out the transfer function such as the type of process required, equipment and tools and application programs.

For each EA required in the system the required inputs are defined in terms of the above input types and from this we can identify all the required materials, information, constraints and resources required to meet all the functions required for the enterprise. From this the Enterprise Modelling level has been completed as all the Enterprise System Requirements have been derived from the viewpoint of the Function, Information, Resource and Organisation aspects.

Therefore it is the implementation of these generic building blocks that give the system flexibility and extensibility. One important thing to note is that functionality (the EA's) and behaviour (Procedural Rule set) are clearly separated meaning that as the requirements of the enterprise change it is easy to change the system to meet those changes.

1.3.4.2 Intermediate Modelling Level

The enterprise modelling level describes what needs to be done within the enterprise in a business sense in terms of the function, information, resource and organisational view (Jorysz and Vernadat, 1990). The Intermediate Level of modelling deals with the design and organisation of the CIM-OSA system and focuses mainly on the information, resource and organisational view.[9]

- **The Information View**

All data used, processed or converted in the enterprise is represented at the intermediate level in terms of the Enterprise Object (EO), the EO being a generic handling system for any data being dealt with. The information view at the requirements level of the model gave a methodology for generically deriving the information needs for the specific CIM-OSA implementation for the enterprise. The Intermediate Modelling Level as explained by Klug and Tsichritzis provides a generic information modelling system defined along the three schema approach using Conceptual schemata[10]. The three schema approach classifies Enterprise Objects as consisting of three enterprise object views - internal, external and conceptual. The Conceptual view is considered at the intermediate level and deals with how the data derived at the requirements level (External View) is generically modelled (represented) in the system whereas the Internal view is developed at the Implementation Modelling level and deals with how the information is physically stored in the system.

- **The Resource View**

At the Enterprise Modelling level the types of resources required to meet the business requirements were derived by considering the inputs required to implement a Business Process. The Resource view at the Intermediate Modelling Level introduces a building block called the Logical Cell - the Logical Cell provides a mechanism to ensure the optimum grouping of equipment and resources to meet the objectives for the enterprise at the requirements modelling level. The nature of the resultant grouping may reflect a job orientated structure or business orientated one.

- **The Organisational View**

The Organisational View at the Intermediate modelling level describes the deployment of responsibility for the control and management of the resources, information and Business Processes. These responsibilities have to be arranged to fulfil the needs of the enterprise for decision making. The building block for the grouping of responsibilities within the enterprise is the Organisational Cell and provides a generic, structured decision making and control system on which decisions about the procurement of resources and subsequent execution of Enterprise Activities can be made.

1.3.4.3 Implementation Modelling Level

The implementation model is a description of the physical CIM system. It is created by examining the design requirements with respect to the different views of the Intermediate Model and making technical

choices about how the functionality is to be provided (functional view), how and where the information is to be stored (Informational View) and how the organisational structure is to be implemented.

The implementation level is mainly concerned with the development of the functional view. At the Enterprise Modelling level the Business Processes and Enterprise Activities were derived to meet the strategic needs of the enterprise, from this generic description which is implementation independent the implementation model is created.

As is shown in Figure 1.6 these Business Processes are controlled by an *executable* Procedural Rule set for each Implemented Business Process.

The Enterprise Activities (EA) are executable functions that can be used by a Business Process and are drawn from an EA Pool - the flow and action from one EA to another being dictated by the Procedural Rule set for that EA. Each Enterprise Activity describes the behaviour of part of the Business Process using a Transformation Function, the Transformation Function uses a set of generic Enterprise Objects as input describing the nature of the input required for the EA to carry out the required logical function.

The Transformation Function describes the behaviour of each required EA. However it is the implemented functional operations that model the behaviour of the required external physical components. Each of these modelled behaviours then interact with the external physical world to carry out the required actions to fulfil the Transformation Function.

The complex structure as shown in Figure 1.6 [9] has to be mapped onto a Physical Environment consisting of computers, machines, people and linked by communication systems. The physical world however involves a certain degree of required distributed functionality - the Implemented Functional Operation may have to communicate with a number of different Physical Devices, it does so using the Functional Entity Concept.

The functional Entity is an object able to transmit, receive and process data, it describes a physical world behaviour. The Functional Entity generically describes physical real world behaviour and not actual physical equipment - it uses a set of defined objects to describe and model the *generic behaviour* of actual equipment, this gives CIM-OSA the required flexibility at the implementation level. The functional entities are able to communicate with each other using transactions based on the MMS system. The MMS system uses a strict protocol of transmission of data packages using the client-server system, the data packages hold the messages being transmitted and each message is stored in predefined slots in the PDU (Protocol Data Unit) depending on the type of message being dealt with - this gives a common generic language between the Functional Entities as well as between the Functional Entities and the Physical

Devices. There has been considerable work already done on the benefits and implementation of such generic messaging systems.[7,11,15]

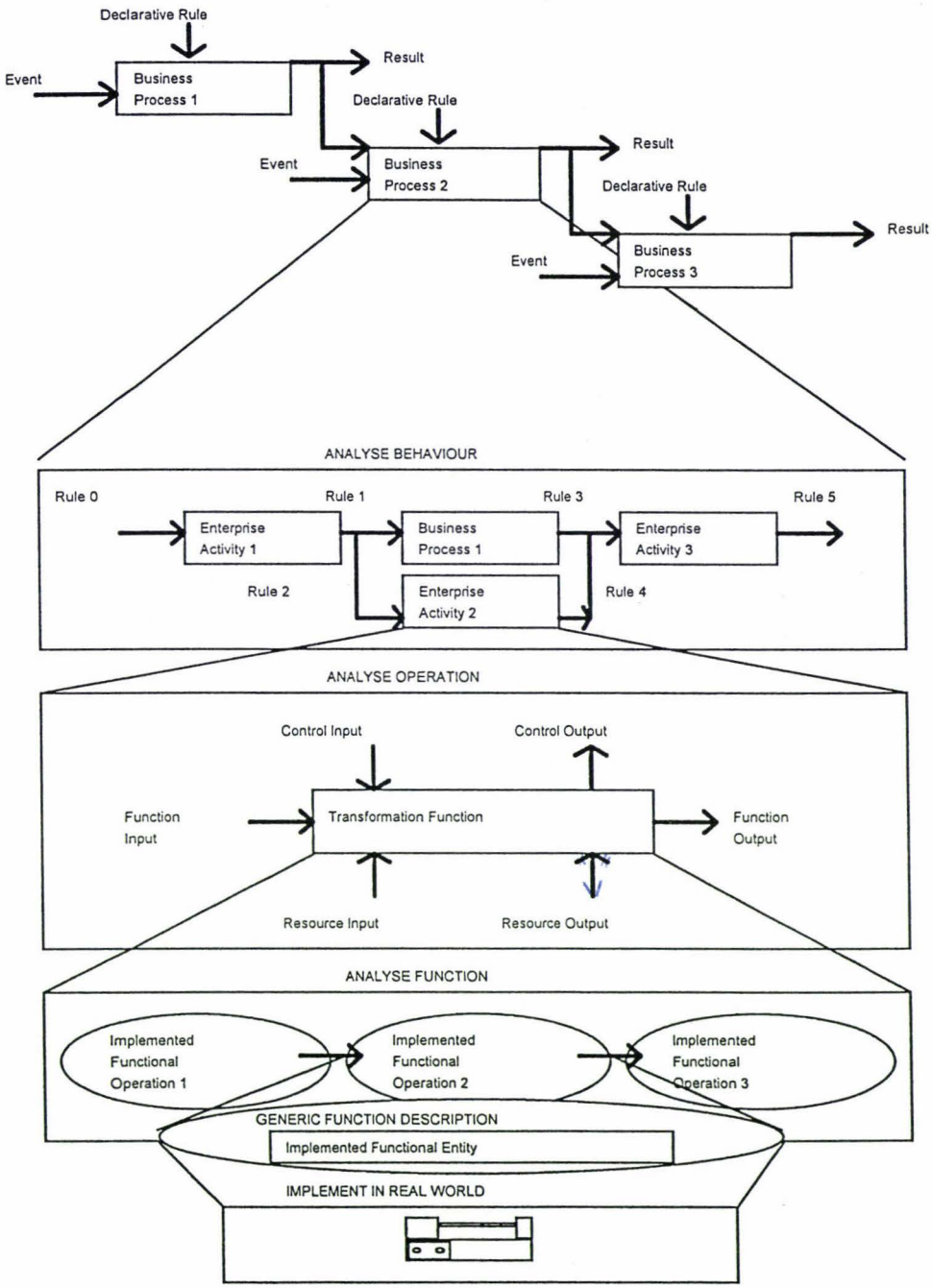


Figure 1.6 - Mapping CIM-OSA system onto Real World Environment

The physical environment is therefore described as a system of communicating Functional Entities that communicate with each other and drive the real physical devices in such a way as to meet the goals of the enterprise. CIM-OSA splits the Functional Entities into a number of sub categories:

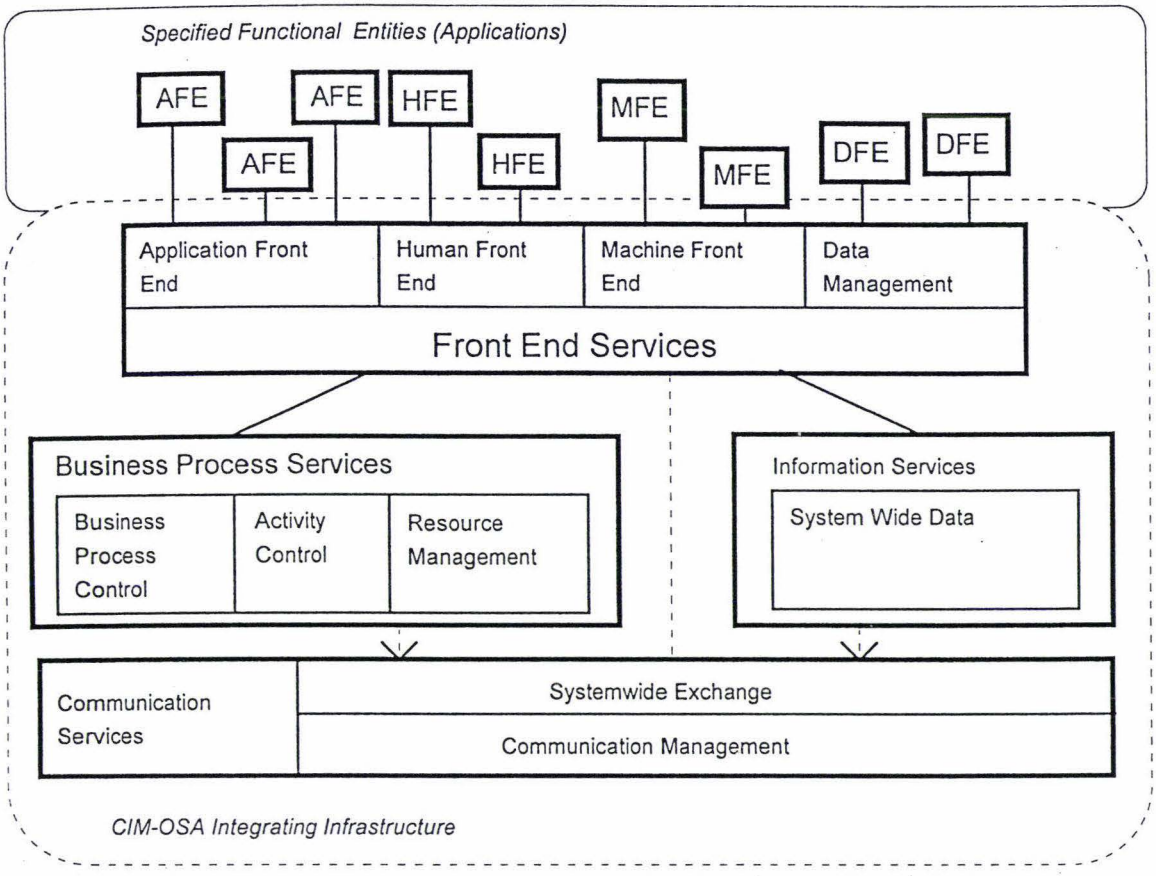
- Implemented Machine Functional Entities - describes generically the behaviour of a physical machine in such a way as this behaviour can be mapped onto a real machine to drive it as required.
- Implemented Human Functional Entities - model the behaviour of a human required to carry out the required Functional Operation in such a way as it can be mapped onto a real person.
- Implemented Application Functional Entities
- Implemented Data Storage Functional Entities
- Implemented Communication Functional Entities

Therefore it is possible in this way to model a complete organisation in terms of requirements to design to implementation by invoking such a top down approach, however it must be understood that the implementation of such a model involves a generic element - this is the role of the Integrating Infrastructure.

1.3.5 The Integrating Infrastructure

The paradigm of the Integrating Infrastructure is that there are a number of common services that are part of any corporate organisation and that there is a need not only to provide them but also implement them in a generic fashion - the ISO seven layer model[7,15] is a graphic realisation of this. The requirements of the model at the implementation level have been established, the integrating infrastructure binds the model together.

The model of the structure of the Integrating Infrastructure[8] is shown in Figure 1.7.



AFE: Application Functional Entities
MFE: Machine Functional Entities
DFE: Data Functional Entities
HFE: Human Functional Entities

Figure 1.7 - Integration of Applications through the services of the Integrating Infrastructure.

1.3.5.1 Front End Services

The Front End services provide a generic mechanism for interrogating the state of the specified functional entity and reacting to its state. The logic being along the lines of a 'generic' PLC - it provides the motor for the execution of physical actions or commands required of the machines, humans, applications and data management systems. The front end services use a Virtual Manufacturing Device[8,14] to emulate the status of the Specified Functional Entity and depending on the status a control decision is made using a generic control loop and a command is sent using generic MMS protocol to effect the behaviour of the real world device.

1.3.5.2 Business Process Services

There are three types of services under this umbrella, these being:

- Business Process Control Service - This service controls the execution of the actual business processes being executed by the SFE's by controlling the order of execution of the business processes and enterprise activities.[9]

- Activity Control Service - Controls execution of enterprise activities.

- Resource Management Service - Dynamically schedules all the resources required in the execution of the Enterprise Activities.

1.3.5.3 Information Services

Theses are generic services responsible for the addressing, storing , locating, retrieving and handling of data - the system wide data service also has to manage the context of the application-specific data according to the information model of the particular enterprise.

1.3.5.4 Communication Services

This service is generic and provides for system wide data handling and communications management providing access to OSI and other communication environments. The service has the following attributes:

- Location Transparency - physical location of services and Functional Entities are not important.

- Failure Transparency - Has mechanisms that provide strategies for the various failure modes that are not obvious and do not disrupt the system when they occur.

- Access Transparency - allows for the different types of access that are used by various communication systems.

- Performance Transparency - minimises the effect as much as possible of intermittent limitations in transmission performance.

1.4 CIM Control Architectures

Research into the design and implementation of computer integrated manufacturing systems has shown that a system is only as good as the architecture that it is developed under and that for a system to perform well it must be based on a sound structure.[1] The control architecture must be able to handle a complex and dynamic environment and must also be robust - this can be a difficult problem to solve but the selection of an effective architecture can go a long way. The challenge has been to design a system architecture that addresses the following performance characteristics:

- High Extensibility\Modifiability
- High Reliability\Fault Tolerance
- High Reconfigurability\Adaptability

By observing these various features it can be appreciated that different systems designed under various system architectures will exhibit different levels of performance in these areas and an understanding of these levels under various schemas is important so as to get an idea of the strengths and weaknesses of the different architectures. The design and implementation of the control architecture is most important as it breaks the overall control system into functional entities and the behaviour of these functional entities and their interactions determines the system's degree of overall performance in terms of the performance characteristics mentioned.

A number of different architectural structures are in existence and have been applied to CIM/FMS in the past and an evaluation of their form is required to gain an appreciation of why alternative control architectures may need to be considered.

Dilts *et.al* outlines four main types of control architecture are illustrated in Figure 1.8[1]

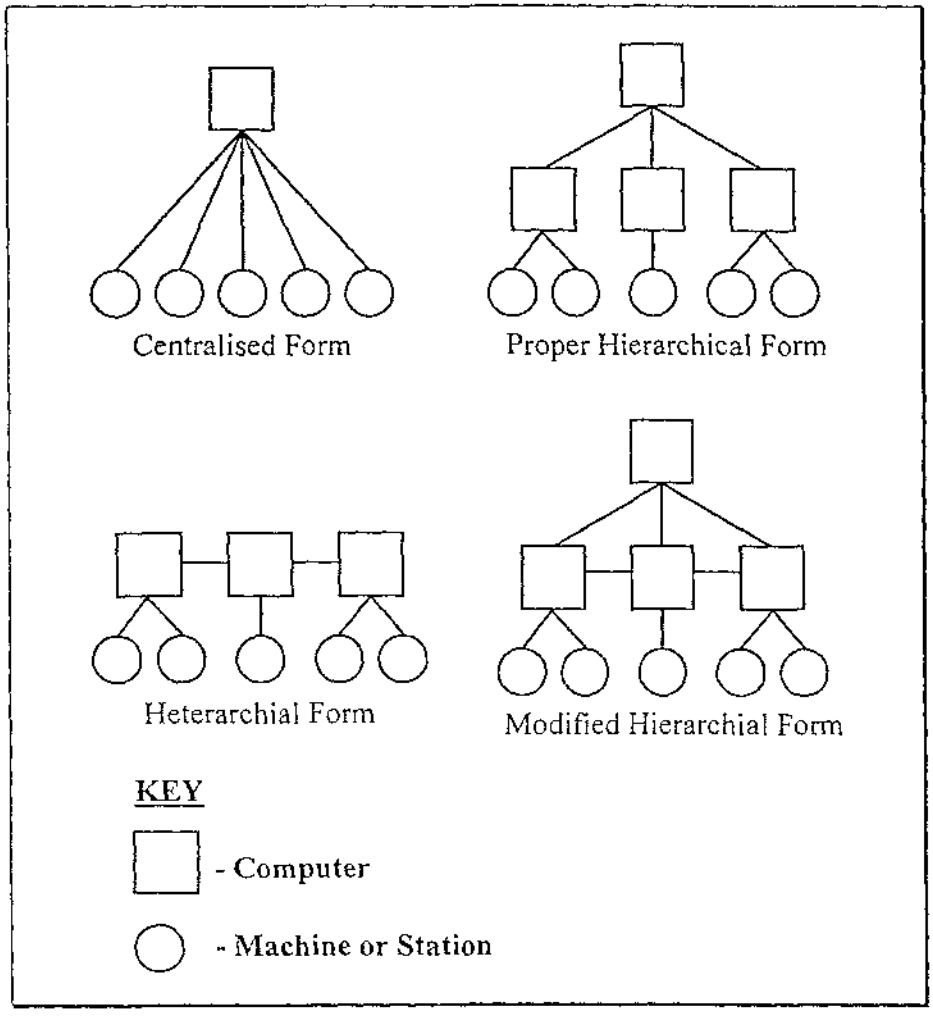


Figure 1.8 Common CIM System Architectures

1.4.1 The Centralised Form

The centralised architecture was the first attempt at CIM and generally involves the implementation of a centralised mainframe computer that acts as a manager and monitors the state of all processes in the factory.[1,16] The machine controllers execute commands received directly from the mainframe and it is these commands that dictate the manufacturing of the part directly from those controllers. In this environment the mainframe is responsible for the supervision of the overall system and the collection of data using shop floor sensors and uses this data to assess the overall enterprise and make decisions accordingly in a centralised fashion. Veeramani, Duffie and others point out that the major problem with the Centralised approach is the fact that failure of the centralised database can cripple the system.[1,16] It is

possible to design redundancy in the event of this failure however the centralised database would struggle in its ability to handle effectively all the database transactions in the CMS - thus "in all but the smallest organisations , performance and cost considerations will dictate distribution and function of data".[17]

1.4.2 The Proper Hierarchical Form

The shortcomings in the centralised control architecture coupled with advances in computer technology resulted in the development of the Proper Hierarchical form. The Proper Hierarchical (or more simply Hierarachial form) form of architecture was developed to emulate the Top-Down management structure that existed in many companies.[18,19] Strategic goals are set at the top most level of the hierarchy and the requirements are transmitted between levels top-down in a Master-Slave relationship, all activities of the subordinate level being dictated to by the supervisor level. The computer at the highest level of the hierarchy is responsible for the setting of the overall goals of the system and dictates and coordinates the lower levels in such a way as to achieve the required goals, the number of levels to the lower most level being undefined.[1,16,18,20-23] The only bottom up communication that is allowed in this architecture is for the reporting of the states of the various processes at the lowest level of the hierarchy to the highest level of the hierarchy.[1,16,18,21-23] Under the hierarchical control architecture the control information is input at the highest level of the structure and is broken down into a more detailed commands at the lower levels - modules at each level make decisions based on the commands that are received from higher levels and feedback received information from lower levels.[1,16,18,21-23] No information exchange occurs at the lower levels between the lower level modules rather communication only occurs with the level above the module concerned, the lower modules relying on supervision from the top to coordinate the required actions between them.[1,16,18,20-23] The lowest level of the architecture is responsible for shop floor activities such as machine control and acquiring sensory information, the top levels are responsible for control and coordination of the entire system.[20] Duffie, Chitturi and Mou state that a positive aspect resulting from the clear lines of communication obtained is that the response times of the system are fast but that the major problem with the architecture is that the structure of the system is fixed early in the design making unforeseen modifications difficult.[18,23] Duffie *et.al* and other researchers say that another problem with the hierarchical form is that once a system gets above a certain size and complexity it is impractical to define precise chains of command and operational detail - fault tolerance becomes expensive and difficult to allow for.[16,18,21-22,24] Hatvany himself made the statement that "no algorithms can be written which foresee every possible failure mode of a highly complex system, nor can remedial strategies be deterministically designed for every situation."[19]

1.4.3 The Modified Hierarchical Form

With the pure hierarchical form there was a major disadvantage - at the subordinate level there was no autonomy meaning that if a "break in the chain of command" occurred anywhere from the top of the structure to the bottom then the whole system would fail. The chain of command and the slave/master relationship is maintained with the Modified Hierarchical architecture but the degree of autonomy of the slaves is increased. When the first operation is carried out on the job at the first cell enough information is passed to it by the SCS (Shop Control System) so that it can arrange for the next operation at the second cell.[1] This allows the CCS (Cell Control System) the ability to supervise the operation and frees up the SCS to carry out more important tasks.

1.4.4 The Heterarchical Form

To overcome the limitations of the previous three architectures Dilts *et al.* proposed a new type of architecture - the Heterarchical architecture.[54] The Heterarchical form is a distributed type of architecture where the control of the overall system is done so in a distributed and opportunistic fashion. Many researchers have found that there are a number of limitations associated with hierarchical architecture, one of the major problems being increased complexity as the system expands and modifiability problems.[1,16,22,23] The core of this problem is in the inflexibility of the strict command protocol and master-slave relationships and because complicated command chains exist between modules.

With the Heterarchical form no Master\Slave relationships exist as inter-cell negotiation protocols such as The Contract Net Protocol (*Tilley et.al; Shaw; Smith; Veeramani; Ramasesh*) are used to distribute processing tasks between the cells in the processing environment.[39-40,57,55,2] Global information is eliminated or minimised in this architecture[1,22-23] which means a copy of all system required data is held and updated in each cell. This approach tends to aid in the designation and implementation of a fault tolerant system - this has been discussed by a number of authors.[1,23-24] A negotiation protocol is used between CCS's to decide where the operation required to be undertaken is best carried out, thus giving any CCS the capability of refusing to carry out an operation.[1] The design of the system is such that each CCS's required software is the same giving an implicitly modular system, each module within each CCS being able to be customised to suit the requirements of the system - this allows for reduced software complexity and implicit fault tolerance.[1] Comparisons of software complexity were made by Duffie and Piper and it was found that the number of lines of code required for a system with Heterarchical architecture was significantly less than the previous three architectures discussed.[26] Gremillion and Duffie *et.al* note that the number of lines of code are a good indicator of software complexity[21,27] while Dilts et.al states that the lower level of

complexity of the Heterarchical architecture coupled with its modularity will result in a system that is easy to extend and maintain.[54] Also the cooperative nature of the CCS's will continue to operate despite the failure of one or more companion CCS. The heterarchical architecture therefore allows the CCS's to redistribute work in accordance with the requirement of load balancing or in the event of machine or CCS failure.[1] Dilts et.al states that because of the weak couplings between CCS's and the modularity of the software within the system new cells can be easily added.[54] Other authors such as Pu *et.al* back up these statements saying that modularity and component reuse are important features that are required by a complete architecture regardless of the architecture used.[29]

Duffie notes that a major feature of hierarchical systems that causes problems is the fact that each level in the system relies on complex communication between levels due to the high level of global information, whereas heterarchical systems have loosely coupled interactions between the modules in the system.[23] Programmers need only be concerned with the inner workings of the module that they are developing and the types of messages required to be exchanged with other modules, they need not be concerned with the systems within other modules - this type of approach results in fewer logical programming errors and unforeseen interactions between modules.[23] Some researchers are of the opinion that the architecture is not based on a sound management philosophy that has been proven to be effective and that the system is still in its infancy and that more work is required before it becomes a recognised and proven architecture.[16,18]

Tables 1.1-1.4 below shows a summary of each architectures features, advantages and disadvantages.

Control Architecture - Features	Advantages	Disadvantages
Centralised • Single Mainframe Computer. • Control decisions made at a single location. • Global database records all system activities.	• Access to global information. • Global Optimisation Possible. • Single Source for System Status Information.	• Slow and Inconsistent Speed of Response. • Reliance on Single Control Unit. • Difficult to Modify Control Software.

Table 1.1 - Advantages and Disadvantages of Centralised Architectures[1]

Control Architecture - Features	Advantages	Disadvantages
Proper Hierarchical Form • A modified form of centralised control. • Based on traditional company management structures. • Supervisor controls activities of subordinates. • Rigid Master/Slave relationship between decisionmaking levels. • Integrated databases at each level. • Levels of control are arranged in a pyramid structure. • Control information flows down the hierarchy, feedback flows up.	• Reduced Software development time and subsequent problems. • Improved Fault Tolerance. • The rigid structure results in faster response times than the centralised design. • Emulates the corporate management structure therefore easy to map this structure onto the architecture.	• Unreliable inter-level links between layers of control. • Limitation of lower level controllers. • Failure at intermediate level of the hierarchy results in paralysis of the subordinate. • Difficult to extend/modify. • Global decisionmaking is based on obsolete data. • Failure recovery routines have to be programmed into the system. • Not all systems can be modelled hierarchically. Lack of availability of software. • Difficult to debug system. • No clear guidelines for system design currently available.

Table 1.2 - Advantages and Disadvantages of Proper Hierarchical Architectures[1,16,18,21-24,29,57]

Control Architecture - Features	Advantages	Disadvantages
Modified Hierarchical Form • Supervisor initiates sequence of activities in subordinates. • Loose Master/Slave relationships between decisionmaking levels. • Subordinates cooperate to complete required sequence of operations.	• All the advantages of proper hierarchical control. • Increased local autonomy means that the cells do not require supervision giving increased fault tolerance. • The overall system supervisor has more time available to carry out its tasks because some of its work is off loaded to subordinates.	• Control system design is more complicated. • Communication system between subordinates can pose connectivity problems. • Limitations of low level controllers.

Table 1.3 - Advantages and Disadvantages of Modified Hierarchical Architectures[1]

Control Architecture - Features	Advantages	Disadvantages
Heterarchical Form • A distributed form of Control. • No Master/Slave relationship • Full Local Autonomy • Distributed decision making for activity coordination. • Local databases only	• Reduced software complexity because of the system modularity and improved fault tolerance. • Shorter software development time than any others architecture • Improved fault tolerance due to local autonomy. • Improved system modifiability and extensibility. • Load balancing between cells easier to obtain. • Improved scheduling employing cooperative systems. Shaw found that distriduted scheduling performed better than centralised. • Amount of communication over LAN minimised because updating of ststem.	• Technical limitations of controllers. • No standards for communications, protocols or operating systems. • High likelihood of only local optimisation. • Requires a high network capacity. • Lack of availability of software. • The architecture is not based on a proven management philosophy and is therefore risky.

Table 1.4 - Advantages and Disadvantages of Heterarchical Architectures[1,23-25]

1.4.5 Discussion

The hierarchical architecture, Duffie et.al explains, is the most commonly applied architecture in the design and development of CIM systems.[23] Duffie et.al and other authors have explained that the reasons that this architecture was picked up quickly was due to the desire to emulate classical management structures.[1,18-19,24,30] Duffie and others authors such as Veeramani explain however that as the hierarchical systems grew they became more and more complicated with increasing associated costs, also because the structure of the systems was rigid the systems tended to become fixed and difficult to modify in the future.[18,21,24,57] Early pioneer of CIM systems Josef Hatvany also made similar comments in his early work stating that for large complex systems that the theology behind hierarchical systems of specifying each and every subsystem and operational detail was impractical and difficult to achieve.[30] Fault tolerance was achieved only at very high expense in this type of architecture and then it was not always reliable.[21,24]

Many researchers started to realise that the underlying architecture was the problem and started investigating other forms of architecture - the evolutionary approach to the development of system architectures is currently pointing to the heterarchical approach.

If a heterarchical architectural design is to be followed a systematic approach as suggested by Hatvany and Jorysz *et.al* is still required so that a comprehensive and complete system is the result, in fact other authors such as Veeramani *et.al* have noted that when the first systems were developed there were no systematic development guidelines to follow and therefore early systems were poor.[9,16,19] Hatvany suggested that one of the best ways to assure a successful design is to take a top-down integrated approach early in the design process, however he also warned that following the typical strict orderly hierarchical structure can lead to catastrophic collapse of the system in the event of failure.[19] The ESPRIT project takes this suggestion of Hatvany's and integrates it with the idea of a distributed non-hierarchical architecture and goal driven global optimisation to derive the CIM-OSA system[4,9] This paradigm helped to combat one of the main fears of the heterarchical architecture of local optimisation that was suggested by Duffie as being one of the possible pitfalls of this architecture and also negated the argument of a lack of system flexibility.

Researchers like Duffie and Hatvany realised that an architecture consisting of weakly coupled cooperating autonomous entities containing local information would promote better fault tolerance and a higher degree of modifiability.[21,24] Duffie stated that the elimination of Master\Slave relationships and Global Information enhances the attainment of implicit fault tolerance and significantly reduces the need for explicitly programmed fault tolerance.[18,21,24]

At the University of Madison-Wisconsin Duffie, Piper, Hartwick and Humphrey developed a flexible manufacturing cell to study hierarchical and heterarchical control strategies.[18] The FMS was identical for both strategies - it was only the control architecture that was modified. The study was set up to compare the differences in performance in a number of areas, the results of this study are shown below in Table 1.5. As can be seen the data from the table shows that the overall performance of the heterarchical system is far better and backs up the comments that have been made.
Much work has been done in the area of analysis of various architecture designs. From the experimental work and research and by gaining results as shown in Table 1.5 Duffie recognised that there were certain features of the architectures studied that led themselves to high levels of performance.

Recently and after extensive research Duffie, Chitturi and Mou presented the following set of design principles relating to a system with a high degree of modifiability and fault tolerance.[18,31]

- Master/Slave relationships should not exist.
- Entities should cooperate with other entities whenever possible.

Aspect	Hierarchical Cell Control	Heterarchical Cell Control
Lines of Source Code	2450	1235
Software Development Cost[1]	$61,250	$30,875
Expansion Software Cost[2]	$960	0
Machine Utilisation[3]	64%	60%
Memory Requirements[4]	50,608	39,2489
Average CPU Utilisation[5]	20%	60%
Complexity	High	Low
Flexibility	High	High
Modifiability	Average	High
Fault Tolerance	Low	High

[1] At $25 per line of Source Code

[2] 24 hours per machine added at $40 per hour

[3] From simulated fault free cell operation

[4] Scheduler bytes plus stack and date segments

[5] Four cell processors, scheduling 50%, control 10%.

Table 1.5: Comparison of Performance between Heterarchical and Hierarachial Architecture.[18]

- Entities should not take it for granted that other entities will cooperate with them.
- Entities should have minimal interaction with each other and where they do the interaction time should be minimised.
- Information generated locally should be retained locally rather than globally but should be available to other entities on request.

Using these rules it was thought that it was possible to develop a system that was highly modifiable, fault tolerant, did not become complicated when extended and was less likely to cause local optimisation. It is clear that the only practical means to achieve the requirements of the above guidelines is to adopt a distributed form of architecture.

1.5 Centralised and Distributed Scheduling Systems

1.5.1 Overview

CIM/FMS systems have as one of their components cells of manufacturing equipment the control of which is implemented by a shop floor controller (SFC) of some description. In the case of the Hierarchical architecture the cells have a centralised factory level controller (FC) that supervises the cells behaviour and gives the appropriate commands and distribute tasks as necessary. In the case of the Heterarchical system the SFC's communicate and negotiate between themselves to acquire and execute tasks. The common denominator of the systems is the need to communicate via Local Area Network Technology. The two different architectures implement two different scheduling paradigms - generally speaking the Heterarchical architecture uses a distributed method and Hierarchical uses centralised.

1.5.2 Centralised Scheduling and Distributed Scheduling: A Definition

Centralised scheduling as shown in Figure 1.9a was the traditional scheduling system and was reflective of the dominant manufacturing architecture at the time (Hierarchical).[31]

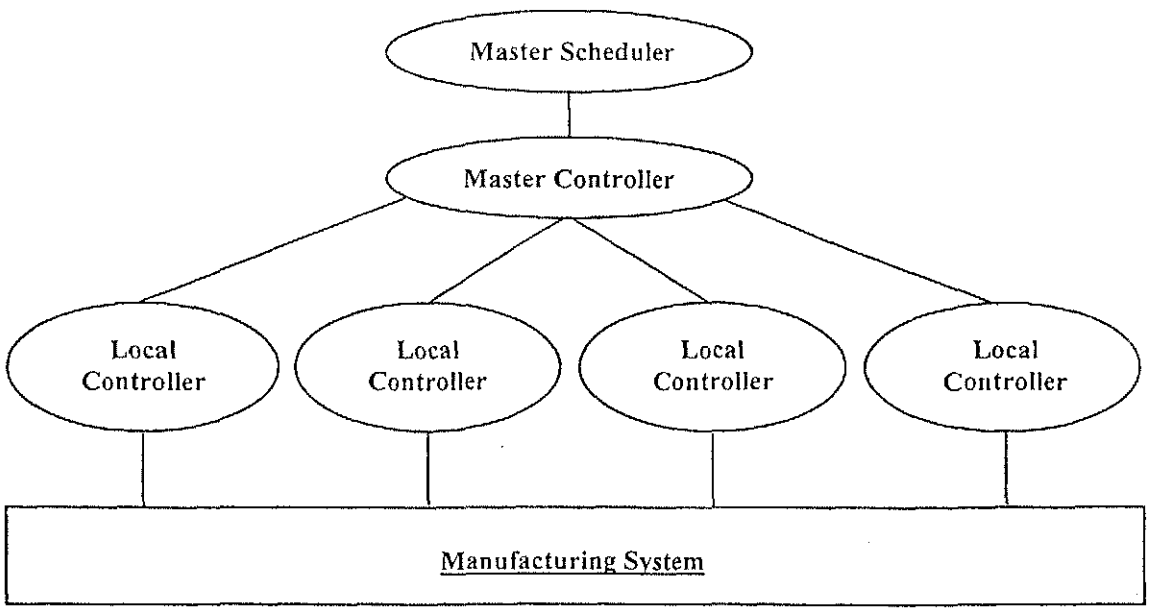


Figure 1.9a - Centralised Scheduling Architecture

As explained by Shaw, Duffie *et.al* and Kung *et.al* the implementation of a centralised scheme revolves around the scheduling of tasks at one location by the implementation of a master scheduler - shop floor

devices have no say in the task allocation process.[25,31-34] The master scheduler maintains a database of knowledge regarding the capability of devices at the shop floor cellular level and based on the devices capabilities assigns tasks to those devices. Whenever the processing status of a machine changes the Master Scheduler is informed - this requires the need for constant updating of the global database which tends to overload the network.[25,48] When tasks are introduced into the manufacturing environment for the first time the Master Scheduler looks at the state of the Manufacturing Environment and based on the state of the environment plans the path that the task will take through the system till completion. Duffie and other authors such as Bakker point out that this type of approach is complicated and may take several hours to execute meaning that by the time the schedule is generated it is obsolete and irrelevant.[31,35] Duffie also states that the simplification process required to make centralised scheduling tractable leads to ineffective or unrealistic schedules. Bakker says that the single attribute that most complicates scheduling is that what occurs in reality has little or no resemblance to that which was planned so that one event change can affect the whole plan.[35] This is the downfall of the centralised system. Kung in his research stated that "even for rather limited FMS, centralised optimal scheduling can be a complex problem defying direct solution".[34] Bakker also made negative comment regarding the centralised system stating that it lacked robustness, required considerable computing resources and is not reactive enough to the real situation.[35] It can be therefore be seen that the Centralised method of scheduling is rather limited in its scope of application.

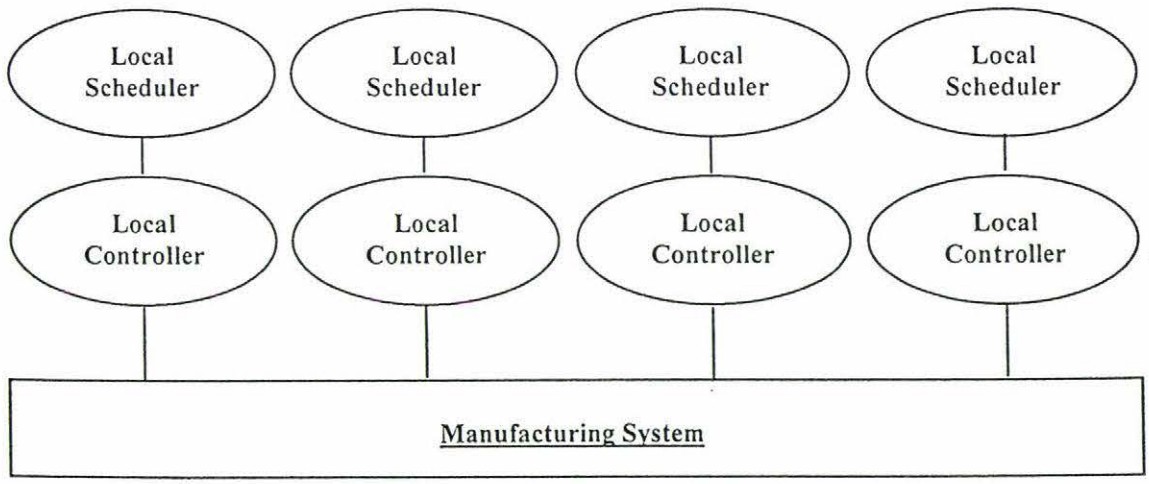


Figure 1.9b - Distributed Manufacturing System Scheduling and Control

Kung *et.al* describes Distributed control as being a means of data management by which "control and data are logically and geographically distributed".[36] The development of the Distributed Scheduling system as shown in Figure 1.9b coincided with the move away from the Hierarchical architectures to architectures such as the Heterarchical type. With the Distributed Scheduling paradigm all active cells have the

opportunity to compete for task assignment via a negotiation process. One form of this negotiation is by the implementation of a task auctioning system such as the Contract Net Protocol.[25,37-43,48] In the Contract Net Protocol approach the Master Scheduler role can be taken by any Local Scheduler - in this role it distributes information regarding the task to be carried out across the network to the cells within the environment. The cells on receiving the information formulate assessments of their current capability to execute the task, and transmit the information to the Local Scheduler for a decision. The Local Scheduler then makes a decision as to which cell receives the task and informs the winning cell.

Features of both Centralised and Distributed Control are summarised in Table 1.6[25,39,48]

	Centralised System	Distributed System
Control Structure	Centralised	Decentralised
Execution of Scheduling	A Master Scheduler	A Local Scheduler in each Cell can take the role as Master Scheduler
Scheduling Control System	Master-Slave control with uni-directional message-passing	Coordination through the exchange of messages between Cells
Vulnerability to failure of Scheduler	Entire system would collapse	Only that particular cell would fail
Manufacturing Database Management	A Global Database	Distributed databases
Maintenance of Dynamic System Information	Constant updating through communication messages	Local updating only required

Table 1.6: Features of Distributed and Centralised Scheduling Systems.

1.5.3 Centralised and Distributed Scheduling: Performance Evaluation

In 1986 Shaw set up a simulation to evaluate the performance of a Centralised Myopic - SPT (Shortest Processing Time) strategy with a Distributed negotiation-based approach using SPT as the basis - the results are presented in Table 1.7[25,32,40,48] The results are a culmination of twelve different simulation runs

based on using statistically generated input data, and on ten occasions the Distributed approach clearly out performed the Centralised paradigm. In his assessment of the results, Shaw stated that the Distributed scheduling method outperformed the Centralised system because of the fact that the Centralised system relied on constant database updating which overloads the network.[4,25,32,40,48] He also noted that the Distributed system scheduled more efficiently because the assessment process was based on more accurate up to date information due to the finite loading able to be placed on the network. Another point made by both Shaw and Smith is that because the Distributed scheduling system is carried out in parallel by a number of nodes on the network computational time is decreased.[4,25,39-40,48]

Shaw pointed out that other advantages that the distributed system had over the centralised system included:[39-40,48]

- Better Reliability because of the graceful degradation of the scheduler in the event of failure.
- Upward Extensibility - the control structure remains the same when new cells are added with minimal increase in network traffic.
- Scheduling is performed in parallel therefore the bottleneck produced by the global scheduler is eliminated.
- Cost Effectiveness -processing requirements are less therefore no necessity exists to buy more expensive computer equipment.

Smith also pointed out these potential advantages in pioneering work undertaken in the late 70's in the area of distributed scheduling using the auctioning system stating that distributed problem solving such as scheduling "offers the promise of speed, reliability and extensibility", Shaws simulation work showed these theories to be correct.[4]

Bakker carried out some as regards the creation of a new control structure based on the Distributed Control Structure and gave it the name DFMS (Distributed FMS).[35] In his work he identified the a number of advantages of the Distributed System including improved system reliability, better extensibility, conceptual simplicity and that it was more reactive to changes in the overall system schedule.

Zhang *et.al* made the point in a Robotics and Manufacturing Symposium in British Columbia in 1990 that another positive aspect with distributed control was that it had "the ability to handle applications with a natural spatial distribution".[36]

Williams pointed out advantages with Distributed system control when he stated in a paper in 1993 that they have "initiated a new era of ease of specification and installation".

In 1995 Ginting *et.al* developed a scheduling simulator as part of the research being undertaken at Massey University's Production Technology Department, the findings of trials undertaken during the research also supported the work carried out by Shaw and Smith and results were indicative of those in Table 1.7[44]

Bidding-SPT Results												
% of Jobs Late	15.85	21.80	32.56	22.05	23.25	28.78	41.20	23.75	43.28	47.43	67.34	39.55
Avg. waiting time	9.53	9.99	11.56	9.98	9.26	9.29	10.34	8.96	8.60	8.15	8.98	7.96
Avg. Lateness	4.90	5.23	6.38	5.24	2.32	2.54	3.41	2.78	1.42	1.44	1.91	1.31
Mean Flow Time	28.13	28.30	30.67	28.60	20.33	21.07	22.70	20.68	15.53	15.51	16.79	15.29

Myopic-SPT Results												
% of Jobs Late	24.11	29.89	37.27	27.28	25.56	29.15	43.46	26.54	44.13	46.95	66.83	39.93
Avg. waiting time	10.22	10.62	11.99	10.52	9.37	9.28	10.92	9.12	8.55	8.20	9.09	7.99
Avg. Lateness	6.18	6.35	7.17	6.89	2.92	3.47	5.00	3.65	1.65	1.87	2.32	1.53
Mean Flow Time	27.81	28.69	30.94	28.85	20.35	21.80	23.28	20.79	15.48	15.50	16.92	15.25

Table 1.7: Results of Simulation based Assessment of a Distributed and Centralised Scheduling Mechanism.

1.6 Contract Net Protocol - A Distributed Problem Solver

1.6.1 Introduction

The Contract Net Protocol is a distributed scheduling system used for scheduling everything from task scheduling in manufacturing companies[25], planning of freight transport[45] to the timely control of traffic light operation in the area of transport logistics. This very fact in itself shows the flexibility of use that the negotiation protocol has in that it can be applied in many varied instances. Smith *et.al* points out that the

framework is based on the human model of experts that cooperate to solve problems by the transfer of messages and use a contract negotiation process to distribute tasks to be executed concurrently.[41]

Shaw and others have already mentioned the benefits of the implementation of a distributed approach to FMS system control. The next step involves the choice of the model for such a system. The Contract Net Protocol and other negotiation type control systems have been used to achieve good system performance measures and have other positive attributes.[25,37-43,48]

Many authors such as Harris *et.al* and Smith *et.al* have shown that the bidding mechanism employed in the Contract Net Protocol can be used as an efficient way of allocating work and tasks within an organisation.[46-47] Cells in the networked environment use the negotiation protocol as a collective decisionmaking tool to resolve situations where there are a number of cells that are able to carry out a task they have in their possession.

When a cell (Manager Cell) has a job that it has worked on already and needs to allocate the next operation in that job (called a 'task') to a cell it initiates the negotiation based task assignment protocol by broadcasting a task announcement message over the network to other cells. Those cells that receive the message will in turn submit a price (called a 'bid') for carrying out the job - such a price maybe the earliest time that the cell can commence processing the task in the case that the Earliest Finishing Time heuristic is being used. If the cell is unable to carry out the task it does not submit a bid. When all bids have been received the manager cell then selects the cell with the cheapest price and the corresponding workpiece is then transported to the cell concerned. Shaw[25,32,48] constructed the information flow model for the which illustrates graphically the bidding process as shown in Figure 1.10.

If a manager cell possesses knowledge about other cells in the environment it can issue a *limited broadcast* to those cells and then the bid assessment by the Manager Cell can be based solely around responses from those target cells - the argument for this approach is that less overall computer time is tied up in the assessment process. Simulation work carried out by Tsukada *et.al* has shown that when the negotiation protocol is used to redistribute the work on a process which has broken down that the effect on the rest of the system is negligible in terms of the number of cells disrupted, the number of reschedulings required and the average makespan of tasks as compared to other recovery algorithms used.[49]

Shaw[25,32-39] defines the following features that are typical of systems that employ a network wide bidding protocol:

- Distributed control is used and no cell has greater importance, as far as control is concerned, than any other cell.

- The information used in the schedule is flexible and can take into account loading of processes, breakdowns and resource constraints in the bidding scheme.
- The system of bidding creates better schedules than traditional methods as they are created from real time data.
- Information processing requirements are mostly executed with the cells thus taking the load off the network.

There are a number of variations on the negotiation protocol theme, some of these are:

- Most negotiation protocols discussed in general literature assume that the system is implemented on a pure heterarchical structure but some authors such as Rabelo *et.al* have proposed implementation of the negotiation protocol on a semi-hierarchical platform.[25]

- In Shaw's system when a manager sends out a task announcement a given time period elapses before it assesses the bids received, if a cell has not prepared a bid in the meantime it misses out.[25]

- Task announcements can be made as general broadcasts (to all cells), as limited broadcasts (to a selection of cells) or as targeted broadcasts (to a cell).

- Tsukada believes that the best approach to scheduling is to preplan the schedule before any work is done and to only use the negotiation protocol as a task redistribution tool. He believes that the negotiation protocol is myopic and unpredictable and unproven in its ability to produce optimised solutions.[49]

- Bakker in his work suggests the use of the EFT (Earliest Finishing Time) heuristic as the best scheduling strategy as it has been shown to be the best performer compared to other heuristics such as the SPT (Shortest Processing Time) approach.[35]

- Fischer *et.al* in his work with truck scheduling heuristics has used the negotiation protocol in the scheduling of trucks. The trucks contain their own localised schedules and the auctioning of delivery orders occurs between the trucks - his work has shown that in the event of a truck breaking down the system allows for the next most appropriate truck to be selected without major global replanning being required.[45]

1.6.2 Advantages of the Negotiation Approach

Shaw has carried out extensive research in the area of negotiation protocols and has carried out much simulation work to verify his preconceptions about system features and performance. In work carried out by Shaw, Bakker and Rabelo *et.al* the following positive aspects of the negotiation protocol have been identified.[25,35,38-39]

- ***Minimal Network Traffic Needed*** - because the system is comprised of loosely coupled cells and communication is based on simple message passing minimal load is placed on the Network. If network capacity is at a premium the bidding system can be further improved using limited broadcasting.

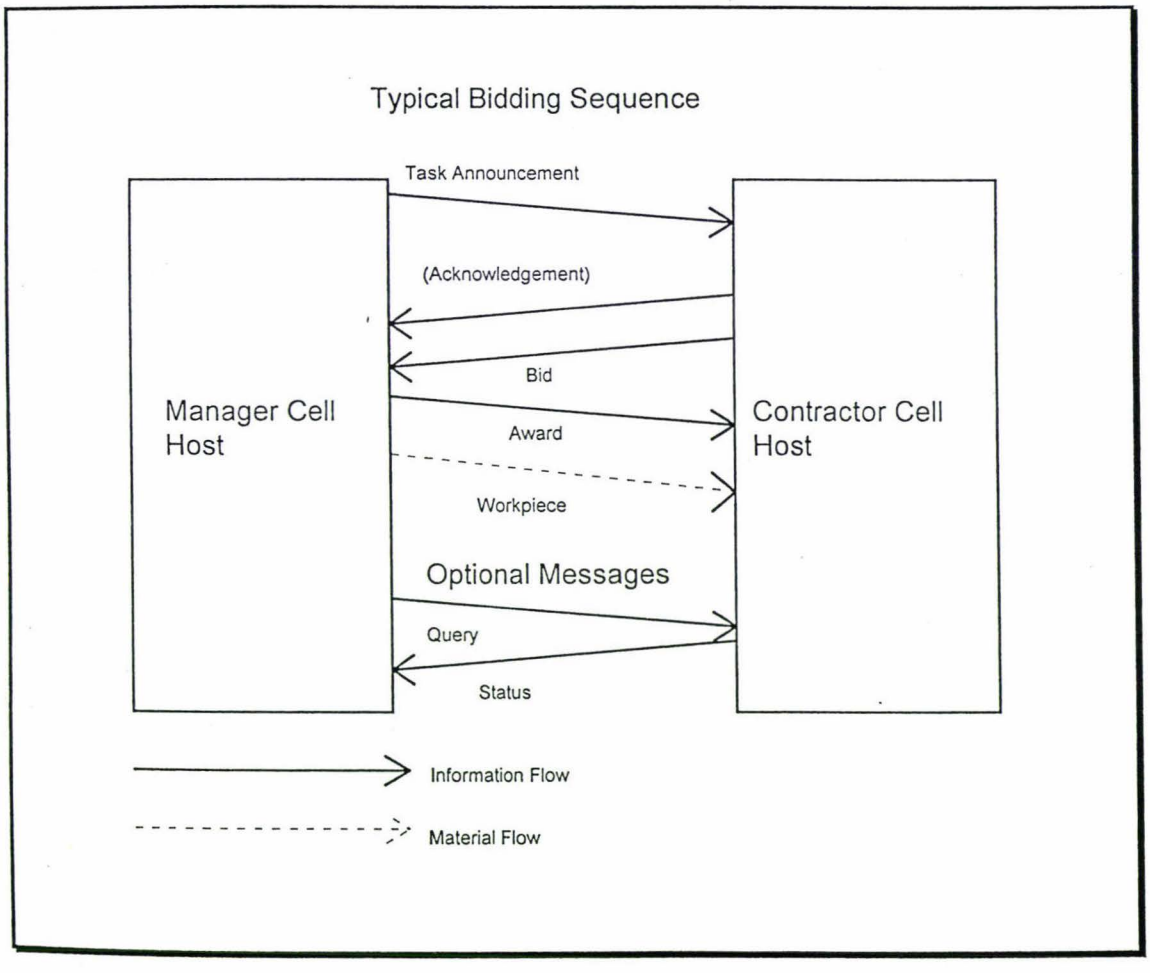


Figure 1.10 - Information Flow Model of the Bidding Process

- ***Implements Real Time Control*** - because the bid reflects dynamic status information near real time control can be achieved.

- ***Bottleneck Elimination*** - because data processing is carried out in parallel by all cells in the system the bottleneck at the centralised processor is eliminated.

- ***Flexibility*** - the bidding mechanism can be designed to reflect a wide variety of dynamic status information and desired levels of performance.
- ***High Reliability*** - Global databases are eliminated and bidder response is optional, therefore the auctioneer does not crash when a bidder does not respond to the auctioneer.

- ***High Extensibility*** - due to the highly localised data processing aspect and modular design it is easy to extend the system.

Simulation work carried out by Fischer found that a form of the negotiation protocol used for truck transportation scheduling gave levels of performance that were comparable to those of heuristic search OR (Operations Research) algorithms. He also discovered that his negotiation protocol had substantial advantages over those algorithms that agree with those findings of Shaw and Bakker such as the potential for reactibility to changes in system schedule, flexibility improvements and the ability to schedule in real time.[45]

Bakker however noted that there are a limited number of disadvantages but that the magnitude of these disadvantages outweighed the positive aspects.[35]

- *Slightly Limited Product Variety* - due to the finite numbers of different tools that can be used Bakker suggested that product variety is limited, but comments that "the impact of these limitations is small because most companies produce a limited set of products".

- *Limited Optimisation Possibilities* - Bakker suggests that the over simplification required using the negotiation protocol will not truly reflect true workplace dynamics.

1.6.3 Discussion

Auction based algorithms such as the Contract Net Protocol are very useful for the scheduling of tasks in that they provide a mechanism for real-time scheduling, are highly agile, easily extendable, and are highly adaptable to system disturbance. Shaw proved in his simulation work along with Ginting *et.al* that a properly designed job bidding system outperforms the centralised myopic approach in the areas such as job waiting time, job lateness and mean flow time[25,32,40,44,48] - this factor coupled with the other positive factors already mentioned provide us with an excellent alternative to traditional scheduling approaches which have proven to be very fragile.

1.7 Summary

This literature review has been presented to give an introduction into the latest theories regarding the modern FMS system so an appreciation of where this previous work fits in can be gained. The model of the Hybrid FMS presented in the next Chapter of this thesis will show how many of these paradigms are implemented. The literature review was written after carefull reading of up to 200 different papers relevant to modern FMS and CIM systems. As such a comprehensive and complete study has been undertaken to ensure an appropriate model of the proposed system can be created.

Chapter 2

Development of the Hybrid Control System for FMS

2.1 Introduction 37
2.2 Purpose of the Research 38
2.3 System Architecture 38
2.4 Hybrid Architecture Overview 39
2.5 The Hybrid Model - An intimate look 46
2.6 Summary 86

2.1 Introduction

The ability to integrate a number of hardware and software vendors into the one physical FMS environment is a very important factor in terms of the saleability and implementation threshold of a FMS package otherwise the factory becomes "a series of isolated processes requiring manual connection."[15] Therefore it is not just the integration of hardware systems that needs to be addressed but also the integration of various software systems that are required to be integrated with any service of a CIM package. Therefore an integrating environment which includes all integration requirements needs to be developed. However many authors such as Dilts, Hatvany, Gremillion and Duffie *et.al* have stated that the ability to integrate software and hardware into a FMS environment, extend that environment as the need arises and to modify it is largely a function relating to the underlying architectural paradigm.[18,21,24,27,54] Research carried out by Shaw showed that the Distributed scheduling architecture outperformed the Centralised architecture because the Centralised system relied on constant database updating which overloads the network.[25,32,40,48,55] In their work at the University of Madison-Wisconsin Duffie, Piper, Hartwick and Humphrey developed a flexible manufacturing cell to study hierarchical and heterarchical control strategies.[18] After an extensive period of trialing it was found that the heterarchical architecture outperformed the hierarchical architecture in the critical areas of flexibility, modifiability and extensibility.

With this in mind it is important to design the system architecture ensuring that the following aspects are considered:

- The main objectives and requirements that the system is being engineered for.
- The customisation or selection of an architecture that fulfils those needs and model development.
- Model validation and verification via simulation to realise objective attainment.
- System Development.

The main purpose of this chapter is to introduce the Hybrid Architecture in the light of the four points mentioned above. This architecture is a generic message based system that allows for the integration of a wide variety of software and hardware options in a fully networked environment. The architecture is termed "Hybrid" in that it is a theoretical combination of positive aspects of both the Heterarchical and Hierarchical systems. Extensive work has been done in other masterate level projects in the form of simulations and the Hybrid architecture has been thoroughly scrutinised and proven to have superior performance in a variety of areas.[44]

2.2 Purpose of the Research

It is clear from the work in Chapter 1 that there is a need to develop a practical integrating environment that utilises many of the latest theories in system and architectural design. Mullins states in his work that 55% of people in the United States are employed in small to medium sized manufacturing firms and that most systems that have been developed are financially out of range for these companies and are not of any practical use to them. [50] He also states that one of the most practical solutions for these companies is the implementation of PC and workstation systems in their quest for FMS. Skevington *et.al* believes that as many as 70% of all manufacturing carried out in the United States is done in small to medium sized companies and that the major problem for these firms is their ability to implement FMS is cost [51] . Skevington also says one of the reasons that FMS systems have failed is that they are based on unsound theories and are not adequately planned. Ferme [52] as well as Inglis[53] have both recognised the opportunity that is now available for implementing an FMS system in a networked PC(Personal Computer) environment. This has come about due to the dramatic increase in the use and power of personal computers, the rapid evolution of Local Area Networks (LANs) and the development of appropriate software over the last 10 years.

What is needed therefore is a well structured and developed FMS system that is useful for small to medium companies that provides the following features:

- Is PC based to minimise cost.
- Has a tried and true architecture that promotes flexibility, extensibility and can integrate a wide variety of software and hardware systems.
- Uses a messaging system that promotes generic and consistent communication.
- Uses a scheduling system that is real time and reactive to real time scenarios and unexpected events such as breakdowns and general unplanned downtime.

Such a system needs to be developed along the lines of the design criteria established in Section 2.1.

2.3 System Architecture

Clearly the requirements of the system have been identified, however careful attention must be paid to architecture selection so that the CIM system developed is based on a sound foundation as is implied by Skevington.[51] There are three main architectures as outlined by Dilts *et.al* - these being the Centralised, Hierarchical and Heterarchical architectures.[54] As outlined in Chapter 1 there are limitations to these

architectures not only in areas such as practical application but also in the management paradigms that they represent - researchers such as Dilts, Duffie and Veeramani [16,18,21-23,27,30,37-39,54] have shown this to be the case.

The Centralised architecture is very slow and inconsistent in its speed of response, relies on a central control unit that could fail and cause the whole system to break down and because it has a very structured approach modifiability and extensibility is very difficult.

The Hierarchical architecture paradigm was derived in an attempt to emulate the classical top down management system that was the norm for company management structures at the time of its conception. This was seen by researchers as being a major problem in that the architecture and structure of the system was fixed early in the design making unforeseen modifications difficult.[18,23] Another major problem was that the theory the architecture was based upon was fast becoming obsolete as other management paradigms such as TQM, Team Building, Quality Circles and JIT manufacturing became the latest theories in effective management. [56] These new theories clearly did not conform to the Heterarchical architecture as they relied on discussion across the management boundaries and feedback up and across the management structure not a unidirectional dictatorial approach. Another problem with the architecture is that not all systems could be modelled hierarchically. Also failure at one part of the system would paralyse the levels below it.

The heterarchical architecture was instigated to solve the problem of system paralysis due to component failure and to allow for improved extensibility and modifiability. It achieved these objectives by implementing a modular approach to software development that incorporated 'loose' inter-modular ties. The architecture, however, suffered because it was not based on any proven management philosophy and therefore users were unable to get any guarantees that it would be able to be driven in a goal orientated fashion. Local optimisation was also a problem because there was no overall system manager overseeing proceedings.

The problems that have been identified throughout the literature review has led to the development of the 'Hybrid' architecture. The architecture is a combination of both the Heterarchical and Hierarchical architectures and combines the powerful features of both architectures in an attempt to nullify the negative aspects of each. This architecture is introduced in the following section.

2.4 Hybrid Architecture Overview

2.4.1 Introduction

The high level model of the hybrid architecture is shown in Figure 2.1 below. The architecture involves the implementation of a generic communication protocol over a Local Area Network whereby all nodes in the system can receive consistent information required for the execution of tasks. In the implementation of the architecture this is carried out over an ethernet system with the following communication features.

- Bus Topology - provides a simple, easy to maintain, wiring layout that is easy to extend and is highly reliable.

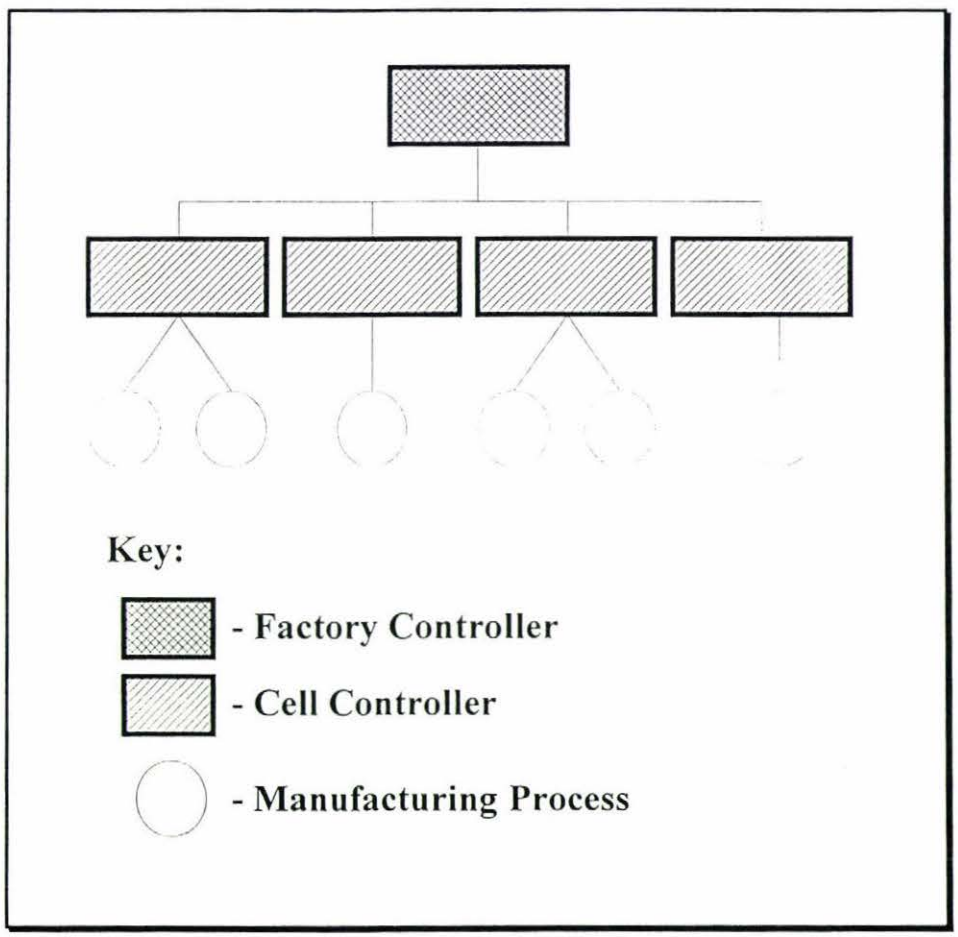


Figure 2.1 - The Hybrid Architecture

- CSMA/CD LAN access protocol - a contention based access scheme that senses message collision and retransmits accordingly, this system works best at ethernet loads of 50% or less and due to the fact that the system being developed is to be implemented in a job shop this loading would be seldom reached. This is because it is unlikely that this type of organisation would have an intense computer network system that would be in place in a large organistation having many departments and functions all sharing the network.

- Coaxial cable pair transmission medium provides the physical link between nodes and is better functionally than twisted pair as it has the ability to accommodate wider band widths and has a lower degree of attenuation than the twisted pair transmission system.

- Transmission Frequency - all nodes in the system transmit at the same frequency , arbitration between them being handled by the CSMA\CD access technique.

The system can also operate over a wide range of other systems (see Chapter 3) but for the purposes of conducting this resaerch the above configuration was used due to availablity.

Examining the model it is easy to speculate that the architecture is a variation of the Modified Hierarchical type, however the pretext behind the Hybrid structure is that negotiation exists not only between the Cell Controllers but also between the Cell Controllers and the Supervisor (in this case the Factory Controller). With the Modified Heterarchical architecture the paradigm is still a Master/Slave relationship between the Supervisor and the subordinate.

2.4.2 Features of the Hybrid System

The Hybrid architecture was designed to make use of many of the features of modern management philosophies - in particular the ideas of teamwork, management utilising worker expertise as well as other philosophies such as Goldratt's "Theory of Constraints" and JIT manufacturing.[6]

The Hybrid architecture also incorporates the ideas of modularity and the loose coupling that exists between them and has been designed to promote a high level of flexibility.

The Hybrid architecture implements a distributed control system. A distributed control system gives the following advantages as outlined by Shaw:[39,40,48]

- Good Reliability because of the graceful degradation of the scheduler in the event of failure.
- Upward Extensibility - the control structure remains the same when new cells are added with minimal increase in network traffic.
- Scheduling is performed in parallel and therefore the bottleneck produced by the global scheduler is eliminated.
- Cost Effectiveness - processing requirements are less and therefore there is no necessity to buy more expensive computer equipment.

Due to its modular nature models of inputs and outputs can be defined for each module and the software written for the module - this makes debugging a relatively simple procedure as data flow is predictable and easy to trace.

The Hybrid Architecture model has also been designed in accordance with the design principles of Duffie, Chitturi and Mou as stated below.[18,31]

- ***Master/Slave relationships should not exist*** - With the Hybrid architecture negotiation exists between the higher levels for both task and resource allocation. No longer can the higher levels of the architecture dictate terms in task and resource allocation, input must be fed from the lowest levels.

- ***Entities should cooperate with other entities whenever possible*** - in the event of process failure or shutdown the Cell Controller notifies the Factory Controller and it retrieves the tasks and redistributes them.

- ***Entities should have minimal interaction with each other and where they do, the interaction time should be minimised.*** In the Hybrid architecture entities are very loosely coupled. A messaging system exists between nodes and the degree of communication is kept to a minimum. An example of this loose coupling occurs where a Cell Controller falls over and the local information is retrieved locally after restart and during breakdown the Factory Controller bidding system does not assume that a response will be given by the cell and therefore it does not crash.

- ***Information generated locally should be retained locally rather than globally but should be available to other entities on request.*** With the Hybrid architecture there are many examples of this philosophy - for instance each cell keeps track of queue statistics, process capabilities and configurations and tools that are available at any instant of real time, however the Factory Controller is able to access this information during the task and tooling procedure. With the Hybrid architecture jobs are defined once as a series of tasks at the Factory Controller level - a job may require four different operations (called tasks eg: CNC milling, CNC lathe work, CNC punching etc) before it can

be despatched to the customer. As jobs are defined they are queued at the Factory Controller according to their degree of urgency as defined by the due date.

- ***Factory Based Job Allocation***

The general model of the Hybrid bidding system used for the allocation of tasks is shown in Figure 2.2 and is a variation of the model presented by Shaw.[48]

When a job is new (no tasks have been carried out) and is ready to be scheduled for the processing of its first task the Factory Controller opens the job's file, extracts the relevant data from it and then disperses it to all active cells in the factory (*Task Announcement*). The cells calculate a bid that is dependent on a given performance criteria (ie: Earliest Finishing Time) and these bids are submitted to the Factory Controller for analysis. The Factory Controller is sent a bid flag from each Cell Controller and it waits until either all flags have been collected or until a system timeout is reached (this timeout occurs if a Cell Controller has become disabled and therefore allows the bidding procedure to continue). The Factory Controller algorithm then makes a decision on the most appropriate cell to carry out the task and awards the task accordingly. This *Bidding Algorithm* is based on the Contract Net Protocol[16,25] as researched by Shaw, Veeramani, Smith and Tilley *et.al* and Ramasesh and has been proven as being an efficient means of task scheduling in the simulation work carried out as part of their research.

- ***Cell Controller Based Job Allocation***

When a Cell Controller is given the opportunity to hold an auction it inspects each processes out going queue. If a job name is present in any manufacturing process's out going queue then there are obviously more tasks to be done before that job is finished. The Cell Controller then sends the job information to the Factory Controller and the remainder of the bidding algorithm is completed as shown in Figure 2.2.

There are a few variations on the bidding theme:

- If there are processes available in the system to carry out the task being auctioned but their in queue buffers are full the jobs are put on a Process Busy queue in the Factory Controller. These jobs are then given priority next time the Factory Controller is given the opportunity to hold an auction. An attempt is made to auction all jobs on the Process Busy queue before auctioning a job off the standard queue.

- If a job is particularly urgent priority is always given to it over less urgent jobs. This may occur when a job needs to be expedited.

- When the Factory Controller is given the opportunity to hold an auction it attempts to auction all jobs off the urgent queue and then a single job off the top of the out going queue before surrendering auction control to the next node. This surrendering of auction is carried out by the passing of a Bidding Token.

- When the Cell Controller is given the opportunity to hold an auction it attempts to auction a single job off each out going queue of the processes under its jurisdiction before releasing the Bidding Token to the next available node.

- Job Rebid - this particular feature makes the overall system far more robust than many conventional systems. When a given machine in a cell becomes disabled a flag is able to be sent to the Cell Controller software advising it of the inactive status of the manufacturing process. As soon as this occurs the processes viability is questionable and as such it is no longer qualified to participate in task and/or resource auctioning activities, however once the device is deemed to be back in an active state it can once again participate. If the process is foreseen to be disabled for any length of time the jobs are redistributed to other cells using the Job Rebid feature. This feature monitors the state of its process and requeues any job to be done on the process's out going queue and marks the allocated tools as being eligible for redistribution.

- ***Node Polling Model***

In order to ensure that equal and equitable access is given to each node in the system for the purposes of auctioning an access model has been established based on the passing of a token. The theology behind this is that there is only one Virtual Token in the system and as cells are unable to hold an auction without it this acts as a polling mechnism. The basic Node Polling Model is shown in Figure 2.2 - it is a general model in that slight variation exists between the Cell Controller and Factory Controller handling procedure. A Virtual Token exists that circulates about the network and polls each node effectively giving equal opportunity for auction to occur at any node in the system. When a Cell Controller node receives the Virtual Token it inspects each process's out going queue to see if there are any jobs needing further work - if there are any jobs to auction the Cell Controller node reserves the token and initiates the bidding algorithm (see Figure 2.3). Once the Cell Controller has attempted to auction a single job from each process's out going queue or if there are no jobs in any out going queue then the Cell Controller releases the Virtual Token so as other nodes maybe given the opportunity to initiate auctioning.

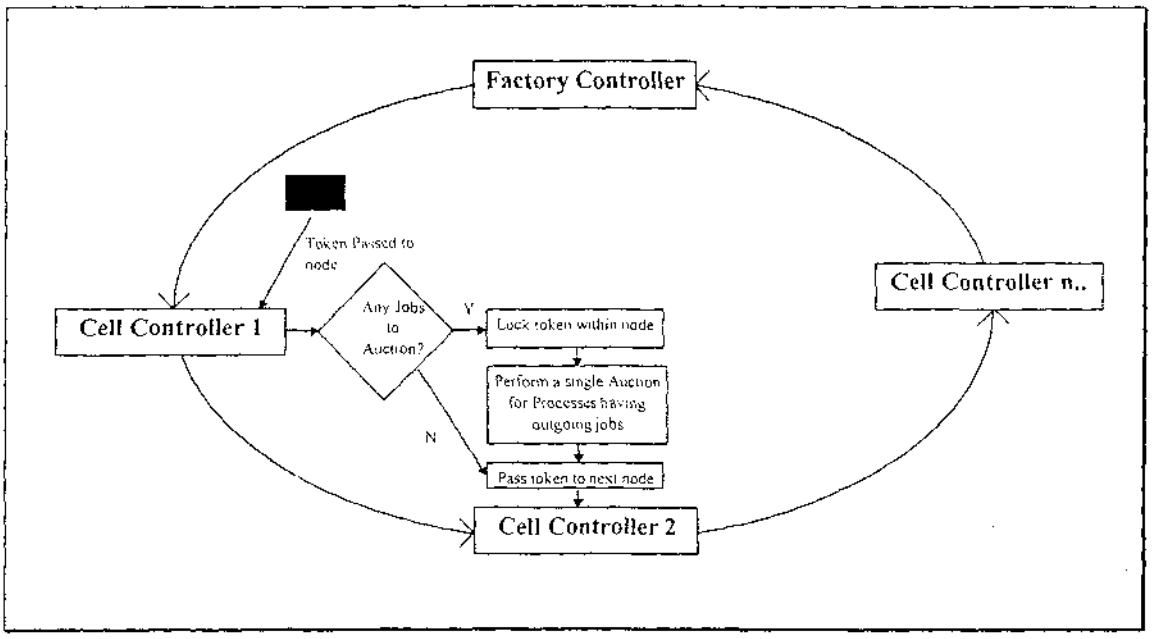


Figure 2.2 - Node Polling Model

One of the major problems that have faced companies wanting to implement FMS is the incompatibility of physical devices, hardware and software. As stated by Moore and Rosmus[7,15] MAP was been developed by the International Standards Organisation in an attempt to provide integration opportunities between different vendor software and hardware systems. A seven layer model has been developed which describes the total functionality of an FMS system - the protocols used are restricted at each layer and have been selected to suit the requirements of the manufacturing industry. What this means in reality is that if a company produces a MAP compatible device then it will able to be integrated with other MAP compatible devices because the protocols at each level of the seven layer model are consistent. This is all very well and good if a company has existing devices which are MAP compatible. However this is unlikely to be the case and there will still be serious integration problems that would need to be addressed.

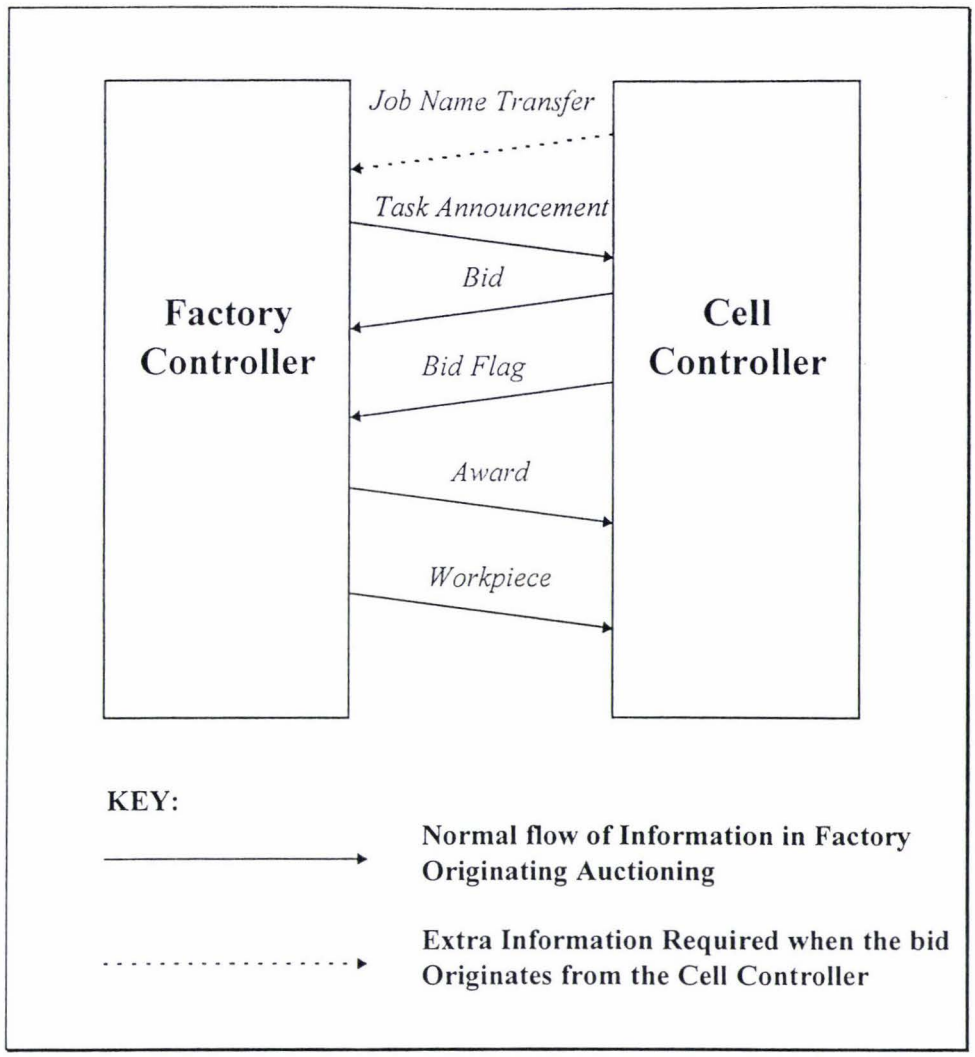


Figure 2.3 - Basic Model of the Bidding Algorithm

2.5 The Hybrid Model - An intimate look

The problem with the MAP approach to integration is that many companies cannot justify economically the expenditure required to research and develop new MAP based products. Consequently there are very few MAP compatible systems available, this has led to a very slow adoption of the MAP paradigm to solve integration problems. As such the problem of integration looks like persisting for some time yet.

The Hybrid architectural design focuses on the common aspects of hardware and software systems and uses these aspects to form the core of the *Hybrid Integrating Infrastructure*. The *Hybrid Integrating Infrastructure* is the core integrating aspect that allows a wide range of hardware and software systems to be integrated by loosely coupling these systems together.

With the MAP protocol the system sets the standard features that must be followed by component producers so that hardware and software systems are compliant with it. The Hybrid Integrating Infrastructure does the opposite - it takes features of Hardware and Software systems that are *currently* available and utilises them for the integration. The advantage of this approach is that it allows companies that want to have an integrated FMS system access to it now, rather than having to wait for other companies to produce MAP compliant equipment.

MMS is the MAP presentation layer protocol - it implements a generic mechanism for the presentation of data between components of the system by using the PDU (Protocol Data Unit). The PDU allows generic 'packets' of information to be presented to MAP compliant software and hardware options - the problem with this is that these options have to be in existence.

The Hybrid Architecture uses its own system called *UMS* (Universal Messaging System) which is configured to meet the standard feature aspects of current hardware and software systems.

Effectively then a non-MAP model is proposed that implements some of the paradigms existing in both the ISO seven layer model empire and parallels some of the philosophies that have existed in the CIM-OSA model [4,8-9] as well as the work carried out by Mo *et.al.* [11-14]

The Hybrid model is presented in Figure 2.4. Observing the model it is obvious that there are a number of high level components - these are explained below:

- **Communication Services Layer** - Provides the communication protocol and physical medium and connection for internodal communication and consequently implements the integrating tools of the integrating infrastructure.

- **Hybrid Integrating Infrastructure Layer** - Provides the integrating mechnism required between the Specified Functional Entities required to be integrated. The Hybrid Integrating Infrastructure contains as an integral part a *Virtual Manufacturing Device (VMD)*. Mo and Tang[11] describe the Virtual Manufacturing Device as "an object that maps the logical organisation of the system to a real manufacturing device such as a robot, Programmable Logic Controller (PLC) or Computer Numerically Controlled machine". The paradigm of the VMD therefore is a pure philosophy and does not refer to any specific software system available. The idea behind it is that the real world can be modelled by the computer system in a generic sense and therefore any real world entity can relate to it and be controlled by it.

- **Specified Functional Entity Layer** - In the context of the Hybrid Architecture a Specified Functional Entity (SFE) is a hardware interface between the VMD and the real world. It is a way of

transposing the virtual action of the VMD to components of the real world and controlling them, conversely it is a means of interrogating real world feedback and converting them into a form that the VMD can comprehend so that it can respond to what is happening in the real world.

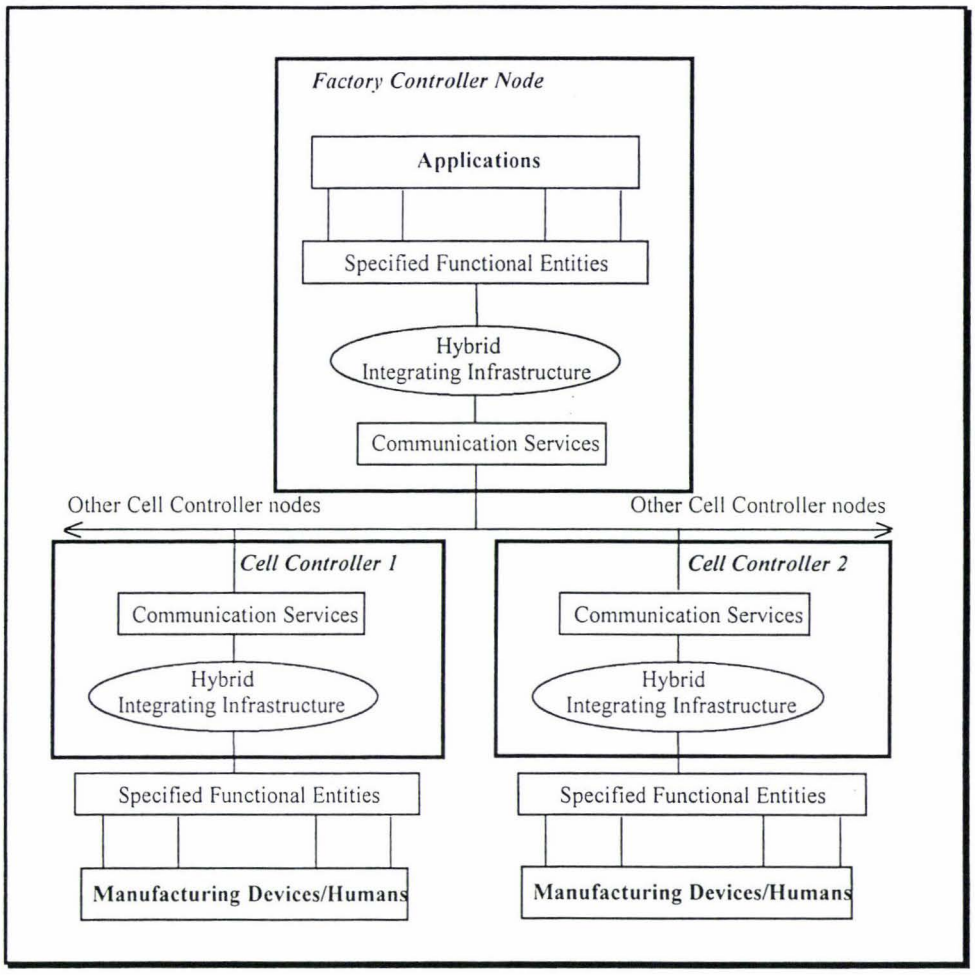


Figure 2.4 - A Definitive look at the Hybrid Model

In the development of the model implementation the software is designed to be completely modular, the only software that is unique is in the Factory Controller. In this way the system is easily and quickly extendable, the only customisation that is required is to each individual VMD in each cell to suit the behaviour of the equipment being controlled via the *Program Invocation Loop*. This customisation involves the selection and arrangement of various permanent and creatable objects to emulate the action and behaviour of the real world device. Mo[11-14] defines these objects as *Capabilities* and describes how they can be grouped as *Domains* in order to create a VMD that can be mapped onto real devices - in his research robots are the real world devices. In the Hybrid architecture we use similar ideas to map the VMD behaviour to the real world where the devices being mapped on are state and logic controllers such as PSC's and PLC's.

A derivation of the model to its lowest level is shown in Figure 2.6.

As can be seen from the model the Integrating Infrastructure is made up of a number of services that communicate over the Network using the Universal Messaging System. The following sections examine the roles of each of these services have and how they work in a distributed sense to meet the objectives of the system.

2.5.1 The Hybrid Architecture - General Services

2.5.1.1 Universal Messaging System (UMS)

The Universal Messaging System (UMS) is a system that enables computer nodes and processes to communicate using a common protocol. The real world system in reality has the only constraint being that the computer system must be IBM compatible. This is not a limitation in reality because part of the design brief was that the system was to be developed on a PC based platform. The UMS system is completely modular and is, in a sense 'object based'. The model of the communication protocol is shown in Figure 2.5:

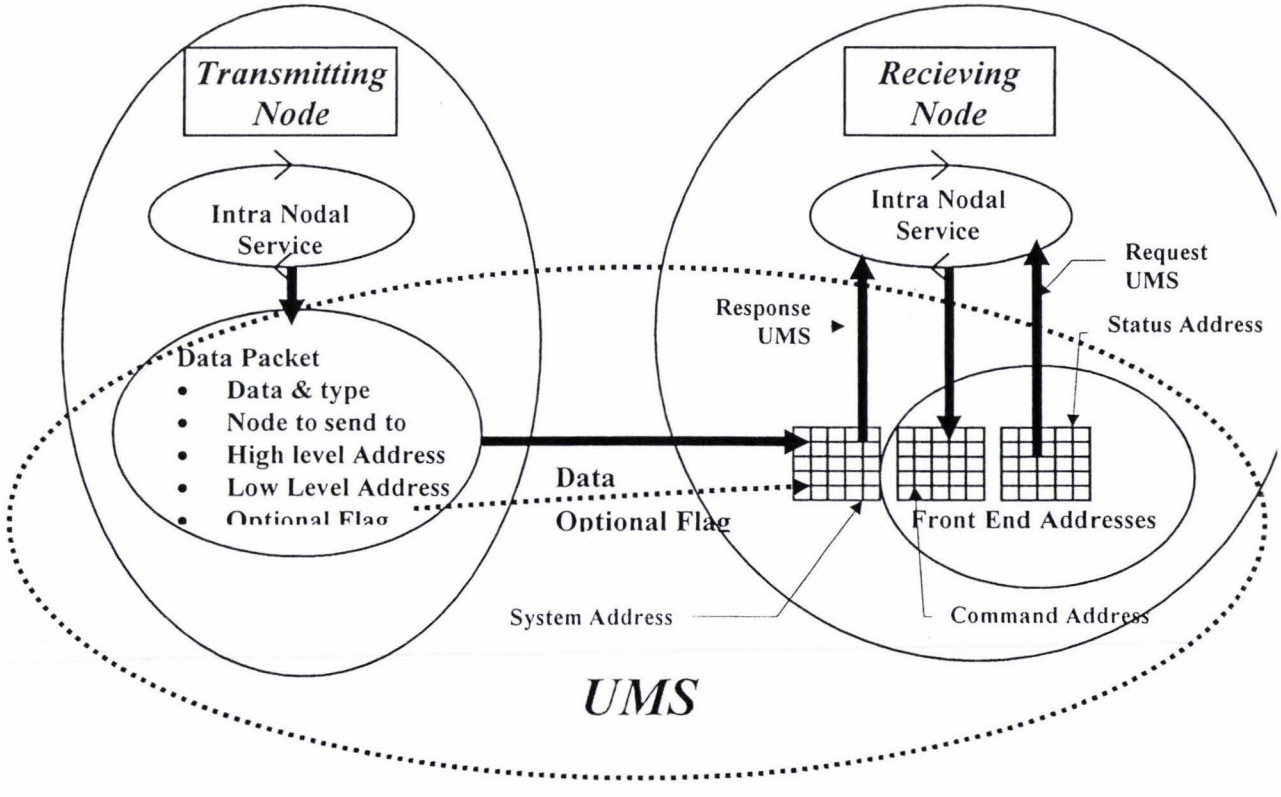


Figure 2.5 - Model of the UMS Communication Protocol

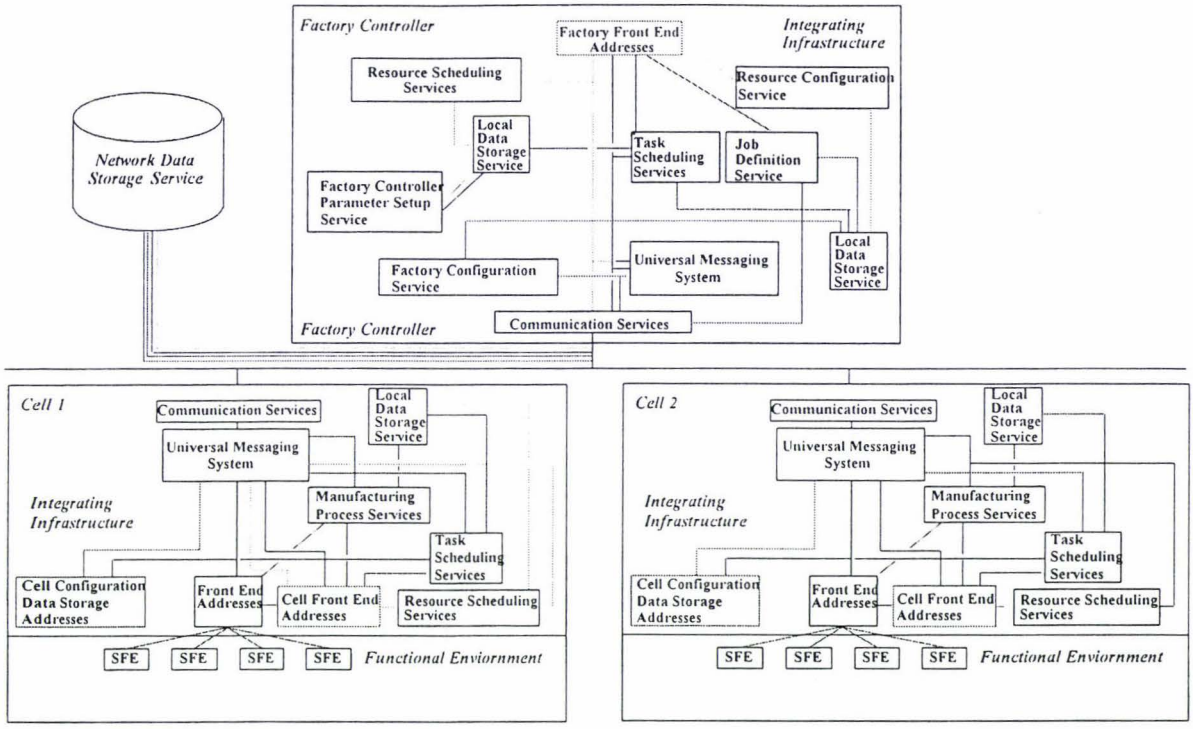


Figure 2.6 Lowest Level Representation of the Hybrid FMS Model

Generic Addresses are the foundation of the UMS system, they act as data stores and flag registers for the overall system. Every Cell Controller node has an identical modular set of these addresses and therefore as the overall function of the system revolves around the behaviour of these addresses and their relationship with internal and external services is constant, the FMS system is highly flexible, modular and therefore extendable. The system is also very predictable and traceable in terms of behaviour which makes the debugging aspect simple.

Not all generic addresses are shown in Figure 2.5. However those required to explain the messaging system have been included.

The UMS messaging system communicates in five main ways:

- ***Global Message Transmit UMS*** - When a particular node wants to transmit a message to another node it breaks the data packet into the classification shown in Figure 2.5 and sends it to the receiving node's system address. The system address is only used for receiving messages from other nodes.
- ***Global Message Request UMS*** - When a particular node wants to request a service or investigate the status of an address in another node a GMR UMS is used. The process is similar to the GMT UMS, however instead of writing to an address in another node it reads it.

- *Message Response UMS* - The UMS system can be used to set up a link between an address and a service local to the node so that the service is able to observe a flag and respond accordingly when the flag is switched. This means that if a request is made by another node for a service a flag can be sent and the service in the requested node is able to respond.

- *Message Request UMS* - a service may request the status of an address so that it is able to respond to it. eg: a status address many monitor the on/off status of a machine and a particular service may request this status before it can proceed further - it does this through a Message Request UMS. Such a message may also be used whereby data is required to be extracted from a Cell Configuration Data Storage Address for instance.

- *Local Message Transmit UMS* - This form of message is used by a service to change the state of a command address by commanding it to change. As the address system is used as the interface to the real world the command addresses are used to change the state of a real world device.

All addresses are completely and totally modular, as are the services that support the system. This means that the system is easily extendable, modifiable and adaptable. The services, addresses and UMS combine to model or emulate the behaviour of a job shop system.

2.5.1.2 Allocation of Tasks - The Task Scheduling Service

The basic idea behind the bidding algorithm has already been presented, however a more in depth analysis is required to get a clearer idea of how the system works in reality.

There are variations of the bidding algorithm that are employed in the overall system - Factory Based Task Allocation and Cell Based Task Allocation.

Factory Based Allocation - the model for Factory Based Task Allocation is shown in Figure 2.7. Some minor aspects have been omitted for clarity but will be discussed in the following paragraph.

The job is defined as a series of tasks in the Job Definition Service. Information that is generated for each task includes information such as task set up time, expected cycle time, required start and finish data and times, tool information, jig and fixture information and toolpath file. The toolpath file is created in a generic sense and once the winning Cell\Manufacturing Process has been decided upon its controller details are accessed. Next a generic post processing executable file is activated to process the toolpath file according to the type of proprietary controller the winning process uses. Once the job has been defined it

can be immediately queued on the Job's Pending File. However, this is optional as the user may wish to define a series of jobs before queueing them.

Once the job is placed on the Jobs Pending Queue, it, along with other jobs are scanned at regular intervals and have their start dates and times compared to the computer clock system. If the scan reveals that any job is due to be started it is placed on the Factory Job Queue for scheduling. If the job is extremely urgent the urgent flag is turned on at the definition stage and the job is placed on the top of the queue, otherwise it is placed on the bottom of the queue.

As defined earlier the Node Polling Application is responsible for giving non-biased access to each node for the purposes of task allocation. When the Factory Node receives the token and has jobs to allocate to the system it locks the token thereby forbidding access to it by any other node. Once the Factory Controller locks the token it checks two of its queues (only one shown in model) - the Factory Job Queue and the Busy Queue. The Busy Queue contains names of jobs that have tried to be auctioned by the Factory Controller in the past but have been temporarily sidelined because all capable processes in the system have full queue buffers. After the token is received all jobs on the busy queue are attempted to be auctioned before release of the token, however if there are no jobs on this queue an attempt is made to auction a single job off the Factory Queue. The reason for having two queues is that if all buffers in a relevant process were full the job would remain on the top of the Factory Queue therefore blocking access to other jobs on the Factory Queue at auction time. Also the two queue system gives more urgent jobs precedence by placing them on the busy queue.

When a job is being auctioned by the Factory Controller its file is opened and relevant information is extracted and written to a network bid file so that the task details can be accessed by all potential cells.

A Global Message Transmit UMS is sent to each node in the system requesting its Task Scheduling Service be activated. Each *active* node responds and the Task Scheduling Service retrieves the bidding objects from the network file. The bidding data is then compared with Cell Configuration information stored in the Cell Configuration Data Storage Address and the Network Data Storage System. There are three types of bids lodged during this process - Null bids are lodged for processes that are not capable of processing the task, Null bids with BusyFlags set to 1 are lodged for processes capable of carrying out the task but having full buffers and True bids with BusyFlags set to 0 are lodged for processes capable of doing the task and having buffer space. This allows the Factory Controller to distinguish between each event. Once a Cell Controller has finished compiling all the bids for its processes it sends a Global Transmit UMS to the Factory Controller Front End Address to inform the Factory Controller that it has finished formulating bids.

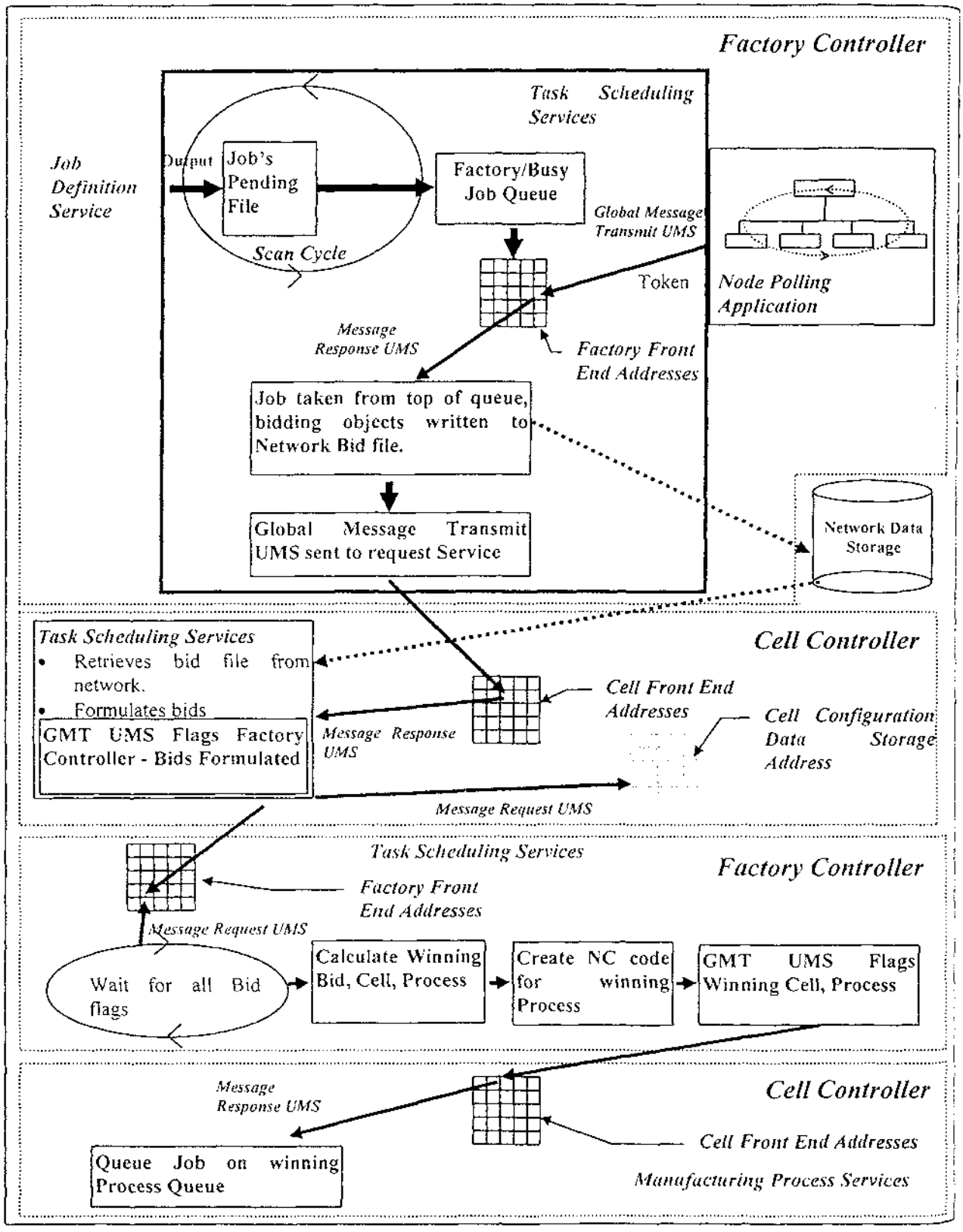


Figure 2.7 - Model of the Factory Based Task Allocation

After the Factory Controller sends the bidding information to all cells and flags them for a response it waits until it has received all bids from all Cell Controllers in the factory. Some of the other systems developed in the past by other researchers [25,37,39,40,48,55] have a timeout system whereby the node that holds the auction begins assessing bids at a set time and throws out any other bids received after that time. The problem with this system is that the optimum solution may not be found. In the system developed as part of this research the delay between Task Announcement (the event of the Factory Controller prompting a response to the bidding objects on the network) and assessment of bids varies as the Factory Controller only starts to compare bids when all have been received - this makes the system adaptable to fluctuations in network loading. It does this by checking that a series of flags in its front end address have all been switched on, it does this intermittently until all flags have been switched on from each Cell Controllers and only then does it formulate a bid - unless a timeout is reached signifying that a cell has crashed or become disabled in someway. If this occurs the flags are reset by the Factory Controller and the comparison of bids is made. If no winning process is found then there are two possible reasons - there is a process available but it is has a full queue, therefore the job is queued on the bottom of the busy queue if not urgent, otherwise it is put on the top of the queue if it is urgent. The other reason is that the user mistakenly mis-defined the job configuration and the job is then put on a rejected job's queue - this may also happen if there was a process capable of carrying out the task when the job was defined but was disabled before auction time (or decommissioned).

Once the winning cell/process has been found the proprietary controller details for the winning process are accessed and the toolpath file is processed by the integrated CAD/CAM software executable file to produce the correct format for the controller being used. The Factory Controller then flags the winning Cell/Process and the winning cell/process queues the job on the top or bottom of its queue depending on its urgency status.

Due to the fact that the behaviour of the system is consistent in every cell, modular components of software can be formulated and used in the same fashion in each cell.

Cell Controller Based Allocation - the Cell Controller based allocation of jobs is very similar to Factory based allocation of jobs, however there are slight variations on the theme:

- When the Cell Controller holds an auction it attempts to auction a job's current task off each active process's out going queue before releasing the token.
- When the Cell Controller auctions a job it informs the Factory Controller of the job it wants to auction and the task and then the algorithm as in Figure 2.7 is actuated.
- If there are processes in the Factory capable of doing the task but they are busy, then if the task is urgent it is placed at the bottom of the busy queue in the Factory Controller, otherwise it is placed back at the bottom of the out going queue of the process it is being

auctioned from. The logic behind this is that if it is on the busy queue and is urgent it will be auctioned more quickly than if it is on the bottom of the auctioning cell/process out going queue. It is put on the bottom if not urgent to prevent blocking of the out going queue.

Rebid Algorithm - this algorithm is, strictly speaking not an algorithm but a link to the current Task Allocation system. This algorithm is activated in the event that a particular machine is shutdown or fails whilst carrying out a particular task. If the event occurs whereby a manufacturing process fails and the length of time it is likely to be disabled is deemed unacceptable by the Cell Controller the Rebid Algorithm is typically activated. Once activated this algorithm takes all jobs on the processes in going queue and places them in the same order on its out going queue where they can be auctioned. During the time that the process is disabled it is unable to compete for any tasks being auctioned in the system and is only eligible once it is in an operable condition. The Rebid Algorithm therefore is a mechanism for the system to react to an unexpected event in a positive and proactive fashion.

2.5.1.3 Communication Services - Requirements

The Communication Services provide the mechanism for the placement of data in predetermined data locations in other nodes as instructed by the logical implementation of the UMS system. The Communication Services are the backbone for the operation of the UMS system and must be able to provide the following:[8]

- Location Transparency - the Communication Services must be able to hide the spatial distribution of all communicating Specific Functional Entities in the environment.

- Failure Transparency - the Communication Services must be able to hide as far as possible the occurrence of failures in the communication system.

- Performance Transparency - the Communication Services must be able to hide, as much as possible, the effects of limited communication performance.

The main point is that without a robust, easily extendable communication backbone the CIM system itself is restricted in the level of performance it is able to achieve.

2.5.1.4 Resource Scheduling Service

Far and away one of the most important services within the Hybrid Environment is the scheduling of resources. Resources include items such as tools, jigs, AGV's - or generically speaking any equipment that

can be shared by a number of devices required to be used at any stage of the products processing life cycle. In the context of this thesis resources have been classified as tools, however the system could be extended to include other types of resources.

Once the tools have been defined in the Factory Controller's Resource Configuration Service the Resource Scheduling Service is used to schedule the tools depending on the requirements of the cells carrying out their task sequence.

The model of the resource scheduling service is shown in Figure 2.8. As in the Task Scheduling service the Resource Scheduling service is executed on a distributed decision making basis. Generally speaking the Resource Scheduling Service is actuated at a certain frequency as specified by the system administrator in the Factory Controller Parameter Set up Service. Once actuated the system follow the pattern as stated below:

- The tool status table has one row scanned each actuation cycle in order to see if the indexed tool is IDLE (not reserved or being used) or BUSY - is currently being used or in a state of transportation to the cell it is to be used in.
- If the tool indexed to in a scan cycle is IDLE then it can be auctioned - a bid file is created and information such as tool type, tool size, machine type and current tool location is printed to it on the network.
- A Global Transmit UMS is sent to the Cell Front End services requesting that the Cell formulates a bid for the tool.
- The Resource Scheduling Services in each cell reacts to the UMS via a Message response UMS and carries out the following activities:
    - ⇒ Checks to see what processes within the cell are active and therefore which of them qualify to compete for the resource.
    - ⇒ Checks to see what processes have already reserved a tool for the task at the top of their queue (the process may already be working on a task at this stage) - if they have already reserved a tool then they do not qualify as reserving too many tools would cause undue idle time throughout the factory.
    - ⇒ For any of the processes that have not reserved a tool and that have a task at the top of their in going queue the service compares the tool required details of the task with the details of the tool being auctioned - if there is a match between the type of tool being auctioned and the tool required for the task being considered then the service formulates a bid. The bid that is formulated at the cell level for a particular process/task is reflective of its urgency - the bid is formulated by considering the number of hours between the beginning of the year and the due date of the job.

⇒ Once a bid is submitted for a tool the bid is written to the network file and a GMT-UMS message is sent to the Factory Controller informing it that the cell has completed the bid. After actuating the bidding process the Factory Controller intermittently checks to see if all bids have been received by checking its addresses to see if all flags are switched on. If they are the Factory Controller opens up the bid file for each cell - each bid file contains data such as the name of the cell bidding, the process number and the size of the bid. As each nodes bid is extracted from its file the internodal travel time between the current place of tool residence and the bidding node is added to it to reflect the influence that current tool locality plays in the equation. This situation is most clearly illustrated in the instance whereby there are two tasks having equal degrees of urgency that are required to be carried out by two different cells and the tool already exists in one of the cells in which the tool is required. In this condition it makes no sense to give it to a cell that is, for instance, 30 minutes travel time away - it is best to start on the task straight away in the other cell that requires the tool and already has it. This internodal factor which in many cases will not effect which cell wins a tool will have a larger impact as the bids lodged *by each cell* become closer.

⇒ Once the internodal travel times are added to each bid the resulting numbers are compared and the lowest number is associated with the particular VMD/Cell that it originated from and the tool is awarded. When a tool is awarded there are a number of changes that are made in the Factory Controller tool table to reflect this. Firstly the tools status is marked 'busy', its location field is given the name of the winning cell and the toolpointer is incremented so the next tool can be auctioned the next time the table is scanned. Next the processes 'tool reserved' status is changed to reserved so that that process cannot bid for any more tools until it starts to process the task that has just won the tool - this stops the process from tying up tools for tasks further down the queue. Once this is done the Factory Controller sends a GMT UMS to the processes front end address telling the process that the tool has arrived and that it can start to process the task. In reality this would be done by an AGV once the tool had been delivered but due to the restrictions involved in this research this was emulated by the Factory Controller.

⇒ When the process receives a flag telling it that the tool has arrived it may be in a series of states - it may already be processing another task or it may be idle and waiting for a tool to arrive. If it is waiting the Program Invocation loop portion of the VMD would have paused at the *Check Tool Available capability* and this capability would be monitoring the tool availability front end address. As soon as the tool arrives at the cell the AGV would in reality switch this address on and the task would start to be processed. If a task was already being done the address would be switched on and the Program Invocation Loop would respond on its next cycle.

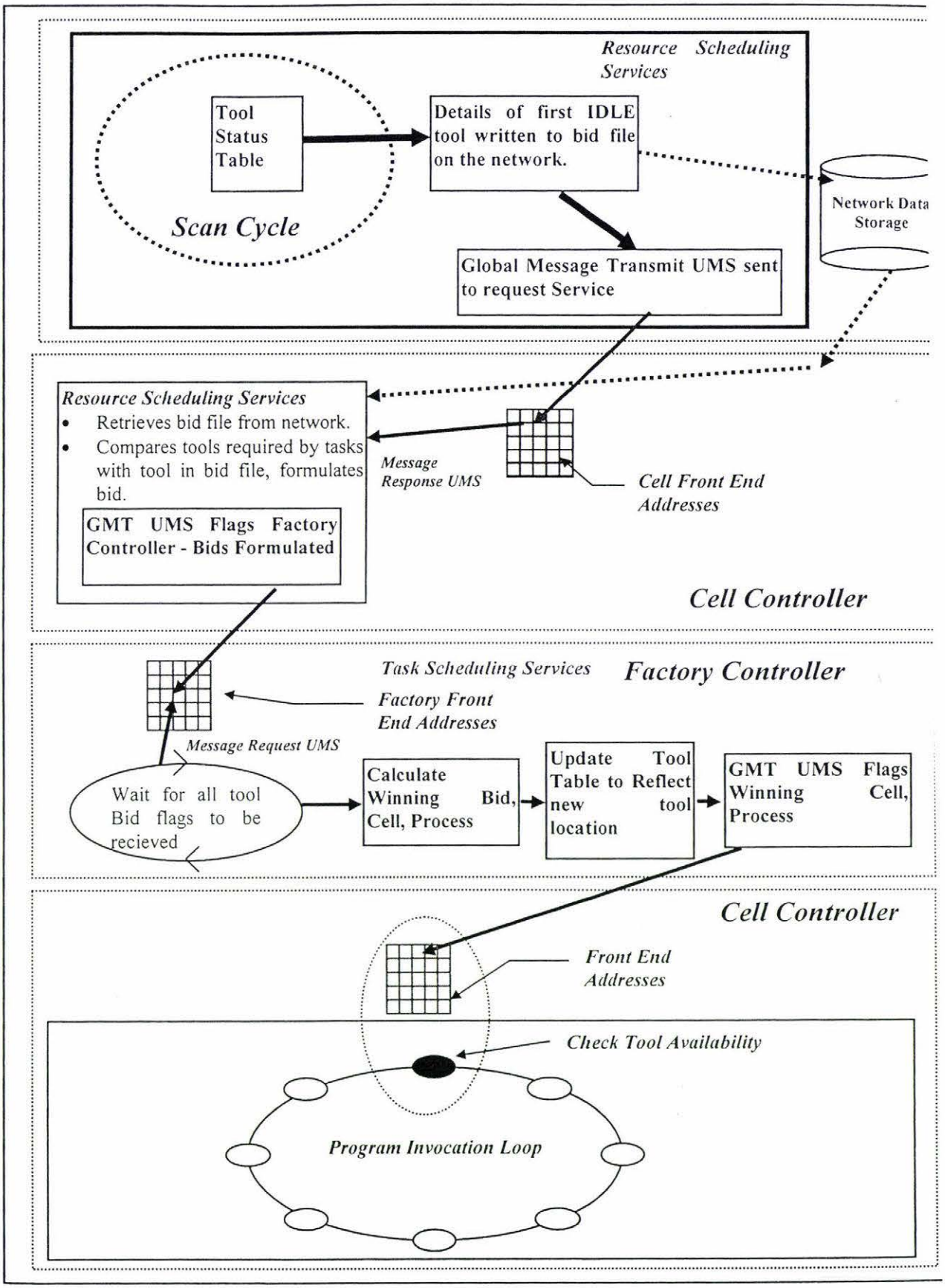


Figure 2.8 - Model of Resource Scheduling Service

As has been mentioned in earlier parts of this thesis the true distributed approach to allocation is of particular assistance in the handling of real dynamic situations where the future cannot be predicted. In the resource bidding situation this fact is used in that the auctioning approach is used when cells formulate the bids - the cells formulate bids for the resource dependent on the current status of their job queues. However the internodal times between cells are constant during run time and therefore this aspect is added on at the Factory Controller in a centralised fashion.

2.5.2 The Hybrid Architecture - Factory Controller based Services

There are a number of services that have their origins based in the Factory Controller - although the services have relations with the Cellular nodes these are for all intents and purposes weak therefore ensuring system robustness and extensibility. This section of the thesis will explain the functionality and purpose of those services and their relationships with other areas of the overall system.

2.5.2.1 Configuring a new Job Shop System - The Factory Configuration Service

The flow of the logic for this process is defined in Figure 2.6 by the red line.
The Factory Configuration Service in the factory node in conjunction with the Cell Controller Configuration Data Storage Addresses are used to carry this out.

When defining a new system the following steps are undertaken:

- The names of the Cell Controller nodes that are to be used in the system are defined in the Factory Configuration Service and stored in the Factory Controller Local Data Storage Service. This is important because the UMS system uses the Cell Controller names to address all forms of messaging.

- For each node the Manufacturing Processes under its control are defined. This is carried out in the following way:
  - ⇒ The number of manufacturing Devices per cell is limited to four.
  - ⇒ Each device within a Cell must be distinctly different.
  - ⇒ For each manufacturing device a code is defined as a 2 digit number. The first digit stands for the type of process, the second digit the capacity rating. This information is sent to the cell's Cell Configuration Data Storage Address using the UMS system. The information is also saved using the Factory Controller Local Data Storage facility so that it can be resent every time the system is reset.

⇒ More complicated information such as number and types of tools available, activity status of the process (it may be down for maintenance) and time buffer (the amount of task time allowed in a process's in going queue) is defined and then sent to the Network Data Storage System, each process in each cell having its own specific location on the network.

- The relative position and distances between nodes are also set up in the Factory Configuration Service. This information is critical when implementing the bidding algorithm for the scheduling of tasks and also in the algorithm used in the distribution of tools.

- The Factory Configuration Service also allows for the disabling of processes within cells so that they are unable to compete for tasks when down for maintenance or for other reasons. This occurs by the sending of a disabled flag to the appropriate Cell Configuration Data Storage Address. The reason this is done is so that a process does not tie up resources such as tools and can redistribute jobs using the bidding algorithm.

This system is again highly modular because common services and addresses can be transported easily from one cell to another meaning that new cells and processes can be quickly configured. Due to the lines of communication being consistent and predictable no debugging is required.

2.5.2.2 Customising the Factory Controller - The Factory Controller Parameter Service

During the construction of the system model a facility was made so that various critical parameters could be controlled in terms of the behaviour of the Factory Controller. Indeed it was found as the software was developed that control over these parameters become more and more important.

With this in mind the *Factory Controller Parameter Service* was created. This service allows for the setting and controlling of the following parameters:

- The time delay between asking the Cell Controllers in the system to respond to a Task Announcement and the Factory Controller comparing bids.
- The time interval between successive scans of the jobs pending file - when a job is defined at the Factory Controller it has a start date and time associated with it and can be made "pending action" by placement in the jobs pending file. Each time this file is scanned jobs in the file have their start dates and times compared with the current date and time and if the job is ready for action it is placed in the Factory Controller queue where it is handled by the bidding algorithm.

- Tool Auction Frequency - Tool auctioning occurs at the Factory Controller and the frequency (interval between successive attempts) can be set depending on the number of tools and system size. It has been found that being able to manually set this is important to ensure that tasks are not waiting too long for tools.
- Tool Auction Timeout - The delay between Tool announcement and Factory Controller bid assessment.

2.5.2.3 The Resource Configuration Service

The Resource Configuration Service is used for defining what types of resources are available to be used in the overall Hybrid FMS environment. Resources will in reality include such things as machinery, tools and fixtures as well as a host of other physical aspects. Generically speaking they can be termed as 'anything having a physical input into the completion of an overall job'. As such Resource Configuration includes the following aspects:

1. **Process Configuration** - generally speaking this involves the definition in a software sense of the characteristics of the manufacturing processes that are relevant to the operation of the system. What makes something relevant would be an aspect of the process that would minimise its ability to carry out a given level of functionality eg: The fact that a lathe has a 12" swing is far more relevant than the fact that it is painted blue. Defining a cellular manufacturing system involves to an extent the definition and arrangement of physical processes in the overall system. The arrangement of the processes within each cell is the function of the *Factory Configuration Service*, the definition of the processes to be arranged is part of the *Resource Configuration Service*. The overall Process Configuration model is shown in Figure 2.9.

There are a number of stages involved in the definition of a process:

- *Definition of the Process Name* - this is very important as it will act as a reference for the processing of data in the system.
- *Identification Number* - once the process has been defined in a descriptive sense the system will automatically give it an identification number which is totally specific for that type of process. This ID number is transparent to the user but is used by the system for purposes such as comparing the processing requirements of a task to the configuration of a cell therefore determining whether a manufacturing device qualifies as a potential processor of a task.

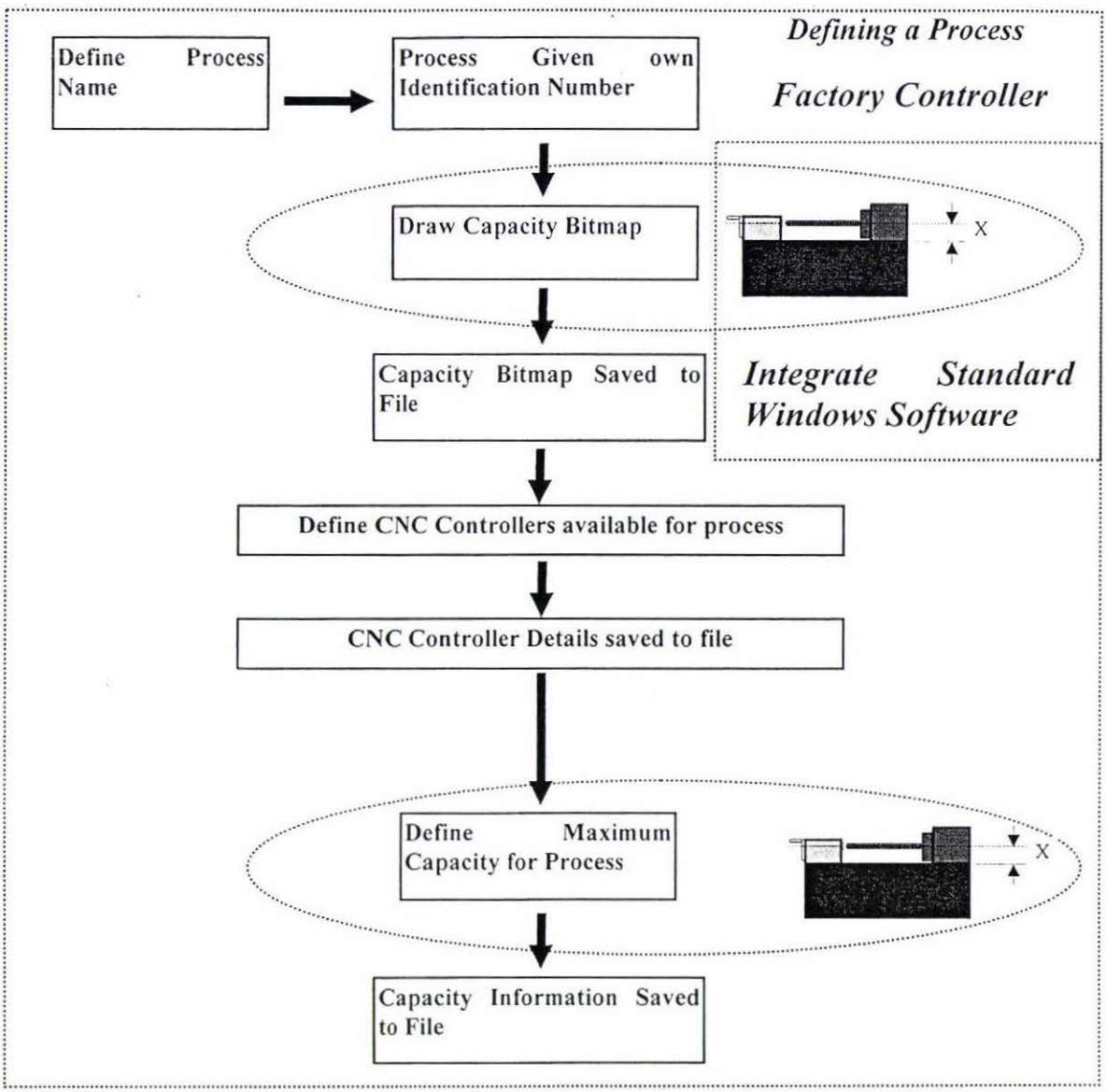


Figure 2.9 - Process Configuration Model

- *Draw Capacity Bitmap* - the capacity bitmap has the following features:

  ⇒ When a new process is defined a *Capacity Bitmap* is drawn for it as shown in Figure 2.9 - when we talk about a capacity in this sense we talk about a 'maximum limiting factor' . In the case of the lathe in Figure 2.9 it is dimension 'X' that is defined as the limiting factor, however this may not always be the case depending on what the user defines this as being. A lifting crane for instance carries out the task of lifting and therefore the limiting factor for this device maybe maximum lifting capacity.

⇒ When the user needs to define in the system what the magnitude of the maximum limiting factor is for all processes in his entire system he defines this as the stage of 'Defining the Maximum Capacity for the Process'. Taking the lathe as an example the user may have at his disposal 30 lathes for his FMS the largest of which has a 30" swing. At this stage of the process definition (see Figure 2.9) the user will therefore define the dimension X as having a maximum magnitude of 30 for manufacturing devices of this type in the physical Hybrid FMS system. Once the maximum dimension is defined the system automatically divides this into increments from zero to the maximum size.

⇒ When the user configures a process for a particular cell he must first define what type of process it is, what tools are available to it, what fixtures/jigs it can use and what size it is compared with the limiting factor defined in the overall process configuration. The user does this by selecting from the list of limiting factor increments the category in which the particular manufacturing device falls into - at task auction time this allows the cell to know whether the particular manufacturing process is eligible for the task by comparing the category code with that defined in the task description.

⇒ When a task is defined for a given job the user will select the manufacturing process required. After process selection the user then selects the minimum size category of the limiting factor that will carry out the task satisfactorily. The capacity bitmap is displayed so the user can understand what aspect of the manufacturing process the limiting factor category pertains to for the particular system set up. In this way he can select the minimum category to process the task.

- *Definition of CNC Controllers Available* - The user must define the types of CNC Controllers available to any individual manufacturing device that falls within the category of the type of process being configured. This allows the user to automatically select the CNC Controller information when configuring a Cell Controller during the factory configuration stage.
- *Definition of the Maximum Capacity for a process type* - the definition of the maximum capacity available for a process of a given type in the entire system is carried out at this stage. This definition process allows the system to be informed of the limitations of any manufacturing process in the system - it sets upper bounds in terms of the maximum limiting factor. In a sense it limits what tasks are able to be put through the system by imposing an upper limit in terms of the limiting factor of the process - ie: It does not allow a task to be defined that requires a 35" swing on a lathe when the maximum size of lathe in the system is only 30" - to do this would be ridiculous. It also allows the user to select from increments of the limiting factor when defining minimum process requirements for a given task and when defining process capacities in a given cell configuration.

In the definition of this model it was deemed important that existing Windows based software be used as this software is a Windows™ standard. Therefore it is envisaged that the main software for this service

will be integrated with Microsoft Paintbrush using Object Linking and Embedding techniques. This is far better than creating a new drawing system for the definition of graphics such as Capacity bitmap definition as it uses a standard Windows™ based package.

An important feature of this service is its flexibility - as processes are ultimately defined as customiseable drawings, flexible capacity definitions as a number to classify the process as a certain type and capacity as such an infinite number of different process types that can be configured for the system. In every aspect of this process an attempt has been made to achieve a total generic state in all aspects to allow for maximum flexibility.

The model shown is only a generalised viewpoint of the overall functionality of the system. Other important aspects that the process configuration entity has are that there is functionality reserved for the removal of processes, redefinition of CNC Controllers available and the redefinition and modification of existing Capacity bitmaps and maximum limiting factors. These extra levels of functionality provide further allowance for modifiability and reconfigureability of the overall system.

2. Tooling Configuration.

In much the same way as for the process configuration function a similar type of definition of tool aspects relevant to the system has to be defined. The generalised flow chart of the tooling configuration function is shown in Figure 2.10, the aspects of this model will be discussed below:

- *Process Configuration must be done first* - before any tools can be defined it is vital that all potential processes be defined so that tools can be stated for the processes defined, the tooling configuration system then loads all process details as required.
- *Process Selection* - once all processes are loaded the user selects a process for which tools are to be defined, once this is done a bitmap is automatically loaded showing the device. Details of existing tool types are loaded into the system and are displayed next to the illustration of the process.

- *Tool Type Definition* - at this stage the user enters the name of the new tool that is being defined and the tool is automatically given its own identification code. The tool definition is then added to any existing tools that have been defined in the form of a tool list for the process.

- *Tool Size Definition* - Tool types can be selected from the list and have sizes defined for them. When a tool is selected a bitmap is added to the system telling the user what aspect of the given tool the particular dimension required pertains to. In Figure 2.10 the situation is shown whereby the user is defining the sizes of parting tools for a lathe. A general description of the critical dimension is given

as the distance across the tip of the tool itself. This bitmap can be modified by the user using Object Linking and Embedding (OLE) to define other aspects of the tool as having the critical dimension. This facility provides a way of assuring that all tools of a particular type are defined in the same way and gives the system a high degree of flexibility. When new tool types are defined and are not part of the standard tool definition for the system no bitmap exists - the user can however create their own as required using OLE. The user then inputs the particular tool dimension as required.

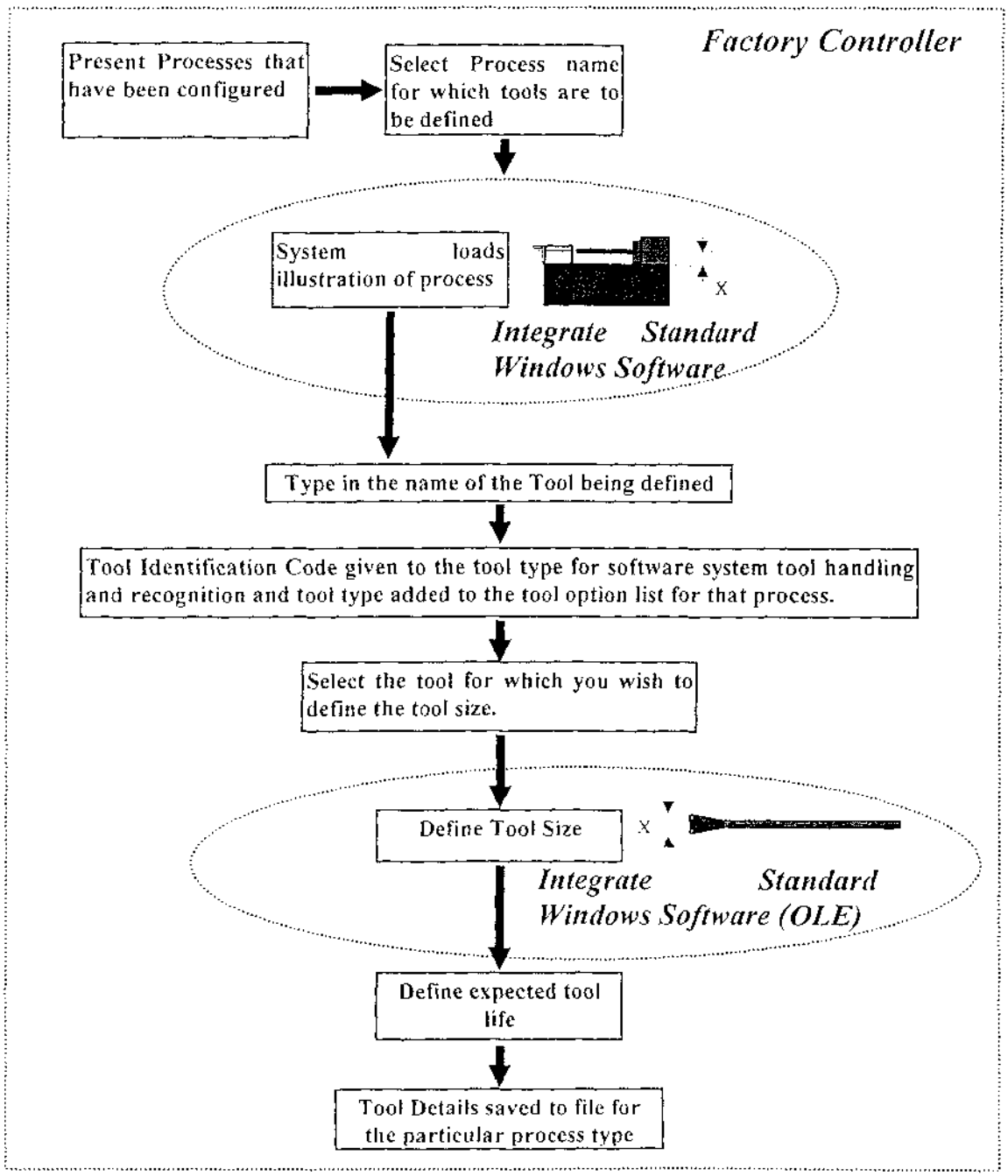


Figure 2.10 - Tool Configuration Model

- *Tool life Definition* - once the tool size has been defined the user inputs the expected tool life for the tool. This is an important aspect of the overall system as it allows the FMS to flag the Factory Controller when the tool has reached its expected life. The tool can then be examined and an updated expected life given or the tool can be replaced if beyond repair.

- *Saving Tool Data* - once all tool sizes are defined for a particular tool type this information is saved to a file for the particular process and tool type.

When this is carried out for all tools that are likely to be used in the FMS there exists a data store holding information regarding all tools in the system and the types of processes they are associated with. When the user defines the jobs to be carried out by the system he can select the process required to execute a particular task and from this select tools available to carry out the requirements of the task. This data store is constructed in a consistent manner as defined by the tooling configuration system and therefore its form does not change - only its size.

2.5.2.4 The Job Definition Service

The job definition service allows the user to define jobs in a generic sense so that they can be recognised by any process within the Hybrid FMS environment.

The generic model for task definition is shown in Figure 2.11.

Each job is broken up into a series of tasks - each task representing one physical operation on a single physical object. This type of approach was also taken by researchers such as Bakker[35] and Shaw *et.al*[25,37,39,40,48] - Bakker says that "In an FMS a series of operations has to be performed on each product" and that the operations are scheduled in the FMS system in a real time distributed fashion, Shaw takes a similar approach. This job defragmentation process occurs to the lowest possible level - a single physical operation on one object and provides the maximum level of system flexibility.

When defining a task there are a number of important aspects that have be defined as is shown in Figure 2.11. Some of these aspects are discussed below:

1. **Batch Size** - Selection of batch size may be dependent on factors such as number of products ordered and policy of the company ie: JIT strategy imposes certain restraints on batch size.

2. **Number of Tasks Required** - When assessing a new product the user will make a judgement as to what tasks are required and the best way of ordering those tasks so as to produce the required finished

product. A good example of this is the shaft as shown in Figure 2.11 - the system user will know that he cannot case harden the material before turning and therefore the obvious task structure for this job is as shown.

3. **Define Job Start Date** - With a new job the job start date will be an estimate based on required due date less estimated processing time. Once the job has been processed the actual processing time for the job could be fed into a job database and when the job was reordered a better estimate for start date could be derived.

4. **Define Set up Time** - When a new job is configured the set up times for each individual task will be based on an estimate. The relevance of the set up time is that it becomes one of the factors used in the calculation of a cells bid for a given task, therefore the accuracy of the task scheduling system is only as good as this estimate. Therefore in reality the real set up time needs to fed back to the task definition as mentioned previously to generate a better estimate - this would typically occur after the task is set up at the particular device.

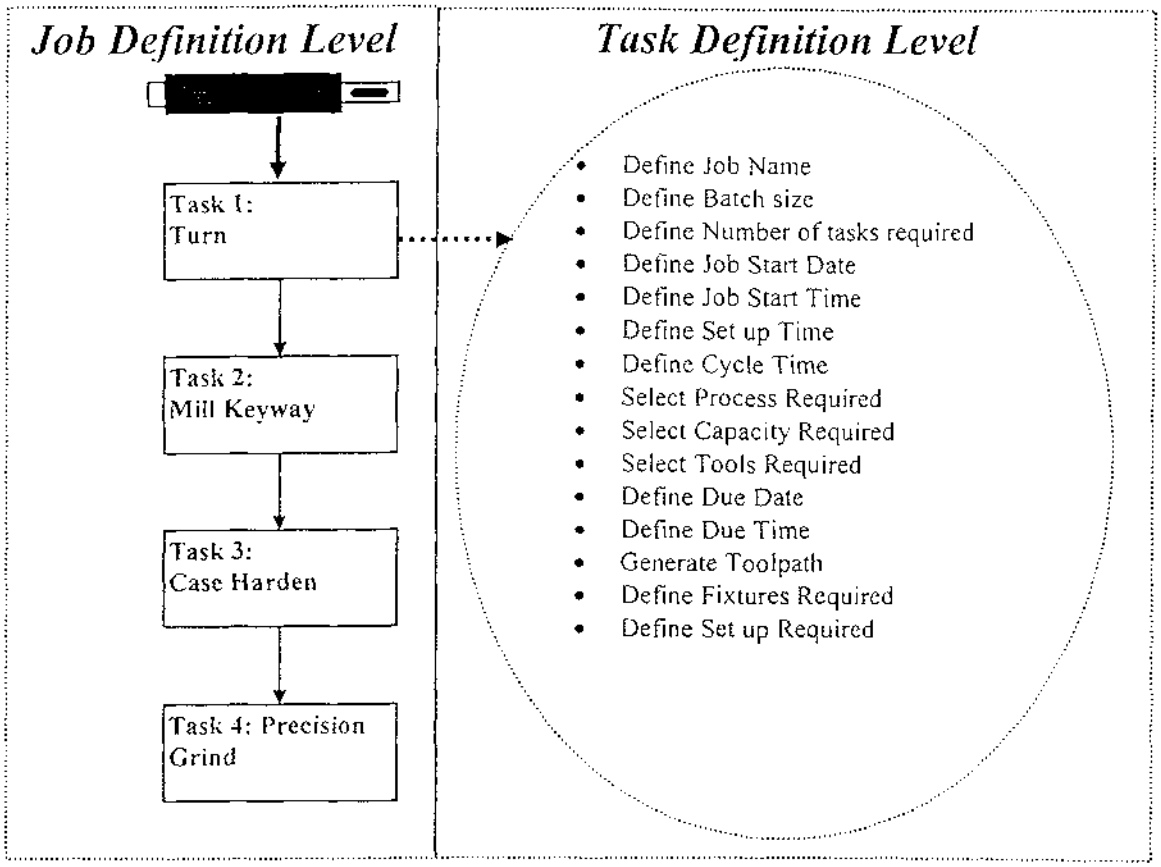


Figure 2.11 - Hybrid Job Definition Model

3. **Define Cycle Time** - This is the time required for one cycle to be completed on one task. As with set up time the original time entered is at best an estimate. However once a cycle for the task has been completed this estimate can be updated.

4. **Select Process Required** - When defining a particular task obviously the form of the manufacturing process required to perform that task is required to be selected. The name of this task is simply selected from a list of processes that have been configured for the system.

5. **Select Capacity Required** - Once the type of process required for the task has been selected a capacity bitmap is shown illustrating the limiting factor for the device (See earlier). The user then selects from a list of limiting factor classes the interval that the task falls into - this allows processes in the system to know the limiting factor requirement for the task. In this way they are able to calculate whether they are capable of performing the task at task auction time.
6. **Select Tools Required** - This definition is very important. Tools required to carry out the task need to be defined so that Cell Controllers within the system can see if manufacturing processes under their control are eligible to carry out the task. The output from this tool selection process is a two digit number identifying the types of tools required and their size.

7. **Define Due Date** - The due date definition is another important aspect in terms of the control logic of the system and needs definition.

8. **Generate Toolpath** - Generation of the toolpath for a task is important in that it must be formatted in a generic sense. The Hybrid architecture takes advantage of the windows environment to integrate a large number of windows based CAD/CAM packages. The model of the integration of the CAD/CAM packages will be presented in Chapter 3. This model will reveal how aspects of the windows environment are used to perform the integration. In short a wide variety of CAD/CAM packages are able to be accessed to generate the required toolpath which is performed by the user. The name of the toolpath file created is unique to the task and this name is stored as an attribute of the task description. When the task is scheduled to be run on a given process the process details are accessed and the toolpath file is processed via the CAD/CAM NC conversion executable file to conform to the type of proprietary controller used on the manufacturing process performing the task. When the task is about to be performed this NC file is downloaded to the appropriate controller over the network and activated via the Program Invocation Loop.

9. **Define Fixtures Required** - Although this part of the system has not be explicitly allowed for within the context of this thesis it would we be defined and handled in a similar way as other resources mentioned in the system. Anything that can be defined as a resource should be handled and scheduled on a consistent basis.

mentioned in the system. Anything that can be defined as a resource should be handled and scheduled on a consistent basis.

10. **Define Set up Required** - Definition of the set up required should be made by performing Object Linking and Embedding(OLE) with a generic windows package such as Microsoft Paintbrush at the Factory Controller Level. The set up bitmap created should show the position and set up of any jigs/fixtures required for a given task and should be displayed at the Cell Controller level during task set up so that the operator at the manufacturing process level can set up/position the jigs & fixtures as instructed by the illustration.

Each task is defined in the fashion described above until all are completely described thus finalising a full task definition for the job. The system is able to manipulate each set of task data in a predictable way to perform the required task. After each task is completed a virtual pointer is moved to the next set of task data to indicate that this is the next set of task data becomes the focus of attention.

2.5.3 The Hybrid Architecture - Cell Controller Services

2.5.3.1 The Specific Functional Entities (SFE)

The SFE is an entity that physically instructs a process or human being to carry out some physical task. The general SFE paradigm that was raised as part of the work on the CIM-OSA project[8-9] has been translated for use on the Hybrid model. The SFE is a hardware interface that is used to translate the generic software control code into a form that can be directly linked to and understood by any real world device/human. The SFE can be broken down into two sub categories in the Hybrid model.

- Machine Functional Entities (MFE) - Machine functional entities coordinate and control the state of the processes directly associated with the physical processing system by implementing some type of state or logic controller eg: PLCs , PSCs

- Human Functional Entities - Human functional entities coordinate and control the activities required to be done by human means by some form of visual display system. This system must be generic and the prompts required will be different for each Cell Controller as they will be dependent on the configuration of the Activity Control and Coordination function. Some form of visual display will be required for each manufacturing process in the cell so that a human operator is able to react to

requests by individual processes. An example of this is in the circumstance whereby three different processes in one cell require three different operator actions at one time.

2.5.3.2 Front End Addresses

The Front End Addresses have a number of functions that they are required to perform and consist of two main partitions - these being the Machine Front End Addresses and the Human Front End Addresses. The Machine Front End Addresses are responsible for presenting command data to the Machine Functional Entities that are being served in a generic fashion so that no matter what type of manufacturing process is connected to the MFE the data interpretation is the same - the Machine Functional Entity itself is connected to the real manufacturing process, a similar scenario exists for the Human Front End Addresses. The structure of the Front End Addresses and their interaction with the Specific Functional Entities is shown in Figure 2.12.

The structure of the Front End Addresses at the Cell control level is illustrated by the lower level model of the Machine Front End as shown in Figure 2.13.

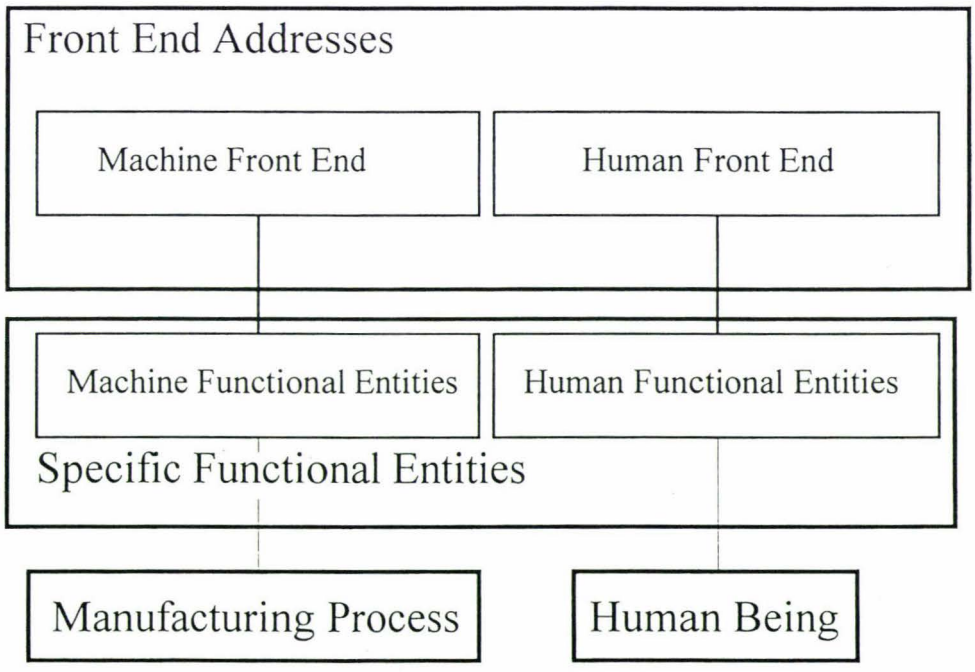


Figure 2.12 Interaction of the Front End Addresses with the SFE

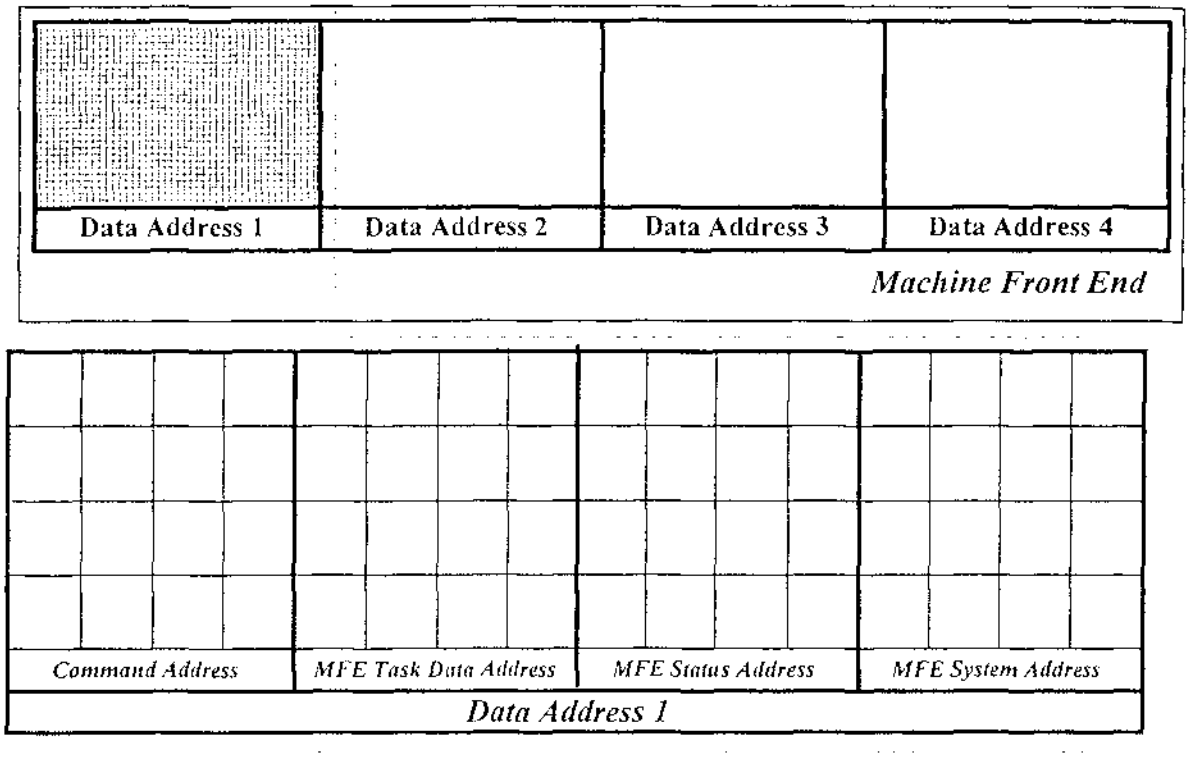


Figure 2.13 - MFE - Lowest Level of Representation

The machine front end addresses of any Cell Controller are broken up into a number of generic data addresses as shown. Each data address is responsible for the total service and support of any *one* Manufacturing Process via its MFE and is modular in nature. This means that as processes are added to a cell a data address module is required to be added to address each process being added.

A Data Address itself is broken up into a series of segments, each segment being responsible for a given type of service. These various address types are explained below:

- *Command Address* - the Command Address segment is responsible for providing a generic interface between the Activity Control and Coordination function and the Manufacturing Support services. When the Activity Control and Coordination module requires a Manufacturing Support Service the request is made by a UMS message passing via the command address.

- *MFE Task Data Address* - this MFE Task Data Address segment is responsible for the retention of data required for the executing of the current manufacturing activity occurring at the associated Manufacturing Process as well as general information pertaining to tasks to be done. The Activity Control and Coordination module uses this data to complete the task as required. The data includes information such as the details of the job being done, the name of the NC file required to be

downloaded, fixtures required and process configuration information. General task information stored in this address relates to information such as how many tasks (or jobs) there are in the queue.

- ***MFE Status Address*** - the MFE Status Address segment is responsible for generically acquiring the state of the Manufacturing Process via the MFE and for the retention of this dynamic data. The Activity Control and Coordination module then processes the acquired data via the Virtual Manufacturing Device (see Fig 2.15) and modifies the MFE Status Address in accordance with the Manufacturing Process's Program Invocation Loop (see Fig 2.15). This information is then transferred to the real world process to influence its behaviour. It is also used to store other *dynamic* information such as the number of tasks a process has in its queue.

- ***MFE System Address*** - The purpose of the MFE System Address is to provide a command and status interface between the Factory Controller and a *specific* SFE. This differs from the Cell Front End Addresses in that it is an interface between the Factory Controller and the Cell Controller's general services.

Examples of some of the functions that the MFE System Address provides include:

⇒ A mechanism by which the Factory Controller informs the SFE whether the resources are available to it to carry out the next task it is required to process, such resources including fixtures, materials and tools. The Program Invocation Loop can then check the status of the MFE system address to assess the availability of the resources before it deals with them.

⇒ A mechanism by which important MFE system information can be held. For example the Program Invocation Loop accesses information regarding how many jobs are in the queue for the manufacturing process associated with the MFE.

Interaction between the Front End Address and the Activity Control and Coordination module is consistent, however the configuration of each Activity Control and Coordination module is dependent on each type of *Manufacturing Process or Human Interface* the SFE is being connected to. The role the Front End Address system plays is important in that when any new SFE is required to be introduced into the system it not only provides a generic interface to the manufacturing process (and therefore presents the data from that process to each capability in the Program Invocation Loop in a generic fashion), it also provides a generic interface for the messages that are expected to be received from the Factory Controller and a generic messaging interface to the Manufacturing Support Module. Once the physical connection of the MFE to the Front End Service is established a new Manufacturing Support Service module can be added to the Manufacturing Process Services as well as a new Activity Control and Coordination module.

It is then simply a matter of configuring the Activity Control and Coordination module to suit the type of *Manufacturing Process or Human Interface* the SFE is being connected to. This is achieved through the selection of Control Routines from the Capability Library and the creation of *Program Invocation* loops using these routines within the desired Activity Control and Coordination module.

This type of approach is similar to that undertaken by Mo *et.al* in his research[11-14]. With his work, however, MMS was used to group variables and capabilities into domains using the Client-Server approach to form the *Virtual Manufacturing Device*. This grouping was carried out by the client node remotely within the server node[14] - the client node groups Capabilities and Variables to form Domains. The Capabilities are short generic portions of program designed to carry out one specific action. For example: go-to-position (used to move a robot arm to a particular position). Each capability would have a variable/s with which it was grouped which relate to a specific input or output required for the capability. Therefore, capabilities and variables could be remotely grouped together by the MMS Client within the Server to form individual Domains. The Client then ordered these Domains in the required sequence (forming the Program Invocation) to model the task required as a *Virtual Manufacturing Device (VMD)* - this generic model could then be connected to the real device in order to control it in the execution of the task. Once the Program Invocation sequence was formed the Client used an MMS service to initiate the Domain sequence in the Server.

In the Hybrid model the approach is similar in that the Variables are the MFE Task Data Addresses and the MFE Status Addresses. The capabilities in the hybrid model relate to routines within the Manufacturing Process Services and the Program Invocation Loop is defined in a *static* sense (the loop is defined once *for the process*) depending on the nature of the manufacturing process that the system is being connected to. Where Mo *et.al*[11-14] use Program Invocation Loops to account for the difference amongst certain tasks, the Hybrid model uses a similar approach to define the difference between processes and relies on the processing of the toolpath file at scheduling by the relevant post processor file at the Factory Controller. It is felt that this type of approach is superior in that it minimises the amount of traffic over the network and is more practical to implement in the small job shop.

2.5.3.3 Processing Tasks - The Manufacturing Process Services

The Manufacturing Process Services are responsible for the organisation of tasks to be carried out, the queueing of tasks to be done as well as a host of other services provided explicitly to deal with the processing of individual tasks.

It is the interaction between the Manufacturing Process Services and the Front End Addresses that provide the mechanism for the processing of a task.

An portion of the Hybrid Model is shown in Figure 2.14 to reiterate this relationship.

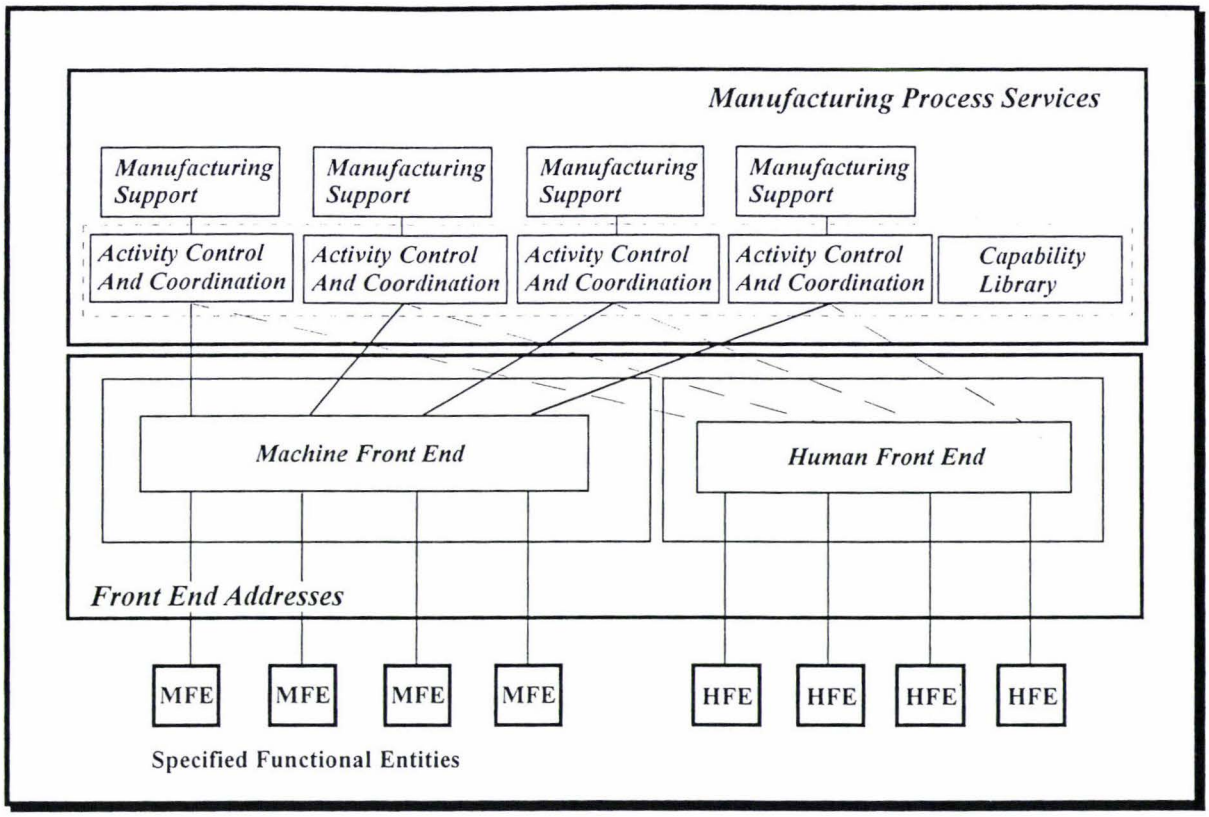


Figure 2.14 - High Level Relationship of the Manufacturing Process Services with the Front End

Within the Hybrid infrastructure the Manufacturing Process Services are the centre of the operation and have the following functions in the context of the Machine Front End:

- To check for a series of process states before starting the processing of any task at any manufacturing process being controlled by the MFE. Some of these states include:

  ⇒ Is the process concerned currently active and can process the task required?

  ⇒ Has the process concerned any tasks required to be done?

  ⇒ Have the resources that are required to carry out the task actually arrived? These resources can include items such as materials, tools and fixtures.

- The monitoring and control of the state of the Manufacturing Process - The monitoring function is carried out by the acquisition of state data from the real manufacturing process, and the transition of the data to the respective MFE addresses. The control function is then carried out by the analysis and modification of the acquired data at each address using the corresponding Activity Control and

Coordination Module. After the control function is complete the data is transmitted back to the corresponding MFE to influence the VMD which in turn directs the behaviour of the real manufacturing device.

- The downloading of task data to the Front End Services - When the Activity Control and Coordination module recognises that the state of its corresponding MFE is such that it is able to start a new task it downloads the generic task data to its corresponding *data address* in the relevant Front End Address. This event typically occurs at the start of the corresponding Activity Control and Coordination *Program Invocation Loop*.

- The loading of NC files onto the associated *Manufacturing Processes* - When the Activity Control and Coordination module recognises that the state of its corresponding MFE is such that it requires an NC file to be downloaded to the *Manufacturing Process* it is connected to, it responds by acquiring data regarding the parameters of the file required to be downloaded from its corresponding *Data Address* in the Machine Front End Service and passes this data to the Manufacturing Support module which gets the required file from the Network Data Storage and downloads it to the corresponding MFE's *Manufacturing Process*.

- The Queueing of *Manufacturing Activities* required to be carried out by each MFE's *Manufacturing Process* - This is one of the services offered by the Manufacturing Support Module. Upon the reception of a task the name of the task is stored in the target MFE's Manufacturing Activity queue in the Manufacturing Support Module, each MFE in the each Cell Controller has a copy of this modular queueing function in the Manufacturing Support module. Each Activity Control and Coordination module dynamically analyses the state of its corresponding MFE Service Data Address and when the state of the address signifies that a new manufacturing activity is required to be carried out it sends a message to the corresponding MFE's Manufacturing Activity queue telling it to send the manufacturing activity data to the MFE's data address.

2.5.3.4 The Manufacturing Process Integrating Infrastructure.

The interaction and the relationship between the Manufacturing Process Services and the Front End Addresses provides the logic in the Hybrid Structure for the integration of a wide variety of manufacturing processes into the system. Therefore we shall introduce a model of the internal functioning of the Manufacturing Process Integrating Infrastructure.

In Figure 2.15 it can be seen that the Manufacturing Process Integrating infrastructure is made up of a number of modules - these modules include

- A Data Address module - A specific module of data addresses that is used for every manufacturing process connected to a cell

- An Activity Control and Coordination Service - A manufacturing process dependent customisable service used for the control of activities at a particular manufacturing device.

- A Manufacturing Support Service - A generic module used to support the manufacturing activities of the cell. Services provided by this module include such activities as task queueing and manipulation as well as downloading of NC files to device proprietary controllers.

- A Capability Library - stores a library of capabilities. Each capability is a functional program designed to carry out the lowest denominator of functionality eg: Switch on, Start, Finish, Download program etc. Capabilities can be created and modified and ordered in different ways to create various desired effects in the system.

The functions and roles of these modules will be discussed in a later section.

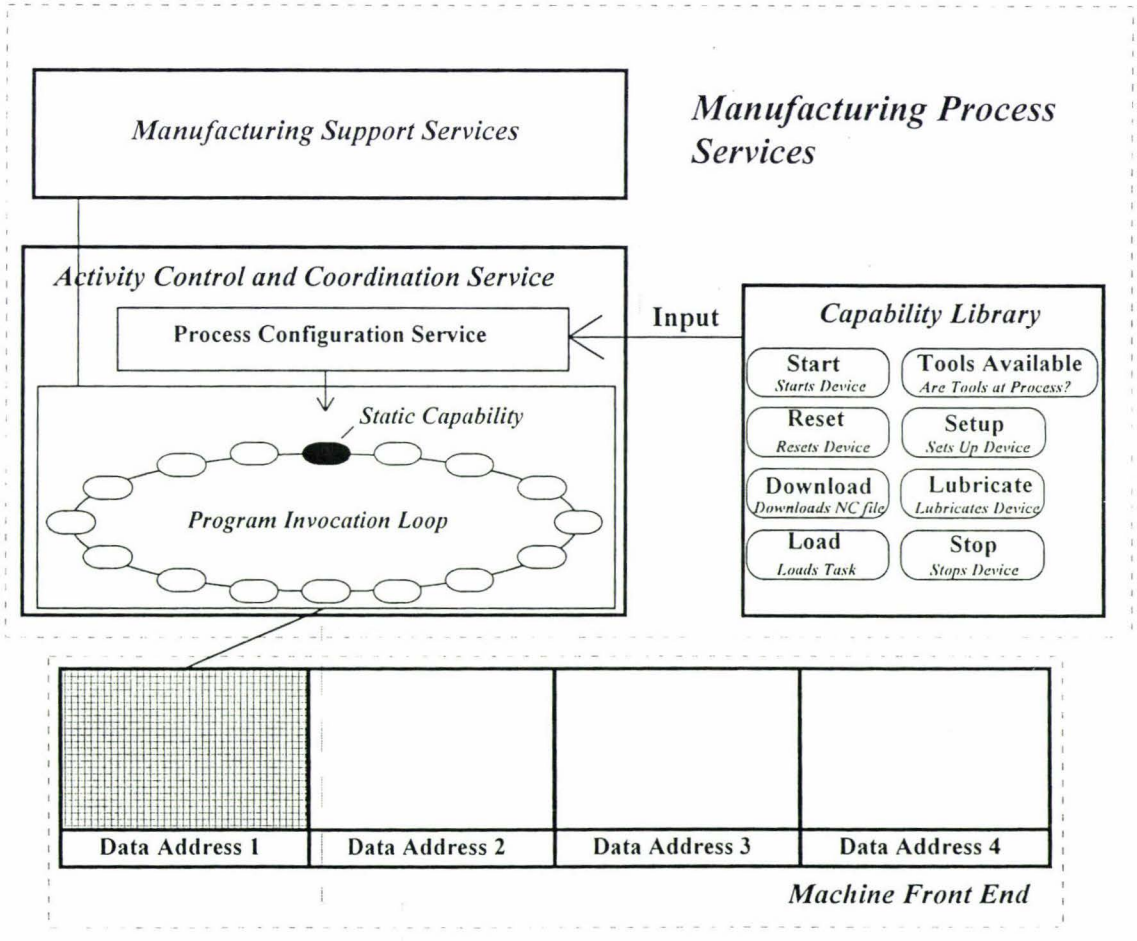


Figure 2.15 Manufacturing Process Integrating Infrastructure

2.5.3.5 The Virtual Manufacturing Device

The centre of the Manufacturing Process Integrating Infrastructure is the *Virtual Manufacturing Device*, the structure of which is made up of the group of modules outlined above. Figure 2.15 shows that the grouping of a Manufacturing Process's Activity Control and Coordination Service and its Data Address (Data Address 1) constitutes the "Virtual Manufacturing Device" or VMD as it is commonly called.

Mo and Tang[11] describe the Virtual Manufacturing Device as "an object that maps the logical organisation of the system to a real manufacturing device such as a robot, Programmable Logic Controller (PLC) or Computer Numerically Controlled machine". Mo and Wang[11] also describe the VMD as being "a virtual construct whose components can be mapped to identifiable parts of a real device" and having a role as being one that "lets the system's logical behaviour be consistent regardless of the physical device it is connected to" [11] - this idea is the heart of the philosophy in the design and development of the Hybrid Integrating Infrastructure in terms of the integration of Manufacturing Processes in the system model.

In the development of the Hybrid integrating infrastructure model it is the Manufacturing Process's Program Invocation loop that describes the logical organisation of the processing system in a generic manner and it is the MFE Status Address (see Figure 2.13) portion of the Data Address that maps this generic processing system onto the MFE (the MFE is typically a PLC or PSC system in the context of this research)

2.5.3.6 The Capability Library

The Capability Library contains a series of capabilities that can be used in the construction of the *Program Invocation Loop* that is used to monitor and control the state of the MFE via that MFE's Data Address - the library contains a list of all the routines that are potentially required in terms of the activities that are to be monitored and executed. The feature of the Capability Library is that it gives the user the ability to choose and order only those routines that are applicable to the behaviour of the *Manufacturing Process* that is being considered. Any other capabilities that are required can be added into the library and used as required. The idea with the capability library is that if new manufacturing devices are added to the cell all that is required is to add a new set of modular software to the cell and to select the required capabilities for the new device and order them as required via the Program Invocation Loop. Any new capabilities that are not part of the standard library must be able to be added and introduced to the loop simply - the hybrid architecture does this and the implementation will be discussed in the following chapter.

The capabilities within the capability library have two distinct roles:

- Logical control - they have the role of controlling the logistics of the system. Such logistics include the transmission of data to the Command Address or the Data Address and the initiation of the downloading of NC file data to an MFE's Manufacturing Process.

- Physical Control - The capabilities also have the role of physically instructing the Manufacturing Process to which they are connected via their MFE Status Address to change their physical state ie: Start, Stop, set up, check for the arrival of resources, tools etc.

2.5.3.7 Activity Control and Coordination Service

This service is responsible for the ordering and linking of the Capabilities to form a Program Invocation Loop. The Program Invocation Loop is a generic model of the typical series of events required to be carried out in the course of processing a particular task using a given type of process.

The Activity Control and Coordination service has two specific areas of functionality:

- ***The Process Configuration Service (PCS)*** - The PCS is responsible for the ordering and execution of the capabilities required to carry out the logical sequence of events to complete the task. The PCS functions by initiating the capabilities in the order that they are required to be executed as defined by the user. The user of the system defines at set up time the order in which the capabilities are executed depending on the generic behaviour of the process.

- ***The Program Invocation Loop (PIL)*** - The PIL consists of the capabilities themselves which are programmed to execute in a certain order. The PIL emulates the logical behaviour of the real device and the MFE data addresses provide an interface between the logical behaviour of the software and the real behaviour of the real world manufacturing process.

The capabilities making up the Program Invocation Loop may themselves carry out the following activities:

- The checking for the availability of resources (Tools, fixtures, materials etc) prior to the downloading of task data to the correct data address in the Machine Front End.

- Checking to see if there are any tasks required to be carried out by the MFE's associated manufacturing process.

- The checking of the availability of the MFE's manufacturing process.

- Executing the downloading of NC files in conjunction with the Manufacturing Support Service and the initiation of the start sequence in task execution by flagging the relevant command address.

- The downloading of task data into the machine front end of the associated manufacturing process prior to starting the execution of the task.

- Start, stop , interrupt, restart of one or more MFE's Manufacturing Process.

- Prompt a human operator to take manual intervention in conjunction with the Human Front End Service.

The Activity Control and Coordination Module can be broken down into two sub - modules as discussed:

- **The Process Configuration Service (PCS)** - Mo and Tang[11] state that a high degree of flexibility is possible by being able to configure the VMD as required. In the Hybrid system it is the PCS that gives the user the ability to statically configure the Program Invocation Loop depending on the type of Manufacturing Process that the loop will be associated with. The Process Configuration Service is also the driving mechanism behind the operation of the Program Invocation Loop. The PCS is configured to execute a series of Capabilities in such an order as to emulate the generic behaviour of the real world manufacturing process. Once activated the PCS sends a message using UMS to each capability telling it to execute, as each capability finishes its procedure. It then responds to the PCS informing it that it can initiate the next capability in its Program Invocation Loop. The reason that this reconfiguration is required for different Manufacturing Processes is that different processes require different services. An example of this is when comparing the difference between the operation of a CNC Mill and CNC Injection Moulding process. The CNC Mill process may require a capability in its invocation loop that actuates a coolant valve when it is required to be activated. However it is unlikely that this will be required for the CNC Injection Moulding Process. Some processes may require that a CNC file is downloaded early in the loop whilst others may require that the CNC file is downloaded late in the loop, this flexibility can be achieved by the implementation of the Process Configuration Service to implement the required VMD.

- **The Program Invocation Loop (PIL)** - the Program Invocation Loop is a high level control loop that is responsible for the state control of the Manufacturing Process for which it has been configured and is designed. The PIL is in effect a generic State Controller in that all state logic control classically implemented at the MFE level is brought to a higher generic level and the MFE becomes simply an

analog input and output device. This is required because of the many logic languages that are employed at the MFE level that would in effect negate the generic nature of the system due to the vast array of control logic systems employed. The PIL has a number of distinct roles:

⇒ Reads and writes data to its MFE Status Address at specified intervals in such a way that the most recent data can be dealt with and the tightest state of control can be maintained.

⇒ Communicates with the Manufacturing Support module via the relevant Command Address as required so that certain services can be accessed when needed.

⇒ Peruse the state control data collected from the relevant MFE Status address after each scan and then react to this data to control the Manufacturing Device by sending modified state data to the relevant data address.

2.5.3.8 Manufacturing Support Services - Supporting the Task Processing Role

The Manufacturing Support Services contain a pre-defined set of generic services that can be implemented in any enterprise and are used as tools by the Activity Control and Coordination Service to ensure that the execution of manufacturing activities is achieved. Within the cell it is the Manufacturing Support Service that provides generic routines that can be accessed by the Activity Control and Coordination service as required.

The types of routines that are provided by the Manufacturing Support Service include

- ***Manufacturing Task Queueing Service*** - This service provides for the queueing of tasks required to be carried out by each manufacturing process associated with each cell. Each manufacturing process at each cell has its own task queueing module associated with it as the design of the overall integrating infrastructure is modular.

Exact services provided by the *Manufacturing Task Queueing* service include:

⇒ The queueing of the tasks required to be carried out by a given manufacturing process in the order in which that manufacturing service acquired them.

⇒ Downloading of Task Data to the appropriate data address for the manufacturing task about to be carried out. When the Program Invocation Loop senses that task data is required to be

downloaded to the appropriate Command Address it sends a UMS message to the *Manufacturing Task Queueing* service telling it to carry this out.

⇒ The indexing of a job's task to the next task when the previous task has been completed and the queueing of the job associated with that task onto the out going queue of that process prior to scheduling using the bidding algorithm.

⇒ Updating the time required to carry out the given tasks in the in going task queue for a process. Each time a task is added to the list of tasks required to be carried out or after a task has been removed from the list prior to being downloaded with its data to the Machine Front End Addresses, the Manufacturing Task Queueing service calculates the time required for the associated Manufacturing Process to execute the list of tasks in the in going queue. Once this calculation is made the new time is recorded by the service in the *Local Data Storage Service*. Updating of this service is vital as this information is used in the bidding algorithm for the distribution and scheduling of tasks.

⇒ The movement of tasks from a processes in going queue to out going queue in the event of process failure.

⇒ The Global Scheduling of tasks off the out going queue for each Manufacturing Process using the bidding algorithm in conjunction with the *Task Scheduling Services*.

- *NC Program Download Service* - this service is responsible for the downloading of NC files to the Manufacturing Process's Proprietary Controller in response to the state of the Manufacturing Process's Program Invocation Loop, when the Program Invocation Loop has reached a state such that an NC file is required to be downloaded this is carried out by this function.

The Communication philosophy between the Manufacturing Support Module and the Activity Control and Coordination Module is shown in Figure 2.16 using the NC Program Download Service for MFE1's Manufacturing Process (MP1) as an example.

The heart of the Cell Controller is the Program Invocation Loop. This manipulates the state of the manufacturing process that it is controlling at a given time eg: Set up State, Run State, Program Download state etc. In Figure 2.16 a snapshot in time reveals the situation whereby the Program Invocation Loop has just initiated the Download capability, when this is executed the following events occur:

1. The Download Capability portion of the Program Invocation Loop sends an Local Message Transmit UMS that switches the Program Download Command Address on.

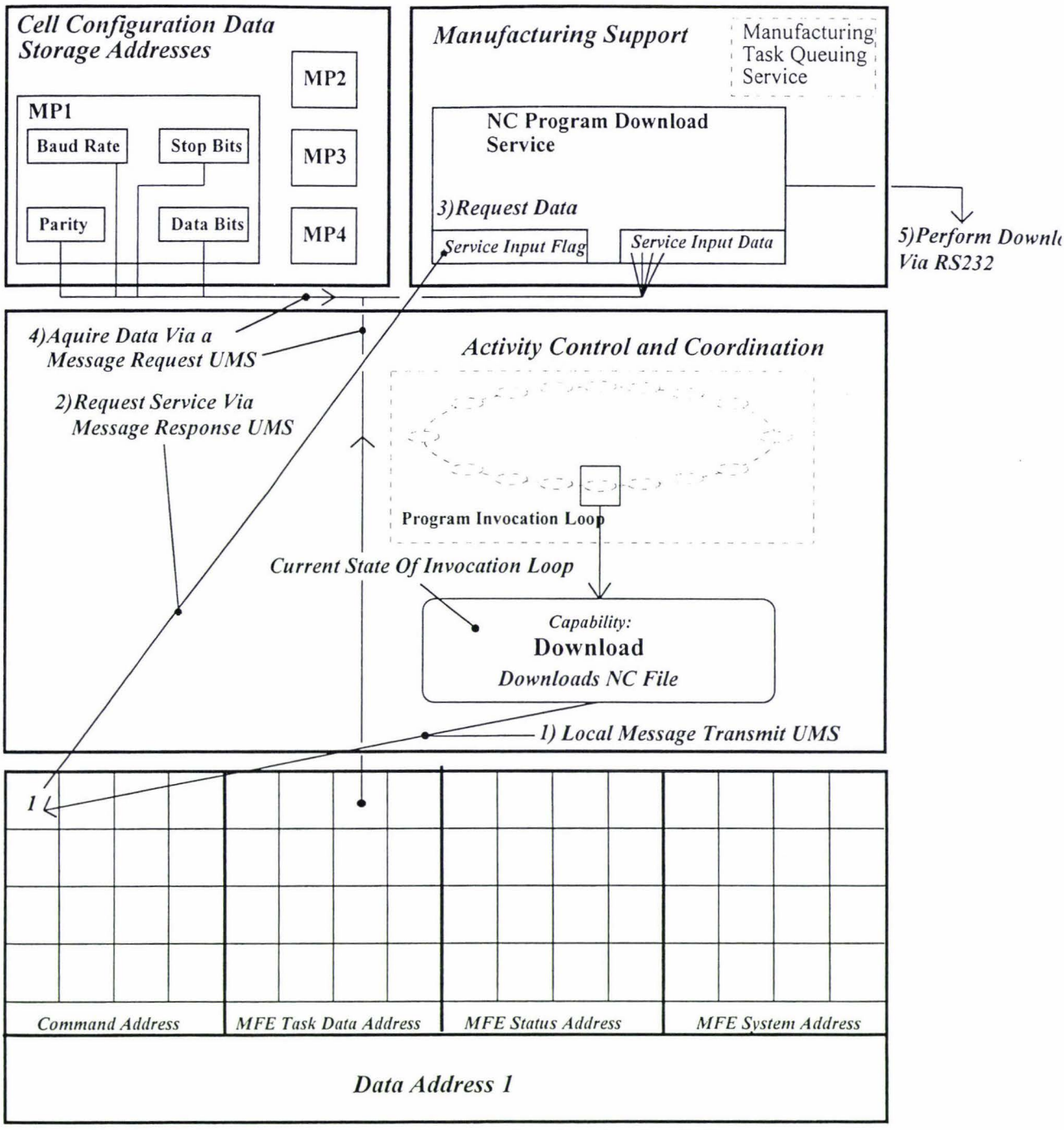


Figure 2.16 Role of Manufacturing Support Services in the Cell

2. The NC Program Download Service is connected to the Program Download Command Address via a Message Response UMS. Therefore when the address is flagged the service is activated.

3. The NC Program Download Service requests the first Manufacturing Process (MP1) transmission parameters from the Cell Configuration Data Storage Addresses so that it can transmit the required NC File under the correct transmitting conditions and via the correct communication port. It also requests the name of the NC file to be downloaded from amongst the Task Data resident in the MFE Task Data Address.

4. The required transmission data and the name of the NC file to be transmitted is received by the NC Program Download Application via the service input data facility.

5. Once the Program Download Service has received the data via its service input data facility it uses it to download the NC file from the Network Data Storage Service to the associated Manufacturing Process's Proprietary Controller via the RS232 link.

The ***Manufacturing Task Queueing Service*** interacts with the Activity Control and Coordination module in a way very similar to the way in which the ***NC Program Download Service*** does in terms of the use of the UMS system. The important aspect is that the logical grouping of such a system allows for a high degree of modularity to be obtained. The modules are loosely coupled therefore minimising the possibility of error. The modular construction also allows for easy and fast debugging. Once a module's software is debugged then all modules in all cells of the same type will be free of bugs. This coupled with the weak links between modules as well as the predictability of the system makes debugging far more logical and simpler.

Therefore Figure 2.16 emphasises the nature of the relationship between the Manufacturing Support services and the Activity Control and Coordination module. It can be observed that the behaviour of the Program Invocation Loop is such that it calls upon the services of the Manufacturing Support Module when and only when those services are required via the relevant Command Address for the associated MFE. This command address interface ensures that the system is generic in that every Capability is programmed to link with a specified command address which in turn is configured to activate a flag in the required service. This means that the Process Configuration Application can be used to execute any capability in the capability library in any desired order without having to reconfigure the interface every time because each capability is intrinsically linked with the required variables or addresses. Extra flexibility is introduced in the fact that each capability can have its linkages with variables changed as required.

2.5.3.9 Cell Configuration Data Storage Service

The model of the Cell Configuration Data Storage Service is shown in Figure 2.17.

The Cell Configuration Data Storage Service is responsible for the storage of data in relation to the configuration of a given cell and can be broken down into a number of levels as shown.
The lowest level of configuration is the detail of the individual Process Features, which details what the specifications of the individual manufacturing processes are.

The Process Useability configuration of a process within a cell is at the next highest level and is indicated by the Type\Size data as shown.

The highest level of cell configuration looks at the overall processes of the cell and results in the Process Type and Quantity configuration of the cell.
This service is responsible for the storage of data that describes the configuration of a given Cell Controller in three areas:

- ***The Type and Quantity of Processes within a Cell*** - the Cell Controller as configured is required to know what types of processes it has within its cell so that it can respond accordingly to requests made to it by the Factory Controller, this knowledge is stored within this service.

- ***The Useability of each Process within a Cell*** - the Cell Controller is required to know the useability of its processes within the cell, for instance a process can be in any of the following states of useability:

  ⇒ Long Term Break Down - The process maybe in a broken down state whereby the duration of the breakdown is deemed to be long term and therefore unusable. In this case it is not acceptable for the cell to accept any tasks to be done by this process. A typical case of a long term breakdown would be where a process component that was not in stock failed and was required to be ordered. In the event of a long term breakdown the Type/Size address would be switched to zero. This address is linked to the *Manufacturing Task Queueing Service* and as such would direct it to place the tasks on the in queue into the out queue therefore automatically causing the system to Rebid the tasks.

  ⇒ Short Term Breakdown - In this state the process is also deemed to be unusable. However there are differences between the Long Term Breakdown and this state that will be explained later. A typical case of Short Term Breakdown maybe gear failure on the headstock of a lathe which could be quickly replaced by a gear in stock. In this case the Rebid of tasks would not automatically occur. However the user at the Factory Controller level can still switch the Type/Size address to zero (a Hierarchical activity).

  ⇒ Implementable State - In the implementable state the process is useable and can potentially accept task requests from the Factory Controller.

  ⇒ Idle State - In the idle state the process is not broken down but is idle for some other reason such as the need to carry out some form of preventive maintenance upon it. In this case the process is said to be unusable. In this state the process does not participate in any scheduling

activities and cannot compete for tasks. However if the gear change is for only a short time and the system administrator still wants this process to bid even though it is idle he can reset the Type/Size address remotely from the Factory Controller.

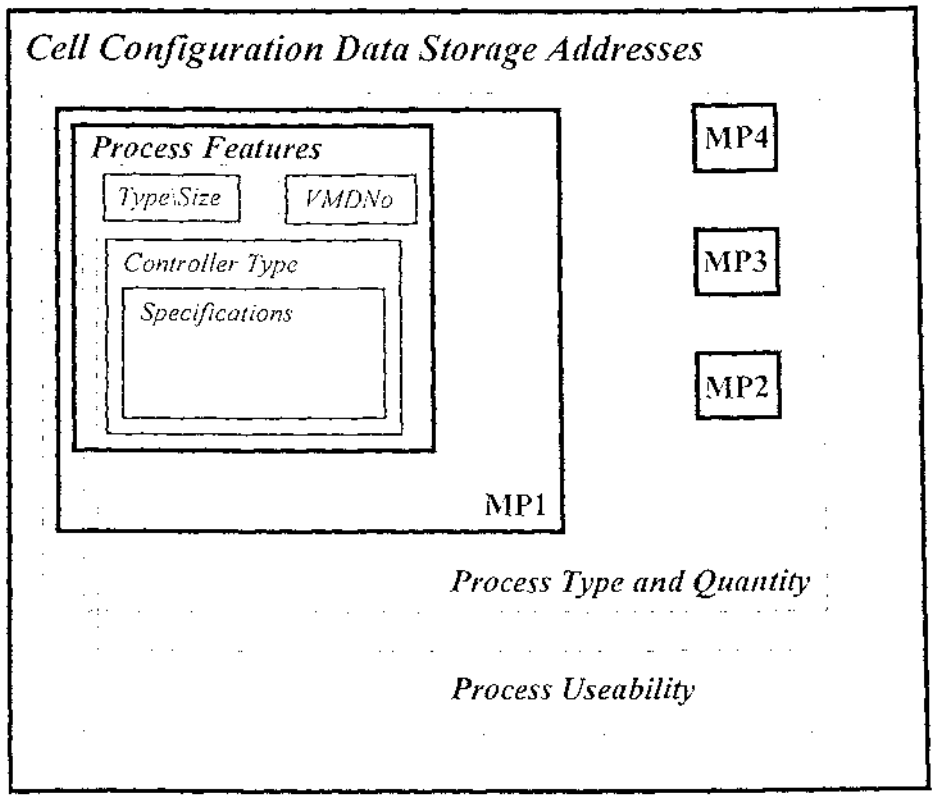


Figure 2.17 Cell Configuration Data Storage Address Model

The Cell Controller is required to know the useability state of the Processes within its cell for the following reasons:

- So that it can decide on whether to participate in the scheduling of a task that has been presented to it by the Factory Controller requiring a certain type of process which maybe within its cell.

- So that it can deactivate the Process Configuration Application associated with a process in the event of its breakdown or for some other reason such as the need to perform preventive maintenance.
- If a process is in a short term breakdown useability state it cannot accept tasks but it can retain the tasks that it has currently queued for it. However if it is in long term useability state the tasks should be recovered and rescheduled

- ***Features of Processes in the Cell*** - The Cell Controller must be able to store data that details the features of the individual processes under its control. Some of these features include:

  ⇒ A Process's type specification and the VMD it is associated with.

  ⇒ The size of the process and therefore its capability to cope with tasks of various dimensions.

  ⇒ A type of Proprietary Controller for a process.

  ⇒ The Proprietary Controller's transmission specifications such as Baud Rate, number of Stop Bits, number of Data Bits and Parity Data information.

2.5.3.10 Local Data Storage Service

The Local Data Storage service is responsible for the storage of various forms of data that are vital to the efficient functioning of a cell. These forms of data include:

- *System Data* - the storage of application based data that is fundamental to the successful execution of the cell controller software.

- *Process Queueing Data* - The storage of data relevant to the in going and out going queues of each process within the cell. Such data includes the length of the queues for each process and details of the tasks in the queues.

2.6 Summary

This chapter of the thesis has given an overview of the behaviour of critical aspects of the Hybrid Architecture FMS system. The system has been developed with a number of important objectives in mind - flexibility, modularity, extensibility, fault tolerance and easy integration of a wide variety of hardware and software systems. To get a real appreciation of how these objectives are implemented in a software sense it is important to explain what software was chosen for the programming and how aspects of that software were used to perform the issues above. The objective of Chapter 3, therefore, is to explain the software used in the system development, why they were used and aspects of the development software that were conducive to meeting the required objectives.

Chapter 3

Implementation of the Hybrid Model - General Services

3.1 Introduction 88

3.2 Overview - Intellution FIXDMACS GUI 89

3.3 Implemention of the Hybrid FMS - General Services 96

3.4 Summary 124

3.1 Introduction

In Chapter 2 the overall Hybrid Model was presented and aspects of all services were discussed in the form of models. The design of an architecture and model is a critical aspect as it provides a high level view of the overall functionality of the FMS and provides rules and guidelines for the software development. Dilts *et.al* stated in his work that an FMS system can only be good as the design of its architecture and the pretext on which it is developed.[54] Researchers such as Hatvany, Dilts and Veeramani *et.al* have stated that if the development guidelines and assumptions behind them are unverified and therefore the designs and system architecture is poor the performance of the system will also be poor.[9,16,19] The main aim behind the design of an architecture, therefore, is to provide a systematic way of developing a structure that will assure that the given objectives are met. This was the problem with the Heirarchial architecture - it was based on a general hierarchical management structure not based on a desire to meet certain performance criteria. Systems such as CIM-OSA[8-9] are, in essence, performance based systems (they have strategic intent) as they are used to model a wide variety of enterprises and as such can be adapted to existing paradigms that a particular company already finds successful.

The way in which the software is developed to meet the requirements of the system is also important. Firstly it must be developed in such a way so as to emulate the model effectively and secondly the amount of programming must be kept to a minimum so that the system is economically viable. FMS systems have to be justified like any other capital project. Obviously substantial costs go into the software development side and so keeping this in check has a major bearing on the viability of the software. High software development costs by the system vendor are transferred to the customer and the customer may find justification of the required capital expenditure impossible. One of the options available as indicated by Inglis[53] is to integrate existing commercially available software into an FMS system. Inglis states that by using this approach small companies can now afford to implement their own FMS system due to the dramatic reduction in investment required. The viability of the software used is, as Inglis mentions, an important factor in that it has to be proven and that the developers must have a proven track record over a number of years.

The approach taken in the context of this thesis was to use two major software packages - *Intellution FixDmacs GUI* and *Microsoft Visual Basic*. Features of these packages are discussed in the following sections:

3.2 Overview - Intellution FIXDMACS GUI

Intellution FIXDMACS GUI is a powerfull SCADA (Supervisory Control and Data Aquisition) package with true distributed processing characteristics. The first version of FixDmacs was released in 1988 and at that time it was the first PC-based SCADA package available with true distributed processing. The software provides realtime aquisition of data from industrial plant and provides programmable routines capable of processing the data and transferring it back to the real device. In this way it can be used as a control tool.

Figure 3.1 illustrates the data flow on a SCADA node.

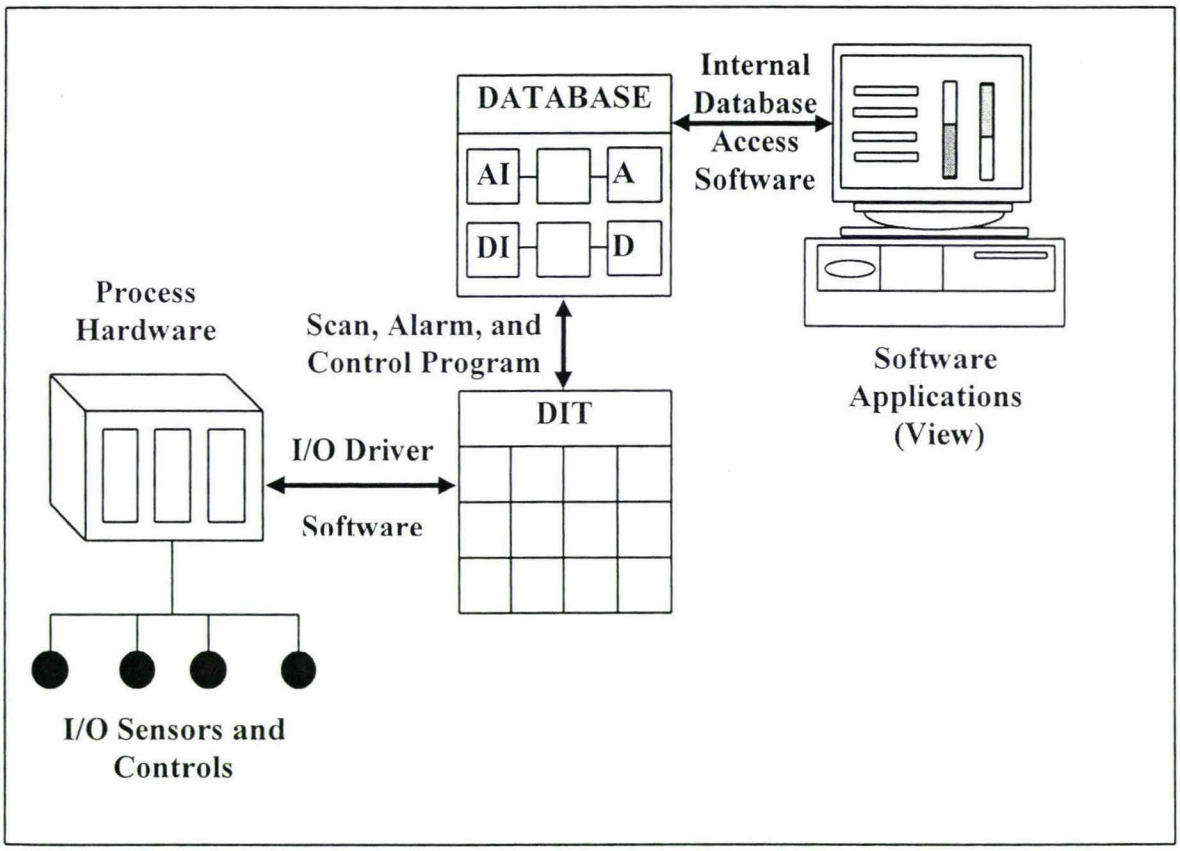


Figure 3.1 - Basic Architecture of FixDmacs

3.2.1 I/O Drivers

The first thing FIXDMACS must do is aquire raw data from the process. Sensors and controllers send data to registers in the PLC/PSC or other process hardware. FIXDMACS provides a software interface to the hardware that can read the data from the specified registers. The data table is called the Driver

Information Table or DIT. There is a high degree of flexibility at this level in that there is a large number of I/O drivers available to service a large range of hardware systems. Therefore in a CIM system having a large variety of hardware systems most, if not all of them, can be catered for.

3.2.2 Database

The heart of the FIXDMACS software is the process database. The database is a representation or virtual model of the process connected to by looping together blocks of process control logic. The user creates the database and process control logic by using the Database Builder program. The information the user enters into the database includes information about the driver and the address of the data in the DIT. Generally speaking there are two different types of blocks used in building a database:

- **Primary Blocks** - Are used for accessing data from and writing data to the DIT
- **Secondary Blocks** - Are used for the manipulation of data passed to them.

A typical linking of blocks is shown in Figure 3.2 in the form of a chain.

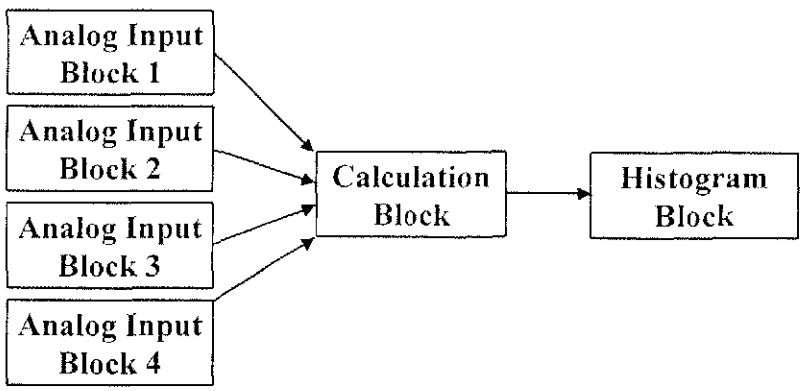


Figure 3.2 - Example of a Database Chain

The example represents a single chain within a database that is used for the calculation of average temperature of a furnace over time. At regular intervals the Analog input blocks are scanned by the SAC (Scan, Alarm and Control, see section 3.2.3). The Analog input blocks fetch data from the DIT relevant to the calculation, pass it to the Calculation block which calculates the average furnace temperature and the result is then transferred to the Histogram Block. The GUI (Graphical User Interface, see section 3.2.4)

can then be linked to the Histogram Block to give a graphical view of the moving average for the furnace temperature.

3.2.3 Scan, Alarm and Control Program

The Scan, Alarm and Control (SAC) program runs on each SCADA node and is responsible for executing the logic of the database chains. The SAC program does this in the following way:

- Fetches data from the DIT.
- Translates the data into a format expected by the database.
- Checks the data against the alarm limits and generates alarm messages.
- Executes the Control logic.
- Writes requested data to the DIT.

There are various means by which the chains are processed:

Time Based Processing of Chains:

The mechanism above is generated using time based processing - primary blocks have scan times associated with them and therefore the chains are processed as above at regular intervals as dictated by the scan time of the primary block proceeding the chain. The scan time intervals that can be set for primary blocks can be in terms of fractions of seconds, seconds, minutes or hours depending on the application. In the example in Figure 3.2 it may be necessary that the moving average of furnace temperature over hour long periods be calculated and displayed - in this instance the scan times of the primary block would be 60 minutes.

Exception Based Processing of Chains:

SAC can initiate exception based processing in reponse to the following events:

- Data Changes in the DIT - This is important as it allows immediate responses to be made if required.
- Operator Actions - The operator can change the status of a block's field. As soon as this status is changed the SAC responds.
- Instructions from external software packages - Many software packages support EDA (Easy Data Access) and DDE (Dynamic Data Exchange). These functions allow writes to be made from other applications to certain database blocks and, therefore, as the blocks status changes the SCA responds.

Exception based processing is particularly useful in the case of EDA/DDE requests from other applications as it allows FIXDMACS to be prompted by other applications requesting a service.

3.2.4 Man-Machine Interface

The most important software in the FIXDMACS system is the software that provides the graphical illustration of the behaviour of the real world plant. Not only is the software able to give an illustration of the current state of actual equipment on the node that the processing equipment is connected to but it also is able to give a representation of the state of equipment connected to other nodes from a remote node using its networking facility. This functionality is provided by the software called the FIXDMACS GUI (Graphical User Interface). The GUI enables the user to create virtual representations of the real equipment the software is connected to. The user can draw the machine or functional objects with buttons/controls etc in the *draw module* of the software (and define links to database blocks) and manipulate those controls in real-time in the *view module* to influence the behaviour of a local or remote device. The beauty of such a system is that a whole process environment can be monitored and manipulated from a single node.

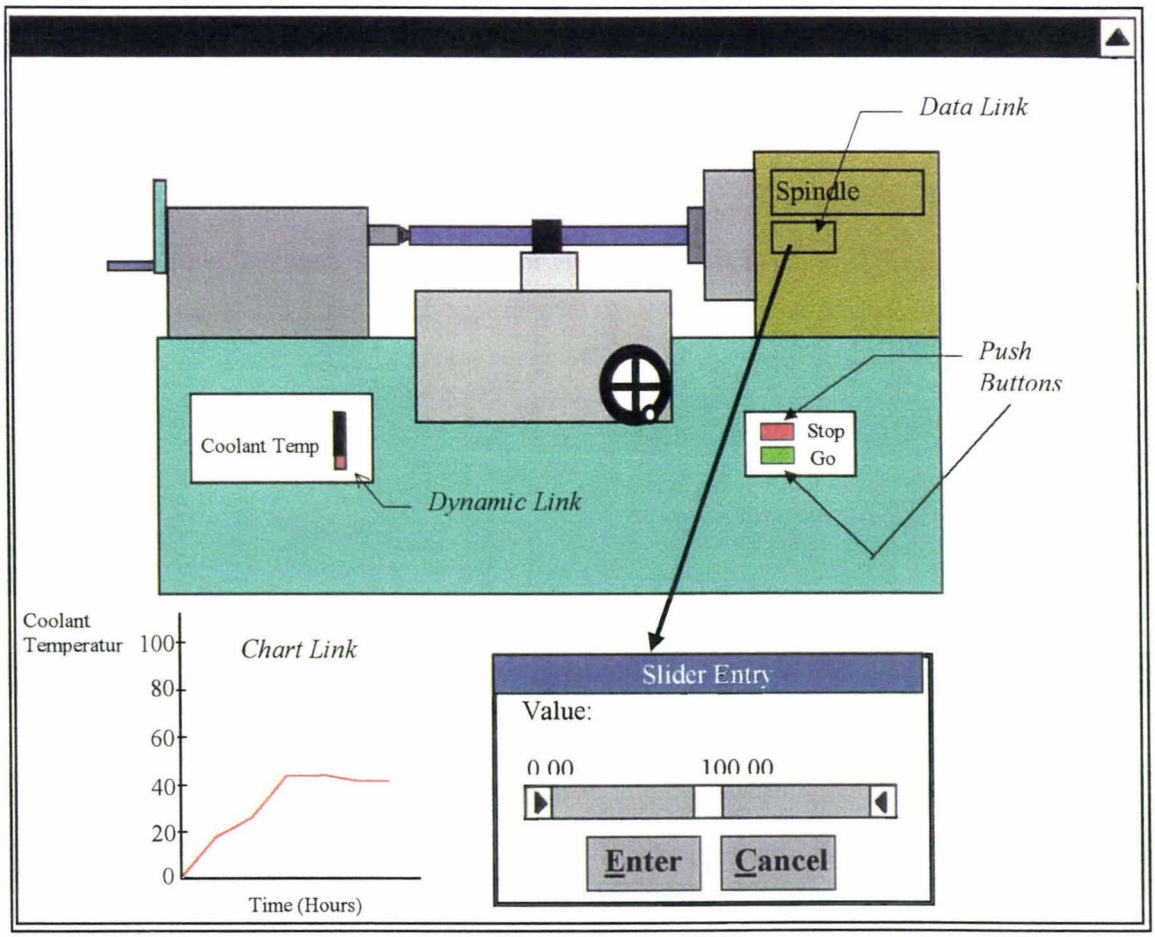


Figure 3.3 - GUI Representation for a lathe

Figure 3.3 provides an illustration of a GUI interface for a lathe showing a number of the features available using the GUI.

The coolant temperature dynamic link is created by linking the GUI to an analog input block and specifying the maximum and minimum expected values when defining the block parameters. The link is created dynamically so that as the temperature varies so to does the amount of fill of red in the rectangle making up the object, thus providing an up to date pictorial reading for the user. A chart link can also be created with the analog block so that the changes can be logged against time and displayed as shown.

Push buttons can also be created so that the user can develop a virtual system in that when the buttons are pushed on the screen, the SAC, I/O Drivers and process hardware ensure that the actions are transformed to real effects on the actual device. The FIXDMACS command language can be used to create command scripts for these buttons to perform a variety of actions.
Data links provide a textbox link showing the current value of the spindle speed (in this case). The user can modify this speed by clicking the text box and using the slider entry (in this case) to set the new speed for the lathe. After modifying the speed the new setting is displayed in the text box.

3.2.5 Handling Multiple Views of System wide Equipment

In a Hybrid FMS architecture it is therefore possible to have a library of picture files representing the generic behaviour of every manufacturing device in the system. A library of these generic files can therefore be held at each Cell Controller so that the user at that cell is able to visually examine the state of any device at any time in the cell.

When a new device is to be installed in a cell of a known type the picture is already generically linked to the database and all that needs to be done is for it to be viewed. The Macro Editor feature of FIXDMACS allows short lines of code to be written and assigned to any function key. In this case for example a generic cell's macro is assigned so that the function key F1 opens Manufacturing Device 1's view file, F2 opens Manufacturing Device 2's view file etc. In this way the user is able to access the view file of each Manufacturing Device on that node and influence its behaviour ie: Stopping, Starting, Intervening etc.

At the Factory Controller level of the hierarchy it may be important for the node to have a library of picture files representing the states of all the devices in the overall system. This system and the access protocol is shown in Figure 3.4. At the Factory Controller level a pictorial representation of the hybrid layout is shown. When the state of a particular manufacturing process in a given cell is required to be accessed, the relevant push button is pressed as shown to give the view of the cell. This push button contains a customisable language script capable of activating the required layout as shown (in this case the picture of the machine state making up manufacturing device 1). Once in this picture additional push

buttons can be used to toggle between various pictures representing different Manufacturing Process states within the cell. These various pictures are then able to be viewed and controlled by the operator from the Factory Controller meaning that he is able to intervene and control devices all around the system from a remote source and, therefore, is able to execute the Heterarchical aspect of the Hybrid Control architecture.

3.2.6 Limitations of FIXDMACS Software:

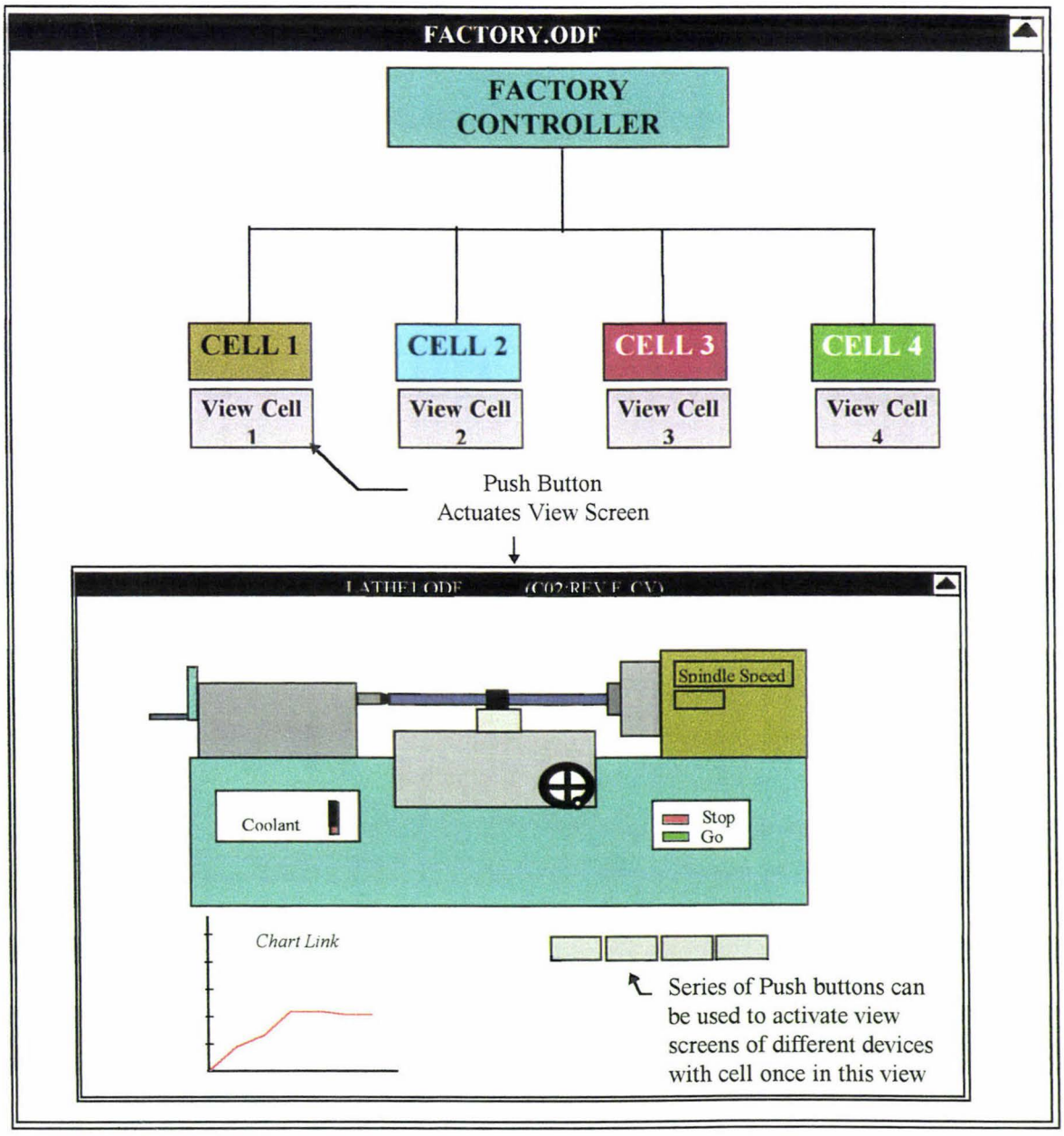


Figure 3.4 - Typical Heirarchy of views - Factory Level

The FIXDMACS architecture is ideal for the supervisory control of equipment throughout the Factory Environment in that it provides a fully distributed tool for access to a large number of processes throughout the Factory Environment. It does not, however, have the functional ability to fulfil a number of other features of the Hybrid Architecture such as file handling, task management and CAD/CAM as well as a number of other activities. The FIXDMACS system can aquire data through its database and SAC feature, manipulate it and transmit it back to the real device. It can also provide an interface between the human operator and the real machine. It provides an ideal platform for the Hybrid FMS system as shown in Figure 3.5 but lacks the Cell Management and System Management functions of the Cell and Factory Controllers.

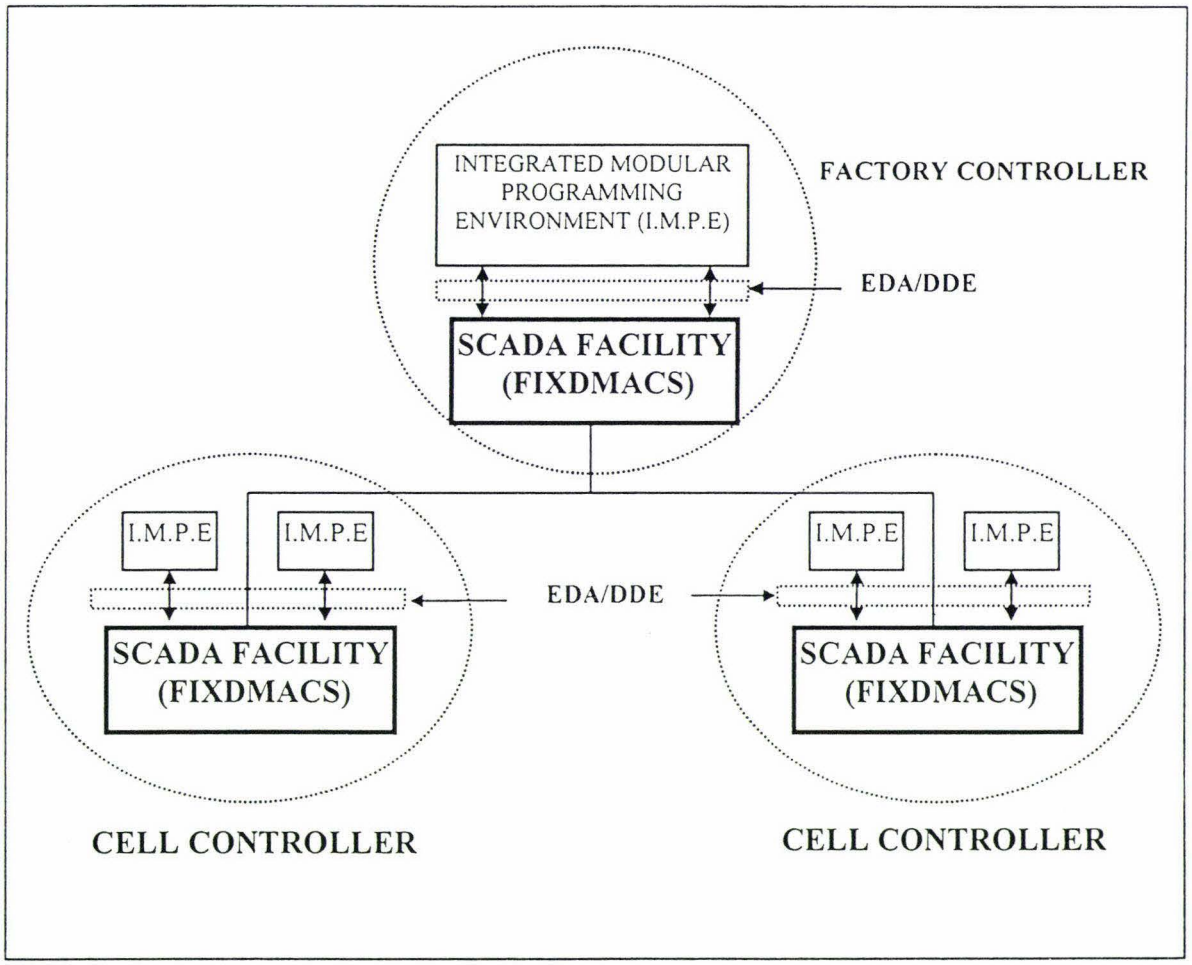


Figure 3.5 - The FIXDMACS role in the Hybrid Architecture

As can be seen from Figure 3.5 it is highly practical to implement the Hybrid model using a FIXDMACS base integrated with a standard programming environment. Fortunately this process is reasonably straightforward using the EDA (Easy Data Access) and DDE (Dynamic Data Exchange) functions to create the link between itself and a number of applications. This means that a programming environment must be implemented for use with FIXDMACS. These functions allow FIXDMACS to be able to

exchange text and numeric data and allows DDE and EDA to be used to provide essential services to the FIXDMACS scada environment. The programming environment that was chosen was Microsoft Visual Basic as it is fast becoming an industry standard in terms of a programming system. It supports DDE & EDA and is quick to program in and it is relatively easy to use when compared with the C++ environment. The programming language is also *event driven* meaning that it is particularly useful in the Hybrid architecture which is, to an extent, driven by the passing and receiving of messages using the UMS system.

3.3 Implemention of the Hybrid FMS - General Services

3.3.1 Implementation of the UMS system Model

The UMS system messaging model as shown in Chapter 2, Figure 2.6 can be implemented with relative ease using the EDA/DDE messaging system. The data packet concept that is part of the UMS system can easily be implemented. All factors within UMS are intrinsically taken care of by the EDA/DDE system. The Global Transmit UMS in the model can be easily taken care of with the system's ability to send messages from the Visual Basic application in one node to a FIXDMACS application in another. A connection between a database block in a node and a Manufacturing Process Services application can be set up using a DDE link, therefore, satisfying the requirements of a Message Response UMS. A service can therefore be requested through the database block interface. A Message Request UMS is used in the circumstance whereby a service needs the status of a real world device and is a simple EDA function. After interrogating the device it can then make a decision. A local Message Transmit UMS can be used to activate a command address and this can be carried out by an EDA message.

Clearly then the EDA/DDE software can implement all messaging requirements of the model.

3.3.2 Implementation of the Node Polling Application

The node polling application window is shown in Figure 3.6. As has been mentioned in the node polling model definition, the purpose of this application is to give every node equitable access to the Task Bidding application. At the moment this application resides in the Factory Controller, the premise being that if the Factory Controller were to crash no bidding would be allowed to take place anyway. When the application is loaded the first thing that occurs is that the timer event actuates. In Visual Basic the timer

event actuates applications to ensure that an item of program runs at regular intervals. For the case of this timer the timing interval was set to 5 seconds as this seemed to be reasonable.

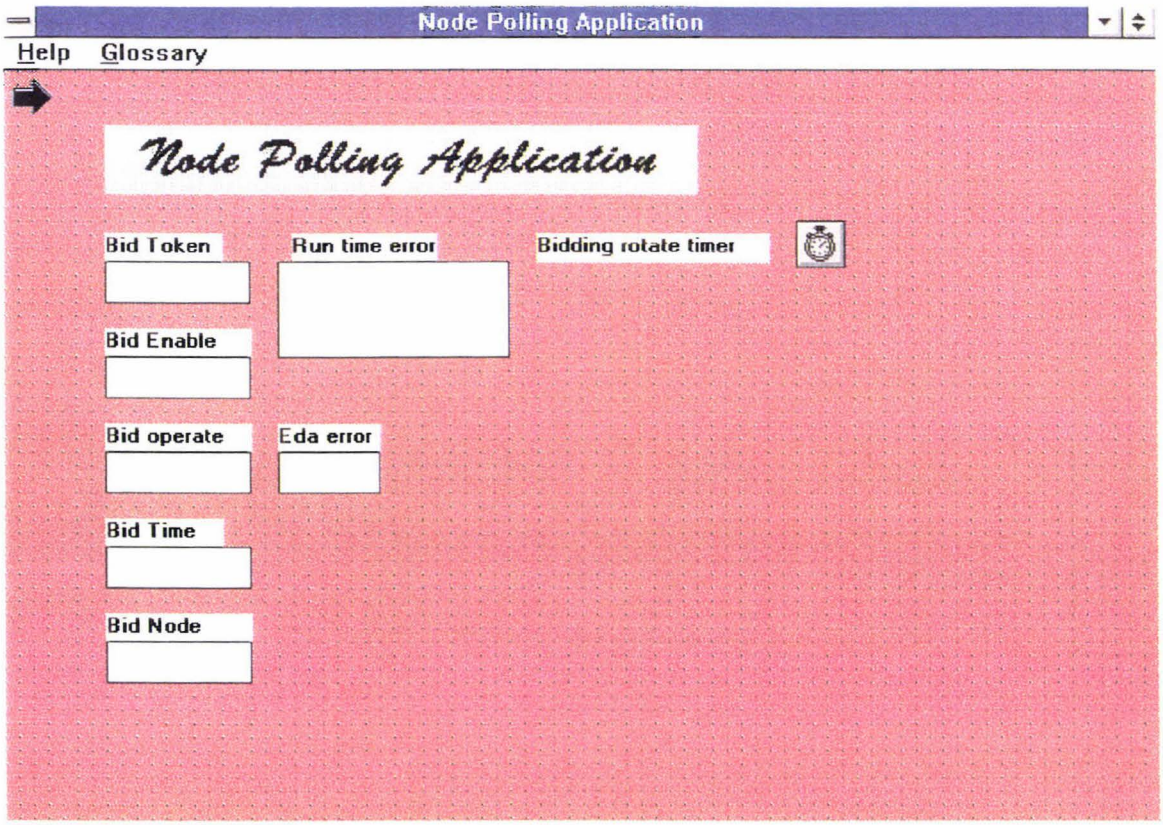


Figure 3.6 - Node Polling Application Window

When the timer reaches the 5 second delay, it executes the following:

1. It opens the file "C:\Factory\Config\nodefile.txt" for read access.
2. It disables the timer so that it cannot respond.
3. It checks a global variable to see if any nodes have been polled (in other words checks to see if the application has just been loaded),
4. If the application has just been loaded it reads the name of the first node from the file containing all nodenames (nodefile.txt - Formed at the Factory Configuration Stage). This node becomes the node to be polled first. The application then sends a flag to the Bidtoken database block (FIXDMACS) in the node via a Global Message Transmit UMS (GMT UMS) thus passing the token to it and giving the opportunity to that node to hold an auction. It then reactivates the timer. If the node has any tasks to auction it secures the token by placing a tag on the Bidoperate database block and starts the auction. When it has finished auctioning the node takes its reserve off the token by taking the flag off the Bidoperate database block so that when the node polling application interrogates the block it will then remove the flag off the Bidtoken database block and pass it to the next node.

5. If the application has already polled another node when the timer reactivates, the node polling application retrieves the value of the Bidoperate database block in the current node.
    - If the value is 0, then the node has no jobs it wants to auction and the node polling application sends a message to the nodes Bidtoken database block thus deactivating the flag using a GMT UMS and activates the next node as stated in clause 4 above.
    - If the value is 1 then the node is currently auctioning. If the timer has been activated 9 times (equates to 45 seconds) and the node is still reserving the token ie: the Bidoperate database block is flagged on, then the node polling application changes its value to 0 using a GMT UMS and also changes the value of the nodes "Bidtoken" database block. It then gets the name of the next node from the nodefile and sends it the token as outlined in clause 4 above.

After each pass, variables and arrays exist within the application to keep track of the current nodes and the state of the timer (ie: Whether or not it has reached the 45 seconds or not).

The node polling application gives each cell the opportunity to auction a maximum of one job from each of the manufacturing devices in going queues. Once the cell controller software has attempted this it changes the value of its Bidoperate database block to zero (flag off) using a Local Message Transmit UMS. Therefore when the node polling application interrogates the FIXDMACS Bidoperate database block and seeing that it no longer reserves the token it removes the token (using a Global Message Transmit UMS) by setting the Bidtoken database block to zero.
The node polling protocol is a generic type of application that can run on any Network Protocol that is supported by FIXDMACS. An excellent feature of the system is that the links it has with the overall cellular environment are very loose. The only inputs that are required to make it work is a file containing the names of the nodes presently configured for the environment and a set of standard FIXDMACS database blocks that are standard with every cell that is constructed. If neither of these exists, the system is designed to degrade gracefully. In the event that no cells have been defined for the system the application will load but no polling will occur and the user is told what the problem is. In the event that a cell crashes during operation the system is programmed to bypass the cell and continue to poll the next cell without any disruption to the system. When new cells are configured the cell names are added to the nodefile along with the new database as part of the normal configuration process. This means that for the normal operation of the node polling system nothing extra has to carried out to make sure it works when new cells of manufacturing devices are added.

3.3.3 Implementing the Task Scheduling Service

The distribution and scheduling of tasks is one of the most important features in any FMS system. As has been discussed earlier in this thesis the implementation of the task scheduling system is based on many

years work by many researchers in the Contract Network Protocol approach. As has been discussed there are two variations on the theme - Factory based and Cell based allocation. Each will be discussed in turn.

3.3.3.1 Factory Based Allocation

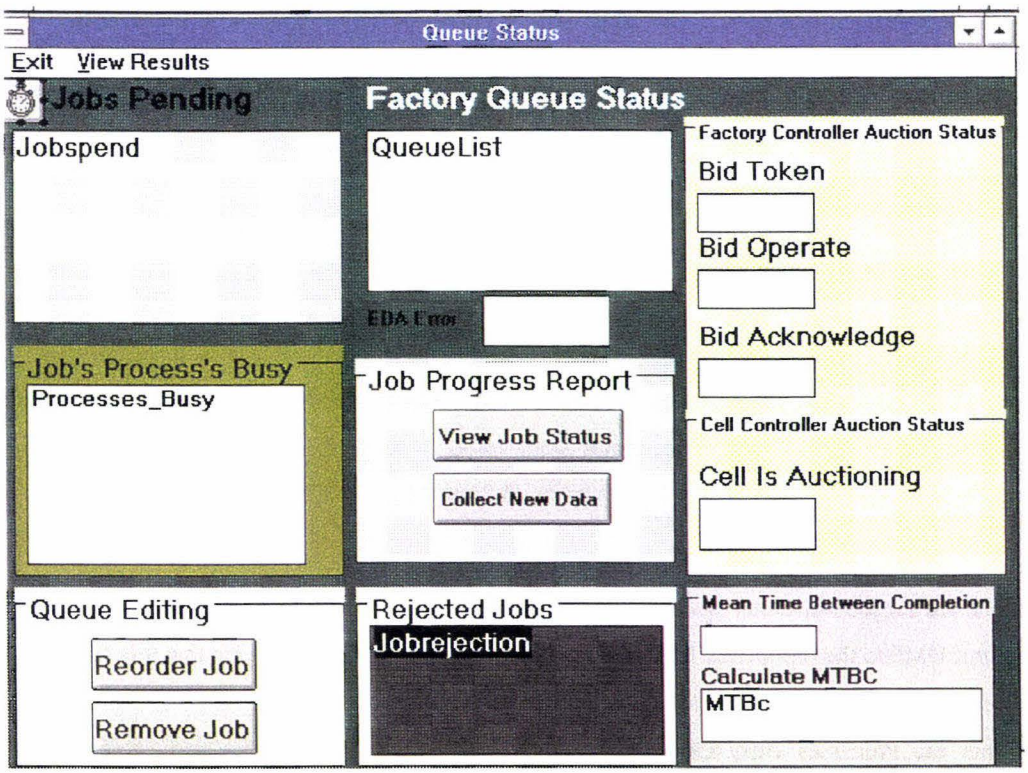


Figure 3.7 - Factory based Task Scheduling Application

With factory based allocation the scheduling of the task activities is initiated within the Factory Controller software. The window that is primarily associated with this activity is the Queue Status window as is shown in Figure 3.7 and is directly accessed from the Factory Controller front end.

- ***Initial Queueing of Jobs at the Factory Queue***

After the jobs are defined in the Job Definition Service (see later in Chapter) they are placed in the Jobs Pending File (JPF). The JPF becomes a holding mechanism for jobs that are required to be undertaken in the future but are not yet ready to be started. In the Factory Controller front end a timer event exists that is actuated at certain time intervals. This timer interval can be set in the Factory Controller Parameter Service. This timer is responsible for scanning jobs in the JPF and looking for matches between todays date and time and the start date and start time required for each job in the queue. If a match is found for any job in the JPF it is put in the Queuelist list (see Figure 3.7). If the job is urgent it puts the job at the

top of the queue otherwise it goes to the bottom of the queue. After a job is extracted and queued, the JPF is updated to reflect the change and the contents of the JPF is written to the Jobspend list so the user is able to view what jobs are due to be queued in the future.

- ***Carrying out a Task Announcement***

A task announcement is the act of distributing the task information relevant to bid fomation around the Cell Controller nodes in the factory environment so that the nodes can evaluate the information and respond accordingly.

A Message Response UMS is set up between the Bidtoken FIXDMACS database block in the Factory Controller and the Bid token textbox. When the Bidtoken arrives at the database block a 1 is inserted in the Bidtoken textbox as shown in Figure 3.7 by the link. This signifies that the token has arrived and the application can respond. It does so in the following way:

1. It checks to see if there are any jobs in the Processes Busy queue. If there are not, it checks to see if there are any jobs in the Queuelist queue.

2. If there are no jobs in either queue the application releases the token by sending a Local Message Transmit UMS to the bidoperate FIXDMACS database block thereby ensuring that it is flagged off.

3. If either the Processes Busy Queue or Queuelist queue contain jobs the application secures the Bidtoken by sending a flag on to the bidoperate block thereby securing the token.

4. If both queues contain jobs the application starts by trying to auction all jobs off the Processes Busy queue and then one job off the Queuelist queue. The reason for this is that jobs on the Processes Busy queue are already behind schedule as an attempt has already been made to schedule them but all potential manufacturing processes have had full buffers. They take priority therefore.

5. If a task announcement is to be carried out from the Processes Busy queue the following steps occur:
    - If all jobs on the queue have attempted to be auctioned (during this node polling) from the Processes Busy queue then the application tries to auction a job off the Queuelist queue.
    - If there are jobs which have not been attempted to be auctioned then the name of the first job is retrieved and its jobfile is opened.
    - The current task information is found in the file and extracted.
    - The current bidding object file (Bidfile.bdf) is deleted from the network and a new copy is opened.

- The new bidding objects (see Table 3.1) are then written to the network bidding object file and the file is closed.
- A copy of the network bidding object file and the internodal travel time file are made for each node to avoid any clash.
- A GMT UMS is sent to every Cell Controllers Bidfrmstrt FIXDMACS database block prompting each Bid Formulation application to start formulating bids.
- The Bidtimer is activated to ensure that the bid evaluation process occurs in a timely manner.

6. The same procedure occurs for the Queuelist queue except that only one job is attempted to be auctioned from the queue before the token is released and this is from the top of the queue.

- ***Cell Controller - Assessing the Bidding Objects***

Once the bidding objects are transmitted a Message Response UMS informs the Bid Formulation application that it must retrieve the bidding objects and formulate a bid. This is carried out through the Bid Formulation application as shown in Figure 3.8.

When this application is loaded the following occurs:

- The name of the cell that the application is working on is retrieved from file and written into the Cell Controller name textbox.
- A Message Response UMS is then created between the FIXDMACS Bidfrmstrt database block and the Commence Task Bid textbox so that when the Cell is prompted to start formulating the bid it is able to start.
- The capacity and type codes for each machine are retrieved from the FIXDMACS database blocks (Cell Configuration Data Storage Addresses) and placed in the relevant Process Type and Capacity text boxes.
- The Commence Task Bid textbox is reset so that it can respond to a prompt from the Factory Controller to start bidding.

When the Factory Controller requests for the cell to start bidding the following occurs:

- The FIXDMACS bidfrmstrt database block is flagged and as a result a "1" is placed in the "Commence Task Bid" textbox (Due to the Message Response UMS).
- The bid file containing the bidding objects is opened from the network and the bidding objects are displayed within the relevant textboxes within the bidding objects frame.

- The task cost is calculated using the following formulae:

Taskcost = ((Batch Size)x (2 (Load/Unload Time)+ (Estimated Cycletime)))+Set up Time

The task cost is, therefore, the total time to carry out all the cycles of the task required for that job and as such represents a critical input into the EFT (Earliest Finishing Time) heuristic used.

- A file (Queucost.cst) on the local data storage service contains the volume of work (in a time sense) that has been scheduled to be carried out on each process. This file is opened and the contents of each array record is written into the relevant queuecost textbox on the form.
- Each process has a maximum volume of work that can sit in front of it (called an in queue buffer). This maximum buffer size is stipulated at the Factory Controller when the factory is configured. This file is opened and the contents are read into an array. For each process the volume of work in its in going queue is added to the Taskcost for the task being auctioned. If this figure is greater than the maximum buffer for the respective process, the process becomes ineligible to compete any further. In this way the system has the option to limit the amount of inventory in progress throughout the system. By having larger work buffers for bottlenecks and smaller work buffers for processes having higher throughput rates management techniques such as Goldratts "Theory of Constraints"[6] can be implemented.
- Next the application checks to make sure that all processes within the cell are active, if any process is inactive it is not eligible to submit a bid. To check whether a process is active the application examines the Process Type and Capacity textboxes. If any of these textboxes contain a zero the associated process is deemed to be in a disabled state and as such can no longer submit a bid.
- Once the application has checked to see what processes are active it then compares each process in the cell with the process required for the task. If a process of a given type and size within the cell matches the requirements of the task then that process remains eligible. Each process in a given cell has a given family of tools that it has at its disposal. This family is stipulated when the factory is configured and is in the form of a toolfile for every process on every cell.
- The next step in the bid formulation process is to check the tools available for each process against the tools required for the given task. If there is a match for any of the processes then the process concerned remains eligible providing that all other eligibility criteria have been met.

If all the criteria have been satisfied and a given process is therefore eligible to submit a bid a partial bid is formed by adding the overall taskcost for the task to the queuecost for the process being considered. In the formation of a cell it is possible that there may be a number of processes eligible to complete the task. In this case a partial bid is formed for each process deemed eligible. The bid that has been formed for each manufacturing process is displayed in the relevant Bidcost textbox as shown in Figure 3.8. The partial bid only contains information about a particular manufacturing process's

ability to carry out the task being auctioned. To allow for the required travel time of the physical parts from the current cell to the bidding cell this factor is required to be introduced into the equation. Therefore, there exists an intracellular bid factor (competition between devices within a cell) as well as an intercellular bid factor (competition between various cells having the best intracellular solution). As such the application then opens a file that exists on the network (and that was created in the Factory Configuration application) that defines the travel time between various cells within the system. This information is then read and
held in memory in the form of an array.

The Bid Formulation Application then carries out the following:

- It compares the bids on an intracellular level to find the optimum solution within the cell. As the system uses Earliest Finishing Time (EFT) as the optimum solution the smallest bid within the cell is selected.
- To produce the full bid on a global scale (for the extracellular solution) the travel time between the bidding cell and the cell currently housing the task to be worked on is added to the winning bid. This forms the full bid that is presented to the Factory Controller for evaluation.

Bidding Object	Meaning
Jobname	Name of the job being scheduled
Machinetype	Coded real number defining machine type required to carry out task.
EstCycletime	An estimation of the time required to carry out the task.
Set uptime	This is the estimated time required to set up the process/machine to carry out the task.
Loadunloadtime	This is the estimated time required to load/unload the workpiece from the machine.
NoofCycles	The number of cycles or batch size of the job being undertaken.
Tool1required	The tool required to carry out the task that is being auctioned.
Force	This a bidding flag that indicates whether or not the job is urgent.
AuctioningCell	The name of the cell that is conducting the auctioning process.

Table 3.1: - Hybrid Architecture bidding objects

- Next the application must write the details of the bid and associated manufacturing device details to a network file so that the Factory Controller is able to read it. There are three items that are written to this file - an item identifying the manufacturing process the bid is associated with, the magnitude of the bid and the busy flag. The busy flag is switched on when there are processes within the cell that are able to carry out the task but all have full job buffers (In going queue buffer exceeded). It is a method of flagging the Factory Controller to process this using a different heuristic.
- Once details of the winning bid and process are written to the cells network file the application sends a Global Message Transmit UMS to the Factory Controller FIXDMACS "C01Bidfin" database block (in the case that the bidding cell concerned has a nodename of C01). This action is a way of informing the Factory Controller that the given cell has finished the formation of the bid.
- In order to make sure that the application responds the next time it is instructed to carry out another bid formulation it resets the FIXDMACS "Bidfrmstrt" database block to zero using a Local Message Transmit UMS.

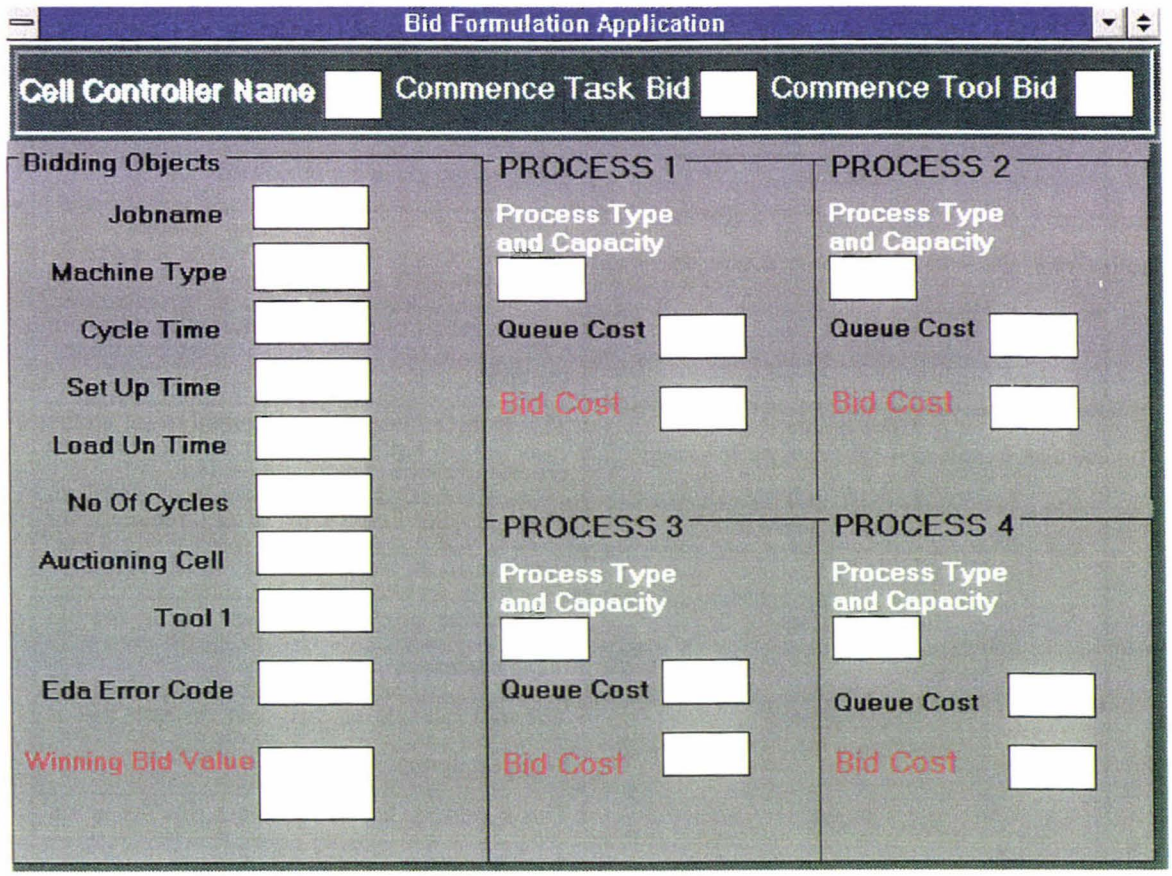


Figure 3.8 Cell Controller Bid Formulation Program

- ***Factory Controller - Assessing the Bids***

When the task data was announced a timer in the Factory Controller was activated to allow sufficient time delay for bid formulation at the cell level. Once this delay is complete a procedure for bid assessment is put in place, the details of which are as follows:

When the timer event is activated at the end of the delay period the following occurs:

- The Factory Based scheduling application uses a Message Request UMS to check the status of all FIXDMACS "Bidfin" database blocks thereby ensuring all bids are recieved. If all blocks have a value of "1" (Flag is on) then the application proceeds to compare bids. If all bids have not been recieved the application waits 10 secs and yields execution to Windows so that any outstanding Windows activities can be processed. It then rechecks all the database blocks again. The system continues this process until one of two things occurs:
    - ⇒ All flags have been received. The system then compares bids.
    - ⇒ A period of 20 seconds has elapsed. The system then decides there has been an error, displays a message "Contract Bid Protocol System Corrupted" and the bid assessment terminates. This type of error can be caused by a Cell Controller PC crashing for some known or unknown reason. A strategy for combatting this would be to develop a procedure within this assessment program (see future work, Chapter 7, Clause 5) to bypass the crashed Cell's bid and to consider only the bids that have been recieved within a certain time. This could be carried out by the following method:
        - ◊ After the elapsed time is up the Factory Controller could request the status of a known block in the subject cells FIXDMACS database and if an EDA error occurs (ie: Cell does not respond) this is a clear signal that the cell is not active. The system could then bypass the bid. This however will be the subject of future work.
- Next the application resets all the local FIXDMACS bidfin database blocks by sending a Local Message Transmit UMS. This enables the application to respond in the future as required.
- For each active node in the factory the bid information (The identification of the process bidding, the size of the bid, nodename and the busy status) is retrieved and stored in an array for reference.
- The system then carries out the following:
    - ⇒ It checks to see if there there are any valid positive bids for all nodes involved in the bidding process.
    - ⇒ It checks to see if there are any nodes that are able to carry out the task but have not put in a bid because their buffers are full (Have indicated this in the bid file).
- If the job is being auctioned off the main Factory Queue the application can handle different circumstances in the following ways:

⇒ If the smallest bid is zero and no manufacturing process in the system has a full buffer this means that no process in the system is able to carry out the task. This may occur in the circumstance whereby the user of the system has accidentally misdefined the task, or in the case whereby a machine capable of carrying out the task has become disabled. In this circumstance the system removes the job from the Factory Queue and places it on the "Rejected Jobs" list as shown in Figure 3.7. This gives the user a visual queue to the effect that this task is unable to be processed given the current state of the system. The changes to the queues are then saved to file and the Factory Controller sends a Local Message Transmit UMS to the FIXDMACS bidoperate block thereby releasing the bidtoken.

⇒ If the smallest bid is zero and there is a manufacturing process/es capable of carrying out the task but they have full buffers (This is indicated by setting a flag in the file to "on") the job is removed from the Factory Queue and if it is urgent (its jobfile is examined) the job is placed at the top of the "Processes Busy" queue so that it is re-auctioned as soon as possible. If it is not urgent it is removed from the Factory Queue and then placed at the bottom of the "Processes Busy" queue. After the queues are modified their new status is then saved to file and the bidtoken is released by sending a Local Message Transmit UMS to the FIXDMACS Bidoperate databse block.

⇒ If the smallest bid is greater than zero a message (0) is sent to the "Bidwin" FIXDMACS database block in the winning Cell using a Global Message Transmit UMS to reset the block. The number of the winning manufacturing process is then sent to the same block using a GMT UMS. The queueing application in the cell then responds by queueing the job in the relevant queue. The Factory based task scheduling application then removes the job off the Factory queue and saves the changes to file. Each manufacturing process has a textbox connected to the FIXDMACS "Bidwin" database block via a Message Response UMS. When a manufacturing process recognises its number in the database block it realises that it has won the auction and queues the job being auctioned. After it has queued the job the cell then sends a Global UMS to the FIXDMACS "Bidacknowl" database block which flags the "Bid Acknowledge" textbox telling it that it has acknowledged the award and as such the Factory Controller then releases the token by sending a null flag (0) using a Local Message Transmit UMS to its FIXDMACS Bidoperate block. This will allow the Node Polling Application to release the token after investigating this block's value. Once this is done the Factory Controller then resets the FIXDMACS "Bidacknowl" database to zero so that it is able to respond to the next flag.

- There is also the circumstance whereby the "Processes Busy" queue may have jobs queued in it. In this event a job in this queue will be the subject of the auction and as such will be the subject of the bids recieved. In this circumstance the "Factory Controller" processes the bids in the following way:

  ⇒ If the smallest bid is equal to zero and there are processes in the system that can do the current task (associated with the job) but *still* have full buffers the system attempts to try and

auction the next job in the busy queue by resetting the Bidtoken textbox to "0" and then to "1". This activates the change procedure in this textbox. The application tries to auction the next job in the queue by putting a task bidding object file together, informing the Cell Controllers that they must formulate bids and then awaits clarification that all bids have been formulated and then go through the bid comparison stage again. The application continues this process of attempting to auction jobs off the Busy Queue until it has successfully auctioned one off the queue or it has got to the end of the queue. If the application successfully auctions a job off the Processes Busy queue it releases the bidtoken via the Node Polling Application. If however, it cannot do this, (all buffers still full) it attempts to auction a *single job* off the main factory queue and regardless of whether it is sucessful or not it releases the token after the attempt.

⇒ If the smallest bid is greater than zero then the job is deleted from the relevant position in the busy queue and the contents of the new Busy Queue are saved to file. The winning node has its FIXDMACS "Bidwin" database block reset to zero and then changed to reflect the number of the winning manufacturing process using a GMT UMS. The queueing application (Manufacturing Support Service) associated with that manufacturing process then responds by queueing the job accordingly as has been already mentioned above. Once the winning cell's manufacturing process has queued the job it sends acknowledgement to the Factory Controller FIXDMACS "Bidacknowl" database block using a GMT Message Transmit UMS. As has been mentioned a "1" is transmitted to the Bid Acknowledge Textbox by the Messsage Response UMS. The Bidacknowledge Textbox then responds by setting the local FIXDMACS "Bidoperate" database block to zero (thereby releasing the token) and then resets the FIXDMACS "Bidacknowl" to zero so that it can respond once more in the future.

The algorithm (for full flowchart see Appendix A) is designed with the following principles in mind:

- Some jobs although urgent may not be able to be done on a particular device because of full buffers, however at each bidding opportunity these jobs must be given precedence.
- If urgent jobs are not able to be done due to full cell buffers other jobs must be given the opportunity to be auctioned. This is particularly important as if a single queue was used with a overall governing heuristic blocking could occur.
- Jobs must be queued in terms of urgency. Jobs that are urgent (ie: must have the quickest path through the system) are flagged as urgent in their jobfile and as such are always placed in the queue in such a way to ensure earliest opportunity for placement.
- The workload on all manufacturing processes of the same type must be balanced, this will ensure that process utilisation and, therefore, the mean time between completion is optimised.

3.3.3.2 Cell Controller Based Allocation

Cell Controller based auctioning of tasks is very similar to Factory Based scheduling. There are, however, a number of other components in the process that have a significant amount of involvement.

- ***The Auction Manager Application***

One of the most important factors that must be taken into account regarding Cell Controller based auctioning of tasks is the polling of the individual Manufacturing Processes within the Cell itself. We therefore have two mechanisms working - the Node Polling Application that is responsible for giving *each Cell* the opportunity to hold an auction (Global Polling) and the *Auction Manager application* that gives *each manufacturing process* within the cell an opportunity to hold an auction (Local Polling). The *Auction Manager application* receives the bid token from the node polling application and then passes it to each manufacturing process in turn to give each manufacturing process the opportunity to hold an auction. The main window of the Auction Manager Application is shown in Figure 3.9.

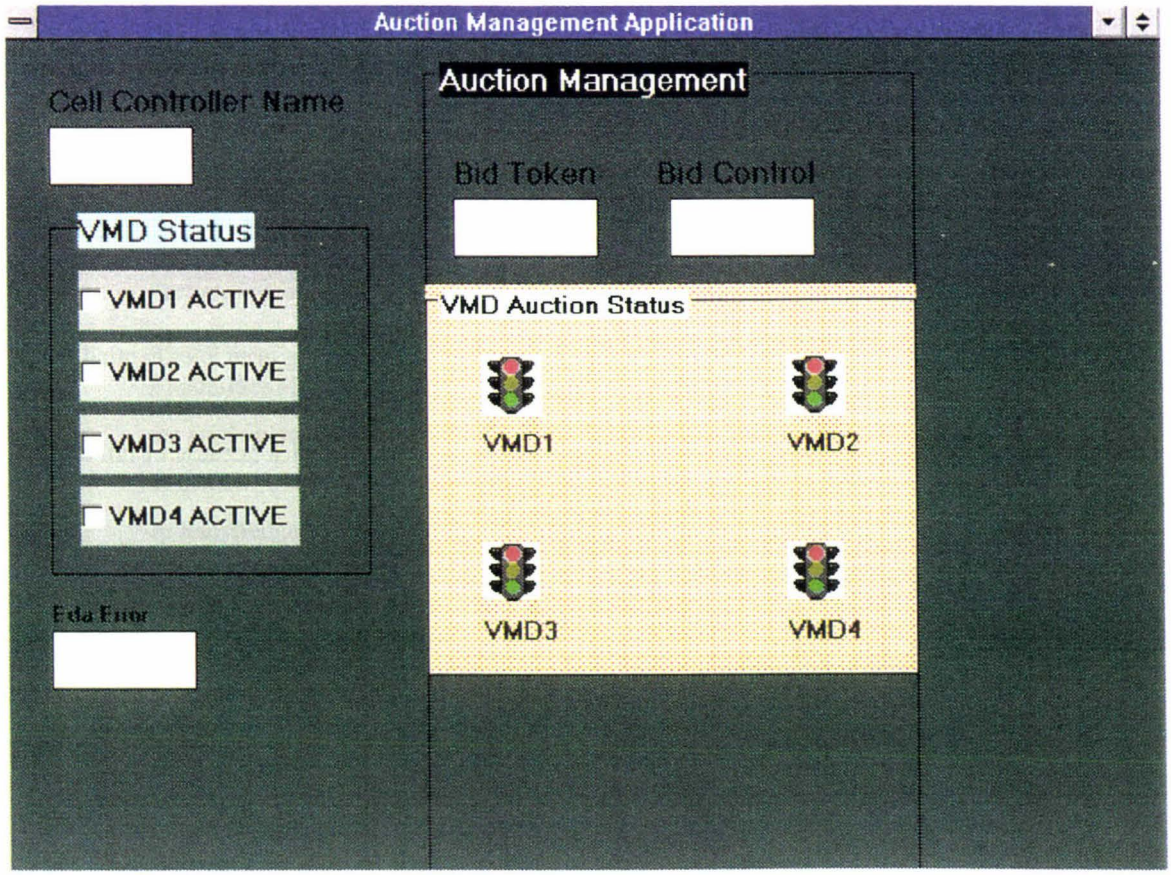


Figure 3.9 Auction Manager Application

On loading this application carries out the following activities:

- It uses a message request UMS to interrogate each FIXDMACS machine type database block (V1MACHTYPE, V2MACHTYPE...etc, etc) to find out whether or not each process within the cell is currently enabled. The operator may disable a manufacturing process by sending a "0" to the relevant block in a particular cell before physically disabling the process for maintenance for instance. This will inform the system of what manufacturing processes are eligible to hold an auction. A Message Response UMS is set up between the "Bid Token" textbox and the FIXDMACS "Bidtoken" database block. This allows the application the ability to respond immediately the Node Polling Application passes the token to the Cell concerned.
- A Message Response UMS is established between the FIXDMACS "VMDCurAuct" database block and the "Bid Control" textbox. This link allows the Auction Manager the ability to keep track of which manufacturing process is currently holding an auction and respond accordingly.

Once loaded the application is in the state in which it is able to respond to the deliverance of the bidtoken to the cell concerned. Once received the application responds in the following way:

- A message response UMS places a "1" in the Bidtoken textbox flagging the application to the effect that the token has arrived.
- The application then reserves the token by using a Local Message Transmit (LMT) UMS to place a "1" in the FIXDMACS "Bidoperate" database block.
- The application then uses a LMT UMS to send a "1" to the FIXDMACS "Vmdcurauct" database block. As a Message Response UMS exists between this block and the Manufacturing Process Services Queueing application this action flags the application telling it to attempt to auction a job off the top of the out going queue for the first manufacturing process.
- This action causes a change event in the Bidcontrol Textbox and causes the application to change the traffic light to green for the "VMD1" component (VMD1 being the Virtual Manufacturing Device for Manufacturing Process 1). This allows the user to have a visual display of what particular device is carrying out the auction at any given time.
- Once the Auction Manager Application has directed each Manufacturing Services Queueing Application (One for each manufacturing process)to hold an auction it releases the bidtoken by sending a LMT UMS to the FIXDMACS Bidoperate block (sets it to "0"). The FIXDMACS Vmdcurauct database block is also reset in this way.

- ***Role of the Manufacturing Process Services Queueing Application***

Once the Auction Manager sends a flag to the FIXDMACS "Vmdcurauct" database block the Cell Controller Queueing application can respond accordingly. This application contains two main windows.

The first window (Newqueue.frm) is responsible for the handling and manipulation of the queues, display of current queue cost and a number of similar roles. The second window (Queue_Status.frm - see Figure 3.10) is responsible for the display of the jobs in the in going and out going queues, assisting in the conducting of Cell based auctions and communication with the *Auction Manager Application.*

Once the Queue_Status.frm window is loaded two Message Response UMS are created. The first one is between the Bid Operate textbox and the FIXDMACS "Vmdcurauct" database block. This link is primarily used for the exchange of flags between the two applications. The second link is created between the Requeue Job textbox and is used for receiving instructions regarding the requeueing of jobs from the Factory Controller. During the loading process the out going queue file is opened and the contents of the file are written to the Out queue list.

As has been stated previously the *Auction Manager Application* places a "1" in the FIXDMACS "Vmdcurrauct" database block thus giving the opportunity to the first manufacturing process to hold an auction. As there exists a link between this database block and the Bid Operate textbox the value of the database block is conveyed to the textbox.

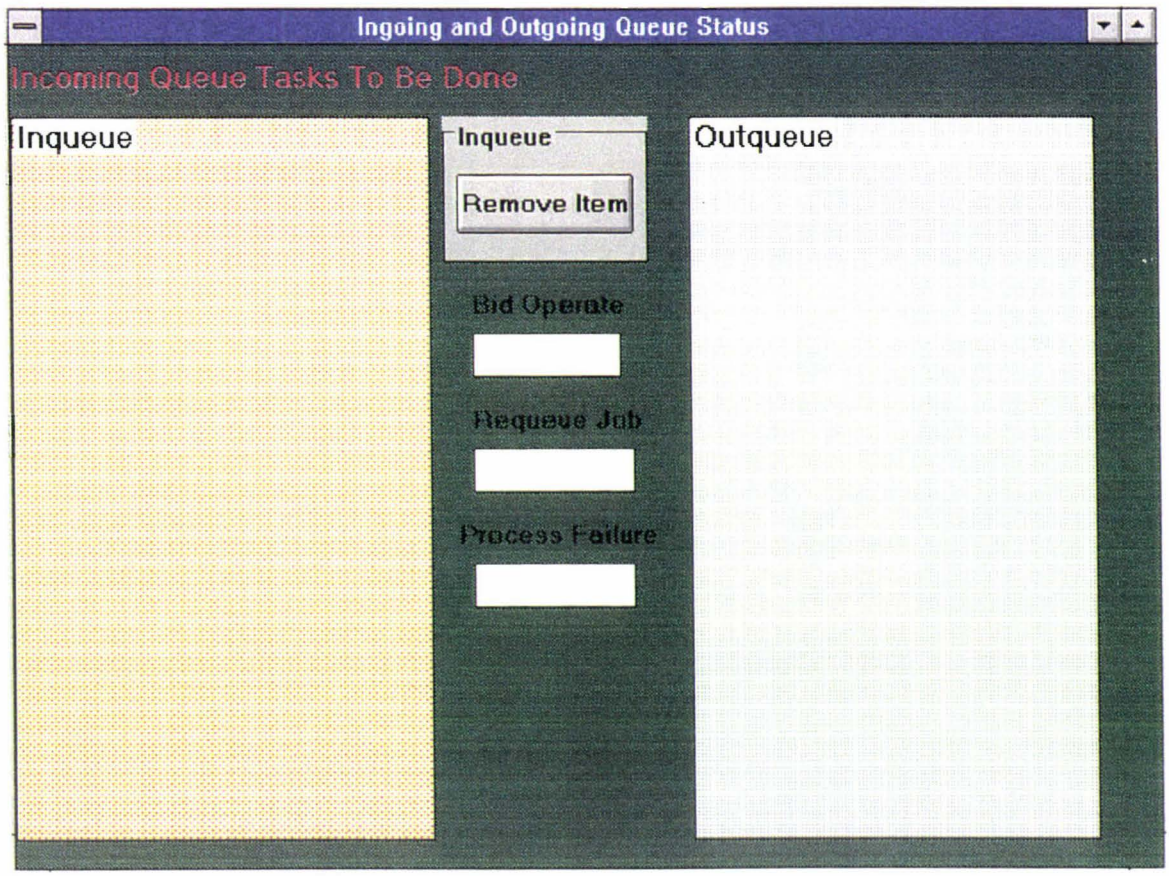


Figure 3.10 Queueing Application - In going and Out going Queue Status

This action results in the following:

- The Queueing application requests the active status of the current manufacturing process in question within the cell by using a Message Request UMS to determine whether it is eligible to hold an auction.
- If the manufacturing process for which the opportunity to auction is being given (in this case manufacturing process 1) is disabled then the value of the Bid Operate textbox is incremented by 1 and the new value transmitted to the FIXDMACS "Vmdcurauct" block using a Local Message Transmit UMS. This action has the following effects:
  - ⇒ The VMD1 traffic light is changed to red to reflect the fact that process 1 is no longer auctioning and also changes the VMD2 traffic light to green to illustrate to the user that manufacturing process 2 is now attempting to auction.
  - ⇒ The focus is changed to the next Manufacturing Processes Queueing application (associated with the next Queueing Application) and as such gives it the opportunity to hold an auction.

- If the manufacturing process being focused on is currently active the application then checks to see if there are any jobs needing to be auctioned in the manufacturing processes out going queue. If there aren't any jobs the value of the Bid Operate textbox is incremented by 1 and the new value transmitted to the FIXDMACS "Vmdcurauct" block using a Local Message Transmit UMS therefore polling the next manufacturing process.
- If jobs are available in the target manufacturing processes out going queue then the Queueing Application carries out the following:
  - ⇒ The name of the job on the top of the manufacturing processes out going queue is read.
  - ⇒ The job to be auctioned from the top of the queue is removed.
  - ⇒ The contents of the new out going queue (Out queue) are saved to file.
  - ⇒ The Auction Data file ("Netdrive\CellCont\Auction\Auctdata.txt") is opened and the name of the Job being auctioned, the name of the Cell carrying out the auction and the number of the process holding the auction is written to the file.
  - ⇒ The Queueing Application then prompts the Factory Controller by sending a "1" to its FIXDMACS Cellisauct database block. This prompts the Factory Controller to retrieve the job information from the network and carry out the Task Announcement process as stipulated in 3.3.3.1.

- ***Task Announcement with Cell Based Auctioning***

The method of task announcement with cell based auctioning is very similar to that employed in Factory based auctioning. There are however slight variations on the task announcement theme compared with factory based auctioning, this is explained below:

1. There exists a Message Response UMS link between the FIXDMACS "CellisAuct" database block and the "Cell is Auctioning" (See Figure 3.7) textbox in the Factory Based Task Scheduling application. When this database block is changed to a "1" (by the auctioning Cell) this change is immediately reflected in the "Cell is Auctioning" textbox in the Factory Controller. It is the change of this textbox, not the Bid Token textbox that begins the Task Announcement phase at the Factory Controller.
2. When the task is announced the Factory Based Scheduling application does not interrogate any queues as was the case with the Factory Based Scheduling - it simply opens the Auction Data file to get the name of the job being auctioned and distributes the bid data from there.
3. All other aspects of the Task Announcement are the same as for the Factory Controller based system.

- ***Bidding Object Assessment/Bid Assessment with Cell based Auctioning***

As discussed earlier in relation to Factory based auctioning it was stated that bidding object evaluation at the cell level was prompted by the transmission of a Global Transmit UMS to the Cell Controller FIXDMACS "Bidfrmstrt" database block. This too is the case with the Cell based auctioning system. As the structure of the task announcement is identical (the bidding objects are the same) the method of evaluation is identical, therefore the bid evaluation software is a purely generic piece of code.

Once the bids are put together by each cell and the Factory Controller is prompted by all cells signifying that all cells have completed the bid formulation process there exists a few differences in the Bid Assessment process. These differences are explained below:

- In Factory Based Auctioning the Bid Evaluation phase involves matching the name of the job currently being auctioned with that on Factory Queuelist given that the job originates from this list. This is necessary because theoretically a pending job could be added to that list during the auction process and the indexing of the job in that list would change. With the Cell based auctioning system the job is removed from the list prior to task announcement so this check is not required.
- If the job being auctioned is not able to have its current task placed because there are no processes able to carry out the task, the job is added to the Reject Job queue in the Factory Controller. The Factory Controller then gets the name of the Cell, the number of the manufacturing process holding the auction, and increments the number of the manufacturing process to indicate that the next manufacturing process in that cell should attempt to hold an auction. It sends this number to the cells FIXDMACS Vmdcurauct block using a GMT UMS and the Auction Manager and the next manufacturing processes Queueing application begins the cycle of holding an auction for the next process in the cell.
- If the job being auctioned is not able to have its current task placed because all eligible manufacturing processes have their buffers full then two separate events can occur depending on the jobs urgency:

⇒ If the job is urgent it is placed on the bottom of the Processes Busy queue, the modified queue is saved to file and the next process in the auctioning cell is given the opportunity to auction as mentioned above.

⇒ If the job is not urgent the name of the Cell from which the job originated and the name of the job are written to a network file (Netdrive & "Cellcont\Auction\Rejectjb.job"). The Factory Controller then sends a GMT UMS to the FIXDMACS "Requeuejob" database block detailing the number of the process out going queue that the job needs to be requeued on. A Message Response UMS exists between the Requeue Job textbox(Queueing application) and the "Requeuejob" database block. If the number that transpires in the Requeue Job textbox is the same as the number of the manufacturing process that the queueing application is associated with, the job to be requeued is placed on the bottom of the queue and the queue changes are saved to file. A Local Message Transmit UMS is then sent to reset the "Requeuejob" database block.

- When the Factory Controller is conducting an auction and it successfully auctions a job off one of its queues, the token is released after an acknowledgement is received from the winning cell, the Factory Controller then releases the token. In Cell based auctioning when a cells manufacturing process has won a task the next manufacturing process must be given the opportunity to hold an auction. Therefore when the auctioning is cell based and a manufacturing process wins the task being auctioned it sends an acknowledgement to the Factory Controller. Once the Factory Controller receives the acknowledgement it increments the number of the manufacturing process that has just been bidding and transmits the new number to the auctioning cells FIXDMACS "Vmdcurauct" database block thereby giving the next manufacturing process an opportunity to auction.

3.3.3.3 General Points Relating to the Task Scheduling Service

- At the Factory Controller level there exists a "Mean Time Between Completion" frame. This frame calculates the mean time between completion for jobs in the overall system for practical testing purposes (see Chapter 6).

The Factory Controller is able to record the results of both Task Auctioning and Tool Auctioning activities at the Factory Controller level (see Figure 3.11). The benefit of this is that it enables the user of the system to see exactly where the tasks have been allocated, when the tasks were allocated, what the size of the winning bid was for the process that won the task and in the case of the factory based auctioning effort what queue the task was auctioned off. This particular window can be accessed from the Factory based Task Scheduling Application (see Figure 3.8) window menu. This facility is even more useful when combined with the Manufacturing Process Event Form as shown in Figure 3.12. This form records the events occuring for each manufacturing process within a given cell

and is part of the Queueing application within the Manufacturing Process Services. The data displayed is real and originates from the testing of the system as outlined in Chapter 6. Each event for each task is clearly displayed on the form showing when the task was awarded to the process, the size of the queue cost(time to carry out tasks currently in the queue) before it was awarded (as well as after), the time that the task was started (and finished) and the average queuelength against time. The system also allows for all the data to be copied and pasted to Microsoft Excel™ for analysis. Originally DDE was used to integrate the system with Excel so the system could be easily analysed but it was found that the memory of some of the PC's used was insufficient. During the debugging phase this facility was very useful as it was easy to see where the overall system was faltering and how to rectify the situation.

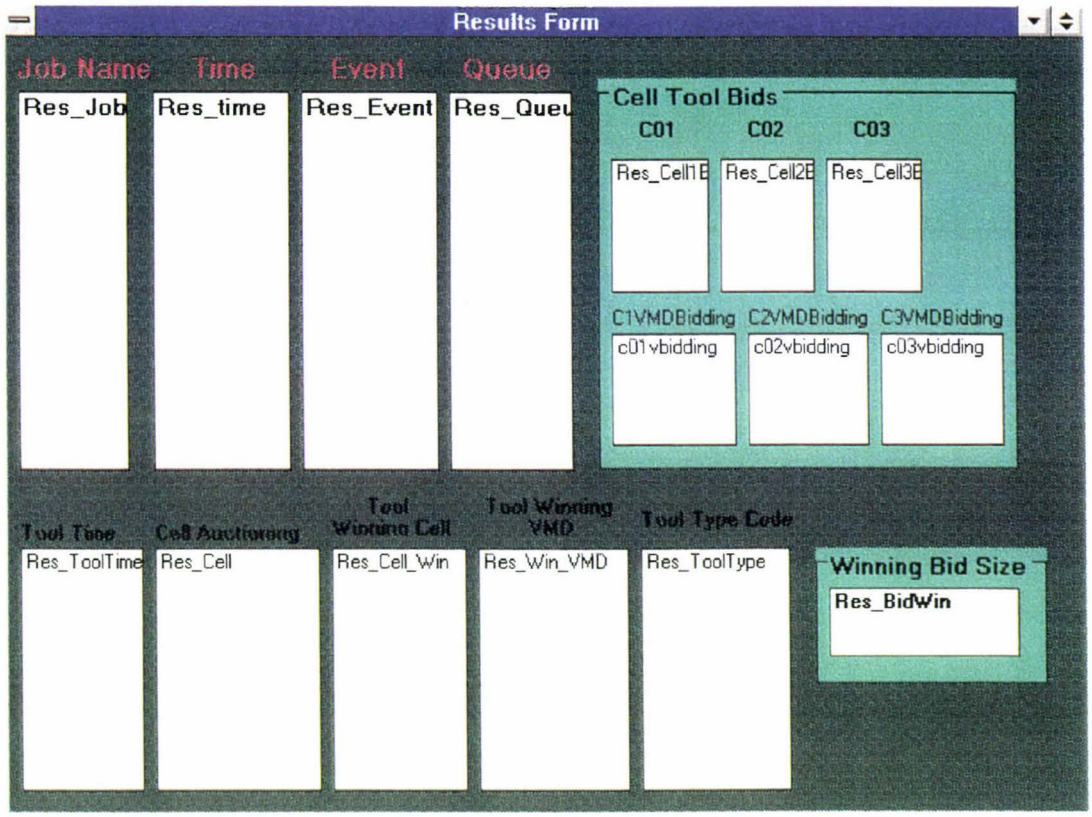


Figure 3.11 - Results Collection Form - Factory Controller

- If a manufacturing process fails or is taken out of service (eg: for maintenance)and the length of down time is significant then the jobs are taken off the processes in going queue and placed on the out going queue for auction. In this state the process will also be prevented from competing for any tasks being auctioned in the system. A Message Response UMS link exists between the Process Failure textbox (see Figure 3.10) and the FIXDMACS "V1machtyp" database block (eg: for process 1)so that the application may respond. When this database block is set to zero the textbox change routine carries out the transfer of tasks between the queues and updates the queue files. This database block can be

changed from either the Factory Controller (ie: taken out for planned maintenance) or from the Cell Controller when the process has become disabled. This can be carried out by the operator from the FIXDMACS View screen within the Cell Controller. If the system coordinator makes an assessment of the manufacturing process and decides that the process will only be temporarily disabled then he may choose to do nothing and as such the process will not operate but will continue to bid for tasks.

- The frequency of pending job scanning at the Factory Controller scheduling level as well as the auction timeout can be set within the Factory Controller Parameter Set up Service. This allows the system to be customised for the perceived loading on the system which can be dependent on the number of cells configured in the system as well as other factors such as network configuration.

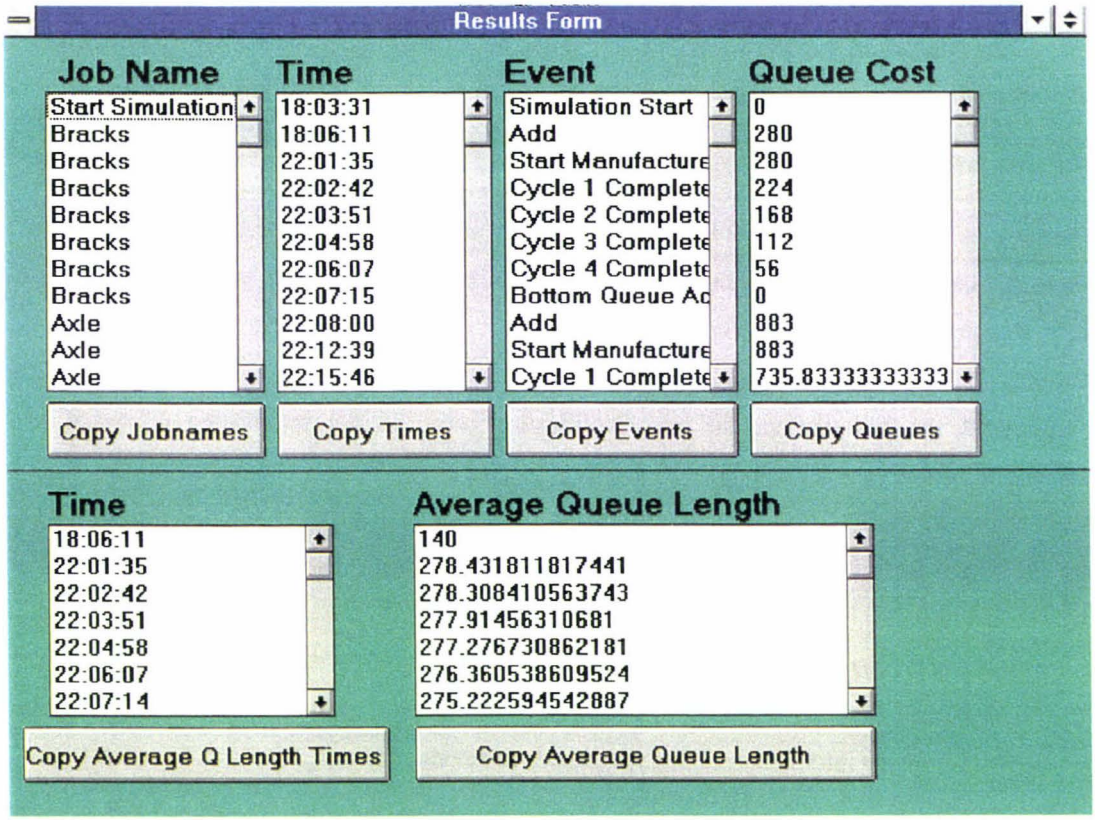


Figure 3.12 - Manufacturing Process Event form

There are many and varied aspects relating to the task scheduling service and many "what if" factors that have been encorporated into the system to allow for a wide range of scenarios. At this stage the benefits of the system have only been explained - these will however become more apparent in the physical testing of the system as outlined in Chapter 6.

3.3.4 Implementing the Resource Scheduling Service

The Resource Scheduling Service is responsible for the scheduling of resources in the Hybrid Architecture. These resources include items required by a particular machine type to complete a given task - such items may include machine tools, jigs & fixtures or even human beings. In the context of this thesis the Resource Scheduling Service will be explained in relation to the tools used within the environment.

As has been stated this service revolves around the scheduling of resources and makes the assumption that the resources have already been configured (see Resource Configuration Service). The main window that is accessed as a precursor to resource scheduling is the Main Factory Resource Scheduling Window as shown in Figure 3.13. The system has been set up to cope with as many Cell Controllers as required. All that would be required is to make a slight change to the contents of the "Tool Returns" frame so that the number of controls could be increased to suit the number of cells configured. This could easily be set up by creating a control array and duplicating the controls as required.

The window as shown in Figure 3.13 has a grid (solid grey area) which can be sized to hold the details of all tools that have been configured for the system. This "Tool Table" includes information such as tool type, type of process for which the tool is used , whether it is currently available or is being used, where the tool is currently located (it may be in the central tool carousel or at a cell), how much of the tool life has been used, the tool size and the tool identification code. All this information is critical in the scheduling of the tools and will be explained below:

When this form is loaded a number of actions occur as part of the load event:

- The Grid that is programmed into the system is configured to hold all the records as mentioned above in the form of a "Tool Table". The grid is sized to be seven columns wide and the top of each column is labeled to contain information of the type already mentioned above.
- The frequency of the tool auction (determines how often an auction will take place) is extracted from a local file ("C:\Factory\System\Config\Toolfreq.sys) and written to memory. This will determine how often a tool will be attempted to be scheduled through the auctioning system. Another file is opened ("C:\Factory\System\Config\TlActOut.sys") and the tool auction timeout is read from the file and written to memory. This file allows the system to know how long to wait before assessing the bids. These files were created via the Factory Controller Parameter Set up Service.
- Message Response UMS links are created between each Cell/Process Tool Return box and the FIXDMACS "C01V1TOOL" (in the case of the link for Cell 1 Process 1) database block. This allows the Cell Controller that has just finished using a tool to send a message to the Factory Controller to tell it that the tool is no longer being used and that it can be auctioned.

3.3.4.1 Initialising the Tool Table

The first thing the user must carry out is to initialise the Tool Table. This involves retrieving information about every type of tool that has been configured for the system, inserting this information into the table, setting each tool's state to idle and resetting the location as the Factory Controller. This activity must occur under two different scenarios:

- When the overall system has been configured and all tools are in the tool carousel at the main Factory Controller node.
- When the system is being reset and all tools moved back to the tool carousel at the main Factory Controller node for whatever reason.

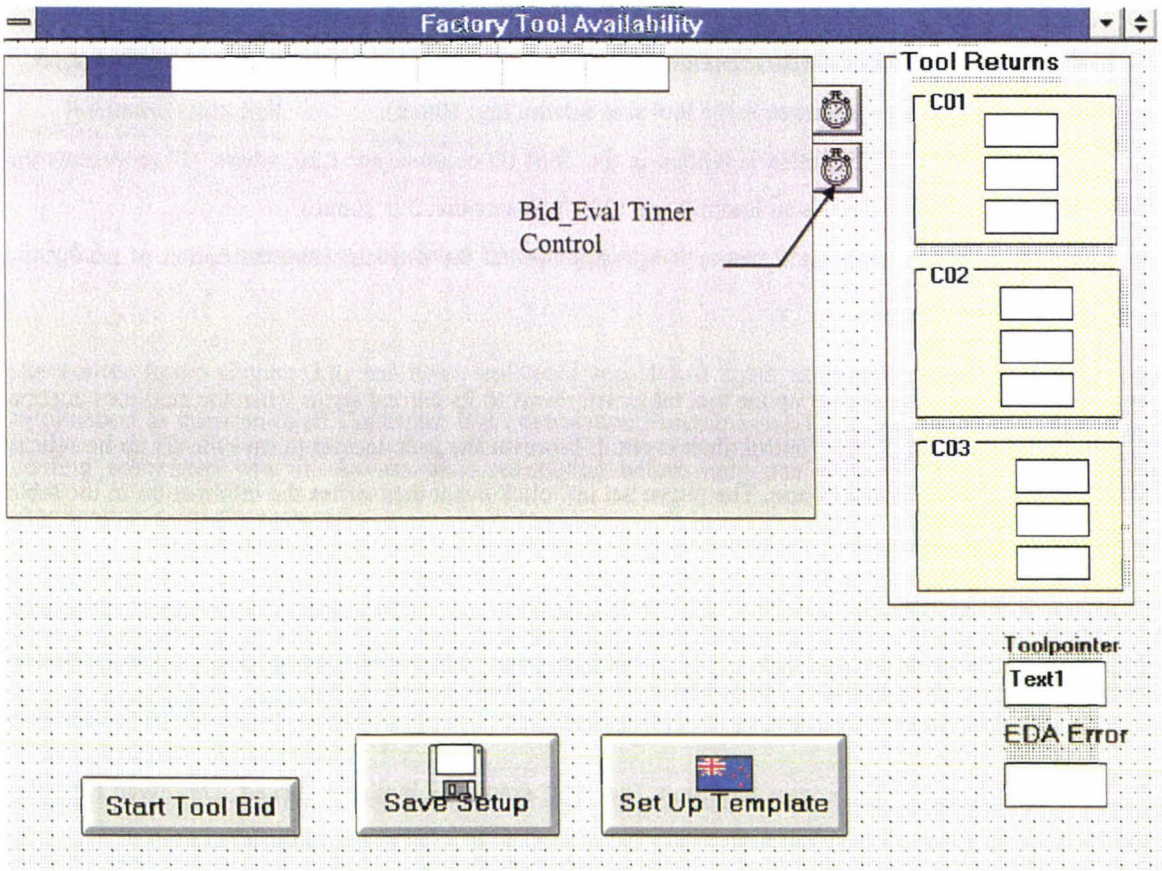


Figure 3.13 - Main Factory Resource Scheduling Window

As such the system needs to be able to be manually activated, as there will be the situation where it maybe shutdown and reactivated for whatever reason and the tool locations would not have changed.

To initialise the tool table the operator presses the Set Up Tool Table command button.

When this button is pressed the following occurs:

- The names of all of the tools that have been configured for the system are loaded into memory.
- For each tool type and size available the percentage of tool life used is read from file and the tool ID is derived by combining the tooltype and size into a three digit number. This makes the system able to recognise the tool as a definitive size and type.
- For each tool definition (each tool type & size has its own row) the following is added to the tool table in each column:
  - ⇒ Tool Name (eg: Endmill)
  - ⇒ Process Type (eg: Mill)
  - ⇒ The availabilty column is marked as "Idle" (ie: the tool is not being used).
  - ⇒ The location is set as "Fact" (ie: The tool is in the Factory Controller tool carousel).
  - ⇒ The percentage of tool life remaining(eg: 60%) is written in the percentage tool life column.
  - ⇒ The tool size is written in the tool size column (eg: 10mm).
  - ⇒ The tool identification is written in the Tool ID column (eg: 1.20, where "1" represents the fact that the Tool is an Endmill and "20" indicates that it is 20mm).
- The "Save Set up" command button is activated so that the user can have the option of saving the initialised tool table.

If the user does not save the set up the tool table will revert to its current status when the next tool auction is invoked by the Tool_Timer control timer event. If however the user decides to save the set up he selects the "Save Set up" command button. The "Save Set up" click event then writes the information in the table to a dynamic array and saves the information to the "C:\Factory\Config\Tools\Toolrecd.tl" file.

3.3.4.2 Initiating a Tool Auction (Tool Announcement)

To start the tool auctioning process the "Start Tool Bid" command button is clicked. This event activates the Tool_Timer timer control. When the Tool_Timer timer interval is up the following occurs:

- The timer control is deactivated to allow it to respond.
- The "C:\Factory\Config\Tools\Toolrecd.tl" file is opened and the tool information is read for each individual tool and size.
- The Tool Table has rows added to hold all the tool records.

- For each tool type the name of the tool, the process that is associated with that tool, the availability status of the tool, the current location of the tool, the % of tool life used and tool code is written to the first available row.
- When the Main Factory Resource Scheduling Window (see Figure 3.13) is loaded there exists a pointer that is reset to "1". This is done so that when the Tool Auctioning algorithm is activated the first row (first tool) in the tool table becomes the subject of the auction. Therefore the tool in the row that the pointer is indicating becomes the subject of the auction.
- The tool being auctioned has its availability status checked. If the tool is marked "BUSY" the Tool Pointer is incremented, the Tool_timer control is reactivated and the procedure is exited to allow the next tool to be auctioned.
- If the tool is marked "IDLE" the tool bid file is opened and the tool location and type code is written to the file by refering to the relevant columns in the current row. This bid file is replicated for each cell and written to the network server to avoid any potential file clash problems. The application then uses a GMT UMS to flag each cells FIXDMACS "Bidtoolstrt" database block. This allows the Bid Formulation Application in each Cell Controller to respond to the request to form a bid via a Message Response UMS link.

3.3.4.3 Assessment of Tool bidding Objects

An earlier figure (Figure 3.8) has been replicated and shown again as Figure 3.14 for clarity as the application in question (Cell Controller Bid Formulation Program) also has a bid formulation aspect for tooling associated with it. As has been mentioned before only one copy of the Bid Formulation Application exists for each cell.

When the application is loaded a Message Response UMS link is created between the Tool_Bid_Start textbox and the FIXDMACS "Bidtoolstrt" database block so that it can respond to requests made by the Factory Controller for tool bidding object assessments.

When the flag is sent from the Factory Controller a "1" appears in the Tool_Bid_Start textbox to signify that the application in each cell can begin the assessment of the tool bid file and check to see if the tool in the tool bid file is required by the task on the top of each manufacturing processes queue. This process occurs in the following way:

- The application first checks to see if any manufacturing processes within the cell are currently active. For each process that is currently active the application performs the following:

⇒ The in going queue file is opened for each process concerned and the name of the tasks on the top of each process queue is read.

⇒ Each task that is read off the top of the queue has its jobfile opened from the network and information vital to bid compilation is extracted. Such information includes the tool type code, the required completion date and the required completion time.

⇒ The file holding the bidding objects (R:\Factory\Bidding\Bids\"C01BdDat.bdf", for cell 1) is opened from the network and the tool bidding objects are read (Tool location & Tool type).

⇒ The application then attempts to match the tool code for the tool being auctioned with the tool code for the tool required by each task on the top of each queue.

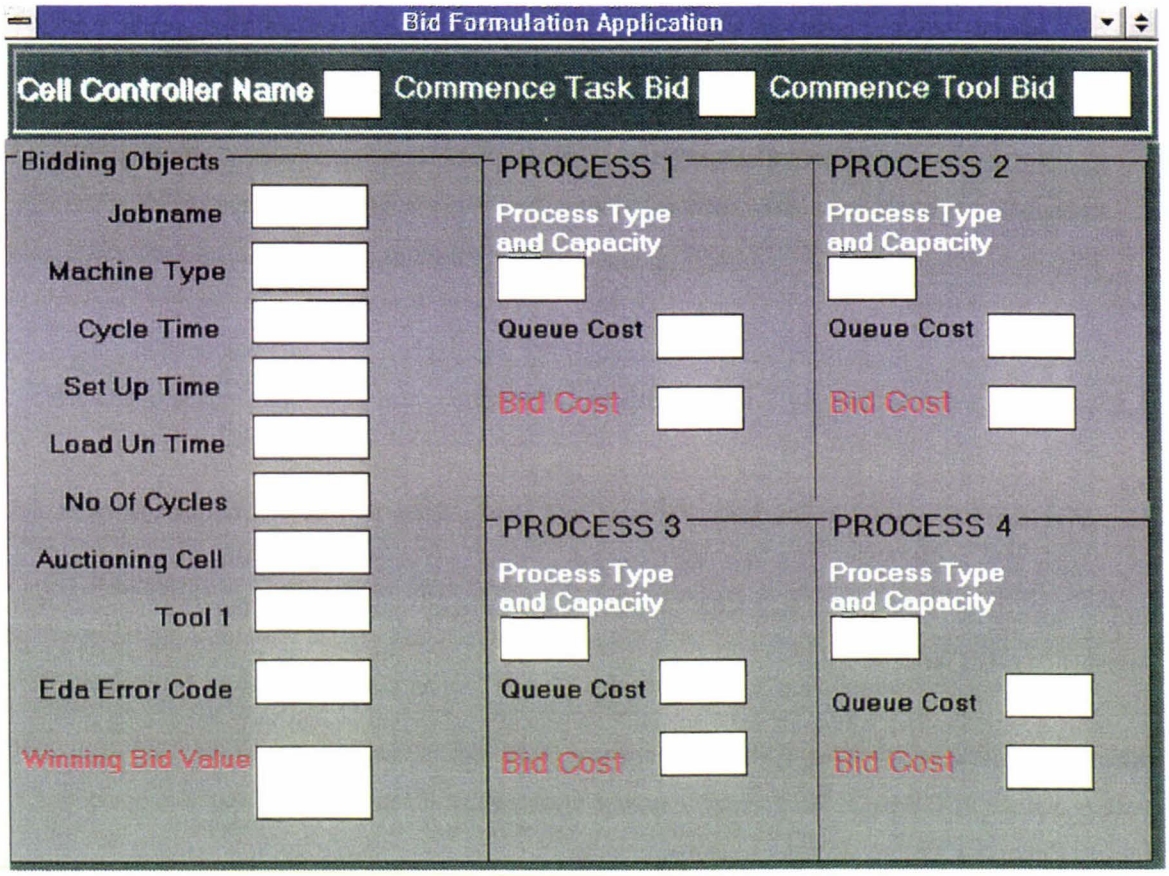


Figure 3.14 - Cell Controller Bid Formulation Program, Tool Bidding Aspect

⇒ If there is a match between the tool being auctioned and that required by a process to carry out its next task (ie: the tool required for the task on the top of its in going queue) a bid is formulated for that process. Once a match is formed between a process's tool requirements and that in the tool bid file a bid is compiled for that manufacturing process and only for that process within that cell. This is because the Hybrid Architecture is designed to have machines within its cells which are dissimilar

(not of the same type) and no tool will therefore be required by two separate processes. The bid is compiled for a process in the following way:

- ◊ The due date of the task in question is taken into account.
- ◊ The urgency of the task is taken into account by calculating the number of hours from the beginning of the year to the due date of the job. This figure becomes the basis of the bid and together with the number of the process bidding for the task is written to a cell specific bid file (ie: "R:\Cellcont\Toolbid\C01TLBid.Bid" for Cell 1).
- ◊ The FIXDMACS "Bdtoolstrt" database block is then reset by the application using a LMT UMS so that it can respond to the next request.
- ◊ A GMT UMS is then sent to the Factory Controller "C01Toolfin" FIXDMACS database block (When Cell 1 is bidding) telling it that the compilation phase is complete.

⇒ If there is no match between the tool requirements of any cell's manufacturing process in going queue tasks and the tool in the bid file then the following occurs:

- ◊ The cell specific bid file is opened and "0" is written to the file for the VMD bidding and an empty string is written for the bid.
- ◊ The "Bdtoolstrt" database block is reset as stated above and a GMT UMS message sent to the Factory Controller confirming that this cell has finished the compilation stage.

3.3.4.4 Assessment of the Bids

Once each Cell has completed the bid formulation process and has indicated such to the Factory Controller, the Resource Scheduling Service within the Factory Controller can assess the bids and award the tool.

After the Main Factory Resource Scheduling application has announced the tool bidding objects the Bid_Eval timer control is enabled (See Figure 3.13). This timer controls the length of time between tool announcement and an attempt to conduct a bid assessment. The length of time between tool announcement and bid assessment is defined in the Factory Controller Parameter Set up Service and can therefore be tuned to suit the size of the system.

When the Bid_Eval timer control interval has come to an end and the bid assessment routine is activated the following occurs at the Factory Controller level:

- The routine checks to see what Cells have been configured.
- The routine uses a Message Request UMS to determine what the value of the "C01TOOLFIN" database block is (in the case of cell 1). If its value is zero the system waits five seconds (and whilst waiting yields execution to windows to allow incoming data to be recieved) and tries again. It keeps performing this loop until the value of the block is "1" indicating that the cell has finished bid formulation. Once this occurs the system opens up the bid file for the cell (eg: "R:\Cellcont\Toolbid\C01TLBid.Bid" for Cell 1) and retrieves the value of the bid. The value of the bid, the cell name and the number of the process bidding is then stored in an array. This process repeats until all bids from all cells have been stored in the array.
- In the tool bidding algorithm there is allowance for the distance between the cell currently having ownership of the tool and the cells bidding (it would be illogical to send the tool to the other end of the job shop to a cell when the cell next to the one currently holding the tool has a job that is just as urgent - see Figure 3.15). Therefore the file that has all the information about intercellular travel times is opened and the contents are read to an array so that the information can be used for the final bid computation..
- Final bids are calculated for each cell by adding together the travel time between each bidding cell and the cell currently holding the tool to the bid for that cell.
- Details of each final bid for each cell are written to the relevant boxes in the Results Collection Form - Factory Controller (see Figure 3.11). During the programming this was used as a debugging tool to check that the tool was awarded as required. However in a practical sense it may also be useful.
- The total bids for each cell are compared and the smallest bid is deemed the winner. The cell is awarded the tool by executing the following actions:
    - ⇒ The tool is marked "BUSY" in the tool table to indicate that it has been awarded to a process.
    - ⇒ The Reserved cell for the tool in the tool table is marked with the name of the Cell that has won the auction.
    - ⇒ Details of the winning cell, the cell currently holding the tool, the winning process, tool type and the time the tool was awarded are written to the Results Collection Form - Factory Controller (see Figure 3.11).
    - ⇒ The changes to the tool table are saved to file.
    - ⇒ A GMT UMS is then sent to the FIXDMACS "V1toolresd" database block in the winning cell (given that process 1 won the tool) informing the process that it has won the tool and can therefore start processing the task when it is ready.
    - ⇒ The toolpointer is then incremented by 1 so that the next item in the Tool Table can be auctioned when the Tool_Timer timer event is reactivated.
    - ⇒ The Tool_Timer control is actuated so that the next tool can be auctioned at the next timer event.

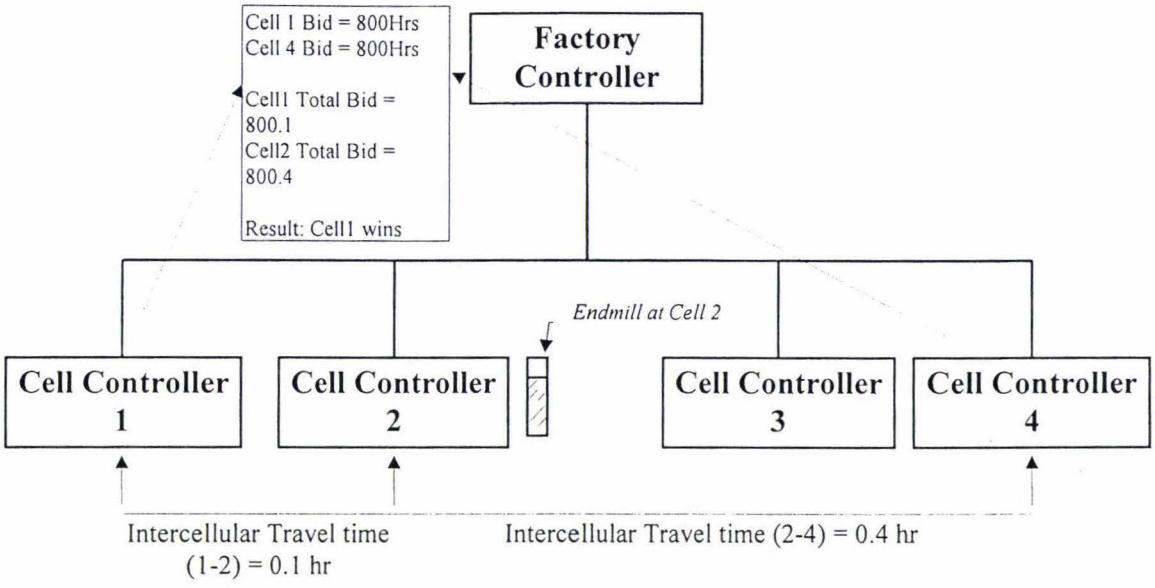


Figure 3.15 - Role of Intercellular Traveltime in the Resource Allocation Process

3.3.4.5 Return of Tools after use

Once a tool has been used by a manufacturing process it sends a message (toolcode) to the Factory Controller FIXDMACS "C01v1ret" database block (in the case of Cell 1 Manufacturing Process 1). This block has a Message Response UMS link with the respective "Tool Returns" textbox and when this message is transferred to the textbox the change routine within the textbox looks for the tool code/Cell in the Tool Table and when it finds the tool it marks it as "IDLE" therefore allowing the tool to be auctioned again.

There are some important and pertinent attributes that must be commented on as regards the system behaviour. As with all other services in the Hybrid FMS the Resource Scheduling Service is extremely robust. If for any reason the whole system crashes (as in the event of power failure) the overall system can be restarted and the Resource Scheduling Service will continue as though nothing has occured. This is because any changes to the system that are made (eg: Tool Table updates) are saved to a file and automatically reloaded upon system reboot as are all database settings. In the event of a system crash the user may wish to retain the system status or reset the tool table and start again depending on their requirements. With this system both of these options can be accomodated. The tool life factor has also been included in the table - it is envisaged that during the tool definition stage a tool life will be stated by the user (or arrived at algorithmically) for each tool. When the tool is used the amount of use will be stored as a percentage in the tool table. This will enable the user to peruse the tool lives in the table and replace the tools as required. It is important to state that in the Hybrid System transportation of tools

would be undertaken by the use of equipment such as an AGV (Automatically Guided Vehicle). This is why a standard Intercellular Travel Time is applied. The assumption is that tools would be transported by a given piece of transportation equipment at a given rate. This standard is therefore applied in respect of the Resource Allocation algorithm. In the context of this system the manner in which the tool is awarded would not strictly be the case in reality as in this system the tool details are changed in the Tool table and the tool awarded by sending a GMT UMS flag to the Cells FIXDMACS "V#toolresd" database block to simulate deliverance of the tool. In reality the AGV would visit the Factory Controller Tool Table, check what tools had been won by what cells and transport those tools to those cells as required. When a tool has been finished with there is no need for the AGV to carry out any process of checking the Tool Table - the manufacturing processes queueing application simply sends a message to the Factory Controller stating that it has finished with the tool and the Factory Controller Resource Scheduling Service marks the tool in the Tool Table as being idle. This allows the tool to be scheduled once more as has been described. When a tool has finished being used it remains with that cell until it is auctioned further - it is not returned to the tool carousel at the Factory Controller.

As is a feature with all Contract Net Protocol scheduling systems there is an individual goal that each party is trying to attain. The protocol uses a process of negotiation so that the best compromise is found. The goal of the bidding cell is to acquire a given resource for its process and the goal of the Factory Controller is to ensure that the most immediate task within the Hybrid FMS environment receives a given tool. A process of non-hierarchical negotiation exist between the two entities to ensure that the most favourable outcome is realised.

3.4 Summary

An attempt has been made to explain the implementation of the general services that involve communications throughout the entire Hybrid FMS. The general services are those activities that require intimate communication between the nodes in order to carry out the intended function of the particular service. As has been explained earlier the level of communication required for the implementation between nodes has been kept to a minimum so that high fault tolerance is achieved. The level of detail given regarding implementation is substantial as this allows an intimate knowledge of the general service function to be gained and also clearly defines the tools and methods used.

Chapter 4

Implementation of the Hybrid Model - Factory Controller Based Services

4.1 Implementation of the Hybrid FMS – Factory Based Services 126

4.2 Summary 161

4.1 Implementation of the Hybrid FMS - Factory Based Services

4.1.1 Implementation of the Factory Configuration Service

As was explained in Chapter 2 the Factory Configuration Service is responsible for the configuration of the overall Hybrid FMS system for a given application. The application is responsible for defining the physical layout of each Cell in the system. This definition includes such factors as:

- Defining the number of manufacturing processes within a cell.
- Defining the types of manufacturing processes within a cell.
- Defining the resources available to each process within a cell.
- Defining the distance between cells (in the context of AGV transit time).
- Setting up the activity status of each manufacturing process within a cell (whether the process is currently active and therefore eligible to participate in auctions).
- Definition of work in progress time buffer sizes for each process within a cell .

The implementation of this service is explained in the following section.

4.1.1.1 Defining New Nodes

The main front end for the Factory Controller is shown in Figure 4.1. From this particular window any factory based service can be accessed by selection of the relevant control button. To access the Factory Configuration Service the "Configure Factory" command button is pressed and this activates the main Factory Configuration window as shown in Figure 4.2.

- The load event causes an application to run which opens the current node file and reads all the names of the nodes currently configured (except the Factory Controller node name) and writes the names to the NodeNames listbox as shown. This allows the system to refer to the list when adding and removing nodes from the system and also allows the user to select nodes for reconfiguration as required.

- When adding new nodes to the system the user puts the name of the node in the "Add New Node Name Here" textbox and selects the "Add New Nodes" command button. This action performs the following:

  ⇒ Checks to see if the node name has already been used. If it has it does not allow the name to be added to the list.

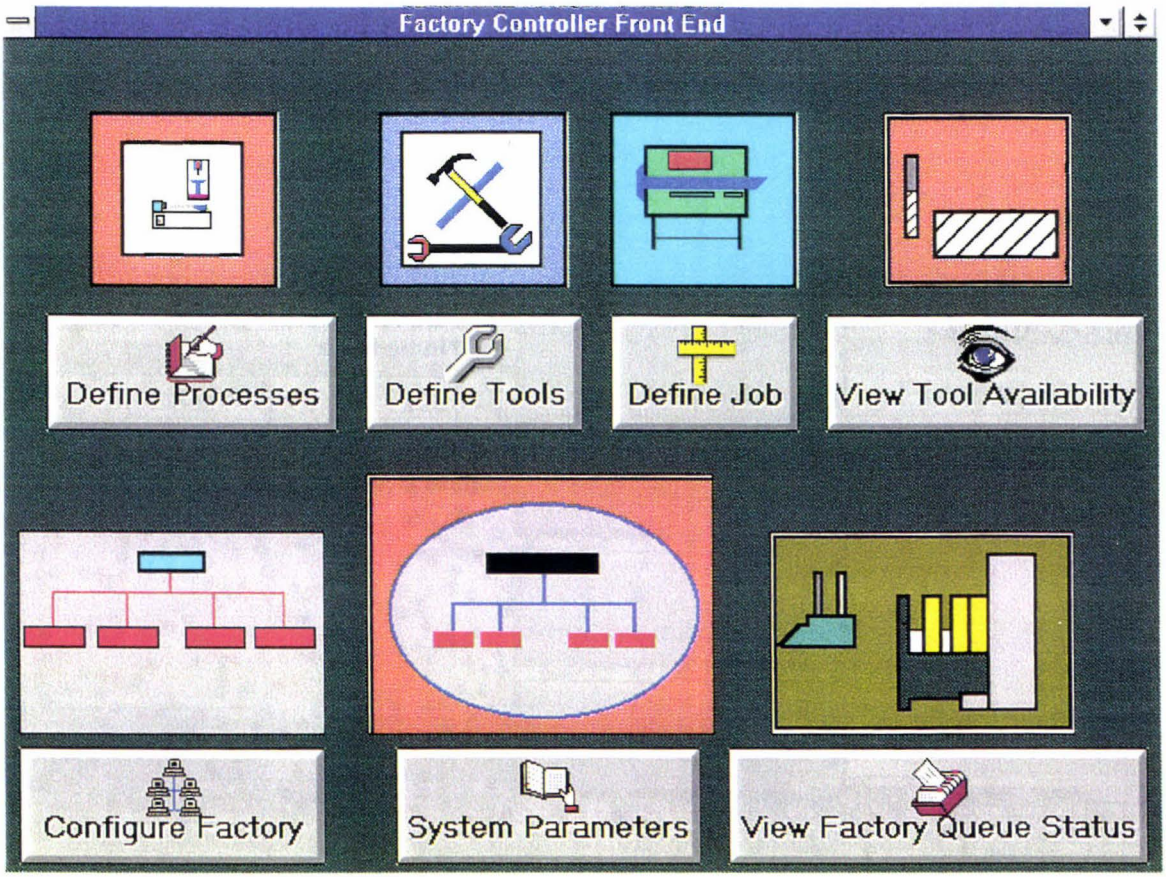


Figure 4.1 - Main Window for Factory Controller

⇒ Checks to see if the node name is of the form "C##" - if it is the name of the node is added to the list, otherwise a message is given to the user stating that the nodename is not valid and that another name should be used. The reason for this is that a set of standard FIXDMACS database blocks would be supplied with "C##" as part of their individual identification. This ensures that the nodenames are compatible with the FIXDMACS database blocks supplied with the system.

⇒ Deletes the old nodefile, opens a new one and writes the contents of the "Node Names" list to it.

⇒ Loads the Node_Def form to allow the user to configure each Manufacturing Process within the cell for which the name has just been stated.

4.1.1.2 Configuring Processes within the Newly Defined Node

When configuring individual manufacturing processes within a cell, the Node_Def form as shown in Figure 4.3, must be loaded. This can be carried out a number of ways:

- It is automatically loaded when a new node is added in the main Configure Factory Window.

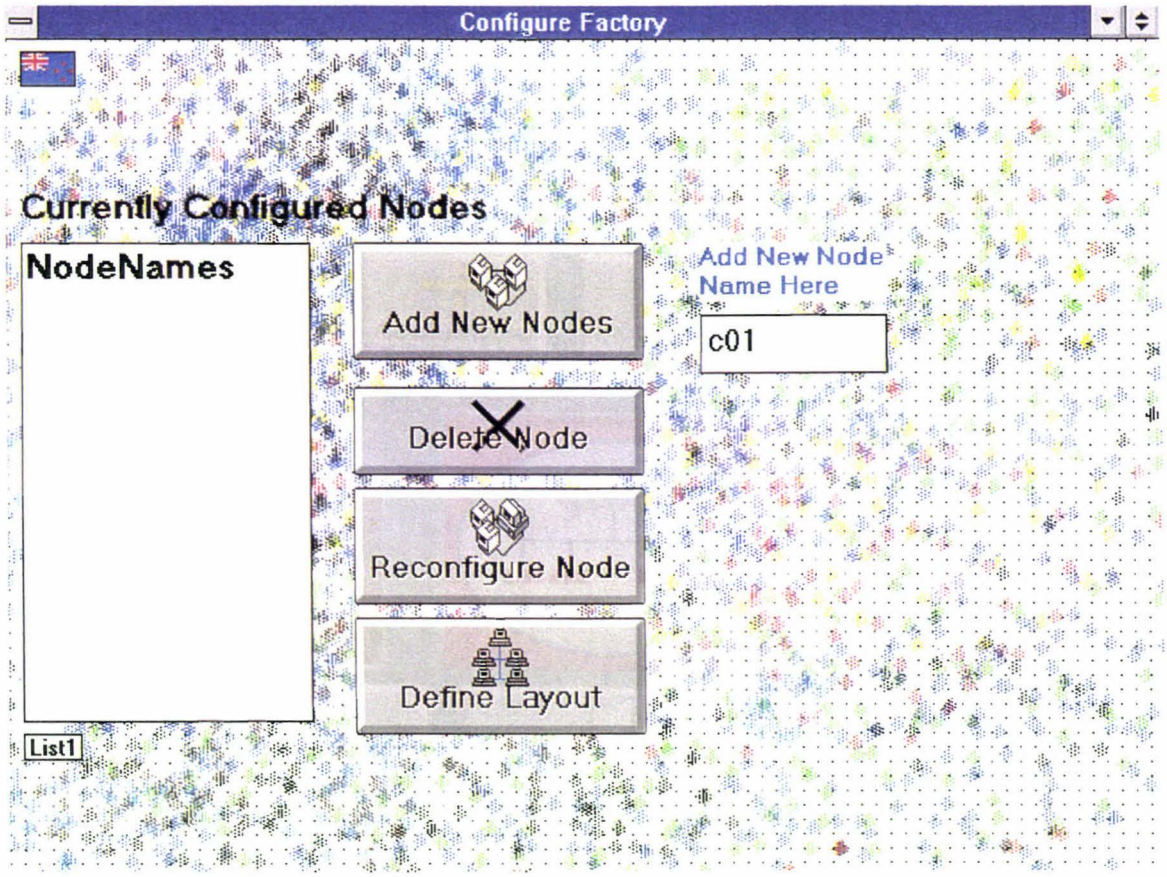


Figure 4.2 - Main Configure Factory Window

- By selecting an existing node that you want to reconfigure and pressing the "Reconfigure Node" command button.

Activities within this window are associated with the definition of the manufacturing processes within each cell. The term "manufacturing process" is used because it is a generic term describing an activity rather than using the term "machine" which implies that a physical device is always used to perform a certain process which is not always correct.

In the Resource Configuration Service there is a routine that deals with the definition of the manufacturing processes used within the system. Part of the definition involves the user drawing an illustration of the process using OLE (Object Linking and Embedding) to Microsoft Paintbrush™. The system has been configured so that when a process type has been assigned to a given manufacturing process number within a cell the bitmap that was created in paintbrush is loaded into the relevant manufacturing process frame as shown in Figure 4.3 when that form is loaded. This gives the user a visual queue as to what current process types have been assigned to what manufacturing process numbers within the cell. Therefore, when this form is loaded the load routine checks to see what process types have been defined for the manufacturing processes by reading each processes configuration file (for each process that has been configured) and then by loading the relevant Microsoft Paintbrush bitmap into the relevant frame as shown in Figure 4.3. If a frame does not have an illustration in it this means that the user has not yet defined a process for it.

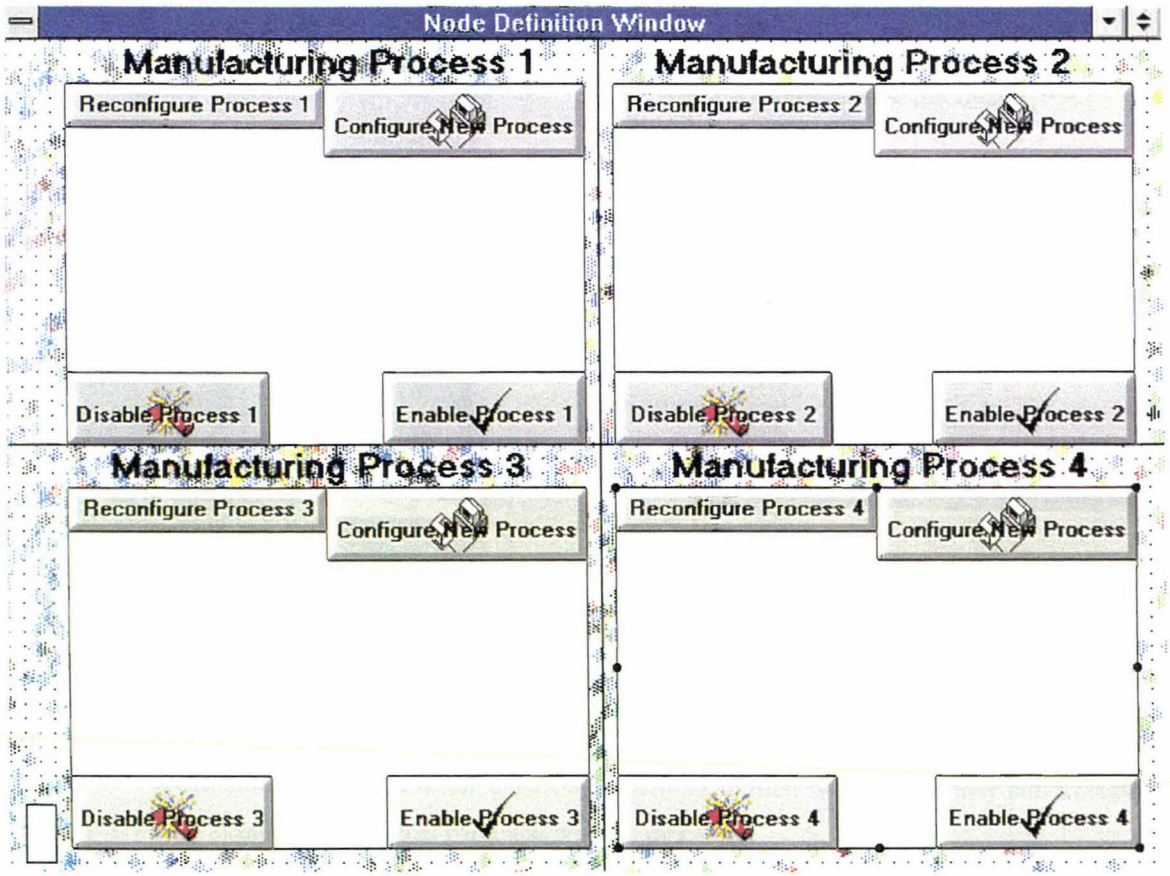


Figure 4.3 - Main Node Definition Window

Once this form has been loaded and the illustrations shown in the process frame, the user is then able to carry out the following activities:

- Reconfigure a currently configured manufacturing process by defining new tool types and sizes that are able to be used (and can therefore bid for), selecting capacity limitations from a range of options, assigning maximium in going task queue sizes and transmitting these data to the relevant FIXDMACS Cell Configuration Data Storage Addresses for reference by the Cell Control system.
- Definition of a totally new manufacturing process within a cell.
- Disabling and Enablement of various manufacturing processes within a given cell.

In order to configure a new manufacturing process for a cell the user selects the "Configure New Process" button for the number of the manufacturing process he wishes to set up for the cell. This event loads the form as shown in Figure 4.4.

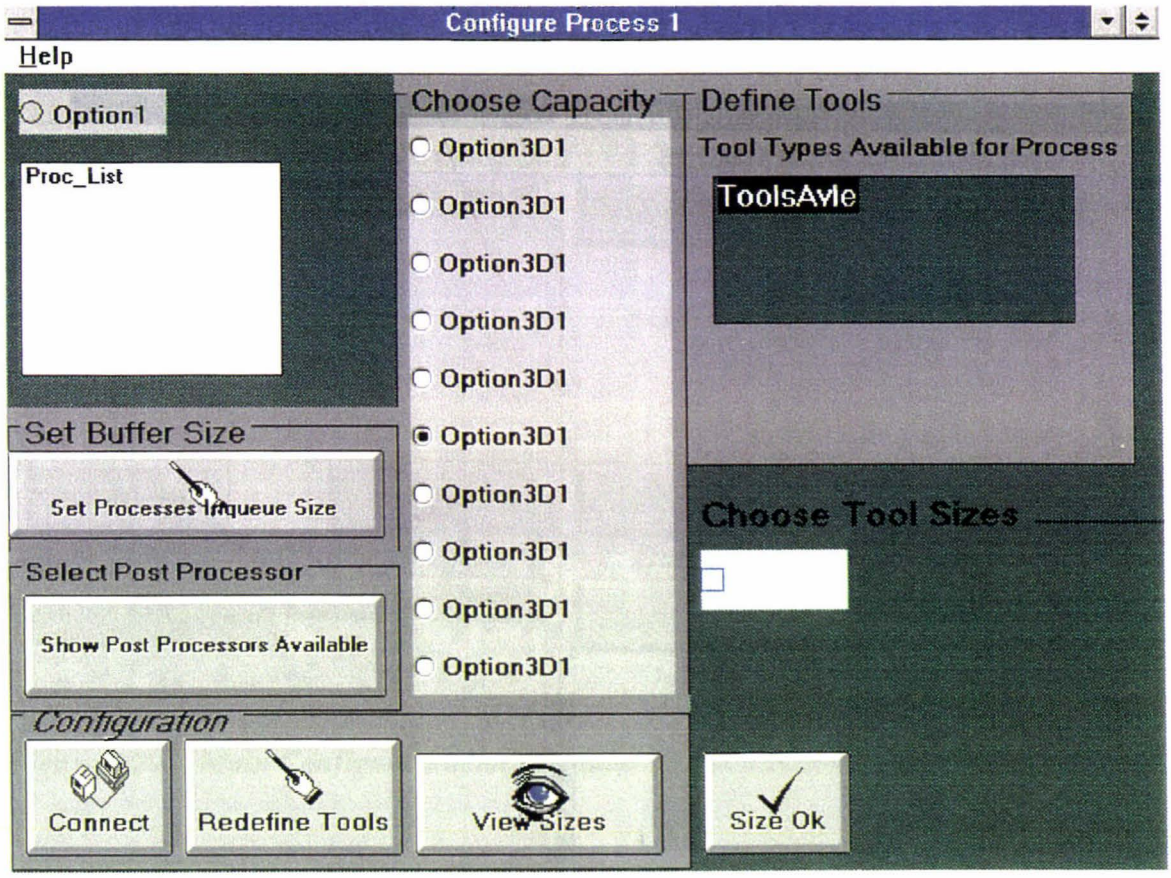


Figure 4.4 - Process Configuration Form

When this form is loaded a number of events occur:

- The file holding all the names of the configured processes is opened ("c:\Factory\Process\files\Proclist.Pfl") and the names of the processes read. The option button as shown is replicated and a process is assigned to each replication. This gives the user the option of selecting the process needed using option button format.

- Two files are opened ("C:\Factory\System\V1C1Mach.sys" & "C:\Factory\System\V1C1McCp.sys", in the case of Cell1, Process1) and the current process (if any) and capacity interval for that machine selected is read and the "Choose Capacity" and process option list modified to indicate that this configuration is the current selection for the manufacturing process. This will only occur however in the case where a manufacturing process has been configured before.

4.1.1.3 Defining a new Process/Capacity Interval for a Cell Controller

The first step in defining a new process for a Cell Controller is to define the process type that is going to be assigned to a given manufacturing process number and the capacity interval for that process. In reality the user would physically install the manufacturing process into the cell and then define the process type and capacity interval for the process using the Factory Configuration Service.

In order to define the process more comprehensively the user carries out the following actions:

- The user selects the name of the process that is being added from the first option list (see Figure 4.4). This action in turn carries out the following:

⇒ Deletes the current specification file for that Cell/Process (eg: "R:/Factory\Config\Cells\V1C1tSpc.txt". In the case where the cell is cell 1 and the process number is 1).

⇒ The file holding the details of the processes that have been defined ("c:\Factory\Process\files\Proclist.Pfl") is opened and a match is attempted between each process in the file and the process selected. Once a match is found the maximum capacity that has been defined for that process type is retrieved, divided by ten and each increment is displayed as the heading for each subsequent option button.

⇒ The system loads all the tool types available for that process type into the "Tool Types Available for Process" listbox so the user is able to see what tools can be chosen for that process. In reality some of the tools may be able to be used whilst others may not. Therefore only those suitable for that particular process are chosen.

⇒ Details of the process selected are then saved to a file (eg: "C:\Factory\System\V1C1Mach.sys" for Process 1, Cell1). This is so that this information can be automatically reloaded and the correct option box be automatically selected when the form is reloaded.

In order to define the process capacity interval the user carries out the following:

- Chooses the capacity interval from the option buttons in the "Choose Capacity" frame that the process being configured falls into. This is very important as it allows the cell controller in question the ability to decide what processes are eligible to bid in a capacity sense for a given task. Choosing a capacity from the list causes the following to occur:

  ⇒ The routine reads the value of the index selected and writes it to a ("C:\Factory\System\V1C1McCp.sys", for Process 1) local file. When the user selects the "Connect" command button (See Figure 4.4) the process code and the capacity index are converted into a multiple digit number and this number is sent to the relevant FIXDMACS database blocks in the Cell Configuration Data Storage Addresses in the relevant cell.

4.1.1.4 Defining Tools available for the process being configured

Generally speaking FMS systems normally have a central tool carousel that house a variety of tools for each different type of process available in the system. This however does not mean that every type of tool designed for use with a given type of process will be able to be used by every process of that type. For instance a lathe may have a toolpost that can carry a maximum 20mm tool holder whereas another may be able to cary a maximum tool holder size of 25mm. As such the user must choose for a given process the tools from the library that it is able to use.

To choose tools for the process being configured the user must carry out the following actions:

- Select from the "Tool Types Available for Process" list the tool type required to be configured for the process.

- Select the "View Sizes" button so that the sizes available for that tool type are displayed. This action causes the following to occur:

  ⇒ Opens a file for that process type and tool type and reads all the tool sizes available for the tool.

  ⇒ Creates and arranges enough checkboxes to hold all the tool sizes available and then writes those sizes next to the checkboxes so they can be selected.

- Select the checkboxes for those sizes available that are suited to the process being configured.
- Select the "Size OK" button. The selection of this button has the following reaction:

  ⇒ The Tool Identification number is derived for the tool type selected.

⇒ The tool specification file for that process and cell is opened from the network (in the case of process 1, Cell1 this will be "R:\Factory\Config\Cells\V1C1TSpc.txt") so that the tooltype and size code can be written to it and the tool identifications for all tools already configured if any can be read to memory.

⇒ The system then checks to see what has been selected in terms of tool sizes from the checkboxes and compares them to those already in the file. If a tool type & size is in the file but has not been checked then the tool is deleted from the file as this assumes the tool is no longer available. If the tool type and size is in the file and has been checked then the tool stays in the file. If the tool type and size has been checked and is not in the file it is added to the file.

⇒ The new tool specification fiile is saved to the network.

The process defined above is carried out a number of times for each tool type until a whole family of tools is defined for each process.

Tool definition for different processes is extremely important as it gives the Task Scheduling service the ability to differentiate from processes capable of carrying out a task to those that are incapable of carrying out a task by virtue of the tooling requirements. The circumstance may arise where a particular task requires a specialised tool type. If this occurs it would be illogical to schedule a task to run on a process that was physically unable to use this tool.

4.1.1.5 Viewing Sizes for Tool Types previously Configured

One factor that may be of interest to the user are the types and sizes of tools that have been included in a configuration for a given process. To accomplish this the user simply selects the tool type from the "Tool Types Available for Process" list and then selects the "View Sizes button. This action opens the current tool specification file for the process currently configured, makes available the required checkboxes and automatically shows the tools that have been chosen by selecting the appropriate check boxes.

4.1.1.6 Setting the Process Maximum In queue Size

When configuring a process for a particular cell the user may want to limit the amount of work in progress in its in going queue. This decision may be made at the strategic management level or as a result of a load management requirement. One particular instance where this maybe required is in Goldratts buffer

management theories.[6] Goldratt has stated as a result of production systems research that the bottleneck needs to be identified in a system and have a controlled time buffer of work in front of it. The idea is that the bottleneck must continue to function without stopping irrespective of what is occuring in the rest of the system as the throughput of the company is dependent on the throughput of the bottleneck. If the task queue in front of the bottleneck is too small and there is a breakdown on machines feeding it then the bottleneck may have to stop, therefore effecting productivity. If, however, a large maximum in queue size is set for the bottleneck and a process feeding it fails, the bottleneck will still have work to carry on with whilst the feeder process is being fixed. Also by carefully controlling the maximum in queue size J.I.T. management can be approximated. Therefore, the setting of a maximum in queue buffer size can be a very useful management tool. In the next chapter a description of a trial undertaken to show how this functions will be presented.

To set the buffer size for the process being configured, the user carries out the following:

- Select the "Set Processes In queue Size" command button, this action initiates the following:
  - ⇒ The user is asked to enter the maximum number of minutes of work allowed in the in going queue for that process.
  - ⇒ The user then either enters the number of minutes or the default value of 3 days work is applied.
  - ⇒ The buffer size stated for that process is then saved to a network file for that particular cell ("R:\Cellcont\queue\"C1Bfsze.Buf" for cell 1) so that it can be referenced by the Task Scheduling Service in the cell at bid compilation time.

If, at bid compilation time, a given process is able to carry out the job being auctioned but has a full buffer, the cell housing that process will submit a null bid, absolving it from any participation in the auction process. The maximum queue size setting process therefore becomes a very effective tool.

4.1.1.7 Defining the Post Processor available for a Manufacturing Process

Within the Hybrid Architecture there exists a Controller Integration Algorithm (See Figure 4.10). For this algorithm to function correctly the type of encoding that must be given to a generic toolpath file before transmission to a given manufacturing processes NC controller must be known. Therefore when a manufacturing process is configured for a given cell the user must select from a list of Post Processors (changes toolpath file to NC code) one that is compatible with the type of controller for that manufacturing process.

To view the currently available Post Processors the user selects the "Show Post Processors Available" command button. This action causes the "Proc_List" listbox to become active and shows all the Post Processors available for that process type. The user then selects the type of post processor that needs to be used to satisfy the requirements of the manufacturing processes NC controller. Once selected the name of the Post Processor is transmitted to the appropriate Cell Configuration Data Storage Address and is also saved to file with the other configuration details.

4.1.1.8 Transmitting the Configuration for a Process

Once the process has been configured it becomes a matter of being able to transmit this information to the target cell. More complicated information such as tool types, buffer sizes etc have all ready been placed on the network in file form and as such are available to be used by the target node for activities such as task scheduling.

In order to finalise the configuration for a particular manufacturing process within a cell the user should select the "Connect" command button. This carries out the following actions:

- Uses a GMT UMS to transmit the process and capacity code to the target cells FIXDMACS "V1MACHTYP" (in the case of process 1) database block. This particular code will be refered to during the task & resource scheduling phase.
- Uses a GMT UMS to transmit the Post Processor details to the target cells FIXDMACS "V1CNCTYPE" (in the case of process 1) database block. These details will be refered to during the task scheduling phase.
- Registers the node as having been configured to some extent by registering it in the "C:\Factory\Config\Cells\Whatnode.sys" file. This is very important as when the Factory Controller starts up it refreshes the configuration settings each time and does this by looking at this particular file so it knows where to send the required information.
- Stores the configuration parameters (Process Number, Process Code, Active Status and Post Processor Details) to the "C:\Factory\Config\Cells\C1Confg.sys" (For Cell 1) file.

This storing of information in the files is very important as it allows the system to investigate what cells have been configured, to get the configuration settings involved and then to transmit them each time the system starts up.

When the Process Configuration form is exited (see Figure 4.4) the Main Node Definition window is reloaded (see Figure 4.3) and an illustration of the process configured for the associated process number is

is loaded into the appropriate frame. This allows the user the luxury of being able to physically see exactly what processes are configured for a given node at any time in one instant.

4.1.1.9 Defining the Layout of Cells within the Factory Controller

One of the most important aspects in the Factory Configuration Service is the definition of the layout of the cells within the manufacturing environment. This is particularly important consideration when taken in the context of the distribution of tasks and resources. The definition of cell layout is concerned with the stating of inter cellular travel times for devices such as AGV's as such devices will be responsible for the transportation of resources and items associated with certain tasks.

The form associated with layout definition is shown in Figure 4.5 and is accessed by the selection of the "Define Layout" button in the Main Configure Factory Window (see Figure 4.2).

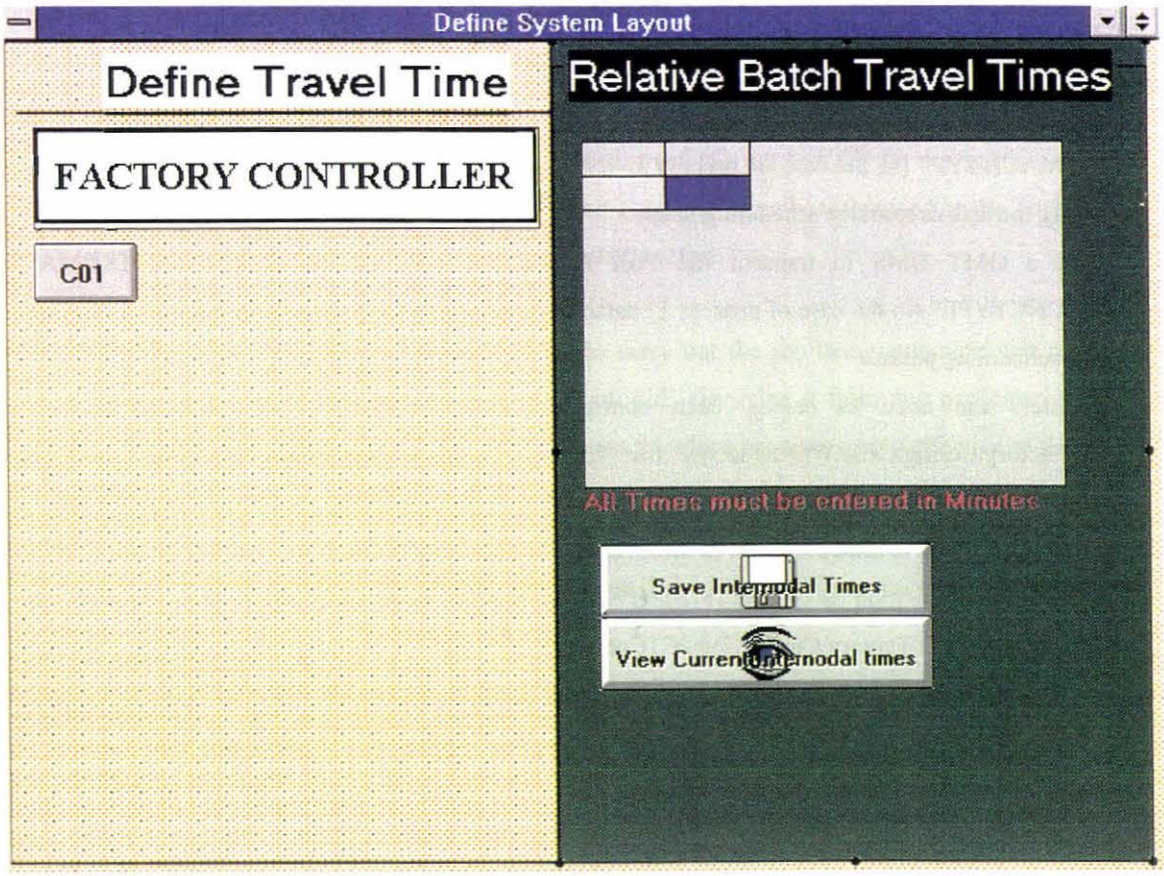


Figure 4.5 - Cell Layout Definition Form

Selecting this button carries out the following:

- The names of the nodes configured are read from the "Currently Configured Nodes" listbox (Figure 4.2) and stores them in an array.
- The Define_Layout form is loaded (see Figure 4.5).

A number of actions occur in response to the load event:

- The "C01" button is replicated so that an instance of this button exists for every configured cell. Internodal distances are defined by selecting one button (Representing one cell) and then another button (Representing another cell) and then defining an internodal travelling time.
- The "R:\Factory\Bidding\Nodetime\NodeDely.txt" (holds all current intercellular distances) is opened.
- The "Relative Batch Times" grid to hold all current intercellular travel time records is organised and then the current intercellular travel times are written in the respective grids as required (see below).

	Cell 1	Cell 2	Cell 3
Cell 1	0	5	10
Cell 2	5	0	20
Cell 3	10	20	0

- The "R:\Factory\Bidding\Nodetime\NodeDely.txt" file is closed when all records have been read.

4.1.1.10 Defining Intercellular Travel Times

To define intercellular travel times the appropriate cell command buttons are selected one after the other and an estimated intercellular travel time is stated by the user. If defining the travel time between the Factory Controller node and a given cell, the "Factory Controller" illustration is selected followed by the cell command button. Selecting the first button eg: "C01" carries out the following:

- The name of the cell selected is recorded via a static variable.

The user then selects the second command button for the second cell and specifies an internodal travel time. Selecting the second button eg: "C02" carries out the following:

- The first cell name and the second cell name are written to memory.
- The user is then asked through a message to input a value for the internodal traveltime. If no time is input then the default internodal travel time of 5 minutes is applied.

- A check is performed to make sure that the user has not selected the same button twice. If he has, he is informed of the problem and asked to select an alternative.
- The appropriate row/column combinations are found and then the internodal time is added. As the cell internodal distances are written in matrix form two records are written in the appropriate grids.

This information is currently added manually but future versions of the program would ideally include algorithms capable of averaging out successive journeys of the AGV (or transportation device used) between cells to give a moving average. This information could then be used to update the Internodal travel time file.

4.1.1.11 Saving Intercellular Travel Times

Once the intercellular travel time information is defined it is required to be saved. To carry this out the user selects the "Save Internodal Times" command button. This action carries out the following:

- The information is written to a two dimensional array.
- The "R:\Factory\Bidding\Nodetime\NodeDely.txt" is opened for write access and the array contents are written to the file before it is closed.

4.1.2 Implementation of the Factory Controller Parameter Setup Service

The Factory Controller Parameter Service is responsible for providing values for certain customisable settings within the factory environment. During the research process, as well as the testing of the system, it became apparent that there were some parameters that would need to be finely tuned to ensure that the system performed adequately. An example of this would be in the Task Auction Timeout. As the system gets larger this may be required to change. There were three other parameters that were deemed necessary to have customisable attributes:

- **Jobs Pending Scan Time** - As the system gets larger (more cells) the throughput of work goes up substantially, as the average start time difference between pending jobs becomes closer. If the Jobs Pending Scan Time is much greater than the average start time then jobs will become late. Therefore it is important to be able to change this.

- **Tool Auction Frequency** - If the frequency is not set correctly for the system configured, tasks could get held up in manufacturing process in going queues.

- **Tool Auction Timeout** - as the system becomes larger response times from cells bidding for tasks may increase. As such it is importanat that this feature can be adjusted.

In order to access this service the "System Parameters" control button is selected from the Main Factory Controller Window (See Figure 4.1). This action then loads the Configure System Parameters Form as shown in Figure 4.6.

In order to set these times the user simply presses the appropriate button and enters the time as requested by the system. For Task Auction timeout this time is specified in seconds. For Jobs Pending Scan Time the time is specified in multiples of 10 seconds, for Tool Auction Frequency the time is specified in seconds and for Tool Auction Time the interval is specified in seconds. All this information is saved to appropriate files within the Local Data Storage Service.

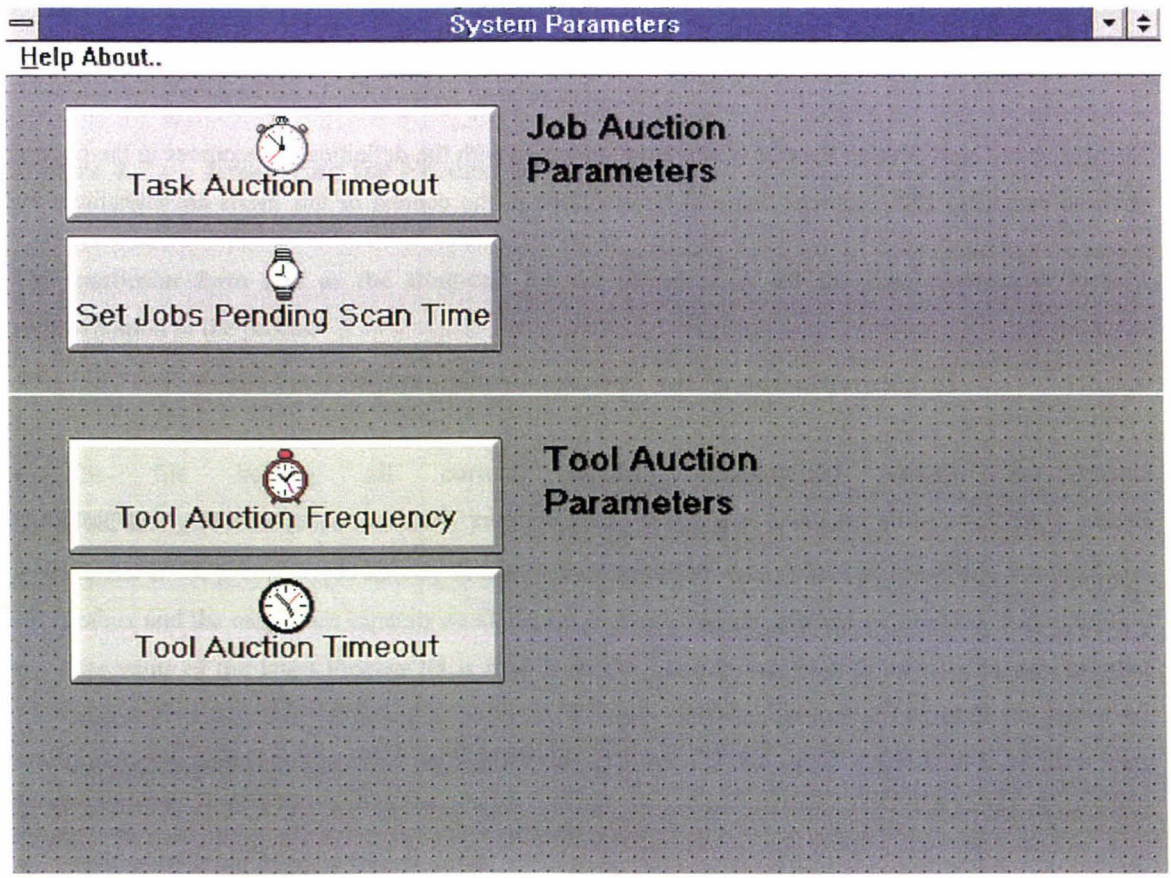


Figure 4.6 - Configure System Parameters Form

4.1.3 Implementation of the Resource Configuration Service

Without resources no manufacturing environment regardless of architecture or organisation is able to function. However, before resources are able to be assigned to various parts of a manufacturing environment they must be defined as a data set that can be interpreted and handled by the system. On a physical level, resources can exist in the factory environment. However, these resources must be represented by data sets that the system is able to recognise and handle in a consistent and logical fashion.

The data set that is used to identify a given tool in the context of tool definition for instance is:

- Tool Type - a physical description of the type of tool being defined.
- ToolId - a 2 digit number describing the tool in a form that is recognisable by the system logic. In the case of this system the first digit represents the tool type, the second digit represents the size of the tool.

Therefore, once a particular resource has been generically defined by the user in the form of a data set that type of resource can be handled and used by the system in a consistent fashion due to its consistent form.

The Resource Configuration Service is, therefore, involved with the definition of resources in the context of these data sets. The resources that will be discussed in the context of this thesis are manufacturing processes and tools.

4.1.3.1 Implementation of the Process Configuration Service

Before a process can be assigned to a given Cell Controller the data set representing that particular process must be defined in such a way that the system is able to recognise each process in a consistent fashion.

To initiate the Process Configuration Service the "Define Processes" command button is selected in the main window of the Factory Controller (Figure 4.1), this action then loads up the Define_Processes form as shown in Figure 4.7.

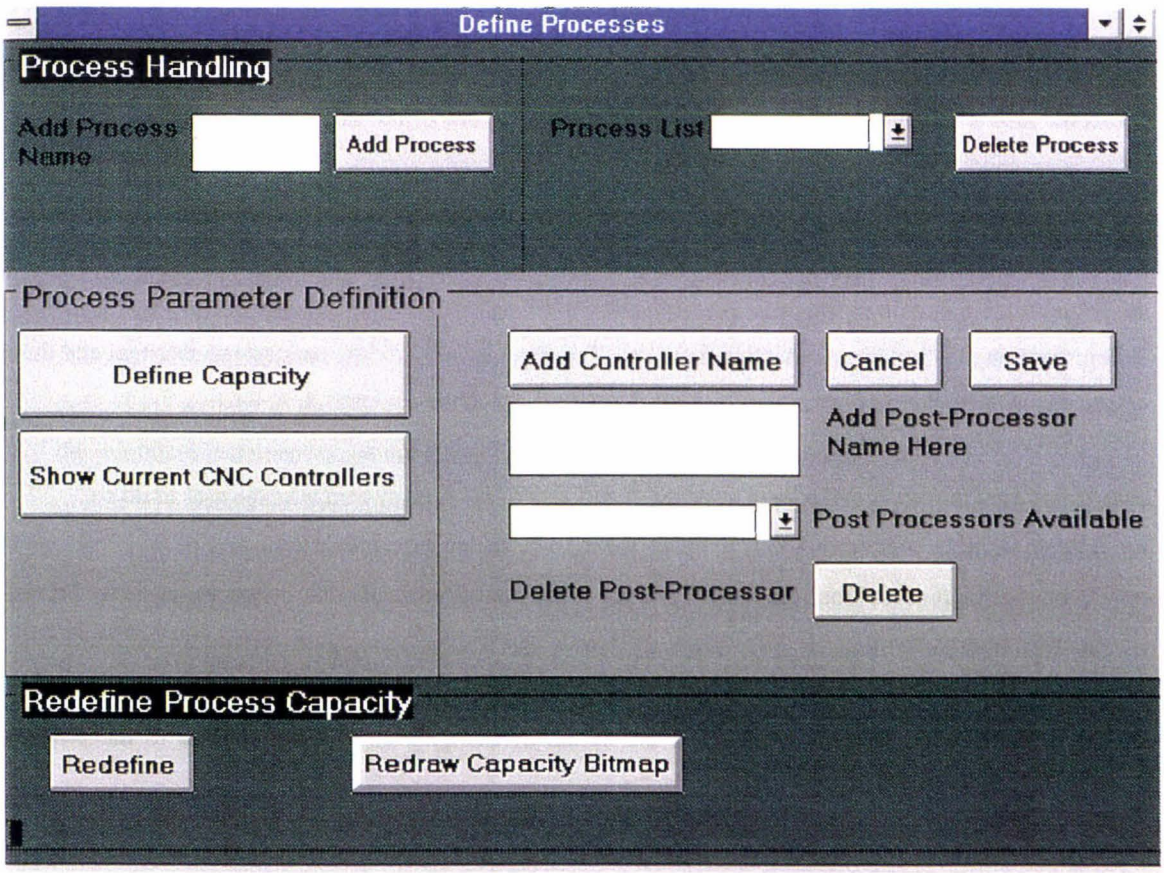


Figure 4.7 - Front End for Process Definition

This particular form acts as the front-end for the definition of all the data required to form a representation of the process.

When this form is loaded a number of actions occur:

- The file holding all current process configuration details is opened ("c:\Factory\Process\files\Proclist.Pfl").
- Details of the process types already configured are read from the file and added to the "Process List" listbox and the maximum capacity available for each process and process ID is added to memory.
- The value of the latest Process ID is read from the file and assigned to memory in case another process is to be added (As processes are added to the system each process ID assigned is equal to the last process plus 1 ie: 1,2,3 ..... etc, etc).

- ***Adding/Deleting Process Descriptions***

When adding process type descriptions the process name is typed into the "Add Process Name" textbox and the "Add Process" command button is selected. This action carries out the following:

- Makes sure that the user has added a process name in the textbox. If he hasn't the user is asked to do so.
- Makes sure that the process hasn't already been defined.
- Adds the name of the process to the "Process List" listbox and assigns a current ID based on the last one assigned.

When removing the process description from the "Process List" listbox the user selects the item and then selects the "Delete Process" command button, this carries out the following:

- Makes sure that an item has been selected from the list. If not the system asks the user to do so.
- Ensures that the user really wants to delete the item by asking for confirmation.
- Removes the process from the listbox and the associated Process ID and maximum capacity for the process from memory.

When the form is exited the user is given the option of changing any additions made to the process descriptions defined.

- ***Defining a Process Capacity***

Process capacity in the terms of the Hybrid Architecture is related to the definition of some process related parameter that may ultimately prevent a given device from carrying out a particular task. In the case of a vertical mill for instance there will be a number of factors that may effect its ability to carry out a task:

- Clearance between spindle and table.
- Power rating of the 3 phase motor driving it.
- Speeds & Feeds available.
- Rigidity of the machine.

The last three in this list although being limitations for the device will not stop it from being able to carry out the task ie: If the motor is not very powerful smaller cuts can be taken. If the speeds and feeds are slow the job will take longer but will still get done and if the rigidity of the machine is poor cutting rates will have to be adjusted to suit.

The clearance between the spindle and the table however is a major concern. If an item of work does not fit between the two it cannot be worked upon. This aspect then becomes the focus of the definition of a processes capacity as it will ultimately limit the ability of the process to carry out a given task.

In the definition of a process's capacity limitation there are two stages that must be carried out:

- Definition of the aspect that limits the process types capacity in a visual sense. The user creates a "Capacity Bitmap" that illustrates what aspect of the process is the limiting factor. This is created by using Object Linking and Embedding (OLE) to integrate the application with Microsoft Paintbrush™.
- Definition of the maximum value of this aspect for all processes of that type in the factory eg: If there were four vertical milling machines in the environment having capacity limitation measurements of 500mm, 600mm, 650mm and 720mm then the maximum value given would be 720mm. During task definition (see later in the Chapter) this measurement is broken up into increments and the user selects the minimum increment range the job fits into. This enables two things to occur:
  - ⇒ It prevents the user from defining a task which cannot be carried out (ie: is outside the ability of all processes in the system) as a capacity interval will not exist if the task is unable to be carried out.
  - ⇒ It enables the system to decide which processes within the system are eligible to compete for the task at scheduling time.

- ***Definition of the Capacity Bitmap for a given Process type.***

To define the Capacity Bitmap for a newly added process the "Define Capacity" command button is pressed after the addition of the new process description to the "Process List", this action loads the Define_CapBitmap form as shown in Figure 4.8.

The user is then able to click the OLE control box when the form is active and this will automatically load Microsoft Paintbrush. The user can then draw a capacity bitmap in the Paintbrush Application and when it is exited this appears in the OLE control box as shown in Figure 4.8.

Once the form is unloaded the Capacity Bitmap for that machine type is saved in the Local Data Storage Service. The name of the process defined is used as part of the bitmap filename so that the Process Configuration for that process type can be directly associated with the bitmap drawn.

Such a Capacity Bitmap is used in the following ways:

- The user is able to define a maximum size for this Capacity Limiting Factor for all processes of this type in the environment.
- Using the Job Definition Service the user is able to select from a series of increments of the maximum capacity defined the minimum increment required for a particular task eg: A task may require a lathe

that has a swing of at least 100mm therefore the user selects the increment that this falls into such as 90-110mm.

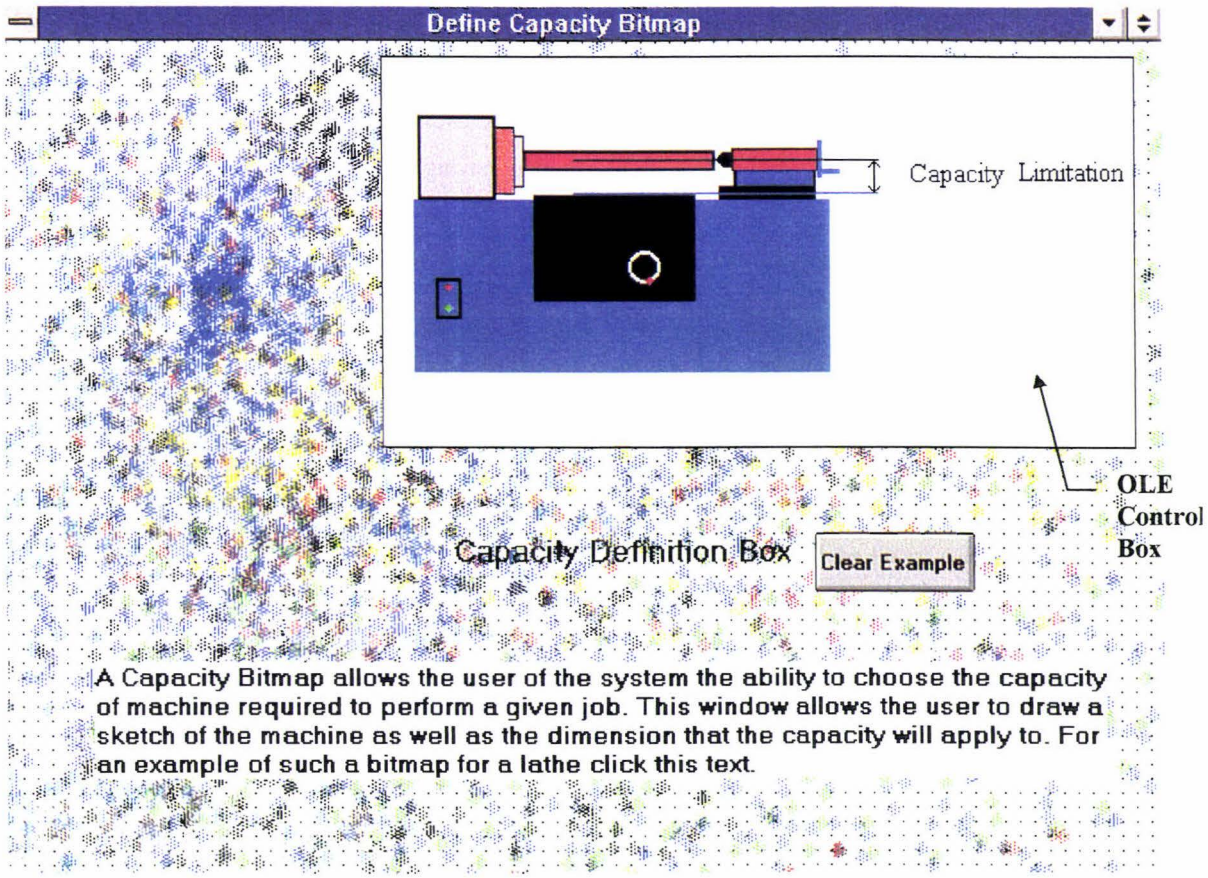


Figure 4.8 - Define Capacity Bitmap form Showing Lathes Limiting Aspect

- When the user defines processes in given cells within the Factory Configuration Service one of the actions that must be carried out is the selection of the capacity increment that the process concerned falls into. This is carried out by the system showing the capacity bitmap for that process type, displaying the incremental values of the limiting factor and having the user select the incremental value that the process falls into. For example if a lathe used for that cell has a swing of 250mm then the increment chosen would encompass this value. Therefore, when an auction occurs and a given process is of the correct type to carry out the task a check can also be made to find out whether it has sufficient ability in terms of the limiting factor to carry out the task.

- ***Definition of the Maximum size of the Limiting factor for a given process type.***

So that the system is able to function correctly it needs to know the largest value of the Limiting Factor for all processes in the factory of the type being defined. This is so that it is able to create increments of this

value from which the user can select the appropriate value for the purpose of Factory Configuration and Job Definition.

To define the largest size of the Limiting Factor for the process type the user exits the form as shown in Figure 4.8 and reselects the "Define Capacity" button in the main process definition form (Figure 4.7). Once the button is reselected the system automatically detects that the bitmap has already been created for the process and loads the "Define Maximum Limiting Factor" window as shown in Figure 4.9.

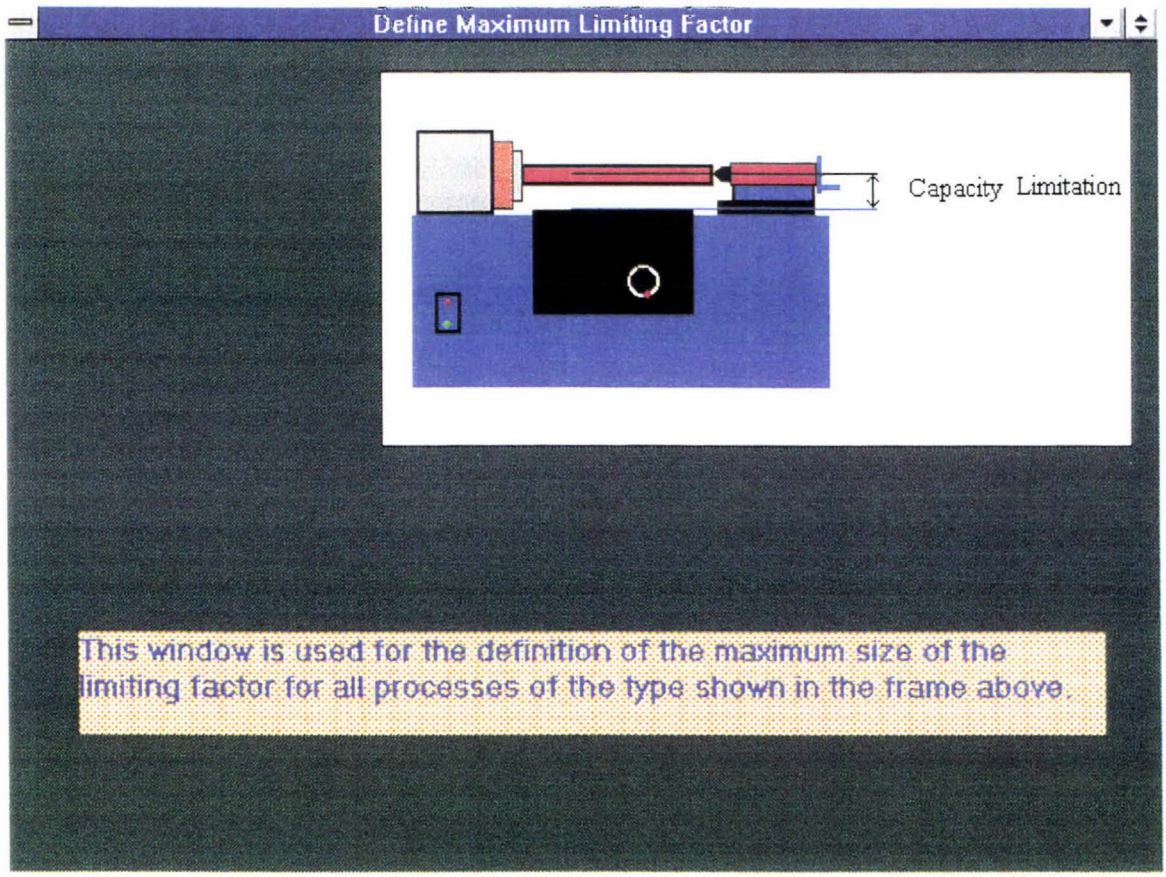


Figure 4.9 - Define Maximum Limiting Factor Window Showing Lathes Limiting Aspect

When this window is loaded the following occurs:

- The bitmap showing the limiting factor is loaded as shown.
- The user is asked to input the maximum limiting factor for processes in the system of the type concerned via an input box.
- If the user enters a value the value given is assigned to a form level variable. Otherwise the user is asked to enter the value again.

- Once the user has defined the capacity interval in this way, the form is exited and the value of the capacity given is stored within the realms of the main process Definition form (see Figure 4.7).

There may be times when the capacity bitmap or the actual process capacity needs to be redefined. This redefinition process can be carried out using the "Redefine" and "Redraw Capacity Bitmap" buttons. The process by which this done is almost identical to that mentioned previously. However it is pertinent to explain other aspects of process definition in the following section.

- ***Defining Post Processors Required for a Process Type***

The definition of the type of Post Processor (and therefore Proprietary NC Controller type) available to a process type is extremely important. When a job is defined in the Hybrid FMS Job Definition Service it is done so on a task by task basis. One of the most important aspects is the definition of the toolpath that a given processes tool must take to perform a given task - this can be undertaken within the realms of the integrated CAD/CAM package. The challenge is to convert this toolpath into an NC language that the Proprietary Controller on the device carrying out the task can understand.

It is not sufficient to post-process the toolpath into any given NC format (eg: Fanuc, Hiedenhain etc) at job conception as the system is not aware of what process (and therefore what controller type) will carry out the task. To overcome this problem the Controller Integration Algorithm as shown in Figure 4.10 was conceived.
The stages involved in the NC program life cycle under this algorithm are as follows:

- The definition of the generic toolpath for the task to be undertaken.
- Storage of the generic toolpath file in the Network Data Storage Service.
- Scheduling of the task.
- Receipt of the Proprietary Controller Details at the Factory Controller Level. Once the task is scheduled the details of the manufacturing processses controller type can be retrieved using a GMR UMS.
- Generation of the appropriate NC file by activation of the appropriate post processor executable file.
- Downloading of the NC file at the Cell's discretion based on the status of the Program Invocation Loop.

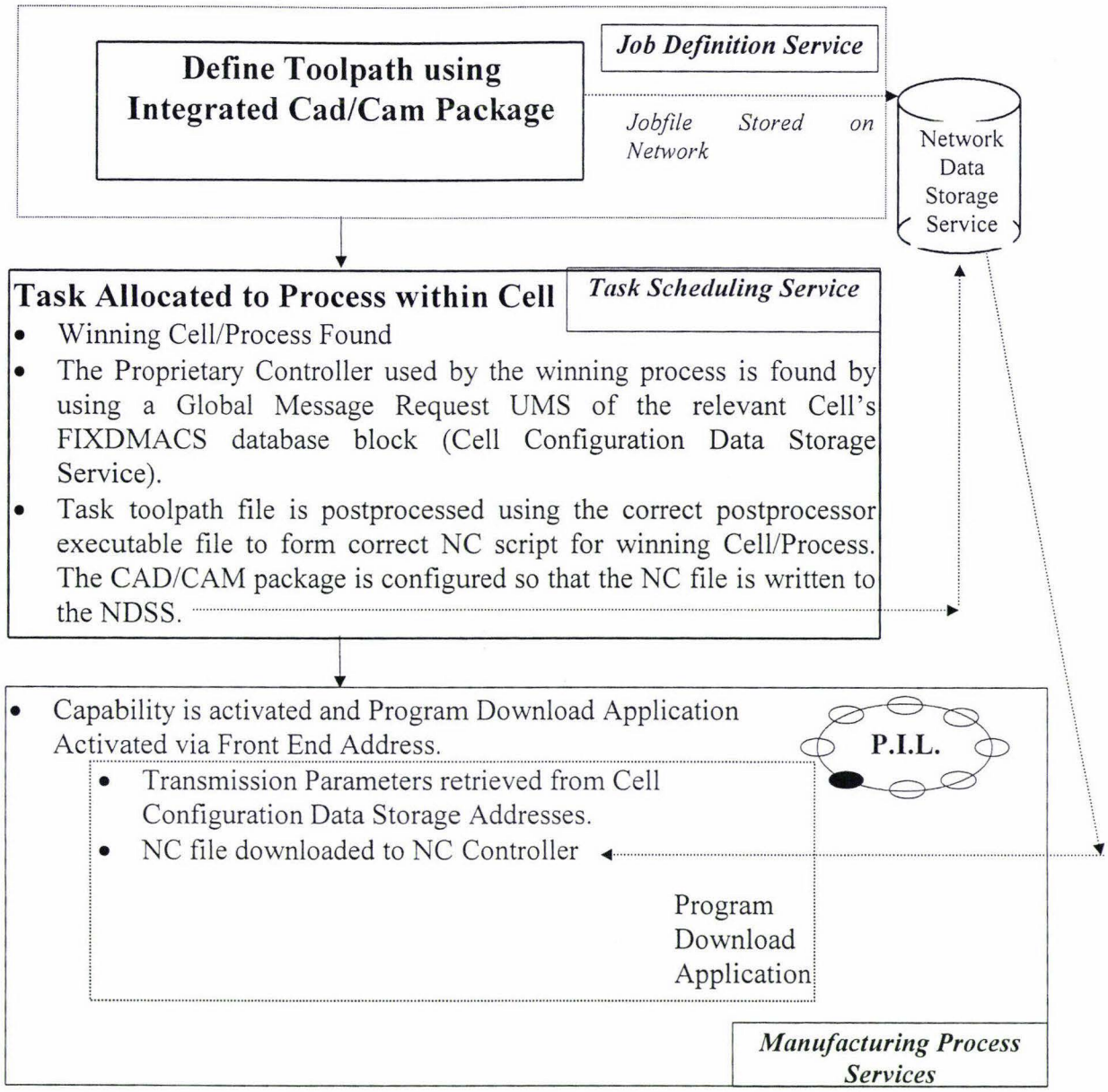


Figure 4.10 - Controller Integration Algorithm

Before a system can be defined one of the steps that must be undertaken is to state the different types of Proprietary Controllers that are available to manufacturing processes of a given type. This is necessary so that when the user defines a manufacturing process within a node (within the realms of the Factory Configuration Service) the process can be selected from a list and the types of NC controllers (or Proprietary Controllers) can be displayed. The user is then able to select the type of controller that will be used on that device (see Figure 4.4).

The definition of the Proprietary Controller types for a given type of process is carried out in the Define_Processes form as shown in Figure 4.7 . To carry this out the user does the following:

- Puts the name of the Post Processor executable file for the Proprietary Controller being added into the "Add Post Processor Name Here" textbox.
- Loads all the names of the currently defined Post Processor types into the "Post Processors Available" listbox.
- Places the process type and the Post Processor name into an array holding all the Post Processor details for that process type.

To save the details the "Save" command button writes the array to a process specific file on the Local Data Storage Service. Selecting the "Delete Post Processor" button and then the "Save" button removes the record selected from the list and saves the changes.

To see the particular post processor types that have already been defined for a process type the user selects the post processor type from the "Process List" listbox and selects the "Show Current CNC Controllers" command button. This allows the user to see what Post Processors have been defined.

All the Post Processors available for a process are defined in this fashion. When the user defines a process within a node the Post Processors available for that process are displayed and the user is able to select the Post Processor required for the NC Controller of the manufacturing process. During the task scheduling phase the Task Scheduling Service is able to award the task to a given manufacturing process, gets its Post Processor details and runs the correct executable file to convert the toolpath file into a valid NC file for that process.

4.1.3.2 Implementation of the Tool Configuration Service

The Tool Configuration Service is a service that resides within the realms of the Resource Configuration Service. Through this service the tool types that are available for a given process type must be stated so that the user is able to select from them when defining what tools are available to a given process during the Factory Configuration stage. It cannot be assumed that all milling tools can be used on every conceivable Vertical Milling machine for instance, and as such the milling tools that can be used on a given process must be selected from a file of tools for that process. Therefore the Tool Configuration Service's role is involved with the definition of tools available for a given process type. In reality the user would have a Central Tool Carousel at the Factory Controller level and as the tools were placed in the carousel for the first time they would be defined using this service (except if they were replacements for broken tools).

To implement the service the following is carried out:

- At the main Factory Controller window (See Figure 4.1) the "Define Tools" button is selected. This loads the Define_Tools form as shown in Figure 4.11.

- When this form is loaded the file holding all the currently configured manufacturing process types ("c:\Factory\Process\files\Proclist.Pfl") is opened and the contents removed and placed in the "Select Process" listbox.

- The user then selects from the listbox the type of process for which the tools are to be defined and selects the "Define Tools For Process Selected" command button.

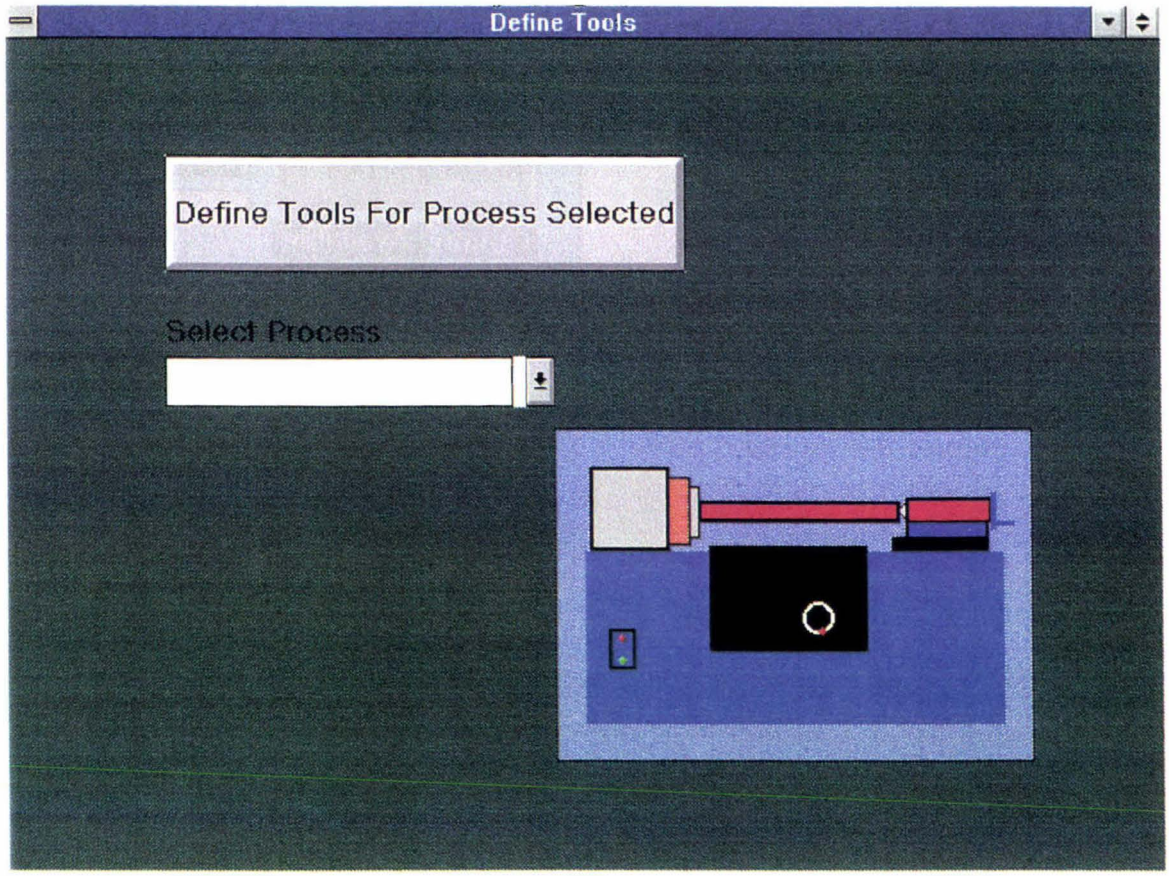


Figure 4.11 - Define_Tools Form

Selection of this button activates the Statetools form as shown in Figure 4.12. The purpose of this form is to allow the user to define the tool types for this particular process type. Once this is carried out the individual tools can have the actual sizes specified for them.

When this form is loaded the following actions occur:

- A bitmap of the process for which tools are to be defined is loaded into the OLE control as shown in Figure 4.11.

- A file is opened (stores options) and the most recently used and defined tool types are stated as captions for the option buttons as shown in Figure 4.12. The idea behind this is that there will most likely be 3-4 tool types most commonly used for a particular device and, therefore, these should be presented as options. Any one of the options can be replaced by selecting the option to be replaced and selecting another tool option from the list box and clicking the "Use Old Option" command button. This type of arrangement keeps everything simple and allows the user to focus on the commonly used tool types.

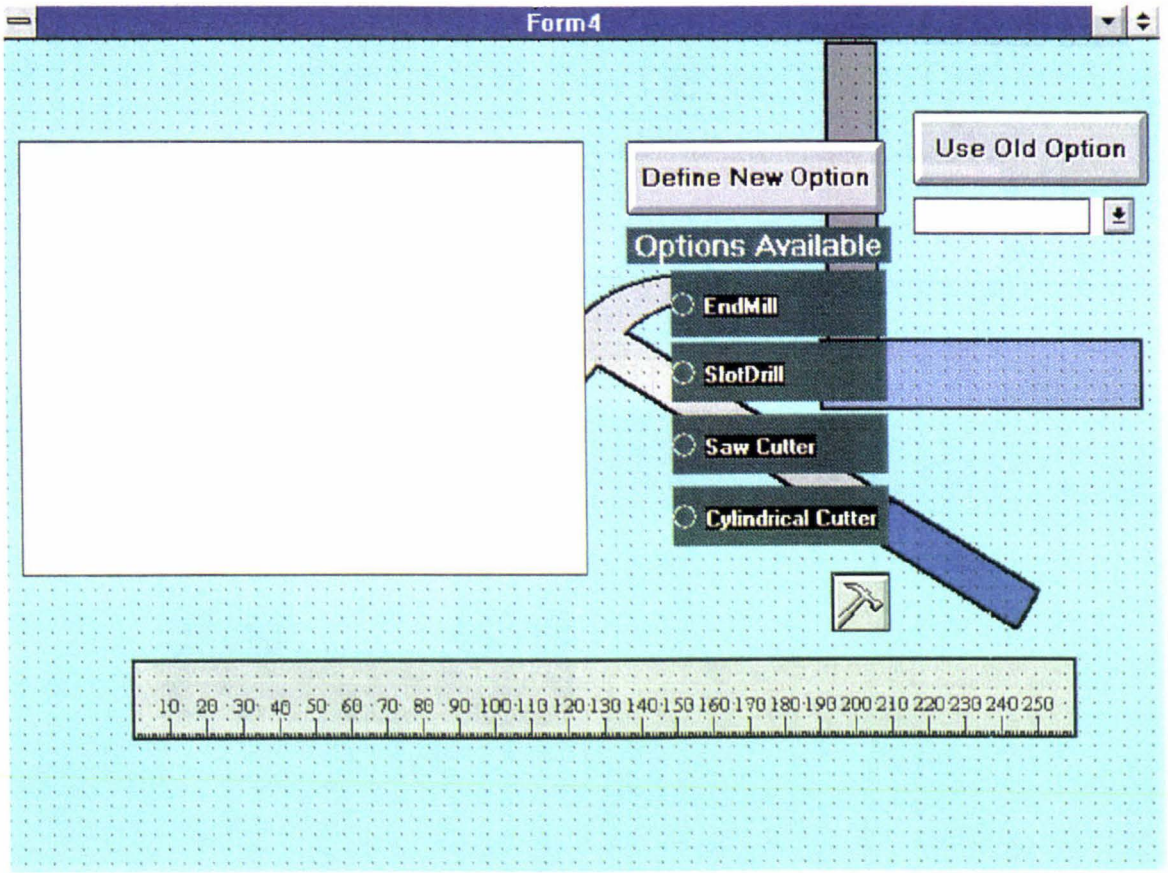


Figure 4.12 - Statetools form

Selecting the "Use Old Option" button loads from the file all other tool types that have been defined in the past but are not deemed to be major tool types. If the user wants to define a new size for a given tool and the tool type is not part of the "Options Available" option then the user selects one of the tool types in the "Options Available" list and replaces it by selecting the required tool type option from the "Use Old

Option" list as shown. The option selected from the "Use Old Option" list replaces the option selected from the "Options Available" list and the option being replaced is added to the old options list. The user then selects the Tool Type required to have sizes defined for it using the option button and activates the Toollists form as shown in Figure 4.13.

To define a completely new tooltype for the associated process the user selects the option that is to be replaced from the "Options Available List" and selects the "Define New Option" command button. This carries out the following action:

- Asks the user for the name of the new tool type.
- Checks to see if that tool type has already been defined previously, if it has the user is informed and the procedure is terminated.
- Replaces the caption of the option button selected with the new tool type definition.
- Opens the file holding the names and identification codes of all the tooltypes that have been defined("c:\factory\system\AlToolTyps.Tl") and reads the contents. A new tool ID code is assigned to the new tool type and the tool name and code is written to a file. The file is then saved with the changes.
- Saves the "Options Available" option list to a file.
- Adds the option replaced to the old option list and saves the changes to a file.

Once this has been carried out the user is then able to define tool sizes for any tool types configured. To do this the user selects the tool from the option list and clicks the hammer icon, this loads the Toollists form as shown in Figure 4.13.

The purpose of this particular form is to allow the user the ability to define tool sizes resident in the Hybrid system for the tool type selected. When this form is loaded the following occurs:

- A file is opened for that particular tool and process type and the contents read. If any tool sizes have been defined their corresponding size is placed in the "Tools Existing" list therefore allowing the user to be aware of the sizes that have already been defined.

  When the user selects the Tool Specification followed by the View menu a tool specification bitmap is shown in the picture box on the left side of the form as illustrated in Figure 4.13. This shows the user the particular aspect of the tool for which the size is being defined. This is important as multiple users may have access at this level and size definition must be consistent with the tool being defined.

To add a new size for a given tool type the user carries out the following:

- The size of the tool is typed in the textbox above the "Add New Size" command button and then the command button is pressed. This action carries out the following:
  - ⇒ The size of tool is added to the bottom of the "Tools Existing" listbox.
  - ⇒ The user is asked to input the expected life for the tool.
  - ⇒ The process and tooltype specification file is opened, and the toolsize, tool life and tool identification code, is written to the file along with all the other sizes available for the tool.

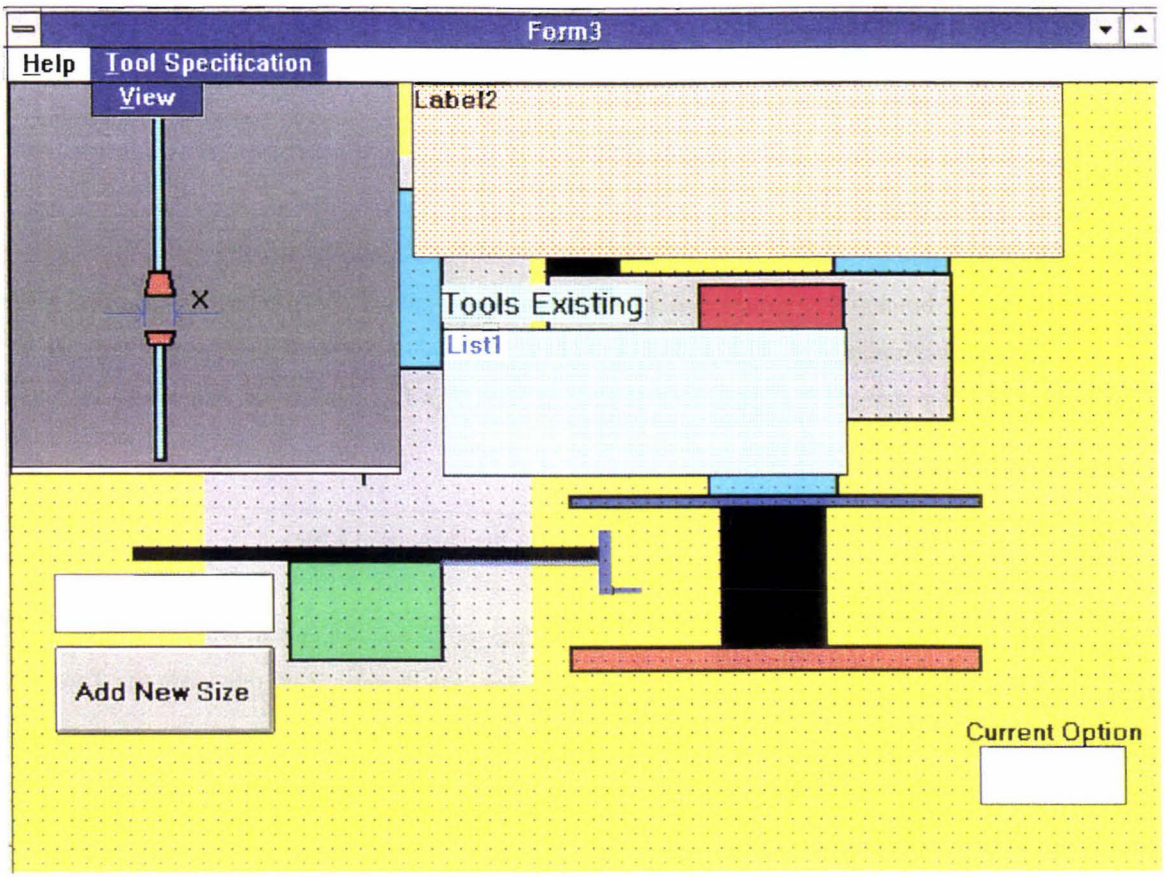


Figure 4.13 - Toollists form

When tools are defined for a particular manufacturing process in a given Cell Controller via the Factory Configuration Service the user defines the types of tools available by first selecting the tool type. Once the tool type is selected the service automatically loads all the tool sizes that are available for that tool type in the form of option controls. This allows the user to select the tools available and sizes available without having to define what they are in the first place. The user is then able to define what tools and sizes are available to a given manufacturing process within a cell without having to define what this means to the system. This makes the system more transparent and user friendly.

4.1.4 Implementation of the Job Definition Service

The definition of a particular job is one of the most important aspects of the overall system. The Hybrid architecture has been designed so that jobs can be defined in a generic sense. This makes the system adaptable to a wide range of manufacturing environments.

There are a number of aspects that need to be defined for each job as has been explained in Chapter 2 and this is the purpose of this application.

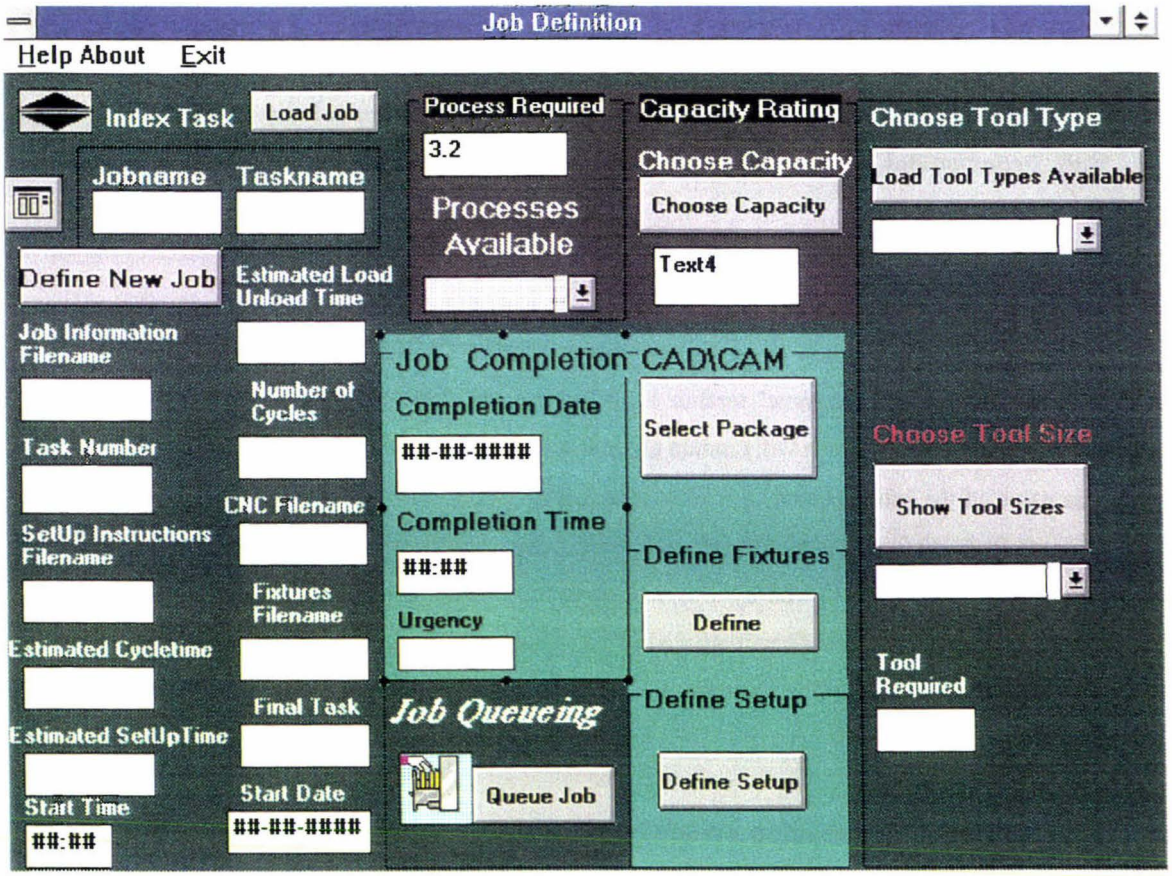


Figure 4.14 - Def_Job form

To start the service the "Define Job" command button in the Main Factory Controller Window (see Figure 4.1) is pressed - this action loads the Def_Job form as shown in Figure 4.14.

This particular window is used as the front end for the definition of all aspects of a job and, therefore, all aspects of each task making up that particular job.

When this particular form is loaded the load event opens the file containing the names of all processes that have been configured for the system ("c:\Factory\Process\files\Proclist.Pfl") are loaded into the "Processes Available" listbox, providing the means by which the user is able to select the process required to carry out the task. This selection of the process acts as a precursor for the rest of the job definition activities.

The method of job definition is based on a task by task basis. The user defines each parameter required for each task (see Chapter 2 for task parameters) and then uses the spinbutton at the top of the form to advance to the next task required to be defined to make up the job. This creates a very structured and logical approach to job definition and one that can be used in a wide variety of real world applications. As such the Job Definition model can be viewed as a series of individual tasks connected together in a string to form a representation of the job. In the ensuing discussion the job definition phase will be discussed at its lowest level - task definition.

- ***Task Definition Basics***

The first part of the task definition process is the definition of the Jobname, adding the name of the job to the "Jobname" textbox causes the following to occur:

- The "Job Information Filename" textbox is given the name of the jobname with the suffix ".iff" , this is the name of the file which will contain all information about the job.
- The name of the Job followed by 1 is written to the CNC Filename textbox. This gives it a unique toolpath filename for that task and job.
- The name of the Job followed by 1 is written to the Set up Instructions filename textbox and the Fixtures Filename textbox. This gives it a unique filename for these aspects.
- The "Processes Available" list is enabled to allow the user to select the required machine to carry out the task.

The user is also required to manually enter a number of other job parameters. These include the following:

- Number of Cycles - The number of parts required to be worked upon (Batch Size).
- Final Task - The number of tasks making up the job being defined.
- Start Date/Time - The user must enter the required start time and date for the overall job. This allows the Task Scheduling Service to queue jobs at the Factory Queue at the required time. The control item that the information is entered into is a Masked Edit Box which restricts the format that can be entered.
- Estimated Set up Time - When a new task is defined the set up time has to be estimated.
- Estimated Cycle Time - Time to complete one task cycle.

- Completion Date - Date by which the job must be completed.
- Completion Time - Time by which the job must be completed on the target date of completion.

- ***Defining Process Type and Capacity required to carry out the task.***

The next step involves the definition of the type of process required to carry out the task. In reality the user of the system would receive a blue printed drawing of the part required to be manufactured and from this would break the part (The job) up into a series of sequential tasks required to complete the job. For example: Task 1 - Lathe, Task 2 - Mill, Task 3 - Case Harden, Task 4 - Cylindrical Grind. It is interesting to note that when defining a process type for a task we assign the capacity of the process relevant to the limiting aspect in the Resource Configuration Service. Even a Case Hardening furnace has a limiting aspect - in this case it may be the maximum length of article that can be fitted within it. The flexibility of the system allows even this to be defined in the Resource Configuration Service. There are two major aspects that must be selected in terms of selecting a process for a task - Type and Capacity required. This is carried out in the following way:

- The user selects the process required from the "Processes Available" list. The name of the process selected is placed in the "Taskname" textbox.
- The user then selects the "Choose Capacity" command button.

Selecting the "Choose Capacity" button causes the following to occur:

- The details of the process selected from the process file ("c:\Factory\Process\files\Proclist.Pfl") are obtained.
- A copy of the CNC_Drill form (see Figure 4.15) is loaded, the Capacity Bitmap is placed in the OLE control and the capacity intervals defined in the Resource Configuration Service are loaded into the listbox.

To select the required capacity interval for the task in operation the user selects the capacity interval from the "Capacity list" and presses the "Select" command button. This creates a two digit number representing the process type and the capacity. This number is then written into the "Process Required" textbox in the Def_job form (see Figure 4.14). Once this has been completed the user exits the form.

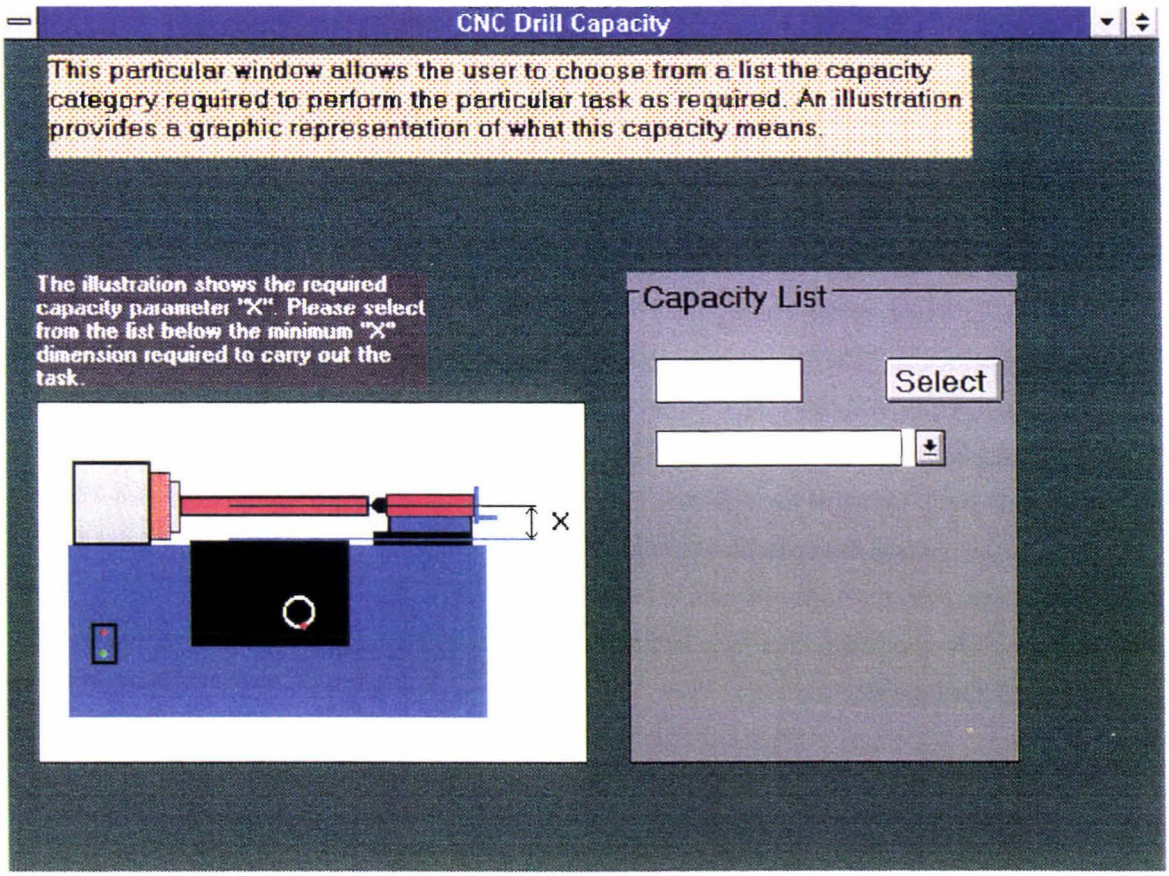


Figure 4.15 - CNC_Drill form

- ***Defining a Tool Type and size required to carry out the task.***

Once the process required has been selected to carry out a given task the next stage is to select the tool required to carry out the task. This is carried out by selecting a tool from the tool file for that particular process type (this file having been created in the Resource Configuration Service) and then selecting the tool size required.

This is carried out in the following way:

- The "Load Tool Types Available" command button is selected. This loads all available tool type names that have been defined for the process selected into the tool type listbox, therefore, allowing the user to select the required tool for the task.
- After selecting the correct tool type, the user then selects the "Show Tool Sizes" command button to allow the sizes configured for that tool type and process to be shown. These sizes are written to the tool size list as shown in Figure 3.30, therefore allowing the user to select the size as required.

- The user then selects from the show tool sizes list the tool size that is required for the task at hand. When the tool is selected the program then opens the file holding toolcodes for all tools available in the system, retrieves the tool type code and using the size selected generates a toolcode eg: If the tool selected was a endmill having a type code of 10 and a size of 15mm diameter, the toolcode generated would be 10.15.

  The code generated is then put into the "Tool Required" textbox, therefore, enabling the user to see that he has stipulated all job parameters before saving. In the circumstance whereby a cell is configured in the Factory Configuration Service the tools allotted to given processes are defined in the same way and the Task Scheduling Service is able to decide whether or not a process is eligible to form a bid for a task.

- ***Defining a NC Toolpath for a given task***

Defining a toolpath for the tool being used to carry out a given task is one of the most important aspects of the system. As discussed in earlier sections of this chapter the NC portion of the system involves the definition of a generic toolpath. At a later date its post processing file is derived once the target process and therefore controller type is known.

In task definition we are required to define a generic toolpath. This is carried out in the following way:

The "Select Package" command button is clicked when all other task parameters are defined. This carries out the following actions:

- The Cad_Cam form as shown in Figure 4.16 is loaded.
- The name of the toolpath file to be used is written to the Task Toolpath File Name textbox.

In this window the user is able to define the name of the Cad_Cam packages that are required to be integrated with the system. The user does this as follows:

- Selects the "Configure New Package" button. The user is then asked for the common name of the package, the name of the command line (Icon property) for the program, the programs window title and the programs working directory. The system then reacts in the following way:
  - ⇒ This information is saved to the "C:\Factory\System\Cadcam.sys" with all the other information for other configured packages.
  - ⇒ The name of the package is written to the "Select Package" listbox.

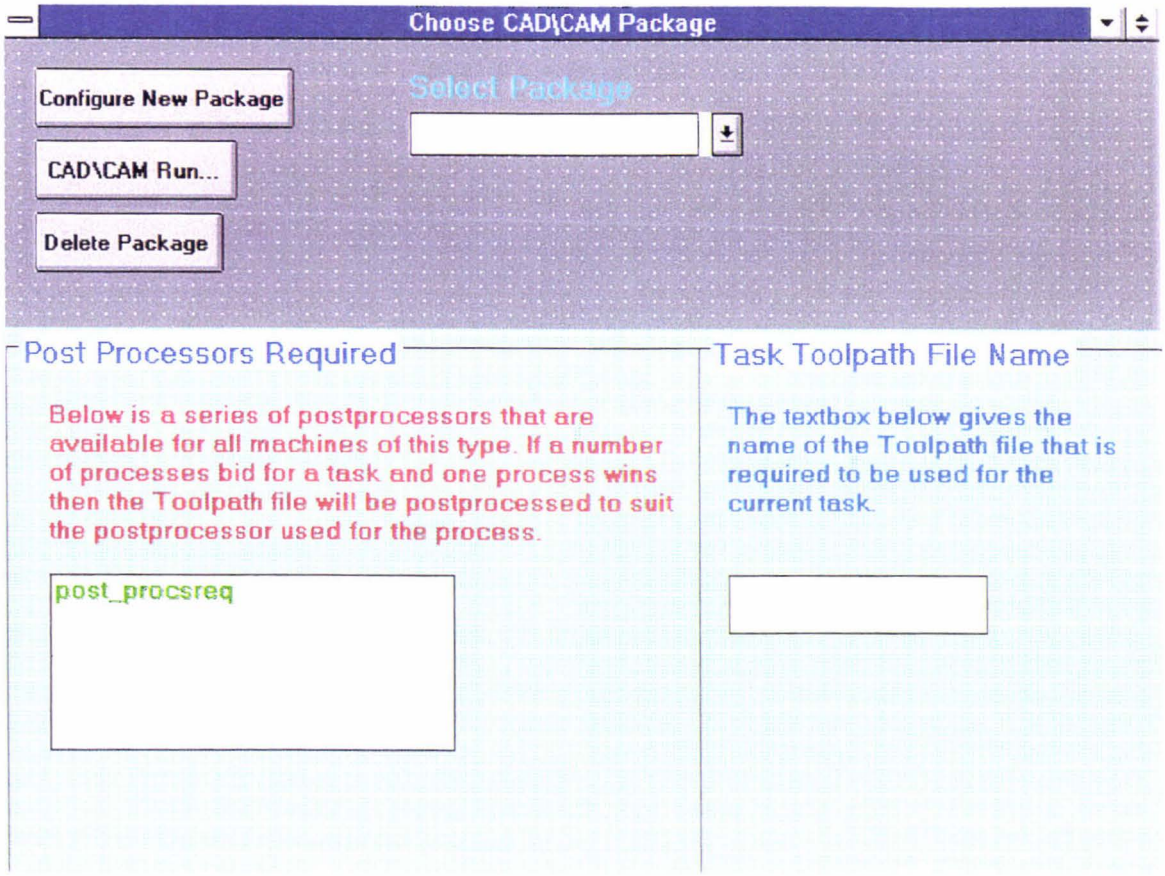


Figure 4.16 - Cad_Cam form

When the user wants to activate the Cad Cam package it is simply a matter of selecting the name of the package from the "Select Package" list and then the CAD\CAM Run... command button. The program will use the saved details to activate the required package to allow the user to create the required drawing and as such the required toolpath file.

In this research the integrated package used was Mastercam(Figure 4.17) as this was the only package available at the time. In the example the geometry of the shape to be machined is drawn as a contour and is being defined as the first task for the job called "Wheel". The job's drawing is saved as "Wheel 1.Ge3" and the tool path file is generated by defining the contour using the chaining method and is also saved using the same name - however using a different suffix (Wheel1.Geo).

When the job is scheduled (in the Task Scheduling Service) the name of the post processor required for that manufacturing process carrying out the task is retrieved (the same name as the required post processor executable file). The name of the file required to be post processed (in this case Wheel1.Geo) is passed to

he post processor executable file and the required NC code is generated. This code is then stored on the ietwork for that particular task and downloaded as instructed by the Program Invocation Loop.

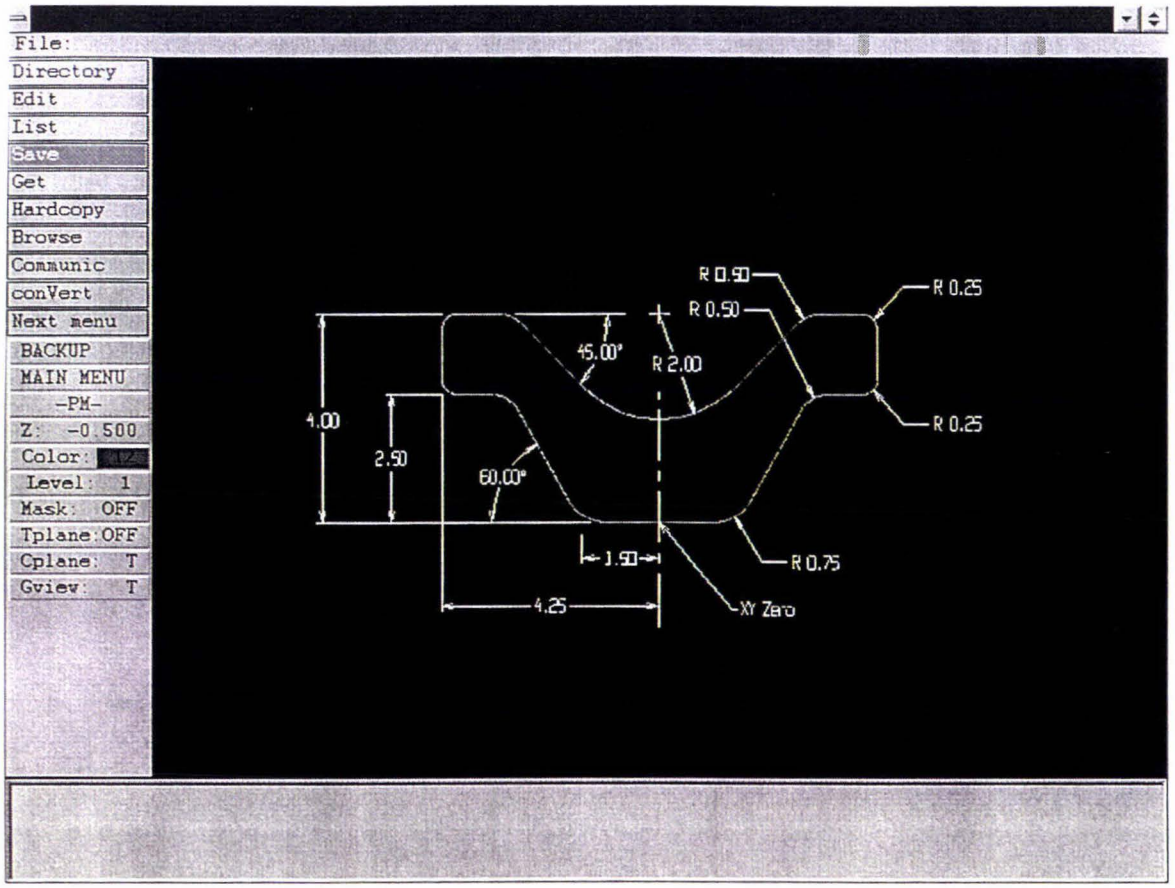


Figure 4.17 - Mastercam Package Main Window Showing Dimensioned Drawing

- *Saving Task Details*

Once the task details have been defined as shown above, the user is able to index up and down between tasks for a given job to define other tasks within the job definition using the "Index Task" spinbutton control. As the button is used to index up or down the task defined before indexing is written to an array and then saved to file. During this process a number of things occur:

- The tasknumber is incremented or decremented depending on whether a spinup or down occurs.
- A check is made to ensure the user has defined all task parameters as required. If not the action is terminated and the user instructed to define all parameters.

- If the user has defined the first task and spins up to the next the systems asks the user whether the job is urgent. If it is, the user will answer in the affirmative and the job will always be placed at the top of the queue at auction time.
- When spinning up or down the task parameters just defined are saved to file and, if there are any, the next task parameters are loaded(will only occur if modifying an existing task).

The way the indexing procedure has been written allows the user to load existing jobs redefine the tasks making up the job and resave them as required. This is particular important in that batch sizes may change (in response to policy changes), methods of manufacture may change (in response to new technology) or a product specification may change eg: an axle dimension may need to be modified in response to a change in bearing supplier. Therefore, the system retains a high degree of flexibility to change. The system is also very user friendly as it allows task data to be displayed by the user as the user indexes up and down the tasks making up the job.

- ***Queueing Jobs for Manufacture***

Once jobs have defined they can be treated in two different ways:

1. They can be queued for manufacture.
2. They can be saved and queued at a later date.

When queueing a job the user must load the job required to be queued by selecting the "Load Job" command button. This action displays a directory listbox and allows the user to select the name of the job to be loaded. The job is then loaded and the first task details are shown - this action of loading the job allows the user the option of checking the task details (using the spin button) before job queueing occurs.

To queue a given job the user selects the "Queue Job" command button. This action carries out the following:

- The Pending jobs file ("C:\Factory\Queue\jobspend.jpf") is opened and the contents are read. The Pending jobs file holds the names of all jobs that have had start dates and times allocated to them but are not as yet due to be started.
- A check is made to see if the job is already in the Jobs Pending file. If it is the user is informed.
- Another record is added to the file and the name of the job is saved.

Once the job is queued the Task Scheduling Application periodically examines the file and determines when the job is queued at the Factory Controller by examining the job start date and time details. Once on

the queue the first task making up the job becomes eligible for scheduling when the job reaches the top of the queue and the node is polled.

4.2 Summary

This chapter has outlined in detail the implementation of the Factory Controller Services. Chapter 2 has dealt comprehensively with an explanation of the Hybrid FMS model. However, to get an appreciation of how the model was implemented from a Factory Controller Services aspect it was necessary to go into sufficient detail regarding the implementation. This is necessary for two reasons - so that that a knowledge of how the software tools were used to implement the model could be gained, and also if future development of this package is to be undertaken then the developer has sufficient information to continue.

The next chapter will deal with the implementation of the cell controller based services and will follow the same format as this and the preceding chapter.

Chapter 5

Implementation of the Hybrid Model - Cell Controller Based Services

5.1 Implementing Cell Controller Based Services 163

5.2 Summary 185

5.1 Implementing Cell Controller Based Services

5.1.1 Implementation of the Address philosophy

The main function of the Front End Addresses (see Chapter 2 for Definition) are to provide a generic interface to the real world devices and in so doing allow any real world controller (Eg: PLC, PSC etc) to be connected to them. The address philosophy creates the environment whereby real world controllers can be integrated regardless of their form, this is shown in Figure 5.1.

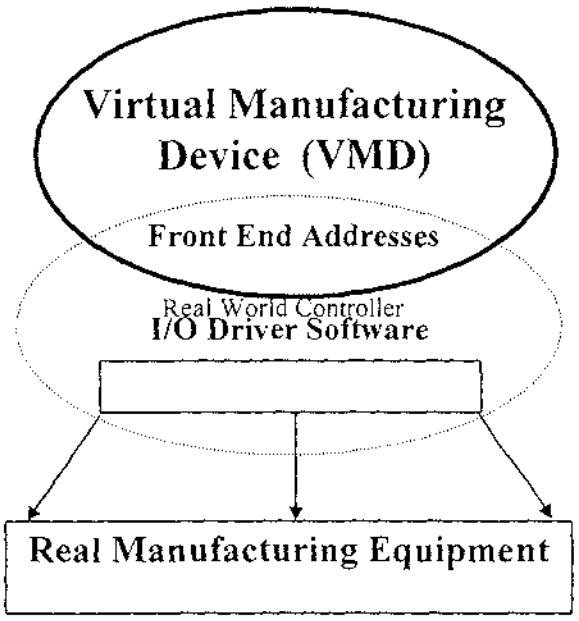


Figure 5.1 - Model Showing Link between Virtual and Real World

The Front End Addresses therefore become the link between the real world and the virtual world. As was mentioned in Chapter 2 the virtual behaviour of a manufacturing device can be modelled using the Program Invocation Loop (within the VMD shell). The Program Invocation loop can be customised to create a generic loop describing the cyclic behaviour of the device being controlled eg: Check Interlocks, Start, Turn on oil Lubrication System, Activate coolant pump etc etc. As such any logic traditionally carried out at the Real World Controller (PLC, PSC etc)level can be taken away and modelled using the Program Invocation Loop. This creates a generic control environment at the Virtual Manufacturing Device level that can be used regardless of the type of Real World Controller logic being implemented. In

reality the Real World Controller (RWC) only becomes a data transition point between the virtual and real world. This means that customised programs are not required for a particular RWC and as such the VMD becomes highly transportable and compatible with all RWC's.

The VMD can be created by using a master Visual Basic program (called the Process Control Application - see later) to execute the program blocks within the FIXDMACS environment using the Universal Messaging System (UMS). In the context of the Hybrid FMS these FIXDMACS Program blocks constitute the ***Capabilities*** mentioned in the previous chapter in discussion of the integrating infrastructure model. These program blocks making up the Program Invocation Loop then acquire and alter the state of the Front End Addresses and as such enable the virtual environment to be mapped onto the real enviornment. The Program blocks are able to be stored in a *Capability* library, selected , ordered and executed by the Process Control Application.

Generally speaking when setting up a new Cell Controller there are a number of steps involved in a software sense:

- Loading the Generic service software required.
- Creating a Process Behaviour Algorithm for each process to be used.
- Creating the real software environment (VMD) to mimic the process behaviour logic for each process 'attached' to the cell.

Figure 5.2 below show a simple Process Behaviour Algorithm that could be created to model the behaviour of a CNC mill. This algorithm is very basic (as it is only an example) and will generally need to be expanded to fully describe what actually occurs so that a process cycle can be completed. However it gives an idea of what is involved.

The key factor is that each part of the logic can be implemented within the realms of a Program Invocation Loop using FIXDMACS program blocks interacting with the various types of blocks making up the front end addresses within the cell. The program blocks required to mimic the algorithm are selected, the order in which they are initiated is programmed using the Process Control Application (see later) to follow the algorithm and as each block is executed it interacts with the Front End Addresses to complete the logic of the VMD.

An example of how this is achieved in reality is shown in Figure 5.3 in the context of the "Down load NC file to device" step shown in the algorithm. This routine has already been generated as it is generic (in other words it performs a generic function ie: downloading an NC file) as oposed to a routine that has been created by the user from a new program block. When the "DOWNLDCNC1" FIXDMACS program

block is activated (placed on scan) by the Process Control Application it passes a "1" (or flag) to the "V1CNCSTART" database block. This block has a DDE

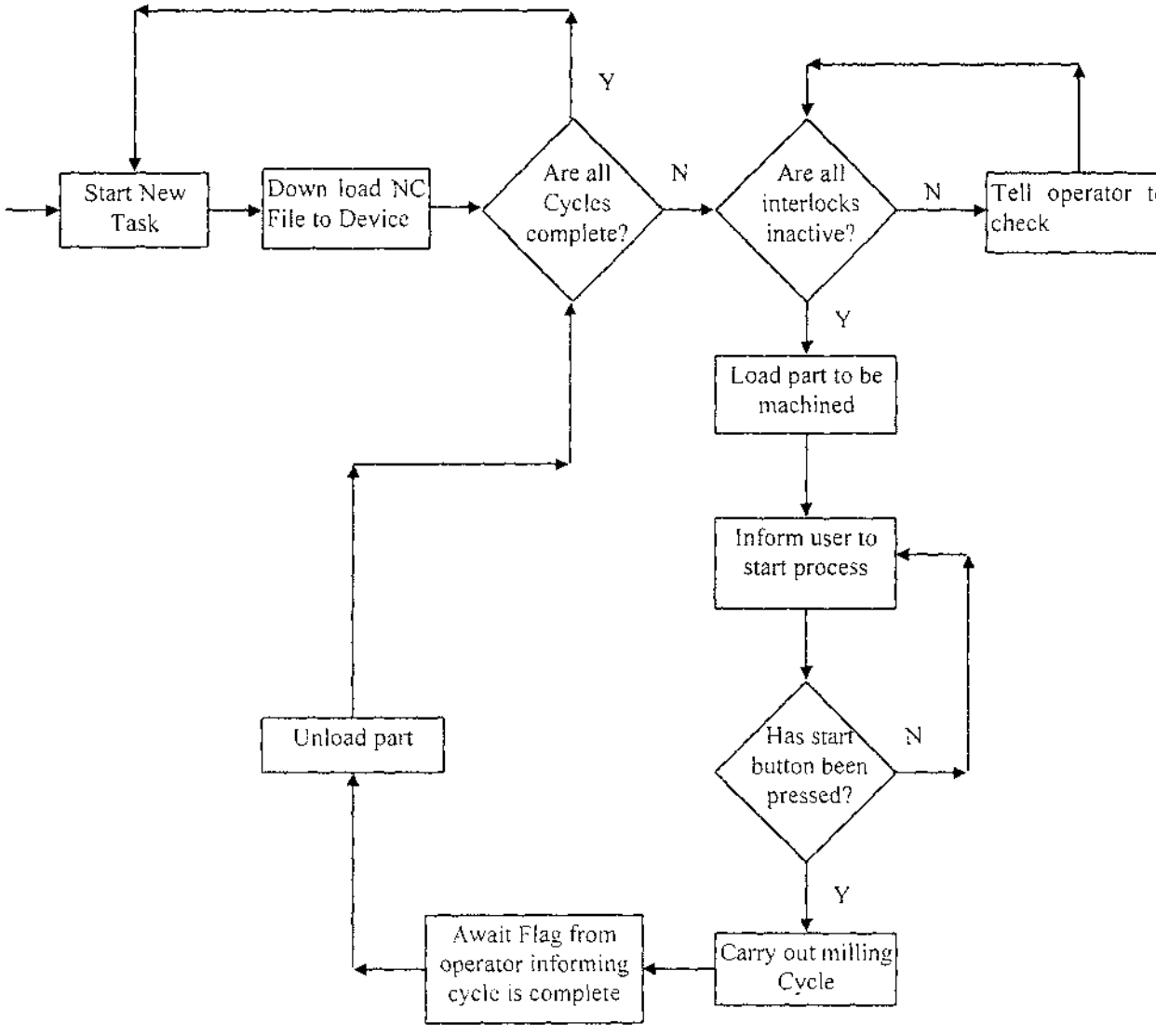


Figure 5.2 - Example of a Process Behaviour Algorithm - Mill

link to the Program Download Application (program within the Manufacturing Process Services) which is programmed to recognise this action as a request for a NC program download to Process 1's NC Controller. The application then responds accordingly by downloading the application and sending a message to the "V1DOWNCOMP" database block informing the program block the download is complete. The Program Configuration Application then initiates the next block within the program invocation loop that completes the next step of the algorithm.

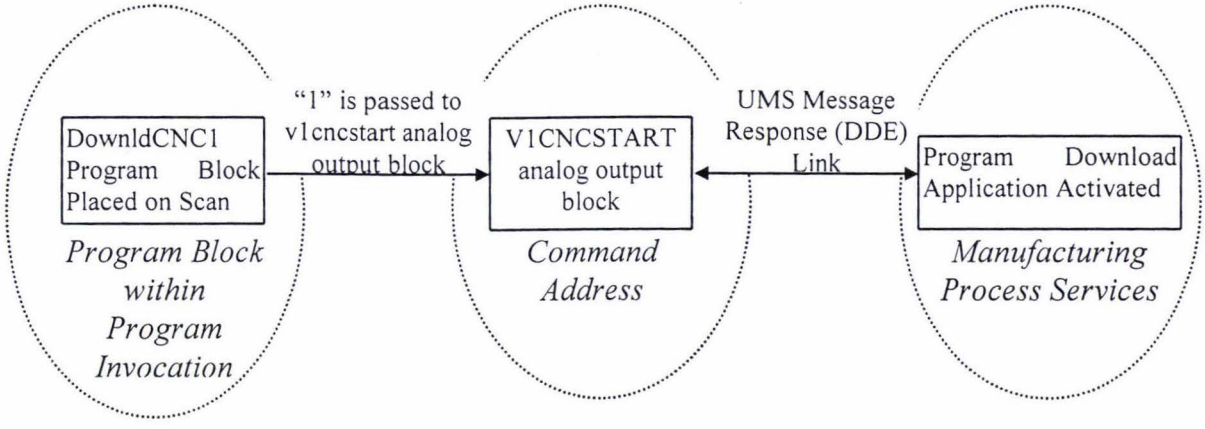


Figure 5.3 - Model of the Generic NC Download Program

There are a number of generic routines that can be supplied with the system to achieve certain standard functions. For example, the "DOWNLDCNC1" program block (see Figure 5.4) is used for the downloading of NC files to the proprietary controller on the process defined as process 1. The "V1STARTIT" program block is generic in that it sends a "1" to the "V1STARTUP" digital output block thereby switching the process on.

In reality these generic type of FIXDMACS database blocks would come as a standard generic package with the system in a marketable version. The program database blocks are then organised in such a way using the Process Control Application to emulate the process behaviour algorithm. However in many instances not all program blocks and Front End Address database blocks (Analog Input, Analog Output, Digital Input, Digital Output Blocks etc) are available to carry out a step in the process behaviour algorithm due to the process being unique. In this case the user can create a custom program block together with the required front end address database blocks to perform the step in the algorithm.

The result of this type of system is a protocol that allows a large variety of controllers and process types to be integrated into the main Hybrid FMS.

The ***Front End Addresses*** therefore are made up of standard FIXDMACS database blocks, the actual type (see Appendix C for listing of all Cell based database blocks) used being dependent on the form the of the data being transmitted *or* recieved from the Real World Controller.

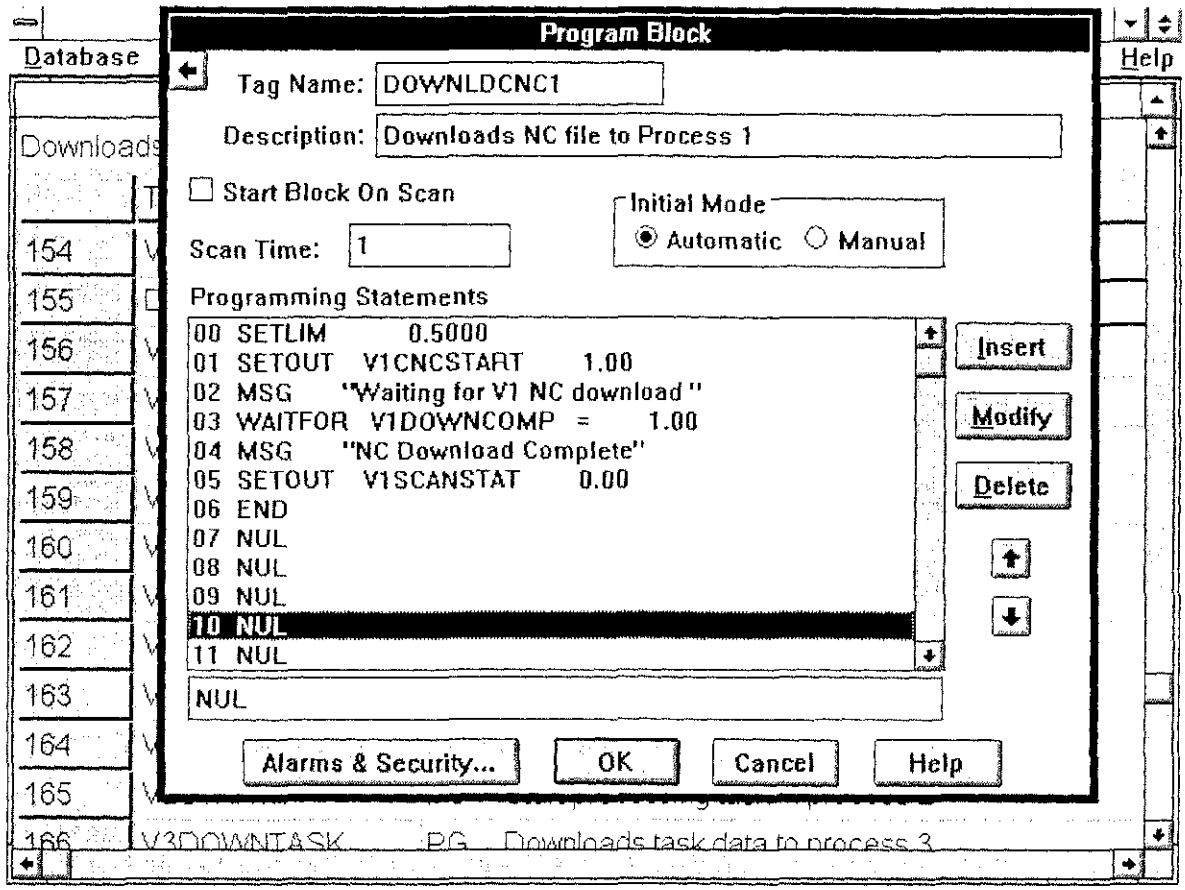


Figure 5.4 - FIXDMACS Database buildershowing code forDOWNLDCNC1 Program Block.

The *Command Addresses* are analog output FIXDMACS database blocks which are part of the system used to integrate the functioning of the database with a service in the Manufacturing Support Services. An example of this is the "V1CNCSTART" analog output block used to interface the FIXDMACS database with the Program Download Application.

The *MFE Task Data Addresses* are analog output and text blocks used to house essential information about the task being manufactured such as number of cycles required, NC file name, task set uptime, cycle time etc. These addresses carry this information and the system refers to it during the processing of a given task.

The *MFE Status Addresses* are FIXDMACS database blocks responsible for acquiring state data for the process they are connected to. A example of where this type of address would be used is where a FIXDMACS program block checks a series of process interlocks before proceeding to the next stage of the operation. A digital input block would be assigned to each sensor on the process and updated by the

FIXDMACS SAC function depending on the scan setting and checked by the Program Invocation Loop before proceeding with startup.

The *MFE System Addresses* allow the Factory Controller the ability to make contact with a *specific process*. An example of where this is used is when the Factory Controller informs a specific process that it has been awarded a tool and can start processing the associated task. In the context of Process 1 it sends a GMT UMS to the relevant cell's "V1TOOLRESD" FIXDMACS database block.

5.1.2 Implementation of the Manufacturing Process Services

The model of the Manufacturing Process Integrating Infrastructure presented in Chapter 2 has been replicated for clarity as shown in Figure 5.5. This model will be referred to when explaining how it is implemented using the software tools presented at the beginning of this chapter. The Manufacturing Process Services is made up of two different sub-classes of services:

1. **Manufacturing Support Services** - Contain applications that are required to support the task processing effort but are not directly associated with carrying out the task. These applications include:
    - *Program Download Application* - This is responsible for the downloading of Numeric Control files to the host cell's processes when instructed to by the Program Invocation Loop.
    - *Queueing Application* - This is responsible for the queueing of tasks to be carried out and also those that have been completed and are awaiting to be rescheduled after processing. The application is also responsible for the downloading of task data to the respective manufacturing process as required.

2. **Activity Control and Coordination Service** - This service is directly responsible with ensuring that activities associated with the processing of a task are carried out as required. The main application involved in this process is the *Process Control Application.* This application is responsible for the initiating of the FIXDMACS program blocks in such a way as to ensure that the logic of the Process Behaviour algorithm is emulated.

5.1.2.1 Implementation of the Program Download Application

The program download application is responsible for the downloading of NC files to the proprietary controller of the device concerned being controlled by its VMD. The application is activated automatically when the Cell Controller node is booted. Each process has its own copy of the application

together with the necessary FIXDMACS blocks forming the required command addresses. The main window for the application is shown in Figure 5.6.

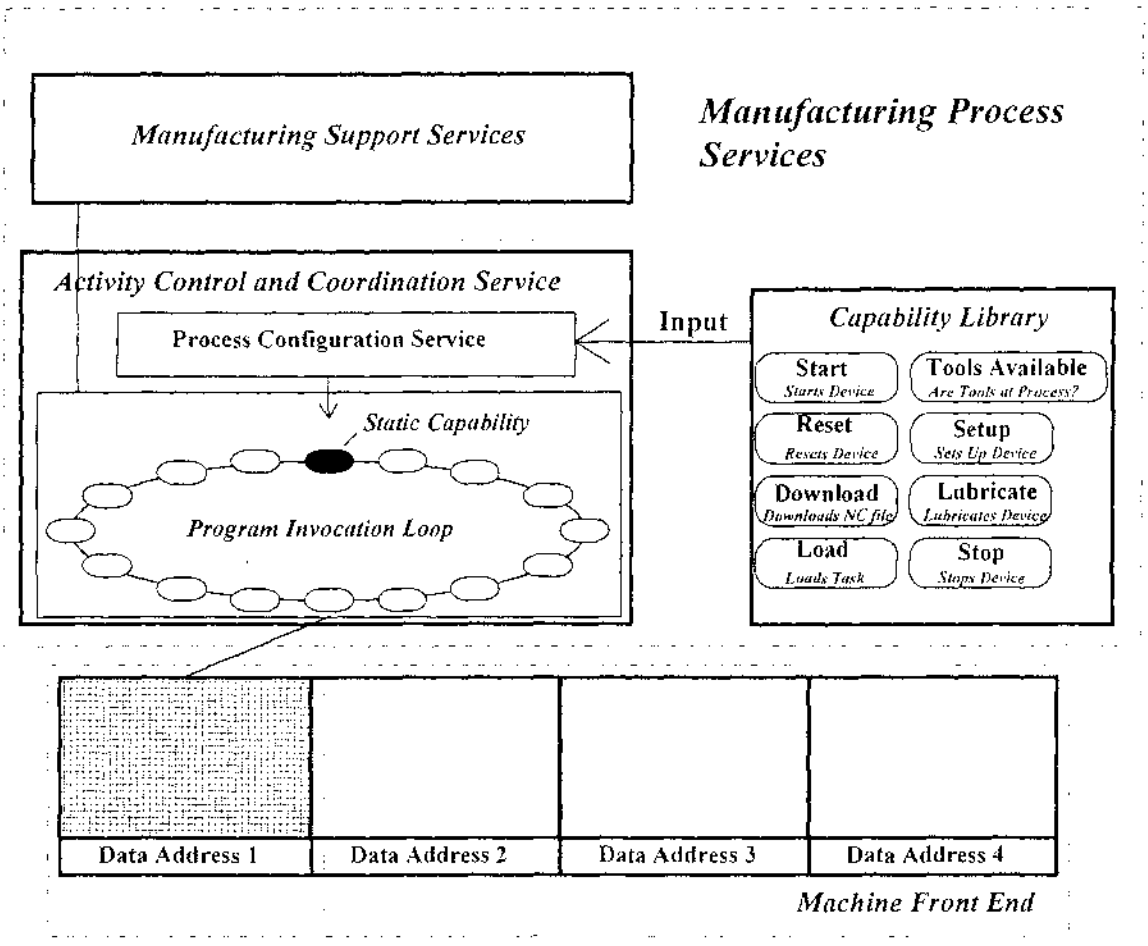


Figure 5.5 - Replication of the Manufacturing Process Integrating Infrastructure Model

When this application is loaded a number of events occur that are essential to its operation:

- A DDE link forms between the applications "Start Download" textbox and the "V1CNCSTART" FIXDMACS database block (in the case for process 1) so that the Program Invocation Loop can start NC download when required.
- A DDE link forms between the applications "Interupt Download" textbox and the "V1STOPDOWN" (in the case for process 1) so that the Program Invocation Loop can terminate NC download if required.

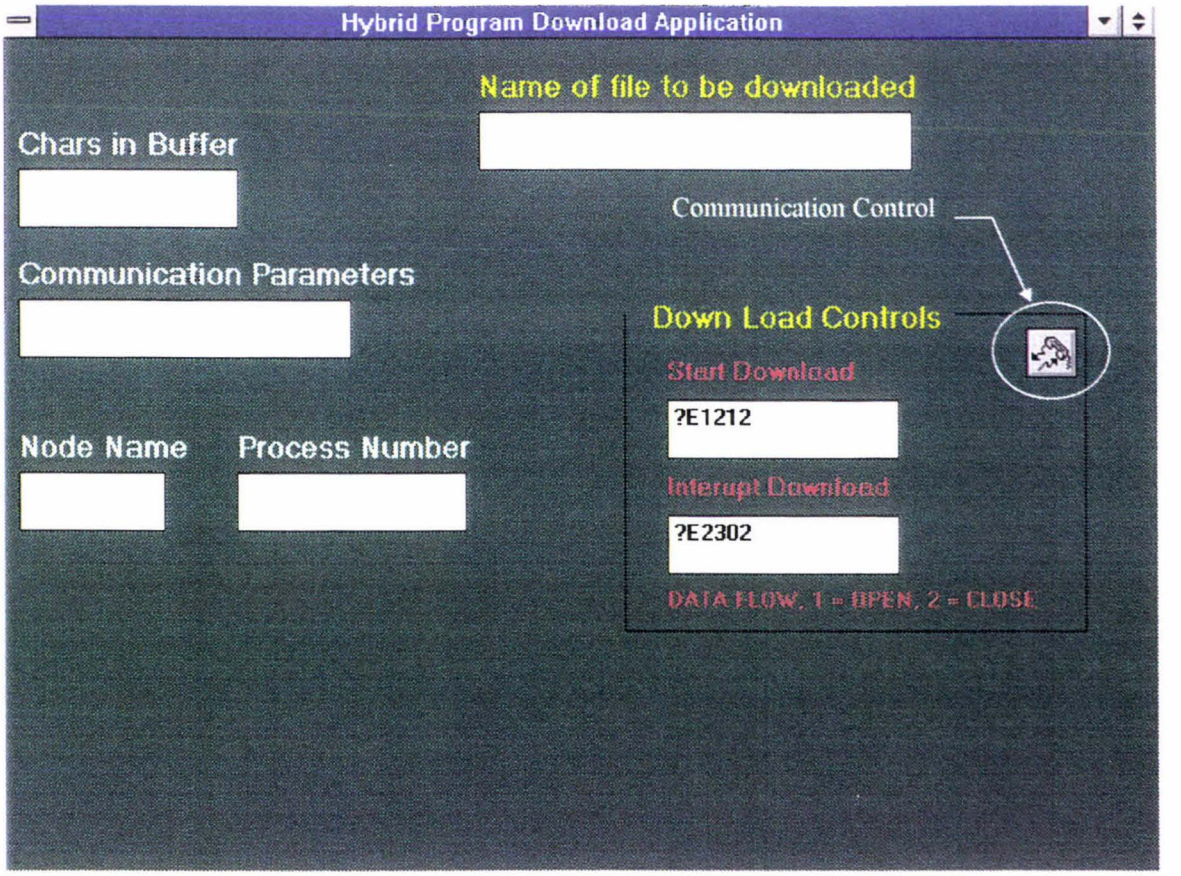


Figure 5.6 - The Hybrid Program Download Application

When the application downloads the NC files for a given process it must know what communication parameters are required to ensure successful transmission occurs. In the Factory Configuration Service these parameters must be stated when configuring a process for a given cell and transmitted to the FIXDMACS database blocks providing the functionality for the Cell Configuration Data Storage Addresses.

- Therefore, when the application is loaded it retrieves the necessary transmission parameters for the device assigned to it (using EDA), forms them into a string and displays the information (Baud Rate, Parity, number of Stop bits and data bits) in the "Communication Parameters" textbox for reference.
- ***Initiating the Download Function***

Once the application has been loaded as stated above it is ready to perform the download function when requested by the Program Invocation loop for the given device. When the Process Control Application (see later) is ready to download the required NC file for the device it transmits a "1" to the

"V1CNCSTART" FIXDMACS database block (in the case of process 1) and this changes the value in the "Start Download" textbox from a "0" to a "1". This causes the following to occur:

- The application retrieves the name of the file to be downloaded by interrogating the FIXDMACS "VMD1NCFILE" database block.
- The name of the file to be downloaded is placed in the "Name of file to be downloaded" textbox.
- The control used to transmit information to the NC controller used is the Communication control as shown in Figure 5.6. This control is able to transmit strings of characthers as required through the port stated in the programming. Therefore the next action to occur is the downloading of the file to the required device. During the download the following occurs:
  - ⇒ The file to be downloaded is opened and 3000 string characthers are downloaded to the device. This downloading occurs using the parameters specified in the "Communication Parameters" textbox.
  - ⇒ A delay of 15 seconds is then applied to ensure that the buffer is cleared before more of the file is transmitted. It was found that if this delay was not applied then overwriting in the recieving NC controller occured.
  - ⇒ The buffer is then cleared to receive more string characthers.
  - ⇒ The cycle continues until the contents of the file has been fully transmitted.
  - ⇒ When the entire file has been transmitted the user is informed and must respond.
  - ⇒ The file and the communications port is then closed and the application sends a message to the "V1DOWNCOMP" (in the context of process 1) informing the Program Invocation Loop that the download is complete.
  - ⇒ Reset the FIXDMACS "V1CNCSTART" database block to "0" so that the system is able to respond to the next request.

To maintain a degree of system flexibility the downloading of string data from the file can be terminated if required at any time. This is achieved by configuring the Program Control Application to pass a "2" to the FIXDMACS "V1STOPDOWN" (in the case of Process 1) database block in the event that this is required. As this block is attached to the "Interupt Download" textbox using a UMS Message response link the resulting change event causes the following to occur:

- A message box appears telling the user that the transmission has been halted.
- The file and the comport are both closed.
- The "V1STOPDOWN" database block is reset so that it can respond next time.

The application has very light connections with the main body of the cell infrastructure and is in effect a standalone service that is called upon when required by the cell system. As the Hybrid FMS system is

designed to have four processes per cell the cell controller PC would require four communication ports, as each NC controller device would require its own RS232 connection. Once again this modular approach to software design has proven to be very effective as it allows an application be written, debugged and duplicated a number of times, and therefore, minimising the amount of debugging required.

5.1.2.2 Implementation of the Queueing Application

The Cell Controller Queueing Application is responsible for the handling of tasks requiring operations at a given process. The application is replicated enough times so that one copy is available for each process within a cell. The application is responsible for the following actions:

- The queueing of tasks required to be processed.
- The queueing of tasks that have been processed and are required to be scheduled at other processes for further work.
- Storage of time required to process work in a processes queue.
- The downloading of task data to the VMD prior to the processing of the task.
- The adjustment and recalculation of the amount of work (time) required to be done by a given process during the dynamic stages of its existence.
- The addition of tasks to the incoming queue in the correct position when tasks have been allocated to a given processed.
- The display of task details relevant to the task being processed on an item of equipment at any given time.
- The display of details required for successful setting up of the associated process to complete a given task.
- The logging and display of data relevant to the past history of the device being serviced. The data includes information such as times tasks were added to and taken off queues, average incoming queue length over time and when tasks were started by the process involved and when they were finished.
- The holding of an auction in the event that the asssociated process has been polled by the Auction Manager and there are tasks present in the out going queue that require scheduling (see earlier section).
- The transfer of tasks from the in going queue to the out going queue in the event of process failure and subsequent redistribution.

As can be seen there is a wide variety of functions that are required to be performed by the queueing application. These will be explained in more depth in the following work. The application itself would be fired along with all the other applications required for successful operation of the cell by the means of a batch file.

The main applications window is shown in Figure 5.7 below:

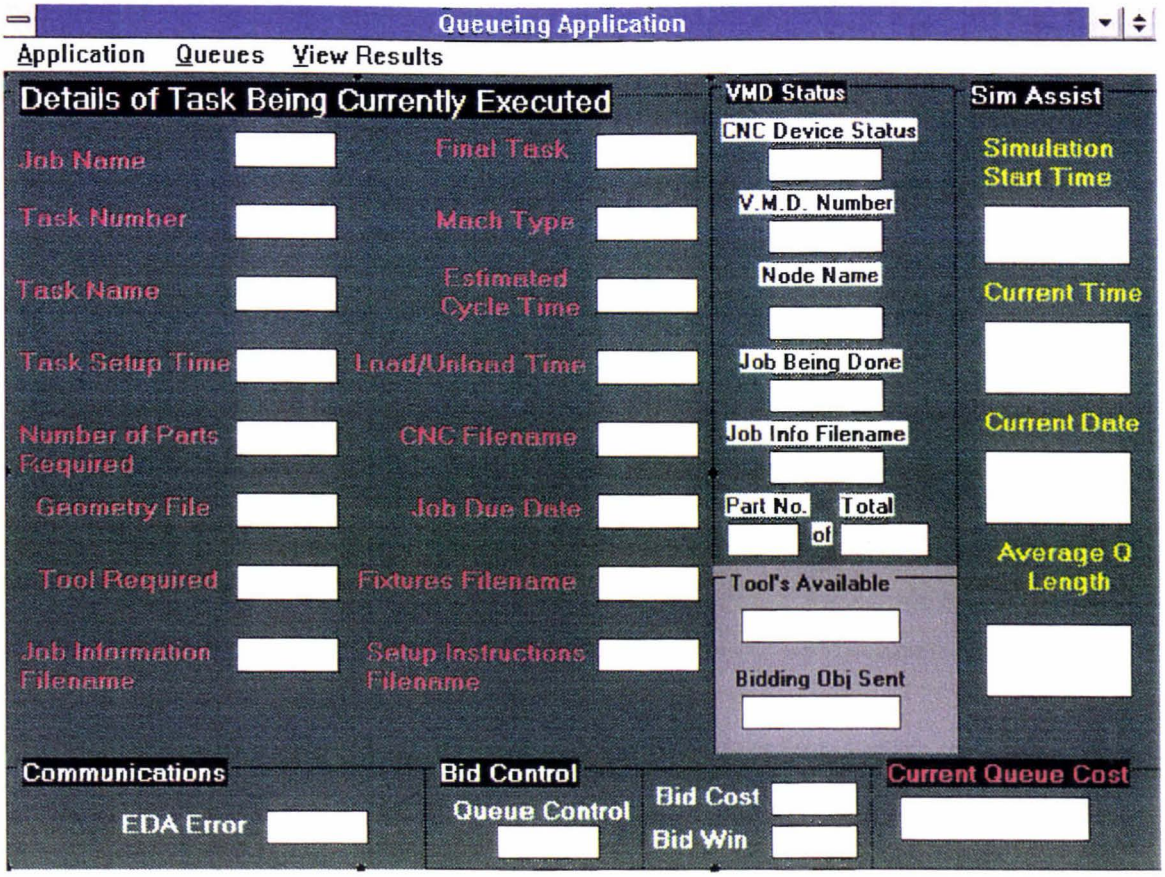


Figure 5.7 - Queueing Application Main Window

When this particular window is loaded a number of important things occur:

- The Queue_Status window which shows the status of the in going and out going queues for the process associated with the copy of the application is displayed (see Figure 5.8). This form also has a wide variety of other functions which will be explained further on.
- The file holding the contents of the in going queue ("C:\Cellcont\Queue\In queue\V1"In queue.txt" - for process 1) is opened and the tasks to be carried out are extracted and written to the In queue list as shown in Figure 5.8.
- A number representing the quantity of tasks in the in coming queue is transmitted to the cells Task Data Address (FIXDMACS "V1INQUSIZE" database block for process 1) using a Local Message Transmit UMS so that the Program Invocation Loop can keep track of how many tasks are to be undertaken for a process and behave accordingly.

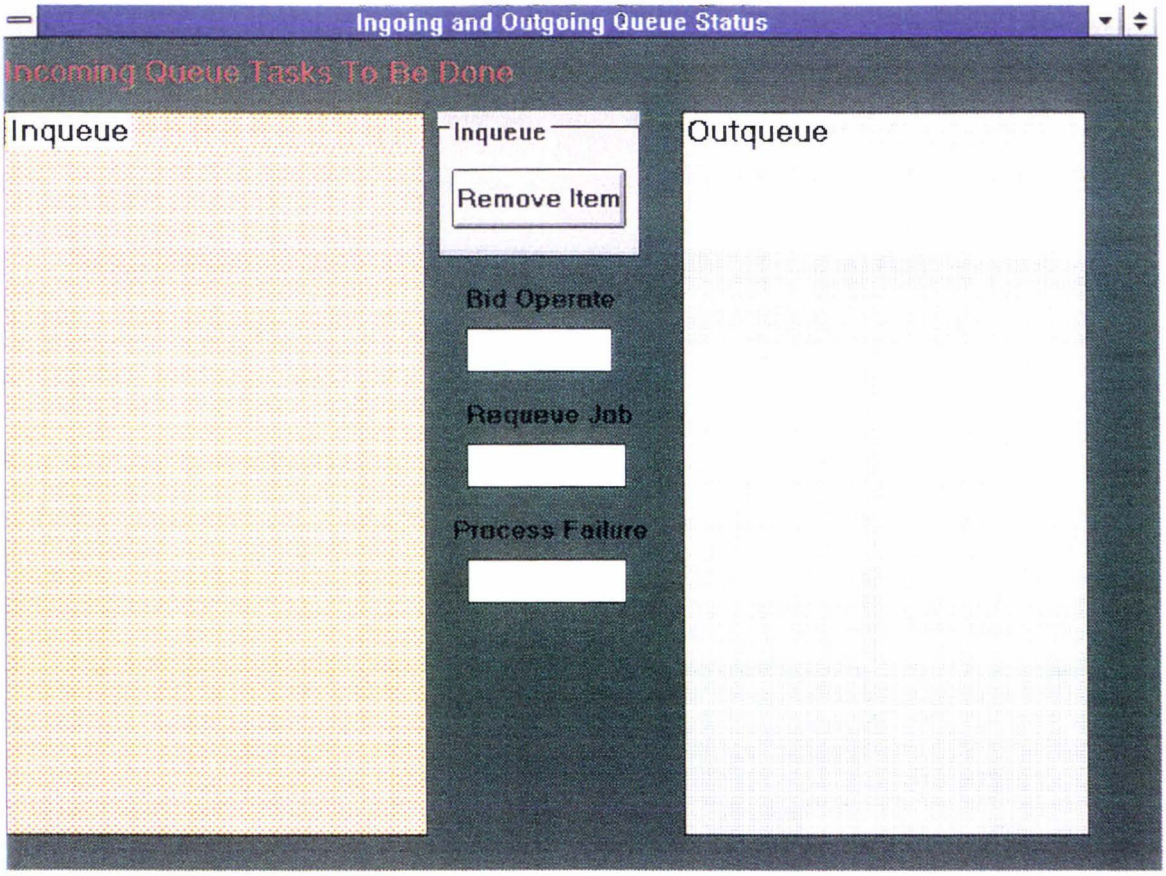


Figure 5.8 - Queue_Status form

- A local file is then opened ("C:\Cellcont\Queue\Queucost.cst") and the time required for the device to complete all tasks in its incoming queue is retrieved and written to the "Current Queue Cost" textbox (see Figure 5.7).
- An automatic DDE link (Message Response UMS) is established between the "Bid Win" textbox (see Figure 5.7) and the FIXDMACS "BIDWIN" database block (system address) so that the application is able to be notified of the awarding of a task after auction by the Factory Controller.
- Another link is created between the "Queue Control" textbox and the "V1QCONTROL" FIXDMACS database block (for process 1). This allows the Program Invocation Loop to request certain services of the Queueing Application.
- Further links are created between the "Part No." textbox and the FIXDMACS "V1CURRPART" database block (for process 1) and the "Simulation Start Time" textbox and the FIXDMACS "STARTTIME" database block. The link associated with the "Part No." textbox allows the VMD to inform the Queueing Application when it has completed an individual cycle or all of the cycles making up a task. This allows the Queueing Application to update the queue costs as each cycle is finished as well as other functions (see later). The link associated with the "Simulation Start Time" textbox allows the Factory Controller to send a message to the FIXDMACS "STARTTIME" database

block when the system starts up. This causes a change event to occur in the "Simulation Start Time" textbox which initialises the data logging function (see later).

- The Results form is then loaded (see Figure 5.9). This form is used to record specific events that occur and can be monitored by the user.

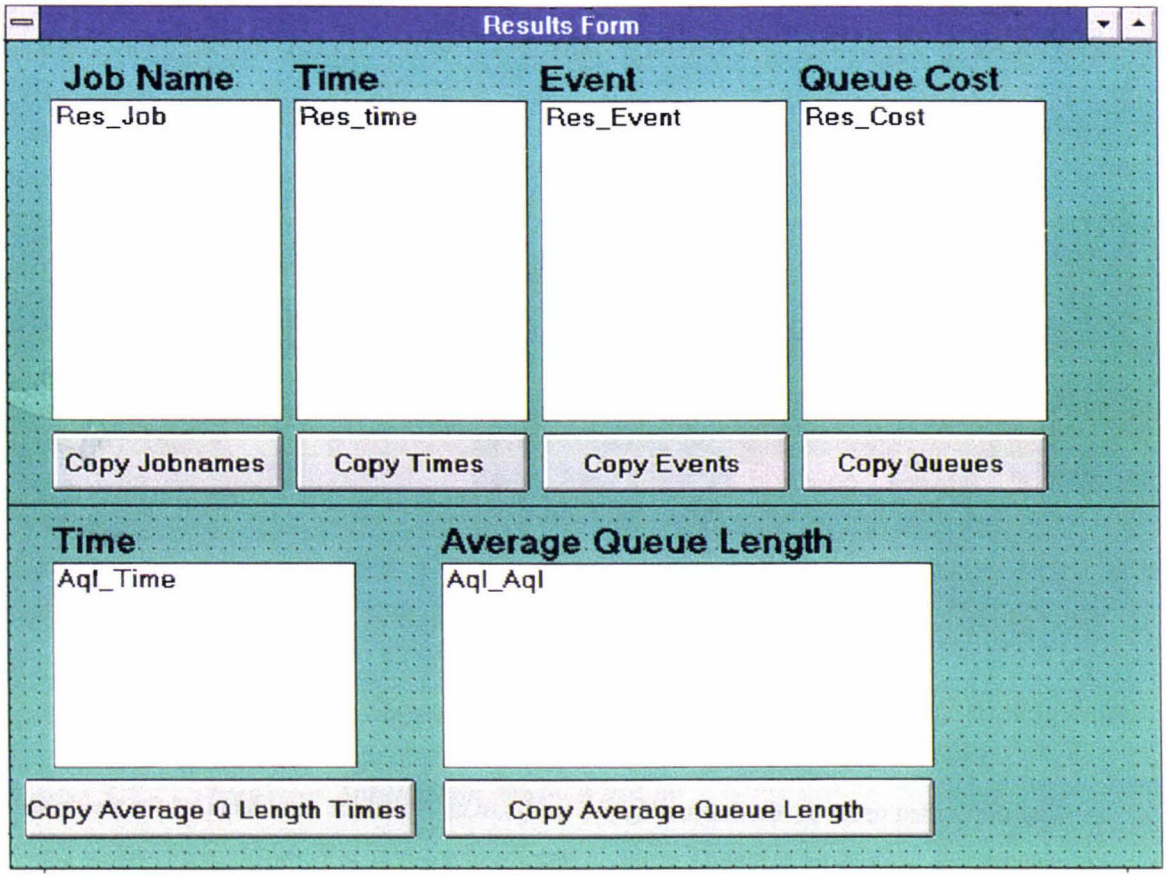


Figure 5.9 - The Results form is used to log events in the system that are important to the user

- *Queueing of New Tasks*

The implementation of the auctioning process (Task scheduling) has been discussed earlier so will not be repeated here. At the end of the auctioning cycle the Factory Controller awards the bid to the winning process by sending the number of the process to the cells FIXDMACS "BIDWIN" database block. The Queueing application "Bid Win" textbox of the winning process responds in the following way:

- The "Netdrive & "Factory\Bidding\Bidfile.bdf" (bid object file) is opened and the task details are retrieved from it. The task details include Information Filename, Urgency status, Number of Cycles required to be carried out, the task set uptime, load and unload time etc.

- The event is logged in the results form by placing the name of the task, the event (ie: Task was added) and the time the event occured in the appropriate list boxes.
- If the task is urgent and there is no other job in the queue then the task is simply added to the queue. If the task is urgent and there is a number of tasks in the queue then the new task is added to position two on the list. This is because the first task may have reserved a tool and the tool maybe in transit and the action of putting the new task at the top of the queue would tie the tool up. If the task is not urgent it is placed at the bottom of the queue.
- Once the task been added the file holding the inqueue task list is updated as is the database.
- The time required to complete the entire task is calculated and the total cost for completing all tasks in the in going queue is recalculated and displayed in the "Current Queue Cost" textbox.
- The file holding the queue cost is opened and the records updated to reflect the addition of the new task.
- The Cell Controller then sends a flag to the Factory Controller FIXDMACS "BidAckNowl" database block to acknowledge the award.

- ***Downloading of Task Data to the MFE Task Data Address***

The downloading of task data to the appropriate MFE Task Data Addresses is of crucial importance as it is information that is vital to ensure successful completion of the task being carried out. The Process Configuration Application is responsible for the firing of the FIXDMACS Program Blocks making up the Program Invocation Loop and as such there will come a time when the task data download sequence will be fired. As there is a Message Response UMS DDE link between the FIXDMACS "V1QCONTROL" database block (for Process 1) and the "Queue Control" textbox this becomes the channel for the download activity.

When the task information download is required the Program Control Application will initiate the FIXDMACS "V1DOWNTASK" database block (for process 1) which will in turn change the value of the "V1QCONTROL" database block to a "3". As such the value of the "Queue Control" textbox changes to a "3" and a change event associated with that textbox causes the following to occur:

- The name of the task/job to be carried out is read from the top of the in going queue.
- Prior to the removal of the task from the queue, the event calls a procedure that calculates the average volume of work in the queue (in terms of time required to carry out all tasks) from the time that the system was started until the present time. This information (and the time) is recorded in the respective lists within the Results form (see Figure 5.9). This information can be copied to a statistics package and the system performance can be analysed.

- Information such as the name of the task/job to be manufactured, the fact that the task is about to be started and the current time and queue cost, is added to the results form.
- The file holding the job information is opened from the network and the current tasks information is retrieved from the file (Set uptime, cycletime, CNCfilename etc). The system then interrogates the job status (by looking at the task pointer) and if the task is the last task to be done for the job, the pointer is reinitialised to point to the first task (for when a new batch of the same job is required in the future). Otherwise if the task is not the last task the pointer is incremented to point to the next task when the file is opened so that the next task can be auctioned.
- The job information is then written to the relevant "Details of the Task Being Currently Executed" textboxes (see Figure 5.7) and then transmitted to the relevant MFE Task Data Addresses. The task data is displayed in the textboxes so that the user is able to see the details of the task currently being processed, whereas the data is transmitted to the data addresses so that the VMD is able to use the information to process the task.
- The job is then removed from the incoming queue of the relevant queueing application and the updated queue is then saved to a file.
- Two other pieces of information are then sent to the MFE Task Data Addresses - these being the updated in queue size and a flag which tells the VMD that all the data has been downloaded and the processing can begin. The FIXDMACS database block that receives this flag is the "V1DOWNCOMPL" (for process 1) block.

This downloading of the task data will only occur when all other states within the devices VMD are met. That is - it has recieved the required tool to carry out the task, the manufacturing process is idle, the process is currently active etc. The full flexibility of the system is apparent in that all actions are modular and distinct. Ultimately the configuration of the VMD (as defined by the Process Control Application) will be dependent on the manufacturing process being integrated.

- *Tracking the Process of a Task*

One of the major functions of the queueing application is to track the process of a task and behave accordingly. As was mentioned earlier a link between the "Part No." textbox and the FIXDMACS "V1CURRPART" database block (for process 1) was established upon the loading of the application. As the VMD finishes each cycle of the task it increments the "V1CURRPART" database block by one to indicate that another cycle has just been completed. This incrementation is used to update the "Part No." textbox in the queueing application and the application responds in the following way:

- If a cycle has been completed but it is not the last cycle, the cost of the cycle completed is subtracted from the queuecost and the queuecost is therefore updated. The result form (see Figure 5.9) is then

updated showing the fact that a cycle is complete, the name of the job, the time and the updated queue cost.

- If the cycle just completed is the last cycle for the task the application responds by calculating the new queuecost given the task is finished and by updating the queuecost. The application also sends the identification code for the tool that was used to the Factory Controller so that it is able to release the tool to be rescheduled for another task.
- If the cycle just completed is the last cycle for the task and the task is the last task required to be carried out for the job the following additional actions occur:
  - ⇒ The result form is updated to show that the job is completed, the time the job was completed and the name of the job. A Global Message Transmit UMS is then sent to the Factory Controller "UPDATEMTBC" FIXDMACS database block. This enables the Factory Controller to calculate the Mean Time Between Completion for all jobs in the system (see next chapter). As a result the systems performance can be monitored.
- If the cycle just completed is the last cycle for the current task but there are further tasks to be carried out before the job is complete the following actions occur:
  - ⇒ If the job is urgent the name of the job is placed at the top of the out going queue in the Queue_Status form (see Figure 5.8). If it is not urgent the job is placed at the bottom of the queue.
  - ⇒ The result form is updated to reflect these actions.
  - ⇒ The updated out going queue is saved to file.

Using this method of tracking the Queueing Application is able to monitor the progress of a task within the VMD environment and respond accordingly. Once a task has been completed the Process Control Application reinitiates the control loop again to start the processing of the next task.

- ***Responding to Process Failures/Long term Maintenance Issues***

Potentially the possibility occurs whereby a process can fail. This could be a minor problem, for example, when a tool fails or in the event of a power failure. The possibility of major equipment failure can cause the process to be idle for a long period of time. Under this scenario the decision may be made at the Factory Controller Level to redistribute the tasks on the in going queue of the process to other suitable processes within the system due to the length of the delay and subsequent job tardiness. To carry this out the Factory Controller Factory Configuration Service is able to send a message to the FIXDMACS "V1MACHTYP" database block (in the case of process 1) at the appropriate cell. This action causes the following actions to occur:

- As a link exists between the "V1MACHTYP" block and the "Process Failure" textbox, the change event is able to respond in the following way:

⇒ The names of all jobs requiring tasks to be undertaken are removed from the processes in going queue and placed on the out going queue.

⇒ The in going queue file is deleted.

⇒ The new out going queue status is saved to the file.

Each time the cell receives the bid token an attempt is made to reallocate the jobs to other potential processes within the system using the normal task scheduling procedure as mentioned previously.

- The process is rendered unable to compete for any future jobs being scheduled until it is in an operable state.

Ultimately a decision over the length of time required to repair/maintain a given process and whether or not the redistribution of the jobs in its in going queue is required will be dependent on the policy of the company. The facility for achieving this is available and its effect will be demonstrated in the next chapter.

- *Viewing the Process History of Activity*

One of the most useful functions of the queueing application is its ability to log the history of the process being carried out (the actual function of this activity has been discussed previously). This logging function is shown in Figure 5.10 which shows a portion of the results gathered in the testing of the system (see next chapter). Information is written to the result form any time a major event takes place along with the type of event, the time the event occured, the job involved and the calculated average queue length for the process's queue. This information is very useful as it enables the overall system to be analysed as each process has its own log as does the Factory Controller. Originally attempts were made to transmit this information directly to Microsoft Excel using DDE (Dynamic Data Exchange) and although the technique proved successful the computers used during the testing phase lacked the resources to operate all the applications at once. Running Norton System Watch software verified this. This caused some of the Cell Controllers to crash during testing. This approach was abandoned and the simple result form was created as shown. Selecting the copy button at the bottom of each column however enabled the contents of each list to be copied to the systems clipboard, from the clipboard the information was pasted directly to Microsoft Excel as required. In reality the best way may have been to write all the information to an array and then to a file for each process. The only problem with this, however, is that the information would not be able to be viewed dynamically.

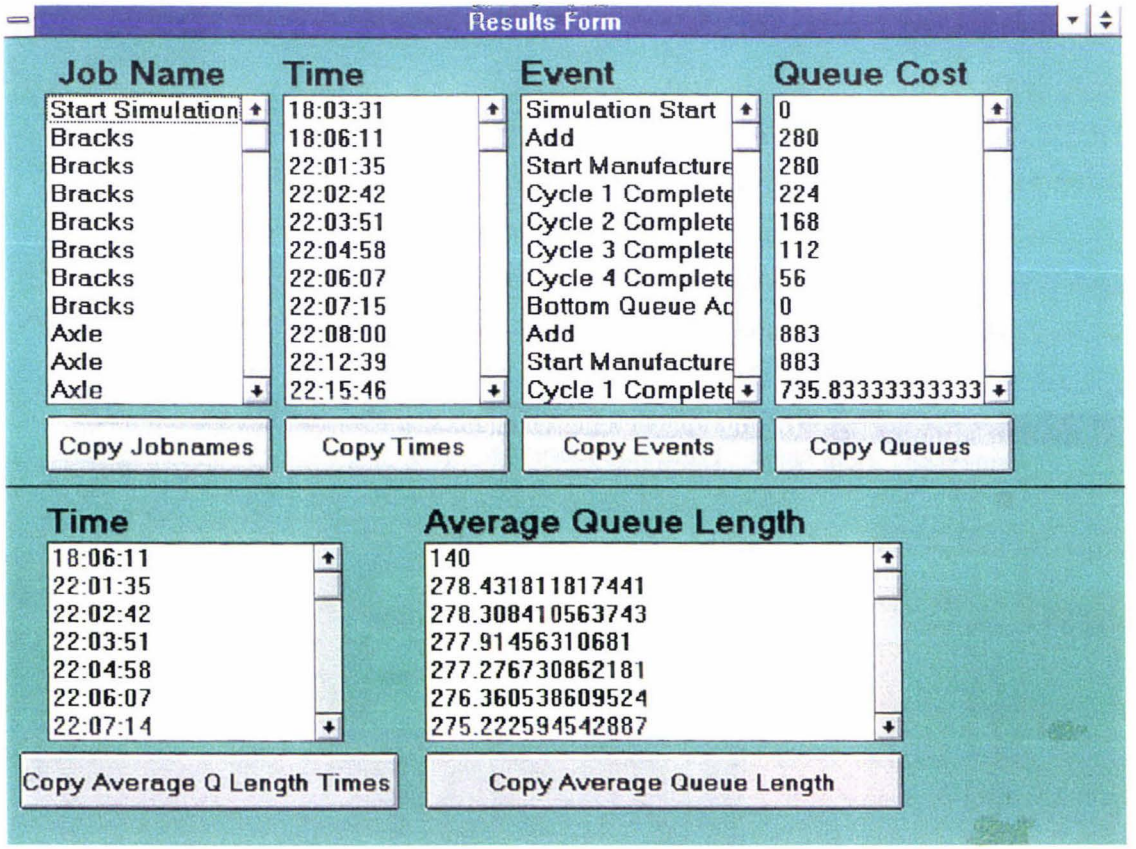


Figure 5.10 - Results form showing the results of a Simulation Run

5.1.2.3 Implementation of Activity Control and Coordination Service - The Process Configuration Service

The physical behaviour of the process within the cell is modelled by the Virtual Manufacturing Device (VMD). As has been mentioned previously when a new process is being configured a *Process Behaviour Algorithm* is developed to describe the generic behaviour of the device. From this the user will select and order the FIXDMACS program blocks required to emulate the algorithm. The Activity Control and Coordination Services Process Configuration Service is responsible for providing the 'engine' for the VMD and for the initiating of the FIXDMACS program blocks to emulate the control logic of the *Process Behaviour Algorithm* The Process Control Application (P.C.A) as shown in Figure 5.11 is responsible for carrying out the function of the Process Configuration Service and allows the following to occur:

- Allows the user to select and order from a list of available capabilities (FIXDMACS Program blocks) those which are required to emulate the behaviour of the real device.
- The P.C.A. archives the capabilities chosen and records the order in which they are to be executed as defined by the user.

- The P.C.A. initiates each capability and awaits a response from the VMD so that the next capability can be initiated.
- The P.C.A. executes each block to emulate the Process Behaviour Algorithm and create the Program Invocation Loop.

The flexibility in the system allows the manufacturing processes behaviour to be modelled by ordering the available blocks to suit. Any blocks which are needed and are not available in the system can be created and added to the capability library as required and then invoked by the Process Control Application.

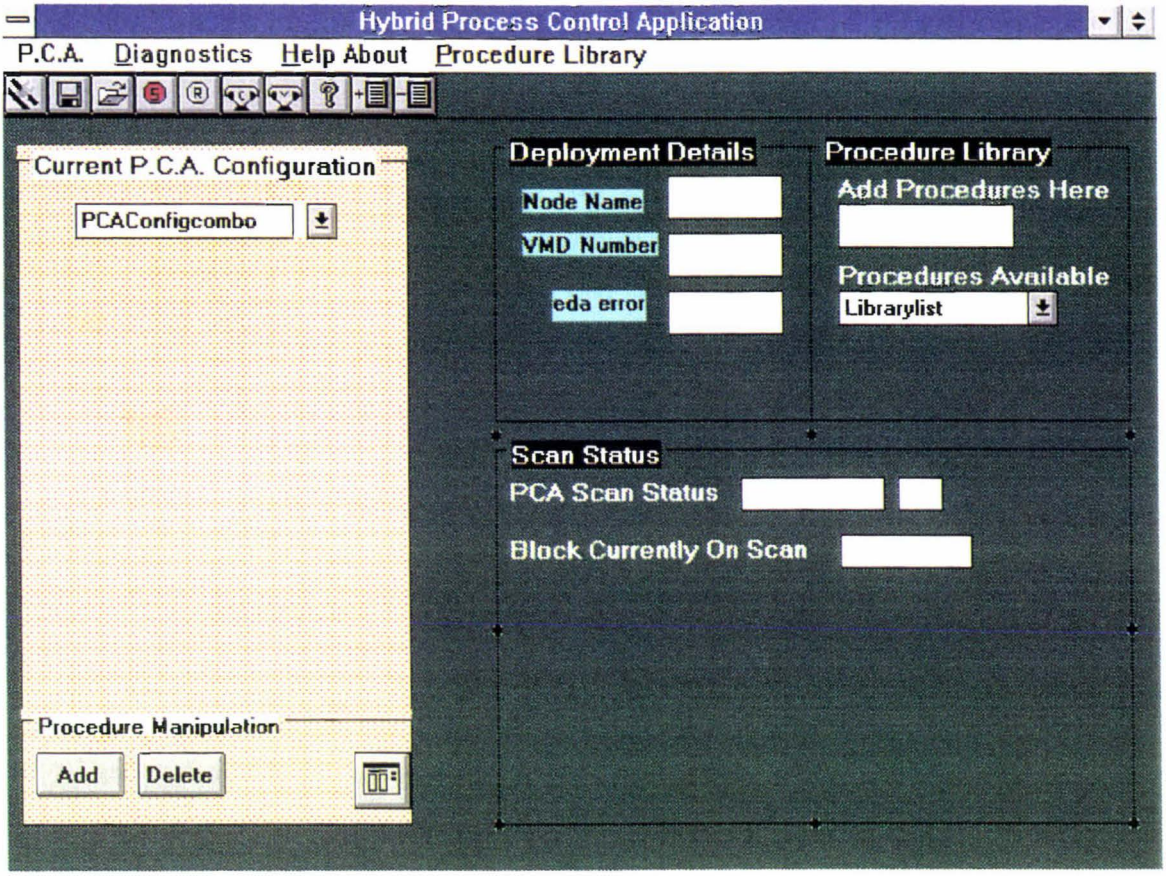


Figure 5.11 - Process Control Application

When the application is loaded all the names of currently defined capabilities are retrieved from a local file ("C:\ V1Proclb.txt") (for process1 on that cell) and are written to the "Procedures Available" list. Also a Message Response UMS is created between the "PCA Scan Status" textbox and the FIXDMACS "V1SCANSTAT" database block(for process 1). Finally the status of the VMD is set to "IDLE".

- ***Creating/Customising Capabilities for the System***

Most of the capabilities (FIXDMACS Program blocks) making up any VMD would in reality be supplied for the supplied system ie: "V1FINCYCLE" is always used to loop the Program Invocation Loop back to

the start of the logic (in the case of Process 1) so the next cycle of a current task can be started or another task in the in going queue loaded. Examples of other existing capabilities are "VMD1START" (starts the processing of a task), "DOWNLDCNC1" (Downloads an NC file to the process controller) and "V1ISREADY" (tells the operator that process 1 is ready to start a task). However there is always the potential situation whereby the uniqueness of a process dictates that other capabilities will be required and are not part of the standard issue. This is where the true flexibility of the system makes its mark. After the definition of the Process Behaviour Algorithm the user is able to order the capabilities required to emulate the algorithm and create extra capabilities for that system as required. This is simply a matter of creating a new FIXDMACS Program Block within the database builder module (along with the required input/output blocks), programming it to perform the function required and then placing it in the appropriate order along with the other capabilities to create the Program Invocation Loop.

- ***Defining Capabilities within the Process Control Application***

As has been defined the application loads all capabilities that are currently available into the "Procedures Available List". From the process behaviour algorithm the user is able to decide what capabilities are required and the order in which they should be initiated in order to construct the Program Invocation Loop part of the VMD. Any capabilities not part of the standard issue as shown in the list can be created as FIXDMACS Program Blocks and coupled with suitable input/output blocks so a connection can be made with the real device via the I/O Driver. Once this is done the user is able to add any additional capability to the "Procedures Available List" by typing the name of the FIXDMACS Program block into the "Add Procedures Here" textbox. As each Program block name is added to the list the system automatically saves the updated capability library to a file. All capabilities that are defined this way and are in the list are able to be encorporated into a Program Invocation Loop.

- ***Defining the Program Invocation Loop***

The "Current PCA Configuration" listbox as shown defines the order in which the capabilities are to be initiated to emulate the process behaviour algorithm and create the Virtual Manufacturing Device logic. To add capabilities to the "Current PCA Configuration" listbox so that they can be started in the desired order, the following procedure is followed:

- If there are no capabilities present in the listbox the user selects the last capability that is required in the Program Invocation loop (will always be the "V1FINCYCLE" FIXDMACS block) as selected from the "Procedures Available" listbox. If the "Add" command button is selected then the procedure will then be added to the "Current PCA Configuration" listbox.
- Once a number of capabilities have been ordered in the "Current PCA Configuration" list each extra capability is added by selecting the capability to be added from the "Procedures Available" list and

then the capability in the "Current PCA Configuration" after which the new capability is added. Finally the "Add" command button is selected to complete the task. This process is continued until all capabilities are in the correct order and have been arranged to emulate the logic of the *Process Behaviour Algorithm.*

- If a capability is added in error it is selected and then the "Delete" command button is pressed. This will remove the capability from the list.
- The configuration can then be saved to its own particular file for the given process via the common dialog control. This allows the saving of multiple configurations on the one particular cell or on the network. This situation is ideal as it allows the user to develop a configuration on one cell, save it to the network server and then download the same configuration for processes on other cells having the same *Process Behaviour Algorithm* if required.

- ***Executing The Program Invocation Loop***

As has been discussed in earlier work and as illustrated in Figure 5.5 the Process Control Application is responsible for the organisation and execution of the capabilities in the correct order to emulate the Process Behaviour Algorithm constructed for the process concerned. The capabilties then interact with the Front End Service to create the Virtual Manufacturing Device which is mirrored onto the real device via its logic controller.

As such communication needs to be established between the Process Control Application (PCA) and the FIXDMACS Program blocks implementing the capability role. This relationship is required so that the PCA is able to direct the capabilities to perform the given required functions and receive feedback as needed.

To start the Virtual Manufacturing Device the user selects PCA and Start from the menu or F1 or the red start button on the tool bar (see Figure 5.11). These actions initiate the following:

- A check is made to make sure the PCA is not already running. If it is the user is informed and the procedure exited.
- A check is made to ensure that a valid configuration has been loaded. If it has the user is asked if they want to continue. If the answer is yes then the next stage continues.
- The procedure then takes the name of the first capability from the Current PCA Configuration listbox and places the FIXDMACS Program block of the same name on scan by using a Local Message Transmit UMS. This is implemented by sending an "On" message to the relevant Program Blocks "A_SCAN" field using EDA. This action causes the code within the block to execute the required logic. The name of the capability being executed is then displayed in the "Block Currently on Scan" textbox.

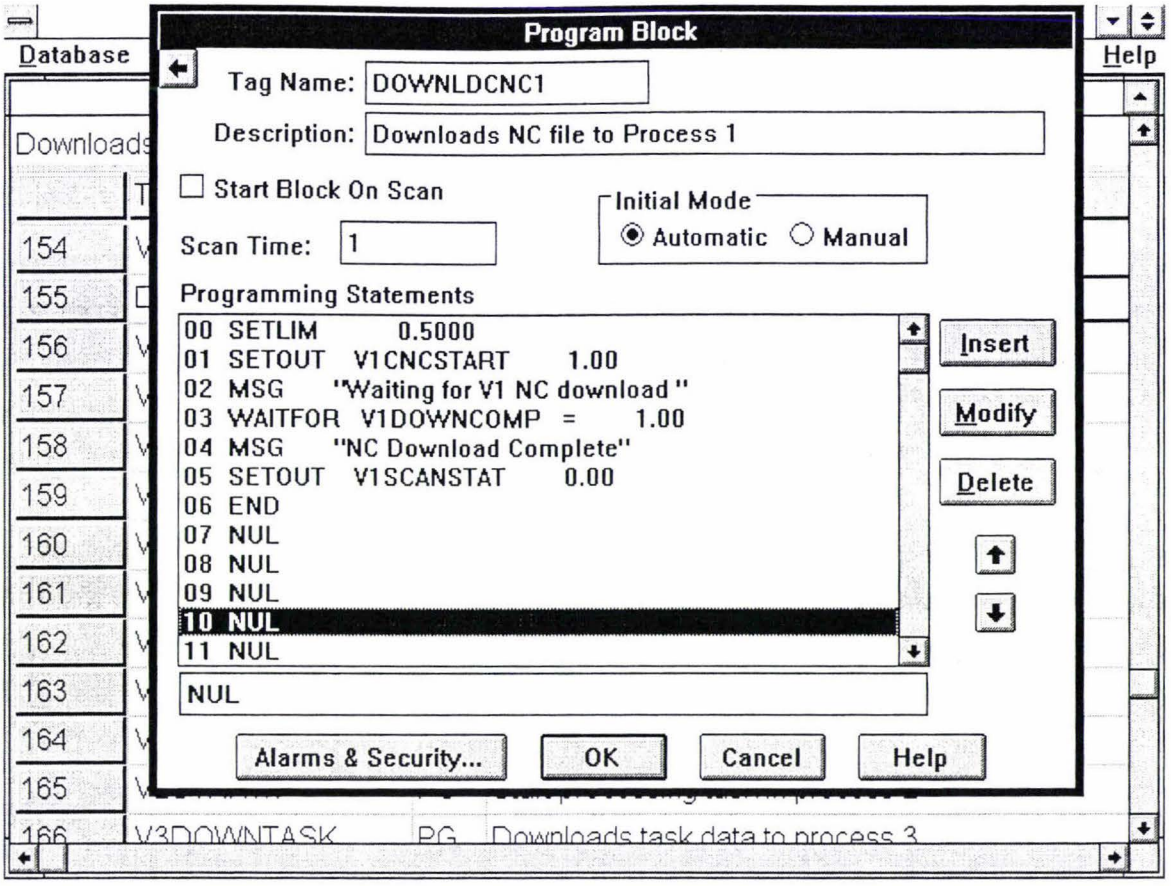


Figure 5.12 - Example of Capability Implementation using FIXDMACS Program Block

- Once the Program Block is placed on scan it is free to implement the logic programmed within it as shown in Figure 5.12 (example as for the "DOWNLDCNC1" program block). As can be shown in the example the block interacts with the Program Download Application by requesting its service, waits for a response to the effect that the download is complete, sends a message to the operator that the download is finished, changes the "V1SCANSTAT" database block to a "0" and then places itself off scan.
- As a Message Response UMS DDE link exists between the FIXDMACS "V1SCANSTAT" database block and the Process Control Application's "PCA Scan Status" block the application then recognises that the capabilities function has been completed.
- Once the capability has been executed the "PCA Scan Status" textbox turns to a zero as a response to the events described. In response to this the PCA takes the name of the next capability off the list and fires the next FIXDMACS Program Block representing the functionality of the capability and the cycle repeats.

There are a couple of variations of the above cycle or the implementation of the Program Invocation Loop:

- When the last capability in the list is implemented the first capability at the start of the list is executed again.
- In the event that there is a pause initiated within the PCA by the operator the current capability will continue in an active state until it is finished and when the system is restarted the next capability will trigger. This emulates the situation whereby the operator may want to manually stop the system as required in the event that manual intervention is required ie: coolant level is too low and needs topping up or he wants a break. This is initiated within the PCA by clicking the "R" button in the toolbar or from the menu.
- In the event that a major failure has occured (eg: a workpiece has been irreparably damaged due to equipment failure) the operator will hit the emergency stop button on the FIXDMACS view screen of the device. This action will carry out two functions:
  - ⇒ Instantly stop the machine to prevent further immediate damage.
  - ⇒ Send an "IDLE" message to the "PCA Scan Status" textbox via the "VISCANSTAT" database block. This will reset the PCA to start at the beginning of the loop once again.

The Process Configuration Application as discussed above is a reasonably straight forward application that is very flexible in accomodating a wide variety of processes. The application can be transported to a number of cells and be used to support a wide range of different processes due to this flexibility. Capabilities can be selected from the current library and used to construct routines emulating real world process behaviour. Any additional capabilities required can be built within the FIXDMACS database builder function and added to the library as needed.

5.2 Summary

This chapter has dealt with the implementation of the major aspects of the services restricted to specific Cell Controller activities. All major aspects of the developed system have been dealt with and explained in a reasonable level of detail so that an idea of how the various features are integrated into the overall system. An in depth explanation of the model was carried out in an earlier chapter, however, a comprehensive explanation as to how the model was implemented was also required. This has been attempted in this chapter as well as those previous.

The next chapter (Chapter 6) will detail the tests carried out on a 3 Cell Controller, Factory Controller configuration and will prove that the overall functionality of the system is in place. It will clearly demonstrate the major strengths of the architecture and show the system to be fully functional.

Chapter 6

Testing the Hybrid FMS System

6.1 Introduction 187

6.2 Hybrid System Configuration 188

6.3 Experiment 1 - Standard System 193

6.4 Experiment 2 - Process Failure on Cell 1 196

6.5 Experiment 3 - Buffer Management 198

6.6 Experiment 4 - Standard test with variable Intercellular Transit times 200

6.7 Conclusions 204

6.1 Introduction

Once the system was programmed to emulate the behaviour of the model as presented in Chapter 2 and its behaviour had been validated it was important to ensure that the claims made by the researchers as mentioned previously were verified, and to observe the behaviour of the overall system. The main aspects that need to be investigated were the following:

- That the system emulated the model and that no problems existed, especially with the timing of the UMS communications over the network.
- One of the claims as defined by Smith[55] and Shaw[39,40] was that distributed decisionmaking systems are easy to extend. It was found in this research that once a basic two Cell Controller, one Factory Controller configuration was up and working it was a simple matter of copying the modular software to a third Cell Controller. This resulted in a 2-3 hour delay to get the third Cell Controller integrated to the point where the system functioned correctly. The predicability of the software and the loose coupling of the cells with the rest of the system made this a simple exercise. Debugging was not required as no reprogramming was necessary because of the modular nature of the system.
- Smith *et.al* claims that systems with distributed control and encorporating negotiation based scheduling, tend to exhibit a high degree of reliability[55]. Tsukada in his work also showed that the negotiation approach to scheduling was very useful in the event of a process breaking down[49]. As a result one of the tests that needs to be carried out is to deliberately crash a process and to witness the rescheduling of the tasks to be done by that process. One important aspect that needs to be observed is the effect on the overall system as well as its ability to recover.
- Shaw[48] in his work stated that one of the best features of the negotiation system was its ability to evenly load similar processes over different cells using the Earliest Finishing Time (EFT) Heuristic. He showed that by using this heuristic system performance in terms of number of late jobs, average waiting time and average lateness outperformed the other systems tested. With the Hybrid system it would be interesting to determine whether the same conclusions could be drawn.
- Buffer Setting - The Hybrid system allows for the setting of maximum inqueue sizes. The effect of this policy was also tested to see what influence it had on the overall system.
- Setting of Intercellular Travel times. The effect of intercellular distance on job completion times was investigated.

6.2 Hybrid System Configuration

Shaw[32] has identified three types of cellular arrangements that exist in the FMS environment all having different configurations required to give varying aspects of functionality.

- Flexible Cells - where general purpose machines are used and the set up is flexible for performing a wide variety of tasks.
- Product Orientated Cells - where a certain type of product is manufactured explicitly in that cell.
- Robot assembly cells - where robots are used for putting subassemblies together.

It has always been the intention of this research to provide a system that is as flexible and as modular as possible (system flexibility) and also able to manufacture a wide variety of products (product flexibility) and as such the system was set up to follow the flexible cell paradigm.

In order to fully test the real life Hybrid System it was imperative that a typical system be set up to fully exploit every degree of functionality as modelled in Chapter 2 with the resources currently available. As a result the system was configured in the following way:

- **Factory Controller, 3 Cell Configuration** - The testbed was set up as a Factory Controller, 3 Cell Configuration. This was the smallest configuration that could be used to fully validate the system and meet the testing requirements within the realms of section 6.1 above. The major limitations in this part of the research were the following:

    ⇒ Availability of Computers - The experimental and software debugging procedure required the tying up four computers for 3-4 months as no other computers were available.
    ⇒ Software Dongling/Copies - Massey University had a limited number of available licences for copies of Intellution FIXDMACS. Therefore this became a limiting factor.
    ⇒ Availability of Network card resources and connections to the existing network.

- **3 Processes per Cell** - The system was set up so that three different process types were included in the architecture. To ensure maximum flexibility within the system an instance of each process was assigned to each cell. This made sure that there were other devices available in the system which could carry out the tasks in the event that a process or cell crashed. The processes set up for the system were the following:

⇒ NC Mill - Numerically Controlled Mill
    ⇒ NC Lathe - Numerically Controlled Lathe
    ⇒ NC Cylindrical Grinder - Numerically Controlled Grinder

- **Number of Jobs/Tasks** - The number of jobs was set at 15 for the testing phase, these jobs were put through the system 3 times. This gave a total of 45 jobs processed by the system for the period of the test procedure. The maximum number of tasks was set as being equal to three for each job and the number of tasks assigned to each job was generated using a 3 random number system.
- **Task Processing Times** - As outlined in a study carried out by Ramasesh[2] on simulation in the context of job shops, the most popular choice for modelling processing times is to use the *Exponential* distribution. A mean processing time of 30 seconds was used and the processing time for each task was created using an exponential random number generator.
- **Task Set up Times** - In the same work as mentioned above Ramasesh[2] stated that the *Exponential* distribution was also the most popular distribution for modelling task set up times. A mean task set up time was set at 10 seconds and the set up time for each task generated using an exponential random number generator.
- **Load & Unload Times** - As Ramasesh[2] states in his research, operation times or any factor combined with other factors making up the operation time are generally treated as being *Exponentially* distributed. As such load and unload times were treated as being one in the same and were created with a mean of 10 seconds.
- **Number of Cycles per Job (Batch Size)** - the range of batch sizes available for the simulation for each task was 1 - 10. Originally a maximum number of cycles of 5 was applied for each task and a random number generator used to select the actual number of cycles for each job. It was found in a test run, however, that the queues did not build large enough so as to be significant. For all jobs a random number generator was used to generate the specific number of tasks for each job.
- **Assignment of Process Type to each Task** - For each task that was created the job required to carry out the task was assigned randomly from those stated above.
- **Assignment of Tools to each task** - A pool of tools was set up at the Factory Controller Resource Configuration Service. Tools relevant to each process assigned to a given task were selected at random from that group during the task definition stage. When the Factory Configuration Service was initiated each cell was configured so that all tools required by the jobs being run through the system were available (that is if they were qualified to use them). This negated the possibility that a job may have been refused in the event of all processes not being eligible to use the required tool.

The resulting job/task configurations are shown in Appendix E.

There were a number of prerequisites for the testing procedure:

- The process type for each task defined had its capacity rating kept low so that every process of the required type qualified to carry out the task. This would restrict any bias in the results.

- Each job visits a process only once in its production life cycle. Therefore each task making up a job was executed on a different process. This is typical of a job shop application.
- The fifteen jobs were released 3 times and in the same order. The in going queue at the start of the application consisted of the first 15 jobs defined in Appendix E, followed by the same 15 jobs again and then the same jobs again to make a total of 45 jobs. The job names of the second and third repetition were changed slightly to aid traceability.
- Transit time was allowed for in the scheduling of tasks. In some of the tests carried out the transit time was set as one second between all nodes. However, this was varied in other tests to measure the effect on the overall system (see later).
- For simplicity the tool scheduling application was left out of the equation.
- The behaviour of the Virtual Manufacturing Device was simulated as some of the necessary features such as the real input/output drivers were not available at the time of testing.
- The function of the Program Download Application and the Process Control Application although verified was not included as part of the testing. The program download function was assumed to occur instantaneously and the behaviour of the PCA was emulated as part of the *Simulated System (see 6.2.1 below).*

6.2.1 Modelling the VMD Behaviour

As has already been stated, the system could not be fully tested as a real functional system due to the unavailability of a wide range of CNC equipment and the necessary proprietary controllers (PLCs, PSCs etc). The full function of the Virtual Manufacturing Device could be implemented, but the feedback to it from the real world devices (PLC,PSC etc) did not exist. Therefore rather than the system waiting for real world feedback to occur(as would occur in the real system), the VMD behaviour was programmed to assume it had already occurred.

- ***The Real World System***

The fully functional VMD is shown in Figure 6.1 together with its interaction with the real world via the I/O driver. The Machine Front End (MFE) is made up of a number of FIXDMACS Input and Output blocks that act as an interface between the virtual world and real world. The FIXDMACS SAC (Scan, Alarm and Control Function)allows for the acquiring of real world data, and then the imaging of the data onto the MFE. The Program Invocation Loop (PIL) peruses the data, causes a series of control events to occur and then changes the status of the MFE. The FIXDMACS SAC then transmits the status of the MFE to the real world device and as such influences its behaviour. There is a data acquisition role, a

decisionmaking role and then a control role. This sequence is cyclic in nature and is only interrupted by process failures or manual intervention.

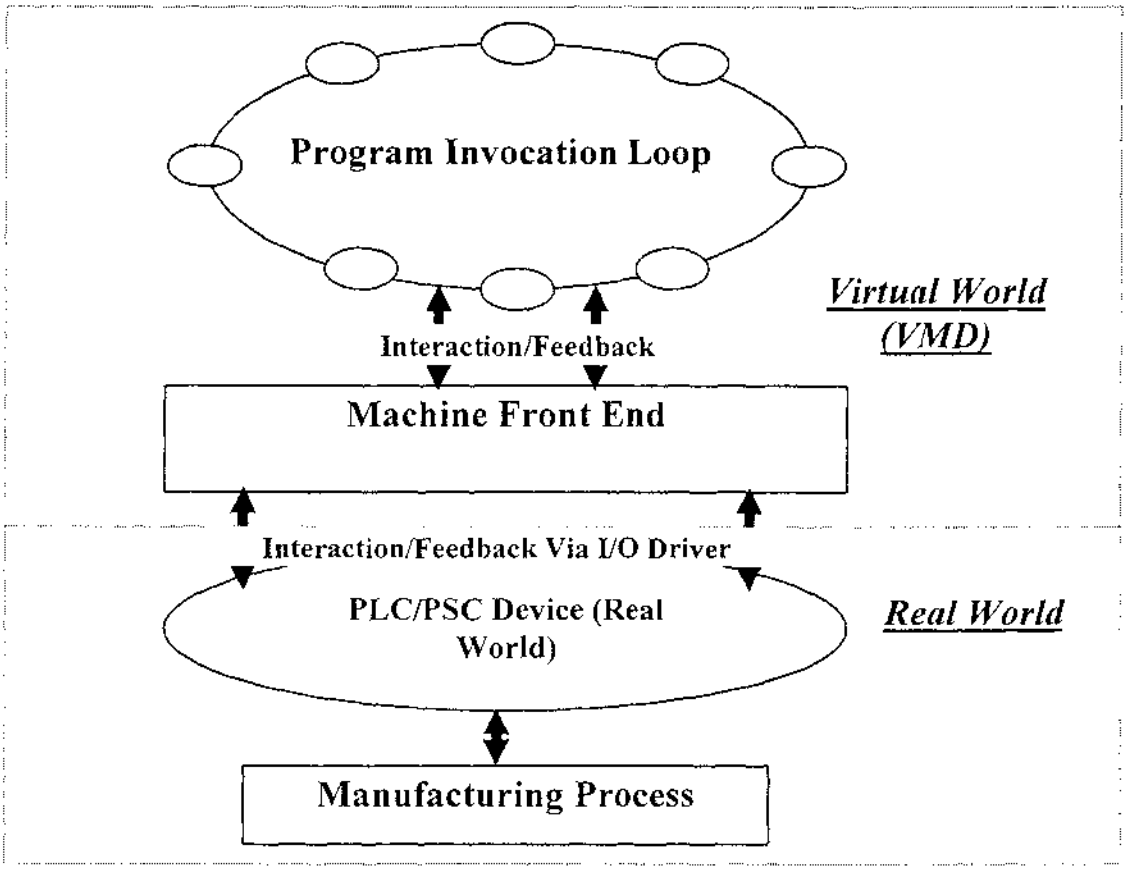


Figure 6.1 - Interaction of the Virtual and Real Worlds - the Real System

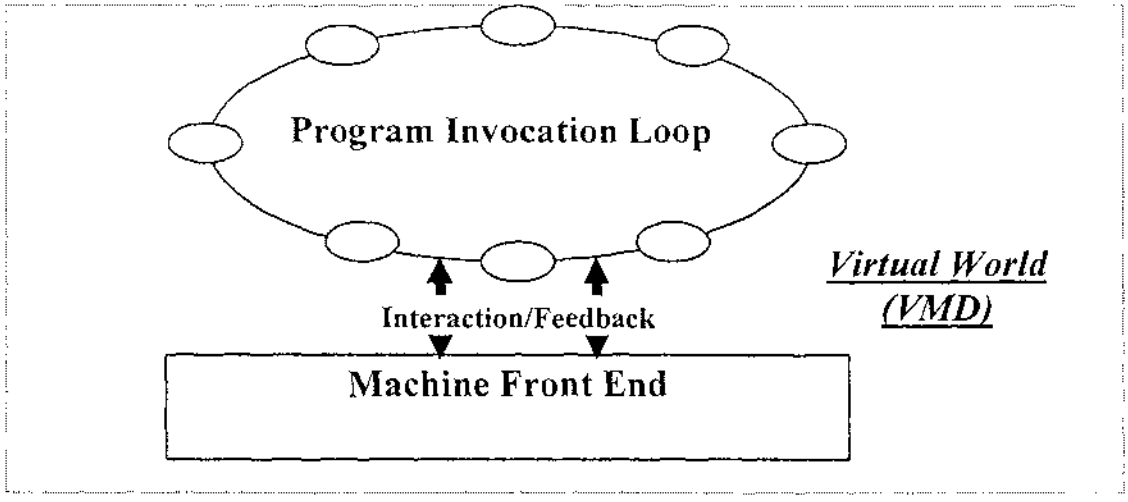


Figure 6.2 - The Simulated System

- ***The Simulated System***

Unlike the Real system as shown in Figure 6.1, the simulated system shown in Figure 6.2 has no real I/O driver or PLC. The behaviour and feedback is modelled using the interaction between the PIL and the MFE . For example, rather than the real manufacturing process providing feedback when it has finished a given cycle, a FIXDMACS "Timer" block is used to time the expected duration of a cycle before unloading the part and loading a new item.

The behaviour of the simulated system was modelled using two FIXDMACS Program Blocks:

- *V1SIMULATE (For VMD1)* - This block carries out the following functions:
  - ⇒ Waits for the Manufacturing Devices queue to have a task in it before proceeding.
  - ⇒ Checks that the process involved is in an active state.
  - ⇒ Waits for the necessary tool to be reserved before carrying out the task.
  - ⇒ Downloads the task data to the machine front end as required.
  - ⇒ Waits for the confirmation of a successful download before proceeding.
  - ⇒ Calls VMD1START

- *VMD1START (For VMD1)* - This block carries out the following functions:
  - ⇒ Checks to see if all cycles are completed.
  - ⇒ Starts the timer which simulates the set up, loading, processing and unloading of a cycle of the current task (Timer interacts with the MFE).
  - ⇒ Awaits confirmation that a given cycle is complete.
  - ⇒ Resets the VMD.
  - ⇒ Increments the variable representing the number of complete cycles.
  - ⇒ Repeats the above functions.

In reality each function within the two blocks would be modeled by its own program Block (or Capability) to ensure maximum flexibility. This would allow the functions to be ordered and initiated by the PCA (Process Control Application) in such a way as to ensure effective emulation of the Process Behaviour Algorithm. In practice a number of other capabilities would be used to ensure correct and effective functioning of the real world device (such those required for the download of NC files etc). However for the object of the test work it was not necessary and was impractical given the limited resources.

6.3 Experiment 1 - Standard System

The first experiment carried out involved the following:

- The fifteen jobs described in Appendix E were randomly queued in the Factory Controller jobs pending queue three consecutive times to give a total of 45 jobs.
- Start times and dates were set the same for all jobs to ensure effective queue buildup in the Hybrid FMS environment. In this way a definitive conclusion could be drawn.
- All transit times between cells were treated as equal. The transit times were set within the Factory Configuration Service to reflect this.
- 1 Job was auctioned at the Factory Controller each time the token was received. No jobs were allowed to build up in the Factory Controller "Processes Busy" queue as the queue buffer for each manufacturing process was set high enough to ensure that this would not occur.
- During the run all activities were recorded at each Cell and Factory Controller including average queue lengths, mean time between completion, task start times, finish times as well as a number of other variables.

6.3.1 Results of Experiment 1

All graphs presented below are a result of the interpolated data collected as shown in Appendix D.

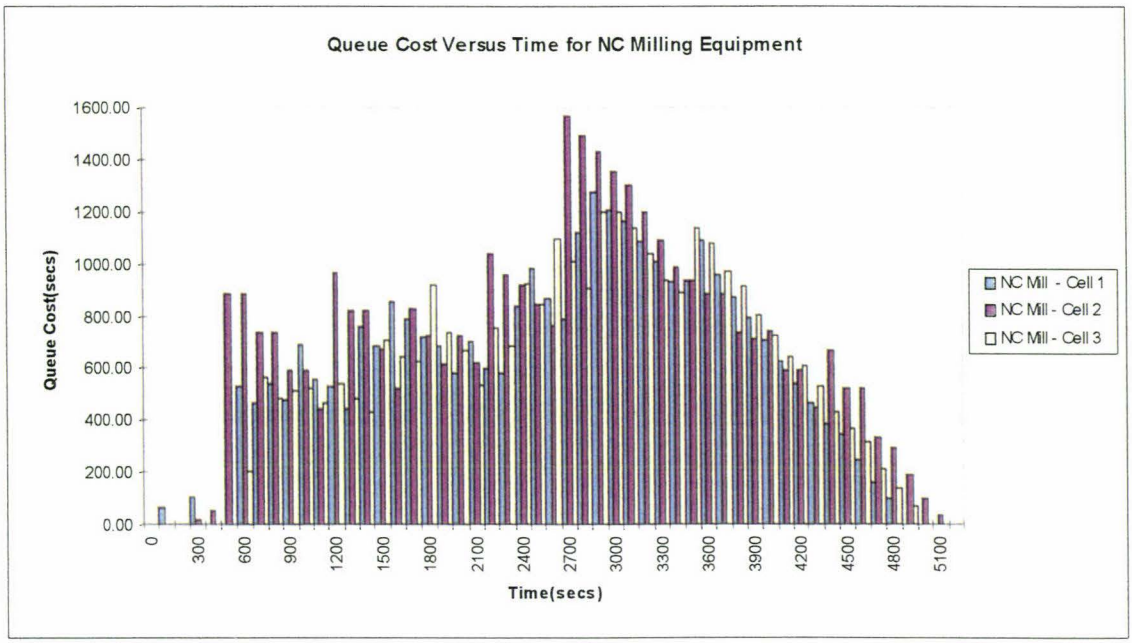


Figure 6.3 - Queuecost Versus Time for the NC Milling Machines in each Cell

A graph of the loading across the various NC Mills within the Hybrid FMS system as shown in Figure 6.3 demonstrates how the loading on each device in the various cells in the system is balanced. As a queue grows at a given device the other queues at the other devices become more competitive due to the use of the Earliest Finishing Time heuristic. This type of balancing will achieve maximum throughput within the system providing other aspects such as buffers are managed satisfactorily as outlined by Goldratt[6]. This balancing of the queues is also evident in Figure 6.4 which shows the equal loading of each of the queues feeding the NC Grinding machines. Examining Figure 6.5 which shows queue loading over the NC lathes it can be seen that at the beginning the loading of the machines are not even due to the period of instability caused by startup. However between the 2000 - 3500 second mark the system starts to settle down and the loads over all machines start to balance. This balanced situation will be most typical of the true manufacturing environment as a manufacturing enterprise will always be busy. Maximum work throughput will be achieved (assuming all processes have similar abilities) because the average queue cost will be shorter over each similar device, and each job will spend less time waiting in queues.

The data presented in Figure 6.6 also shows the feature of queue length balancing for the NC Milling process. The graph shown demonstrates the average queue cost over the duration of the test and it can be seen that as the simulation progresses the average queue costs for each device becomes comparable with the same device in each other cell. Over a long period of time it is highly probable that the average queue cost would be equal.

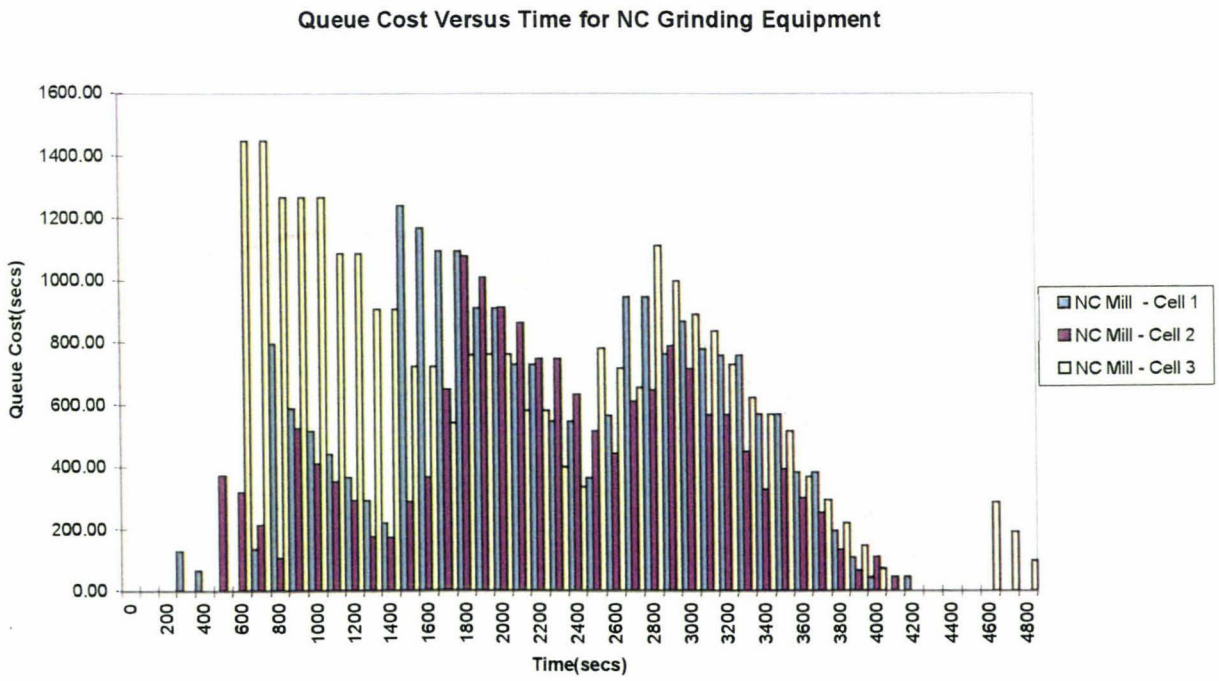


Figure 6.4 - Queuecost Versus Time for the NC Grinding Machines in each Cell

To maximise throughput and approximate J.I.T (Just -in-time) manufacturing queue buffers could be altered to ensure a minimum of jobs were present in each process queue - an added benefit of this would be a lowering of inventory in the system and therefore less money tied up in capital. As shown by Goldratt[6] in his studies on production systems the ability to manage work in process buffers can pay exceptionally large dividends.

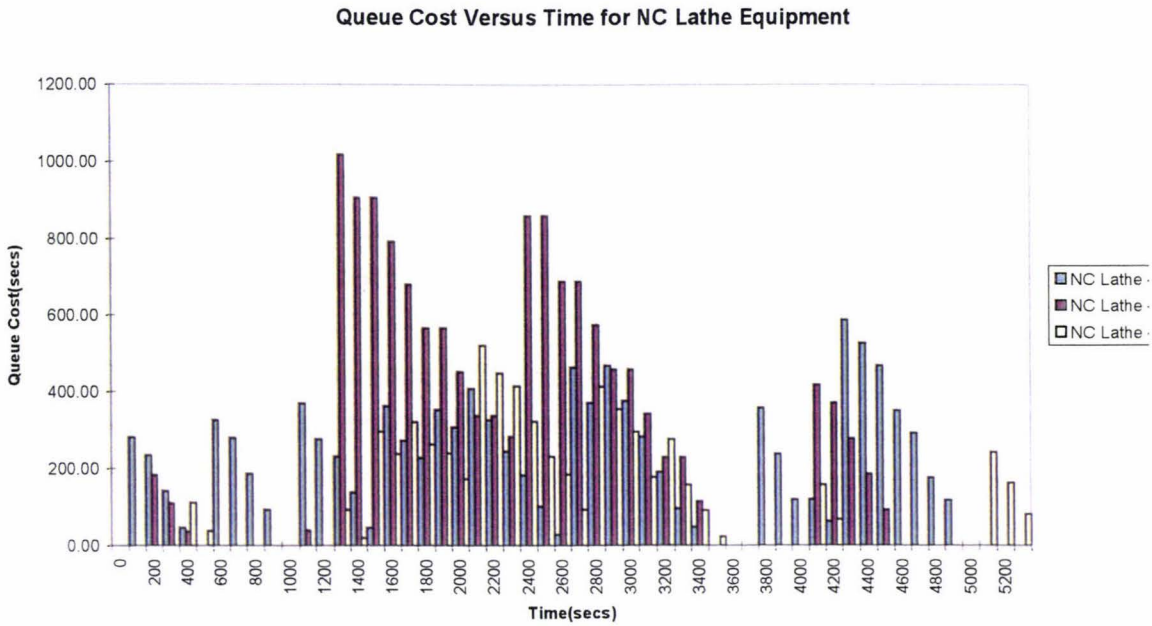


Figure 6.5 - Queuecost Versus Time for the NC Lathe Equipment in each Cell

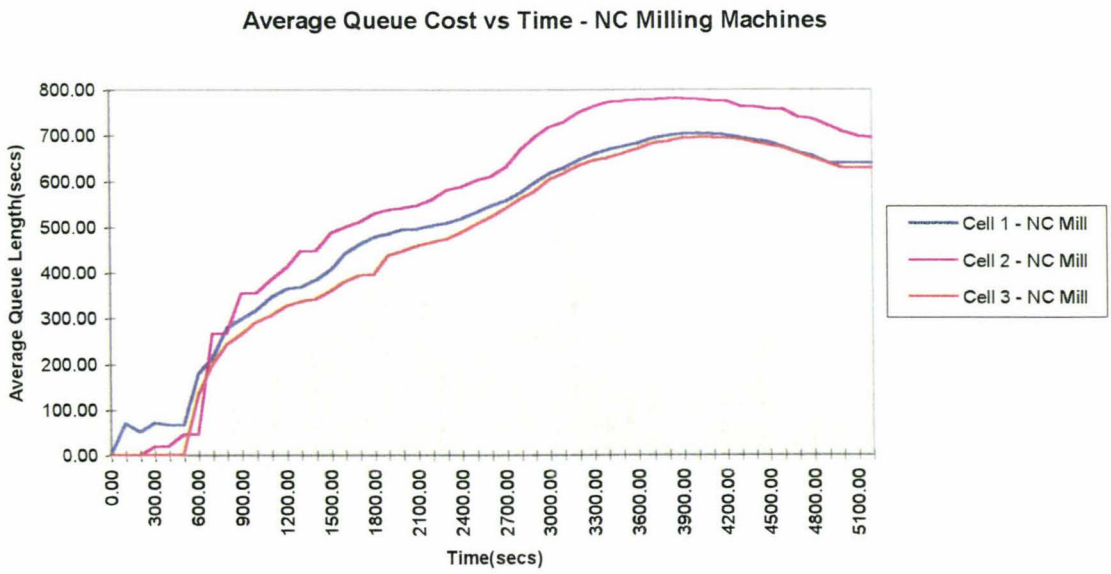


Figure 6.6 - Average Queuecost Versus Time for the NC Milling Machines in each Cell

6.4 Experiment 2 - Process Failure on Cell 1

One of the most important aspects of any system is its ability to recover from an unexpected event, as this ability to recover has a major influence on the disruption caused to the overall system. Tsukada in his work also showed that the negotiation approach to scheduling was very useful in the event of a process breaking down[49]. Therefore one of the tests that needed to be carried out was to deliberately cause a manufacturing process to fail and demonstrate how the Hybrid FMS system could recover.

The conditions of the experiment were replicated as for Experiment 1, and at some randomly selected time interval the NC Mill on Cell 1 was crashed.

6.4.1 Results of Experiment 2

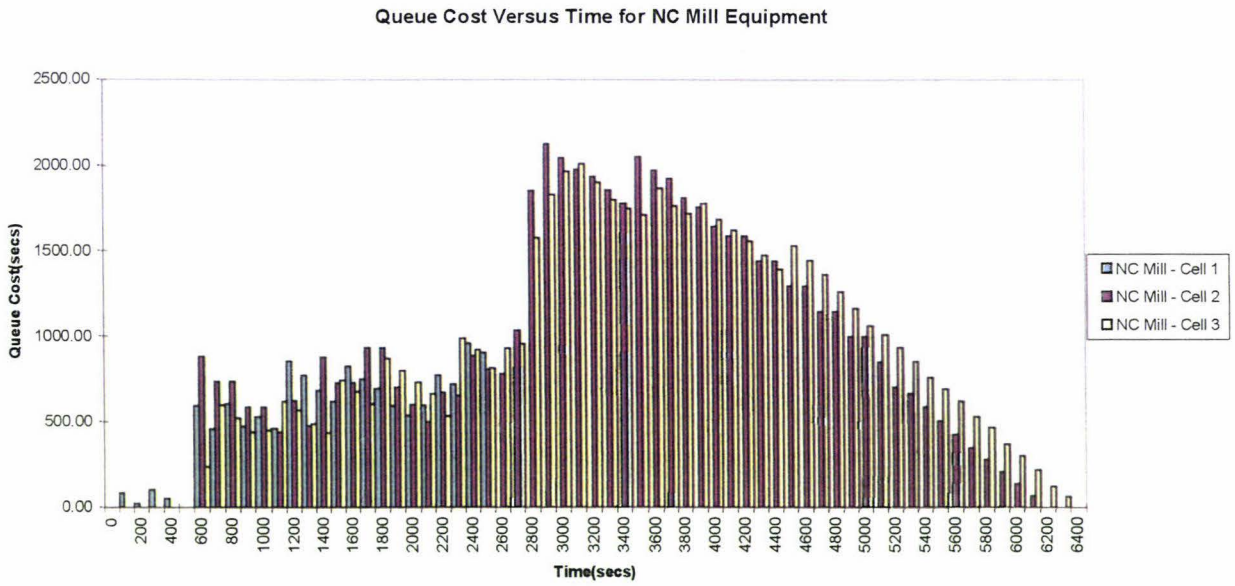


Figure 6.7 - Queuecost Versus Time for the NC Milling Machines in each Cell

The results of this experiment were very interesting. One of the first things that can be noted from the results is that the Mean Time Between Completion for the 45 jobs processed increased from 114.47 seconds to 143.63 seconds, which is expected as one resource - the NC mill on Cell 1, failed approximately 40 minutes into the run.

As can be seen in Figure 6.7 the queuecost stays the same up until the breakdown on Cell 1 after which the other two NC Mill processes on Cell 2 and Cell 3 takeover the workload. As such the queuecost for Cells 1 & 2 peak at around 2000 - 2200 seconds as opposed to the maximum queuecost over all the cells in the

first experiment which peaked at 1400 - 1600 seconds. The time to complete all milling operations also increased from approximately 5100 seconds to 6400 seconds.

The system was able to recover from the problem and redistribute the jobs on the failed process using the bidding protocol and this fact is clearly demonstrated in Figure 6.7.

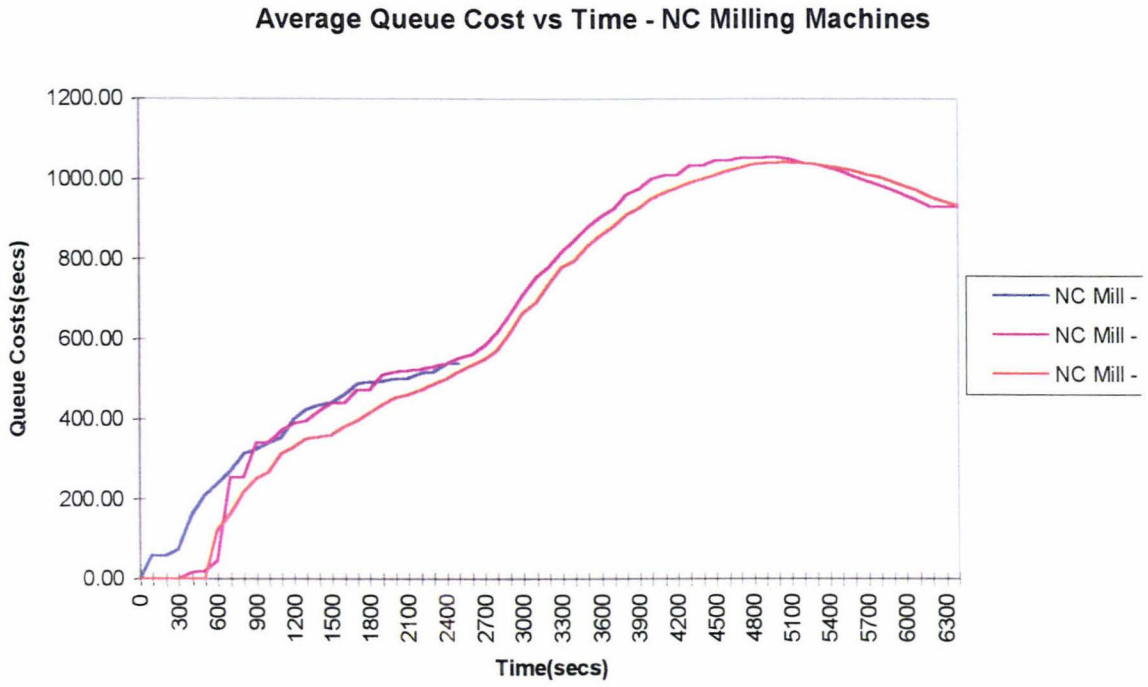


Figure 6.8 - Average Queuecost Versus Time for the NC Milling Machines in each Cell - Cell 1 Mill Failure

The graph of average queue cost for the NC Milling equipment (Figure 6.8) shows that the maximum average queue cost for the NC Milling equipment in Cells 2 and 3 increases from approximately 780 to 1000 seconds when the Mill fails in Cell 1. However, after redistribution of the jobs queued at the failed device the workload at the remaining two NC Mills in Cells 1 and 2 is almost exactly the same, therefore, ensuring maximum utilisation of the remaining equipment. This clearly demonstrates how the remaining cells negotiate to take ownership of the remaining tasks and how the failure on one process does not mean failure of the entire system.

6.5 Experiment 3 - Buffer Management

As has been mentioned previously the Hybrid FMS is able to incorporate the use of variable buffer sizes to control the volume of work in any one queue. This concept limits the volume of work that exists in an in going queue for a given manufacturing device and is generally used as an implementation of a given management strategy. Such a system is useful in the following ways:

- To approximate J.I.T Manufacturing where this philosophy is required to be implemented. This would be carried out by setting the queue buffer to the time required to process the largest existing task.
- To execute a program of buffer management for the various manufacturing devices within the system thereby increasing throughput and lowering inventory costs within the manufacturing environment.
- To implement any other management strategy that involves controlling the volume of work allowed in the various queues.

The conditions for the experiment were the same as for experiment 1 in that the jobs were introduced into the system in the same order, transit time between cells was assumed to be the same and each jobs description was kept consistent. In experiment 1 all in going queue buffer sizes were set at 30,000 seconds to ensure that the queue sizes were allowed to build naturally without any constraint. In the Buffer Management test the in going queue buffer sizes for the NC Milling Machine on Cell 1 was set at 400 seconds, the buffer for the NC Milling Machine on Cell 2 was set as 30,000 seconds and the buffer for the NC Milling Machine on Cell 3 was set at 800 seconds.

6.5.1 Results of Experiment 3

As can be seen in Figure 6.9 the NC Mill on Cell 2 which has the unrestricted in going queue buffer takes on the majority of the workload whereas the NC Mills on Cells 1 and 3 have queue costs which are not able to exceed the buffer sizes of 400 and 800 seconds respectively. The total milling time required to complete all milling tasks was 7050 seconds as opposed to 5116 seconds in Experiment 1 where the load was balanced over all common processes using the bidding algorithm. Experiment 1, however, was carried out under the assumption that all NC Mills over all three cells had the same ability to carry out the job. In reality this may not be the case as different NC Mills may have more horsepower or rigidity for instance and are able to carry out the work at a faster rate. In order to solve this problem, processing, set up and load/unload times could be allowed for in job descriptions to allow for a wide range of machinery of a given type in the one environment. This approach however would be very difficult and rather simplistic. A

better approach would be to look at the bottlenecks in the system over a long period of time and vary the in queue buffers to suit.

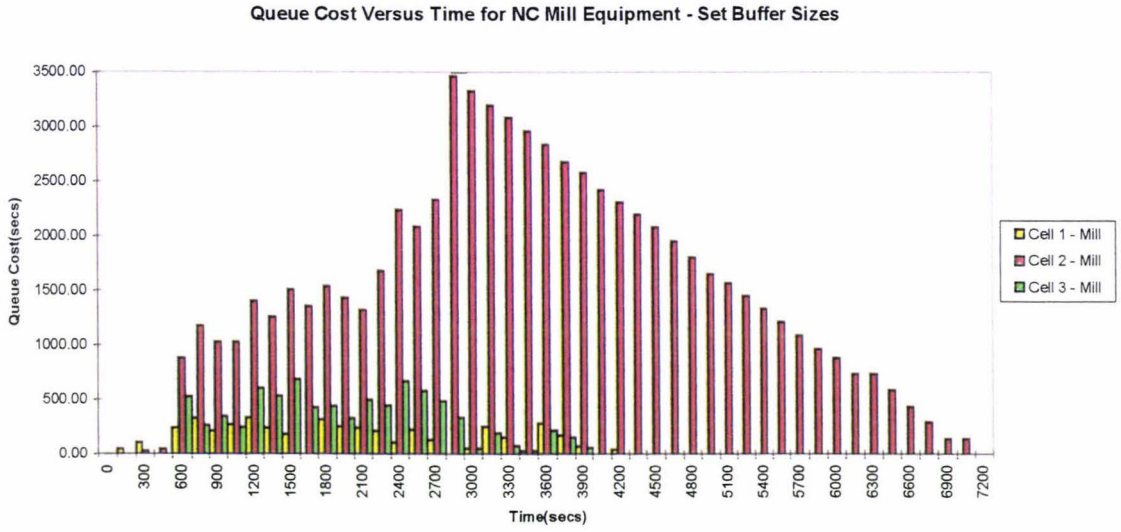


Figure 6.9 - Queuecost Versus Time for the NC Milling Machines in each Cell - Variable Buffer Sizes

As can be observed from Figure 6.9 each NC Milling Machine in going queue in each cell is restricted so that its size is not allowed to exceed the buffer stated for that machine. At scheduling time a check is made by the Bid Formulation Application in each cell to ensure that the proposed task cost added to an otherwise eligible processes in going queue cost does not exceed the buffer of the process. If the case arises whereby it does exceed the buffer the process is not able to submit a bid for the task.

Although the bidding system developed in this thesis performs well in terms of process utilisation it does not explicitly allow for variations in throughput between different machines of the same type over different cells. The load over processes of the same type over different cells may not be perfectly balanced as the time taken to process a task is task specific not process specific. This would tend to manifest itself as the same process types over various cells having different average in going queue lengths. For instance an NC Mill on one cell may be, on average, processing products far faster than its counterparts in other cells - as such its average queue cost is going to be smaller than the slower units. By adjusting the NC Mills buffer sizes over the three cells it would be possible to keep the amount of work in front of each process constant and therefore maximising machine utilisation.

The addition of variable in going queue buffers to the system is in reality an option which the user may find useful in an attempt to implement a certain policy or strategy within a particular organisation and needs to be included within the Hybrid FMS system.

It has been shown in Goldratts work[6] that the in going queue buffer for a constraint process must be strictly controlled to maximise throughput and an attempt has been made to accommodate this paradigm within the Hybrid system.

The main aim of this test was to verify that the in going queue sizes could be regulated over a long of period of time in response to an organisations strategic or political intent and this has clearly been demonstrated.

6.6 Experiment 4 - Standard test with variable Intercellular Transit times

The intercellular transit time is the time it takes for a particular job to travel from one cell to another in order to have a task processed. It is important to have such an aspect incorporated into the bidding algorithm as the travel time for a task from one cell to another delays the overall job. During the bid formulation stage at the cell level, once a manufacturing process is found to qualify for a task, the Bid Formulation Application calculates a bid and then adds the Intercellular Transit Time from the current jobs resident cell to the cell bidding to give a total bid. This then becomes the total bid which is transmitted to the Factory Controller.

The idea of including the transit time is to get around the problem as follows:

- An NC Milling task is required to be undertaken at a process within the manufacturing system.

    ⇒ A job is resident in a given cell (Cell 1) which has a NC Mill, the Queue Cost is 30 minutes, time to complete task is 10 minutes and the transit time is zero, giving a total bid of 40 minutes.

    ⇒ The job could also be carried out in a cell (Cell 2) which is 30 minutes transit time away but has a Queue Cost of 10 minutes giving a total bid of 50 minutes.

In this scenario the winning bid is 40 minutes as the potential time of transit (delay) to Cell 2 makes it uncompetitive. If the transit time was not considered Cell 2 would win the task with a bid of 20 minutes. However this is erroneous as the total time delay for a job in the system is equal to processing time plus transit time.

The experimental procedure was exactly the same as that carried out in Experiment 1 except that the intercellular transit times were different. In Experiment 1 the intercellular travel times were set as 5 seconds as shown in Figure 6.10.

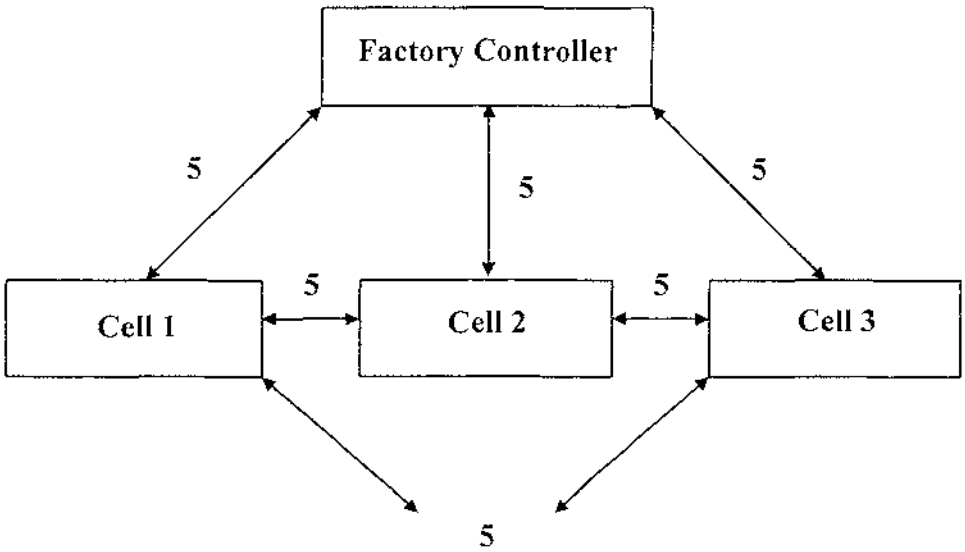


Figure 6.10 - Intercellular Transit Times for Test 1

In Experiment 4 Cell 1 is isolated to see how the overall distribution of jobs is affected. As can be seen in Figure 6.11 Cell 1 is alienated from the rest of the cells involved by increasing the transit time from 5 to 300, 305 and 310 respectively as shown for the various Cells and the Factory Controller. The results of this will be examined in the results section below.

6.6.1 Results

The graph of queue cost versus time for the NC Mill manufacturing process is presented in Figure 6.12. One of the most immediate effects of increasing the transit times for Cell 1 is a delay in loading of that cell with tasks required for NC Milling operations. It can be seen from the graph that the NC Mill on Cell 1 does not become loaded until the NC mills on Cell 2 and 3 become substantially loaded - as such the workload does not become shared equally between all the available devices. The fact that the NC Mill on Cell 1 finished earlier on this test than in the original test (Test 1) - 4800 secs compared to 4900 secs also

shows a slight under loading of the NC Mill on Cell 1. This would be expected as its isolation makes it less competitive in its ability to win tasks during auctioning.

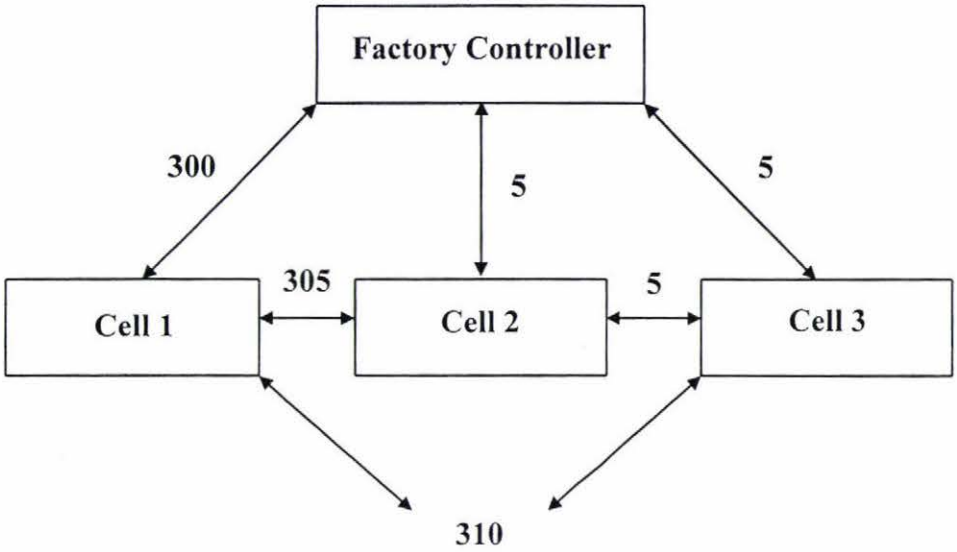


Figure 6.11 - Intercellular Transit Times for Test 4

Figure 6.13 indicates that the average queue cost over the test period for the NC Mill on Cell 1 was substantially less than that on the other cells. The average difference between the average queue cost on Cell 1 and that on Cells 2 and 3 equates to 233.8 seconds.

There are a number of negative ramifications that will result from this:

- If the system is lightly loaded with NC Milling tasks it is likely that only Cells 2 and 3 will be loaded with the tasks and that Cell 1 will be starved of any of the tasks as the queues on Cells 2 and 3 do not build sufficiently to allow Cell 1 to become competitive.
- If the NC Milling operation has been identified as being a bottleneck and the system is lightly loaded throughput will suffer due to poor utilisation of the bottleneck as a result of the isolation problem.
- Once Cell 1 has won a task it is likely to keep hold of the job and complete all remaining tasks associated with it (providing it has the manufacturing processes to carry them out). This is because the other competing Cells (2 and 3) are handicapped by the fact that Cell 1 is so isolated. This in turn will further affect the overall manufacturing process utilisation in a negative sense.

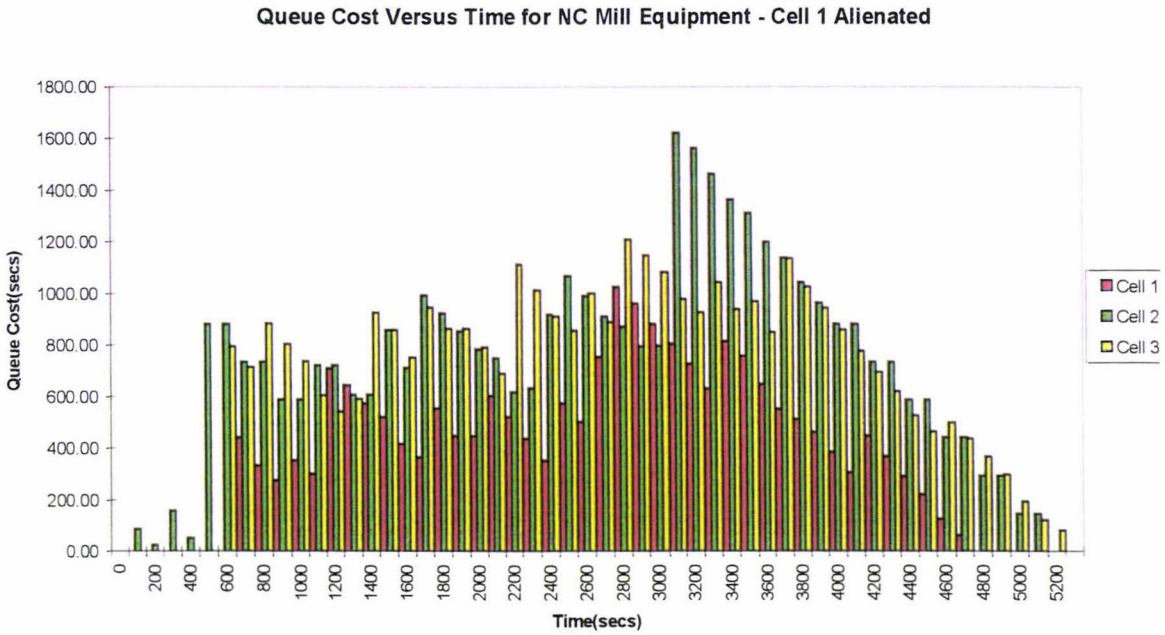


Figure 6.12 - Queuecost Versus Time for the NC Milling Machines in each Cell - Cell 1 Isolated

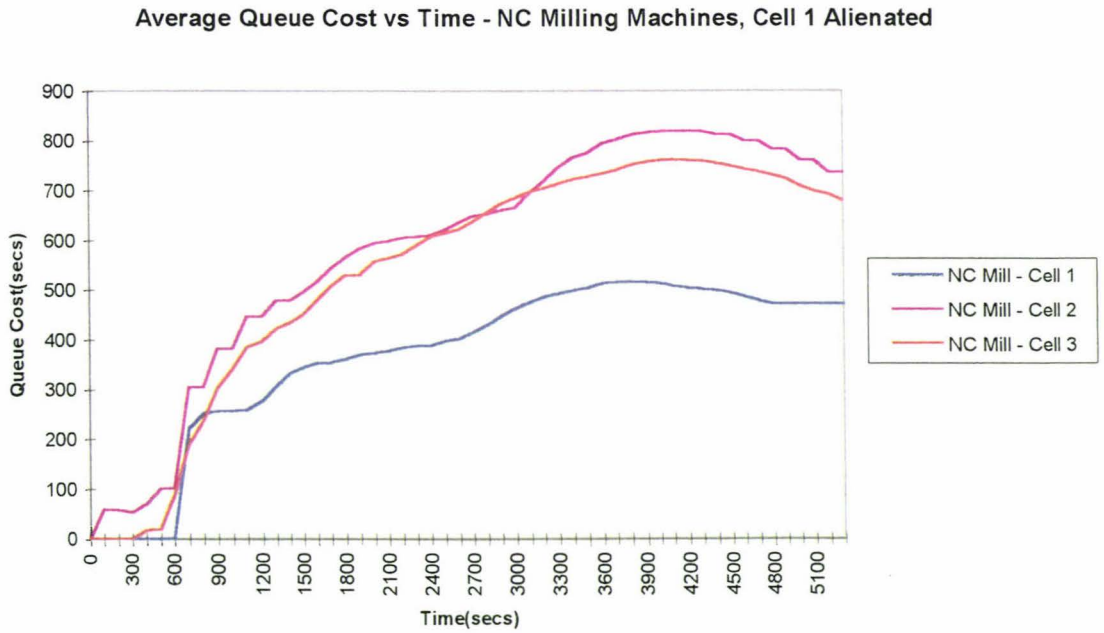


Figure 6.13 - Average Queuecost Vs Time for the NC Milling Machines in each Cell - Cell 1 Isolated

The main conclusions that can be drawn from this test is that wherever possible manufacturing cells must be physically located in order to ensure swift and efficient task transfer between them. Efficient and logistically correctly use of automated transport systems such as AGV's is of great importance to ensure timely delivery of the physical parts between the cells involved. Identification of the bottlenecks within the system is important. If in the circumstance a cell is spatially isolated it is most important to ensure that any bottleneck is not located in that cell if at all possible.

6.7 Conclusions

This chapter has shown that the Hybrid FMS system is operational and verifies aspects of the model as stipulated in Chapter 2. It has also verified many aspects addressed by earlier researchers. It has showed the Hybrid FMS to be robust due to its modular nature, it has demonstrated the positive aspects of the bidding approach to scheduling in terms of the loading of manufacturing processes and it has looked at a number of aspects related to the setting of buffers and location of cells.

Chapter 7 of this thesis will investigate the potential ramifications and implications of the possible use of Hybrid FMS systems in a real manufacturing organisation. This is important as the Hybrid system may have shown itself to be effective but if the method of implementation is not sound its overall positive impact may be compromised.

Chapter 7

Manufacturing Organisation Implications and Recommendations

7.1 Hybrid System Strengths and Limitations 206

7.2 Implications of Hybrid FMS for Industry 210

7.3 Future Work 212

The small to medium sized manufacturing organisation often has little spare capital to spend on complicated and expensive Flexible Manufacturing Systems (FMS). As has been stated the majority of manufacturing concerns within New Zealand as well as the United States fall within this category. The objective of this thesis was to overview current trends in FMS development and from this develop a Cellular based FMS system that took the positive aspects of these developments and also eliminated the negative aspects. The outcome from this work is the practical implementation of a cell based Hybrid FMS system which has shown to have a high degree of flexibility, expandability and to be robust in the event of unexpected failures of equipment, manufacturing processes and the like.

This Chapter discusses the positive and negative aspects of the system as discovered from the evaluation and testing undertaken and looks at the contribution the product can make to the manufacturing organisations in the small to medium size category. Lastly relevant points regarding future aspects of the system that could be further improved or developed are discussed.

7.1 Hybrid System Strengths and Limitations

A Hybrid Flexible Manufacturing System (FMS) has been developed to meet the needs and budget of the small to medium sized manufacturing organisation.

The software development side uses two well known software packages for the implementation of the system model - Intellution FIXDMACS and Microsoft Visual Basic. Intellution FIXDMACS is a very user friendly package that is continuously under development and Microsoft Visual Basic is fast becoming the most commonly chosen Windows programming environment. Use of these packages well into the future should not be of undue concern.

In the development of the model and the resulting software most aspects of the real manufacturing world were incorporated and during the testing process (see Chapter 6) it was shown that these aspects of the model were implemented and functioned as required. The only portion of the system that did not function as a real entity was the connection of the VMD with the real world devices. However simulation of this proved to be adequate.

The Hybrid system developed has a number of strengths and limitations that will govern its effectiveness and practicality in a real world sense. These strengths and weaknesses are outlined in Table 7.1

Hybrid FMS Strengths	Hybrid FMS Weaknesses
• High Degree of Flexibility and Modifiability • High Extensibility • Comparatively Cheap • Modular approach to model design ensures durability and graceful degradation in the event of equipment failure. • Hands on Graphical User Interface/Screen Toggling facility. • Totally customisable to handle a variety of applications. • Integrates with existing hardware and software systems easily. • Is Windows based and therefore compatible with current major software systems. • High Network Compatibility • Easy to use. • Data Logging system enables user to see progress of jobs throughout system.	• FIXDMACS Software may not be reactive enough to cope with all manufacturing scenarios. • Computer power required needs to be reasonable - at least 32Mb of Ram.

Table 7.1 - Strengths and Weaknesses of the Hybrid System

- ***Hybrid System Strengths***

The Hybrid FMS system has a number of strengths as follows:

- As has been shown the system is highly modifiable and flexible. As the system is modular in its design new cells can easily be created. The software can be added in as well defined units and once the hardware has been connected and the VMD configured the system is ready to go.

- Compared to many available systems requiring large computer systems the Hybrid FMS requires only one standalone PC and can operate over many network configurations. Nearly all existing network types are able to be used in conjunction with the software.

- The modular software design and the loose coupling that exists between the modules and cells ensures that a high degree of fault tolerance is achieved. Many systems today can crash due to some unexpected event occurring. However in the Hybrid system the fact that communication with other modules is minimised the effect on the overall system is small and can be contained.

- The Intellution FIXDMACS Graphical User Interface provides a clear animated illustration of what is occurring in a particular process anywhere in the cellular system. For instance, in the event that the user on Cell 1 wants to find out the status of a process on Cell 2 it is possible for that person to access the FIXDMACS View Screen for that manufacturing process. This is very useful as it saves the user having to go to the PC associated with the cell concerned.

- The Hybrid FMS system is customisable for a wide variety of situations. Any type of manufacturing process from an NC lathe to an automated sheep skinning machine can be generically defined by the system as can the resources required by those processes. The Process Control Algorithm can be configured to define the behaviour of any process being controlled thereby making the system extremely flexible and adaptable to a wide variety of situations.

- The system can be easily integrated with a wide variety of hardware and software systems. As the control logic is removed from the proprietary controller (PLC, PSC's etc) by virtue of the VMD, the proprietary controller has the role of a data interface between the computer and the real manufacturing process. As such a wide variety of proprietary controllers can be used with no need for reprogramming each one in a series of different languages. The only thing required is suitable I/O software and this is readily supplied by Intellution.

- Network Compatibility - As the network communication is implemented using the Intellution FIXDMACS system the Hybrid FMS software is able to operate using a wide variety of network

communication protocols ie: IBM Token Ring with IBM LAN Support Program version 1.25, 3COM Ethernet with IBM LAN Support Program version 1.25, 3COM Ethernet with Novell NetWare version 2.2 or 3.11, IBM PC Network with IBM LAN Support Program version 1.25, 3COM Ethernet with Pathworks version 4.1 and Digital Ethernet with pathworks version 4.1. The likelihood of the system being compatible with an existing LAN system is very high and even if it isn't can be changed at relatively low cost.

- The system is programmed for the Microsoft Windows environment and as this is the standard for PC's it is compatible with the modern computer training course. This makes the system easy to use as most people who are likely to use it are familiar with Windows based systems.

- The data logging function on each Cell Controller and Factory Controller gives the user the ability to get a history of what has occurred on each node at any point in time. The system has been programmed to allow this material to be cut and pasted into Microsoft Excel thereby allowing the user to produce graphs and to follow trends. These graphs can then be formatted into reports or saved for later use.

- ***Hybrid System Weaknesses***

Although no serious weaknesses of the system have been identified there are a few potential shortcomings:

- For some applications requiring very fast responses there may be a problem due to the finite scan time allowable with the FIXDMACS package. Generally speaking, however, this would probably involve going outside the target area of application set for this project.

- Computer Power - During the trial stage it was found that some of the lesser powered computers, eg: IBM 486 compatible PC's with 8Mb of Ram and 33Mhz processor speed, were struggling to deal with the volume of data being processed by the system. This was shown to be the case when the Norton System Watch software was installed on all the PC's utilised in the system and this showed that the virtual and physical memory ran very low (see Figure 7.1 below).

This indicates that some of the older computers will struggle in their roles as cell controllers or as a factory controller. However this will not be a problem in the future with the proliferation of faster more modern systems.

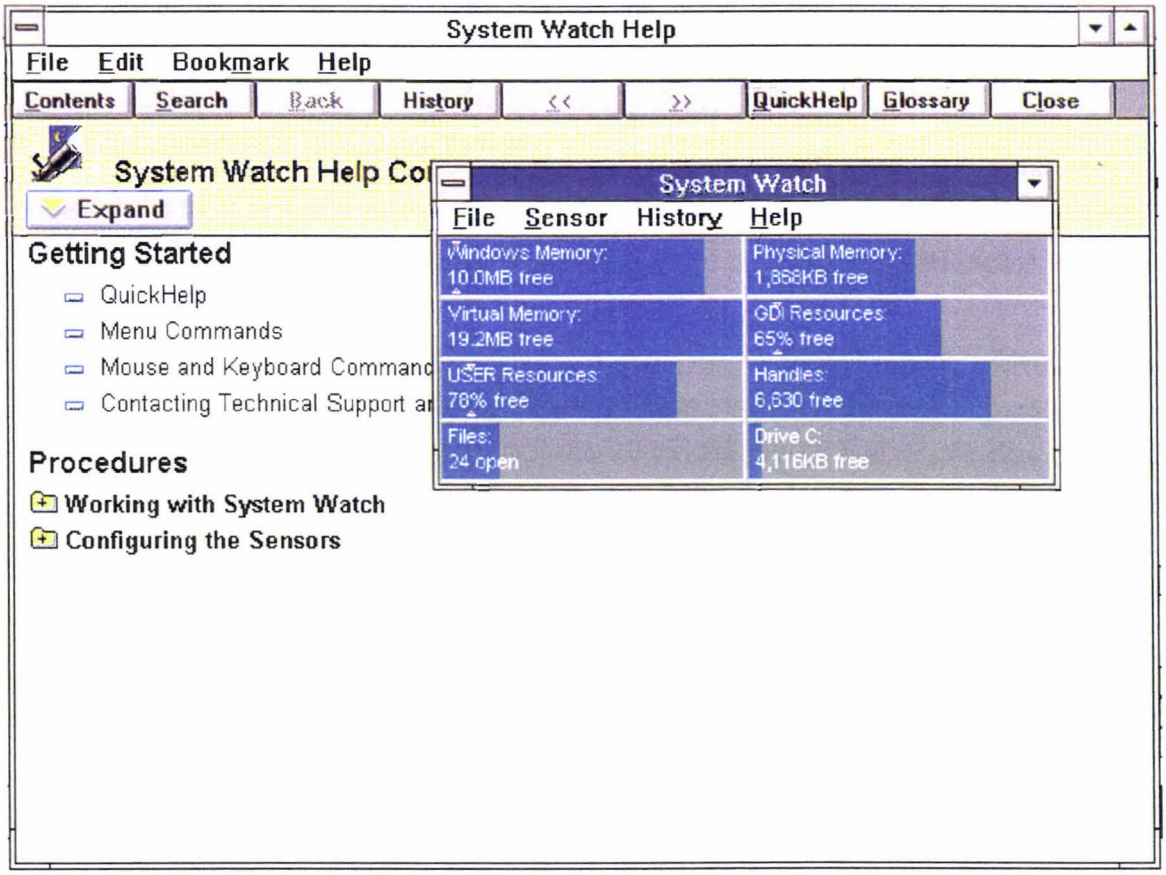


Figure 7.1 - Norton System Watch as Used during Testing

7.2 Implications of Hybrid FMS for Industry

The largest barrier to implementing FMS in the small to medium sized manufacturing enterprise is convincing the financial controller of the viability of any potential system. In many cases the Project Engineer researching the case may or may not know a lot about such a system and unless he carries out a lot of research he may be only able to speculate to a certain degree on the tangible benefits. Converting this research into possible savings to the company and subsequent preparation of a financial justification may also prove difficult, as such system cost is very important. Many companies employ accountants as CEO's and as such the focus is always on value for money and payback periods. A companies Financial Controller can be held accountable for any capital investment that does not give reasonable financial returns therefore he is likely to be cautious.

The fact that the Hybrid system is PC based and can implemented on commonly existing hardware means that cost is reduced compared with many systems which require significant hardware outlay. This fact

makes the system far easier to justify in the first place and lowers the risk factor involved. As such the paybacks are a lot faster thereby freeing up more capital for further expenditure.

The other benefit of the Hybrid FMS System is that one or two cells can be set up as a test bed at very little cost, and once the financial rewards are realised, the system can be extended to include up to 100 cells. This creates less risk for the company. The Hybrid FMS therefore has an advantage over any other system that requires a full expenditure to be undertaken. Many systems in the past have suffered from being non-modular meaning that the extension of an existing system requires complicated and expensive reprogramming. The modular approach taken with the Hybrid System avoids this expense and makes the system more financially viable for companies.

The most expensive part of the Hybrid FMS implementation would probably be the purchase of the Intellution FIXDMACS software, but this could also be carried out in stages thereby decreasing the required expenditure.

One of the more difficult issues to overcome is the effect that implementing such technology has on the disposition of the workforce. Potentially the Hybrid FMS has the ability to lessen the workforce required within the work environment (eg: very little work would be required to link the system to a robot or AGV) and in many capital expenditure requests this is used as a cost saving feature to help obtain the capital. In many cases union officials become involved and workers can become very resistant to the introduction of the technology and to overcome this a very tactful and diplomatic approach is required.

The skills required in the workforce to operate the FMS is another aspect that would be required to be addressed. The workforce employed within the company may have little or no computer literacy and may need to be completely retrained. This can become a problem as there are often personnel within the company who are nearing retirement and are not interested in learning a new skill - others maybe fearful of change and cause resistance. It is imperative that all personnel involved or influenced by any installation are kept up to date about what is happening every step of the installation - from the initial research all the way to commissioning. This is best carried out by conducting regular well structured meetings.

In the previous chapter of this thesis results of all tests carried out showed the system to function correctly and follow the requirements of the model. Proving that a system performs as required is only a small step to ensuring that the system is effective in the modern manufacturing environment. Many companies introduce an FMS system hoping for a 'quick fix', however all aspects need to be addressed such as skills assessment of the people involved, correct and thorough project planning and awareness. If personnel on the shop floor have input from the initiation of the project they will take ownership. This is vital as they will ultimately determine the success or failure of the project.

7.3 Future Work

In a complex system such as the Hybrid FMS it is conceivable that the system will be never be finished as there are always improvements that can be made. As the system currently stands the next stage would be to create a real Virtual Manufacturing Device using the Fisher and Paykel PSC's and link the PSC's to real NC manufacturing equipment and carry out real trials on the system. The system would probably require another 12 months work to get it to the stage where it could be used in the real manufacturing environment.

Once this has been completed there are a number of enhancements which could be made to the software as mentioned below:

1. **Alternative Scheduling Heuristics** - There are a number of scheduling heuristics that can be used with the auction based scheduling system used in Hybrid FMS. In this project the EFT (Earliest Finishing Time) heuristic was used. Shaw[48] in his simulation work found that the EFT heuristic used in his auction based program was far superior to the SPT (Shortest Processing Time) heuristic. The EFT heuristic was superior in all aspects measured - Job Tardiness, Job waiting time, Average lateness and Mean Flow Time. Many companies may however may adopt other measures of system performance and use other heuristic rules for scheduling eg: The SPT rule is known to maximise throughput. An algorithm for integrating a wide variety of heuristic rules into the existing auction based system should be investigated.

2. **Tool Bidding Algorithm** - the tool bidding algorithm was developed as part of the Resource Scheduling function. The system ran satisfactorily but had an intermittent fault which caused tools not to be returned so that they could be released from the current holder's tool carousel. This problem was not resolved during the time available for the research and needs to be looked at further.

3. **View Screen Development** - A generic FIXDMACS screen was developed as part of a final year research project. This generic view screen enabled images of various machines to be connected to database blocks within the FIXDMACS environment and allowed for the development of custom view screens. This software needs to be connected with the FIXDMACS database environment to give the user a virtual representation of the action of the manufacturing processes accessed. The system will be very easily linked with the existing Hybrid FMS and should be a major part of any future work.

4. **Cell Crash Algorithm** - As part of this project a test was carried out to show how the system is allowed to recover in the event of a manufacturing process crash. The system has also been programmed to allow the software to continue to behave normally in the event that a cell crashes. This

needs to be fully checked to verify that this is able to occur. Some slight adjustment to the software may be required to ensure full functioning of this facility.

5. **No Bid Submission Recovery** - At the Factory Controller level during the comparison of task bids there is a slight problem that needs addressing. If one of the Cells does not respond to an auction there is no recovery procedure written. A cells failure to respond may lead to an unpredictable system reaction. A circumstance where this would occur is in the event that a cell crashes and as such cannot respond. It is logical to tie this algorithm in with clause 4 above.

6. **Real Time Data for Manufacturing and Transit Activities** - The system needs to be able to record actual set up times, processing times for the various tasks and also transit times between cells. This will allow for the estimates of these times as stated in job definition files and intercellular travel time files to be updated giving a far more accurate representation of what is actually occurring. As a result the scheduling algorithm employed in the Hybrid FMS will become far more efficient.

Chapter 8

Summary

8.1 Summary 215

8.1 Summary

Computer controlled manufacturing systems are becoming more and more accepted in the modern world. Some projects such as the ESPRIT[4,8-9] research program have focused on the modelling of the entire manufacturing environment including all areas of functionality.

As has been stated previously such modelling of the entire environment is of little practical use to the small and medium sized manufacturing organisations which make up the bulk of manufacturing in the world today. Large complex systems modelling complete manufacturing environments are expensive and require high levels of maintenance. These type of systems are, therefore, seen as being financially unviable. There needs to be a cheap alternative that can be easily justified and follows modern ways of thinking regarding computer managed systems.

The opening chapter of this thesis provides a summary of over seventy different papers outlining modern trends in FMS and CIM related issues. In the chapter many of the positive and negative aspects regarding system architecture, scheduling systems as well as a host of other issues are raised. The benefits of cell based manufacturing are explained and are quantified to give the benefits true meaning. The theories behind distributed and centralised manufacturing systems are discussed and evidence is provided of the benefits of the distributed style of data flow. The Contact Net Protocol and its application to distributed problem solving is documented comprehensively, and the advantages over existing scheduling paradigms are explained.

Chapter 2 takes the positive aspects from the literature review presented in Chapter 1 and creates a new architecture termed Hybrid FMS. This architecture is built from a culmination of modern thinking in the areas of distributed manufacturing, cooperative scheduling, agile manufacturing and fills a void for the small to medium sized manufacturing enterprise. All aspects of the system are postulated as models to assist the reader in understanding how the Hybrid FMS functions without going into the full detail.

Chapter 3 introduces the two main software packages that were used to emulate or follow the behaviour of the models presented in Chapter 2. The packages that were introduced were Intellution FIXDMACS and Microsoft Visual Basic. The implementation of each model was explained in Chapters 3-5 in significant detail to enable the reader to have a high level of understanding of how the system actually functioned at the lowest level.

Chapter 6 detailed a number of experiments showing the actually functionality of the system therefore providing evidence that the system actually functioned as a fully active software package. The model was verified in critical areas such as agility and robustness in the event of process failure.

Chapter 7 summarises the positive and negative aspects of the system and raises issues about the implementation of the system into the modern manufacturing environment. Some of the issues looked at include the potential problems that may be encountered in the installation of the system and how to overcome them. Also mentioned in this chapter is the fact that installation of such a system into the modern environment should be carefully managed and address human and skill related aspects.

This thesis has provided a Hybrid FMS system for use in the small to medium sized manufacturing environment. The system is agile, robust and can be used to integrate a wide variety of existing hardware and manufacturing plant. As such it is very cost effective and easy to justify in the real manufacturing organisation whose focus is largely on financial aspects.

References

1. **Ranky, PJ**, "The Design and Operation of FMS"; IFS (Publications) Limited., UK; North-Holland Publishing Company (1983). Chapter 1, pp. 1-11

2. **Ramasesh, R** , "Dynamic Job Shop Scheduling: A Survey of Simulation Research", *OMEGA International Journal of Management Science*, Vol 18, #1, pp 43-57, 1990

3. **Bemelman, R and Seabrook, T.D** , "A Methodology for Cellular Manufacturing", *1993 Austrailian Conference on Manufacturing Engineering*, Adelaide, 22-24 November 1993.

4. **Beeckman, Dirk** , CIM-OSA: "Computer Integrated Manufacturing - Open System Architecture",*International Journal of Computer Integrated Manufacturing*, Vol 2 , # 2 pp. 94 - 105. March, 1989

5. **CIM Reference Model Committee, Purdue University** , "A Reference Model for Computer Integrated Manufacturing from the viewpoint of Industrial Automation", *International Journal of Computer Integrated Manufacturing*, 1989, Vol 2, #2, pp114 - 127

6. **Goldratt, EM,** Sifting Information Out of the Data Ocean - The Haystack Syndrome, 1990

7. **Rozmus, R** , "The Relevance Of MAP in Control", *Process and Control Engineering, April 1990*

8. **Klittich, M** , "CIM-OSA Part 3: CIM - OSA Integrating Infrastructure - the operational basis for Integrated Manufacturing Systems", *International Journal of Computer Integrated Manufacturing*, 1990, Vol3, #3 & #4, pp168 - 180

9. **Jorysz, H.R and Vernadat, F.B** , "CIM-OSA Part 1: Total Enterprise Modelling and function view", *International Journal of Computer Integrated Manufacturing*, 1990, Vol 3, #3 & #4, pp144 - 156

10. **Klug, A, and Tsichritis, D,** "Study Group on Database Management Systems (IFIP Press, Reston), 1977

11. **Mo John PT & Chi Kang Tang** , "Task Orientated Messaging System for Flexible Robot Control".; *Proceedings of the first Asian Control Conference*, Vol 2, 27-30 July , 1995 Tokyo, Japan. pp 781-784

12. **Rajabpour R & Mo JPT** , "Real Time Event Driven Scheduling for Message Controlled Manufacturing Devices", *6th International Conference on Manufacturing Engineering,* Melbourne 29 November - 1 December 1995

13. **Wang Y & Mo JPT** , "Integrated Robot Control Using Manufacturing Message Specification Protocol Based on NetBios", *Control Engineering Practice* , Vol. 1 #6, pp. 971 - 978, 1993

14. **Mo JPT & Yeung R** , "An MMS-Based Open Architecture for Flexible Assembly Cell", *Royal Melbourne Institute of Technology*

15. **Moore, Glennis, "Manufacturing Automation Protocol** - Mapping the Factory of the Future", *Electronics & Power*, April 1986.

16. **Veeramani, D, Bhargava, B & Barash, M.M,** "Information System Architecture for Heterarchical Control of large FMS", *Computer Integrated Manufacturing Systems* ,Vol 6, #2, pp 76-90, May 1993.

17. **Libes, D and Barkmeyer, E**, "The Integrated Manufacturing Data Administration system (IMDAS) - an overview", *International Journal of Coimputer Integrated Manufacturing*, Vol. 1, # 1, 1989, pp. 44-49.

18. **Duffie, N.A,** "Synthesis of Heterarchical Manufacturing Systems", *Computers in Industry*, Vol 14, pp 167-174, 1990

19. **Hatvany, J**, "Intelligence and Cooperation in Heterarchical Manufacturing Systems", *Proceedings of the 16th CRIP International Seminar on Manufacturing Systems.*, Tokyo, Japan. p.p 1-4, 1984.

20. **Jones, A and Saleh, A,** "A Decentralised Control Architecture for Computer Integrated Manufacturing Systems", *Proceedings of the IEEE International Symposium on Intelligent Control,* 1989, Albany, NY, pp. 44-409.

21. **Duffie, N.A, Piper, R.S, Humphrey, B.J. & Hartwick, J.P,** "Hierarchical and Non-Hierarchical Manufacturing Cell Control with Dynamic Part-Orientated Scheduling" *Manufacturing Systems Engineering*, University of Wisconsin-Madison.

22. **Duffie, N.A & Piper R.S,** "Non-Hierarchical Control of Manufacturing Systems", *Journal of Manufacturing Systems*, Vol5, #2, pp 137-139, 1986.

23. **Duffie, N.A, Chitturi, R & Mou J.I,** "Fault-tolerant Heterarchical Control of Heterogeneous Manufacturing System Entities", *Journal of Manufacturing Systems*, Vol 7, #4, pp 315-328.

24. **Duffie, N.A,** "Non-Hierarachial Cell Control", *Manufacturing Cells, Control, Programming and Integration*, Edited by D Williams & P Rogers, 1991

25. **Shaw, M.J,** "Dynamic Scheduling in Cellular Manufacturing Systems: A Framework for Networked Decision Making," *Journal of Manufacturing Systems*, Vol. 7, # 2, 1988, pp. 83-94.

26. **Duffie, N.A & Piper, R.S,** "Non-Hierarchical Control of a Flexible Manufacturing Cell", *Robotics & Computer-Integrated Manufacturing* , Vol 3, #2, pp. 175-179, 1987.

27. **Gremillion, L.L,** "Determinants of Program Repair Maintenance Requirements", *Communications of the ACM,* Vol. 27 , # 8, August 1984, pp. 826-32.

28. **Williams, TJ,** "One View of the Future of Industrial Control" , *Control Engineering Practice*, Vol1, #3, pp. 423-433, 1993

29. **Pu, J; Harrison, R; Weston, RH & Moore PR,** "Architecture/Software Design for Machine Control", *Advanced Manufacturing Engineering*, Vol 2, October 1990

30. **Hatvany, J and Nemes, L,** "Intelligent Manufacturing Systems - a tentative forecast", *Proceedings of the 7th World Congress of IFAC,* Helsinki, 1988.

31. **Duffie, NA & Prabhu, VV,** "Real-Time Distributed Scheduling of Heterachial Manufacturing Systems" *Journal of Manufacturing Systems* , Vol 13, #2, pp. 94-107

32. **Shaw, MJ** , "A two-level planning and scheduling approach for Computer Integrated Manufacturing", *Real-time optimisation in Automated Manufacturing Facilities.* Gaithersburg, Md.: U.S Department of Commerce, National Bureau of Standards pp.185-195.

33. **Kung, SK and Marsden, JR,** "Development and Implementation of a Dispersed Decision Process: an FMS Scheduling Example", *Computer Integrated Manufacuring Systems*, Vol.8, #2, pp. 93-103, 1995.

34. **Kung, SK and Marsden, JR,** "A Methodology and Experimental Shell for Formally Addressing Centralised/Distributed Decisionmaking Choices", *Decision Support Systems,* Vol 15, #1, pp. 45-61.

35. **Bakker, JJA**, "DFMS: A New Control Structure for FMS", *Computers in Industry*, Vol 10, Part 1, 1988, pp.1-9

36. **Zhang, SJ, Song, GN, Zhang, Y, Li, L, Wang, W, Gao, Z & Peng , H**, "The Application of Distributed Problem Solving in CIM", *Proceedings of the 3rd Symposium on Robotics and Manufacturing* , July 1990, British Columbia, Canada

37. **Shaw, MJ and Whinston, AB** , "Task Bidding and Distributed Planning in Flexible Manufacturing", *Proceedings of the Second Conference on Artificial Intelligence Applications,* 1985, pp. 184-189

38. **Rasbelo, RJ and Camarinha-Matos, LM** , "Negotiation in Multi-Agent Based Dynamic Scheduling", *Robotics & Computer-Integrated Manufacturing*, Vol. 11, #4, pp.303-309.

39. **Shaw, MJ** , "FMS Scheduling as Cooperative Problem Solving", *Annals of Operations Research*, Vol 17 (1-4), Jan 1989, pp. 323-346.

40. **Shaw, MJ**, "Distributed Planning in Cellular Flexible Manufacturing Systems" *Infor*, Vol 25, #11, 1987

41. **Smith, RG & Davis, R**, "Applications of the Contract Net Framework: Distributed Sensing", *Distributed Sensor Nets: Proceedings of a workshop held at Carnegie Mellon University* , 1978, pp.12-20.

42. **Couzens, A, Skitmnore, M, Thorpe, T & McAffer, R**, "Strategic Information Requirements in Contract Bidding: Framework for Development of a Decision Support System", *Civil Engineering Systems* , Vol. 13, pp.121-139

43. **Chan, DY & Bedworth, DD**, "Design of a Scheduling System for Flexible Manufacturing Cells", *International Journal of Production Research* , Vol 28, #11, pp. 2037-2049.

44. **Ginting, DS, Nahavandi, S & Hurley, S**, "Development of a Scheduling Simulation Tool", *Masterate Thesis* , Massey University, Palmerston North, 1995

45. **Fischer, K, Muller, JP & Pischel, M,** "Cooperative Transportation Scheduling: An Application Domain for DAI", *Applied Artificial Intelligence,* Vol.10, pp.1-33, 1996

46. **Harris, M & Raviv, A** , "Allocation Mechanisms and the Design of Auction", *Econometrica 49,* #6, 1981, pp. 1477-1499

47. **Malone, T & Smith, S ,** "Tradeoffs in Designing Organisations: Implications for new forms of Human Organisations and computer systems", *Working Paper # 151-84,* Sloane School of Management, Massachusetts Institute of Technology, 1984.

48. **Shaw, Michael J ,** "A Distributed Scheduling Method For Computer Integrated Manufacturing: The Use Of Local Area Networks In Cellular Systems", *International Journal Of Production Research,* Vol 25 ,#9, pp. 1285 - 1303, 1987.

49. **Tsukada, TK & Shin, KG,** "PRIAM: Polite Rescheduler for Intelligent Manufacturing", *IEEE Transactions on Robotics and Automation* , Vol 12, #2, April 1996

50. **Mullins, JH,** "CIM Startup in a low Manufacturing Density, High Tech Environment", *Proceedings of the 3rd International Symposium on Robotics and Manufacturing",* July 1990, British Columbia, Canada.

51. **Skevington, C & Hsu, C,** "Manufacturing Architecture for Integrated Systems", *Robotics & Computer Integrated Manufacturing,* Vol. 4, #3/4, pp.619-623, 1988

52. **Ferme, WD,** "CIM in Australia (1,2): Ten Years On", *1993 Australian Conference on Manufacturing Engineering",* Adelaide, 22-24 November,1993

53. **Inglis, AE,** "Implementing CIM in a Networked PC Environment", *1993 Australian Conference on Manufacturing Engineering",* Adelaide, 22-24 November,1993

54. **Dilts, DM, Boyd N.P. and Whorms H.H ,** "The Evolution Of Control Architectures for Automated Manufacturing Systems", *Journal of Manufacturing Systems,* Vol 10, #1, pp. 79-93 (1991).

55. **Smith, Reid G ,** "The Contract Net Protocol: High - Level Communication and Control in a Distributed Problem Solver", *IEEE Transactions on Computers,* Vol C-29, #12, Dec 1980.

56. **Schemmerhorn, JR ,** *Management for Productivity - 4th Edition*

57. **Veeramani, Dharmaraj ,** "Task and Resource Allocation Via Auctioning", *Proceedings Of The 1993 Winter Simulation Conference,* ed J.J Swain, D Goldzman, R.C. Crain and J.R. Wilson

58. **Tilley, Kevin J and Williams, David J ,** "Modelling Of Communications and Control in an Auction-based Manufacturing Control System"

59. **Mahmood, F ; Dooley K.J and Starr, PJ** , "An Investigation of Dynamic Group Scheduling Heuristics in a Job Shop Manufacturing Cell", *International Journal of Production Research*, 1990 , Vol 28, # 9, pp 169-1711

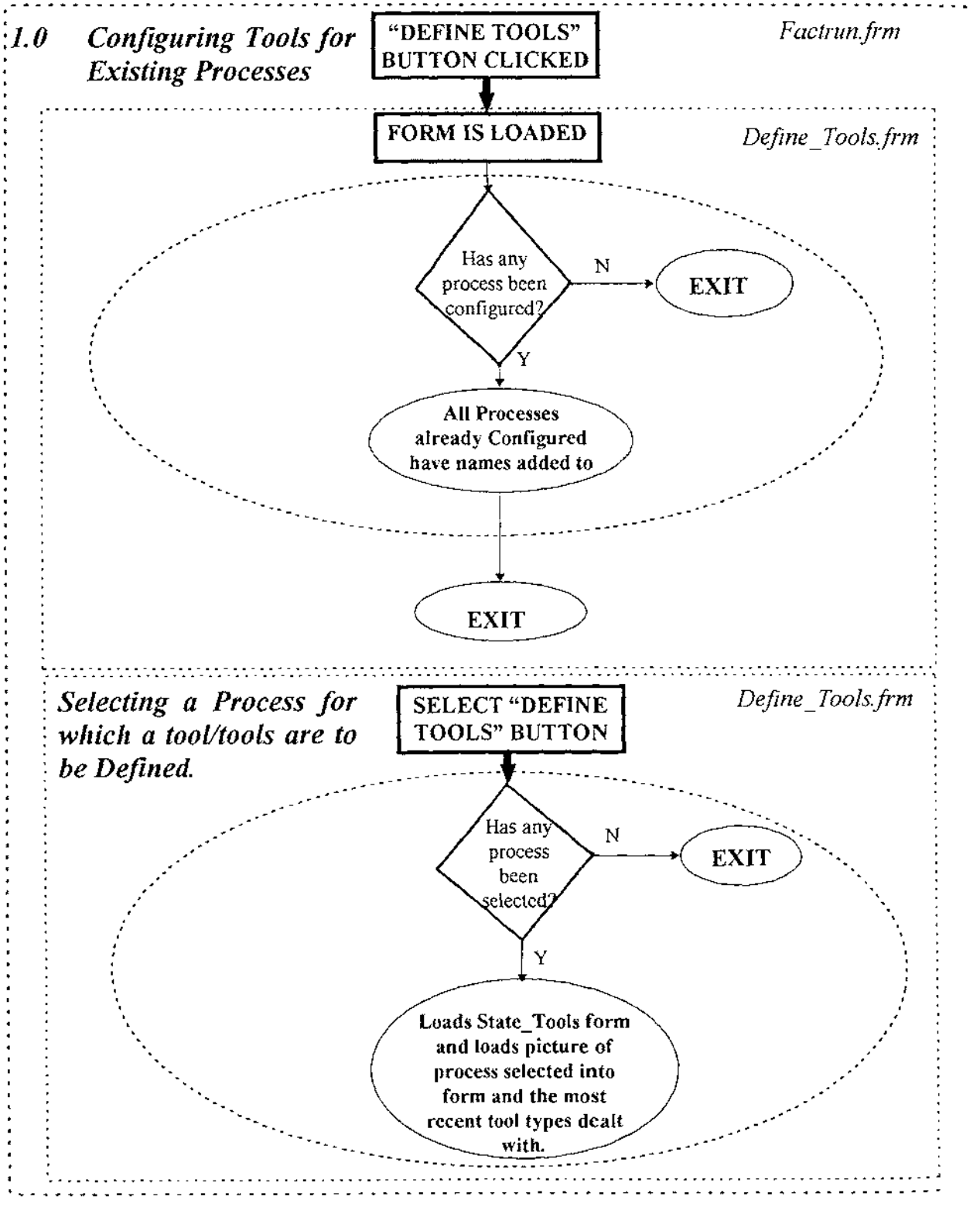
60. **Kusiak, A**, "Group Technology in Flexible Manufacturing Systems" *Intelligent Manufacturing Systems"*, Prentice Hall, Englewood Cliffs, New Jersey

Appendix A

Hybrid FMS Algorithms

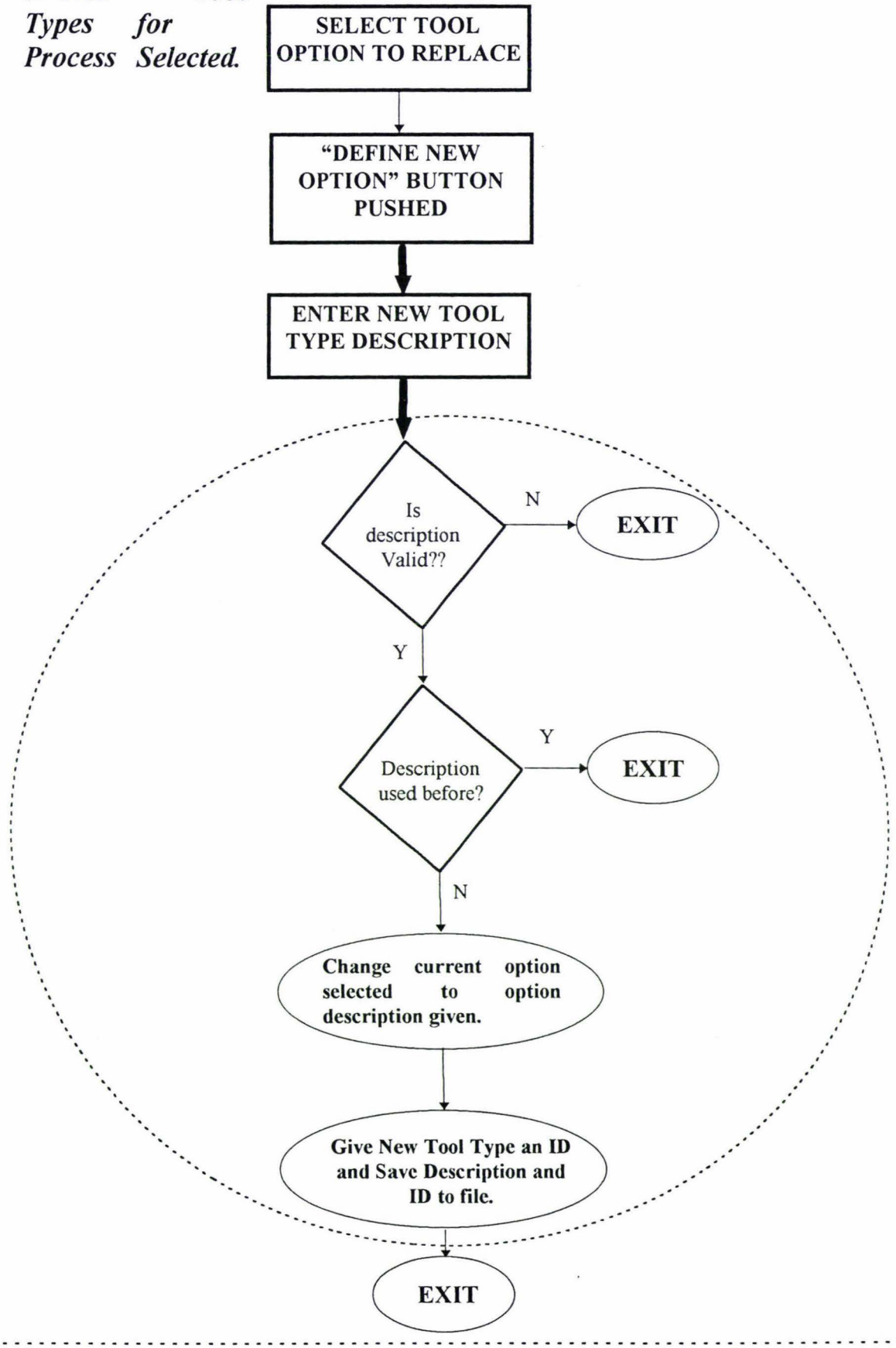
This Appendix provides all the major algorithms associated with the development of the Hybrid FMS. In so doing it gives a condensed view into the nature of the implementation so that the reader has an alternative to looking into the more comprehensive detail of earlier chapters. Alternately it provides an additional tool for future research as the algorithms can be extended as required.

A.1 Resource Configuration Service - Tooling Configuration Algorithm



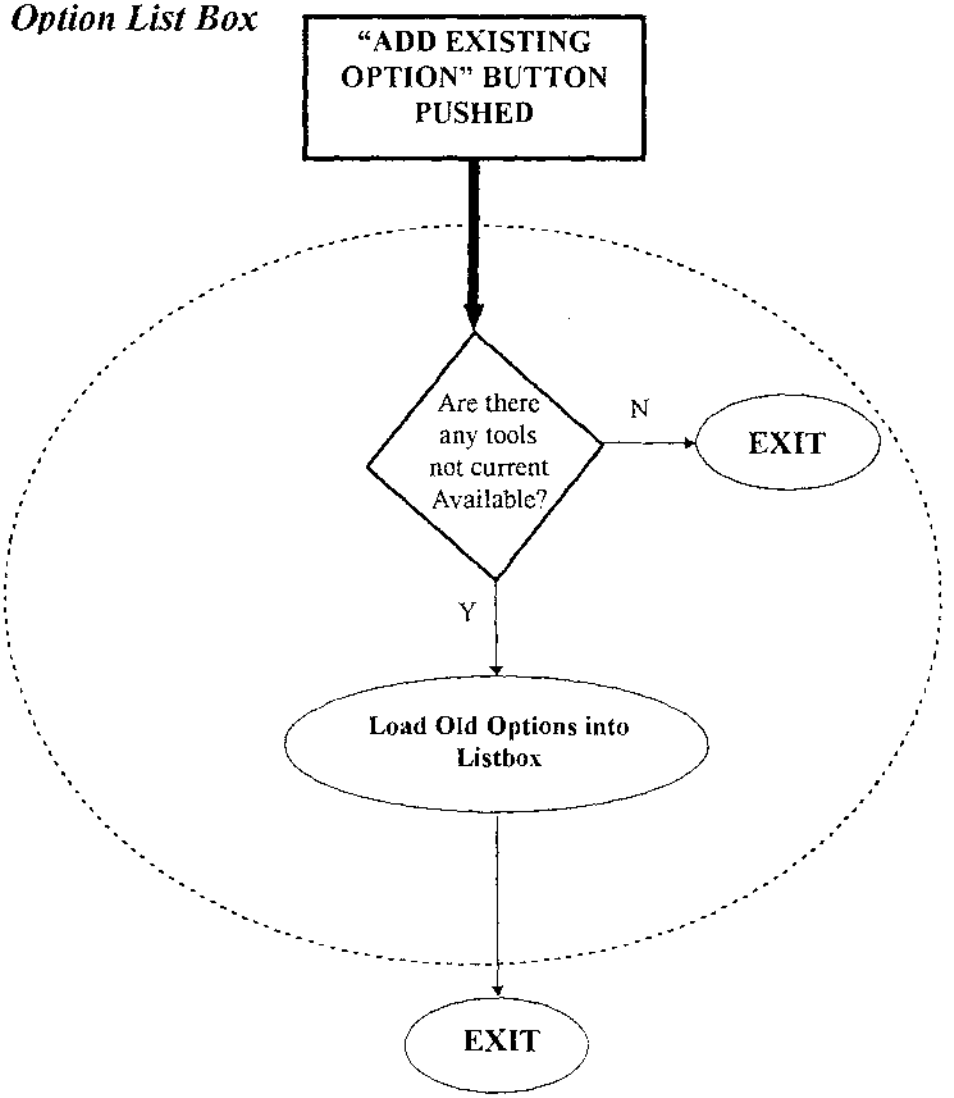
1.1 Defining first or New Tool Types for Process Selected.

StateTools.frm



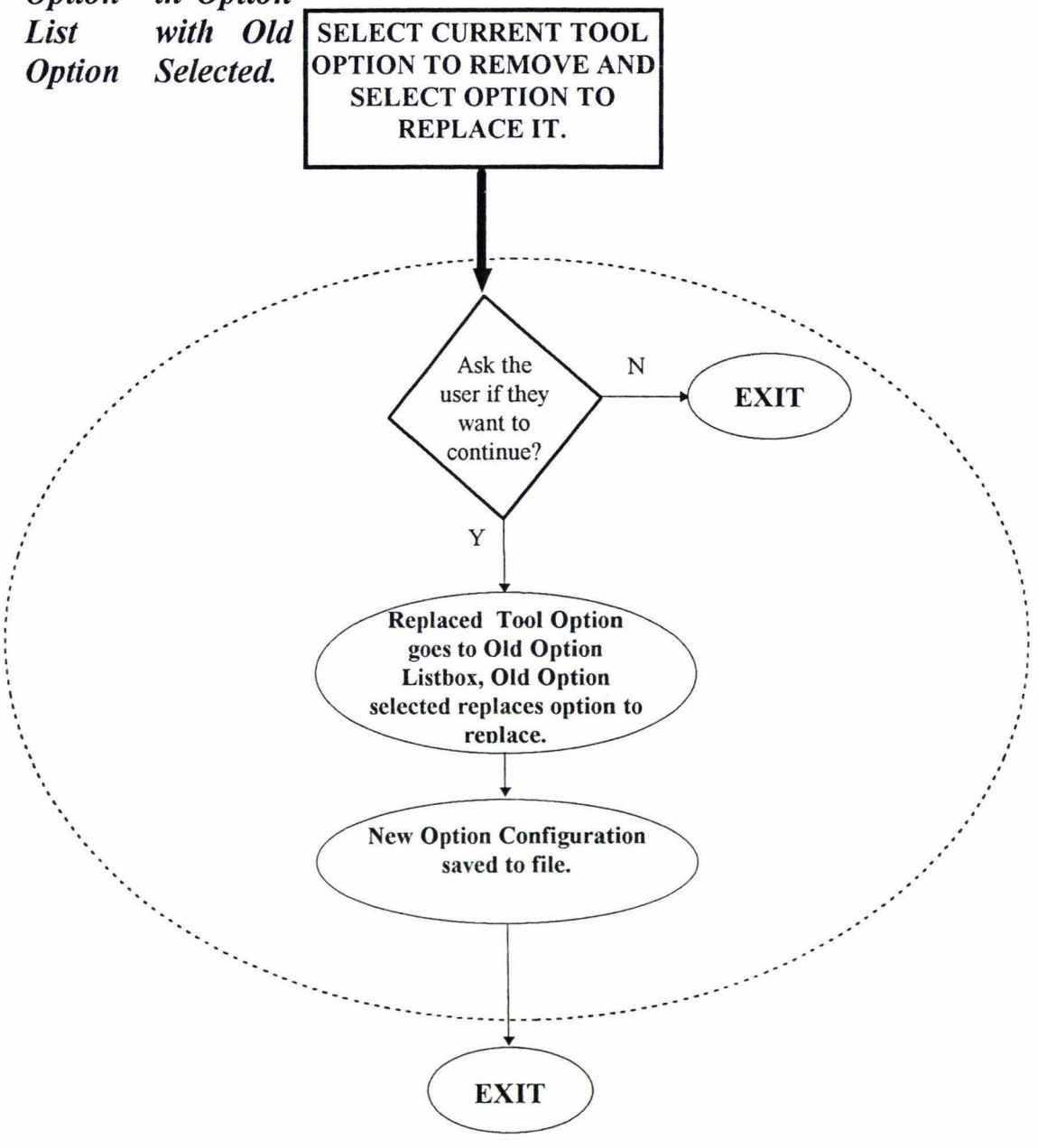
StateTools.frm

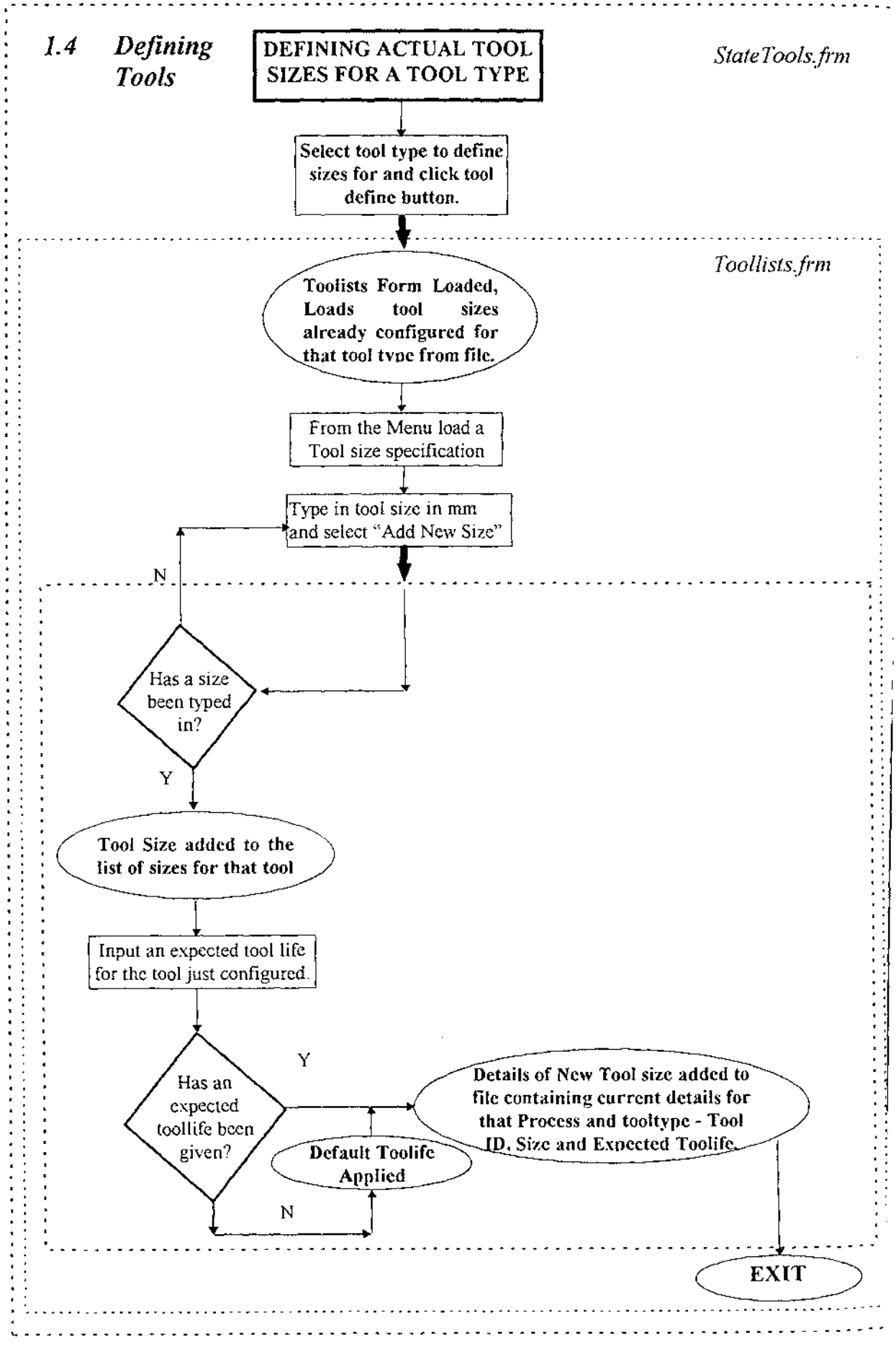
1.2 Refreshening or Displaying Old Option List Box



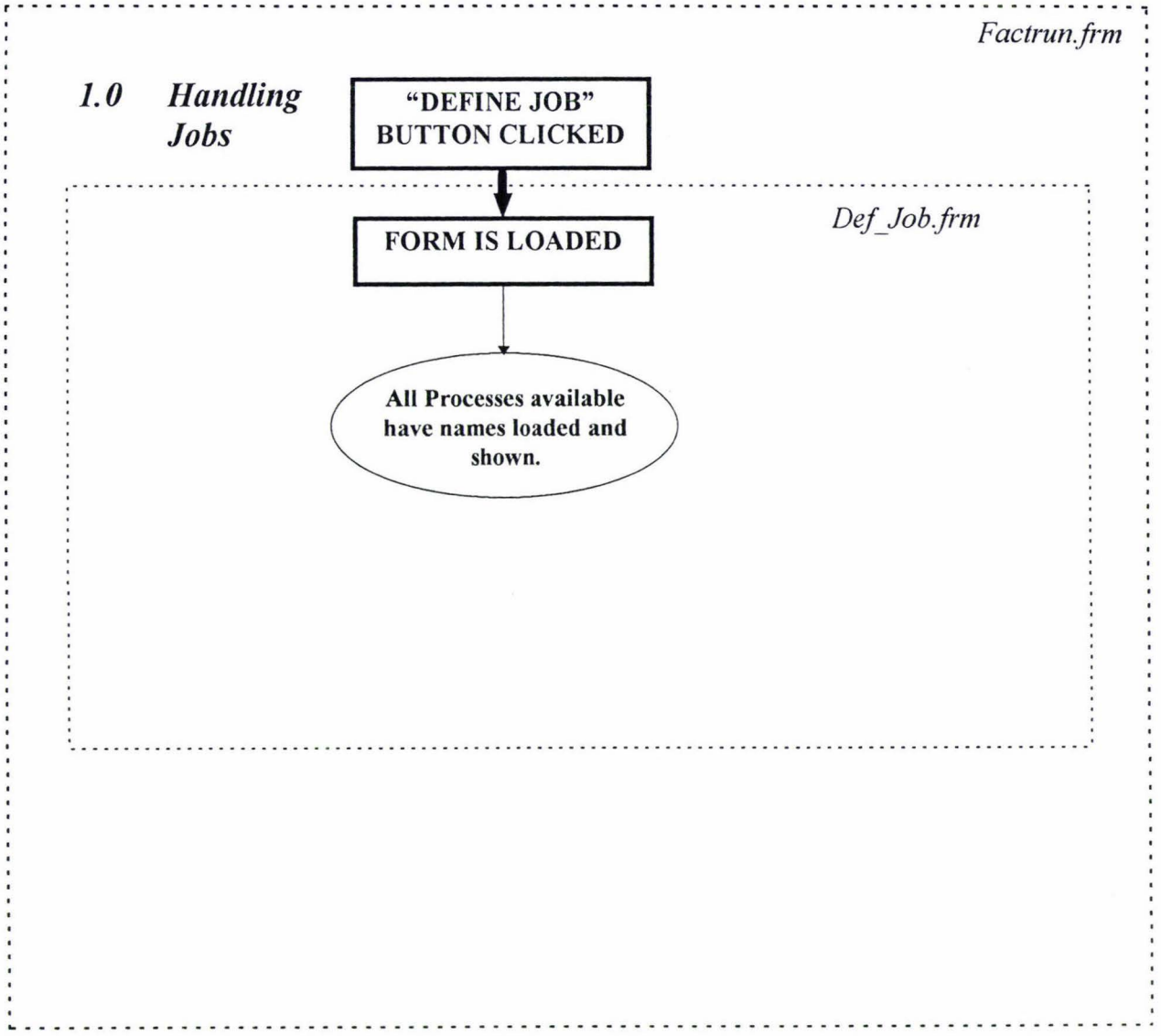
StateTools.frm

1.3 Changing Option in Option List with Old Option Selected.



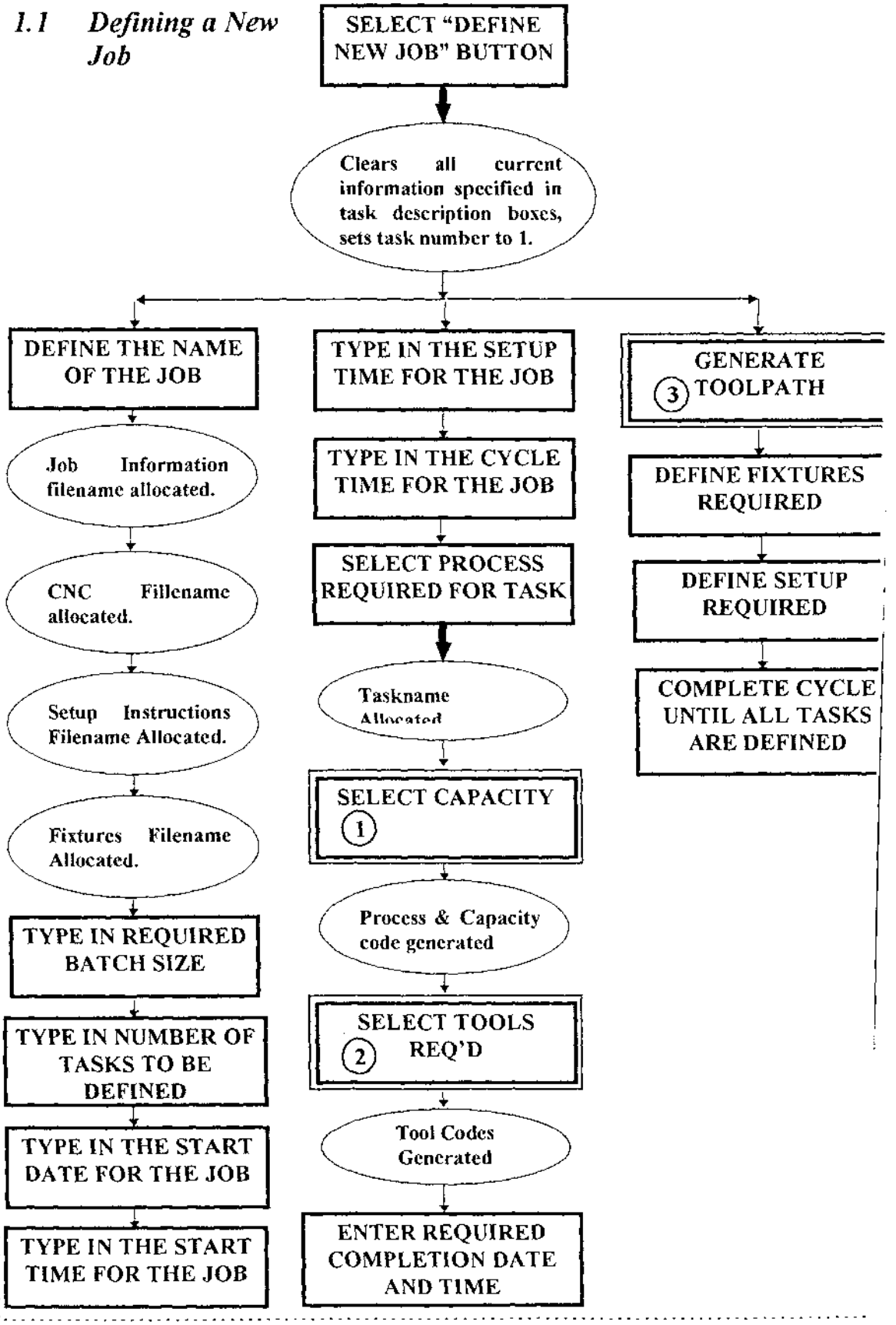

1.4 Defining Tools
DEFINING ACTUAL TOOL SIZES FOR A TOOL TYPE
StateTools.frm
Select tool type to define sizes for and click tool define button.
Toollists.frm
Toolists Form Loaded, Loads tool sizes already configured for that tool type from file.
From the Menu load a Tool size specification
Type in tool size in mm and select "Add New Size"
N
Has a size been typed in?
Y
Tool Size added to the list of sizes for that tool
Input an expected tool life for the tool just configured.
Has an expected toollife been given?
Y
Details of New Tool size added to file containing current details for that Process and tooltype - Tool ID, Size and Expected Toolife.
Default Toolife Applied
N
EXIT

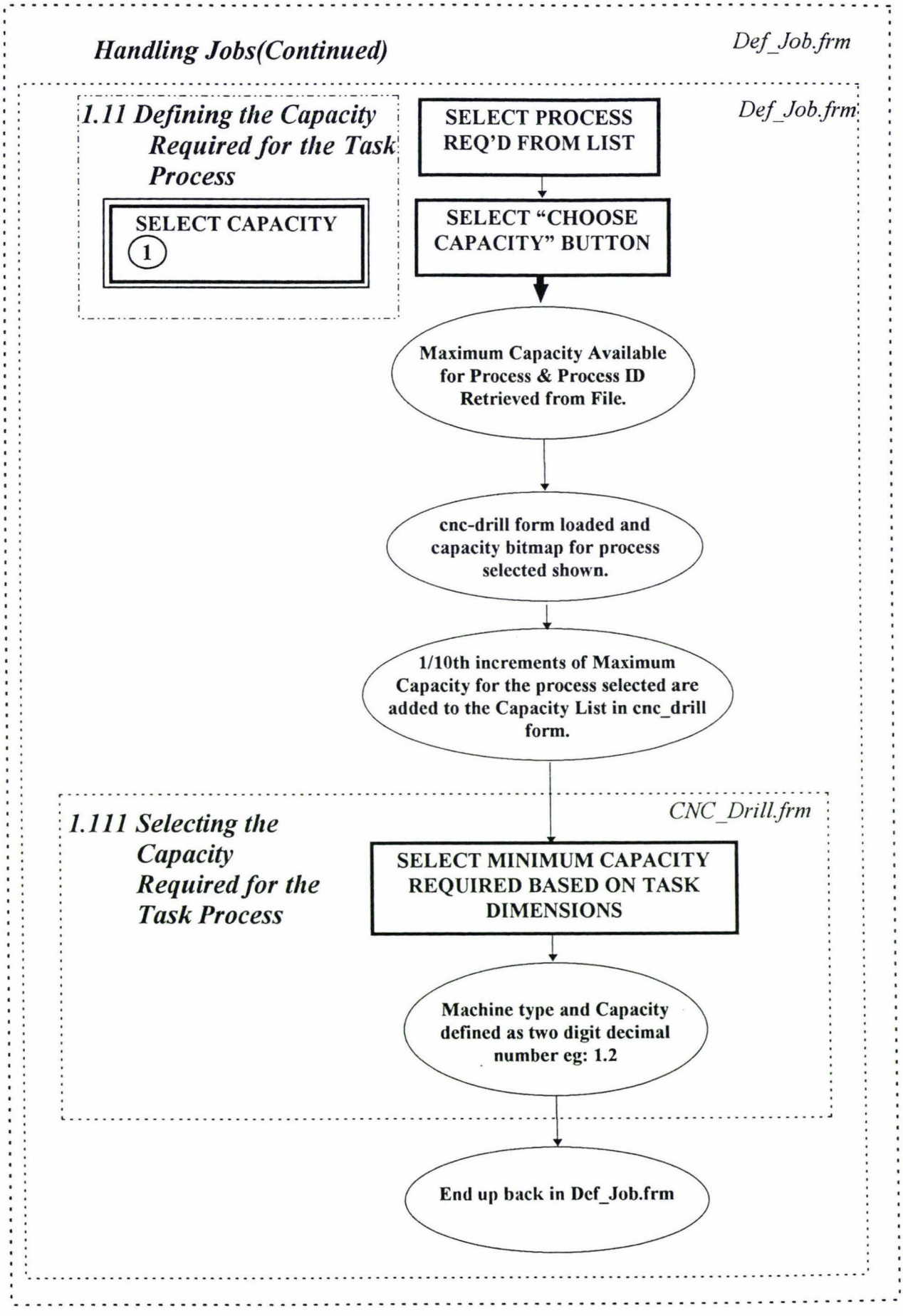
A.2 Job Definition Service Algorithm



Handling Jobs(Continued) *Def_Job.frm*

1.1 Defining a New Job



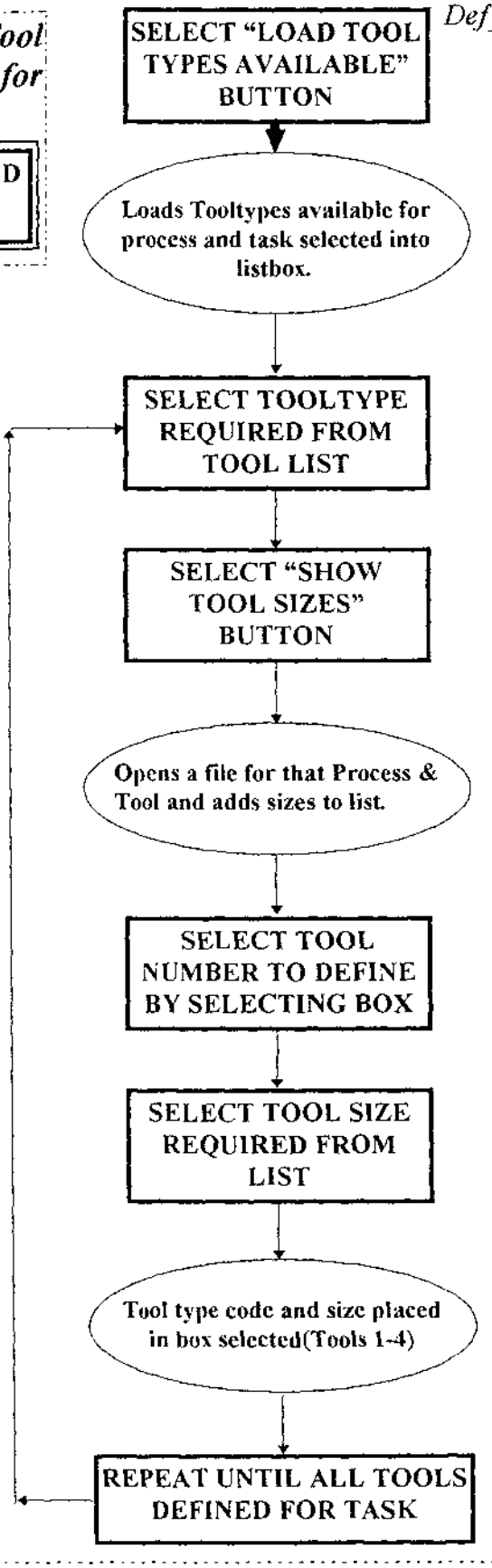

Handling Jobs(Continued)
Def_Job.frm
1.11 Defining the Capacity Required for the Task Process
Def_Job.frm
SELECT CAPACITY
1
SELECT PROCESS REQ'D FROM LIST
SELECT "CHOOSE CAPACITY" BUTTON
Maximum Capacity Available for Process & Process ID Retrieved from File.
cnc-drill form loaded and capacity bitmap for process selected shown.
1/10th increments of Maximum Capacity for the process selected are added to the Capacity List in cnc_drill form.
1.111 Selecting the Capacity Required for the Task Process
CNC_Drill.frm
SELECT MINIMUM CAPACITY REQUIRED BASED ON TASK DIMENSIONS
Machine type and Capacity defined as two digit decimal number eg: 1.2
End up back in Def_Job.frm

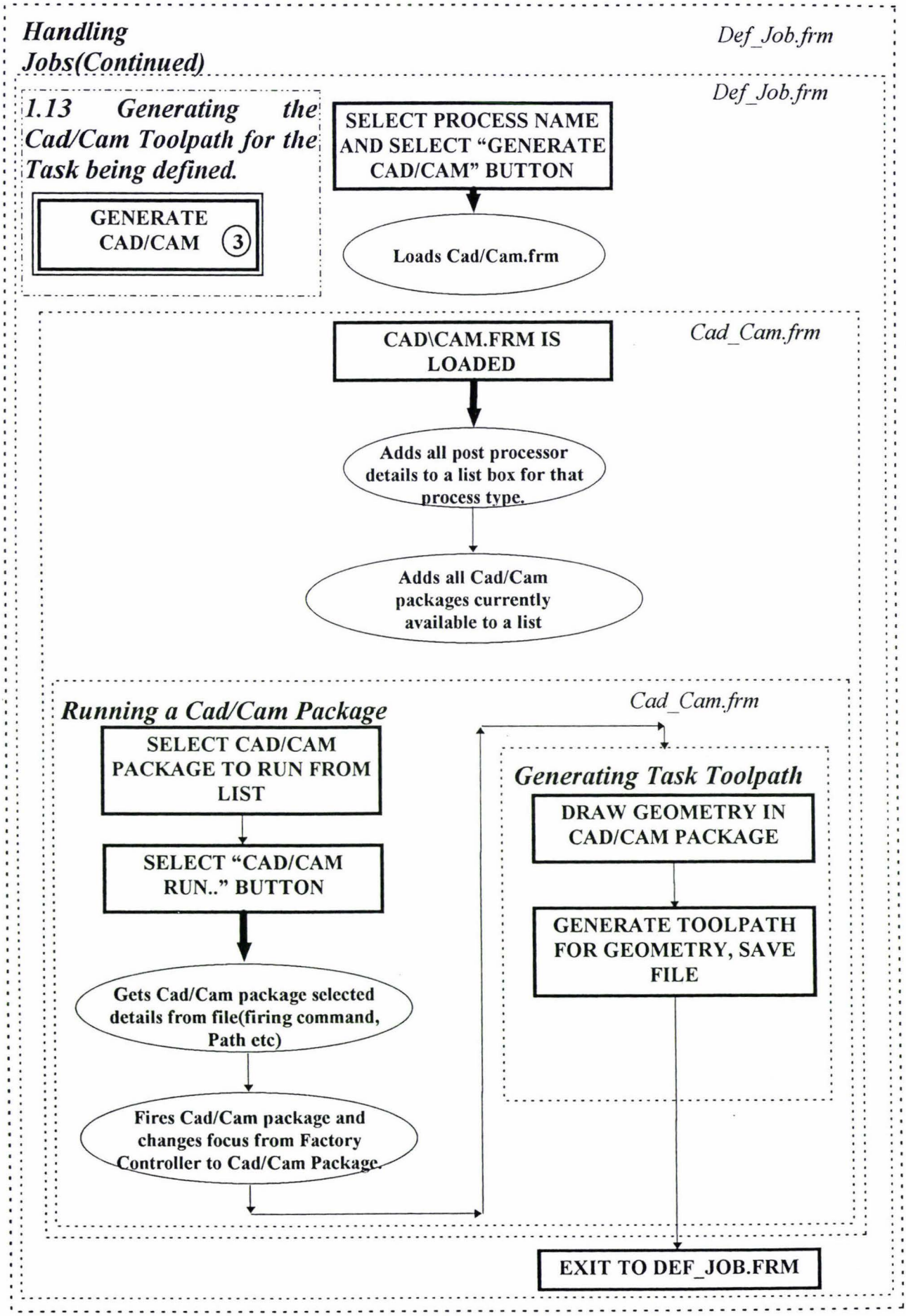
Handling Jobs(Continued) *Def_Job.frm*

1.12 Defining Tool Types and sizes for a Task.

SELECT TOOLS REQ'D (2)

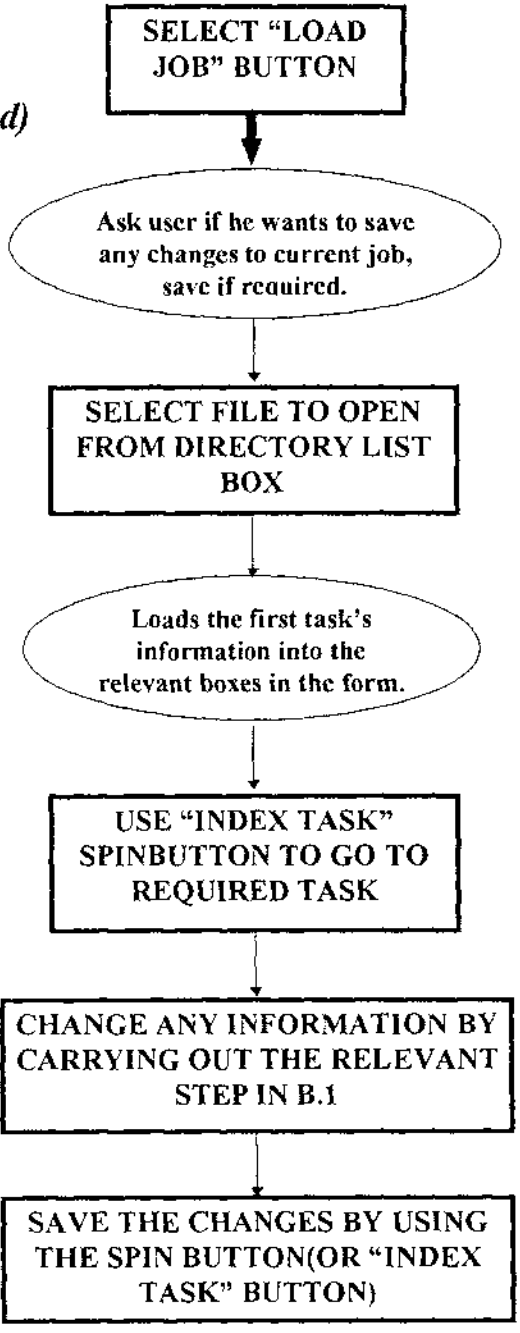
Def_Job.frm




Handling Jobs(Continued)
Def_Job.frm
Def_Job.frm
1.13 Generating the Cad/Cam Toolpath for the Task being defined.
GENERATE CAD/CAM (3)
SELECT PROCESS NAME AND SELECT "GENERATE CAD/CAM" BUTTON
Loads Cad/Cam.frm
Cad_Cam.frm
CAD\CAM.FRM IS LOADED
Adds all post processor details to a list box for that process type.
Adds all Cad/Cam packages currently available to a list
Running a Cad/Cam Package
Cad_Cam.frm
SELECT CAD/CAM PACKAGE TO RUN FROM LIST
SELECT "CAD/CAM RUN.." BUTTON
Gets Cad/Cam package selected details from file(firing command, Path etc)
Fires Cad/Cam package and changes focus from Factory Controller to Cad/Cam Package.
Generating Task Toolpath
DRAW GEOMETRY IN CAD/CAM PACKAGE
GENERATE TOOLPATH FOR GEOMETRY, SAVE FILE
EXIT TO DEF_JOB.FRM

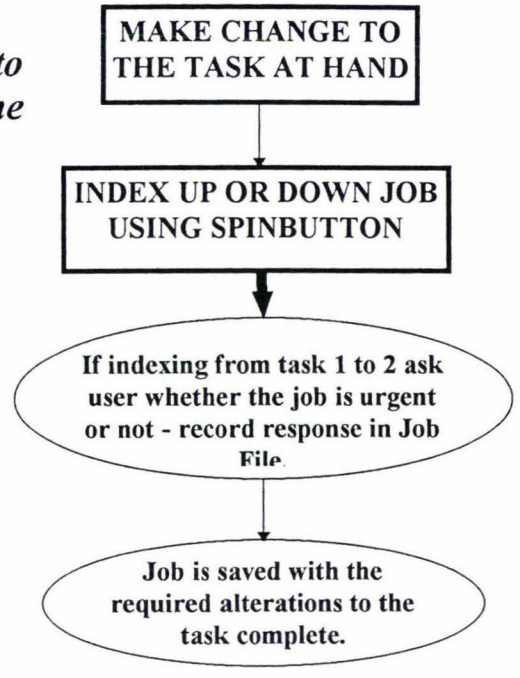
Handling Jobs(Continued) *Def_Job.frm*

1.2 Redefining Existing Jobs(not loaded) *Def_Job.frm*

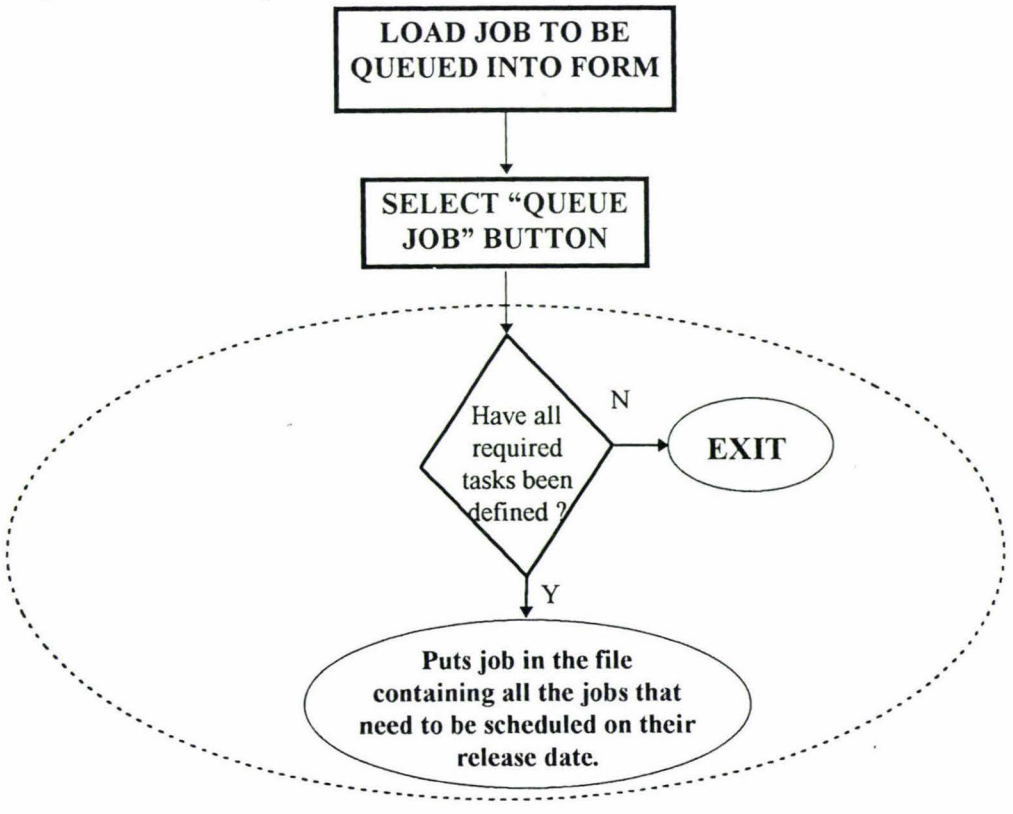


Handling Jobs(Continued) *Def_Job.frm*

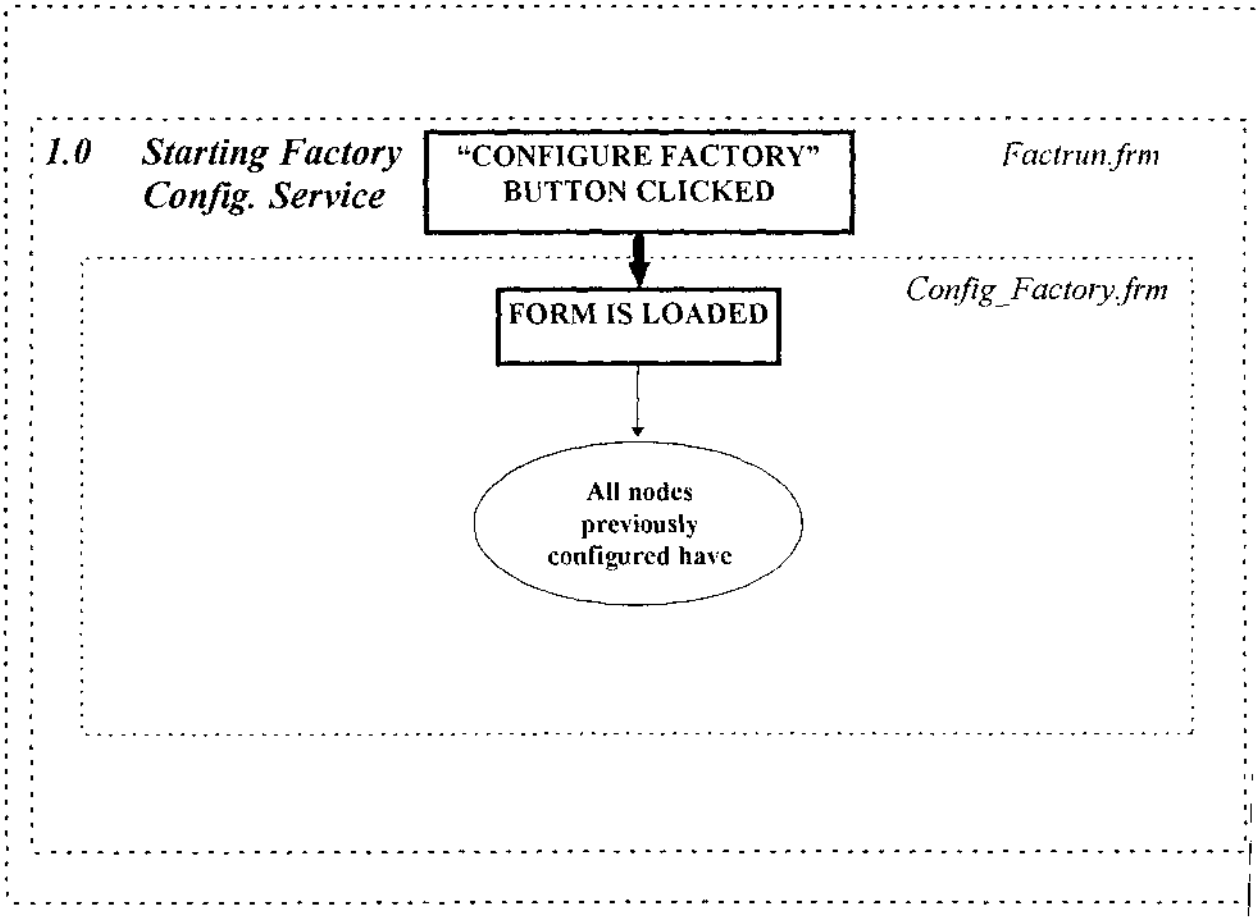
1.3 Saving Changes to Jobs using the Spin Button *Def_Job.frm*

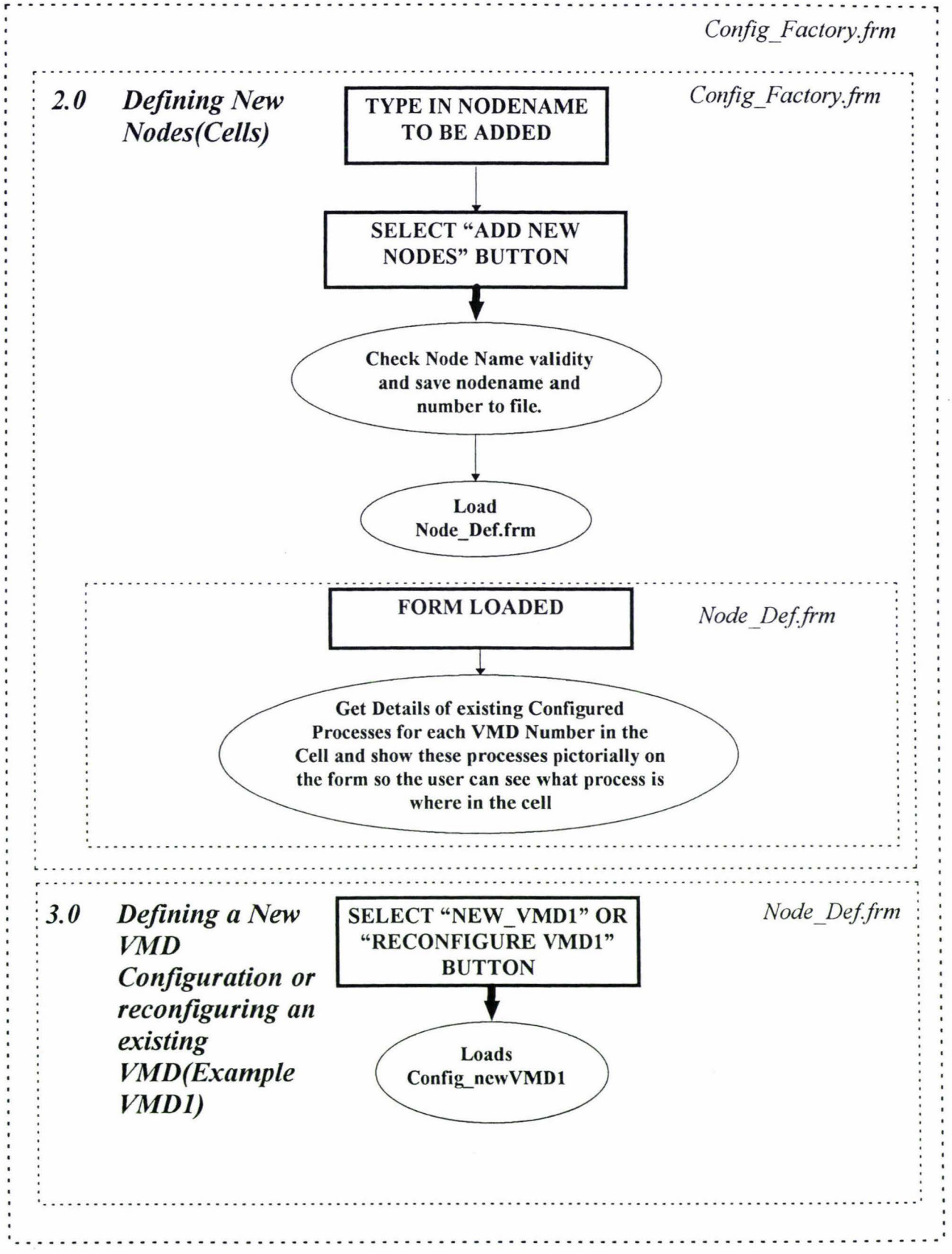


1.4 Queuing Jobs for Scheduling *Def_Job.frm*



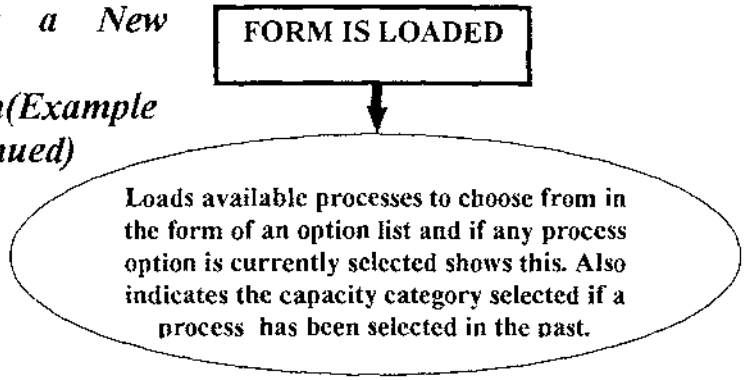
A.3 Factory Configuration Service




Config_Factory.frm
2.0 Defining New Nodes(Cells)
Config_Factory.frm
TYPE IN NODENAME TO BE ADDED
SELECT "ADD NEW NODES" BUTTON
Check Node Name validity and save nodename and number to file.
Load Node_Def.frm
FORM LOADED
Node_Def.frm
Get Details of existing Configured Processes for each VMD Number in the Cell and show these processes pictorially on the form so the user can see what process is where in the cell
3.0 Defining a New VMD Configuration or reconfiguring an existing VMD(Example VMD1)
SELECT "NEW_VMD1" OR "RECONFIGURE VMD1" BUTTON
Node_Def.frm
Loads Config_newVMD1

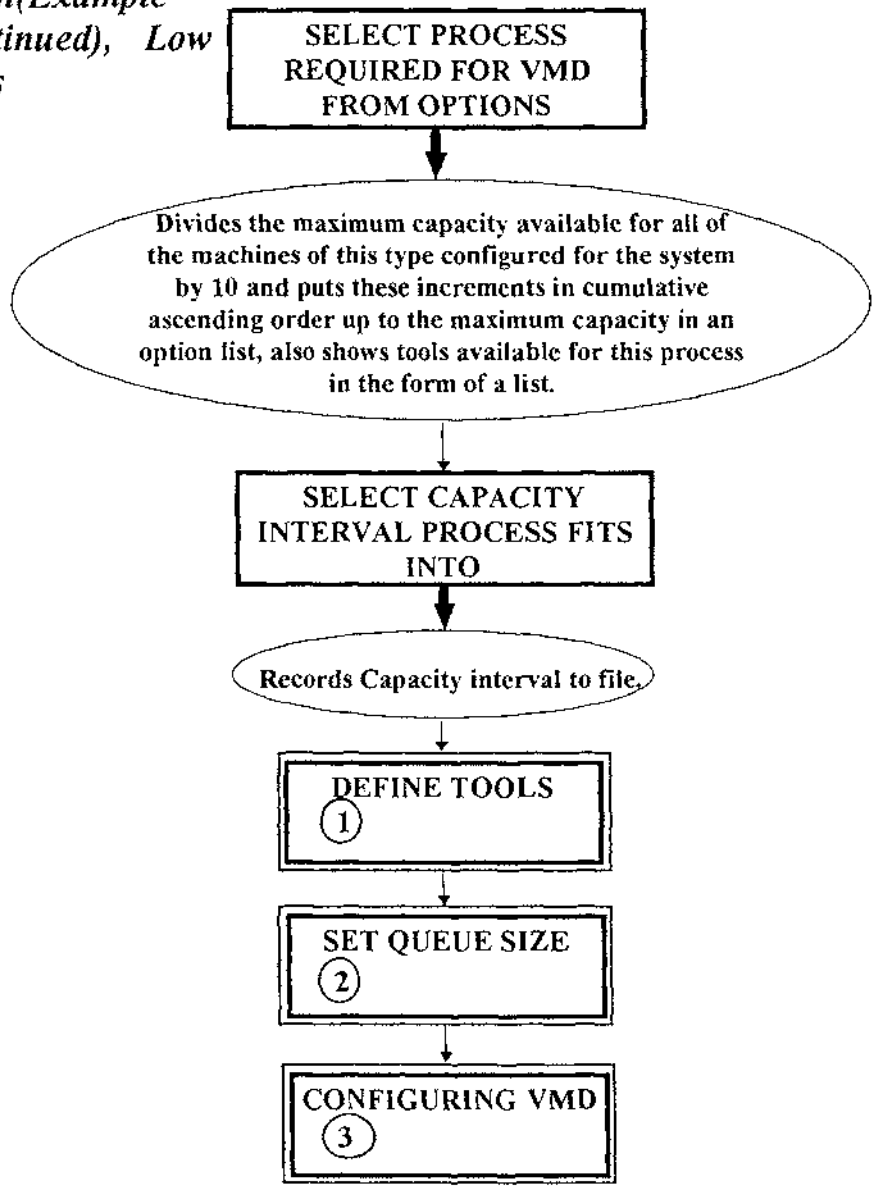
Defining or reconfiguring a New VMD Configuration(Example VMD1-Continued)

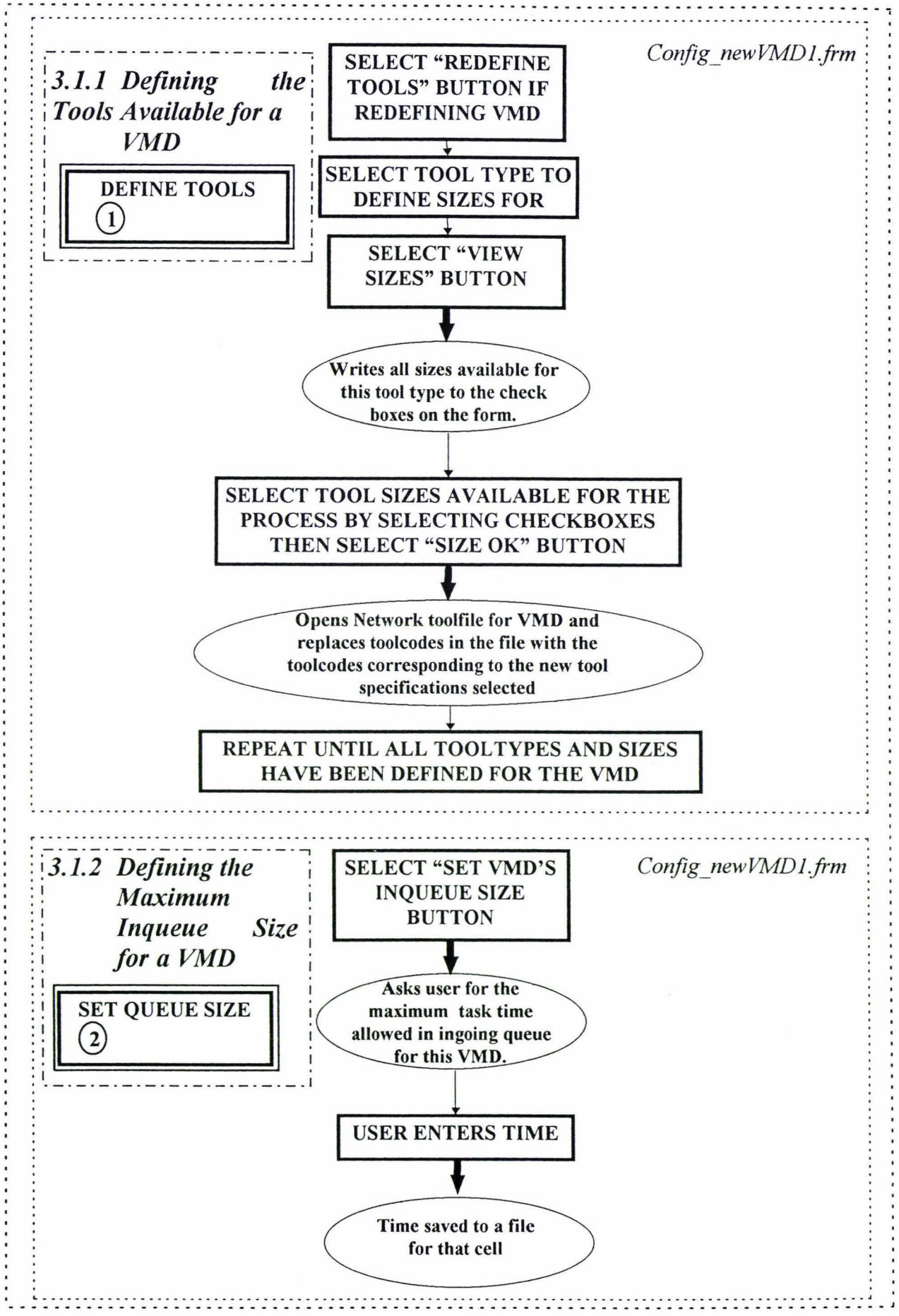
Config_newVMD1.frm

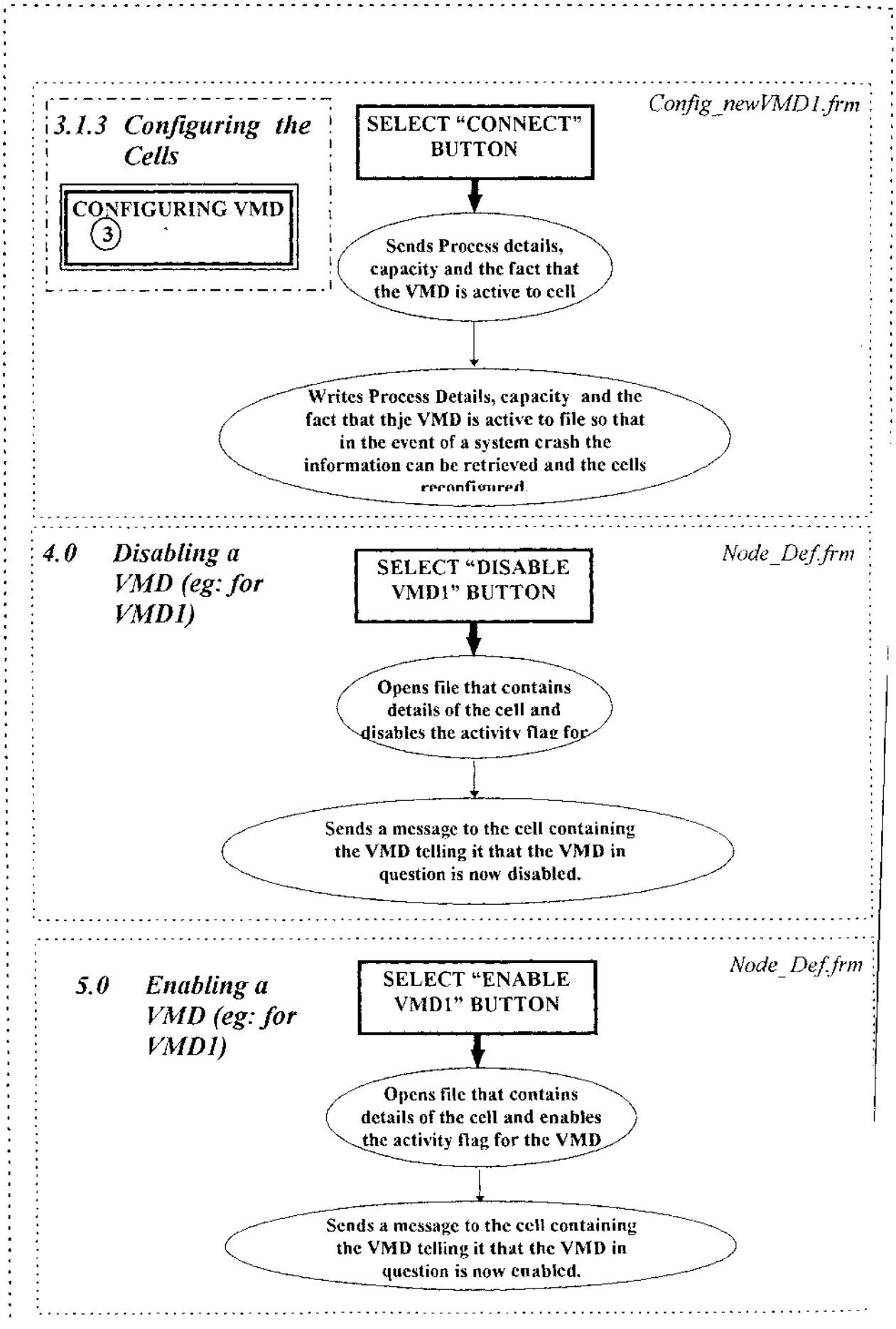


3.1 Defining a New VMD Configuration(Example VMD1- Continued), Low Level Actions

Config_newVMD1.frm

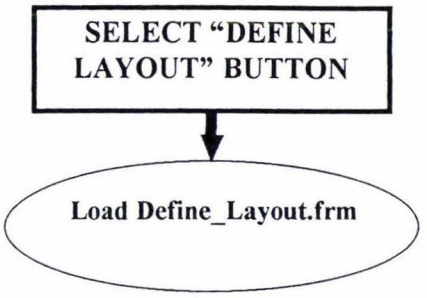



3.1.1 Defining the Tools Available for a VMD
DEFINE TOOLS
1
Config_newVMD1.frm
SELECT "REDEFINE TOOLS" BUTTON IF REDEFINING VMD
SELECT TOOL TYPE TO DEFINE SIZES FOR
SELECT "VIEW SIZES" BUTTON
Writes all sizes available for this tool type to the check boxes on the form.
SELECT TOOL SIZES AVAILABLE FOR THE PROCESS BY SELECTING CHECKBOXES THEN SELECT "SIZE OK" BUTTON
Opens Network toolfile for VMD and replaces toolcodes in the file with the toolcodes corresponding to the new tool specifications selected
REPEAT UNTIL ALL TOOLTYPES AND SIZES HAVE BEEN DEFINED FOR THE VMD
3.1.2 Defining the Maximum Inqueue Size for a VMD
SET QUEUE SIZE
2
Config_newVMD1.frm
SELECT "SET VMD'S INQUEUE SIZE BUTTON
Asks user for the maximum task time allowed in ingoing queue for this VMD.
USER ENTERS TIME
Time saved to a file for that cell


3.1.3 Configuring the Cells
CONFIGURING VMD
3
Config_newVMD1.frm
SELECT "CONNECT" BUTTON
Sends Process details, capacity and the fact that the VMD is active to cell
Writes Process Details, capacity and the fact that thje VMD is active to file so that in the event of a system crash the information can be retrieved and the cells
4.0 Disabling a VMD (eg: for VMD1)
Node_Def.frm
SELECT "DISABLE VMD1" BUTTON
Opens file that contains details of the cell and disables the activity flag for
Sends a message to the cell containing the VMD telling it that the VMD in question is now disabled.
5.0 Enabling a VMD (eg: for VMD1)
Node_Def.frm
SELECT "ENABLE VMD1" BUTTON
Opens file that contains details of the cell and enables the activity flag for the VMD
Sends a message to the cell containing the VMD telling it that the VMD in question is now enabled.

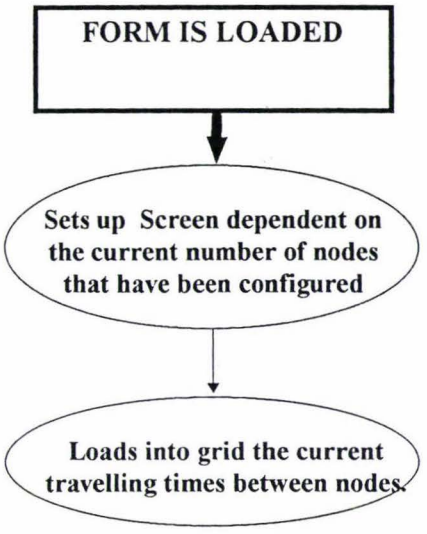
6.0 Defining Inter-Nodal Travel time.

Config_Factory.frm



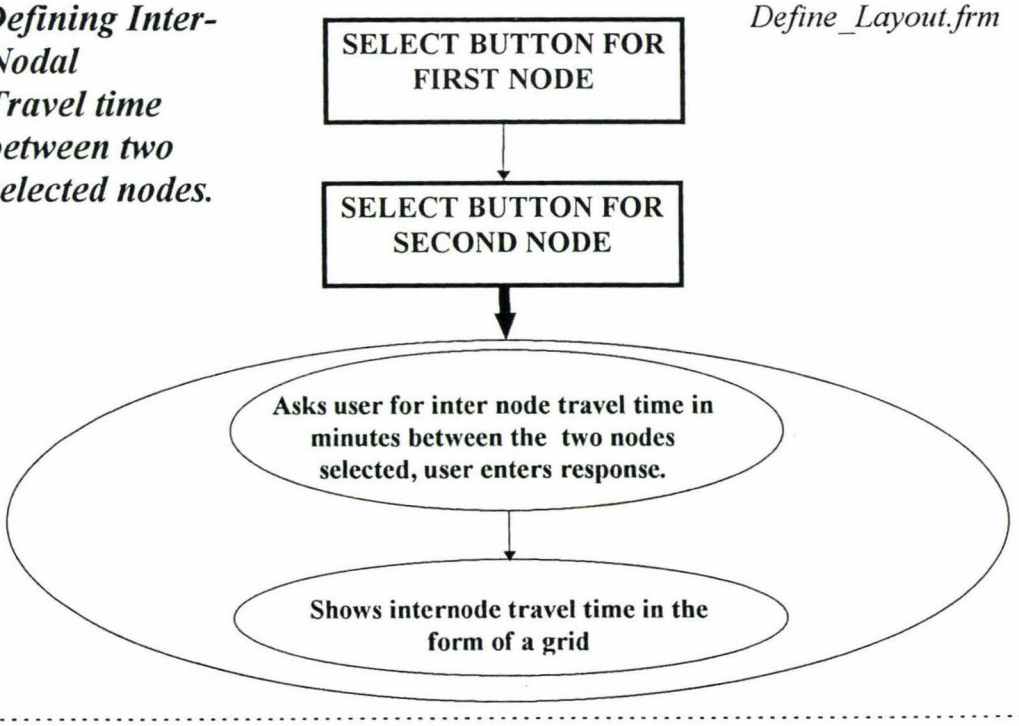
6.0 Defining Inter-Nodal Travel time(Continued)

Define_Layout.frm



6.1 Defining Inter-Nodal Travel time between two selected nodes.

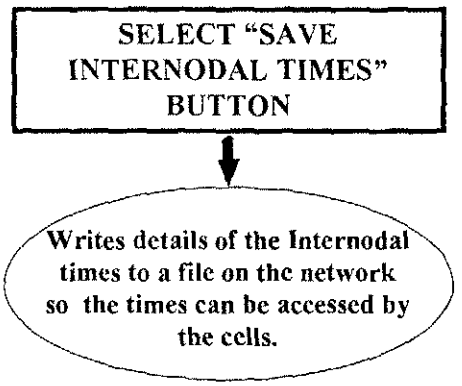
Define_Layout.frm



6.0 Defining Inter-Nodal Travel time. (Continued)

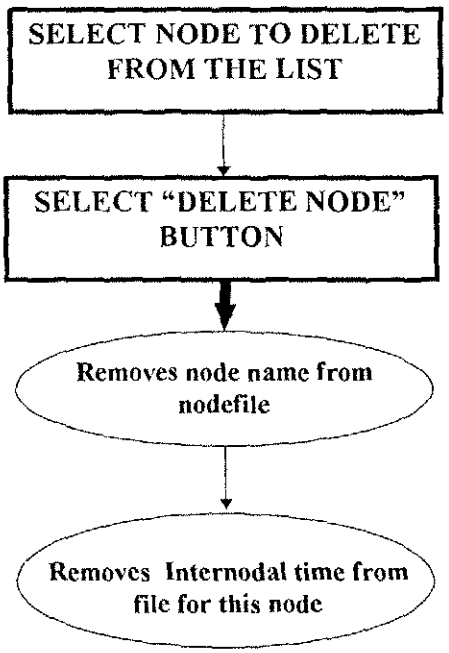
6.2 Saving Inter-Nodal Travel Times

Define_Layout.frm

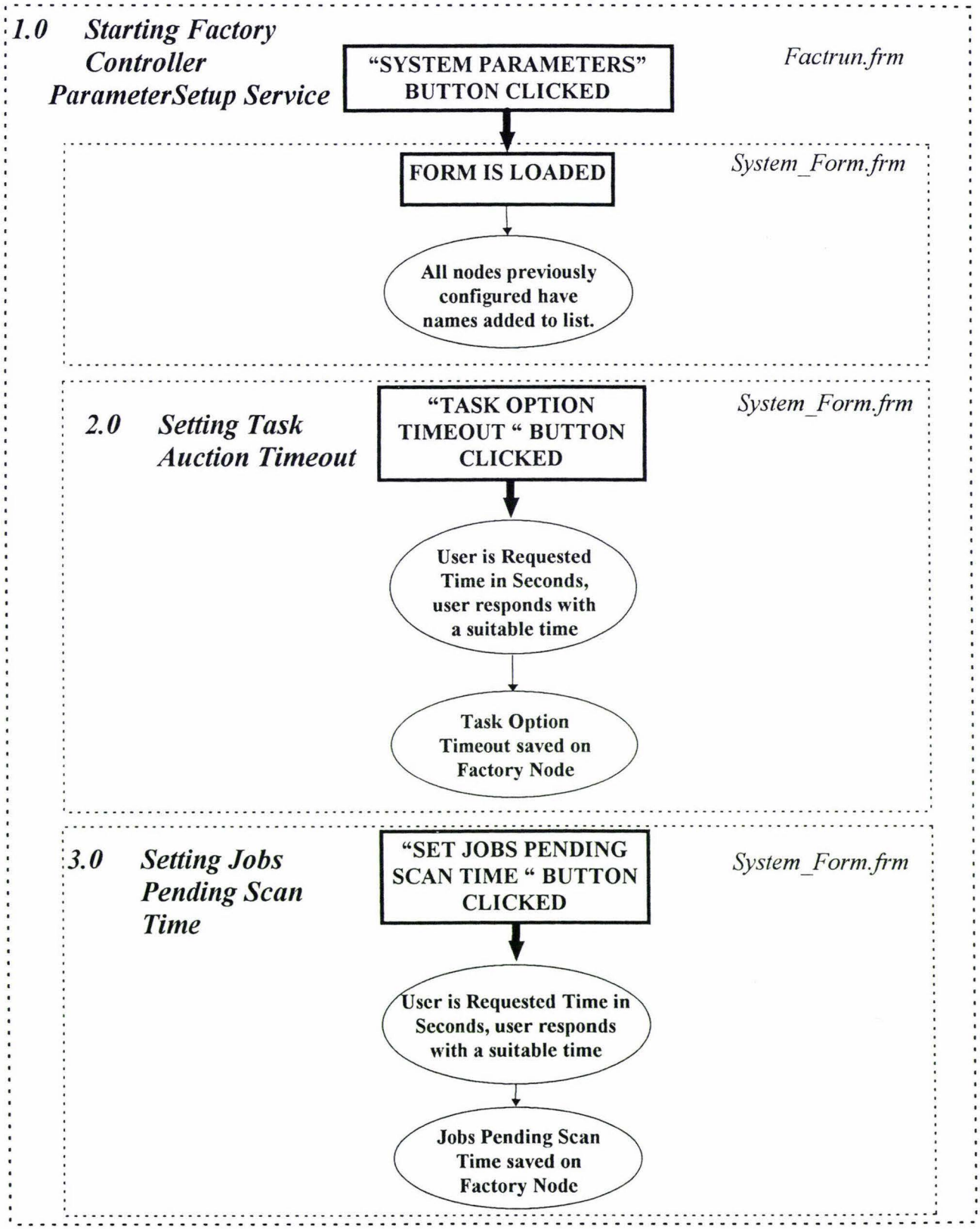


7.0 Removing a previously configured node.

Config_Factory.frm

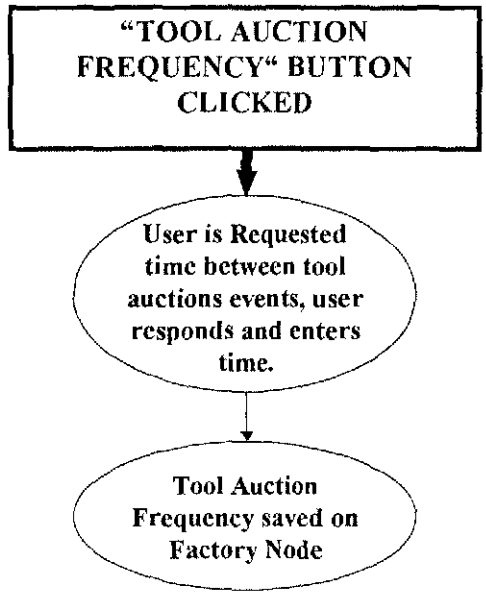


A.4 Factory Controller Parameter Service Algorithm



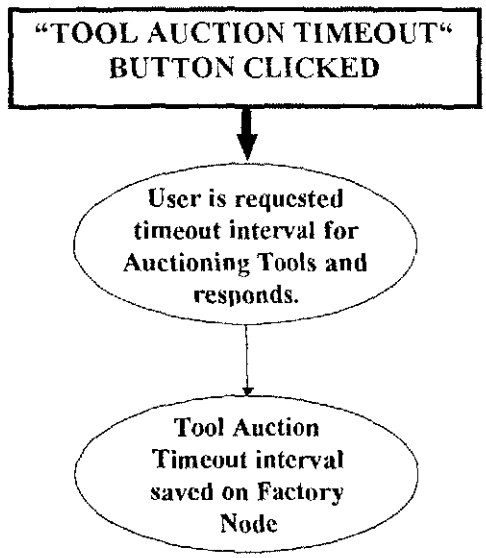
4.0 Setting Tool Auction Frequency

System_Form.frm

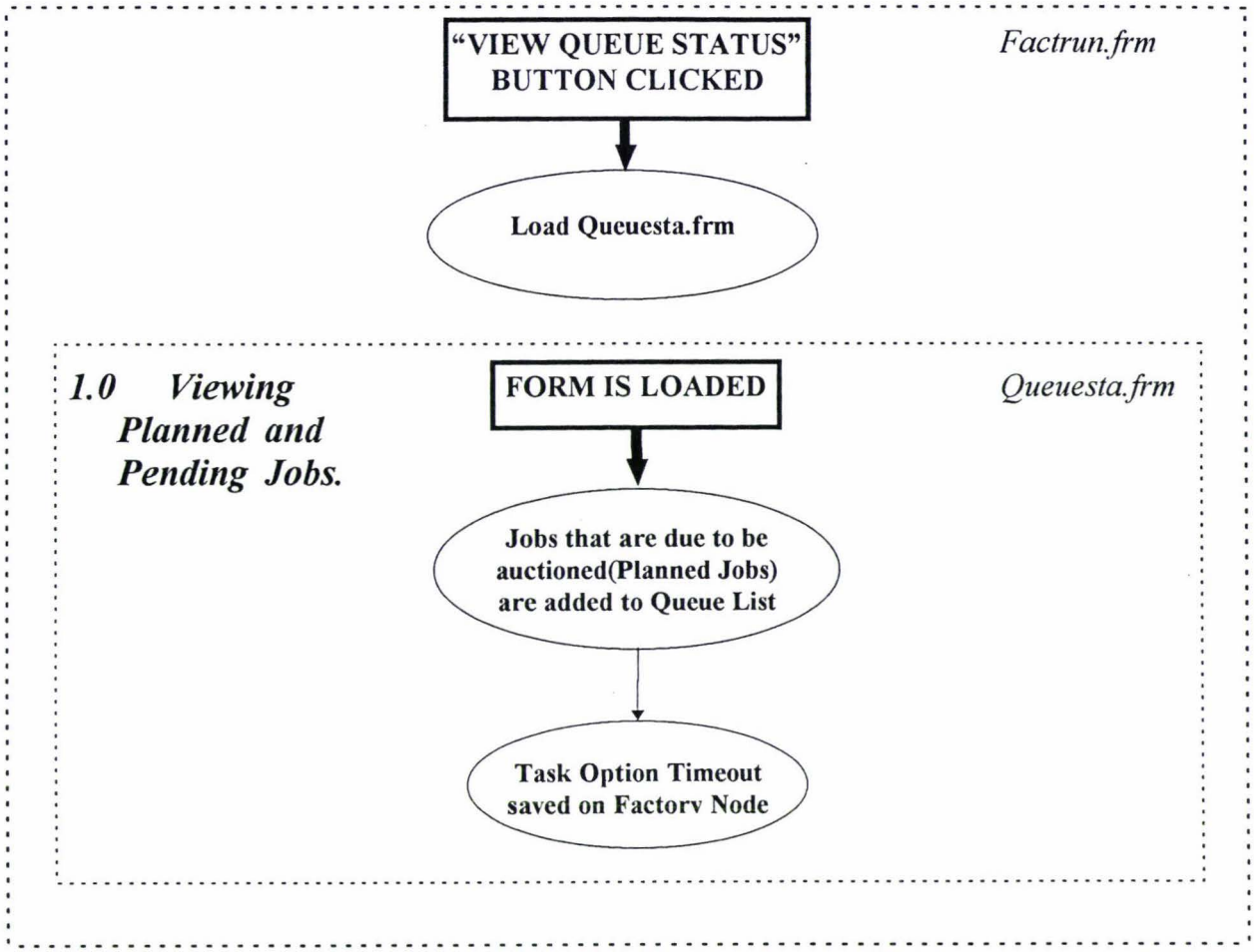


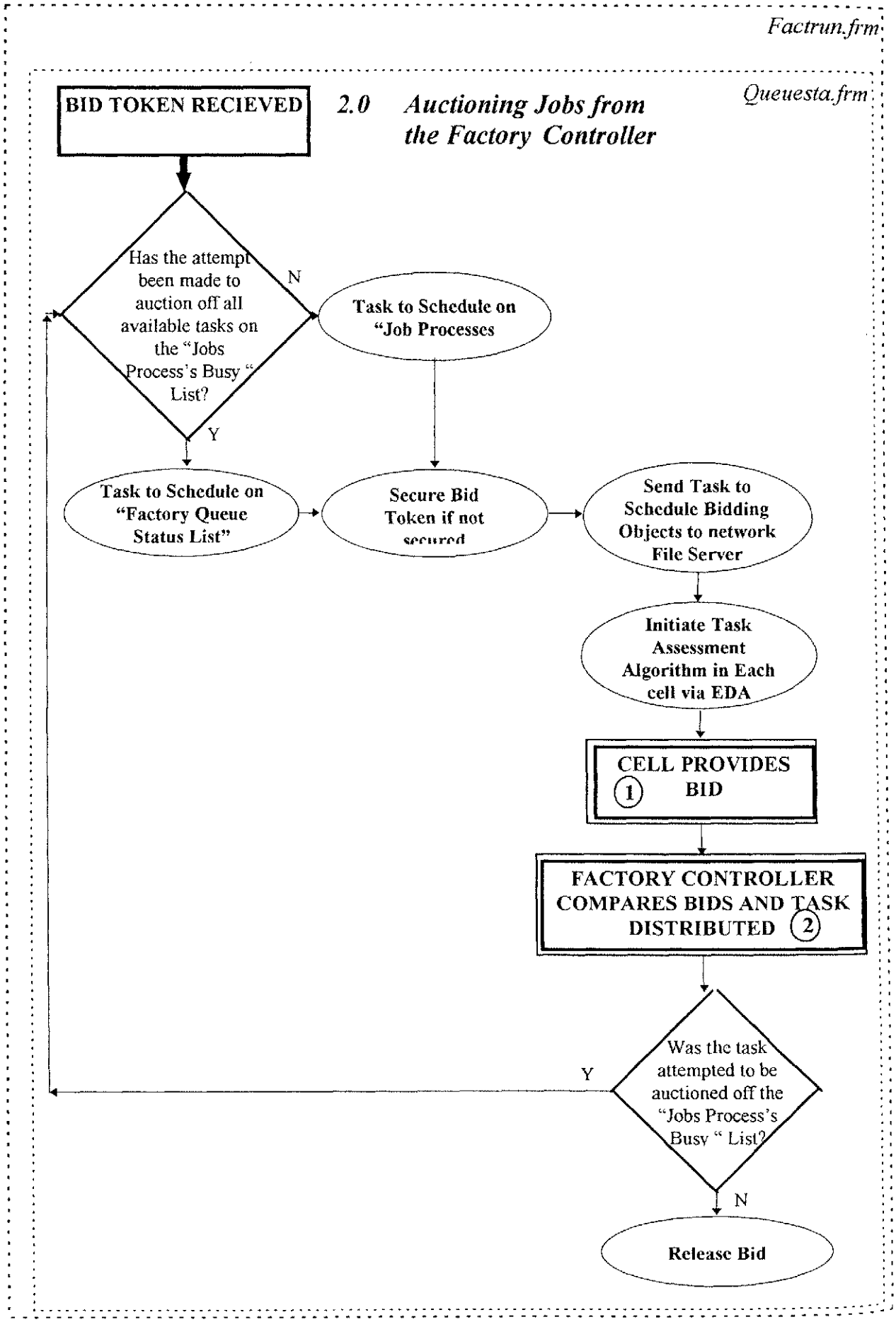
5.0 Setting Tool Auction Timeout

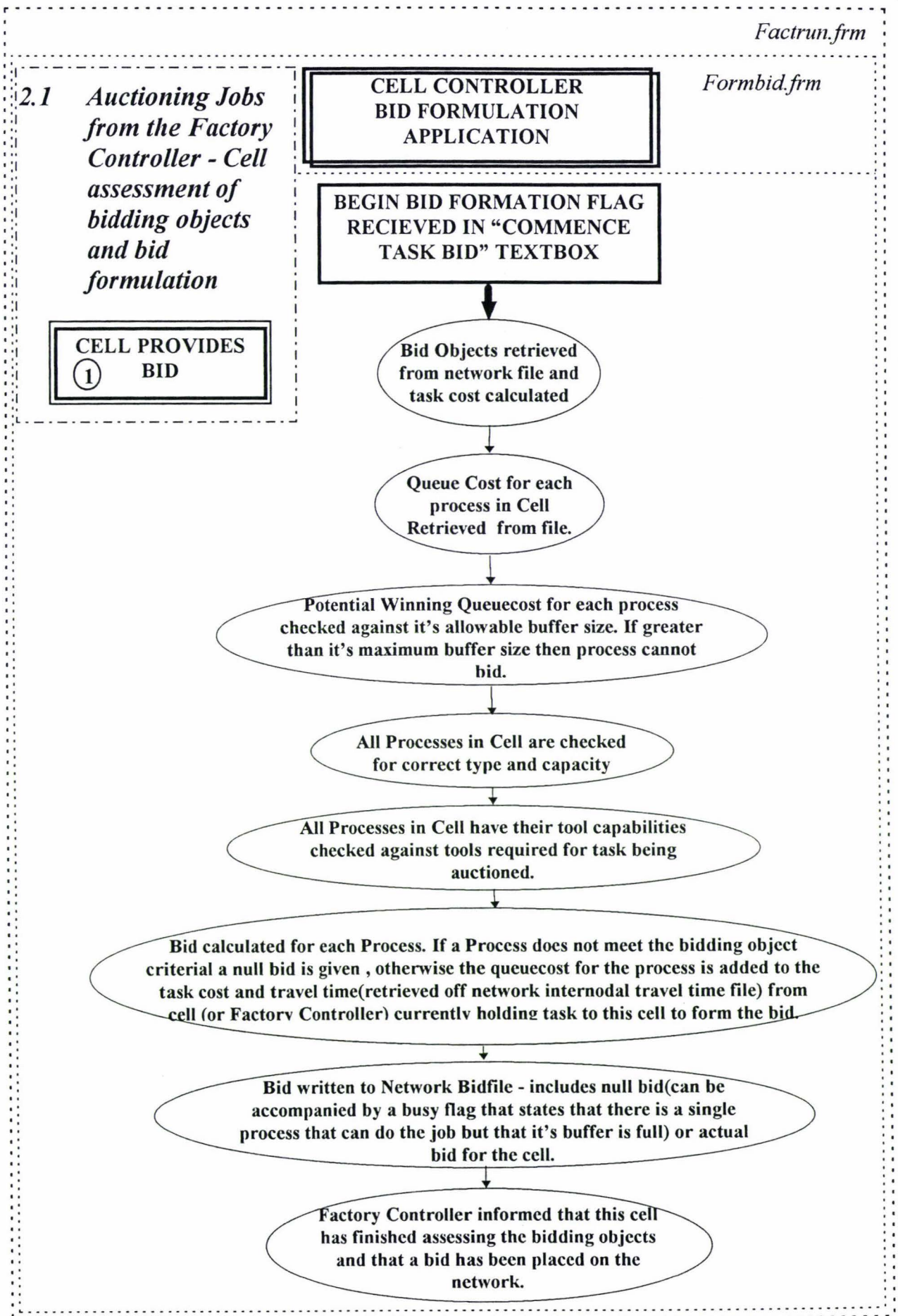
System_Form.frm

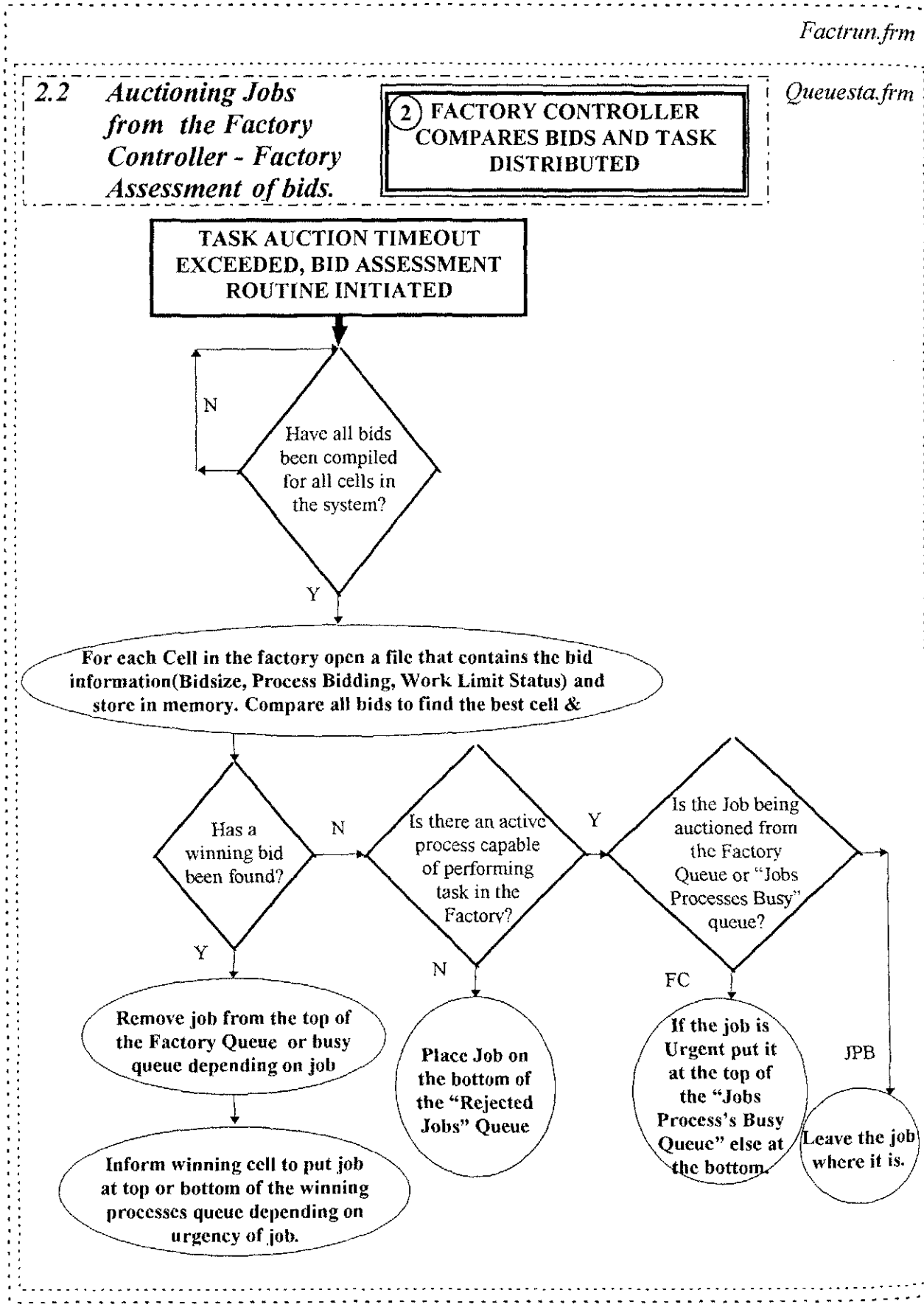


A.5 Task Scheduling Service Algorithm




Factrun.frm
Queuesta.frm
BID TOKEN RECIEVED
2.0 Auctioning Jobs from the Factory Controller
Has the attempt been made to auction off all available tasks on the "Jobs Process's Busy " List?
N
Y
Task to Schedule on "Job Processes
Task to Schedule on "Factory Queue Status List"
Secure Bid Token if not secured
Send Task to Schedule Bidding Objects to network File Server
Initiate Task Assessment Algorithm in Each cell via EDA
CELL PROVIDES (1) BID
FACTORY CONTROLLER COMPARES BIDS AND TASK DISTRIBUTED (2)
Was the task attempted to be auctioned off the "Jobs Process's Busy " List?
Y
N
Release Bid


Factrun.frm
2.1 Auctioning Jobs from the Factory Controller - Cell assessment of bidding objects and bid formulation
CELL PROVIDES ① BID
CELL CONTROLLER BID FORMULATION APPLICATION
Formbid.frm
BEGIN BID FORMATION FLAG RECIEVED IN "COMMENCE TASK BID" TEXTBOX
Bid Objects retrieved from network file and task cost calculated
Queue Cost for each process in Cell Retrieved from file.
Potential Winning Queuecost for each process checked against it's allowable buffer size. If greater than it's maximum buffer size then process cannot bid.
All Processes in Cell are checked for correct type and capacity
All Processes in Cell have their tool capabilities checked against tools required for task being auctioned.
Bid calculated for each Process. If a Process does not meet the bidding object criterial a null bid is given , otherwise the queuecost for the process is added to the task cost and travel time(retrieved off network internodal travel time file) from cell (or Factory Controller) currently holding task to this cell to form the bid.
Bid written to Network Bidfile - includes null bid(can be accompanied by a busy flag that states that there is a single process that can do the job but that it's buffer is full) or actual bid for the cell.
Factory Controller informed that this cell has finished assessing the bidding objects and that a bid has been placed on the network.

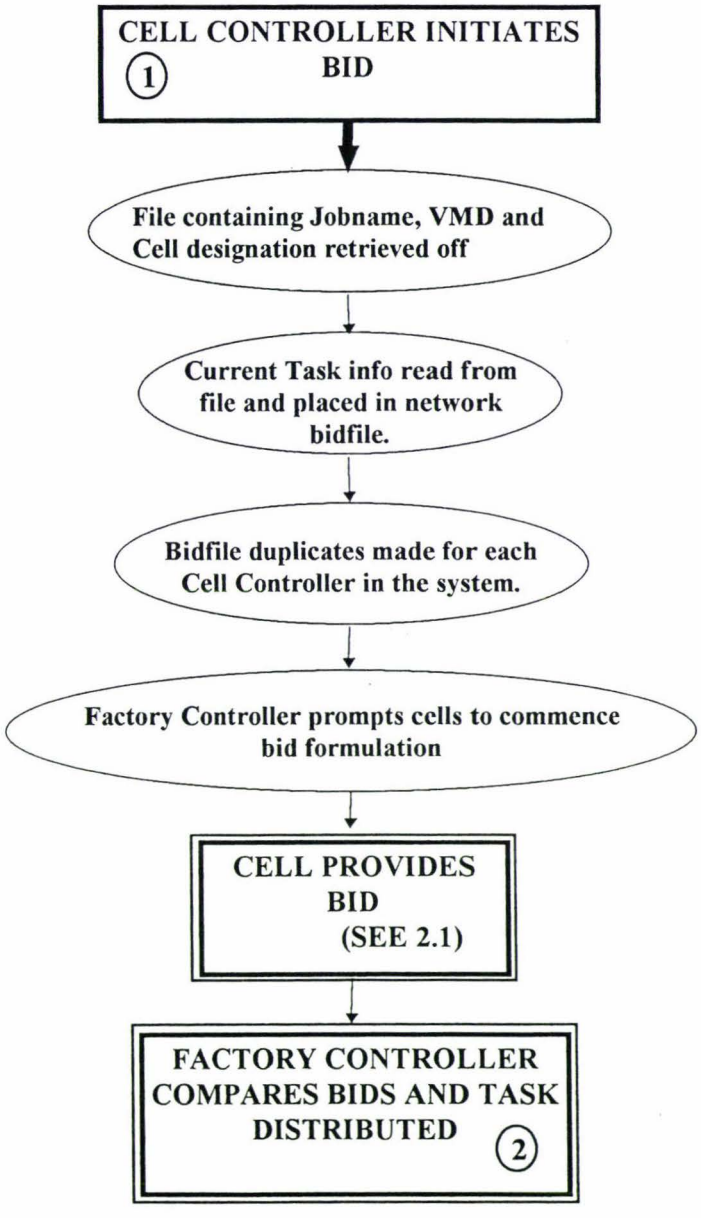

Factrun.frm
2.2 Auctioning Jobs from the Factory Controller - Factory Assessment of bids.
Queuesta.frm
2 FACTORY CONTROLLER COMPARES BIDS AND TASK DISTRIBUTED
TASK AUCTION TIMEOUT EXCEEDED, BID ASSESSMENT ROUTINE INITIATED
N
Have all bids been compiled for all cells in the system?
Y
For each Cell in the factory open a file that contains the bid information(Bidsize, Process Bidding, Work Limit Status) and store in memory. Compare all bids to find the best cell &
Has a winning bid been found?
N
Is there an active process capable of performing task in the Factory?
Y
Is the Job being auctioned from the Factory Queue or "Jobs Processes Busy" queue?
Y
N
FC
JPB
Remove job from the top of the Factory Queue or busy queue depending on job
Inform winning cell to put job at top or bottom of the winning processes queue depending on urgency of job.
Place Job on the bottom of the "Rejected Jobs" Queue
If the job is Urgent put it at the top of the "Jobs Process's Busy Queue" else at the bottom.
Leave the job where it is.

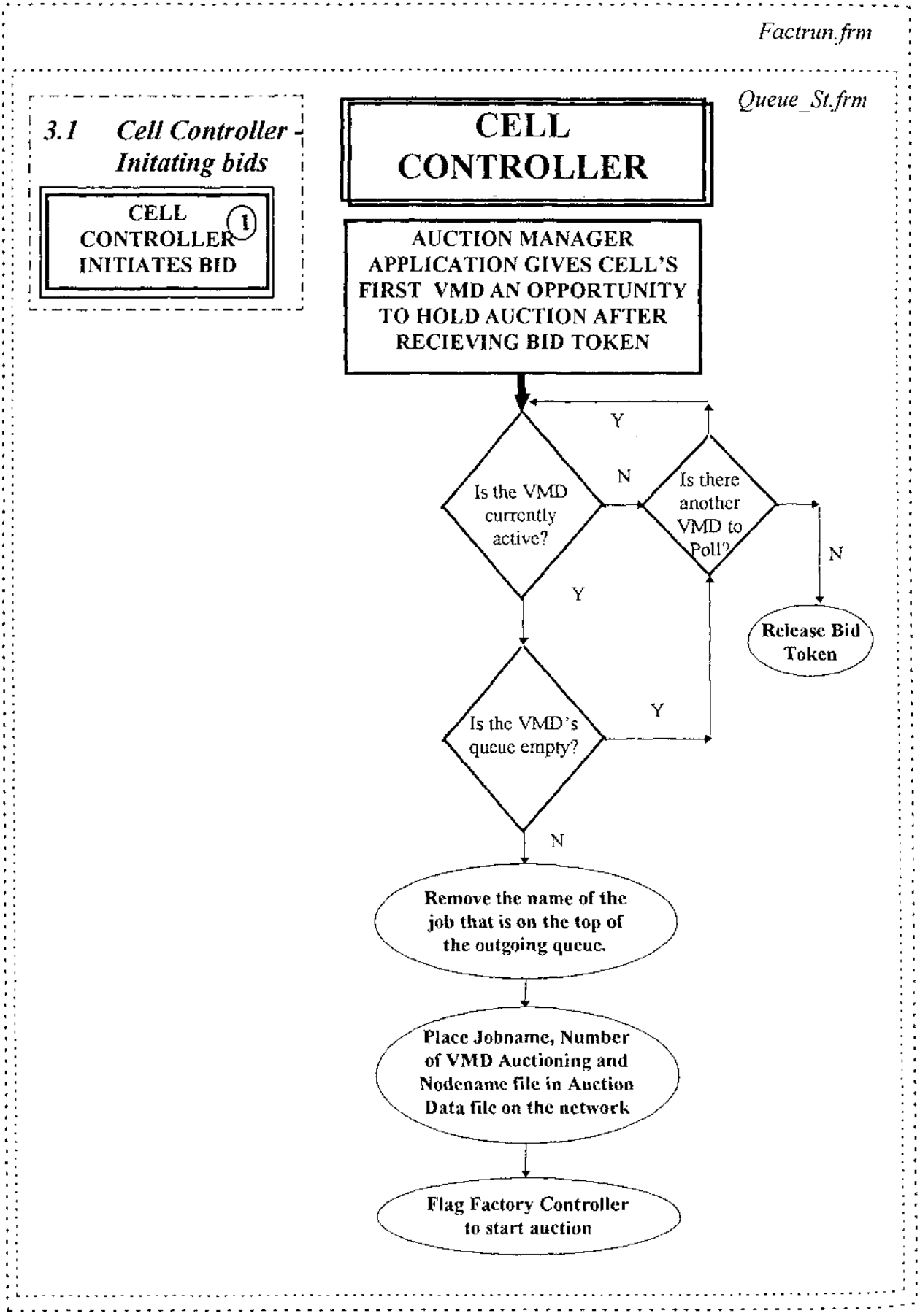
Factrun.frm

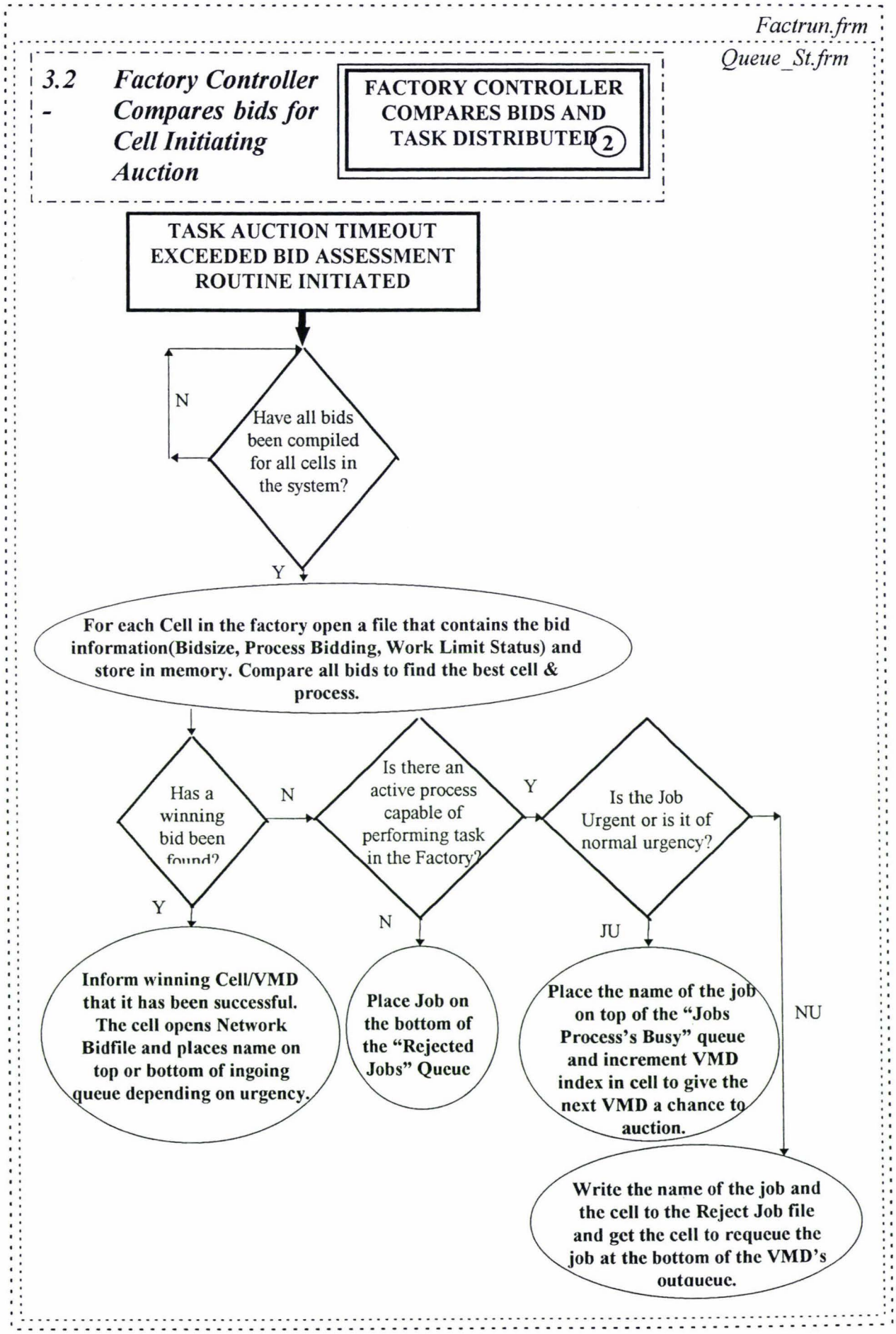
Queuesta.frm

3.0 Auctioning Jobs fromCell Controllers

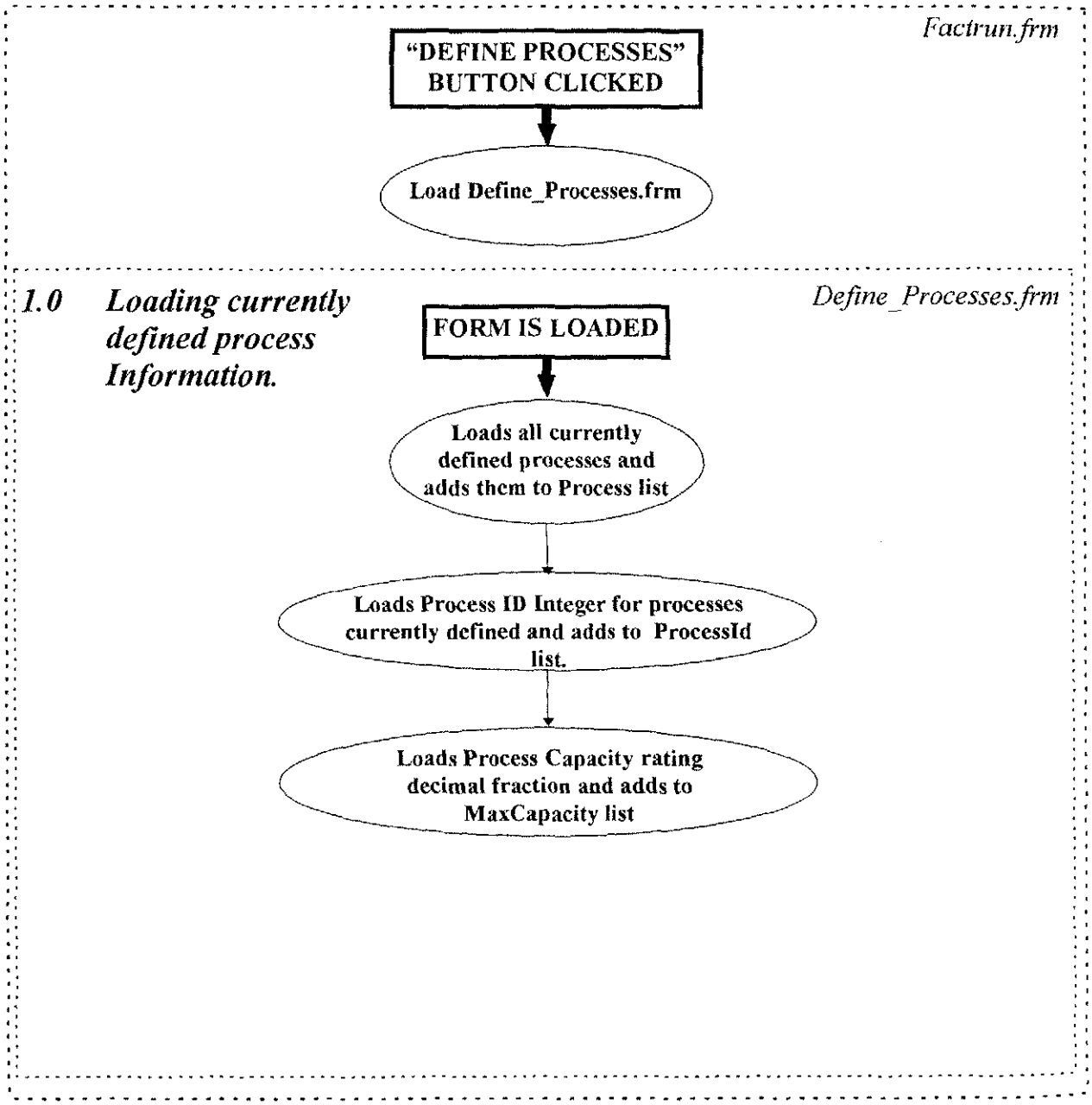
- *Factory Controller's Role.*




Factrun.frm
Queue_St.frm
3.1 Cell Controller Initating bids
CELL CONTROLLER INITIATES BID
1
CELL CONTROLLER
AUCTION MANAGER APPLICATION GIVES CELL'S FIRST VMD AN OPPORTUNITY TO HOLD AUCTION AFTER RECIEVING BID TOKEN
Is the VMD currently active?
Y
N
Is there another VMD to Poll?
N
Release Bid Token
Y
Is the VMD's queue empty?
Y
N
Remove the name of the job that is on the top of the outgoing queue.
Place Jobname, Number of VMD Auctioning and Nodename file in Auction Data file on the network
Flag Factory Controller to start auction


Factrun.frm
Queue_St.frm
3.2 - Factory Controller Compares bids for Cell Initiating Auction
FACTORY CONTROLLER COMPARES BIDS AND TASK DISTRIBUTED (2)
TASK AUCTION TIMEOUT EXCEEDED BID ASSESSMENT ROUTINE INITIATED
N
Have all bids been compiled for all cells in the system?
Y
For each Cell in the factory open a file that contains the bid information(Bidsize, Process Bidding, Work Limit Status) and store in memory. Compare all bids to find the best cell & process.
Has a winning bid been found?
N
Is there an active process capable of performing task in the Factory?
Y
Is the Job Urgent or is it of normal urgency?
Y
N
JU
NU
Inform winning Cell/VMD that it has been successful. The cell opens Network Bidfile and places name on top or bottom of ingoing queue depending on urgency.
Place Job on the bottom of the "Rejected Jobs" Queue
Place the name of the job on top of the "Jobs Process's Busy" queue and increment VMD index in cell to give the next VMD a chance to auction.
Write the name of the job and the cell to the Reject Job file and get the cell to requeue the job at the bottom of the VMD's outqueue.

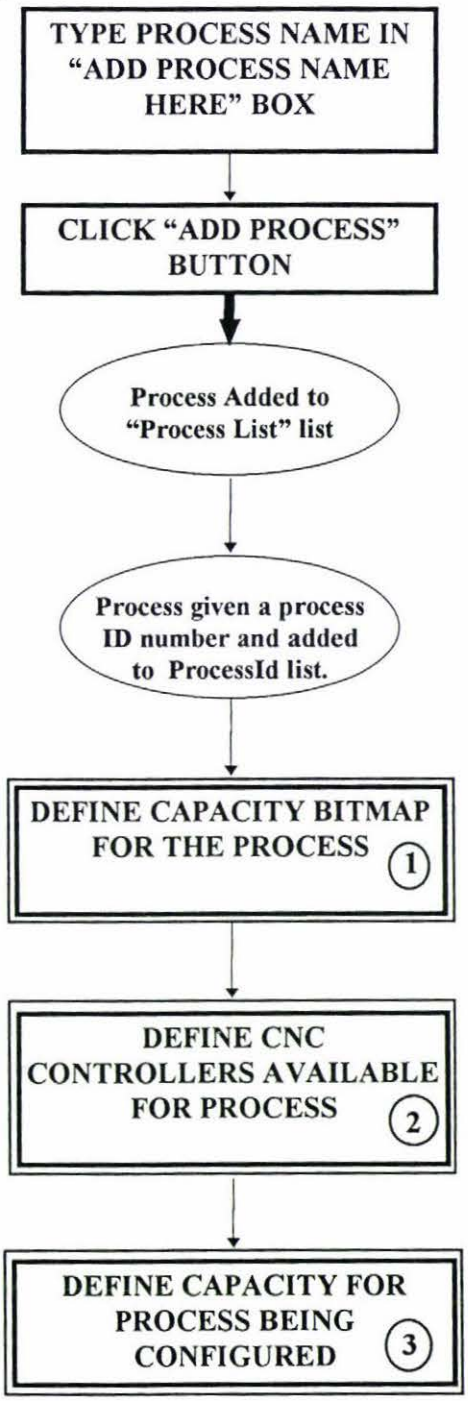
A.6 Resource Configuration Service - Process Configuration Algorithm

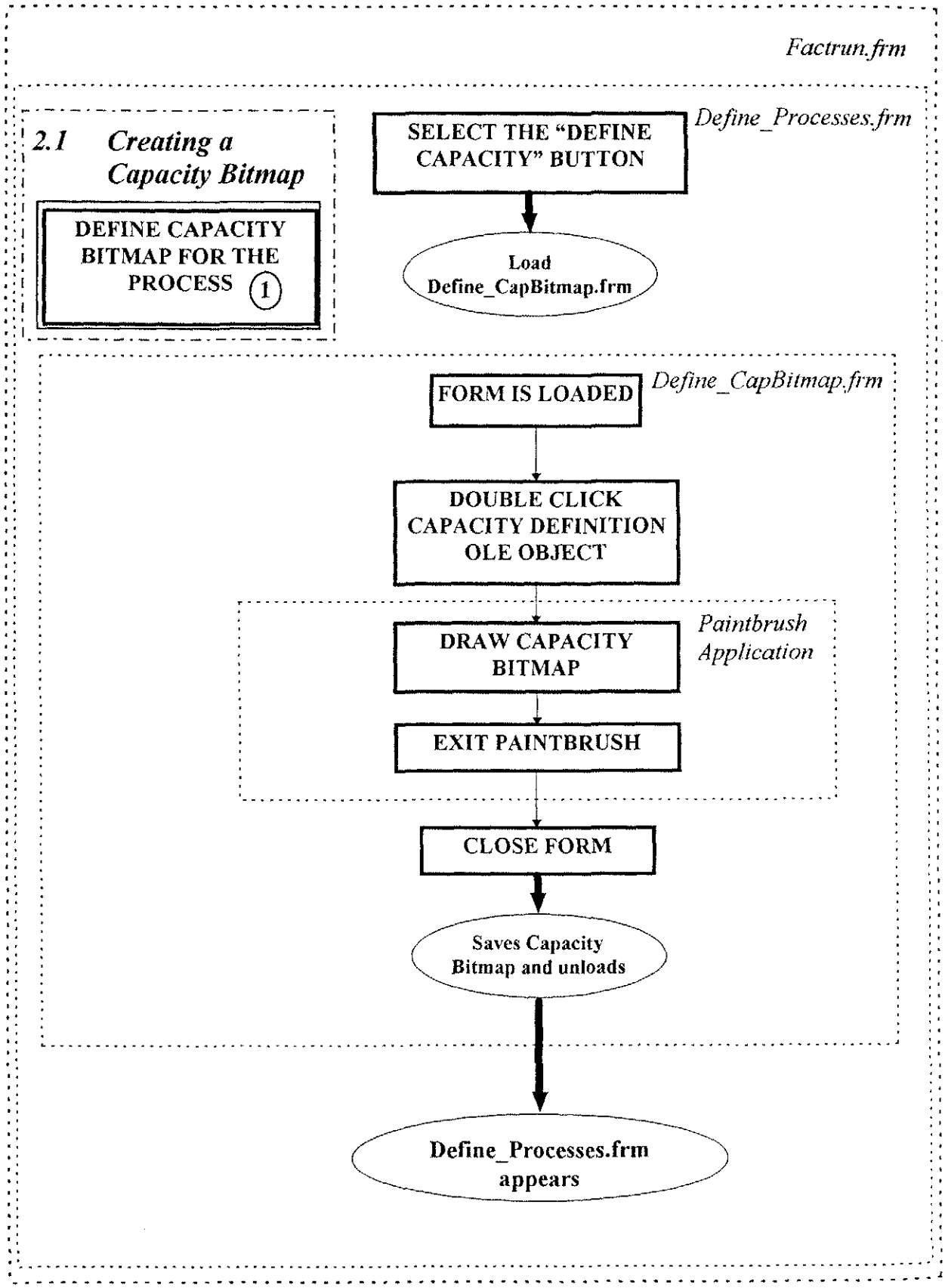


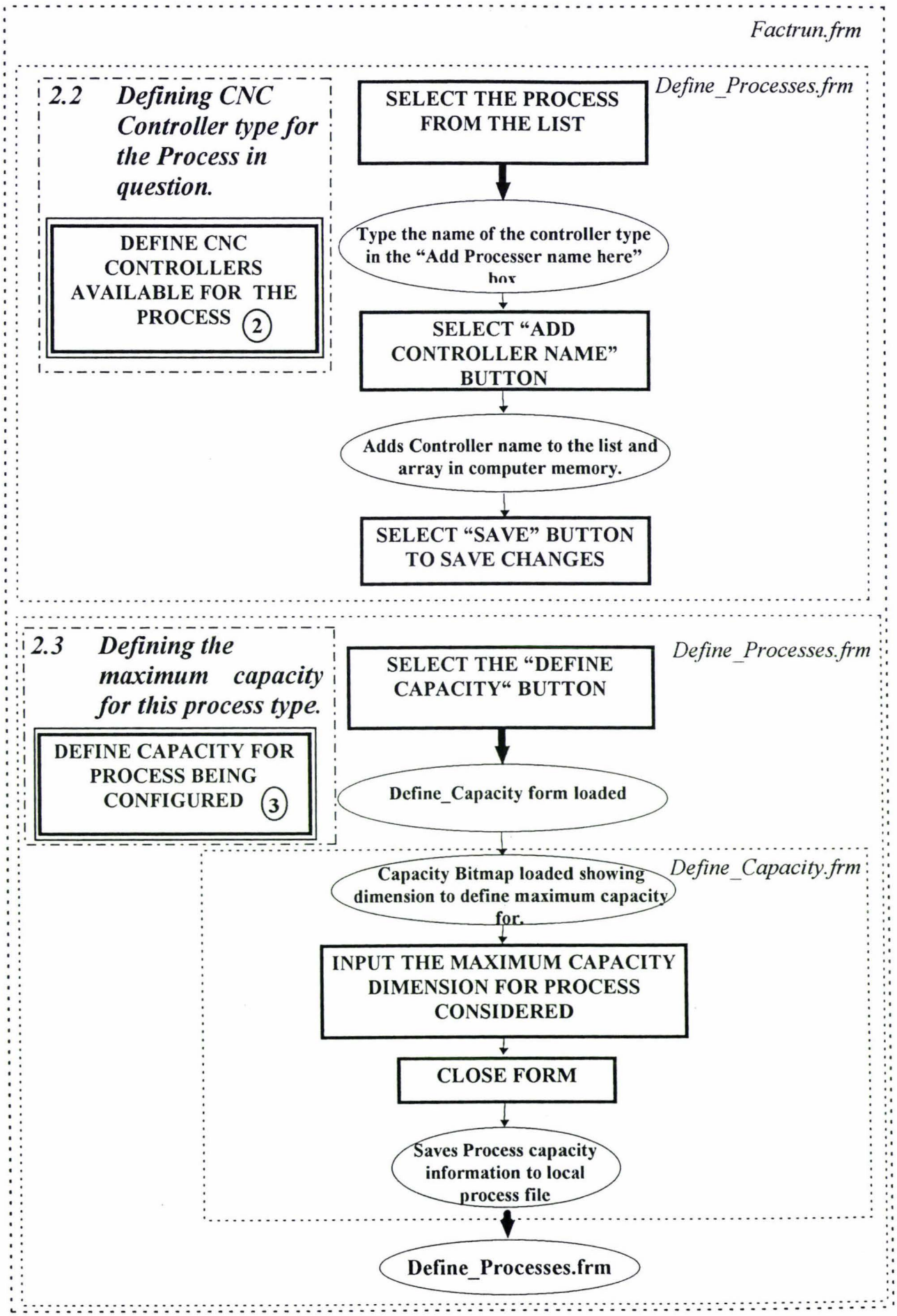
Factrun.frm

2.0 Defining a New Process Type

Define_Processes.frm

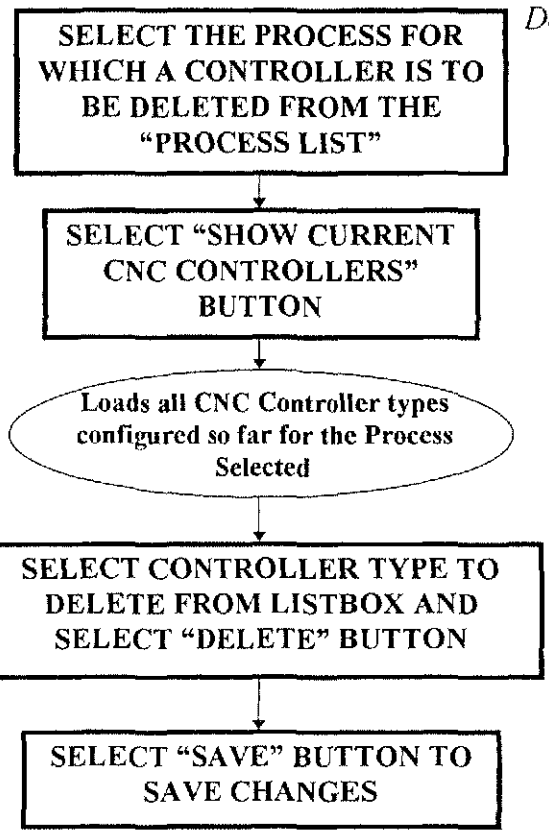



Factrun.frm
2.1 Creating a Capacity Bitmap
DEFINE CAPACITY BITMAP FOR THE PROCESS (1)
SELECT THE "DEFINE CAPACITY" BUTTON
Define_Processes.frm
Load Define_CapBitmap.frm
FORM IS LOADED
Define_CapBitmap.frm
DOUBLE CLICK CAPACITY DEFINITION OLE OBJECT
DRAW CAPACITY BITMAP
Paintbrush Application
EXIT PAINTBRUSH
CLOSE FORM
Saves Capacity Bitmap and unloads
Define_Processes.frm appears


Factrun.frm
2.2 Defining CNC Controller type for the Process in question.
DEFINE CNC CONTROLLERS AVAILABLE FOR THE PROCESS (2)
Define_Processes.frm
SELECT THE PROCESS FROM THE LIST
Type the name of the controller type in the "Add Processer name here" box
SELECT "ADD CONTROLLER NAME" BUTTON
Adds Controller name to the list and array in computer memory.
SELECT "SAVE" BUTTON TO SAVE CHANGES
2.3 Defining the maximum capacity for this process type.
DEFINE CAPACITY FOR PROCESS BEING CONFIGURED (3)
Define_Processes.frm
SELECT THE "DEFINE CAPACITY" BUTTON
Define_Capacity form loaded
Define_Capacity.frm
Capacity Bitmap loaded showing dimension to define maximum capacity for.
INPUT THE MAXIMUM CAPACITY DIMENSION FOR PROCESS CONSIDERED
CLOSE FORM
Saves Process capacity information to local process file
Define_Processes.frm

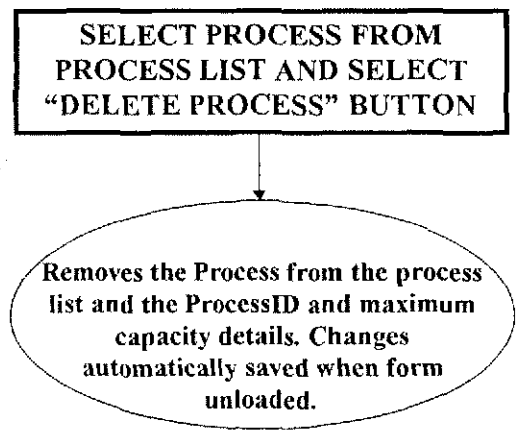
3.0 *Removing a CNC Controller type for a Process.*

Define_Processes.frm

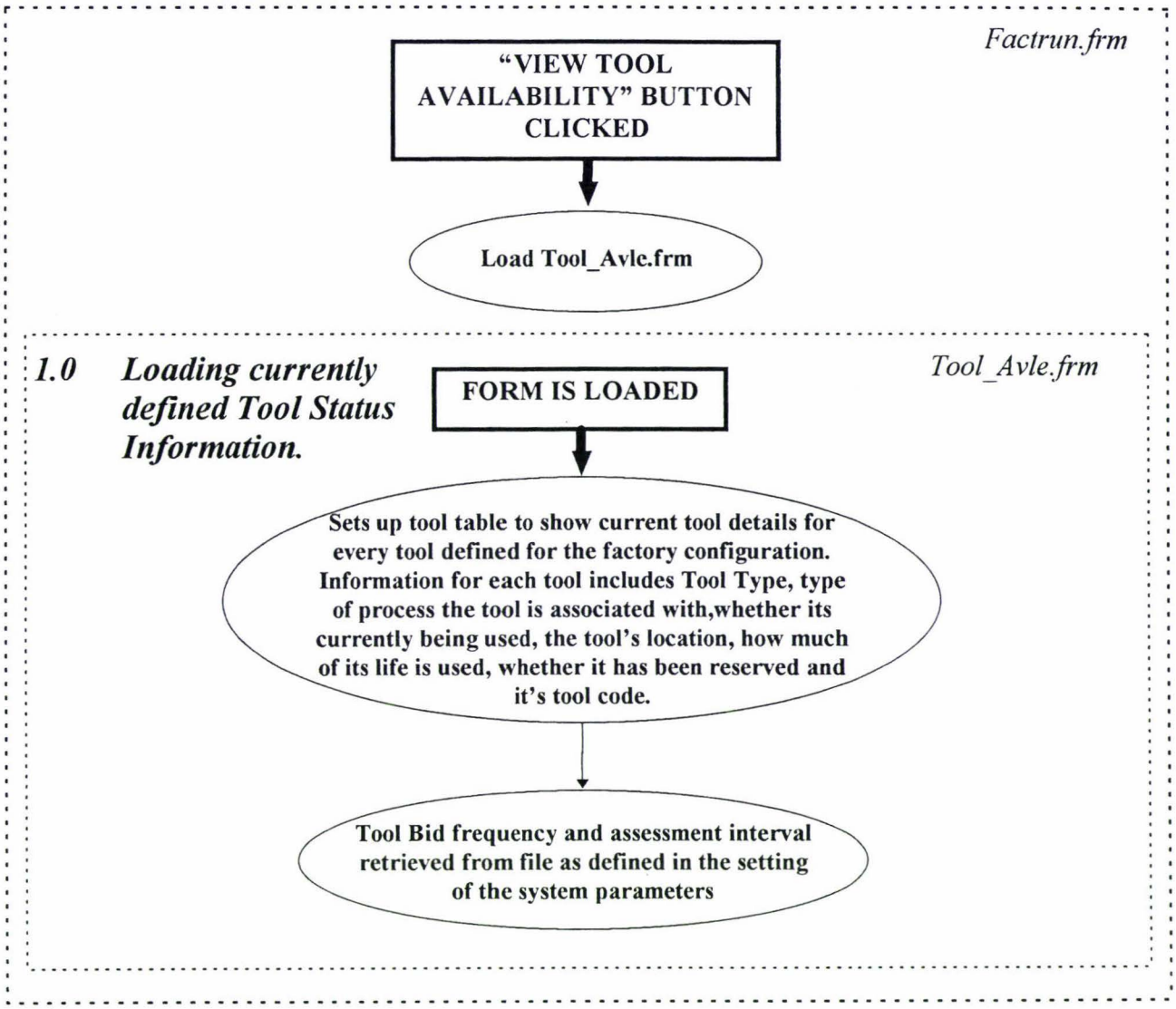


4.0 *Deleting a Process*

Define_Processes.frm



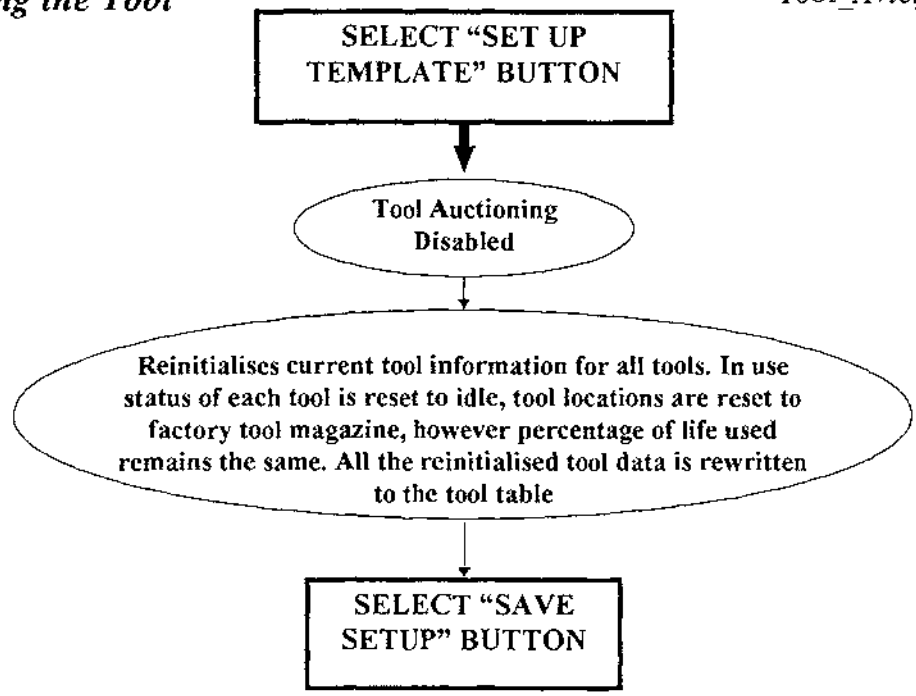
A.7 Resource Scheduling Service Algorithm



Factrun.frm

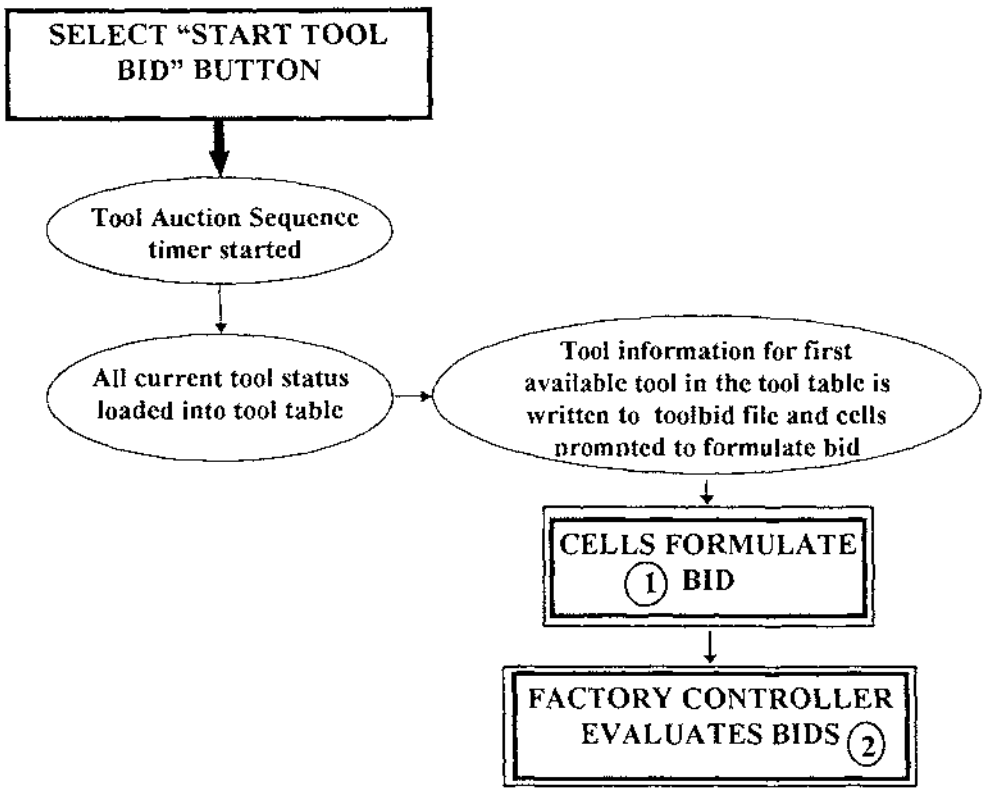
2.0 Resetting the Tool Details

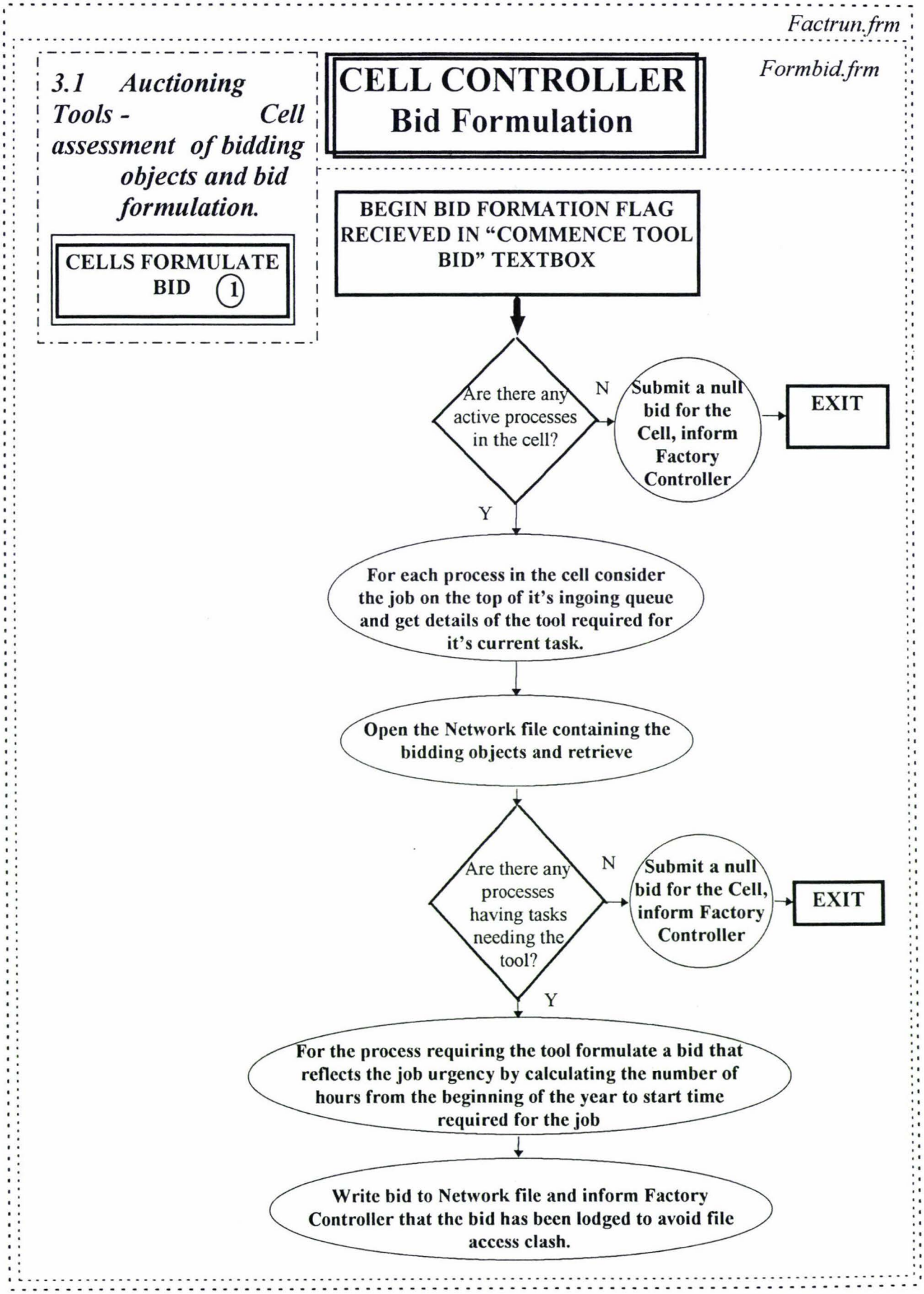
Tool_Avle.frm



3.0 Auctioning Tools

Tool_Avle.frm



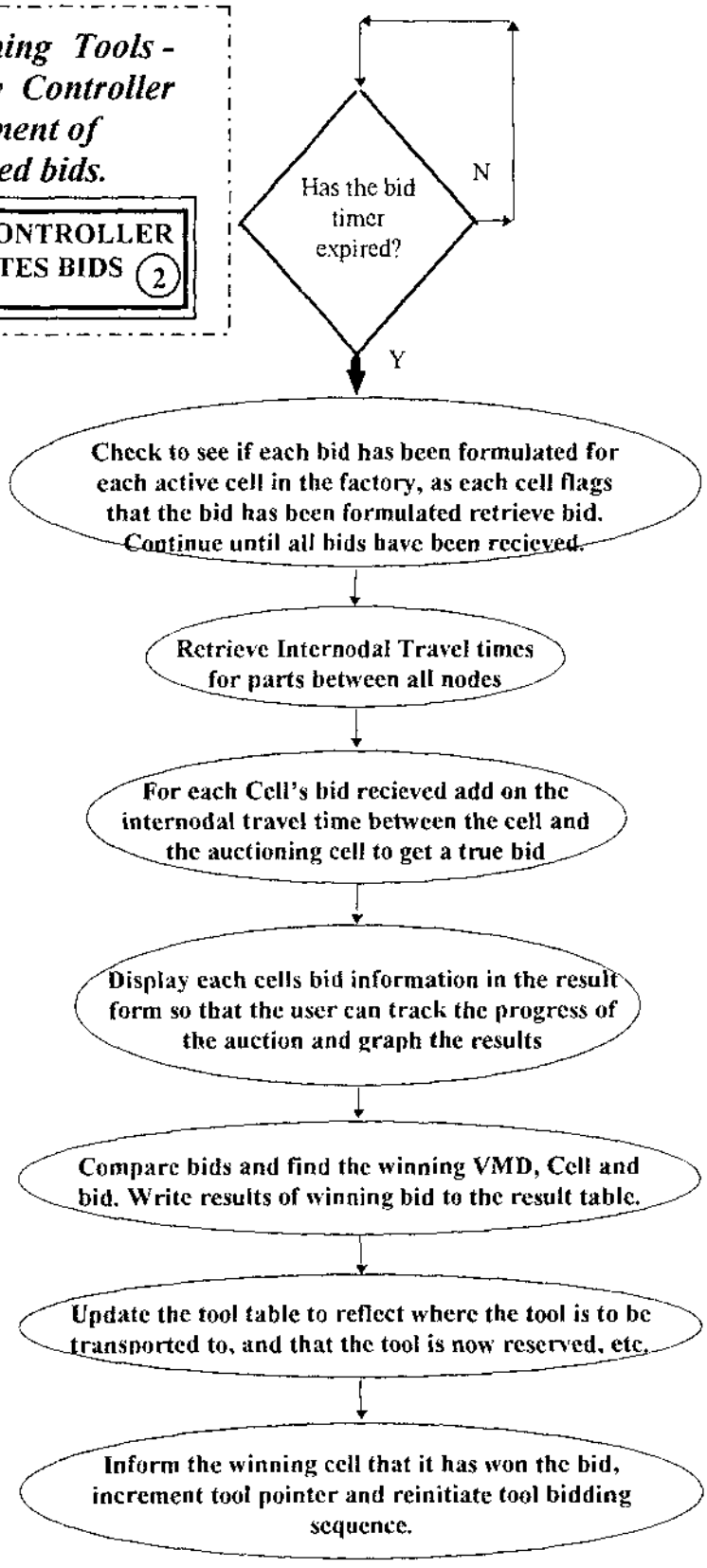

Factrun.frm
Formbid.frm
3.1 Auctioning Tools - Cell assessment of bidding objects and bid formulation.
CELLS FORMULATE BID (1)
CELL CONTROLLER
Bid Formulation
BEGIN BID FORMATION FLAG RECIEVED IN "COMMENCE TOOL BID" TEXTBOX
Are there any active processes in the cell?
N
Submit a null bid for the Cell, inform Factory Controller
EXIT
Y
For each process in the cell consider the job on the top of it's ingoing queue and get details of the tool required for it's current task.
Open the Network file containing the bidding objects and retrieve
Are there any processes having tasks needing the tool?
N
Submit a null bid for the Cell, inform Factory Controller
EXIT
Y
For the process requiring the tool formulate a bid that reflects the job urgency by calculating the number of hours from the beginning of the year to start time required for the job
Write bid to Network file and inform Factory Controller that the bid has been lodged to avoid file access clash.

Factrun.frm

Formbid.frm

3.2 Auctioning Tools - Factory Controller Assessment of submitted bids.

FACTORY CONTROLLER EVALUATES BIDS (2)



Appendix B

FIXDMACS Factory Database Blocks

This Appendix provides the exact specifications of all FIXDMACS database blocks used in the implementation of the Factory Front End addresses.

Fixdmacs Factory Database Blocks

	Tag Name	Type	Description
1	AUCTIONSTD	AO	CACA-RB A Node in Factory is auctioning
2	BIDACKNOWL	AO	C/FACA-RB Bid Acknowledge Block
3	BIDCYCTM	AO	C/FACA-RB Cyc tm of tsk curently bg auct
4	BIDENABLE	AO	Factory Controller bidenable block
5	BIDFORCE	AO	C/FACA-RB Bidding object force status
6	BIDFRMSTRT	AO	Tells the Cell Controll to formulate bid
7	BIDLOADUN	AO	C/FACA-RB Loaduntm for curently auct tsk
8	BIDMACHTYP	AO	C/FACA-RB Mach typ reqd for current task
9	BIDNOPARTS	AO	C/FACA-RB Number of parts to be made
10	BIDOBJSENT	AO	Tells Q programs that bid obj's are sent
11	BIDOPERATE	AO	Factory node currently bidding
12	BIDSUPTM	AO	C/FACA-RB Mach Suptime for curr auct tsk
13	BIDTOKEN	AO	Factory node opportunity to hold auction
14	BIDTOOL1RQ	AO	C/FACA-RB Tool Rq for 1st op on tsk auct
15	BIDTOOL2RQ	AO	C/FACA-RB Tool Rq for 2nd op on tsk auct
16	BIDTOOL3RQ	AO	C/FACA-RB Tool Rq for 3rd op on tsk auct
17	BIDTOOL4RQ	AO	C/FACA-RB Tool Rq for 4th op on tsk auct
18	C01BIDFIN	AO	Cell 1 Bid is finished being formed
19	C01TOOLFIN	AO	Cell 1 Has finished Tool Bidding
20	C01V1BDCST	AO	BCA-RG CC1 VMD1's Bid as lodged
21	C01V1TOOL	AO	Cell1 VMD1has finished using the tool
22	C01V2BDCST	AO	BCA-RG CC1 VMD2's Bid as lodged
23	C01V2TOOL	AO	Cell1 VMD2 has finished using the tool
24	C01V3BDCST	AO	BCA-RG CC1 VMD3's Bid as lodged
25	C01V4BDCST	AO	BCA-RG CC1 VMD4's Bid as lodged
26	C01V4TOOL	AO	Cell1 VMD4 has finished using the tool
27	C01VMD1WIN	AO	BCA-RG Cell Controller1 V1 has won bid
28	C01VMD2WIN	AO	BCA-RG Cell Controller1 V2 has won bid
29	C01VMD3WIN	AO	BCA-RG Cell Controller1 V3 has won bid
30	C01VMD4WIN	AO	BCA-RG Cell Controller1 V4 has won bid
31	C02BIDFIN	AO	Cell 2 Bid is finished being formed
32	C02TOOLFIN	AO	Cell 2 Has finished Tool Bidding
33	C02V1BDCST	AO	BCA-RG CC2 VMD1's Bid as lodged
34	C02V1TOOL	AO	Cell2 VMD1has finished using the tool
35	C02V2BDCST	AO	BCA-RG CC2 VMD2's Bid as lodged

Tag Name	Type	Description
36 C02V2TOOL	AO	Cell2 VMD2 has finished using the tool
37 C02V3BDCST	AO	BCA-RG CC2 VMD3's Bid as lodged
38 C02V4BDCST	AO	BCA-RG CC2 VMD4's Bid as lodged
39 C02V4TOOL	AO	Cell2 VMD4 has finished using the tool
40 C02VMD1WIN	AO	BCA-RG Cell Controller2 V1has won bid
41 C02VMD2WIN	AO	BCA-RG Cell Controller2 V2 has won bid
42 C02VMD3WIN	AO	BCA-RG Cell Controller2 V3 has won bid
43 C02VMD4WIN	AO	BCA-RG Cell Controller2 V4 has won bid
44 C03BIDFIN	AO	Cell 3 Bid is finished being formed
45 C03TOOLFIN	AO	Cell 3 Has finished Tool Bidding
46 C03V1BDCST	AO	BCA-RG CC3 VMD1's Bid as lodged
47 C03V1TOOL	AO	Cell3 VMD1has finished using the tool
48 C03V2BDCST	AO	BCA-RG CC3 VMD2's Bid as lodged
49 C03V2TOOL	AO	Cell3 VMD2 has finished using the tool
50 C03V3BDCST	AO	BCA-RG CC3 VMD3's Bid as lodged
51 C03V4BDCST	AO	BCA-RG CC3 VMD4's Bid as lodged
52 C03V4TOOL	AO	Cell3 VMD4 has finished using the tool
53 C03VMD1WIN	AO	BCA-RG Cell Controller3 V1 has won bid
54 C03VMD2WIN	AO	BCA-RG Cell Controller3 V2 has won bid
55 C03VMD3WIN	AO	BCA-RG Cell Controller3 V3 has won bid
56 C03VMD4WIN	AO	BCA-RG Cell Controller3 V4 has won bid
57 C04V1BDCST	AO	BCA-RG CC4 VMD1's Bid as lodged
58 C04V2BDCST	AO	BCA-RG CC4 VMD2's Bid as lodged
59 C04V3BDCST	AO	BCA-RG CC4 VMD3's Bid as lodged
60 C04V4BDCST	AO	BCA-RG CC4 VMD4's Bid as lodged
61 C04VMD1WIN	AO	BCA-RG Cell Controller4 V1 has won bid
62 C04VMD2WIN	AO	BCA-RG Cell Controller4 V2 has won bid
63 C04VMD3WIN	AO	BCA-RG Cell Controller4 V3 has won bid
64 C04VMD4WIN	AO	BCA-RG Cell Controller4 V4 has won bid
65 C05V1BDCST	AO	BCA-RG CC5 VMD1's Bid as lodged
66 C05V2BDCST	AO	BCA-RG CC5 VMD2's Bid as lodged
67 C05V3BDCST	AO	BCA-RG CC5 VMD3's Bid as lodged
68 C05V4BDCST	AO	BCA-RG CC5 VMD4's Bid as lodged
69 C05VMD1WIN	AO	BCA-RG Cell Controller5 V1 has won bid
70 C05VMD2WIN	AO	BCA-RG Cell Controller5 V2 has won bid
71 C05VMD3WIN	AO	BCA-RG Cell Controller5 V3 has won bid
72 C05VMD4WIN	AO	BCA-RG Cell Controller5 V4 has won bid
73 C06V1BDCST	AO	BCA-RG CC6 VMD1's Bid as lodged
74 C06V2BDCST	AO	BCA-RG CC6 VMD2's Bid as lodged
75 C06V3BDCST	AO	BCA-RG CC6 VMD3's Bid as lodged

Tag Name	Type	Description
76 C06V4BDCST	AO	BCA-RG CC6 VMD4's Bid as lodged
77 C06VMD1WIN	AO	BCA-RG Cell Controller6 V1 has won bid
78 C06VMD2WIN	AO	BCA-RG Cell Controller6 V2 has won bid
79 C06VMD3WIN	AO	BCA-RG Cell Controller6 V3 has won bid
80 C06VMD4WIN	AO	BCA-RG Cell Controller6 V4 has won bid
81 C07V1BDCST	AO	BCA-RG CC7 VMD1's Bid as lodged
82 C07V2BDCST	AO	BCA-RG CC7 VMD2's Bid as lodged
83 C07V3BDCST	AO	BCA-RG CC7 VMD3's Bid as lodged
84 C07V4BDCST	AO	BCA-RG CC7 VMD4's Bid as lodged
85 C07VMD1WIN	AO	BCA-RG Cell Controller7 V1 has won bid
86 C07VMD2WIN	AO	BCA-RG Cell Controller7 V2 has won bid
87 C07VMD3WIN	AO	BCA-RG Cell Controller7 V3 has won bid
88 C07VMD4WIN	AO	BCA-RG Cell Controller7 V4 has won bid
89 C08V1BDCST	AO	BCA-RG CC8 VMD1's Bid as lodged
90 C08V2BDCST	AO	BCA-RG CC8 VMD2's Bid as lodged
91 C08V3BDCST	AO	BCA-RG CC8 VMD3's Bid as lodged
92 C08V4BDCST	AO	BCA-RG CC8 VMD4's Bid as lodged
93 C08VMD1WIN	AO	BCA-RG Cell Controller8 V1 has won bid
94 C08VMD2WIN	AO	BCA-RG Cell Controller8 V2 has won bid
95 C08VMD3WIN	AO	BCA-RG Cell Controller8 V3 has won bid
96 C08VMD4WIN	AO	BCA-RG Cell Controller8 V4 has won bid
97 C09V1BDCST	AO	BCA-RG CC9 VMD1's Bid as lodged
98 C09V2BDCST	AO	BCA-RG CC9 VMD2's Bid as lodged
99 C09V3BDCST	AO	BCA-RG CC9 VMD3's Bid as lodged
100 C09V4BDCST	AO	BCA-RG CC9 VMD4's Bid as lodged
101 C09VMD1WIN	AO	BCA-RG Cell Controller9 V1 has won bid
102 C09VMD2WIN	AO	BCA-RG Cell Controller9 V2 has won bid
103 C09VMD3WIN	AO	BCA-RG Cell Controller9 V3 has won bid
104 C09VMD4WIN	AO	BCA-RG Cell Controller9 V4 has won bid
105 C10V1BDCST	AO	BCA-RG CC10 VMD1's Bid as lodged
106 C10V2BDCST	AO	BCA-RG CC10 VMD2's Bid as lodged
107 C10V3BDCST	AO	BCA-RG CC10 VMD3's Bid as lodged
108 C10V4BDCST	AO	BCA-RG CC10 VMD4's Bid as lodged
109 C10VMD1WIN	AO	BCA-RG Cell Controller10 V1 has won bid
110 C10VMD2WIN	AO	BCA-RG Cell Controller10 V2 has won bid
111 C10VMD3WIN	AO	BCA-RG Cell Controller10 V3 has won bid
112 C10VMD4WIN	AO	BCA-RG Cell Controller10 V4 has won bid
113 C11V1BDCST	AO	BCA-RG CC11 VMD1's Bid as lodged
114 C11V2BDCST	AO	BCA-RG CC11 VMD2's Bid as lodged

Tag Name	Type	Description
115 C11V3BDCST	AO	BCA-RG CC11 VMD3's Bid as lodged
116 C11V4BDCST	AO	BCA-RG CC11 VMD4's Bid as lodged
117 C11VMD1WIN	AO	BCA-RG Cell Controller11 V1 has won bid
118 C11VMD2WIN	AO	BCA-RG Cell Controller11 V2 has won bid
119 C11VMD3WIN	AO	BCA-RG Cell Controller11 V3 has won bid
120 C11VMD4WIN	AO	BCA-RG Cell Controller11 V4 has won bid
121 C12V1BDCST	AO	BCA-RG CC12 VMD1's Bid as lodged
122 C12V2BDCST	AO	BCA-RG CC12 VMD2's Bid as lodged
123 C12V3BDCST	AO	BCA-RG CC12 VMD3's Bid as lodged
124 C12V4BDCST	AO	BCA-RG CC12 VMD4's Bid as lodged
125 C12VMD1WIN	AO	BCA-RG Cell Controller12 V1 has won bid
126 C12VMD2WIN	AO	BCA-RG Cell Controller12 V2 has won bid
127 C12VMD3WIN	AO	BCA-RG Cell Controller12 V3 has won bid
128 C12VMD4WIN	AO	BCA-RG Cell Controller12 V4 has won bid
129 CELLISAUCT	AO	Tells Fact Cont that a cell is auction
130 COMPAREBID	AO	BCA-RG Starts bid comparison algorithm
131 FBIDRQUEST	AO	C/FACA-RB Set to 1 to q pg t retrv bdobj
132 UPDATEMTBC	AO	Updates Mean Time Between Completion
133 V1TOOLAVLE	AO	Are tools available for VMD 1?
134 CONTBIDCOM	PG	BCA_PG Controls the bid compares process
135 DISTBIDOBJ	PG	F/CACA-PG Strt Auction & Dist bidobjects
136 BIDINFO	TX	----
137 BIDJOB	TX	----

Appendix C

FIXDMACS Cell Controller Database Blocks

This Appendix provides the exact specifications of all FIXDMACS database blocks used in the implementation of the Cell Front End addresses.

Fixdmacs Cell Database Blocks

Tag Name	Type	Description
1 BIDMACH1	AI	BCA-RG Bidmach for vmd1
2 BIDMACH2	AI	BCA-RG Bidmach for vmd2
3 BIDMACH3	AI	BCA-RG Bidmach for vmd3
4 BIDMACH4	AI	BCA-RG Bidmach for vmd4
5 BIDMACHTYP	AI	BCA-RG Type of machine required for task
6 AUCTIONSTD	AO	An auction is occurring on this Cell Con
7 BDTOOLSTRT	AO	Start Bid formation for the tool auction
8 BIDENABLE	AO	Cell Controller bidenable block
9 BIDFRMSTRT	AO	BCA-RG Set = 1 Starts BCA progrm in Cell
10 BIDOPERATE	AO	ACA-RB Current node currently bidding
11 BIDTOKEN	AO	ACA-RB Curr node opportunity to auction
12 BIDWIN	AO	Contains the number of the winning VMD
13 CELLACTIVE	AO	This waits for a msg to start processes
14 ISV1CONFIG	AO	VMD-RB Is vmd1 setup and ready to run?
15 ISV2CONFIG	AO	VMD-RB Is vmd2 setup and ready to run?
16 ISV3CONFIG	AO	VMD-RB Is vmd3 setup and ready to run?
17 ISV4CONFIG	AO	VMD-RB Is vmd4 setup and ready to run?
18 REQUEUEJOB	AO	This reques the job thats proces is busy
19 STARTTIME	AO	Tells queue app the simulate start time
20 V1BAUDRATE	AO	The baud rate required for NC Controller
21 V1CNCSTART	AO	Starts download of NC progrm - Process1
22 V1CURRPART	AO	vmd1 current part number*
23 V1DOWNCOMP	AO	CNC Download for VMD1 is completed
24 V1DOWNCONT	AO	VMD1 Download Control block
25 V1DWNCOMPL	AO	The download from the q app is complete
26 V1INQUSIZE	AO	Number of tasks in vmd1's ingoing queue
27 V1MACHTYP	AO	the type of process and size in vmd1

Tag Name	Type	Description
28 V1NOTASKS	AO	ACA-RB Number of tasks in vmd1 outqueue
29 V1QCONTROL	AO	VMD1's queue Control block
30 V1QUEUCOST	AO	BCA-RB VMD1 Queue Cost
31 V1RNC01CCN	AO	VMD1 - RB Current Cycle Number of task
32 V1RNC01CTM	AO	VMD1 - RB Cycle Time For Current Task
33 V1RNC01LUN	AO	VMD1 - RB Load Unloadtme for curent task
34 V1RNC01NOP	AO	Number of parts to be made
35 V1RNC01SUP	AO	VMD1 - RB Setuptime for task to be done
36 V1RNC02CCN	AO	VMD1_C02 Current Cycle number
37 V1RNC02CTM	AO	VMD1-RB Cycle time for Current task
38 V1RNC02LUN	AO	VMD1_RB Load Unloadttime for curent task
39 V1RNC02NOP	AO	VMD1-RB Number of Parts to be made
40 V1RNC02SUP	AO	VMD1-RB Setuptime for task to be done
41 V1RNC03CCN	AO	VMD1 C03 Current cycle number
42 V1RNC03CTM	AO	VMD1 C03 Cycle time
43 V1RNC03LUN	AO	VMD1 C03 Load UnloadTime
44 V1RNC03NOP	AO	VMD1 C03 Number of parts to be made
45 V1RNC03SUP	AO	VMD1 C03 Set up time
46 V1RNC04NOP	AO	VMD1 C03 Number of parts to be made
47 V1SCANSTAT	AO	Indicates vmd1 scan status
48 V1STOPDOWN	AO	Stops download of NC file to Process 1
49 V1STPSTART	AO	Stop/Start data bits for Vmd1 controller
50 V1TASKTIME	AO	The time required to do the task in vmd1
51 V1TOOLAVLE	AO	BCA-RB V1 tools available to do req tsk
52 V1TOOLRESD	AO	VMD1 - tools reserved for current task?
53 V2BAUDRATE	AO	The baud rate required for NC Controller
54 V2CNCSTART	AO	Starts download of NC progrm - Process2
55 V2CURRPART	AO	vmd2 current part number*
56 V2DOWNCOMP	AO	CNC Download for VMD2 is completed
57 V2DWNCOMPL	AO	The download from the q app is complete
58 V2INQUSIZE	AO	Number of tasks in vmd2's ingoing queue
59 V2MACHTYP	AO	VMD2_RG Block What process is in VMD2
60 V2NOTASKS	AO	ACA-RB Number of tasks in vmd2 outqueue

Tag Name	Type	Description
61 V2QCONTROL	AO	ACA-CB Queue Control block for VMD2
62 V2QUEUCOST	AO	BCA-RB VMD2 Queue Cost
63 V2RNC01CCN	AO	VMD2 C01 CURR CYCLE NUMBER
64 V2RNC01CTM	AO	VMD2 C01 Cycle Time of Curr Part
65 V2RNC01LUN	AO	VMD2 C01 Load Unloadtime
66 V2RNC01NOP	AO	VMD2 C01 Number of parts to be made
67 V2RNC01SUP	AO	VMD2 C01 Seetup Time of Curr Task
68 V2RNC02CCN	AO	VMD2 C02 Current Cycle Number
69 V2RNC02CTM	AO	VMD2 C02 Cycle Time
70 V2RNC02LUN	AO	VMD2 C02 Load UnloadTime of Curr task
71 V2RNC02NOP	AO	VMD2 C02 Number of parts to be made
72 V2RNC02SUP	AO	VMD2 C02 Setup time
73 V2RNC03CCN	AO	VMD2 C03 Current Cycle Number
74 V2RNC03CTM	AO	VMD2 C03 Cycle Time
75 V2RNC03LUN	AO	VMD2 C03 Load Unload Time
76 V2RNC03NOP	AO	VMD2 C03 Number of parts to be made
77 V2RNC03SUP	AO	VMD2 C03 Setup Time
78 V2SCANSTAT	AO	Indicates vmd2 scan status
79 V2STOPDOWN	AO	Stops download of NC file to Process 2
80 V2STPSTART	AO	Stop/Start data bits for Vmd2 controller
81 V2TASKTIME	AO	The time required to do the task in vmd2
82 V2TOOLAVLE	AO	BCA-RB V2 Tools available to do reqd tsk
83 V2TOOLRESD	AO	VMD2 - tools reserved for current task?
84 V3BAUDRATE	AO	The baud rate required for NC Controller
85 V3CNCSTART	AO	Starts download of NC progrm - Process3
86 V3CURRPART	AO	vmd3 current part number*
87 V3DOWNCOMP	AO	CNC Download for VMD3 is completed
88 V3DWNCOMP	AO	The download from the q app is complete
89 V3INQUSIZE	AO	Number of tasks in VMD3's ingoing queue
90 V3MACHTYP	AO	VMD3_RG Block What process is in VMD3
91 V3MCCAPCOM	AO	BCA-RB Result of cap comp v3 bidmachtype
92 V3MHTYPCOM	AO	BCA-RB Comp vmd3 with bidmachtype
93 V3NOTASKS	AO	ACA-RB Number of tasks in vmd3 outqueue
94 V3QCONTROL	AO	ACA-CB Queue Control block for VMD3

Tag Name	Type	Description
95 V3QUEUCOST	AO	BCA-RB VMD3 Queue Cost
96 V3SCANSTAT	AO	Indicates vmd3 scan status
97 V3STOPDOWN	AO	Stops download of NC file to Process 3
98 V3STPSTART	AO	Stop/Start data bits for Vmd3 controller
99 V3TOOLAVLE	AO	BCA-RB V3 Tools available to do reqd tsk
100 V3TOOLRESD	AO	VMD3 - tools reserved for current task?
101 V4BAUDRATE	AO	The baud rate required for NC Controller
102 V4CNCSTART	AO	Starts download of NC progrm - Process4
103 V4CURRPART	AO	vmd4 current part number*
104 V4DOWNCOMP	AO	CNC Download for VMD4 is completed
105 V4DWNCOMPL	AO	The download from the q app is complete
106 V4INQUSIZE	AO	Number of tasks in vmd4's ingoing queue
107 V4MACHTYP	AO	The type of process and size in vmd4
108 V4MCCAPCOM	AO	BCA-RB Result of cap comp v4 bidmachtype
109 V4MHTYPCOM	AO	BCA-RB Comp vmd4 with bidmachtype
110 V4NOTASKS	AO	ACA-RB Number of tasks in vmd4 outqueue
111 V4QCONTROL	AO	VMD4's queue control block
112 V4QUEUCOST	AO	BCA-RB VMD4 Queue Cost
113 V4RNC01CCN	AO	VMD4 C01 Current Cycle number of task
114 V4RNC01CTM	AO	VMD4 C01 Cycle time of Curr task
115 V4RNC01LUN	AO	VMD4 C01 Load Unloadtime
116 V4RNC01NOP	AO	Number of parts to be made
117 V4RNC01SUP	AO	VMD4 C01 Setup Time
118 V4RNC02CCN	AO	VMD4 C02 Curr Cycle number
119 V4RNC02CTM	AO	VMD4 C02 Cycle time
120 V4RNC02LUN	AO	VMD4 C02 Load Unload time
121 V4RNC02NOP	AO	VMD4 C02 Number of parts to be made
122 V4RNC02SUP	AO	VMD4 C02 Setup time
123 V4RNC03CCN	AO	VMD4 C03 Current Cycle number
124 V4RNC03CTM	AO	VMD4 C03 Cycle time
125 V4RNC03LUN	AO	VMD4 C03 Load Unload Time
126 V4RNC03NOP	AO	VMD4 C03 Number of parts to be made

Tag Name	Type	Description
127 V4RNC03SUP	AO	VMD4 C03 Set up time
128 V4SCANSTAT	AO	Indicates vmd4 scan status
129 V4STOPDOWN	AO	Stops download of NC file to Process 4
130 V4STPSTART	AO	Stop/Start data bits for Vmd4 controller
131 V4TASKTIME	AO	The time required to do the task in vmd4
132 V4TOOLAVLE	AO	BCA-RB V4 Tools available to do reqd tsk
133 V4TOOLRESD	AO	VMD4 - tools reserved for current task?
134 VMD1PARITY	AO	Even/Odd Parity Status of v1 controller
135 VMD2PARITY	AO	Even/Odd Parity Status of v2 controller
136 VMD3PARITY	AO	Even/Odd Parity Status of v3 controller
137 VMD4PARITY	AO	Even/Odd Parity Status of v1 controller
138 VMDCURAUCT	AO	Number of VMD currently holding auction
139 RESETVMD1	DI	this resets vmd1
140 RESETVMD2	DI	this resets vmd2
141 RESETVMD4	DI	this resets vmd4
142 TASKTIMER	DI	starts the timing of the task
143 V1STARTTSK	DI	Start the task in VMD 1
144 V2STARTTSK	DI	Start the task in VMD 2
145 V4STARTTSK	DI	Start the task in vmd4
146 FINISHTIME	DO	turns to a "1' when finished job
147 V1MACHBUSY	DO	VMD1 Machine state
148 V1STARTUP	DO	Starts process 1
149 V1TSKISFIN	DO	The task in VMD1 is finished
150 V2STARTUP	DO	Starts process 2
151 V2TSKISFIN	DO	The task in VMD2 is finished
152 V3STARTUP	DO	Starts process 3
153 V4STARTUP	DO	Starts process 4
154 V4TSKISFIN	DO	The task in VMD4 is finished
155 DOWNLDCNC1	PG	Downloads NC file to Process 1
156 V1DOWNTASK	PG	Downloads task data to process 1
157 V1FINCYCLE	PG	VMD1 Cycle is finished
158 V1ISREADY	PG	Process 1 is ready to be started
159 V1SIMULATE	PG	This is vmd1's Master simulator program
160 V1STARTIT	PG	Start processing task in process 1
161 V2DOWNTASK	PG	Downloads task data to process 2
162 V2FINCYCLE	PG	VMD2 Cycle is finished
163 V2ISREADY	PG	Process 2 is ready to be started
164 V2SIMULATE	PG	This is vmd2's Master simulator program

Tag Name	Type	Description
165 V2STARTIT	PG	Start processing task in process 2
166 V3DOWNTASK	PG	Downloads task data to process 3
167 V3FINCYCLE	PG	VMD3 Cycle is finished
168 V3ISREADY	PG	Process 3 is ready to be started
169 V3STARTIT	PG	Start processing task in process 3
170 V4DOWNTASK	PG	Downloads task data to process 4
171 V4FINCYCLE	PG	VMD4 Cycle is finished
172 V4ISREADY	PG	Process 4 is ready to be started
173 V4SIMULATE	PG	This is VMD4's Master simulator program
174 V4STARTIT	PG	Start processing task in process 4
175 VMD1START	PG	Starts Manufacture of Task In VMD1
176 VMD2START	PG	Starts Manufacture of Task In VMD2
177 VMD4START	PG	Starts manufacture of task in vmd4
178 V1TSKTIMER	TM	The timer for the task in VMD1
179 V2TSKTIMER	TM	The timer for the task in VMD2
180 V4TSKTIMER	TM	The timer for the task in VMD4
181 CELLSNAME	TX	----
182 V1CNCTYPE	TX	----
183 V2CNCTYPE	TX	----
184 V3CNCTYPE	TX	----
185 V4CNCTYPE	TX	----
186 VMD1NCFILE	TX	----
187 VMD2NCFILE	TX	----
188 VMD3NCFILE	TX	----
189 VMD4NCFILE	TX	----

Appendix D

Experimental Data - Testing the Hybrid FMS System

This Appendix outlines the raw data used for the construction of the graphs as outlined in Chapter 6 "Testing the Hybrid FMS System". Only the data used to construct the graphs in Chapter 6 is presented as to present all data would have been excessive.

D.1 Experiment 1 - Raw Data

Cell Controller Data Collection - NC Milling Equipment, Experiment 1

VMD4 Cell 1(1.8)

Jobname	Actual Time(Clock)	Relative Time	TimeReal(Secs)	Event	Queuecost
Start Simulation	15:49:51	56991	0	Simulation Start	0.00
Shaft	15:50:24	57024	33	Add	99.00
Shaft	15:50:27	57027	36	Start Manufacture	99.00
Shaft	15:50:46	57046	55	Cycle 1 Complete.	86.63
Shaft	15:51:05	57065	74	Cycle 2 Complete.	74.25
Shaft	15:51:24	57084	93	Cycle 3 Complete.	61.88
Shaft	15:51:43	57103	112	Cycle 4 Complete.	49.50
Shaft	15:52:02	57122	131	Cycle 5 Complete.	37.13
Shaft	15:52:22	57142	151	Cycle 6 Complete.	24.75
Shaft	15:52:40	57160	169	Cycle 7 Complete.	12.38
Shaft	15:52:59	57179	188	Bottom Queue Add	0.00
Bush	15:53:12	57192	201	Add	159.00
Bush	15:53:14	57194	203	Start Manufacture	159.00
Bush	15:54:13	57253	262	Cycle 1 Complete.	106.00
Bush	15:55:12	57312	321	Cycle 2 Complete.	53.00
Bush	15:56:11	57371	380	Whole job Completed	0.00
Mount	15:58:18	57498	507	Add	595.00
Mount	15:58:20	57500	509	Start Manufacture	595.00
Mount	15:59:30	57570	579	Cycle 1 Complete.	528.89
Mount	16:00:41	57641	650	Cycle 2 Complete.	462.78
Key	16:01:49	57709	718	Add	672.78
Mount	16:01:53	57713	722	Cycle 3 Complete.	606.67
Mount	16:03:04	57784	793	Cycle 4 Complete.	540.56
Mount	16:04:15	57855	864	Cycle 5 Complete.	474.44
Mount	16:05:27	57927	936	Cycle 6 Complete.	408.33
Journ	16:06:05	57965	974	Add	690.33
Mount	16:06:36	57996	1005	Cycle 7 Complete.	624.22
Mount	16:07:48	58068	1077	Cycle 8 Complete.	558.11
Mount	16:08:59	58139	1148	Bottom Queue Add	492.00
Key	16:09:04	58144	1153	Start Manufacture	492.00
Houses	16:09:46	58186	1195	Add	526.00
Key	16:09:55	58195	1204	Cycle 1 Complete.	484.00
Key	16:10:43	58243	1252	Cycle 2 Complete.	442.00
Key	16:11:34	58294	1303	Cycle 3 Complete.	400.00
Stubs	16:11:49	58309	1318	Add	801.00
Key	16:12:24	58344	1353	Cycle 4 Complete.	759.00
Key	16:13:13	58393	1402	Bottom Queue Add	717.00
Journ	16:13:18	58398	1407	Start Manufacture	717.00

Journ	16:14:05	58445	1454	Cycle 1 Complete.	681.75
Journ	16:14:53	58493	1502	Cycle 2 Complete.	646.50
Bracks	16:15:30	58530	1539	Add	926.50
Journ	16:15:41	58541	1550	Cycle 3 Complete.	891.25
Journ	16:16:30	58590	1599	Cycle 4 Complete.	856.00
Journ	16:17:18	58638	1647	Cycle 5 Complete.	820.75
Journ	16:18:05	58685	1694	Cycle 6 Complete.	785.50
Journ	16:18:54	58734	1743	Cycle 7 Complete.	750.25
Journ	16:19:42	58782	1791	Bottom Queue Add	715.00
Houses	16:19:47	58787	1796	Start Manufacture	715.00
Houses	16:20:01	58801	1810	Cycle 1 Complete.	706.50
Houses	16:20:15	58815	1824	Cycle 2 Complete.	698.00
Houses	16:20:28	58828	1837	Cycle 3 Complete.	689.50
Houses	16:20:43	58843	1852	Whole job Completed	681.00
Stubs	16:20:48	58848	1857	Start Manufacture	681.00
Stubs	16:21:42	58902	1911	Cycle 1 Complete.	630.88
Stubs	16:22:37	58957	1966	Cycle 2 Complete.	580.75
Stubs	16:23:33	59013	2022	Cycle 3 Complete.	530.63
Stubs	16:24:28	59068	2077	Cycle 4 Complete.	480.50
Piston	16:24:48	59088	2097	Add	698.50
Stubs	16:25:23	59123	2132	Cycle 5 Complete.	648.38
Stubs	16:26:18	59178	2187	Cycle 6 Complete.	598.25
Hous	16:26:50	59210	2219	Add	680.25
Stubs	16:27:13	59233	2242	Cycle 7 Complete.	630.13
Stubs	16:28:07	59287	2296	Whole job Completed	580.00
Bracks	16:28:12	59292	2301	Start Manufacture	580.00
Bearig	16:28:33	59313	2322	Add	893.00
Bracks	16:29:20	59360	2369	Cycle 1 Complete.	837.00
Bracks	16:30:28	59428	2437	Cycle 2 Complete.	781.00
Mouns	16:31:14	59474	2483	Add	980.00
Bracks	16:31:36	59496	2505	Cycle 3 Complete.	924.00
Bracks	16:32:45	59565	2574	Cycle 4 Complete.	868.00
Bracks	16:33:53	59633	2642	Bottom Queue Add	812.00
Piston	16:33:58	59638	2647	Start Manufacture	812.00
Piston	16:34:31	59671	2680	Cycle 1 Complete.	784.75
Keyss	16:34:54	59694	2703	Add	1199.75
Piston	16:35:03	59703	2712	Cycle 2 Complete.	1172.50
Piston	16:35:36	59736	2745	Cycle 3 Complete.	1145.25
Piston	16:36:09	59769	2778	Cycle 4 Complete.	1118.00
Piston	16:36:42	59802	2811	Cycle 5 Complete.	1090.75
Piston	16:37:15	59835	2844	Cycle 6 Complete.	1063.50
Piston	16:37:48	59868	2877	Cycle 7 Complete.	1036.25
Bearin	16:38:09	59889	2898	Add	1273.25
Piston	16:38:22	59902	2911	Whole job Completed	1246.00
Hous	16:38:27	59907	2916	Start Manufacture	1246.00
Hous	16:38:40	59920	2929	Cycle 1 Complete.	1237.80
Hous	16:38:55	59935	2944	Cycle 2 Complete.	1229.60
Hous	16:39:09	59949	2958	Cycle 3 Complete.	1221.40
Hous	16:39:24	59964	2973	Cycle 4 Complete.	1213.20
Hous	16:39:37	59977	2986	Cycle 5 Complete.	1205.00
Hous	16:39:51	59991	3000	Cycle 6 Complete.	1196.80
Hous	16:40:05	60005	3014	Cycle 7 Complete.	1188.60

Hous	16:40:18	60018	3027	Cycle 8 Complete.	1180.40
Hous	16:40:33	60033	3042	Cycle 9 Complete.	1172.20
Hous	16:40:46	60046	3055	Whole job Completed	1164.00
Bearig	16:40:52	60052	3061	Start Manufacture	1164.00
Bearig	16:41:42	60102	3111	Cycle 1 Complete.	1124.88
Bearig	16:42:33	60153	3162	Cycle 2 Complete.	1085.75
Bearig	16:43:24	60204	3213	Cycle 3 Complete.	1046.63
Bearig	16:44:16	60256	3265	Cycle 4 Complete.	1007.50
Bearig	16:45:06	60306	3315	Cycle 5 Complete.	968.38
Bearig	16:45:57	60357	3366	Cycle 6 Complete.	929.25
Bearig	16:46:48	60408	3417	Cycle 7 Complete.	890.13
Pistos	16:47:32	60452	3461	Add	973.13
Bearig	16:47:40	60460	3469	Whole job Completed	934.00
Mouns	16:47:44	60464	3473	Start Manufacture	934.00
Mouns	16:48:54	60534	3543	Cycle 1 Complete.	867.67
Rotors	16:49:10	60550	3559	Add	1088.67
Mouns	16:50:05	60605	3614	Cycle 2 Complete.	1022.33
Mouns	16:51:16	60676	3685	Bottom Queue Add	956.00
Keyss	16:51:22	60682	3691	Start Manufacture	956.00
Keyss	16:52:11	60731	3740	Cycle 1 Complete.	914.50
Keyss	16:53:02	60782	3791	Cycle 2 Complete.	873.00
Keyss	16:53:51	60831	3840	Cycle 3 Complete.	831.50
Keyss	16:54:41	60881	3890	Cycle 4 Complete.	790.00
Keyss	16:55:32	60932	3941	Cycle 5 Complete.	748.50
Keyss	16:56:21	60981	3990	Cycle 6 Complete.	707.00
Keyss	16:57:11	61031	4040	Cycle 7 Complete.	665.50
Keyss	16:58:03	61083	4092	Cycle 8 Complete.	624.00
Keyss	16:58:50	61130	4139	Cycle 9 Complete.	582.50
Keyss	16:59:42	61182	4191	Bottom Queue Add	541.00
Bearin	16:59:46	61186	4195	Start Manufacture	541.00
Bearin	17:00:36	61236	4245	Cycle 1 Complete.	501.50
Bearin	17:01:27	61287	4296	Cycle 2 Complete.	462.00
Bearin	17:02:18	61338	4347	Cycle 3 Complete.	422.50
Bearin	17:03:09	61389	4398	Cycle 4 Complete.	383.00
Bearin	17:04:00	61440	4449	Cycle 5 Complete.	343.50
Bearin	17:04:52	61492	4501	Whole job Completed	304.00
Pistos	17:04:56	61496	4505	Start Manufacture	304.00
Pistos	17:05:28	61528	4537	Cycle 1 Complete.	276.33
Pistos	17:06:01	61561	4570	Cycle 2 Complete.	248.67
Pistos	17:06:35	61595	4604	Whole job Completed	221.00
Rotors	17:06:40	61600	4609	Start Manufacture	221.00
Rotors	17:07:19	61639	4648	Cycle 1 Complete.	189.43
Rotors	17:07:58	61678	4687	Cycle 2 Complete.	157.86
Rotors	17:08:37	61717	4726	Cycle 3 Complete.	126.29
Rotors	17:09:16	61756	4765	Cycle 4 Complete.	94.71
Rotors	17:09:55	61795	4804	Cycle 5 Complete.	63.14
Rotors	17:10:35	61835	4844	Cycle 6 Complete.	31.57
Rotors	17:11:13	61873	4882	Whole job Completed	0.00

VMD4 Cell 2(1.8)

Jobname	Actual Time(Clock)	Relative Time	TimeReal(Secs)	Event	Queuecost
Start Simulation	15:49:49	56989	0	Simulation Start	0.00
House	15:54:18	57258	269	Add	34.00
House	15:54:20	57260	271	Start Manufacture	34.00
House	15:54:34	57274	285	Cycle 1 Complete.	25.50
House	15:54:47	57287	298	Cycle 2 Complete.	17.00
House	15:55:02	57302	313	Cycle 3 Complete.	8.50
House	15:55:16	57316	327	Whole job Completed	0.00
Stub	15:56:10	57370	381	Add	51.00
Stub	15:56:12	57372	383	Start Manufacture	51.00
Stub	15:57:08	57428	439	Whole job Completed	0.00
Axle	15:57:36	57456	467	Add	883.00
Axle	15:57:39	57459	470	Start Manufacture	883.00
Axle	16:00:46	57646	657	Cycle 1 Complete.	735.83
Axle	16:03:53	57833	844	Cycle 2 Complete.	588.67
Axle	16:07:00	58020	1031	Cycle 3 Complete.	441.50
Bushes	16:08:33	58113	1124	Add	964.50
Axle	16:10:06	58206	1217	Cycle 4 Complete.	817.33
Axle	16:13:13	58393	1404	Cycle 5 Complete.	670.17
Axle	16:16:20	58580	1591	Whole job Completed	523.00
Bushes	16:16:25	58585	1596	Start Manufacture	523.00
Pins	16:16:54	58614	1625	Add	877.00
Bushes	16:17:24	58644	1655	Cycle 1 Complete.	824.70
Bushes	16:18:22	58702	1713	Cycle 2 Complete.	772.40
Bushes	16:19:22	58762	1773	Cycle 3 Complete.	720.10
Bushes	16:20:20	58820	1831	Cycle 4 Complete.	667.80
Bushes	16:21:20	58880	1891	Cycle 5 Complete.	615.50
Valves	16:21:39	58899	1910	Add	665.50
Bushes	16:22:19	58939	1950	Cycle 6 Complete.	613.20
Shaf	16:22:35	58955	1966	Add	724.20
Bushes	16:23:18	58998	2009	Cycle 7 Complete.	671.90
Bushes	16:24:16	59056	2067	Cycle 8 Complete.	619.60
Bushes	16:25:15	59115	2126	Cycle 9 Complete.	567.30
bushs	16:25:50	59150	2161	Add	1090.30
Bushes	16:26:15	59175	2186	Whole job Completed	1038.00
Pins	16:26:19	59179	2190	Start Manufacture	1038.00
Pins	16:27:11	59231	2242	Cycle 1 Complete.	998.67
Pins	16:28:06	59286	2297	Cycle 2 Complete.	959.33
Pins	16:29:00	59340	2351	Cycle 3 Complete.	920.00
Pins	16:29:55	59395	2406	Cycle 4 Complete.	880.67
Pins	16:30:49	59449	2460	Cycle 5 Complete.	841.33
Pins	16:31:42	59502	2513	Cycle 6 Complete.	802.00
Pins	16:32:36	59556	2567	Cycle 7 Complete.	762.67
Pins	16:33:31	59611	2622	Cycle 8 Complete.	723.33
axless	16:33:33	59613	2624	Add	1606.33
Pins	16:34:25	59665	2676	Whole job Completed	1567.00
Valves	16:34:30	59670	2681	Start Manufacture	1567.00
Valves	16:34:52	59692	2703	Cycle 1 Complete.	1550.33
Valves	16:35:14	59714	2725	Cycle 2 Complete.	1533.67

Valves	16:35:37	59737	2748	Bottom Queue Add	1517.00
Shaf	16:35:41	59741	2752	Start Manufacture	1517.00
Shaf	16:36:00	59760	2771	Cycle 1 Complete.	1504.67
Shaf	16:36:19	59779	2790	Cycle 2 Complete.	1492.33
Shaf	16:36:37	59797	2808	Cycle 3 Complete.	1480.00
Shaf	16:36:56	59816	2827	Cycle 4 Complete.	1467.67
Shaf	16:37:15	59835	2846	Cycle 5 Complete.	1455.33
Shaf	16:37:35	59855	2866	Cycle 6 Complete.	1443.00
Shaf	16:37:54	59874	2885	Cycle 7 Complete.	1430.67
Shaf	16:38:11	59891	2902	Cycle 8 Complete.	1418.33
Shaf	16:38:31	59911	2922	Bottom Queue Add	1406.00
bushs	16:38:36	59916	2927	Start Manufacture	1406.00
bushs	16:39:35	59975	2986	Cycle 1 Complete.	1353.70
bushs	16:40:33	60033	3044	Cycle 2 Complete.	1301.40
bushs	16:41:33	60093	3104	Cycle 3 Complete.	1249.10
bushs	16:42:31	60151	3162	Cycle 4 Complete.	1196.80
bushs	16:43:31	60211	3222	Cycle 5 Complete.	1144.50
bushs	16:44:29	60269	3280	Cycle 6 Complete.	1092.20
bushs	16:45:28	60328	3339	Cycle 7 Complete.	1039.90
bushs	16:46:27	60387	3398	Cycle 8 Complete.	987.60
bushs	16:47:26	60446	3457	Cycle 9 Complete.	935.30
bushs	16:48:25	60505	3516	Whole job Completed	883.00
axless	16:48:30	60510	3521	Start Manufacture	883.00
axless	16:51:37	60697	3708	Cycle 1 Complete.	735.83
Bears	16:53:12	60792	3803	Add	858.83
axless	16:54:43	60883	3894	Cycle 2 Complete.	711.67
Pistss	16:55:35	60935	3946	Add	740.67
axless	16:57:50	61070	4081	Cycle 3 Complete.	593.50
axless	17:00:57	61257	4268	Cycle 4 Complete.	446.33
Rotos	17:01:41	61301	4312	Add	667.33
axless	17:04:05	61445	4456	Cycle 5 Complete.	520.17
axless	17:07:11	61631	4642	Whole job Completed	373.00
Bears	17:07:16	61636	4647	Start Manufacture	373.00
Bears	17:08:07	61687	4698	Cycle 1 Complete.	332.00
Bears	17:08:58	61738	4749	Cycle 2 Complete.	291.00
Bears	17:09:49	61789	4800	Whole job Completed	250.00
Pistss	17:09:54	61794	4805	Start Manufacture	250.00
Pistss	17:10:27	61827	4838	Whole job Completed	221.00
Rotos	17:10:32	61832	4843	Start Manufacture	221.00
Rotos	17:11:11	61871	4882	Cycle 1 Complete.	189.43
Rotos	17:11:50	61910	4921	Cycle 2 Complete.	157.86
Rotos	17:12:29	61949	4960	Cycle 3 Complete.	126.28
Rotos	17:13:08	61988	4999	Cycle 4 Complete.	94.71
Rotos	17:13:47	62027	5038	Cycle 5 Complete.	63.14
Rotos	17:14:25	62065	5076	Cycle 6 Complete.	31.57
Rotos	17:15:05	62105	5116	Whole job Completed	0.00

VMD2 Cell 3(1.7)

Jobname	Actual Time(Clock)	Relative Time	TimeReal(Secs)	Event	Queuecost
Start Simulation	15:49:51	56991	0	Simulation Start	0.00
Pin	15:58:37	57517	526	Add	240.00
Pin	15:58:40	57520	529	Start Manufacture	240.00
Pin	15:59:34	57574	583	Cycle 1 Complete.	200.00
Brack	15:59:55	57595	604	Add	642.00
Pin	16:00:28	57628	637	Cycle 2 Complete.	602.00
Pin	16:01:22	57682	691	Cycle 3 Complete.	562.00
Pin	16:02:16	57736	745	Cycle 4 Complete.	522.00
Pin	16:03:09	57789	798	Cycle 5 Complete.	482.00
Pin	16:04:04	57844	853	Whole job Completed	442.00
Brack	16:04:08	57848	857	Start Manufacture	442.00
Valve	16:04:37	57877	886	Add	508.00
Brack	16:05:15	57915	924	Cycle 1 Complete.	452.75
Shafts	16:05:33	57933	942	Add	575.75
Brack	16:06:23	57983	992	Cycle 2 Complete.	520.50
Brack	16:07:31	58051	1060	Cycle 3 Complete.	465.25
Brack	16:08:40	58120	1129	Cycle 4 Complete.	410.00
Axles	16:09:06	58146	1155	Add	593.00
Brack	16:09:48	58188	1197	Cycle 5 Complete.	537.75
Brack	16:10:55	58255	1264	Cycle 6 Complete.	482.50
Brack	16:12:03	58323	1332	Cycle 7 Complete.	427.25
Brack	16:13:11	58391	1400	Bottom Queue Add	372.00
Valve	16:13:16	58396	1405	Start Manufacture	372.00
Valve	16:13:37	58417	1426	Cycle 1 Complete.	355.50
Mounts	16:13:53	58433	1442	Add	752.50
Valve	16:14:00	58440	1449	Cycle 2 Complete.	736.00
Valve	16:14:21	58461	1470	Cycle 3 Complete.	719.50
Valve	16:14:44	58484	1493	Bottom Queue Add	703.00
Shafts	16:14:49	58489	1498	Start Manufacture	703.00
Shafts	16:15:08	58508	1517	Cycle 1 Complete.	690.70
Shafts	16:15:27	58527	1536	Cycle 2 Complete.	678.40
Shafts	16:15:46	58546	1555	Cycle 3 Complete.	666.10
Shafts	16:16:05	58565	1574	Cycle 4 Complete.	653.80
Shafts	16:16:24	58584	1593	Cycle 5 Complete.	641.50
Shafts	16:16:44	58604	1613	Cycle 6 Complete.	629.20
Shafts	16:17:03	58623	1632	Cycle 7 Complete.	616.90
Shafts	16:17:22	58642	1651	Cycle 8 Complete.	604.60
Journs	16:17:38	58658	1667	Add	648.60
Shafts	16:17:41	58661	1670	Cycle 9 Complete.	636.30
Shafts	16:17:59	58679	1688	Bottom Queue Add	624.00
Axles	16:18:04	58684	1693	Start Manufacture	624.00
Keys	16:18:14	58694	1703	Add	916.00
Axles	16:21:10	58870	1879	Whole job Completed	733.00
Mounts	16:21:15	58875	1884	Start Manufacture	733.00
Mounts	16:22:25	58945	1954	Cycle 1 Complete.	666.83
Mounts	16:23:37	59017	2026	Cycle 2 Complete.	600.67
Mounts	16:24:48	59088	2097	Cycle 3 Complete.	534.50

Rotor	16:25:25	59125	2134	Add	817.50
Mounts	16:25:59	59159	2168	Cycle 4 Complete.	751.33
Mounts	16:27:09	59229	2238	Cycle 5 Complete.	685.17
Mounts	16:28:21	59301	2310	Bottom Queue Add	619.00
Journs	16:28:25	59305	2314	Start Manufacture	619.00
Stubss	16:28:55	59335	2344	Add	970.00
Journs	16:29:12	59352	2361	Bottom Queue Add	926.00
Keys	16:29:17	59357	2366	Start Manufacture	926.00
Keys	16:30:07	59407	2416	Cycle 1 Complete.	884.29
Keys	16:30:57	59457	2466	Cycle 2 Complete.	842.57
Keys	16:31:48	59508	2517	Cycle 3 Complete.	800.86
Bracs	16:32:08	59528	2537	Add	1134.86
Keys	16:32:38	59558	2567	Cycle 4 Complete.	1093.14
Keys	16:33:27	59607	2616	Cycle 5 Complete.	1051.43
Keys	16:34:17	59657	2666	Cycle 6 Complete.	1009.71
Keys	16:35:07	59707	2716	Bottom Queue Add	968.00
Rotor	16:35:13	59713	2722	Start Manufacture	968.00
Rotor	16:35:51	59751	2760	Cycle 1 Complete.	936.56
Rotor	16:36:30	59790	2799	Cycle 2 Complete.	905.11
Pinss	16:37:07	59827	2836	Add	1259.11
Rotor	16:37:09	59829	2838	Cycle 3 Complete.	1227.67
Rotor	16:37:48	59868	2877	Cycle 4 Complete.	1196.22
Rotor	16:38:27	59907	2916	Cycle 5 Complete.	1164.78
Valvs	16:38:58	59938	2947	Add	1262.78
Rotor	16:39:07	59947	2956	Cycle 6 Complete.	1231.33
Rotor	16:39:46	59986	2995	Cycle 7 Complete.	1199.89
Rotor	16:40:25	60025	3034	Cycle 8 Complete.	1168.44
Rotor	16:41:03	60063	3072	Whole job Completed	1137.00
Stubss	16:41:08	60068	3077	Start Manufacture	1137.00
Stubss	16:42:02	60122	3131	Cycle 1 Complete.	1086.86
Stubss	16:42:58	60178	3187	Cycle 2 Complete.	1036.71
Stubss	16:43:52	60232	3241	Cycle 3 Complete.	986.57
Stubss	16:44:47	60287	3296	Cycle 4 Complete.	936.43
Stubss	16:45:42	60342	3351	Cycle 5 Complete.	886.29
Stubss	16:46:37	60397	3406	Cycle 6 Complete.	836.14
Jourss	16:47:02	60422	3431	Add	1186.14
Stubss	16:47:32	60452	3461	Whole job Completed	1136.00
Bracs	16:47:38	60458	3467	Start Manufacture	1136.00
Bracs	16:48:46	60526	3535	Cycle 1 Complete.	1080.33
Bracs	16:49:55	60595	3604	Cycle 2 Complete.	1024.67
Bracs	16:51:02	60662	3671	Cycle 3 Complete.	969.00
Bracs	16:52:10	60730	3739	Cycle 4 Complete.	913.33
Bracs	16:53:18	60798	3807	Cycle 5 Complete.	857.67
Bracs	16:54:26	60866	3875	Bottom Queue Add	802.00
Pinss	16:54:31	60871	3880	Start Manufacture	802.00
Pinss	16:55:24	60924	3933	Cycle 1 Complete.	762.67
Pinss	16:56:19	60979	3988	Cycle 2 Complete.	723.33
Pinss	16:57:13	61033	4042	Cycle 3 Complete.	684.00
Pinss	16:58:06	61086	4095	Cycle 4 Complete.	644.67
Pinss	16:59:00	61140	4149	Cycle 5 Complete.	605.33
Pinss	16:59:55	61195	4204	Cycle 6 Complete.	566.00
Pinss	17:00:49	61249	4258	Cycle 7 Complete.	526.67

Pinss	17:01:42	61302	4311	Cycle 8 Complete.	487.33
Pinss	17:02:36	61356	4365	Whole job Completed	448.00
Valvs	17:02:42	61362	4371	Start Manufacture	448.00
Valvs	17:03:04	61384	4393	Cycle 1 Complete.	431.67
Valvs	17:03:25	61405	4414	Cycle 2 Complete.	415.33
Valvs	17:03:47	61427	4436	Cycle 3 Complete.	399.00
Valvs	17:04:09	61449	4458	Cycle 4 Complete.	382.67
Valvs	17:04:31	61471	4480	Cycle 5 Complete.	366.33
Valvs	17:04:53	61493	4502	Bottom Queue Add	350.00
Jourss	17:04:58	61498	4507	Start Manufacture	350.00
Jourss	17:05:45	61545	4554	Cycle 1 Complete.	315.00
Jourss	17:06:32	61592	4601	Cycle 2 Complete.	280.00
Jourss	17:07:21	61641	4650	Cycle 3 Complete.	245.00
Jourss	17:08:09	61689	4698	Cycle 4 Complete.	210.00
Jourss	17:08:57	61737	4746	Cycle 5 Complete.	175.00
Jourss	17:09:44	61784	4793	Cycle 6 Complete.	140.00
Jourss	17:10:33	61833	4842	Cycle 7 Complete.	105.00
Jourss	17:11:22	61882	4891	Cycle 8 Complete.	70.00
Jourss	17:12:08	61928	4937	Cycle 9 Complete.	35.00
Jourss	17:12:57	61977	4986	Bottom Queue Add	0.00

Cell Controller Data Collection - NC Grinding Equipment, Experiment 1

VMD2 Cell 1(3.7)

Jobname	Actual Time(Clock)	Relative Time	TimeReal(Secs)	Event	Queuecost
Start Simulation	15:49:51	56991	0	Simulation Start	0
Shaft	15:53:34	57214	223	Add	174
Shaft	15:53:37	57217	226	Start Manufacture	174
Shaft	15:54:08	57248	257	Cycle 1 Complete.	152.3
Shaft	15:54:39	57279	288	Cycle 2 Complete.	130.5
Shaft	15:55:10	57310	319	Cycle 3 Complete.	108.8
Shaft	15:55:41	57341	350	Cycle 4 Complete.	87
Shaft	15:56:12	57372	381	Cycle 5 Complete.	65.3
Shaft	15:56:43	57403	412	Cycle 6 Complete.	43.5
Shaft	15:57:15	57435	444	Cycle 7 Complete.	21.8
Shaft	15:57:45	57465	474	Bottom Queue Add	0
Spline	16:00:54	57654	663	Add	135
Spline	16:00:55	57655	664	Start Manufacture	135
Rotor	16:02:46	57766	775	Add	795
Spline	16:03:14	57794	803	Whole job Completed	660
Rotor	16:03:19	57799	808	Start Manufacture	660
Rotor	16:04:47	57887	896	Cycle 1 Complete.	586.7
Rotor	16:06:15	57975	984	Cycle 2 Complete.	513.3
Rotor	16:07:42	58062	1071	Cycle 3 Complete.	440
Rotor	16:09:10	58150	1159	Cycle 4 Complete.	366.7
Rotor	16:10:39	58239	1248	Cycle 5 Complete.	293.3
Rotor	16:12:06	58326	1335	Cycle 6 Complete.	220
Rotor	16:13:34	58414	1423	Cycle 7 Complete.	146.7

Bearin	16:14:49	58489	1498	Add	1239.7
Rotor	16:15:02	58502	1511	Cycle 8 Complete.	1166.3
Rotor	16:16:31	58591	1600	Bottom Queue Add	1093
Bearin	16:16:35	58595	1604	Start Manufacture	1093
Bearin	16:20:07	58807	1816	Cycle 1 Complete.	910.8
Bearin	16:23:39	59019	2028	Cycle 2 Complete.	728.7
Bearin	16:27:11	59231	2240	Cycle 3 Complete.	546.5
Bearin	16:30:43	59443	2452	Cycle 4 Complete.	364.3
Mount	16:32:40	59560	2569	Add	563.3
Bears	16:34:04	59644	2653	Add	1125.3
Bearin	16:34:15	59655	2664	Cycle 5 Complete.	943.2
Bearin	16:37:46	59866	2875	Bottom Queue Add	761
Mount	16:37:51	59871	2880	Start Manufacture	761
Mount	16:38:19	59899	2908	Cycle 1 Complete.	738.9
Mount	16:38:46	59926	2935	Cycle 2 Complete.	716.8
Mount	16:39:13	59953	2962	Cycle 3 Complete.	694.7
Shaf	16:39:33	59973	2982	Add	889.7
Mount	16:39:40	59980	2989	Cycle 4 Complete.	867.6
Mount	16:40:08	60008	3017	Cycle 5 Complete.	845.4
Mount	16:40:34	60034	3043	Cycle 6 Complete.	823.3
Mount	16:41:01	60061	3070	Cycle 7 Complete.	801.2
Mount	16:41:28	60088	3097	Cycle 8 Complete.	779.1
Mount	16:41:54	60114	3123	Whole job Completed	757
Bears	16:41:59	60119	3128	Start Manufacture	757
Bears	16:45:31	60331	3340	Cycle 1 Complete.	569.7
Bears	16:49:04	60544	3553	Cycle 2 Complete.	382.3
Bears	16:52:37	60757	3766	Bottom Queue Add	195
Shaf	16:52:42	60762	3771	Start Manufacture	195
Shaf	16:53:12	60792	3801	Cycle 1 Complete.	173.3
Shaf	16:53:43	60823	3832	Cycle 2 Complete.	151.7
Shaf	16:54:14	60854	3863	Cycle 3 Complete.	130
Shaf	16:54:46	60886	3895	Cycle 4 Complete.	108.3
Shaf	16:55:17	60917	3926	Cycle 5 Complete.	86.7
Shaf	16:55:48	60948	3957	Cycle 6 Complete.	65
Shaf	16:56:19	60979	3988	Cycle 7 Complete.	43.3
Shaf	16:56:49	61009	4018	Cycle 8 Complete.	21.7
Shaf	16:57:21	61041	4050	Bottom Queue Add	0
Mouns	16:59:12	61152	4161	Add	67
Mouns	16:59:13	61153	4162	Start Manufacture	67
Mouns	16:59:40	61180	4189	Cycle 1 Complete.	44.7
Mouns	17:00:07	61207	4216	Cycle 2 Complete.	22.3
Mouns	17:00:34	61234	4243	Whole job Completed	0

VMD2 Cell 2(3.8)

Jobname	Actual Time(Clock)	Relative Time	TimeReal(Secs)	Event	Queuecost
Start Simulation	15:49:49	56989	0	Simulation Start	0
Journ	15:57:05	57425	436	Add	426
Journ	15:57:08	57428	439	Start Manufacture	426
Journ	15:58:07	57487	498	Cycle 1 Complete.	372.75
Journ	15:59:06	57546	557	Cycle 2 Complete.	319.5
Journ	16:00:05	57605	616	Cycle 3 Complete.	266.25
Journ	16:01:04	57664	675	Cycle 4 Complete.	213
Journ	16:02:04	57724	735	Cycle 5 Complete.	159.75
Journ	16:03:03	57783	794	Cycle 6 Complete.	106.5
Piston	16:03:38	57818	829	Add	575.5
Journ	16:04:02	57842	853	Cycle 7 Complete.	522.25
Journ	16:05:01	57901	912	Bottom Queue Add	469
Piston	16:05:05	57905	916	Start Manufacture	469
Piston	16:06:19	57979	990	Cycle 1 Complete.	410.38
Piston	16:07:33	58053	1064	Cycle 2 Complete.	351.75
Piston	16:08:47	58127	1138	Cycle 3 Complete.	293.13
Piston	16:10:00	58200	1211	Cycle 4 Complete.	234.5
Piston	16:11:14	58274	1285	Cycle 5 Complete.	175.88
Piston	16:12:28	58348	1359	Cycle 6 Complete.	117.25
Journs	16:12:42	58362	1373	Add	172.25
Brack	16:13:24	58404	1415	Add	346.25
Piston	16:13:43	58423	1434	Cycle 7 Complete.	287.63
Piston	16:14:56	58496	1507	Bottom Queue Add	229
Valve	16:15:01	58501	1512	Add	422
Journs	16:15:02	58502	1513	Start Manufacture	422
Journs	16:16:01	58561	1572	Bottom Queue Add	367
Brack	16:16:06	58566	1577	Start Manufacture	367
Splins	16:16:37	58597	1608	Add	714
Brack	16:16:38	58598	1609	Cycle 1 Complete.	692.25
Brack	16:17:08	58628	1639	Cycle 2 Complete.	670.5
Brack	16:17:39	58659	1670	Cycle 3 Complete.	648.75
Brack	16:18:10	58690	1701	Cycle 4 Complete.	627
Brack	16:18:41	58721	1732	Cycle 5 Complete.	605.25
Brack	16:19:11	58751	1762	Cycle 6 Complete.	583.5
Rotors	16:19:21	58761	1772	Add	1099.5
Brack	16:19:43	58783	1794	Cycle 7 Complete.	1077.75
Brack	16:20:14	58814	1825	Bottom Queue Add	1056
Valve	16:20:18	58818	1829	Start Manufacture	1056
Valve	16:21:13	58873	1884	Cycle 1 Complete.	1007.75
Valve	16:22:10	58930	1941	Cycle 2 Complete.	959.5
Valve	16:23:06	58986	1997	Cycle 3 Complete.	911.25
Valve	16:24:01	59041	2052	Whole job Completed	863
Splins	16:24:06	59046	2057	Start Manufacture	863
Splins	16:26:26	59186	2197	Cycle 1 Complete.	747.33
Splins	16:28:44	59324	2335	Cycle 2 Complete.	631.67
Splins	16:31:03	59463	2474	Whole job Completed	516
Rotors	16:31:08	59468	2479	Start Manufacture	516
Rotors	16:32:35	59555	2566	Cycle 1 Complete.	442.29

Splin	16:33:13	59593	2604	Add	683.29
Rotors	16:34:04	59644	2655	Cycle 2 Complete.	609.57
Bracks	16:35:12	59712	2723	Add	720.57
Rotors	16:35:31	59731	2742	Cycle 3 Complete.	646.86
Valves	16:36:45	59805	2816	Add	792.86
Rotors	16:36:59	59819	2830	Cycle 4 Complete.	719.14
Pistss	16:37:44	59864	2875	Add	789.14
Rotors	16:38:27	59907	2918	Cycle 5 Complete.	715.43
Rotors	16:39:56	59996	3007	Cycle 6 Complete.	641.71
Rotors	16:41:24	60084	3095	Bottom Queue Add	568
Splin	16:41:29	60089	3100	Start Manufacture	568
Splin	16:43:48	60228	3239	Cycle 1 Complete.	447.5
Splin	16:46:07	60367	3378	Whole job Completed	327
Bracks	16:46:12	60372	3383	Start Manufacture	327
Bracks	16:46:43	60403	3414	Cycle 1 Complete.	304.8
Bracks	16:47:14	60434	3445	Cycle 2 Complete.	282.6
Bracks	16:47:45	60465	3476	Cycle 3 Complete.	260.4
Mounts	16:47:54	60474	3485	Add	393.4
Bracks	16:48:17	60497	3508	Cycle 4 Complete.	371.2
Bracks	16:48:48	60528	3539	Bottom Queue Add	349
Valves	16:48:52	60532	3543	Start Manufacture	349
Valves	16:49:48	60588	3599	Cycle 1 Complete.	300.33
Valves	16:50:44	60644	3655	Cycle 2 Complete.	251.67
Valves	16:51:41	60701	3712	Whole job Completed	203
Pistss	16:51:46	60706	3717	Start Manufacture	203
Pistss	16:53:00	60780	3791	Bottom Queue Add	133
Mounts	16:53:05	60785	3796	Start Manufacture	133
Mounts	16:53:32	60812	3823	Cycle 1 Complete.	110.83
Mounts	16:54:00	60840	3851	Cycle 2 Complete.	88.67
Mounts	16:54:27	60867	3878	Cycle 3 Complete.	66.5
Bracs	16:54:50	60890	3901	Add	198.5
Mounts	16:54:53	60893	3904	Cycle 4 Complete.	176.33
Mounts	16:55:21	60921	3932	Cycle 5 Complete.	154.17
Mounts	16:55:48	60948	3959	Whole job Completed	132
Bracs	16:55:53	60953	3964	Start Manufacture	132
Bracs	16:56:24	60984	3995	Cycle 1 Complete.	110
Bracs	16:56:54	61014	4025	Cycle 2 Complete.	88
Bracs	16:57:25	61045	4056	Cycle 3 Complete.	66
Bracs	16:57:57	61077	4088	Cycle 4 Complete.	44
Bracs	16:58:28	61108	4119	Cycle 5 Complete.	22
Bracs	16:58:59	61139	4150	Bottom Queue Add	0

VMD4 Cell 3(1.7)

Jobname	Actual Time(Clock)	Relative Time	TimeReal(Secs)	Event	Queuecost
Start Simulation	15:49:51	56991	0	Simulation Start	0
Bearig	15:59:28	57568	577	Add	1447
Bearig	15:59:30	57570	579	Start Manufacture	1447
Bearig	16:03:01	57781	790	Cycle 1 Complete.	1266.13
Bearig	16:06:34	57994	1003	Cycle 2 Complete.	1085.25
Bearig	16:10:05	58205	1214	Cycle 3 Complete.	904.38

Bearig	16:13:37	58417	1426	Cycle 4 Complete.	723.5
Bearig	16:17:10	58630	1639	Cycle 5 Complete.	542.63
Shafts	16:18:59	58739	1748	Add	758.63
Pistos	16:20:20	58820	1829	Add	942.63
Bearig	16:20:41	58841	1850	Cycle 6 Complete.	761.75
Bearig	16:24:13	59053	2062	Cycle 7 Complete.	580.88
Bearig	16:27:46	59266	2275	Bottom Queue Add	400
Shafts	16:27:50	59270	2279	Start Manufacture	400
Shafts	16:28:21	59301	2310	Cycle 1 Complete.	378.4
Shafts	16:28:52	59332	2341	Cycle 2 Complete.	356.8
Shafts	16:29:22	59362	2371	Cycle 3 Complete.	335.2
Shafts	16:29:53	59393	2402	Cycle 4 Complete.	313.6
Jourss	16:30:04	59404	2413	Add	845.6
Shafts	16:30:25	59425	2434	Cycle 5 Complete.	824
Shafts	16:30:56	59456	2465	Cycle 6 Complete.	802.4
Shafts	16:31:26	59486	2495	Cycle 7 Complete.	780.8
Shafts	16:31:57	59517	2526	Cycle 8 Complete.	759.2
Shafts	16:32:28	59548	2557	Cycle 9 Complete.	737.6
Shafts	16:32:59	59579	2588	Bottom Queue Add	716
Pistos	16:33:04	59584	2593	Start Manufacture	716
Pistos	16:34:19	59659	2668	Cycle 1 Complete.	654.67
Pistos	16:35:32	59732	2741	Cycle 2 Complete.	593.33
Rotos	16:36:13	59773	2782	Add	1109.33
Pistos	16:36:46	59806	2815	Bottom Queue Add	1048
Jourss	16:36:51	59811	2820	Start Manufacture	1048
Jourss	16:37:49	59869	2878	Cycle 1 Complete.	994.8
Jourss	16:38:48	59928	2937	Cycle 2 Complete.	941.6
Jourss	16:39:48	59988	2997	Cycle 3 Complete.	888.4
Jourss	16:40:46	60046	3055	Cycle 4 Complete.	835.2
Jourss	16:41:46	60106	3115	Cycle 5 Complete.	782
Jourss	16:42:44	60164	3173	Cycle 6 Complete.	728.8
Jourss	16:43:43	60223	3232	Cycle 7 Complete.	675.6
Jourss	16:44:43	60283	3292	Cycle 8 Complete.	622.4
Jourss	16:45:41	60341	3350	Cycle 9 Complete.	569.2
Jourss	16:46:40	60400	3409	Bottom Queue Add	516
Rotos	16:46:46	60406	3415	Start Manufacture	516
Rotos	16:48:13	60493	3502	Cycle 1 Complete.	442.29
Rotos	16:49:41	60581	3590	Cycle 2 Complete.	368.57
Rotos	16:51:10	60670	3679	Cycle 3 Complete.	294.86
Rotos	16:52:37	60757	3766	Cycle 4 Complete.	221.14
Rotos	16:54:05	60845	3854	Cycle 5 Complete.	147.43
Rotos	16:55:33	60933	3942	Cycle 6 Complete.	73.71
Rotos	16:57:01	61021	4030	Bottom Queue Add	0
Valvs	17:05:51	61551	4560	Add	287
Valvs	17:05:53	61553	4562	Start Manufacture	287
Valvs	17:06:48	61608	4617	Cycle 1 Complete.	239.17
Valvs	17:07:45	61665	4674	Cycle 2 Complete.	191.33
Valvs	17:08:41	61721	4730	Cycle 3 Complete.	143.5
Valvs	17:09:37	61777	4786	Cycle 4 Complete.	95.67
Valvs	17:10:33	61833	4842	Cycle 5 Complete.	47.83
Valvs	17:11:29	61889	4898	Whole job Completed	0

Cell Controller Data Collection - NC Lathe Equipment, Experiment 1

VMD1 Cell 1(2.3)

Jobname	Actual Time(Clock)	Relative Time	TimeReal(Secs)	Event	Queuecost
Start Simulation	15:49:51	56991	0	Simulation Start	0.00
Pin	15:51:19	57079	88	Add	282.00
Pin	15:51:20	57080	89	Start Manufacture	282.00
Pin	15:52:22	57142	151	Cycle 1 Complete.	235.00
Pin	15:53:22	57202	211	Cycle 2 Complete.	188.00
Pin	15:54:23	57263	272	Cycle 3 Complete.	141.00
Pin	15:55:24	57324	333	Cycle 4 Complete.	94.00
Pin	15:56:25	57385	394	Cycle 5 Complete.	47.00
Pin	15:57:25	57445	454	Bottom Queue Add	0.00
Shaft	15:58:54	57534	543	Add	373.00
Shaft	15:58:57	57537	546	Start Manufacture	373.00
Shaft	15:59:51	57591	600	Cycle 1 Complete.	326.38
Shaft	16:00:46	57646	655	Cycle 2 Complete.	279.75
Shaft	16:01:42	57702	711	Cycle 3 Complete.	233.13
Shaft	16:02:37	57757	766	Cycle 4 Complete.	186.50
Shaft	16:03:32	57812	821	Cycle 5 Complete.	139.88
Shaft	16:04:27	57867	876	Cycle 6 Complete.	93.25
Shaft	16:05:22	57922	931	Cycle 7 Complete.	46.63
Shaft	16:06:17	57977	986	Whole job Completed	0.00
Pins	16:06:43	58003	1012	Add	417.00
Pins	16:06:44	58004	1013	Start Manufacture	417.00
Pins	16:07:44	58064	1073	Cycle 1 Complete.	370.67
Pins	16:08:46	58126	1135	Cycle 2 Complete.	324.33
Pins	16:09:47	58187	1196	Cycle 3 Complete.	278.00
Pins	16:10:48	58248	1257	Cycle 4 Complete.	231.67
Pins	16:11:49	58309	1318	Cycle 5 Complete.	185.33
Pins	16:12:49	58369	1378	Cycle 6 Complete.	139.00
Pins	16:13:50	58430	1439	Cycle 7 Complete.	92.67
Pins	16:14:51	58491	1500	Cycle 8 Complete.	46.33
Pins	16:15:52	58552	1561	Bottom Queue Add	0.00
Piston	16:16:03	58563	1572	Add	365.00
Piston	16:16:06	58566	1575	Start Manufacture	365.00
Piston	16:17:07	58627	1636	Cycle 1 Complete.	319.38
Piston	16:18:08	58688	1697	Cycle 2 Complete.	273.75
Piston	16:19:09	58749	1758	Cycle 3 Complete.	228.13
Piston	16:20:10	58810	1819	Cycle 4 Complete.	182.50
Journ	16:20:44	58844	1853	Add	399.50
Piston	16:21:10	58870	1879	Cycle 5 Complete.	353.88
Piston	16:22:11	58931	1940	Cycle 6 Complete.	308.25
Piston	16:23:13	58993	2002	Cycle 7 Complete.	262.63
Piston	16:24:13	59053	2062	Bottom Queue Add	217.00
Journ	16:24:18	59058	2067	Start Manufacture	217.00
axless	16:24:31	59071	2080	Add	436.00
Journ	16:24:50	59090	2099	Cycle 1 Complete.	408.88

Journ	16:25:22	59122	2131	Cycle 2 Complete.	381.75
Journ	16:25:54	59154	2163	Cycle 3 Complete.	354.63
Journ	16:26:26	59186	2195	Cycle 4 Complete.	327.50
Journ	16:26:59	59219	2228	Cycle 5 Complete.	300.38
Journ	16:27:30	59250	2259	Cycle 6 Complete.	273.25
Journ	16:28:01	59281	2290	Cycle 7 Complete.	246.13
Journ	16:28:33	59313	2322	Whole job Completed	219.00
axless	16:28:38	59318	2327	Start Manufacture	219.00
axless	16:29:20	59360	2369	Cycle 1 Complete.	182.50
axless	16:30:03	59403	2412	Cycle 2 Complete.	146.00
axless	16:30:47	59447	2456	Cycle 3 Complete.	109.50
Journs	16:30:49	59449	2458	Add	137.50
axless	16:31:29	59489	2498	Cycle 4 Complete.	101.00
axless	16:32:13	59533	2542	Cycle 5 Complete.	64.50
axless	16:32:55	59575	2584	Bottom Queue Add	28.00
Journs	16:33:00	59580	2589	Start Manufacture	28.00
Journs	16:33:31	59611	2620	Whole job Completed	0.00
Shafts	16:34:33	59673	2682	Add	465.00
Shafts	16:34:35	59675	2684	Start Manufacture	465.00
Shafts	16:35:30	59730	2739	Cycle 1 Complete.	418.50
Shafts	16:36:25	59785	2794	Cycle 2 Complete.	372.00
Shafts	16:37:20	59840	2849	Cycle 3 Complete.	325.50
Pistos	16:37:28	59848	2857	Add	470.50
Shafts	16:38:15	59895	2904	Cycle 4 Complete.	424.00
Shafts	16:39:11	59951	2960	Cycle 5 Complete.	377.50
Shafts	16:40:06	60006	3015	Cycle 6 Complete.	331.00
Shafts	16:41:01	60061	3070	Cycle 7 Complete.	284.50
Shafts	16:41:55	60115	3124	Cycle 8 Complete.	238.00
Shafts	16:42:50	60170	3179	Cycle 9 Complete.	191.50
Shafts	16:43:45	60225	3234	Whole job Completed	145.00
Pistos	16:43:50	60230	3239	Start Manufacture	145.00
Pistos	16:44:51	60291	3300	Cycle 1 Complete.	96.67
Pistos	16:45:52	60352	3361	Cycle 2 Complete.	48.33
Pistos	16:46:55	60415	3424	Bottom Queue Add	0.00
Mouns	16:51:35	60695	3704	Add	359.00
Mouns	16:51:38	60698	3707	Start Manufacture	359.00
Mouns	16:54:01	60841	3850	Cycle 1 Complete.	239.33
Mouns	16:56:23	60983	3992	Cycle 2 Complete.	119.67
Mouns	16:58:46	61126	4135	Bottom Queue Add	0.00
Bracs	16:59:38	61178	4187	Add	63.00
Bracs	16:59:40	61180	4189	Start Manufacture	63.00
Bracs	16:59:57	61197	4206	Cycle 1 Complete.	52.50
Bracs	17:00:14	61214	4223	Cycle 2 Complete.	42.00
Keyss	17:00:28	61228	4237	Add	630.00
Bracs	17:00:31	61231	4240	Cycle 3 Complete.	619.50
Bracs	17:00:48	61248	4257	Cycle 4 Complete.	609.00
Bracs	17:01:05	61265	4274	Cycle 5 Complete.	598.50
Bracs	17:01:22	61282	4291	Whole job Completed	588.00
Keyss	17:01:27	61287	4296	Start Manufacture	588.00
Keyss	17:02:37	61357	4366	Cycle 1 Complete.	529.20
Keyss	17:03:47	61427	4436	Cycle 2 Complete.	470.40
Keyss	17:04:57	61497	4506	Cycle 3 Complete.	411.60

Keyss	17:06:07	61567	4576	Cycle 4 Complete.	352.80
Keyss	17:07:17	61637	4646	Cycle 5 Complete.	294.00
Keyss	17:08:27	61707	4716	Cycle 6 Complete.	235.20
Keyss	17:09:37	61777	4786	Cycle 7 Complete.	176.40
Keyss	17:10:47	61847	4856	Cycle 8 Complete.	117.60
Keyss	17:11:58	61918	4927	Cycle 9 Complete.	58.80
Keyss	17:13:08	61988	4997	Whole job Completed	0.00

VMD1 Cell 2(2.7)

Jobname	Actual Time(Clock)	Relative Time	TimeReal(Secs)	Event	Queuecost
Start Simulation	15:49:49	56989	0	Simulation Start	0.00
Axle	15:52:13	57133	144	Add	219.00
Axle	15:52:15	57135	146	Start Manufacture	219.00
Axle	15:52:58	57178	189	Cycle 1 Complete.	182.50
Axle	15:53:42	57222	233	Cycle 2 Complete.	146.00
Axle	15:54:25	57265	276	Cycle 3 Complete.	109.50
Axle	15:55:08	57308	319	Cycle 4 Complete.	73.00
Axle	15:55:51	57351	362	Cycle 5 Complete.	36.50
Axle	15:56:34	57394	405	Bottom Queue Add	0.00
Axles	16:07:37	58057	1068	Add	39.00
Axles	16:07:40	58060	1071	Start Manufacture	39.00
Axles	16:08:22	58102	1113	Bottom Queue Add	0.00
Mount	16:10:04	58204	1215	Add	1019.00
Mount	16:10:06	58206	1217	Start Manufacture	1019.00
Mount	16:12:29	58349	1360	Cycle 1 Complete.	905.78
Mount	16:14:52	58492	1503	Cycle 2 Complete.	792.56
Mount	16:17:15	58635	1646	Cycle 3 Complete.	679.33
Mount	16:19:38	58778	1789	Cycle 4 Complete.	566.11
Mount	16:22:02	58922	1933	Cycle 5 Complete.	452.89
Mount	16:24:24	59064	2075	Cycle 6 Complete.	339.67
Mount	16:26:47	59207	2218	Cycle 7 Complete.	226.44
Bears	16:27:42	59262	2273	Add	283.44
Mount	16:29:11	59351	2362	Cycle 8 Complete.	170.22
Mounts	16:29:36	59376	2387	Add	859.22
Mount	16:31:34	59494	2505	Bottom Queue Add	746.00
Bears	16:31:38	59498	2509	Start Manufacture	746.00
Bears	16:32:02	59522	2533	Cycle 1 Complete.	727.00
Bears	16:32:27	59547	2558	Cycle 2 Complete.	708.00
Bears	16:32:52	59572	2583	Bottom Queue Add	689.00
Mounts	16:32:58	59578	2589	Start Manufacture	689.00
Mounts	16:35:20	59720	2731	Cycle 1 Complete.	574.17
Mounts	16:37:43	59863	2874	Cycle 2 Complete.	459.33
Mounts	16:40:06	60006	3017	Cycle 3 Complete.	344.50
Mounts	16:42:29	60149	3160	Cycle 4 Complete.	229.67
Mounts	16:44:52	60292	3303	Cycle 5 Complete.	114.83
Mounts	16:47:15	60435	3446	Bottom Queue Add	0.00
Bracks	16:49:54	60594	3605	Add	53.00
Bracks	16:49:56	60596	3607	Start Manufacture	53.00
Bracks	16:50:13	60613	3624	Cycle 1 Complete.	42.40
Bracks	16:50:30	60630	3641	Cycle 2 Complete.	31.80

Bracks	16:50:47	60647	3658	Cycle 3 Complete.	21.20
Bracks	16:51:04	60664	3675	Cycle 4 Complete.	10.60
Bracks	16:51:21	60681	3692	Whole job Completed	0.00
Pistss	16:53:37	60817	3828	Add	57.00
Pistss	16:53:39	60819	3830	Start Manufacture	57.00
Pistss	16:54:40	60880	3891	Bottom Queue Add	0.00
Shaf	16:58:08	61088	4099	Add	419.00
Shaf	16:58:11	61091	4102	Start Manufacture	419.00
Shaf	16:59:06	61146	4157	Cycle 1 Complete.	372.44
Shaf	17:00:01	61201	4212	Cycle 2 Complete.	325.89
Shaf	17:00:56	61256	4267	Cycle 3 Complete.	279.33
Shaf	17:01:51	61311	4322	Cycle 4 Complete.	232.78
Shaf	17:02:46	61366	4377	Cycle 5 Complete.	186.22
Shaf	17:03:42	61422	4433	Cycle 6 Complete.	139.67
Shaf	17:04:36	61476	4487	Cycle 7 Complete.	93.11
Shaf	17:05:31	61531	4542	Cycle 8 Complete.	46.56
Shaf	17:06:26	61586	4597	Whole job Completed	0.00

VMD2 Cell 3(1.7)

Jobname	Actual Time(Clock)	Relative Time	TimeReal(Secs)	Event	Queuecost
Start Simulation	15:49:51	56991	0	Simulation Start	0.00
Bearig	15:55:16	57316	325	Add	147.00
Bearig	15:55:18	57318	327	Start Manufacture	147.00
Bearig	15:55:43	57343	352	Cycle 1 Complete.	128.63
Bearig	15:56:08	57368	377	Cycle 2 Complete.	110.25
Bearig	15:56:33	57393	402	Cycle 3 Complete.	91.88
Bearig	15:56:57	57417	426	Cycle 4 Complete.	73.50
Bearig	15:57:23	57443	452	Cycle 5 Complete.	55.13
Bearig	15:57:48	57468	477	Cycle 6 Complete.	36.75
Bearig	15:58:12	57492	501	Cycle 7 Complete.	18.38
Bearig	15:58:38	57518	527	Bottom Queue Add	0.00
Bearin	16:10:51	58251	1260	Add	111.00
Bearin	16:10:54	58254	1263	Start Manufacture	111.00
Bearin	16:11:19	58279	1288	Cycle 1 Complete.	92.50
Bearin	16:11:44	58304	1313	Cycle 2 Complete.	74.00
Bearin	16:12:09	58329	1338	Cycle 3 Complete.	55.50
Bearin	16:12:33	58353	1362	Cycle 4 Complete.	37.00
Bearin	16:12:58	58378	1387	Cycle 5 Complete.	18.50
Bearin	16:13:23	58403	1412	Bottom Queue Add	0.00
Key	16:14:14	58454	1463	Add	298.00
Key	16:14:17	58457	1466	Start Manufacture	298.00
Key	16:15:26	58526	1535	Cycle 1 Complete.	238.40
Key	16:16:37	58597	1606	Cycle 2 Complete.	178.80
Rotor	16:17:12	58632	1641	Add	381.80
Key	16:17:46	58666	1675	Cycle 3 Complete.	322.20
Key	16:18:57	58737	1746	Cycle 4 Complete.	262.60
Key	16:20:07	58807	1816	Whole job Completed	203.00
Rotor	16:20:12	58812	1821	Start Manufacture	203.00
Rotor	16:20:43	58843	1852	Cycle 1 Complete.	180.44
Brack	16:21:04	58864	1873	Add	263.44
Rotor	16:21:14	58874	1883	Cycle 2 Complete.	240.89

Rotor	16:21:45	58905	1914	Cycle 3 Complete.	218.33
Rotor	16:22:16	58936	1945	Cycle 4 Complete.	195.78
Rotor	16:22:47	58967	1976	Cycle 5 Complete.	173.22
Rotor	16:23:19	58999	2008	Cycle 6 Complete.	150.67
Pinss	16:23:34	59014	2023	Add	567.67
Rotor	16:23:49	59029	2038	Cycle 7 Complete.	545.11
Rotor	16:24:20	59060	2069	Cycle 8 Complete.	522.56
Rotor	16:24:52	59092	2101	Bottom Queue Add	500.00
Brack	16:24:56	59096	2105	Start Manufacture	500.00
Brack	16:25:13	59113	2122	Cycle 1 Complete.	489.63
Brack	16:25:29	59129	2138	Cycle 2 Complete.	479.25
Brack	16:25:47	59147	2156	Cycle 3 Complete.	468.88
Brack	16:26:03	59163	2172	Cycle 4 Complete.	458.50
Brack	16:26:20	59180	2189	Cycle 5 Complete.	448.13
Brack	16:26:37	59197	2206	Cycle 6 Complete.	437.75
Brack	16:26:54	59214	2223	Cycle 7 Complete.	427.38
Brack	16:27:11	59231	2240	Whole job Completed	417.00
Pinss	16:27:16	59236	2245	Start Manufacture	417.00
Pinss	16:28:17	59297	2306	Cycle 1 Complete.	370.67
Pinss	16:29:17	59357	2366	Cycle 2 Complete.	324.33
Pinss	16:30:19	59419	2428	Cycle 3 Complete.	278.00
Pinss	16:31:19	59479	2488	Cycle 4 Complete.	231.67
Pinss	16:32:21	59541	2550	Cycle 5 Complete.	185.33
Pinss	16:33:21	59601	2610	Cycle 6 Complete.	139.00
Pinss	16:34:22	59662	2671	Cycle 7 Complete.	92.67
Pinss	16:35:24	59724	2733	Cycle 8 Complete.	46.33
Keys	16:35:49	59749	2758	Add	460.33
Pinss	16:36:24	59784	2793	Bottom Queue Add	414.00
Keys	16:36:29	59789	2798	Start Manufacture	414.00
Keys	16:37:39	59859	2868	Cycle 1 Complete.	354.86
Keys	16:38:48	59928	2937	Cycle 2 Complete.	295.71
Keys	16:39:58	59998	3007	Cycle 3 Complete.	236.57
Keys	16:41:08	60068	3077	Cycle 4 Complete.	177.43
Rotors	16:41:59	60119	3128	Add	336.43
Keys	16:42:18	60138	3147	Cycle 5 Complete.	277.29
Keys	16:43:28	60208	3217	Cycle 6 Complete.	218.14
Keys	16:44:39	60279	3288	Whole job Completed	159.00
Rotors	16:44:43	60283	3292	Start Manufacture	159.00
Rotors	16:45:14	60314	3323	Cycle 1 Complete.	136.29
Rotors	16:45:44	60344	3353	Cycle 2 Complete.	113.57
Rotors	16:46:15	60375	3384	Cycle 3 Complete.	90.86
Rotors	16:46:47	60407	3416	Cycle 4 Complete.	68.14
Rotors	16:47:18	60438	3447	Cycle 5 Complete.	45.43
Rotors	16:47:48	60468	3477	Cycle 6 Complete.	22.71
Rotors	16:48:19	60499	3508	Bottom Queue Add	0.00
Rotos	16:57:40	61060	4069	Add	159.00
Rotos	16:57:43	61063	4072	Start Manufacture	159.00
Rotos	16:58:14	61094	4103	Cycle 1 Complete.	136.29
Rotos	16:58:44	61124	4133	Cycle 2 Complete.	113.57
Rotos	16:59:16	61156	4165	Cycle 3 Complete.	90.86
Rotos	16:59:47	61187	4196	Cycle 4 Complete.	68.14
Rotos	17:00:19	61219	4228	Cycle 5 Complete.	45.43

Rotos	17:00:50	61250	4259	Cycle 6 Complete.	22.71
Rotos	17:01:20	61280	4289	Bottom Queue Add	0.00
Jourss	17:13:54	62034	5043	Add	271.00
Jourss	17:13:56	62036	5045	Start Manufacture	271.00
Jourss	17:14:28	62068	5077	Cycle 1 Complete.	243.90
Jourss	17:14:59	62099	5108	Cycle 2 Complete.	216.80
Jourss	17:15:31	62131	5140	Cycle 3 Complete.	189.70
Jourss	17:16:03	62163	5172	Cycle 4 Complete.	162.60
Jourss	17:16:35	62195	5204	Cycle 5 Complete.	135.50
Jourss	17:17:07	62227	5236	Cycle 6 Complete.	108.40
Jourss	17:17:39	62259	5268	Cycle 7 Complete.	81.30
Jourss	17:18:11	62291	5300	Cycle 8 Complete.	54.20
Jourss	17:18:43	62323	5332	Cycle 9 Complete.	27.10
Jourss	17:19:15	62355	5364	Whole job Completed	0.00

D.2 Experiment 2 - Raw Data

Cell Controller Data Collection - NC Milling Equipment, Experiment_2

VMD4 Cell 1(1.8)

Jobname	Actual Time(Clock)	Relative Time	TimeReal(Secs)	Event	Queuecost
Start Simulation	03:19:55	11995	0	Simulation Start	0.0
Shaft	03:21:02	12062	67	Add	99.0
Shaft	03:21:05	12065	70	Start Manufacture	99.0
Shaft	03:21:24	12084	89	Cycle 1 Complete.	86.6
Shaft	03:21:43	12103	108	Cycle 2 Complete.	74.3
Shaft	03:22:01	12121	126	Cycle 3 Complete.	61.9
Shaft	03:22:20	12140	145	Cycle 4 Complete.	49.5
Shaft	03:22:40	12160	165	Cycle 5 Complete.	37.1
Shaft	03:22:59	12179	184	Cycle 6 Complete.	24.8
Shaft	03:23:18	12198	203	Cycle 7 Complete.	12.4
Shaft	03:23:21	12201	206	Bottom Queue Add	0.0
Bush	03:23:49	12229	234	Add	159.0
Bush	03:23:52	12232	237	Start Manufacture	159.0
Bush	03:24:50	12290	295	Cycle 1 Complete.	106.0
Bush	03:25:49	12349	354	Cycle 2 Complete.	53.0
Bush	03:26:48	12408	413	Whole job Completed	0.0
Mount	03:28:57	12537	542	Add	595.0
Mount	03:29:00	12540	545	Start Manufacture	595.0
Mount	03:30:10	12610	615	Cycle 1 Complete.	528.9
Mount	03:31:22	12682	687	Cycle 2 Complete.	462.8
Key	03:32:28	12748	753	Add	672.8
Mount	03:32:32	12752	757	Cycle 3 Complete.	606.7
Mount	03:33:43	12823	828	Cycle 4 Complete.	540.6
Mount	03:34:54	12894	899	Cycle 5 Complete.	474.4
Mount	03:36:05	12965	970	Cycle 6 Complete.	408.3
Shafts	03:36:17	12977	982	Add	531.3
Mount	03:37:17	13037	1042	Cycle 7 Complete.	465.2
Mount	03:38:28	13108	1113	Cycle 8 Complete.	399.1
Bushes	03:39:18	13158	1163	Add	922.1
Mount	03:39:39	13179	1184	Bottom Queue Add	856.0
Key	03:39:43	13183	1188	Start Manufacture	856.0
Key	03:40:32	13232	1237	Cycle 1 Complete.	814.0
Key	03:41:23	13283	1288	Cycle 2 Complete.	772.0
Key	03:42:12	13332	1337	Cycle 3 Complete.	730.0
Key	03:43:03	13383	1388	Cycle 4 Complete.	688.0
Key	03:43:53	13433	1438	Bottom Queue Add	646.0
Shafts	03:43:58	13438	1443	Start Manufacture	646.0
Shafts	03:44:17	13457	1462	Cycle 1 Complete.	633.7
Shafts	03:44:36	13476	1481	Cycle 2 Complete.	621.4

Shafts	03:44:55	13495	1500	Cycle 3 Complete.	609.1
Shafts	03:45:14	13514	1519	Cycle 4 Complete.	596.8
Shafts	03:45:34	13534	1539	Cycle 5 Complete.	584.5
Shafts	03:45:52	13552	1557	Cycle 6 Complete.	572.2
Bracks	03:46:10	13570	1575	Add	852.2
Shafts	03:46:12	13572	1577	Cycle 7 Complete.	839.9
Shafts	03:46:30	13590	1595	Cycle 8 Complete.	827.6
Shafts	03:46:50	13610	1615	Cycle 9 Complete.	815.3
Shafts	03:47:08	13628	1633	Bottom Queue Add	803.0
Bushes	03:47:13	13633	1638	Start Manufacture	803.0
Bushes	03:48:12	13692	1697	Cycle 1 Complete.	750.7
Bushes	03:49:11	13751	1756	Cycle 2 Complete.	698.4
Bushes	03:50:10	13810	1815	Cycle 3 Complete.	646.1
Bushes	03:51:09	13869	1874	Cycle 4 Complete.	593.8
Bushes	03:52:07	13927	1932	Cycle 5 Complete.	541.5
Valves	03:52:12	13932	1937	Add	591.5
Bushes	03:53:06	13986	1991	Cycle 6 Complete.	539.2
Shaf	03:53:21	14001	2006	Add	650.2
Bushes	03:54:05	14045	2050	Cycle 7 Complete.	597.9
Bushes	03:55:05	14105	2110	Cycle 8 Complete.	545.6
Bushes	03:56:03	14163	2168	Cycle 9 Complete.	493.3
Rotor	03:56:10	14170	2175	Add	776.3
Bushes	03:57:03	14223	2228	Whole job Completed	724.0
Bracks	03:57:08	14228	2233	Start Manufacture	724.0
Bracks	03:58:17	14297	2302	Cycle 1 Complete.	668.0
Bracks	03:59:24	14364	2369	Cycle 2 Complete.	612.0
Stubss	03:59:39	14379	2384	Add	963.0
Bracks	04:00:32	14432	2437	Cycle 3 Complete.	907.0
Bracks	04:01:40	14500	2505	Cycle 4 Complete.	851.0
Bracks	04:01:51	14511	2516	VMD4 Process Equipment Failure	0.0

VMD4 Cell 2(1.8)

Jobname	Actual Time(Clock)	Relative Time	TimeReal(Secs)	Event	Queuecost
Start Simulation	03:19:52	11992	0	Simulation Start	0.00
House	03:24:56	12296	304	Add	34.00
House	03:24:58	12298	306	Start Manufacture	34.00
House	03:25:13	12313	321	Cycle 1 Complete.	25.50
House	03:25:27	12327	335	Cycle 2 Complete.	17.00
House	03:25:41	12341	349	Cycle 3 Complete.	8.50
House	03:25:55	12355	363	Whole job Completed	0.00
Stub	03:26:48	12408	416	Add	51.00
Stub	03:26:49	12409	417	Start Manufacture	51.00
Stub	03:27:45	12465	473	Whole job Completed	0.00
Axle	03:28:15	12495	503	Add	883.00
Axle	03:28:17	12497	505	Start Manufacture	883.00

Axle	03:31:23	12683	691	Cycle 1 Complete.	735.83
Axle	03:34:31	12871	879	Cycle 2 Complete.	588.67
Axle	03:37:37	13057	1065	Cycle 3 Complete.	441.50
Axles	03:39:45	13185	1193	Add	624.50
Axle	03:40:43	13243	1251	Cycle 4 Complete.	477.33
Stubs	03:42:28	13348	1356	Add	878.33
Axle	03:43:52	13432	1440	Cycle 5 Complete.	731.17
Axle	03:46:58	13618	1626	Whole job Completed	584.00
Axles	03:47:03	13623	1631	Start Manufacture	584.00
Pins	03:47:34	13654	1662	Add	938.00
Axles	03:50:10	13810	1818	Whole job Completed	755.00
Stubs	03:50:14	13814	1822	Start Manufacture	755.00
Stubs	03:51:09	13869	1877	Cycle 1 Complete.	704.87
Stubs	03:52:04	13924	1932	Cycle 2 Complete.	654.75
Stubs	03:52:58	13978	1986	Cycle 3 Complete.	604.62
Stubs	03:53:53	14033	2041	Cycle 4 Complete.	554.50
Stubs	03:54:49	14089	2097	Cycle 5 Complete.	504.37
Piston	03:55:28	14128	2136	Add	722.37
Stubs	03:55:44	14144	2152	Cycle 6 Complete.	672.25
Stubs	03:56:38	14198	2206	Cycle 7 Complete.	622.12
Hous	03:57:29	14249	2257	Add	704.12
Stubs	03:57:34	14254	2262	Whole job Completed	654.00
Pins	03:57:38	14258	2266	Start Manufacture	654.00
Pins	03:58:32	14312	2320	Cycle 1 Complete.	614.67
Bearig	03:59:12	14352	2360	Add	927.67
Pins	03:59:25	14365	2373	Cycle 2 Complete.	888.33
Pins	04:00:20	14420	2428	Cycle 3 Complete.	849.00
Pins	04:01:14	14474	2482	Cycle 4 Complete.	809.67
Pins	04:02:08	14528	2536	Cycle 5 Complete.	770.33
Valves	04:02:13	14533	2541	Add	820.33
Pins	04:03:01	14581	2589	Cycle 6 Complete.	781.00
Bracs	04:03:14	14594	2602	Add	1115.00
Pins	04:03:56	14636	2644	Cycle 7 Complete.	1075.67
Pins	04:04:49	14689	2697	Cycle 8 Complete.	1036.33
axless	04:05:43	14743	2751	Add	1919.33
Pins	04:05:45	14745	2753	Whole job Completed	1880.00
Piston	04:05:49	14749	2757	Start Manufacture	1880.00
Piston	04:06:21	14781	2789	Cycle 1 Complete.	1852.75
Piston	04:06:55	14815	2823	Cycle 2 Complete.	1825.50
Piston	04:07:28	14848	2856	Cycle 3 Complete.	1798.25
Pinss	04:07:47	14867	2875	Add	2152.25
Piston	04:08:01	14881	2889	Cycle 4 Complete.	2125.00
Piston	04:08:34	14914	2922	Cycle 5 Complete.	2097.75
Piston	04:09:08	14948	2956	Cycle 6 Complete.	2070.50
Piston	04:09:40	14980	2988	Cycle 7 Complete.	2043.25
Piston	04:10:13	15013	3021	Whole job Completed	2016.00
Hous	04:10:18	15018	3026	Start Manufacture	2016.00
Hous	04:10:32	15032	3040	Cycle 1 Complete.	2007.80
Hous	04:10:47	15047	3055	Cycle 2 Complete.	1999.60
Hous	04:11:00	15060	3068	Cycle 3 Complete.	1991.40
Hous	04:11:14	15074	3082	Cycle 4 Complete.	1983.20
Hous	04:11:28	15088	3096	Cycle 5 Complete.	1975.00

Hous	04:11:43	15103	3111	Cycle 6 Complete.	1966.80
Hous	04:11:57	15117	3125	Cycle 7 Complete.	1958.60
Hous	04:12:11	15131	3139	Cycle 8 Complete.	1950.40
Hous	04:12:24	15144	3152	Cycle 9 Complete.	1942.20
Hous	04:12:39	15159	3167	Whole job Completed	1934.00
Bearig	04:12:43	15163	3171	Start Manufacture	1934.00
Bearig	04:13:34	15214	3222	Cycle 1 Complete.	1894.88
Bearig	04:14:25	15265	3273	Cycle 2 Complete.	1855.75
Bearig	04:15:17	15317	3325	Cycle 3 Complete.	1816.63
Bearig	04:16:08	15368	3376	Cycle 4 Complete.	1777.50
Bearig	04:16:58	15418	3426	Cycle 5 Complete.	1738.38
Bearig	04:17:49	15469	3477	Cycle 6 Complete.	1699.25
Jourss	04:18:11	15491	3499	Add	2049.25
Bearig	04:18:40	15520	3528	Cycle 7 Complete.	2010.13
Bearig	04:19:31	15571	3579	Whole job Completed	1971.00
Valves	04:19:37	15577	3585	Start Manufacture	1971.00
Valves	04:19:59	15599	3607	Cycle 1 Complete.	1954.33
Valves	04:20:21	15621	3629	Cycle 2 Complete.	1937.67
Valves	04:20:43	15643	3651	Bottom Queue Add	1921.00
Bracs	04:20:48	15648	3656	Start Manufacture	1921.00
Bracs	04:21:55	15715	3723	Cycle 1 Complete.	1865.33
Bracs	04:23:04	15784	3792	Cycle 2 Complete.	1809.67
Bracs	04:24:13	15853	3861	Cycle 3 Complete.	1754.00
Bracs	04:25:20	15920	3928	Cycle 4 Complete.	1698.33
Bracs	04:26:28	15988	3996	Cycle 5 Complete.	1642.67
Bracs	04:27:36	16056	4064	Bottom Queue Add	1587.00
axless	04:27:41	16061	4069	Start Manufacture	1587.00
axless	04:30:48	16248	4256	Cycle 1 Complete.	1439.83
axless	04:33:55	16435	4443	Cycle 2 Complete.	1292.67
axless	04:37:03	16623	4631	Cycle 3 Complete.	1145.50
axless	04:40:09	16809	4817	Cycle 4 Complete.	998.33
axless	04:43:16	16996	5004	Cycle 5 Complete.	851.17
axless	04:46:24	17184	5192	Whole job Completed	704.00
Pinss	04:46:28	17188	5196	Start Manufacture	704.00
Pinss	04:47:21	17241	5249	Cycle 1 Complete.	664.67
Pinss	04:48:15	17295	5303	Cycle 2 Complete.	625.33
Pinss	04:49:09	17349	5357	Cycle 3 Complete.	586.00
Pinss	04:50:03	17403	5411	Cycle 4 Complete.	546.67
Pinss	04:50:57	17457	5465	Cycle 5 Complete.	507.33
Pinss	04:51:51	17511	5519	Cycle 6 Complete.	468.00
Pinss	04:52:45	17565	5573	Cycle 7 Complete.	428.67
Pinss	04:53:40	17620	5628	Cycle 8 Complete.	389.33
Pinss	04:54:33	17673	5681	Whole job Completed	350.00
Jourss	04:54:38	17678	5686	Start Manufacture	350.00
Jourss	04:55:26	17726	5734	Cycle 1 Complete.	315.00
Jourss	04:56:14	17774	5782	Cycle 2 Complete.	280.00
Jourss	04:57:00	17820	5828	Cycle 3 Complete.	245.00
Jourss	04:57:48	17868	5876	Cycle 4 Complete.	210.00
Jourss	04:58:36	17916	5924	Cycle 5 Complete.	175.00
Jourss	04:59:25	17965	5973	Cycle 6 Complete.	140.00
Jourss	05:00:13	18013	6021	Cycle 7 Complete.	105.00
Jourss	05:01:00	18060	6068	Cycle 8 Complete.	70.00

Jourss	05:01:49	18109	6117	Cycle 9 Complete.	35.00
Jourss	05:02:37	18157	6165	Bottom Queue Add	0.00

VMD2 Cell 3(1.7)

Jobname	Actual Time(Clock)	Relative Time	TimeReal(Secs)	Event	Queuecost
Start Simulation	03:19:55	11995	0	Simulation Start	0.00
Pin	03:29:16	12556	561	Add	240.00
Pin	03:29:18	12558	563	Start Manufacture	240.00
Pin	03:30:11	12611	616	Cycle 1 Complete.	200.00
Brack	03:30:35	12635	640	Add	642.00
Pin	03:31:05	12665	670	Cycle 2 Complete.	602.00
Pin	03:31:59	12719	724	Cycle 3 Complete.	562.00
Pin	03:32:52	12772	777	Cycle 4 Complete.	522.00
Pin	03:33:47	12827	832	Cycle 5 Complete.	482.00
Pin	03:34:40	12880	885	Whole job Completed	442.00
Brack	03:34:46	12886	891	Start Manufacture	442.00
Valve	03:35:19	12919	924	Add	508.00
Brack	03:35:52	12952	957	Cycle 1 Complete.	452.75
Journ	03:36:47	13007	1012	Add	734.75
Brack	03:37:00	13020	1025	Cycle 2 Complete.	679.50
Brack	03:38:09	13089	1094	Cycle 3 Complete.	624.25
Brack	03:39:16	13156	1161	Cycle 4 Complete.	569.00
Brack	03:40:24	13224	1229	Cycle 5 Complete.	513.75
Houses	03:40:27	13227	1232	Add	547.75
Brack	03:41:33	13293	1298	Cycle 6 Complete.	492.50
Brack	03:42:40	13360	1365	Cycle 7 Complete.	437.25
Brack	03:43:48	13428	1433	Bottom Queue Add	382.00
Valve	03:43:53	13433	1438	Start Manufacture	382.00
Valve	03:44:15	13455	1460	Cycle 1 Complete.	365.50
Valve	03:44:36	13476	1481	Cycle 2 Complete.	349.00
Mounts	03:44:37	13477	1482	Add	746.00
Valve	03:44:58	13498	1503	Cycle 3 Complete.	729.50
Valve	03:45:20	13520	1525	Bottom Queue Add	713.00
Journ	03:45:25	13525	1530	Start Manufacture	713.00
Journ	03:46:13	13573	1578	Cycle 1 Complete.	677.75
Journ	03:47:01	13621	1626	Cycle 2 Complete.	642.50
Journ	03:47:48	13668	1673	Cycle 3 Complete.	607.25
Journs	03:48:31	13711	1716	Add	651.25
Journ	03:48:36	13716	1721	Cycle 4 Complete.	616.00
Keys	03:49:08	13748	1753	Add	908.00
Journ	03:49:25	13765	1770	Cycle 5 Complete.	872.75
Journ	03:50:12	13812	1817	Cycle 6 Complete.	837.50
Journ	03:51:01	13861	1866	Cycle 7 Complete.	802.25
Journ	03:51:48	13908	1913	Bottom Queue Add	767.00
Houses	03:51:52	13912	1917	Start Manufacture	767.00
Houses	03:52:06	13926	1931	Cycle 1 Complete.	758.50
Houses	03:52:20	13940	1945	Cycle 2 Complete.	750.00
Houses	03:52:34	13954	1959	Cycle 3 Complete.	741.50
Houses	03:52:48	13968	1973	Whole job Completed	733.00
Mounts	03:52:53	13973	1978	Start Manufacture	733.00

Mounts	03:54:04	14044	2049	Cycle 1 Complete.	666.83
Mounts	03:55:14	14114	2119	Cycle 2 Complete.	600.67
Mounts	03:56:26	14186	2191	Cycle 3 Complete.	534.50
bushs	03:56:37	14197	2202	Add	1057.50
Mounts	03:57:36	14256	2261	Cycle 4 Complete.	991.33
Mounts	03:58:48	14328	2333	Cycle 5 Complete.	925.17
Mounts	03:59:58	14398	2403	Bottom Queue Add	859.00
Journs	04:00:03	14403	2408	Start Manufacture	859.00
Journs	04:00:51	14451	2456	Bottom Queue Add	815.00
Keys	04:00:56	14456	2461	Start Manufacture	815.00
Keys	04:01:45	14505	2510	Cycle 1 Complete.	773.29
Mouns	04:01:55	14515	2520	Add	972.29
Keys	04:02:35	14555	2560	Cycle 2 Complete.	930.57
Keys	04:03:26	14606	2611	Cycle 3 Complete.	888.86
Shaf	04:03:40	14620	2625	Add	999.86
Keys	04:04:15	14655	2660	Cycle 4 Complete.	958.14
Keys	04:05:05	14705	2710	Cycle 5 Complete.	916.43
Rotor	04:05:14	14714	2719	Add	1199.43
Keys	04:05:55	14755	2760	Cycle 6 Complete.	1157.71
Keyss	04:06:11	14771	2776	Add	1572.71
Stubss	04:06:35	14795	2800	Add	1923.71
Keys	04:06:45	14805	2810	Bottom Queue Add	1882.00
bushs	04:06:50	14810	2815	Start Manufacture	1882.00
bushs	04:07:49	14869	2874	Cycle 1 Complete.	1829.70
bushs	04:08:49	14929	2934	Cycle 2 Complete.	1777.40
Bearin	04:09:14	14954	2959	Add	2014.40
bushs	04:09:47	14987	2992	Cycle 3 Complete.	1962.10
Valvs	04:10:05	15005	3010	Add	2060.10
bushs	04:10:46	15046	3051	Cycle 4 Complete.	2007.80
bushs	04:11:45	15105	3110	Cycle 5 Complete.	1955.50
bushs	04:12:44	15164	3169	Cycle 6 Complete.	1903.20
bushs	04:13:43	15223	3228	Cycle 7 Complete.	1850.90
bushs	04:14:43	15283	3288	Cycle 8 Complete.	1798.60
bushs	04:15:41	15341	3346	Cycle 9 Complete.	1746.30
bushs	04:16:40	15400	3405	Whole job Completed	1694.00
Mouns	04:16:46	15406	3411	Start Manufacture	1694.00
Pistos	04:17:56	15476	3481	Add	1777.00
Mouns	04:17:56	15476	3481	Cycle 1 Complete.	1710.67
Mouns	04:19:07	15547	3552	Cycle 2 Complete.	1644.33
Rotors	04:19:44	15584	3589	Add	1865.33
Mouns	04:20:18	15618	3623	Bottom Queue Add	1799.00
Shaf	04:20:24	15624	3629	Start Manufacture	1799.00
Shaf	04:20:44	15644	3649	Cycle 1 Complete.	1786.67
Shaf	04:21:03	15663	3668	Cycle 2 Complete.	1774.33
Shaf	04:21:21	15681	3686	Cycle 3 Complete.	1762.00
Shaf	04:21:41	15701	3706	Cycle 4 Complete.	1749.67
Shaf	04:21:58	15718	3723	Cycle 5 Complete.	1737.33
Pistss	04:22:09	15729	3734	Add	1766.33
Shaf	04:22:18	15738	3743	Cycle 6 Complete.	1754.00
Shaf	04:22:38	15758	3763	Cycle 7 Complete.	1741.67
Shaf	04:22:56	15776	3781	Cycle 8 Complete.	1729.33
Shaf	04:23:15	15795	3800	Bottom Queue Add	1717.00

Rotor	04:23:20	15800	3805	Start Manufacture	1717.00
Bears	04:23:22	15802	3807	Add	1840.00
Rotor	04:23:58	15838	3843	Cycle 1 Complete.	1808.56
Rotor	04:24:37	15877	3882	Cycle 2 Complete.	1777.11
Rotor	04:25:17	15917	3922	Cycle 3 Complete.	1745.67
Rotor	04:25:56	15956	3961	Cycle 4 Complete.	1714.22
Rotor	04:26:35	15995	4000	Cycle 5 Complete.	1682.78
Rotor	04:27:15	16035	4040	Cycle 6 Complete.	1651.33
Rotor	04:27:54	16074	4079	Cycle 7 Complete.	1619.89
Rotor	04:28:33	16113	4118	Cycle 8 Complete.	1588.44
Rotor	04:29:12	16152	4157	Whole job Completed	1557.00
Keyss	04:29:17	16157	4162	Start Manufacture	1557.00
Keyss	04:30:08	16208	4213	Cycle 1 Complete.	1515.50
Keyss	04:30:58	16258	4263	Cycle 2 Complete.	1474.00
Keyss	04:31:49	16309	4314	Cycle 3 Complete.	1432.50
Keyss	04:32:38	16358	4363	Cycle 4 Complete.	1391.00
Keyss	04:33:28	16408	4413	Cycle 5 Complete.	1349.50
Rotos	04:33:29	16409	4414	Add	1570.50
Keyss	04:34:18	16458	4463	Cycle 6 Complete.	1529.00
Keyss	04:35:08	16508	4513	Cycle 7 Complete.	1487.50
Keyss	04:35:58	16558	4563	Cycle 8 Complete.	1446.00
Keyss	04:36:47	16607	4612	Cycle 9 Complete.	1404.50
Keyss	04:37:38	16658	4663	Bottom Queue Add	1363.00
Stubss	04:37:43	16663	4668	Start Manufacture	1363.00
Stubss	04:38:37	16717	4722	Cycle 1 Complete.	1312.86
Stubss	04:39:33	16773	4778	Cycle 2 Complete.	1262.71
Stubss	04:40:28	16828	4833	Cycle 3 Complete.	1212.57
Stubss	04:41:23	16883	4888	Cycle 4 Complete.	1162.43
Stubss	04:42:18	16938	4943	Cycle 5 Complete.	1112.29
Stubss	04:43:12	16992	4997	Cycle 6 Complete.	1062.14
Stubss	04:44:08	17048	5053	Whole job Completed	1012.00
Bearin	04:44:13	17053	5058	Start Manufacture	1012.00
Bearin	04:45:04	17104	5109	Cycle 1 Complete.	972.50
Bearin	04:45:55	17155	5160	Cycle 2 Complete.	933.00
Bearin	04:46:46	17206	5211	Cycle 3 Complete.	893.50
Bearin	04:47:38	17258	5263	Cycle 4 Complete.	854.00
Bearin	04:48:28	17308	5313	Cycle 5 Complete.	814.50
Bearin	04:49:18	17358	5363	Whole job Completed	775.00
Valvs	04:49:24	17364	5369	Start Manufacture	775.00
Valvs	04:49:46	17386	5391	Cycle 1 Complete.	758.67
Valvs	04:50:08	17408	5413	Cycle 2 Complete.	742.33
Valvs	04:50:30	17430	5435	Cycle 3 Complete.	726.00
Valvs	04:50:52	17452	5457	Cycle 4 Complete.	709.67
Valvs	04:51:14	17474	5479	Cycle 5 Complete.	693.33
Valvs	04:51:36	17496	5501	Bottom Queue Add	677.00
Pistos	04:51:41	17501	5506	Start Manufacture	677.00
Pistos	04:52:14	17534	5539	Cycle 1 Complete.	649.33
Pistos	04:52:47	17567	5572	Cycle 2 Complete.	621.67
Pistos	04:53:20	17600	5605	Whole job Completed	594.00
Rotors	04:53:25	17605	5610	Start Manufacture	594.00
Rotors	04:54:03	17643	5648	Cycle 1 Complete.	562.43
Rotors	04:54:43	17683	5688	Cycle 2 Complete.	530.86

Rotors	04:55:21	17721	5726	Cycle 3 Complete.	499.29
Rotors	04:56:00	17760	5765	Cycle 4 Complete.	467.71
Rotors	04:56:39	17799	5804	Cycle 5 Complete.	436.14
Rotors	04:57:18	17838	5843	Cycle 6 Complete.	404.57
Rotors	04:57:57	17877	5882	Whole job Completed	373.00
Pistss	04:58:02	17882	5887	Start Manufacture	373.00
Pistss	04:58:34	17914	5919	Whole job Completed	344.00
Bears	04:58:39	17919	5924	Start Manufacture	344.00
Bears	04:59:30	17970	5975	Cycle 1 Complete.	303.00
Bears	05:00:21	18021	6026	Cycle 2 Complete.	262.00
Bears	05:01:12	18072	6077	Whole job Completed	221.00
Rotos	05:01:17	18077	6082	Start Manufacture	221.00
Rotos	05:01:55	18115	6120	Cycle 1 Complete.	189.43
Rotos	05:02:33	18153	6158	Cycle 2 Complete.	157.86
Rotos	05:03:13	18193	6198	Cycle 3 Complete.	126.29
Rotos	05:03:52	18232	6237	Cycle 4 Complete.	94.71
Rotos	05:04:31	18271	6276	Cycle 5 Complete.	63.14
Rotos	05:05:10	18310	6315	Cycle 6 Complete.	31.57
Rotos	05:05:48	18348	6353	Whole job Completed	0.00

D.3 Experiment 3 - Raw Data

Cell Controller Data Collection - NC Milling Equipment, Experiment 3

VMD4 Cell 1(1.8)

Jobname	Actual Time(Clock)	Relative Time	TimeReal(Secs)	Event	Queuecost
Start Simulation	03:09:46	11386	0	Simulation Start	0.0
Brack	03:20:26	12026	640	Add	442.0
Brack	03:20:29	12029	643	Start Manufacture	442.0
Brack	03:21:37	12097	711	Cycle 1 Complete.	386.8
Brack	03:22:45	12165	779	Cycle 2 Complete.	331.5
Brack	03:23:52	12232	846	Cycle 3 Complete.	276.3
Brack	03:25:00	12300	914	Cycle 4 Complete.	221.0
Valve	03:25:18	12318	932	Add	287.0
Brack	03:26:09	12369	983	Cycle 5 Complete.	231.8
Shafts	03:26:15	12375	989	Add	354.8
Brack	03:27:17	12437	1051	Cycle 6 Complete.	299.5
Brack	03:28:25	12505	1119	Cycle 7 Complete.	244.3
Bushes	03:29:17	12557	1171	Add	767.3
Brack	03:29:32	12572	1186	Bottom Queue Add	712.0
Valve	03:29:38	12578	1192	Start Manufacture	712.0
Valve	03:30:00	12600	1214	Cycle 1 Complete.	695.5
Valve	03:30:22	12622	1236	Cycle 2 Complete.	679.0
Valve	03:30:44	12644	1258	Cycle 3 Complete.	662.5
Valve	03:31:05	12665	1279	Bottom Queue Add	646.0
Shafts	03:31:10	12670	1284	Start Manufacture	646.0
Shafts	03:31:28	12688	1302	Cycle 1 Complete.	633.7
Shafts	03:31:47	12707	1321	Cycle 2 Complete.	621.4
Shafts	03:32:05	12725	1339	Cycle 3 Complete.	609.1
Shafts	03:32:25	12745	1359	Cycle 4 Complete.	596.8
Shafts	03:32:44	12764	1378	Cycle 5 Complete.	584.5
Shafts	03:33:03	12783	1397	Cycle 6 Complete.	572.2
Shafts	03:33:23	12803	1417	Cycle 7 Complete.	559.9
Shafts	03:33:42	12822	1436	Cycle 8 Complete.	547.6
Shafts	03:34:00	12840	1454	Cycle 9 Complete.	535.3
Shafts	03:34:19	12859	1473	Bottom Queue Add	523.0
Bushes	03:34:24	12864	1478	Start Manufacture	523.0
Bushes	03:35:23	12923	1537	Cycle 1 Complete.	470.7
Bushes	03:36:21	12981	1595	Cycle 2 Complete.	418.4
Bushes	03:37:20	13040	1654	Cycle 3 Complete.	366.1
Bushes	03:38:19	13099	1713	Cycle 4 Complete.	313.8
Keys	03:38:32	13112	1726	Add	605.8
Bushes	03:39:18	13158	1772	Cycle 5 Complete.	553.5
Bushes	03:40:18	13218	1832	Cycle 6 Complete.	501.2

Bushes	03:41:16	13276	1890	Cycle 7 Complete.	448.9
Valves	03:41:48	13308	1922	Add	498.9
Bushes	03:42:16	13336	1950	Cycle 8 Complete.	446.6
Bushes	03:43:15	13395	2009	Cycle 9 Complete.	394.3
Journs	03:43:56	13436	2050	Add	438.3
Bushes	03:44:13	13453	2067	Whole job Completed	386.0
Keys	03:44:17	13457	2071	Start Manufacture	386.0
Piston	03:44:37	13477	2091	Add	604.0
Keys	03:45:07	13507	2121	Cycle 1 Complete.	562.3
Keys	03:45:57	13557	2171	Cycle 2 Complete.	520.6
Keys	03:46:48	13608	2222	Cycle 3 Complete.	478.9
Keys	03:47:37	13657	2271	Cycle 4 Complete.	437.1
Keys	03:48:27	13707	2321	Cycle 5 Complete.	395.4
Keys	03:49:17	13757	2371	Cycle 6 Complete.	353.7
Bearig	03:50:01	13801	2415	Add	666.7
Keys	03:50:06	13806	2420	Bottom Queue Add	625.0
Valves	03:50:12	13812	2426	Start Manufacture	625.0
Valves	03:50:33	13833	2447	Cycle 1 Complete.	608.3
Valves	03:50:55	13855	2469	Cycle 2 Complete.	591.7
Valves	03:51:17	13877	2491	Bottom Queue Add	575.0
Journs	03:51:22	13882	2496	Start Manufacture	575.0
Journs	03:52:09	13929	2543	Bottom Queue Add	531.0
Piston	03:52:14	13934	2548	Start Manufacture	531.0
Piston	03:52:47	13967	2581	Cycle 1 Complete.	503.8
Bracs	03:53:21	14001	2615	Add	837.8
Piston	03:53:21	14001	2615	Cycle 2 Complete.	810.5
Piston	03:53:53	14033	2647	Cycle 3 Complete.	783.3
Piston	03:54:27	14067	2681	Cycle 4 Complete.	756.0
Piston	03:54:59	14099	2713	Cycle 5 Complete.	728.8
Piston	03:55:33	14133	2747	Cycle 6 Complete.	701.5
Pinss	03:55:43	14143	2757	Add	1055.5
Piston	03:56:06	14166	2780	Cycle 7 Complete.	1028.3
Piston	03:56:39	14199	2813	Whole job Completed	1001.0
Bearig	03:56:43	14203	2817	Start Manufacture	1001.0
Bearig	03:57:34	14254	2868	Cycle 1 Complete.	961.9
Bearig	03:58:25	14305	2919	Cycle 2 Complete.	922.8
Bearig	03:59:15	14355	2969	Cycle 3 Complete.	883.6
Bearig	04:00:07	14407	3021	Cycle 4 Complete.	844.5
Bearig	04:00:58	14458	3072	Cycle 5 Complete.	805.4
Bearig	04:01:50	14510	3124	Cycle 6 Complete.	766.3
Bearig	04:02:41	14561	3175	Cycle 7 Complete.	727.1
Bearig	04:03:31	14611	3225	Whole job Completed	688.0
Bracs	04:03:36	14616	3230	Start Manufacture	688.0
Bracs	04:04:43	14683	3297	Cycle 1 Complete.	632.3
Bearin	04:05:04	14704	3318	Add	869.3
Bracs	04:05:51	14751	3365	Cycle 2 Complete.	813.7
Bracs	04:06:59	14819	3433	Cycle 3 Complete.	758.0
Bracs	04:08:07	14887	3501	Cycle 4 Complete.	702.3
Bracs	04:09:15	14955	3569	Cycle 5 Complete.	646.7

Bracs	04:10:23	15023	3637	Bottom Queue Add	591.0
Pinss	04:10:29	15029	3643	Start Manufacture	591.0
Pinss	04:11:22	15082	3696	Cycle 1 Complete.	551.7
Pinss	04:12:16	15136	3750	Cycle 2 Complete.	512.3
Pistss	04:13:10	15190	3804	Add	541.3
Pinss	04:13:10	15190	3804	Cycle 3 Complete.	502.0
Pinss	04:14:03	15243	3857	Cycle 4 Complete.	462.7
Pinss	04:14:58	15298	3912	Cycle 5 Complete.	423.3
Pinss	04:15:52	15352	3966	Cycle 6 Complete.	384.0
Pinss	04:16:46	15406	4020	Cycle 7 Complete.	344.7
Pinss	04:17:39	15459	4073	Cycle 8 Complete.	305.3
Pinss	04:18:34	15514	4128	Whole job Completed	266.0
Bearin	04:18:38	15518	4132	Start Manufacture	266.0
Rotos	04:18:42	15522	4136	Add	487.0
Bearin	04:19:29	15569	4183	Cycle 1 Complete.	447.5
Bearin	04:20:20	15620	4234	Cycle 2 Complete.	408.0
Bearin	04:21:11	15671	4285	Cycle 3 Complete.	368.5
Bearin	04:22:01	15721	4335	Cycle 4 Complete.	329.0
Bearin	04:22:53	15773	4387	Cycle 5 Complete.	289.5
Bearin	04:23:44	15824	4438	Whole job Completed	250.0
Pistss	04:23:49	15829	4443	Start Manufacture	250.0
Pistss	04:24:23	15863	4477	Whole job Completed	221.0
Rotos	04:24:27	15867	4481	Start Manufacture	221.0
Rotos	04:25:06	15906	4520	Cycle 1 Complete.	189.4
Rotos	04:25:44	15944	4558	Cycle 2 Complete.	157.9
Rotos	04:26:24	15984	4598	Cycle 3 Complete.	126.3
Rotos	04:27:03	16023	4637	Cycle 4 Complete.	94.7
Rotos	04:27:42	16062	4676	Cycle 5 Complete.	63.1
Rotos	04:28:21	16101	4715	Cycle 6 Complete.	31.6
Rotos	04:29:00	16140	4754	Whole job Completed	0.0

VMD4 Cell 2(1.7)

Jobname	Actual Time(Clock)	Relative Time	TimeReal(Secs)	Event	Queuecost
Start Simulation	03:09:44	11384	0	Simulation Start	0.00
Shaft	03:11:01	11461	77	Add	99.00
Shaft	03:11:04	11464	80	Start Manufacture	99.00
Shaft	03:11:22	11482	98	Cycle 1 Complete.	86.63
Shaft	03:11:41	11501	117	Cycle 2 Complete.	74.25
Shaft	03:12:01	11521	137	Cycle 3 Complete.	61.88
Shaft	03:12:20	11540	156	Cycle 4 Complete.	49.50
Shaft	03:12:38	11558	174	Cycle 5 Complete.	37.13
Shaft	03:12:57	11577	193	Cycle 6 Complete.	24.75
Shaft	03:13:17	11597	213	Cycle 7 Complete.	12.38
Shaft	03:13:35	11615	231	Bottom Queue Add	0.00
Bush	03:13:54	11634	250	Add	159.00
Bush	03:13:55	11635	251	Start Manufacture	159.00
Bush	03:14:54	11694	310	Cycle 1 Complete.	106.00

Bush	03:15:53	11753	369	Cycle 2 Complete.	53.00
Bush	03:16:52	11812	428	Whole job Completed	0.00
Axle	03:17:37	11857	473	Add	883.00
Axle	03:17:39	11859	475	Start Manufacture	883.00
Axle	03:20:47	12047	663	Cycle 1 Complete.	735.83
Axle	03:23:52	12232	848	Cycle 2 Complete.	588.67
Journ	03:26:58	12418	1034	Add	870.67
Axle	03:27:00	12420	1036	Cycle 3 Complete.	723.50
Axle	03:30:08	12608	1224	Cycle 4 Complete.	576.33
Houses	03:30:22	12622	1238	Add	610.33
Axle	03:33:13	12793	1409	Cycle 5 Complete.	463.17
Mounts	03:34:30	12870	1486	Add	860.17
Axle	03:36:20	12980	1596	Whole job Completed	713.00
Journ	03:36:25	12985	1601	Start Manufacture	713.00
Pins	03:36:53	13013	1629	Add	1067.00
Journ	03:37:12	13032	1648	Cycle 1 Complete.	1031.75
Journ	03:38:00	13080	1696	Cycle 2 Complete.	996.50
Journ	03:38:49	13129	1745	Cycle 3 Complete.	961.25
Journ	03:39:36	13176	1792	Cycle 4 Complete.	926.00
Journ	03:40:25	13225	1841	Cycle 5 Complete.	890.75
Journ	03:41:13	13273	1889	Cycle 6 Complete.	855.50
Journ	03:42:02	13322	1938	Cycle 7 Complete.	820.25
Journ	03:42:49	13369	1985	Bottom Queue Add	785.00
Houses	03:42:54	13374	1990	Start Manufacture	785.00
Houses	03:43:08	13388	2004	Cycle 1 Complete.	776.50
Houses	03:43:21	13401	2017	Cycle 2 Complete.	768.00
Houses	03:43:35	13415	2031	Cycle 3 Complete.	759.50
Houses	03:43:50	13430	2046	Whole job Completed	751.00
Mounts	03:43:54	13434	2050	Start Manufacture	751.00
Mounts	03:45:04	13504	2120	Cycle 1 Complete.	684.83
Mounts	03:46:16	13576	2192	Cycle 2 Complete.	618.67
Hous	03:47:18	13638	2254	Add	700.67
Mounts	03:47:26	13646	2262	Cycle 3 Complete.	634.50
Mounts	03:48:38	13718	2334	Cycle 4 Complete.	568.33
Stubss	03:49:24	13764	2380	Add	919.33
Mounts	03:49:49	13789	2405	Cycle 5 Complete.	853.17
Mounts	03:50:59	13859	2475	Bottom Queue Add	787.00
Pins	03:51:04	13864	2480	Start Manufacture	787.00
Rotor	03:51:16	13876	2492	Add	1070.00
Pins	03:51:58	13918	2534	Cycle 1 Complete.	1030.67
Pins	03:52:52	13972	2588	Cycle 2 Complete.	991.33
Pins	03:53:47	14027	2643	Cycle 3 Complete.	952.00
Pins	03:54:41	14081	2697	Cycle 4 Complete.	912.67
Pins	03:55:35	14135	2751	Cycle 5 Complete.	873.33
Pins	03:56:29	14189	2805	Cycle 6 Complete.	834.00
Pins	03:57:23	14243	2859	Cycle 7 Complete.	794.67
Pins	03:58:17	14297	2913	Cycle 8 Complete.	755.33
Valvs	03:59:06	14346	2962	Add	853.33

Pins	03:59:12	14352	2968	Whole job Completed	814.00
Hous	03:59:16	14356	2972	Start Manufacture	814.00
Hous	03:59:30	14370	2986	Cycle 1 Complete.	805.80
Hous	03:59:43	14383	2999	Cycle 2 Complete.	797.60
axless	03:59:51	14391	3007	Add	1680.60
Hous	03:59:58	14398	3014	Cycle 3 Complete.	1672.40
Hous	04:00:12	14412	3028	Cycle 4 Complete.	1664.20
Hous	04:00:25	14425	3041	Cycle 5 Complete.	1656.00
Hous	04:00:40	14440	3056	Cycle 6 Complete.	1647.80
Hous	04:00:54	14454	3070	Cycle 7 Complete.	1639.60
Hous	04:01:09	14469	3085	Cycle 8 Complete.	1631.40
Hous	04:01:22	14482	3098	Cycle 9 Complete.	1623.20
Hous	04:01:36	14496	3112	Whole job Completed	1615.00
Stubss	04:01:41	14501	3117	Start Manufacture	1615.00
Stubss	04:02:37	14557	3173	Cycle 1 Complete.	1564.86
Stubss	04:03:32	14612	3228	Cycle 2 Complete.	1514.71
Stubss	04:04:27	14667	3283	Cycle 3 Complete.	1464.57
Stubss	04:05:21	14721	3337	Cycle 4 Complete.	1414.43
Stubss	04:06:16	14776	3392	Cycle 5 Complete.	1364.29
Stubss	04:07:11	14831	3447	Cycle 6 Complete.	1314.14
Stubss	04:08:07	14887	3503	Whole job Completed	1264.00
Rotor	04:08:12	14892	3508	Start Manufacture	1264.00
Rotor	04:08:51	14931	3547	Cycle 1 Complete.	1232.56
Rotor	04:09:29	14969	3585	Cycle 2 Complete.	1201.11
Rotor	04:10:08	15008	3624	Cycle 3 Complete.	1169.67
Rotor	04:10:48	15048	3664	Cycle 4 Complete.	1138.22
Rotor	04:11:27	15087	3703	Cycle 5 Complete.	1106.78
Rotor	04:12:05	15125	3741	Cycle 6 Complete.	1075.33
Rotor	04:12:46	15166	3782	Cycle 7 Complete.	1043.89
Rotor	04:13:24	15204	3820	Cycle 8 Complete.	1012.44
Rotor	04:14:02	15242	3858	Whole job Completed	981.00
Valvs	04:14:07	15247	3863	Start Manufacture	981.00
Valvs	04:14:28	15268	3884	Cycle 1 Complete.	964.67
Valvs	04:14:50	15290	3906	Cycle 2 Complete.	948.33
Valvs	04:15:12	15312	3928	Cycle 3 Complete.	932.00
Valvs	04:15:34	15334	3950	Cycle 4 Complete.	915.67
Valvs	04:15:57	15357	3973	Cycle 5 Complete.	899.33
Valvs	04:16:18	15378	3994	Bottom Queue Add	883.00
axless	04:16:24	15384	4000	Start Manufacture	883.00
axless	04:19:29	15569	4185	Cycle 1 Complete.	735.83
axless	04:22:36	15756	4372	Cycle 2 Complete.	588.67
axless	04:25:43	15943	4559	Cycle 3 Complete.	441.50
axless	04:28:51	16131	4747	Cycle 4 Complete.	294.33
axless	04:31:58	16318	4934	Cycle 5 Complete.	147.17
axless	04:35:05	16505	5121	Whole job Completed	0.00

VMD2 Cell 3(1.7)

Jobname	Actual Time(Clock)	Relative Time	TimeReal(Secs)	Event	Queuecost
Start Simulation	03:09:46	11386	0	Simulation Start	0.00
House	03:15:06	11706	320	Add	34.00
House	03:15:08	11708	322	Start Manufacture	34.00
House	03:15:21	11721	335	Cycle 1 Complete.	25.50
House	03:15:36	11736	350	Cycle 2 Complete.	17.00
House	03:15:49	11749	363	Cycle 3 Complete.	8.50
House	03:16:04	11764	378	Whole job Completed	0.00
Stub	03:16:58	11818	432	Add	51.00
Stub	03:16:59	11819	433	Start Manufacture	51.00
Stub	03:17:53	11873	487	Whole job Completed	0.00
Pin	03:18:38	11918	532	Add	240.00
Pin	03:18:40	11920	534	Start Manufacture	240.00
Mount	03:19:16	11956	570	Add	835.00
Pin	03:19:34	11974	588	Cycle 1 Complete.	795.00
Pin	03:20:27	12027	641	Cycle 2 Complete.	755.00
Pin	03:21:22	12082	696	Cycle 3 Complete.	715.00
Pin	03:22:15	12135	749	Cycle 4 Complete.	675.00
Key	03:22:30	12150	764	Add	885.00
Pin	03:23:09	12189	803	Cycle 5 Complete.	845.00
Pin	03:24:03	12243	857	Whole job Completed	805.00
Mount	03:24:09	12249	863	Start Manufacture	805.00
Mount	03:25:20	12320	934	Cycle 1 Complete.	738.89
Mount	03:26:31	12391	1005	Cycle 2 Complete.	672.78
Mount	03:27:41	12461	1075	Cycle 3 Complete.	606.67
Mount	03:28:52	12532	1146	Cycle 4 Complete.	540.56
Axles	03:30:00	12600	1214	Add	723.56
Mount	03:30:03	12603	1217	Cycle 5 Complete.	657.44
Mount	03:31:15	12675	1289	Cycle 6 Complete.	591.33
Mount	03:32:26	12746	1360	Cycle 7 Complete.	525.22
Stubs	03:32:42	12762	1376	Add	926.22
Mount	03:33:36	12816	1430	Cycle 8 Complete.	860.11
Mount	03:34:47	12887	1501	Bottom Queue Add	794.00
Key	03:34:52	12892	1506	Start Manufacture	794.00
Key	03:35:41	12941	1555	Cycle 1 Complete.	752.00
Bracks	03:36:27	12987	1601	Add	1032.00
Key	03:36:32	12992	1606	Cycle 2 Complete.	990.00
Key	03:37:21	13041	1655	Cycle 3 Complete.	948.00
Key	03:38:11	13091	1705	Cycle 4 Complete.	906.00
Key	03:39:02	13142	1756	Bottom Queue Add	864.00
Axles	03:39:07	13147	1761	Start Manufacture	864.00
Axles	03:42:13	13333	1947	Whole job Completed	681.00
Stubs	03:42:19	13339	1953	Start Manufacture	681.00
Shaf	03:42:56	13376	1990	Add	792.00
Stubs	03:43:14	13394	2008	Cycle 1 Complete.	741.88
Stubs	03:44:09	13449	2063	Cycle 2 Complete.	691.75

Stubs	03:45:04	13504	2118	Cycle 3 Complete.	641.63
Stubs	03:45:59	13559	2173	Cycle 4 Complete.	591.50
bushs	03:46:26	13586	2200	Add	1114.50
Stubs	03:46:54	13614	2228	Cycle 5 Complete.	1064.38
Stubs	03:47:49	13669	2283	Cycle 6 Complete.	1014.25
Stubs	03:48:44	13724	2338	Cycle 7 Complete.	964.13
Stubs	03:49:39	13779	2393	Whole job Completed	914.00
Bracks	03:49:44	13784	2398	Start Manufacture	914.00
Bracks	03:50:52	13852	2466	Cycle 1 Complete.	858.00
Bracks	03:52:01	13921	2535	Cycle 2 Complete.	802.00
Mouns	03:52:12	13932	2546	Add	1001.00
Bracks	03:53:08	13988	2602	Cycle 3 Complete.	945.00
Bracks	03:54:16	14056	2670	Cycle 4 Complete.	889.00
Bracks	03:55:24	14124	2738	Bottom Queue Add	833.00
Keyss	03:55:26	14126	2740	Add	1248.00
Shaf	03:55:29	14129	2743	Start Manufacture	1248.00
Shaf	03:55:48	14148	2762	Cycle 1 Complete.	1235.67
Shaf	03:56:07	14167	2781	Cycle 2 Complete.	1223.33
Shaf	03:56:26	14186	2800	Cycle 3 Complete.	1211.00
Shaf	03:56:45	14205	2819	Cycle 4 Complete.	1198.67
Shaf	03:57:04	14224	2838	Cycle 5 Complete.	1186.33
Shaf	03:57:23	14243	2857	Cycle 6 Complete.	1174.00
Shaf	03:57:42	14262	2876	Cycle 7 Complete.	1161.67
Shaf	03:58:02	14282	2896	Cycle 8 Complete.	1149.33
Shaf	03:58:21	14301	2915	Bottom Queue Add	1137.00
bushs	03:58:25	14305	2919	Start Manufacture	1137.00
bushs	03:59:24	14364	2978	Cycle 1 Complete.	1084.70
bushs	04:00:24	14424	3038	Cycle 2 Complete.	1032.40
bushs	04:01:22	14482	3096	Cycle 3 Complete.	980.10
bushs	04:02:21	14541	3155	Cycle 4 Complete.	927.80
bushs	04:03:20	14600	3214	Cycle 5 Complete.	875.50
Rotors	04:03:49	14629	3243	Add	1096.50
bushs	04:04:19	14659	3273	Cycle 6 Complete.	1044.20
bushs	04:05:17	14717	3331	Cycle 7 Complete.	991.90
bushs	04:06:17	14777	3391	Cycle 8 Complete.	939.60
Pistos	04:06:45	14805	3419	Add	1022.60
bushs	04:07:17	14837	3451	Cycle 9 Complete.	970.30
bushs	04:08:15	14895	3509	Whole job Completed	918.00
Mouns	04:08:20	14900	3514	Start Manufacture	918.00
Mouns	04:09:32	14972	3586	Cycle 1 Complete.	851.67
Jourss	04:10:39	15039	3653	Add	1201.67
Mouns	04:10:42	15042	3656	Cycle 2 Complete.	1135.33
Mouns	04:11:52	15112	3726	Bottom Queue Add	1069.00
Keyss	04:11:57	15117	3731	Start Manufacture	1069.00
Keyss	04:12:47	15167	3781	Cycle 1 Complete.	1027.50
Keyss	04:13:36	15216	3830	Cycle 2 Complete.	986.00
Keyss	04:14:27	15267	3881	Cycle 3 Complete.	944.50
Keyss	04:15:17	15317	3931	Cycle 4 Complete.	903.00
Keyss	04:16:08	15368	3982	Cycle 5 Complete.	861.50
Keyss	04:16:57	15417	4031	Cycle 6 Complete.	820.00

Keyss	04:17:47	15467	4081	Cycle 7 Complete.	778.50
Keyss	04:18:37	15517	4131	Cycle 8 Complete.	737.00
Keyss	04:19:27	15567	4181	Cycle 9 Complete.	695.50
Keyss	04:20:16	15616	4230	Bottom Queue Add	654.00
Rotors	04:20:22	15622	4236	Start Manufacture	654.00
Rotors	04:21:00	15660	4274	Cycle 1 Complete.	622.43
Rotors	04:21:39	15699	4313	Cycle 2 Complete.	590.86
Rotors	04:22:18	15738	4352	Cycle 3 Complete.	559.29
Rotors	04:22:56	15776	4390	Cycle 4 Complete.	527.71
Rotors	04:23:36	15816	4430	Cycle 5 Complete.	496.14
Rotors	04:24:15	15855	4469	Cycle 6 Complete.	464.57
Rotors	04:24:54	15894	4508	Whole job Completed	433.00
Pistos	04:24:59	15899	4513	Start Manufacture	433.00
Pistos	04:25:32	15932	4546	Cycle 1 Complete.	405.33
Bears	04:25:51	15951	4565	Add	528.33
Pistos	04:26:06	15966	4580	Cycle 2 Complete.	500.67
Pistos	04:26:38	15998	4612	Whole job Completed	473.00
Jourss	04:26:43	16003	4617	Start Manufacture	473.00
Jourss	04:27:31	16051	4665	Cycle 1 Complete.	438.00
Jourss	04:28:18	16098	4712	Cycle 2 Complete.	403.00
Jourss	04:29:07	16147	4761	Cycle 3 Complete.	368.00
Jourss	04:29:55	16195	4809	Cycle 4 Complete.	333.00
Jourss	04:30:43	16243	4857	Cycle 5 Complete.	298.00
Jourss	04:31:32	16292	4906	Cycle 6 Complete.	263.00
Jourss	04:32:19	16339	4953	Cycle 7 Complete.	228.00
Jourss	04:33:06	16386	5000	Cycle 8 Complete.	193.00
Jourss	04:33:54	16434	5048	Cycle 9 Complete.	158.00
Jourss	04:34:42	16482	5096	Bottom Queue Add	123.00
Bears	04:34:47	16487	5101	Start Manufacture	123.00
Bears	04:35:37	16537	5151	Cycle 1 Complete.	82.00
Bears	04:36:29	16589	5203	Cycle 2 Complete.	41.00
Bears	04:37:19	16639	5253	Whole job Completed	0.00

D.4 Experiment 4 - Raw Data

Cell Controller Data Collection - NC Milling Equipment, Experiment 4

VMD4 Cell 1(1.8)

Jobname	Actual Time(Clock)	Relative Time	TimeReal(Secs)	Event	Queuecos
Start Simulation	22:59:52	82792	0	Simulation Start	0.0
Shaft	23:00:49	82849	57	Add	99.0
Shaft	23:00:50	82850	58	Start Manufacture	99.0
Shaft	23:01:08	82868	76	Cycle 1 Complete.	86.6
Shaft	23:01:28	82888	96	Cycle 2 Complete.	74.3
Shaft	23:01:47	82907	115	Cycle 3 Complete.	61.9
Shaft	23:02:06	82926	134	Cycle 4 Complete.	49.5
Shaft	23:02:25	82945	153	Cycle 5 Complete.	37.1
Shaft	23:02:28	82948	156	Cycle 6 Complete.	24.8
Shaft	23:02:47	82967	175	Cycle 7 Complete.	12.4
Shaft	23:03:06	82986	194	Bottom Queue Add	0.0
Bush	23:03:44	83024	232	Add	159.0
Bush	23:03:46	83026	234	Start Manufacture	159.0
Bush	23:04:44	83084	292	Cycle 1 Complete.	106.0
Bush	23:05:43	83143	351	Cycle 2 Complete.	53.0
Bush	23:06:43	83203	411	Whole job Completed	0.0
Pin	23:09:09	83349	557	Add	240.0
Pin	23:09:12	83352	560	Start Manufacture	240.0
Pin	23:10:05	83405	613	Cycle 1 Complete.	200.0
Pin	23:10:59	83459	667	Cycle 2 Complete.	160.0
Pin	23:11:54	83514	722	Cycle 3 Complete.	120.0
Key	23:12:18	83538	746	Add	330.0
Pin	23:12:47	83567	775	Cycle 4 Complete.	290.0
Pin	23:13:41	83621	829	Cycle 5 Complete.	250.0
Pin	23:14:36	83676	884	Whole job Completed	210.0
Key	23:14:41	83681	889	Start Manufacture	210.0
Valve	23:15:07	83707	915	Add	276.0
Key	23:15:31	83731	939	Cycle 1 Complete.	234.0
Shafts	23:16:03	83763	971	Add	357.0
Key	23:16:20	83780	988	Cycle 2 Complete.	315.0
Key	23:17:11	83831	1039	Cycle 3 Complete.	273.0
Key	23:18:01	83881	1089	Cycle 4 Complete.	231.0
Key	23:18:51	83931	1139	Bottom Queue Add	189.0
Valve	23:18:56	83936	1144	Start Manufacture	189.0
Valve	23:19:19	83959	1167	Cycle 1 Complete.	172.5
Valve	23:19:41	83981	1189	Cycle 2 Complete.	156.0
Axles	23:19:46	83986	1194	Add	339.0
Valve	23:20:03	84003	1211	Cycle 3 Complete.	322.5
Valve	23:20:25	84025	1233	Bottom Queue Add	306.0

Shafts	23:20:30	84030	1238	Start Manufacture	306.0
Shafts	23:20:49	84049	1257	Cycle 1 Complete.	293.7
Shafts	23:21:08	84068	1276	Cycle 2 Complete.	281.4
Shafts	23:21:27	84087	1295	Cycle 3 Complete.	269.1
Shafts	23:21:46	84106	1314	Cycle 4 Complete.	256.8
Shafts	23:22:05	84125	1333	Cycle 5 Complete.	244.5
Shafts	23:22:24	84144	1352	Cycle 6 Complete.	232.2
Shafts	23:22:44	84164	1372	Cycle 7 Complete.	219.9
Shafts	23:23:03	84183	1391	Cycle 8 Complete.	207.6
Shafts	23:23:21	84201	1409	Cycle 9 Complete.	195.3
Shafts	23:23:41	84221	1429	Bottom Queue Add	183.0
Axles	23:23:46	84226	1434	Start Manufacture	183.0
Axles	23:26:53	84413	1621	Whole job Completed	0.0
Pins	23:27:23	84443	1651	Add	354.0
Pins	23:27:25	84445	1653	Start Manufacture	354.0
Pins	23:28:18	84498	1706	Cycle 1 Complete.	314.7
Journs	23:28:51	84531	1739	Add	358.7
Pins	23:29:12	84552	1760	Cycle 2 Complete.	319.3
Pins	23:30:06	84606	1814	Cycle 3 Complete.	280.0
Pins	23:31:00	84660	1868	Cycle 4 Complete.	240.7
Valves	23:31:45	84705	1913	Add	290.7
Pins	23:31:55	84715	1923	Cycle 5 Complete.	251.3
Shaf	23:32:43	84763	1971	Add	362.3
Pins	23:32:49	84769	1977	Cycle 6 Complete.	323.0
Pins	23:33:43	84823	2031	Cycle 7 Complete.	283.7
Pins	23:34:36	84876	2084	Cycle 8 Complete.	244.3
Pins	23:35:30	84930	2138	Whole job Completed	205.0
Journs	23:35:35	84935	2143	Start Manufacture	205.0
Journs	23:36:22	84982	2190	Bottom Queue Add	161.0
Valves	23:36:27	84987	2195	Start Manufacture	161.0
Valves	23:36:49	85009	2217	Cycle 1 Complete.	144.3
Hous	23:37:05	85025	2233	Add	226.3
Valves	23:37:11	85031	2239	Cycle 2 Complete.	209.7
Valves	23:37:33	85053	2261	Bottom Queue Add	193.0
Shaf	23:37:38	85058	2266	Start Manufacture	193.0
Shaf	23:37:58	85078	2286	Cycle 1 Complete.	180.7
Shaf	23:38:16	85096	2304	Cycle 2 Complete.	168.3
Shaf	23:38:36	85116	2324	Cycle 3 Complete.	156.0
Shaf	23:38:54	85134	2342	Cycle 4 Complete.	143.7
Shaf	23:39:13	85153	2361	Cycle 5 Complete.	131.3
Shaf	23:39:32	85172	2380	Cycle 6 Complete.	119.0
Shaf	23:39:51	85191	2399	Cycle 7 Complete.	106.7
Shaf	23:40:10	85210	2418	Cycle 8 Complete.	94.3
Shaf	23:40:30	85230	2438	Bottom Queue Add	82.0
Hous	23:40:34	85234	2442	Start Manufacture	82.0
Hous	23:40:48	85248	2456	Cycle 1 Complete.	73.8
Hous	23:41:02	85262	2470	Cycle 2 Complete.	65.6
Hous	23:41:16	85276	2484	Cycle 3 Complete.	57.4
Hous	23:41:30	85290	2498	Cycle 4 Complete.	49.2
Hous	23:41:44	85304	2512	Cycle 5 Complete.	41.0
Hous	23:41:57	85317	2525	Cycle 6 Complete.	32.8
Mouns	23:42:07	85327	2535	Add	231.8

Hous	23:42:11	85331	2539	Cycle 7 Complete.	223.6
Hous	23:42:25	85345	2553	Cycle 8 Complete.	215.4
Hous	23:42:39	85359	2567	Cycle 9 Complete.	207.2
Hous	23:42:53	85373	2581	Whole job Completed	199.0
Mouns	23:42:58	85378	2586	Start Manufacture	199.0
Mouns	23:44:08	85448	2656	Cycle 1 Complete.	132.7
Mouns	23:45:20	85520	2728	Cycle 2 Complete.	66.3
Mouns	23:46:30	85590	2798	Bottom Queue Add	0.0
Pistos	23:49:08	85748	2956	Add	83.0
Pistos	23:49:11	85751	2959	Start Manufacture	83.0
Pistos	23:49:44	85784	2992	Cycle 1 Complete.	55.3
Valvs	23:50:00	85800	3008	Add	153.3
Pistos	23:50:16	85816	3024	Cycle 2 Complete.	125.7
Rotors	23:50:42	85842	3050	Add	346.7
Pistos	23:50:49	85849	3057	Whole job Completed	319.0
Valvs	23:50:54	85854	3062	Start Manufacture	319.0
Valvs	23:51:16	85876	3084	Cycle 1 Complete.	302.7
Valvs	23:51:38	85898	3106	Cycle 2 Complete.	286.3
Valvs	23:52:00	85920	3128	Cycle 3 Complete.	270.0
Valvs	23:52:22	85942	3150	Cycle 4 Complete.	253.7
Valvs	23:52:44	85964	3172	Cycle 5 Complete.	237.3
Valvs	23:53:06	85986	3194	Bottom Queue Add	221.0
Rotors	23:53:11	85991	3199	Start Manufacture	221.0
Rotors	23:53:50	86030	3238	Cycle 1 Complete.	189.4
Rotors	23:54:30	86070	3278	Cycle 2 Complete.	157.9
Rotors	23:55:09	86109	3317	Cycle 3 Complete.	126.3
Rotors	23:55:47	86147	3355	Cycle 4 Complete.	94.71
Rotors	23:56:27	86187	3395	Cycle 5 Complete.	63.14
Rotors	23:57:05	86225	3433	Cycle 6 Complete.	31.57
Jourss	23:57:33	86253	3461	Add	381.57
Rotors	23:57:44	86264	3472	Whole job Completed	350.00
Jourss	23:57:49	86269	3477	Start Manufacture	350.00
Jourss	23:58:37	86317	3525	Cycle 1 Complete.	315.00
Jourss	23:59:25	86365	3573	Cycle 2 Complete.	280.00
Jourss	00:00:14	86414	3622	Cycle 3 Complete.	245.00
Jourss	00:01:02	86462	3670	Cycle 4 Complete.	210.00
Jourss	00:01:51	86511	3719	Cycle 5 Complete.	175.00
Jourss	00:02:39	86559	3767	Cycle 6 Complete.	140.00
Jourss	00:03:27	86607	3815	Cycle 7 Complete.	105.00
Jourss	00:04:14	86654	3862	Cycle 8 Complete.	70.00
Jourss	00:05:02	86702	3910	Cycle 9 Complete.	35.00
Jourss	00:05:50	86750	3958	Bottom Queue Add	0.00
Bears	00:07:28	86848	4056	Add	123.00
Bears	00:07:31	86851	4059	Start Manufacture	123.00
Bears	00:08:21	86901	4109	Cycle 1 Complete.	82.00
Bears	00:09:13	86953	4161	Cycle 2 Complete.	41.00
Bears	00:10:04	87004	4212	Whole job Completed	0.00

VMD4 Cell 2(1.8)

Jobname	Actual Time(Clock)	Relative Time	TimeReal(Secs)	Event	Queuecost
Start Simulation	22:59:50	82790	0	Simulation Start	0.00
House	23:04:49	83089	299	Add	34.00
House	23:04:53	83093	303	Start Manufacture	34.00
House	23:05:06	83106	316	Cycle 1 Complete.	25.50
House	23:05:20	83120	330	Cycle 2 Complete.	17.00
House	23:05:34	83134	344	Cycle 3 Complete.	8.50
House	23:05:48	83148	358	Whole job Completed	0.00
Stub	23:06:43	83203	413	Add	51.00
Stub	23:06:46	83206	416	Start Manufacture	51.00
Stub	23:07:40	83260	470	Whole job Completed	0.00
Axle	23:08:11	83291	501	Add	883.00
Axle	23:08:13	83293	503	Start Manufacture	883.00
Brack	23:10:23	83423	633	Add	1325.00
Axle	23:11:19	83479	689	Cycle 1 Complete.	1177.83
Axle	23:14:27	83667	877	Cycle 2 Complete.	1030.67
Axle	23:17:33	83853	1063	Cycle 3 Complete.	883.50
Bushes	23:19:01	83941	1151	Add	1406.50
Axle	23:20:40	84040	1250	Cycle 4 Complete.	1259.33
Axle	23:23:47	84227	1437	Cycle 5 Complete.	1112.17
Mounts	23:24:45	84285	1495	Add	1509.17
Axle	23:26:55	84415	1625	Whole job Completed	1362.00
Brack	23:26:59	84419	1629	Start Manufacture	1362.00
Brack	23:28:07	84487	1697	Cycle 1 Complete.	1306.75
Keys	23:28:28	84508	1718	Add	1598.75
Brack	23:29:14	84554	1764	Cycle 2 Complete.	1543.50
Brack	23:30:22	84622	1832	Cycle 3 Complete.	1488.25
Brack	23:31:31	84691	1901	Cycle 4 Complete.	1433.00
Brack	23:32:38	84758	1968	Cycle 5 Complete.	1377.75
Brack	23:33:46	84826	2036	Cycle 6 Complete.	1322.50
Brack	23:34:54	84894	2104	Cycle 7 Complete.	1267.25
bushs	23:35:53	84953	2163	Add	1790.25
Brack	23:36:02	84962	2172	Bottom Queue Add	1735.00
Bushes	23:36:07	84967	2177	Start Manufacture	1735.00
Bushes	23:37:05	85025	2235	Cycle 1 Complete.	1682.70
Bushes	23:38:04	85084	2294	Cycle 2 Complete.	1630.40
Bushes	23:39:03	85143	2353	Cycle 3 Complete.	1578.10
Bearig	23:39:24	85164	2374	Add	1891.10
Stubss	23:39:47	85187	2397	Add	2242.10
Bushes	23:40:02	85202	2412	Cycle 4 Complete.	2189.80
Bushes	23:41:01	85261	2471	Cycle 5 Complete.	2137.50
Bushes	23:42:00	85320	2530	Cycle 6 Complete.	2085.20
Bushes	23:42:59	85379	2589	Cycle 7 Complete.	2032.90
Pinss	23:43:58	85438	2648	Add	2386.90
Bushes	23:43:59	85439	2649	Cycle 8 Complete.	2334.60
Bushes	23:44:57	85497	2707	Cycle 9 Complete.	2282.30
Keyss	23:45:47	85547	2757	Add	2697.30
Bushes	23:45:57	85557	2767	Whole job Completed	2645.00
Mounts	23:46:01	85561	2771	Start Manufacture	2645.00

axless	23:46:44	85604	2814	Add	3528.00
Mounts	23:47:12	85632	2842	Cycle 1 Complete.	3461.83
Mounts	23:48:23	85703	2913	Cycle 2 Complete.	3395.67
Mounts	23:49:35	85775	2985	Cycle 3 Complete.	3329.50
Mounts	23:50:46	85846	3056	Cycle 4 Complete.	3263.33
Mounts	23:51:56	85916	3126	Cycle 5 Complete.	3197.17
Mounts	23:53:07	85987	3197	Bottom Queue Add	3131.00
Keys	23:53:12	85992	3202	Start Manufacture	3131.00
Keys	23:54:02	86042	3252	Cycle 1 Complete.	3089.29
Keys	23:54:53	86093	3303	Cycle 2 Complete.	3047.57
Keys	23:55:44	86144	3354	Cycle 3 Complete.	3005.86
Keys	23:56:33	86193	3403	Cycle 4 Complete.	2964.14
Keys	23:57:24	86244	3454	Cycle 5 Complete.	2922.43
Keys	23:58:13	86293	3503	Cycle 6 Complete.	2880.71
Keys	23:59:03	86343	3553	Bottom Queue Add	2839.00
bushs	23:59:08	86348	3558	Start Manufacture	2839.00
bushs	00:00:05	86405	3615	Cycle 1 Complete.	2786.70
bushs	00:01:04	86464	3674	Cycle 2 Complete.	2734.40
bushs	00:02:04	86524	3734	Cycle 3 Complete.	2682.10
bushs	00:03:02	86582	3792	Cycle 4 Complete.	2629.80
bushs	00:04:01	86641	3851	Cycle 5 Complete.	2577.50
bushs	00:05:00	86700	3910	Cycle 6 Complete.	2525.20
bushs	00:05:58	86758	3968	Cycle 7 Complete.	2472.90
bushs	00:06:58	86818	4028	Cycle 8 Complete.	2420.60
bushs	00:07:57	86877	4087	Cycle 9 Complete.	2368.30
bushs	00:08:56	86936	4146	Whole job Completed	2316.00
Bearig	00:09:01	86941	4151	Start Manufacture	2316.00
Bearig	00:09:52	86992	4202	Cycle 1 Complete.	2276.87
Bearig	00:10:44	87044	4254	Cycle 2 Complete.	2237.75
Bearig	00:11:35	87095	4305	Cycle 3 Complete.	2198.62
Bearig	00:12:24	87144	4354	Cycle 4 Complete.	2159.50
Bearig	00:13:16	87196	4406	Cycle 5 Complete.	2120.37
Bearig	00:14:07	87247	4457	Cycle 6 Complete.	2081.25
Bearig	00:14:58	87298	4508	Cycle 7 Complete.	2042.12
Bearig	00:15:49	87349	4559	Whole job Completed	2003.00
Stubss	00:15:55	87355	4565	Start Manufacture	2003.00
Stubss	00:16:50	87410	4620	Cycle 1 Complete.	1952.86
Stubss	00:17:46	87466	4676	Cycle 2 Complete.	1902.71
Stubss	00:18:40	87520	4730	Cycle 3 Complete.	1852.57
Stubss	00:19:34	87574	4784	Cycle 4 Complete.	1802.43
Stubss	00:20:30	87630	4840	Cycle 5 Complete.	1752.28
Stubss	00:21:26	87686	4896	Cycle 6 Complete.	1702.14
Stubss	00:22:20	87740	4950	Whole job Completed	1652.00
Pinss	00:22:26	87746	4956	Start Manufacture	1652.00
Pinss	00:23:20	87800	5010	Cycle 1 Complete.	1612.67
Pinss	00:24:14	87854	5064	Cycle 2 Complete.	1573.33
Pinss	00:25:09	87909	5119	Cycle 3 Complete.	1534.00
Pinss	00:26:01	87961	5171	Cycle 4 Complete.	1494.67
Pinss	00:26:56	88016	5226	Cycle 5 Complete.	1455.33
Pinss	00:27:50	88070	5280	Cycle 6 Complete.	1416.00
Pinss	00:28:45	88125	5335	Cycle 7 Complete.	1376.67
Pinss	00:29:37	88177	5387	Cycle 8 Complete.	1337.33

Pinss	00:30:32	88232	5442	Whole job Completed	1298.00
Keyss	00:30:37	88237	5447	Start Manufacture	1298.00
Keyss	00:31:27	88287	5497	Cycle 1 Complete.	1256.50
Keyss	00:32:16	88336	5546	Cycle 2 Complete.	1215.00
Keyss	00:33:06	88386	5596	Cycle 3 Complete.	1173.50
Keyss	00:33:57	88437	5647	Cycle 4 Complete.	1132.00
Keyss	00:34:46	88486	5696	Cycle 5 Complete.	1090.50
Keyss	00:35:36	88536	5746	Cycle 6 Complete.	1049.00
Keyss	00:36:26	88586	5796	Cycle 7 Complete.	1007.50
Keyss	00:37:17	88637	5847	Cycle 8 Complete.	966.00
Keyss	00:38:06	88686	5896	Cycle 9 Complete.	924.50
Keyss	00:38:56	88736	5946	Bottom Queue Add	883.00
axless	00:39:02	88742	5952	Start Manufacture	883.00
axless	00:42:09	88929	6139	Cycle 1 Complete.	735.83
axless	00:45:16	89116	6326	Cycle 2 Complete.	588.67
axless	00:48:23	89303	6513	Cycle 3 Complete.	441.50
axless	00:51:30	89490	6700	Cycle 4 Complete.	294.33
axless	00:54:37	89677	6887	Cycle 5 Complete.	147.17
axless	00:57:44	89864	7074	Whole job Completed	0.00

VMD2 Cell 3(1.7)

Jobname	Actual Time(Clock)	Relative Time	TimeReal(Secs)	Event	Queuecost
Start Simulation	22:59:53	82793	0	Simulation Start	0.00
Mount	23:08:52	83332	526	Add	595.00
Mount	23:08:54	83334	529	Start Manufacture	595.00
Mount	23:10:04	83404	583	Cycle 1 Complete.	528.89
Mount	23:11:14	83474	604	Cycle 2 Complete.	462.78
Mount	23:12:26	83546	637	Cycle 3 Complete.	396.67
Mount	23:13:38	83618	691	Cycle 4 Complete.	330.56
Mount	23:14:49	83689	745	Cycle 5 Complete.	264.44
Mount	23:15:59	83759	798	Cycle 6 Complete.	198.33
Journ	23:16:36	83796	853	Add	480.33
Mount	23:17:10	83830	857	Cycle 7 Complete.	414.22
Mount	23:18:21	83901	886	Cycle 8 Complete.	348.11
Mount	23:19:31	83971	924	Bottom Queue Add	282.00
Journ	23:19:37	83977	942	Start Manufacture	282.00
Journ	23:20:24	84024	992	Cycle 1 Complete.	246.75
Houses	23:20:37	84037	1060	Add	280.75
Journ	23:21:13	84073	1129	Cycle 2 Complete.	245.50
Journ	23:22:00	84120	1155	Cycle 3 Complete.	210.25
Stubs	23:22:43	84163	1197	Add	611.25
Journ	23:22:47	84167	1264	Cycle 4 Complete.	576.00
Journ	23:23:36	84216	1332	Cycle 5 Complete.	540.75
Journ	23:24:24	84264	1400	Cycle 6 Complete.	505.50
Journ	23:25:12	84312	1405	Cycle 7 Complete.	470.25
Bracks	23:25:45	84345	1426	Add	750.25
Journ	23:26:00	84360	1442	Bottom Queue Add	715.00
Houses	23:26:05	84365	1449	Start Manufacture	715.00
Houses	23:26:19	84379	1470	Cycle 1 Complete.	706.50
Houses	23:26:32	84392	1493	Cycle 2 Complete.	698.00

Houses	23:26:47	84407	1498	Cycle 3 Complete.	689.50
Houses	23:27:01	84421	1517	Whole job Completed	681.00
Stubs	23:27:06	84426	1536	Start Manufacture	681.00
Stubs	23:28:01	84481	1555	Cycle 1 Complete.	630.88
Stubs	23:28:56	84536	1574	Cycle 2 Complete.	580.75
Stubs	23:29:52	84592	1593	Cycle 3 Complete.	530.63
Stubs	23:30:45	84645	1613	Cycle 4 Complete.	480.50
Stubs	23:31:41	84701	1632	Cycle 5 Complete.	430.38
Stubs	23:32:36	84756	1651	Cycle 6 Complete.	380.25
Stubs	23:33:32	84812	1667	Cycle 7 Complete.	330.13
Stubs	23:34:26	84866	1670	Whole job Completed	280.00
Bracks	23:34:30	84870	1688	Start Manufacture	280.00
Piston	23:35:17	84917	1693	Add	498.00
Bracks	23:35:38	84938	1703	Cycle 1 Complete.	442.00
Bracks	23:36:46	85006	1879	Cycle 2 Complete.	386.00
Bracks	23:37:54	85074	1884	Cycle 3 Complete.	330.00
Rotor	23:38:34	85114	1954	Add	613.00
Bracks	23:39:02	85142	2026	Cycle 4 Complete.	557.00
Bracks	23:40:10	85210	2097	Bottom Queue Add	501.00
Piston	23:40:15	85215	2134	Start Manufacture	501.00
Piston	23:40:47	85247	2168	Cycle 1 Complete.	473.75
Piston	23:41:20	85280	2238	Cycle 2 Complete.	446.50
Piston	23:41:52	85312	2310	Cycle 3 Complete.	419.25
Piston	23:42:25	85345	2314	Cycle 4 Complete.	392.00
Piston	23:42:58	85378	2344	Cycle 5 Complete.	364.75
Bracs	23:43:31	85411	2361	Add	698.75
Piston	23:43:33	85413	2366	Cycle 6 Complete.	671.50
Piston	23:44:05	85445	2416	Cycle 7 Complete.	644.25
Piston	23:44:38	85478	2466	Whole job Completed	617.00
Rotor	23:44:43	85483	2517	Start Manufacture	617.00
Rotor	23:45:22	85522	2537	Cycle 1 Complete.	585.56
Rotor	23:46:00	85560	2567	Cycle 2 Complete.	554.11
Rotor	23:46:39	85599	2616	Cycle 3 Complete.	522.67
Rotor	23:47:19	85639	2666	Cycle 4 Complete.	491.22
Rotor	23:47:58	85678	2716	Cycle 5 Complete.	459.78
Rotor	23:48:36	85716	2722	Cycle 6 Complete.	428.33
Rotor	23:49:15	85755	2760	Cycle 7 Complete.	396.89
Rotor	23:49:55	85795	2799	Cycle 8 Complete.	365.44
Rotor	23:50:34	85834	2836	Whole job Completed	334.00
Bracs	23:50:39	85839	2838	Start Manufacture	334.00
Bracs	23:51:47	85907	2877	Cycle 1 Complete.	278.33
Bracs	23:52:55	85975	2916	Cycle 2 Complete.	222.67
Bracs	23:54:03	86043	2947	Cycle 3 Complete.	167.00
Bracs	23:55:10	86110	2956	Cycle 4 Complete.	111.33
Bracs	23:56:18	86178	2995	Cycle 5 Complete.	55.67
Bracs	23:57:27	86247	3034	Bottom Queue Add	0.00
Bearin	23:58:57	86337	3072	Add	237.00
Bearin	23:58:59	86339	3077	Start Manufacture	237.00
Bearin	23:59:50	86390	3131	Cycle 1 Complete.	197.50
Bearin	00:00:41	86441	3187	Cycle 2 Complete.	158.00
Bearin	00:01:33	86493	3241	Cycle 3 Complete.	118.50
Bearin	00:02:22	86542	3296	Cycle 4 Complete.	79.00

Bearin	00:03:14	86594	3351	Cycle 5 Complete.	39.50
Bearin	00:04:05	86645	3406	Whole job Completed	0.00
Pistss	00:05:56	86756	3431	Add	29.00
Pistss	00:05:59	86759	3461	Start Manufacture	29.00
Pistss	00:06:31	86791	3467	Whole job Completed	0.00
Rotos	00:13:13	87193	3535	Add	221.00
Rotos	00:13:15	87195	3604	Start Manufacture	221.00
Rotos	00:13:53	87233	3671	Cycle 1 Complete.	189.43
Rotos	00:14:32	87272	3739	Cycle 2 Complete.	157.86
Rotos	00:15:11	87311	3807	Cycle 3 Complete.	126.29
Rotos	00:15:50	87350	3875	Cycle 4 Complete.	94.71
Rotos	00:16:29	87389	3880	Cycle 5 Complete.	63.14
Rotos	00:17:08	87428	3933	Cycle 6 Complete.	31.57
Rotos	00:17:48	87468	3988	Whole job Completed	0.00

Appendix E

Experimental Job and Task Details

Appendix E details the makeup of the fifteen Jobs used for the purposes of testing the Hybrid FMS system.

Experiment Job and Task Details

Job 1: Axle

Task 1:		**Task 2:**	
Machine Type:	NC Lathe	Machine Type:	NC Mill
Machine Code:	2.2	Machine Code:	1.2
Load/Unloadtime:	14	Load/Unloadtime:	6
# Cycles	8	# Cycles	8
# Tasks	2	# Tasks	2
Task Setup Time:	3	Task Setup Time:	43
Task Cycle Time:	8	Task Cycle Time:	128
Tool Code:	6.03	Tool Code:	4.6

Job 2: Bearing

Task 1:		**Task 2:**		**Task 3:**	
Machine Type:	NC Lathe	Machine Type:	NC Grinder	Machine Type:	NC Mill
Machine Code:	2.2	Machine Code:	3.2	Machine Code:	1.2
Load/Unloadtime:	4	Load/Unloadtime:	42	Load/Unloadtime:	6
# Cycles	8	# Cycles	8	# Cycles	8
# Tasks	3	# Tasks	3	# Tasks	3
Task Setup Time:	8	Task Setup Time:	31	Task Setup Time:	9
Task Cycle Time:	10	Task Cycle Time:	93	Task Cycle Time:	26
Tool Code:	6.03	Tool Code:	8.8	Tool Code:	2.04

Job 3: Bracket

Task 1:		**Task 2:**		**Task 3:**	
Machine Type:	NC Mill	Machine Type:	NC Grinder	Machine Type:	NC Lathe
Machine Code:	1.2	Machine Code:	3.2	Machine Code:	2.2
Load/Unloadtime:	12	Load/Unloadtime:	1	Load/Unloadtime:	1
# Cycles	8	# Cycles	8	# Cycles	8
# Tasks	3	# Tasks	3	# Tasks	3
Task Setup Time:	10	Task Setup Time:	6	Task Setup Time:	3
Task Cycle Time:	30	Task Cycle Time:	19	Task Cycle Time:	8
Tool Code:	1.65	Tool Code:	9.89	Tool Code:	7.1

Job 4: Bush

Task 1:	
Machine Type:	NC Mill
Machine Code:	1.2
Load/Unloadtime:	22
# Cycles	3
# Tasks	1
Task Setup Time:	3
Task Cycle Time:	8
Tool Code:	2.03

Job 5: Housing

Task 1:	
Machine Type:	NC Mill
Machine Code:	1.2
Load/Unloadtime:	1
# Cycles	4
# Tasks	1
Task Setup Time:	2
Task Cycle Time:	6
Tool Code:	4.6

Job 6: Journ(Journal)

Task 1:	
Machine Type:	NC Grinder
Machine Code:	3.2
Load/Unloadtime:	23
# Cycles	8
# Tasks	3
Task Setup Time:	2
Task Cycle Time:	7
Tool Code:	9.89

Task 2:	
Machine Type:	NC Mill
Machine Code:	1.2
Load/Unloadtime:	2
# Cycles	8
# Tasks	3
Task Setup Time:	10
Task Cycle Time:	30
Tool Code:	1.33

Task 3:	
Machine Type:	NC Lathe
Machine Code:	2.1
Load/Unloadtime:	13
# Cycles	8
# Tasks	3
Task Setup Time:	1
Task Cycle Time:	1
Tool Code:	7.1

Job 7: Key(Keyway/Shaft)

Task 1:	
Machine Type:	NC Mill
Machine Code:	1.2
Load/Unloadtime:	13
# Cycles	5
# Tasks	2
Task Setup Time:	5
Task Cycle Time:	15
Tool Code:	2.05

Task 2:	
Machine Type:	NC Lathe
Machine Code:	2.2
Load/Unloadtime:	18
# Cycles	5
# Tasks	2
Task Setup Time:	8
Task Cycle Time:	22
Tool Code:	7.1

Job 8: Mount(Mounting Block)

Task 1:	
Machine Type:	NC Mill
Machine Code:	1.2
Load/Unloadtime:	31
# Cycles	9

Task 2:	
Machine Type:	NC Lathe
Machine Code:	2.2
Load/Unloadtime:	12
# Cycles	9

Task 3:	
Machine Type:	NC Grinder
Machine Code:	3.2
Load/Unloadtime:	10
# Cycles	9

# Tasks	3	# Tasks	3	# Tasks	3
Task Setup Time:	1	Task Setup Time:	29	Task Setup Time:	1
Task Cycle Time:	4	Task Cycle Time:	86	Task Cycle Time:	2
Tool Code:	2.04	Tool Code:	6.05	Tool Code:	9.89

Job 9: Pin

Task 1:		**Task 2:**	
Machine Type:	NC Lathe	Machine Type:	NC Mill
Machine Code:	2.2	Machine Code:	1.2
Load/Unloadtime:	4	Load/Unloadtime:	1
# Cycles	6	# Cycles	6
# Tasks	2	# Tasks	2
Task Setup Time:	12	Task Setup Time:	12
Task Cycle Time:	37	Task Cycle Time:	36
Tool Code:	6.03	Tool Code:	4.6

Job 10: Piston

Task 1:		**Task 2:**		**Task 3:**	
Machine Type:	NC Grinder	Machine Type:	NC Lathe	Machine Type:	NC Mill
Machine Code:	3.1	Machine Code:	2.2	Machine Code:	1.2
Load/Unloadtime:	9	Load/Unloadtime:	3	Load/Unloadtime:	11
# Cycles	8	# Cycles	8	# Cycles	8
# Tasks	3	# Tasks	3	# Tasks	3
Task Setup Time:	13	Task Setup Time:	13	Task Setup Time:	2
Task Cycle Time:	39	Task Cycle Time:	38	Task Cycle Time:	5
Tool Code:	8.9	Tool Code:	5.05	Tool Code:	1.65

Job 11: Rotor

Task 1:		**Task 2:**		**Task 3:**	
Machine Type:	NC Grinder	Machine Type:	NC Lathe	Machine Type:	NC Mill
Machine Code:	3.1	Machine Code:	2.1	Machine Code:	1.2
Load/Unloadtime:	18	Load/Unloadtime:	4	Load/Unloadtime:	9
# Cycles	9	# Cycles	9	# Cycles	9
# Tasks	3	# Tasks	3	# Tasks	3
Task Setup Time:	12	Task Setup Time:	5	Task Setup Time:	4
Task Cycle Time:	36	Task Cycle Time:	14	Task Cycle Time:	13
Tool Code:	8.9	Tool Code:	5.05	Tool Code:	2.04

Job 12: Shaft

Task 1:		**Task 2:**		**Task 3:**	
Machine Type:	NC Mill	Machine Type:	NC Grinder	Machine Type:	NC Lathe
Machine Code:	1.2	Machine Code:	3.2	Machine Code:	2.2
Load/Unloadtime:	1	Load/Unloadtime:	1	Load/Unloadtime:	15

# Cycles	8
# Tasks	3
Task Setup Time:	3
Task Cycle Time:	10
Tool Code:	1.44

# Cycles	8
# Tasks	3
Task Setup Time:	6
Task Cycle Time:	19
Tool Code:	8.8

# Cycles	8
# Tasks	3
Task Setup Time:	5
Task Cycle Time:	16
Tool Code:	6.05

Job 13: Spline

Task 1:

Machine Type:	NC Grinder
Machine Code:	3.2
Load/Unloadtime:	9
# Cycles	1
# Tasks	1
Task Setup Time:	29
Task Cycle Time:	88
Tool Code:	9.35

Job 14: Stub

Task 1:

Machine Type:	NC Mill
Machine Code:	1.2
Load/Unloadtime:	23
# Cycles	1
# Tasks	1
Task Setup Time:	1
Task Cycle Time:	4
Tool Code:	1.77

Job 15: Valve

Task 1:

Machine Type:	NC Mill
Machine Code:	1.2
Load/Unloadtime:	5
# Cycles	4
# Tasks	2
Task Setup Time:	2
Task Cycle Time:	6
Tool Code:	1.44

Task 2:

Machine Type:	NC Grinder
Machine Code:	3.2
Load/Unloadtime:	16
# Cycles	4
# Tasks	2
Task Setup Time:	5
Task Cycle Time:	15
Tool Code:	9.89

Design and Development of a Hybrid Flexible Manufacturing System

A thesis presented in fulfilment of the requirements for the degree of

Master of Technology at

Massey University

Volume 2 - Hybrid FMS Programming

By

Matthew J Jolly

1999

MASSEY UNIVERSITY
1061570464

MASSEY

University

Introduction

Volume two of this thesis documents all Microsoft Visual Basic programming undertaken as part of the implementation of the Hybrid FMS Model as presented in Volume I of this work. Due to the sheer magnitude of programming undertaken it has been decided that the work be separated into a different volume so that the readability of the overall thesis is not affected.

Appendix A will detail the implementation of the services making up the Cell Controller software. Appendix B the implementation of the Factory Controller services.

Vault copy

Appendix A

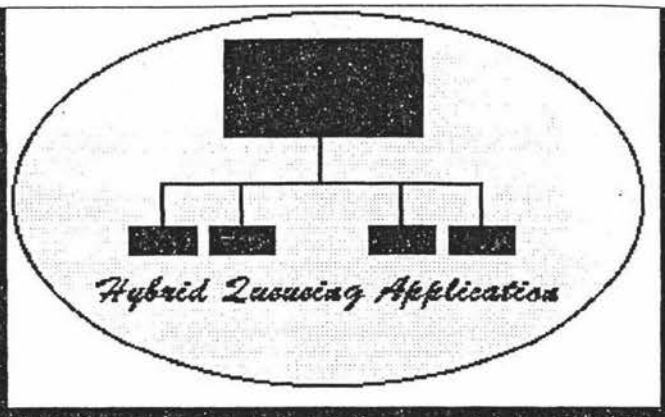
Cell Controller Services - Visual Basic Implementation

This Appendix details the programming employed in the implementation of the Cell Controller Services. It is intended that the programming be used in conjunction with the implementation chapter in Volume I, in this way the reader is able to get exact knowledge as to the way various features of the system have been implemented.

A.1 Implementation of the Queueing Application

The Cell Controller Queueing Application is responsible for the handling and queueing of tasks requiring operations at a given process. The application has been designed and implemented to be completely modular and easily transportable between cells. It is also loosely coupled to the rest of the system thereby giving high fault tolerance.

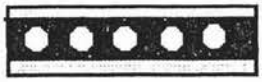
The following material is a print out from Microsoft Visual Basic showing all programming carried out for the applications implementation.



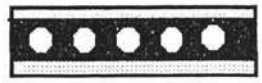
Hybrid Cell Controller Job Queueing Application

Continue...

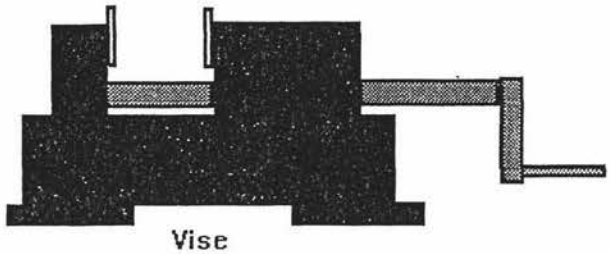
Rename Node...



Parallels No.1



Parallels No.2



Vise

```
Sub Exit_Click ()
Unload FixturesForm
End Sub
```

```
EWQINT.FRM - 1

ption Compare Text
ption Explicit
option Base 1

im Nodrecs(1) As noodesrecord

Sub Command1_Click ()
nload Startform
ewqueue.Show
End Sub

ub Exit_Click ()

nload Startform
nd Sub

ub Form_Unload (Cancel As Integer)
Unload Queue_Status
End Sub

ub Rename_Node_Click ()
'Declare local variables
im Title As String
im typ As Integer
Dim Msg As String
im answer As Integer
im Filelength As Integer
im Nodrec As noodesrecord
im Filenum As Integer
im Cellname As String

On Error Resume Next

sg = "Do you really want to change the name of this node?"
yp = MB_ICONSTOP + MB_YESNO
itle = "Are you sure?"

answer = MsgBox(Msg, typ, Title)

f answer = IDNO Then
    Exit Sub
nd If

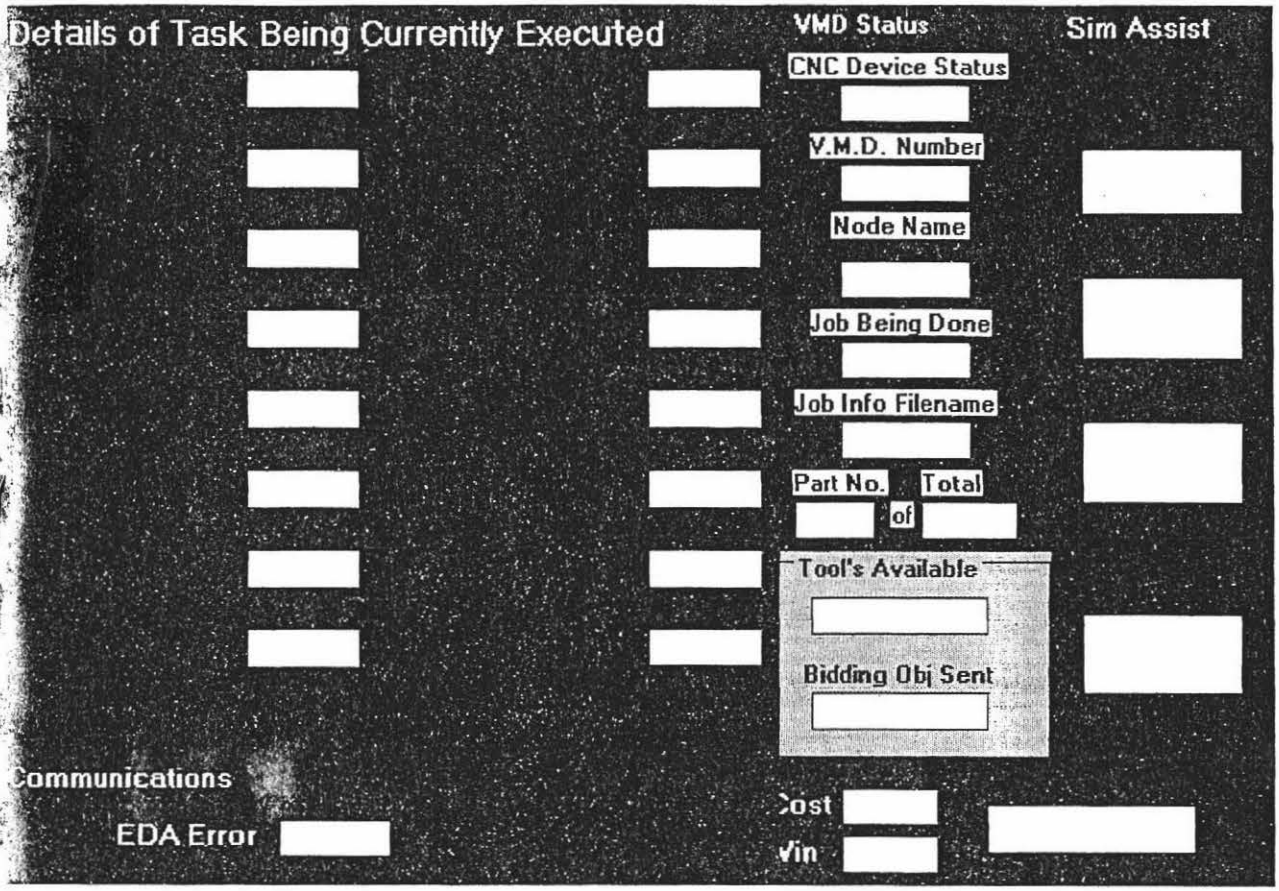

ill "C:\Cellcont\queue\config\Noderec.txt"


Msg = "What would you like to rename the node as?"
Title = "Rename Node"

Cellname = InputBox(Msg, Title)
```

```
'Filelength = Len(Nodrec)
'Filenum = FreeFile

'Open "C:\Cellcont\queue\config\Noderec.txt" For Random Access Read Write Share
 As Filenum Len = Filelength
'Get Filenum, 1, Nodrecs(1)
End Sub
```


Details of Task Being Currently Executed
VMD Status
Sim Assist
CNC Device Status
V.M.D. Number
Node Name
Job Being Done
Job Info Filename
Part No.
Total
of
Tool's Available
Bidding Obj Sent
Communications
EDA Error
Cost
Vin

```
'**************** THE QUEUEING APPLICATION ***************

'Queueing Application:
'Designed and Written by Matthew John Jolly(BTech[Hons])
'PhD Student
'Massey University
'Palmerston North
'New Zealand

Option Explicit

'Set all arrays to start at position 1
Option Base 1

'Nullify Case sensitivity in "like" Functions
Option Compare Text

'Declare incoming Queue variable
Dim Currentqueue(100) As Inqueue

Dim Nodrecs(1) As noodesrecord

Dim Thisqueuecost(1) As VMDQueueCost



'Declare pointers
Dim Incomingqueue As InQueuePointers
Dim OutGoingQueue As OutQueuePointers

'Declare Begin and end of IncomingQueue pointer Constants
Const InComingBeginOfQueue = 1
Const InComingFinOfQueue = 100

'Declare Begin and end of OutGoingQueue pointer Constants
Const OutGoingBeginOfQueue = 1
Const OutGoingEndOfQueue = 100

'Declare outgoing queue variable
Dim Nextqueue(100) As Outqueue
'Queue Control Variables
Dim FullInQFlag As Integer
Dim FullOutQFlag As Integer
'General Declarations
Dim X As Variant
Dim Dummy As Variant
Dim Text As Variant
Dim NodeNam As String * 10
Dim Msg As String * 40
Dim title As String * 35
Dim UpdateInQueueCost As Double

'EDA Function Declarations

' functions specific to VB (not standard eda fcns)
Declare Sub eda_version Lib "vdba.dll" (ByVal V$)
```

```
Declare Sub eda_get1_ascii Lib "vdba.dll" (ByVal n$, ByVal t$, ByVal f$, ByVal
s)
Declare Function eda_set1_ascii% Lib "vdba.dll" (ByVal n$, ByVal t$, ByVal f$,
ByVal V$)

' Note define_group returns a long
'
Declare Function eda_define_group& Lib "vdba.dll" (ByVal Count%, ByVal Detect%)
Declare Sub eda_delete_group Lib "vdba.dll" (ByVal Handle&)
'
  should always pass 0 for vsp
Declare Function eda_define_ntf& Lib "vdba.dll" (ByVal Handle&, ByVal enode$, By
Val etag$, ByVal efield$, ByVal vsp&)
Declare Sub eda_delete_ntf Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)

Declare Sub eda_lookup Lib "vdba.dll" (ByVal Handle&)
Declare Sub eda_read Lib "vdba.dll" (ByVal Handle&)
Declare Sub eda_write Lib "vdba.dll" (ByVal Handle&)
Declare Function eda_write1% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
Declare Sub eda_wait Lib "vdba.dll" (ByVal Handle&)
Declare Function eda_query_wait% Lib "vdba.dll" (ByVal Handle&)
'
Declare Function eda_get_error% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
Declare Function eda_get_ascii% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, ByVal
value$, ByVal maxlen%)
Declare Function eda_set_ascii% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, ByVal
value$)

' note pointer to float is passed
Declare Function eda_get_float% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, value
!)
Declare Function eda_set_float% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, value
!)

'eda communication variables
Dim Ghandler As Long
Dim eda_err As Variant
Dim tex As Integer
Dim Newnum As Variant
Dim Node As String * 15
Dim Tags As String * 15
Dim Field As String * 15
Dim answer As String * 80
Dim ans As String * 10
Dim defnode As Long
Dim trap As String * 10
Dim texts As String * 6
Dim Nodetyp As String * 20
Dim Nodes As String * 20
Dim bidrotenable As Variant
Dim Blocktyp As String * 20
Dim reply As Variant
Dim Bidoperatestate As Variant
Dim Bidnodetext As String * 10
Dim bidtokenstate As Variant
Dim Tokenstate As Integer
```

```
Dim InTas(2) As Jobrecord


'Declare  bidobject reception variables
Dim Currentbidjob As String * 15
Dim Currentbidnoprts As String * 5'As Integer
Dim CurrentBidInformationFilename As String * 20
Dim CurrentBidSetUpTime As String * 5
Dim CurrentBidmachtyp As Single
Dim CurrentBidCycleTime As Integer
Dim Currentbidloadunloadtime As Integer
Dim CurrentBidtool1Rq As Single
Dim CurrentBidtool2Rq As Single
Dim CurrentBidtool3Rq As Single
Dim CurrentBidtool4Rq As Single


Dim Thisbidding(1) As Bid_Data_to_Be_sent

'******IN HERE WE HAVE THE ROW POINTER VARIABLE DECLARATIONS FOR THE EXCELL DATA
COLLECTION PART**
Dim VMD1Rowpointer As Integer
Dim VMD2Rowpointer As Integer
Dim VMD3Rowpointer As Integer
Dim VMD4Rowpointer As Integer

'***Excell declaration****
Const MANUAL = 2, none = 0


'Aql Declarations
Dim Montharray(12) As Integer
Dim OldArea As Double
Dim AreaUnderGraph As Double
Dim OldQlength As Double
Dim Thefirsttime As Double
Dim Starttime As Double
Dim Oldnumber As Double

'Aql Row Pointer definitions
Dim Rowpointer As Integer ' defines the number of the row required to insert th
 aql data



Dim Thisvalue As Integer
Dim Firstpassed As Integer

Sub AqlCalc ()

Dim Currentmonth As String * 2
Dim Daystring As String * 2
Dim CurrentDay As Integer
Dim Arrayposition As Integer
Dim p As Integer
```

```
Dim NoDays As Integer
Dim NumberOfHours As String
Dim firsttime As Integer
Dim Secondtime As Double
Dim Totaltime As Double
Dim Thirdtime As Double
'due date bid calc dec
Dim Thistime As String * 8
Dim Thisdate As String * 15
Dim Queuelength As Double
Dim Abstime As Double
Dim Aql As Double


Thistime = Time$
'MsgBox (Thistime)

'02:07:35

'hisdate = Date$
MsgBox (Thisdate)

'get the first part of the due date time which is the number of hours
'up to the present time
firsttime = Val(Thistime)


'in the 09:20 hrs example above we also want to get the :20 part in
'terms of hours to get the total hrs in the day,this part does this
econdtime = (Val(Right$(Thistime, 5))) / 60


hirdtime = (Val(Right$(Thistime, 2))) / 3600

'MsgBox ("the number of hours is " & FirstTime)
MsgBox ("the number of Minutes is " & CStr(SecondTime) & " Hours")

'calculate the total hours in the day
otaltime = firsttime + Secondtime + Thirdtime



'get the string value of the task's due date month
Currentmonth = (Thisdate)

'transfer the current month into an integer eg:for 08-20-1996 we are getting the
'fact that it is month "8"
rrayposition = CInt(Currentmonth)

'extract the right part of the "thisdate" string so we can
'access the day part of it ie: the fact that it is day 20 for the above
aystring = (Right$(Trim(Thisdate), 7))

'transfer the Daystring variable into an integer so that we can use it mathemati
'ally
'and minus a day to say that although we are in the day we have not used all of
```

```
it
CurrentDay = CInt(Daystring) - 1

If Arrayposition = 1 Then
GoTo skipmonthlycount
End If

'total up the days of the months up to the current month
For p = 1 To (Arrayposition - 1)
    NoDays = Montharray(p) + NoDays
Next

skipmonthlycount:

'add the number of days in the current month that have been used
NoDays = NoDays + CurrentDay
'MsgBox (CStr(NoDays))

'calculate the number of hours this represents
NumberOfHours = CStr((NoDays * 24) + Totaltime)


Totaltime = Totaltime + (NoDays * 24)


'MsgBox CStr(NumberOfHours)
'MsgBox CStr(TotalTime)
'we now have the number of hours up to now and we require the calculation of the
 average
'queue length
Queuelength = Val(QueueCost.Text)


'AQL decs
'Dim OldArea As Single
'Dim AreaUnderGraph As Single
'Dim OldQLength As Single
'Dim Starttime As Single


'calculate the absolute time from the start of the simulation until now(the time
 the date is collected)
Abstime = Totaltime - Starttime 'absolute time = TimeNow - Start time

'MsgBox (CStr(Abstime))

'calculate the area under the graph
AreaUnderGraph = (OldArea) + (((OldQlength + Queuelength) * .5) * ((Abstime) - (
ldnumber))

'assign old values to temporary
Oldnumber = Abstime
OldArea = AreaUnderGraph
OldQlength = Queuelength

'MsgBox CStr(AreaUnderGraph)
```

```
'average the area over the current time increment(Calculate average queuelength
Aql = AreaUnderGraph / (Abstime)

'MsgBox CStr(Aql)


Curr_Time.Text = Time$

Res_Form.Aql_Time.AddItem Curr_Time.Text

'*TConst MANUAL = 2, none = 0

'*TCurr_Time.LinkMode = none
'*TCurr_Time.LinkTopic = "EXCEL|c:\CELL.XLS"
'*TCurr_Time.LinkItem = "R" & CStr(Rowpointer) & "C30"
'*TCurr_Time.LinkMode = MANUAL

 *TCurr_Time.LinkPoke



Aqltext.Text = Aql

es_Form.Aql_Aql.AddItem Aqltext.Text
'*TAqLtext.LinkMode = none
'*TAqLtext.LinkTopic = "EXCEL|c:\CELL.XLS"
 *TAqLtext.LinkItem = "R" & CStr(Rowpointer) & "C31"
'*TAqLtext.LinkMode = MANUAL

 *TAqLtext.LinkPoke


urrDate.Text = Date$

'*TCurrDate.LinkMode = none
 *TCurrDate.LinkTopic = "EXCEL|c:\CELL.XLS"
 *TCurrDate.LinkItem = "R" & CStr(Rowpointer) & "C32"
'*TCurrDate.LinkMode = MANUAL

 *TCurrDate.LinkPoke

 nd Sub

ub AuctionInitiated_Change ()
This particular event procedure is responsible for telling the Factory
Controller that the bidding objects have been sent and to start conducting
he actual auctioning procedure - it does this by placing a "1" in this textbox
which is done via the "Auction Control Algorithm" - the placing of a "1" in
this textbox causes this application to place a "1" in the Factory Controller
"AUCTIONSTD" (Auction Started) Database Block which tells the Factory Controlle

to start the auction. If this textbox Changes to "1" the program will be
ritten so as to reset the "AUCTIONSTD" datebase block in the Factory Controlle
```

```
r

Select Case AuctionInitiated

Case Is = 0#
Node = "FIX"
Blocktyp = "FBIDRQUEST"
Field = "A_CV"
texts = AuctionInitiated.Text
Call SendInfo(Node, Blocktyp, Field, texts)

Case Is = 1#
Node = "FIX"
Blocktyp = "AUCTIONSTD"
Field = "A_COLD"
texts = AuctionInitiated.Text
Call SendInfo(Node, Blocktyp, Field, texts)


End Select

End Sub

Sub Bidrequest_Change ()
'This particular textbox exists so that the Factory Controller
'can ask each of the VMD's to "Come and get the Bidding
'objects from it's bidding blocks and put them in their
'Bidding Object Reception Textboxes prior to a bid being
'Formulated

'For some reason this application fires immediately upon loading so the firstup
'variable is there to stop this and it does!
'If Firstup = 1 Then
 '   Firstup = 0
  '  Exit Sub
'End If

'On Error Resume Next
'Select Case Bidrequest

'Case Is = 1
'If this is a 1 the Factory Controller is saying to all
'VMD's "Come get the Bidding objects!" Which they do as
'below using eda

'node = "Fact"
'Field = "A_CV"

'Tags = "BidJob"
'bidjob.Text = eda_requestinfolink(node, Tags, Field)

'Tags = "BidNoParts"
'Bid_No_Of_Parts.Text = eda_requestinfolink(node, Tags, Field)

''Tags = "BidInfo"
'BidInformationFilename.Text = eda_requestinfolink(node, Tags, Field)
```

```
'Tags = "BidSUpTm"
'Bidsetuptime.Text = (eda_requestinfolink(node, Tags, Field))

'Tags = "BidMachTyp"
'BidMachTyp.Text = eda_requestinfolink(node, Tags, Field)

 Tags = "BidCycTm"
'Bidcycletime.Text = eda_requestinfolink(node, Tags, Field)

 Tags = "BidLoadUn"
'BidloadUnloadtime.Text = eda_requestinfolink(node, Tags, Field)

 Tags = "BidTool1Rq"
 Tool1required.Text = eda_requestinfolink(node, Tags, Field)


 Tags = "BidTool2Rq"
'Tool2required.Text = eda_requestinfolink(node, Tags, Field)

 ' Tags = "BidTool3Rq"
'Tool3required.Text = eda_requestinfolink(node, Tags, Field)

 Tags = "BidTool4Rq"
  ool4required.Text = eda_requestinfolink(node, Tags, Field)

 We are also required to set the "Tools Available" Textbox to '3' which opens
'up this VMD's library of tools(A File) and compares the tools in the library
'with those required in the "Tool#Required" textboxes as above.If all tools
 are available to complete the task then the "Tools Available" textbox resets
'itself to "1" which is then transmitted to the "V1TOOLAVLE" database block
'in the case of VMD1. The A.C.A. will then check to see if the machine(VMD1)
 i  s of the correct type and capacity to do the job and if it is will check the
'"V1TOOLAVLE" Block to see if all tools are available - if they are a bid
 ill be lodged

'ToolAvailability.Text = 3
'Case Else
 End Select

'this activates the event that sends the bidding objects to the BCA bidding bloc

 BidObjectControl.Text = 1

 d Sub

Function CalcStartTime ()

 im Currentmonth As String * 2
Dim Daystring As String * 2
 im CurrentDay As Integer
 im Arrayposition As Integer
Dim p As Integer
 m NoDays As Integer
 im NumberOfHours As String
Dim firsttime As Integer
```

```
Dim Secondtime As Double
Dim Thirdtime As Double
Dim Totaltime As Double

'due date bid calc dec
Dim Thistime As String * 8'5
Dim Thisdate As String * 15

'tells the program where to start recording the data
'Rowpointer = 1

Montharray(1) = 31
Montharray(2) = 29
Montharray(3) = 31
Montharray(4) = 30
Montharray(5) = 31
Montharray(6) = 30
Montharray(7) = 31
Montharray(8) = 31
Montharray(9) = 30
Montharray(10) = 31
Montharray(11) = 30
Montharray(12) = 31



Thistime = Time$
'MsgBox (Thistime)

'02:07:35

Thisdate = Date$
'MsgBox (Thisdate)

'get the first part of the due date time which is the number of hours
'up to the present time
firsttime = Val(Thistime)


'in the 09:20 hrs example above we also want to get the :20 part in
'terms of hours to get the total hrs in the day,this part does this
Secondtime = (Val(Right$(Thistime, 5))) / 60


Thirdtime = (Val(Right$(Thistime, 2))) / 3600

'MsgBox ("the number of hours is " & FirstTime)
'MsgBox ("the number of Minutes is " & CStr(SecondTime) & " Hours")

'calculate the total hours in the day
Totaltime = firsttime + Secondtime + Thirdtime

'MsgBox (CStr(Totaltime))

'get the string value of the task's due date month
Currentmonth = (Thisdate)
```

```
transfer the current month into an integer eg:for 08-20-1996 we are getting th
fact that it is month "8"
rrayposition = CInt(Currentmonth)

extract the right part of the "thisdate" string so we can
access the day part of it ie: the fact that it is day 20 for the above
aystring = (Right$(Trim(Thisdate), 7))

transfer the Daystring variable into an integer so that we can use it mathemati
ally
and minus a day to say that although we are in the day we have not used all of
t
urrentDay = CInt(Daystring) - 1

f Arrayposition = 1 Then
oTo skipmonthlicount
nd If

otal up the days of the months up to the current month
r p = 1 To (Arrayposition - 1)
   NoDays = Montharray(p) + NoDays
ext

kipmonthlicount:

dd the number of days in the current month that have been used
oDays = NoDays + CurrentDay
MsgBox (CStr(NoDays))

calculate the number of hours this represents
umberOfHours = CStr((NoDays * 24) + Totaltime)


alcStartTime = Totaltime + (NoDays * 24)


nd Function

unction CheckNodeName (Msg)

tle = "Configure Node Name"
eckNodeName = InputBox(Msg, title)
nd Function

b Command2_Click ()

d Sub

b CommDDELink (Text As TextBox, LinkItem)
his Sub Procedure creates an Automatic link between the
ueueing application and either the Factory Controller or
he Cell Controller

xt.LinkMode = none  'If the link is not already
                    'created then create it
```

```
Text.LinkTopic = "DMDDE|DATA"  'Link topic is FixDmacs

Text.LinkItem = LinkItem       'This is the Node,
                               'databaseblock and field we
                               'want to link to

Text.LinkMode = Link_Automatic 'Automatic update of data


End Sub

Function eda_requestinfolink (n, t, f)
Ghandler = eda_define_group(1, 0)
defnode = eda_define_ntf(Ghandler, n, t, f, 0)

Call eda_lookup(Ghandler)
Call eda_wait(Ghandler)
Call eda_read(Ghandler)
Call eda_wait(Ghandler)
Newnum = eda_get_ascii(Ghandler, defnode, answer, 50)
Call eda_delete_group(Ghandler)
eda_requestinfolink = answer
End Function

Sub Fixt_Click ()
On Error Resume Next
FixturesForm.Show
FixturesForm!FixPicture.Picture = Fixtures.Text
FixturesForm.Caption = Jobname.Text & " " & "Task" & Tasknumber.Text & " Fixtur
 Details"
End Sub

Sub RunDDEServer ()
AppActivate "Microsoft Visual Basic" 'Subject is VB
X = Shell("c:\wdmacs\dmdde.exe", 1) 'Makes sure The DDE  server is going
Dummy = DoEvents()

End Sub

Sub SaveOQPointers (OQueue, ByVal FileNo, ByVal File_Length, ByVal Fnam)
position = 1
Open Defaultpath & Fnam For Random As FileNo Len = File_Length

Put FileNo, position, OQueue
Close FileNo

End Sub

Sub SaveOutQueue (ByVal OutQ, ByVal file, ByVal File_Length, ByVal Fnam)
Open Defaultpath & Fnam For Random As file Len = File_Length
position = 1
Do Until position = 101
Put file, position, Nextqueue(position)
position = position + 1
Loop
Close file
```

```
:nd Sub

'ub SaveToOutqueue ()


'nd Sub

;ub SendInfo (n, t, f, Text)
:andler = eda_define_group(1, 0)
efnode = eda_define_ntf(Ghandler, n, t, f, 0)

all eda_lookup(Ghandler)
all eda_wait(Ghandler)
da_err = eda_get_error(Ghandler, defnode)
ext5.Text = Str$(eda_err)
ewnum = Text
eply = eda_set_ascii(Ghandler, defnode, Newnum)
all eda_write(Ghandler)
all eda_wait(Ghandler)
all eda_delete_group(Ghandler)

id Sub

ub setup_Click ()
1 Error Resume Next
etUpForm.Show
etUpForm.Caption = Jobname.Text & Tasknumber.Text & " " & "Task Setup Informati
1"
etUpForm!Picture1.Picture = SetUpInstructionsFilename.Text
nd Sub

ib Timer1_Timer ()

f Firstpassed = 0 Then
   Firstpassed = 1
   Thisvalue = 1
   Else
   Thisvalue = Thisvalue + 1
nd If


ieueCost.Text = Thisvalue




d Sub

b Tool4Required_Change ()
his particular procedure is a dummy procedure which
imulates the development of a job file in the factory
ontroller.It is developed purely for simulation purposes
nd creates two dummy files - matts.iff and patts.iff
```

```
'Dim filenum As Integer
'Dim Length As Single
'Dim InComingJob As jobrecord
'Dim Current_Task As Integer
'Dim FinalTaskNo As Integer
'Const StartLookingPoint = 1 'Look at first record to
'Dim Posi As Integer

'Select Case Tool4Required
'Case Is = 1#

'Length = Len(InComingJob)
'filenum = FreeFile

'Open Defaultpath & "Matts.iff" For Random Access Read Write Shared As filenum 'L
en = Length
'InTas(1).JobName = "Matts"
'InTas(1).InFoFilename = "Matts.iff"
'InTas(1).FinalTask = 2
'InTas(1).Tasknumber = 1
'InTas(1).Machinetype = 3.3
'InTas(1).Taskname = "Fart"
'InTas(1).EstCycletime = 2
'InTas(1).setuptime = 2
'InTas(1).EstLoadUnLoadTime = 5
'InTas(1).NooFCycles = 10
'InTas(1).CNCfilename = "Farter"
'InTas(1).Tool1Required = 1.1
'InTas(1).Tool2Required = 4.6
'InTas(1).Tool3Required = 5.4
'InTas(1).Tool4Required = 8.5
'InTas(1).Fixturesfilename = "Fixtures.bmp"
'InTas(1).Status = 1
'InTas(1).Duedate = "bou"
'InTas(1).Force = 1
'InTas(1).SetUpInstructionsFilename = "Test.bmp"
'InTas(1).Geometricfilename = "Fart"


'InTas(2).JobName = "Matts"
'InTas(2).InFoFilename = "Matts.iff"
'InTas(2).FinalTask = 2
'InTas(2).Tasknumber = 2
'InTas(2).Machinetype = 5.2
'InTas(2).Taskname = "Far"
'InTas(2).EstCycletime = 6
'InTas(2).setuptime = 3
'InTas(2).EstLoadUnLoadTime = 4
'InTas(2).NooFCycles = 15
'InTas(2).CNCfilename = "Farted"
'InTas(2).Tool1Required = 3.1
'InTas(2).Tool2Required = 4.3
'InTas(2).Tool3Required = 3.4
'InTas(2).Tool4Required = 5.5
'InTas(2).Fixturesfilename = "Fixtures.bmp"
'InTas(2).Duedate = "Date"
```

```
EWQUEUE.FRM - 14

InTas(2).Force = 0
InTas(2).SetUpInstructionsFilename = "Test.Bmp"
InTas(2).Geometricfilename = "Yes"

For Posi = 1 To 2
Put filenum, Posi, InTas(Posi)
Next Posi
Close filenum

Case Is = 2#

Length = Len(InComingJob)
filenum = FreeFile

Open Defaultpath & "Patts.iff" For Random Access Read Write Shared As filenum
n = Length
InTas(1).JobName = "Patts"
InTas(1).InFoFilename = "Patts.iff"
InTas(1).FinalTask = 2
InTas(1).Tasknumber = 1
InTas(1).Machinetype = 3.3
InTas(1).Taskname = "Tart"
InTas(1).EstCycletime = 2
InTas(1).setuptime = 2
InTas(1).EstLoadUnLoadTime = 5
InTas(1).NooFCycles = 10
InTas(1).CNCfilename = "tarter"
InTas(1).Tool1Required = 1.1
InTas(1).Tool2Required = 4.9
InTas(1).Tool3Required = 5.4
InTas(1).Tool4Required = 8.5
InTas(1).Fixturesfilename = "Fixtures.bmp"
InTas(1).Status = 1
InTas(1).Duedate = "tou"
InTas(1).Force = 1
InTas(1).SetUpInstructionsFilename = "Test.bmp"



InTas(2).JobName = "Patts"
InTas(2).InFoFilename = "Patts.iff"
InTas(2).FinalTask = 2
InTas(2).Tasknumber = 2
InTas(2).Machinetype = 4.3
InTas(2).Taskname = "Fat"
InTas(2).EstCycletime = 6
InTas(2).setuptime = 3
InTas(2).EstLoadUnLoadTime = 4
InTas(2).NooFCycles = 15
InTas(2).CNCfilename = "Fatted"
InTas(2).Tool1Required = 3.1
InTas(2).Tool2Required = 4.3
InTas(2).Tool3Required = 3.4
InTas(2).Tool4Required = 5.5
InTas(2).Fixturesfilename = "Fixtures.bmp"
InTas(2).Duedate = "Datt"
```

```
'InTas(2).Force = 1
'InTas(2).SetUpInstructionsFilename = "Test.bmp"


'For Posi = 1 To 2
'Put filenum, Posi, InTas(Posi)
'Next Posi
'Close filenum

'End Select



End Sub

Sub updateqcost ()
 'Delare queuecost file variables
Dim recordlength As Integer
Dim QueueCosts As VMDQueueCost
Dim filename As String
Dim Directoryname As String
Dim Thislength As Integer
Dim Filenum As Integer
Dim I As Integer

On Error Resume Next


Filenum = FreeFile 'get a freefile reference  number

recordlength = Len(QueueCosts) 'get length of storage array
'MsgBox (Queuecost.Text)
Directoryname = "c:\Cellcont\Queue\"  'denote directory name

filename = "Queucost.cst" 'denote filename

'open the file
Open Trim(Directoryname) & Trim(filename) For Random Access Read Write Shared As
 Filenum Len = recordlength

Get Filenum, 1, Thisqueuecost(1) 'get the record into memory

'Type VMDQueueCost
    '  Cellname As String * 8
   ' VMD1_Queue_Cost As Single
    'VMD2_Queue_Cost As Single
'    VMD3_Queue_Cost As Single
 '   VMD4_Queue_Cost As Single
'End Type

Thisqueuecost(1).Cellname = Nodename.Text

Select Case Val(VMDNo.Text)
```

```
Case Is = 1# 'if this queue application belongs to VMD1 then
Thisqueuecost(1).VMD1_Queue_Cost = Val(QueueCost.Text)

Case Is = 2# 'if this queue application belongs to VMD2 then
Thisqueuecost(1).VMD2_Queue_Cost = Val(QueueCost.Text)

Case Is = 3# 'if this queue application belongs to VMD3 then
Thisqueuecost(1).VMD3_Queue_Cost = Val(QueueCost.Text)

Case Is = 4#  'if this queue application belongs to VMD4 then
Thisqueuecost(1).VMD4_Queue_Cost = Val(QueueCost.Text)

End Select


Put Filenum, 1, Thisqueuecost(1) 'put everything back into the file

Close Filenum 'close the file



End Sub

Sub WriteToExcel (ThePointervalue As Integer, ColumnNumber As Integer)

poke the name of the job to be done to the spreadsheet
Jobsname.LinkMode = none
Jobsname.LinkTopic = "EXCEL|c:\CELL.XLS"
Jobsname.LinkItem = "R" & CStr(ThePointervalue) & "C" & CStr(ColumnNumber)
Jobsname.LinkMode = MANUAL
Jobsname.LinkPoke

increment the columnnumber to address the next column
ColumnNumber = ColumnNumber + 1

poke the value of the time the job was added to the queue to the spreadsheet
Thistime.LinkMode = none
Thistime.LinkTopic = "EXCEL|c:\CELL.XLS"
Thistime.LinkItem = "R" & CStr(ThePointervalue) & "C" & CStr(ColumnNumber)
Thistime.LinkMode = MANUAL
Thistime.LinkPoke

increment the columnnumber to address the next column
ColumnNumber = ColumnNumber + 1

inform the spreadsheet that the type of event we are doing is adding a job to t
e queue.
Event.LinkMode = none
Event.LinkTopic = "EXCEL|c:\CELL.XLS"
Event.LinkItem = "R" & CStr(ThePointervalue) & "C" & CStr(ColumnNumber)
Event.LinkMode = MANUAL
Event.LinkPoke

ncrement the columnnumber to address the next column
ColumnNumber = ColumnNumber + 1
```

```
'poke the date to the spreadsheet
'Thisdate.LinkMode = none
'Thisdate.LinkTopic = "EXCEL|c:\CELL.XLS"
'Thisdate.LinkItem = "R" & CStr(ThePointervalue) & "C" & CStr(ColumnNumber)
'Thisdate.LinkMode = MANUAL
'Thisdate.LinkPoke

'increment the columnnumber to address the next column
'ColumnNumber = ColumnNumber + 1


'poke the current queuecost to the spreadsheet
'Queuecost.LinkMode = none
'Queuecost.LinkTopic = "EXCEL|c:\CELL.XLS"
'Queuecost.LinkItem = "R" & CStr(ThePointervalue) & "C" & CStr(ColumnNumber)
'Queuecost.LinkMode = MANUAL
'Queuecost.LinkPoke

'increment the columnnumber to address the next column
'ColumnNumber = ColumnNumber + 1


'inform the spreadsheet that the type of event we are doing is adding a job to t
he queue.
'Queuetype.LinkMode = none
'Queuetype.LinkTopic = "EXCEL|c:\CELL.XLS"
'Queuetype.LinkItem = "R" & CStr(ThePointervalue) & "C" & CStr(ColumnNumber)
'Queuetype.LinkMode = MANUAL
'Queuetype.LinkPoke


'increment the columnnumber to address the next column
'ColumnNumber = ColumnNumber + 1

'inform the spreadsheet that the type of event we are doing is adding a job to t
he queue.
'Urgency.LinkMode = none
'Urgency.LinkTopic = "EXCEL|c:\CELL.XLS"
'Urgency.LinkItem = "R" & CStr(ThePointervalue) & "C" & CStr(ColumnNumber)
'Urgency.LinkMode = MANUAL
'Urgency.LinkPoke



'VMD1Rowpointer = 4
' VMD2Rowpointer = 4
' VMD3Rowpointer = 4
' VMD4Rowpointer = 4

End Sub

Sub BidObjectControl_Change ()

Dim Filelength As Integer
```

```
Dim ReceptQueue As Inqueue
Dim Fname As String * 30
Dim OutGoingQue As Outqueue
Dim Msg As String * 70
Dim title As String * 15
Dim Length As Single
Dim InComingJob As Jobrecord
im InTask() As Jobrecord
im OutTask() As Jobrecord
Dim I As Integer              'Redim Handle Variable
im Current_Task As Integer
im FinalTaskNo As Integer
Dim X As Integer
im Typ As Integer
im QueueFull As Variant
Dim TempNoParts As Integer
im TempBidCycleTime As Integer
im Newtaskcost As Variant
Dim TempBidLoadUnLoadTime As Integer
im TempBidSetUpTime As Integer
im Dummy As Variant
Dim ExQueue As Outqueue
im QueueEmp As Variant
m BidObj As Variant
onst StartLookingPoint = 1 'Look at first record to get current task
im j As Integer
m Columnfocus As Integer
im Fileconcerned As String * 20

his textbox has an automatic DDE Link to a Cell Controller
atabase block called "BidControl" - this is so the Cell
Controller Database Blocks can instruct the Queueing
pplication to execute a variety of functions for it.
his textbox also has a link to a VMD database block called
"V1QControl" which allows V.M.D. specific operations to
ake place i.e:The BidControl database block is general whereas
'V1Qcontrol is V.M.D. specific
unctions are outlined in "case is" 2 to 6 as below."Case
s 1" is put there by the Bid Request textbox Change event
fter the bidding objects have been retrieved from the
actory Controller bidding blocks and put into the "Bidding
bject Reception" bid object textboxes using eda.

hen this textbox changes to a 1 as a result of this it
eans the Queue Application has recieved the bidding
bjects and that they can now be shunted from the
'Bidding Object Reception" textboxes and into the Cell Controller
idding Blocks using eda where a decision can be made
bout whether or not to place a bid and if so calculate
he bid cost using FixDmacs Calculation blocks.The Current
ueue Cost must also be transferred to a FixDmacs database
lock Called "V" & VMDNo.Text & NodeName.Text & "QCost"
hich will be added to the cost of the currently bidded
ask that is being auctioned. Also when this event occurs
his application opens this VMD'S Tool Specification
ile and to sees whether or not the Tools that appear in the
```

```
'"Bidding Object Reception" textboxes are part of this VMD'S
'equipment - if the tools are available then this application
'puts a "1" in the "Tools Available" textbox and sends it
'to the '"V" & VMDNO.text & "TOOLAVLE" Cell Controller
'database block where the A.C.A. checks to see for the VMD'S
'Bid Eligibility

On Error GoTo checkthis

'MsgBox (Bidobjectcontrol)
If Bidobjectcontrol Like ("?E****") Then
Exit Sub
End If


Select Case Bidobjectcontrol


Case Is = 1#

'Sends the Job Name resident in the "Job Name" textbox
'which is in the "Bidding Object Reception" frame to
'the "BidJob" database block in the Cell Controller
'node = "fact"
'Blocktyp = "BidJob"
'Field = "A_CV"
'texts = Trim(bidjob.Text)
'Call SendInfo(node, Blocktyp, Field, texts)
'MsgBox ("Bidjob")
'Send the JobInfoFilename resident in the "JobInformationfilename"
'textbox which is in the "Bidding Object Reception" frame
'to the "BidInfo" database block in the Cell Controller
'Blocktyp = "BidInfo"
'texts = Trim(BidInformationFilename.Text)
'Call SendInfo(node, Blocktyp, Field, texts)
'MsgBox ("bidinfo")
'Send the No Of Parts Information resident in the "NoOfParts"
'textbox which is in the "Bidding Object Reception" frame
'to the "BidNoofParts" database block in the Cell Controller
'Blocktyp = "BidNoParts"
'texts = Trim(Bid_No_Of_Parts.Text)
'Call SendInfo(node, Blocktyp, Field, texts)
'MsgBox ("Bidnoparts")
'Send the task setuptime Information resident in the
'"Set Up Time" textbox which is in the "Bidding Object
'Reception " frame to the "BidSUpTime" database block
'in the Cell Controller
'Blocktyp = "BidSupTime"
'texts = Trim(Bidsetuptime.Text)
'Call SendInfo(node, Blocktyp, Field, texts)
'MsgBox ("Bidsuptime")
'Send the task Machine Type Information resident in the
'"Machine Type" textbox which is in the "Bidding Object
'Reception " frame to the "BidMachTyp" database block
'in the Cell Controller
```

```
'Dim Currentbidjob As String * 15
'Dim Currentbidnoprts As Integer
'Dim CurrentBidInformationFilename As String * 20
'Dim CurrentBidSetUpTime As Integer
'Dim CurrentBidmachtyp As Single
'Dim CurrentBidCycleTime As Integer
'Dim Currentbidloadunloadtime As Integer
 Dim CurrentBidtool1Rq As Single
 Dim CurrentBidtool2Rq As Single
 Dim CurrentBidtool3Rq As Single
 Dim CurrentBidtool4Rq As Single

 Blocktyp = "BidMachTyp"
 texts = Trim(BidMachTyp.Text)
 Call SendInfo(node, Blocktyp, Field, texts)
 MsgBox ("Bidmachtyp")
 Send the task Cycletime Information resident in the
 "Cycle Time" textbox which is in the "Bidding Object
 Reception " frame to the "BidCycTime" database block
 in the Cell Controller
 Blocktyp = "BidCycTime"
 texts = Trim(Bidcycletime.Text)
 Call SendInfo(node, Blocktyp, Field, texts)
 ' MsgBox ("Bidcyctime")
 Send the task first operation Tool Type Information
 resident in the '"Tool1" textbox which is in the "Bidding
 Object Reception " frame to the "BidTool1Rq" database
 block in the Cell Controller
 Blocktyp = "BidTool1Rq"
 texts = Trim(Tool1required.Text)
 Call SendInfo(node, Blocktyp, Field, texts)
 MsgBox ("Bidtool1")
 Send the task second operation Tool Type Information
 resident in the '"Tool2" textbox which is in the "Bidding
 Object Reception " frame to the "BidTool2Rq" database
 block in the Cell Controller
 Blocktyp = "BidTool2Rq"
 texts = Trim(Tool2required.Text)
 Call SendInfo(node, Blocktyp, Field, texts)
 MsgBox ("Bidtool2")
 Send the task third operation Tool Type Information
 resident in the '"Tool3" textbox which is in the "Bidding
 Object Reception " frame to the "BidTool3Rq" database
 block in the Cell Controller
 Blocktyp = "BidTool3Rq"
 texts = Trim(Tool3required.Text)
 Call SendInfo(node, Blocktyp, Field, texts)
 MsgBox ("Bidtool3")
 Send the task fourth operation Tool Type Information
 resident in the '"Tool4" textbox which is in the "Bidding
 Object Reception " frame to the "BidTool4Rq" database
 block in the Cell Controller
 Blocktyp = "BidTool4Rq"
 texts = Trim(Tool4required.Text)
 Call SendInfo(node, Blocktyp, Field, texts)
 MsgBox ("Bidtool4")
```

```
'Send the individual task load/unload time Information
'resident in the '"LoadUnTime" textbox which is in the
'"Bidding Object Reception" frame to the "BidloadUn"
'database block in the Cell Controller

'BlockTyp = "BidLoadUn"
'texts = Trim(BidloadUnloadtime.Text)
'Call SendInfo(Node, BlockTyp, Field, texts)
'MsgBox ("Bidloadun")

'Send the Current Queue Cost from the Queueing application
'to the FixDmacs database block which monitors the Current
'cost of this VMD's Incoming queue

'Trigger the commencement of the bid formulation process by sending a "1"
'to the "Bidfrmstrt" database block in the Cell Controller. The Cell Controller
'will monitor this block and when it turns to a "1" then the comparison of job
'parameters with VMD parameters and subsequent bid formulation will begin.

'Blocktyp = "BIDFRMSTRT"
'Texts = "1"
'Call Sendinfo(Node, Blocktyp, Field, Texts)

Case Is = 2#
'When this Textbox changes to a 2 this program sends the
'calculated bid for this VMD associated with the Bid
'Objects resident in the "Bidding Object Reception" frame
'from the "Bid Cost" textbox to the database Block in the
'Factory Controller called - NodeName.Text & "V" &
'VMDNo.Text & BIDCST ie: If the NodeName was "CC1" and the
'the VMDNo was "1" then this block would be "CC1V1BDCST"

Node = Factory
Blocktyp = Trim(Nodename.Text) & "V" & Trim(VMDNo.Text) & "BDCST"
Field = "A_CV"
texts = BidCost.Text
Call SendInfo(Node, Blocktyp, Field, texts)

'We also have to send a prompt to the Factory Controller which tells the factor
'Controller to start the auctioning procedure.
Node = Factory
Blocktyp = "COMPAREBID"
Field = "A_CV"
texts = "1"
Call SendInfo(Node, Blocktyp, Field, texts)

Case Is = 3#
Dim Jobtobedone As String
Dim Thistask As Jobrecord
Dim Thisjob() As Jobrecord
Dim recordlength As Integer
Dim Amounttosubtract As Single
Dim Oldstatus As Integer
Dim CurrentVMD As String
Dim NumberOftasks As Integer
Dim Thisfilesnum As Integer
```

```
m Filenum As Integer
m CurrTaskcst As Integer
im Thisfilesname As String
atic ToolStatus(4) As String

When the V1QCONTROL VMD database block turns this into a 3
his program takes a job off the top of the incoming queue
nd loads that job's jobname,JobInfo,No.of parts required,
etuptime,loadunloadtime and cycletime fields straight
rom the queue into the "Incoming Queue Data" frames
Job Name" and "Job Information Filename" textboxes. It
hen opens the "JobInfo" File(eg:Matt.iff) reads the status
it and then reads the field information at that position
n the file into the remaining "Job Information Filename"
extboxes.The Status bit is then incremented by one and
resaved to file so that when the file is reopened the next
ask information can be read into the "Outgoing Queue Data"
extboxes when the job is finished


First Check to see if any tasks are in the queue

f Queue_Status.Inqueue.ListCount = 0 Then
   GoSub Queue_is_empty
   Exit Sub
   Else

      'get the name of the item at the top of the list and open up that jobfi
 and then put that
      'information into the "Task Being Currently Executed" frame.
      Jobtobedone = Queue_Status.Inqueue.List(0)


      Call AqlCalc
      '*****THIS IS THE RESULT COLLECTION PART
      Res_Form.Res_Job.AddItem Trim(Jobtobedone)
      Res_Form.Res_Event.AddItem "Start Manufacture"

      'we are required to find out what time it is so that we can shunt this
formation to the
      'results sheet
      Res_Form.Res_time.AddItem Time$

      'tell the results sheet the value of the queuecost when this was done
      Res_Form.Res_Cost.AddItem QueueCost.Text

      '*****************************************************************
**************

      'MsgBox (JobToBeDone)
      recordlength = Len(Thistask)

      Filenum = FreeFile
      'now open the jobfile that has this name and then put the information i
o the textboxes
```

```
        Fileconcerned = "Jobfile"
        Open Netdrive & "Factory\Jobfiles\" & Trim(Jobtobedone) & ".iff" For Ra
dom Access Read Write Shared As Filenum Len = recordlength

         MsgBox (Netdrive & "Factory\Jobfiles\" & Trim(JobToBeDone) & ".iff")

        'Type JobRecord
        'Jobname As String * 13
        'InFoFilename As String * 20
        'FinalTask As Integer
        'Tasknumber As Integer
        'Machinetype As Single 'Type as 3.1 the ".1" represents capacity
        'Taskname As String * 22
        'EstCycletime As Integer
        'Setuptime As Integer
        'Estloadunloadtime As Integer
        'NoofCycles As Integer
        'CNCfilename As String * 13
        'Geometricfilename As String * 13
        'Tool1required As Single
        'Fixturesfilename As String * 13
        'Status As Integer
        'Duedate As String * 8
        'Force As Integer
        'SetUpInstructionsFilename As String * 20 'Shows datum graphically
        'End Type

        ReDim Thisjob(1)

        Get Filenum, 1, Thisjob(1)

'        MsgBox (CStr(Thisjob(1).Status))
        ReDim Preserve Thisjob(Thisjob(1).Status)

        For I = 1 To Thisjob(1).Status
            Get Filenum, I, Thisjob(I)
        Next

        'Now read all the information at the status position into the required
extboxes

        Jobname.Text = Thisjob(Thisjob(1).Status).Jobname
        Jobinf.Text = Thisjob(Thisjob(1).Status).InFoFilename
        No_OfParts.Text = Thisjob(Thisjob(1).Status).NoOfCycles
        Tasksetuptime.Text = Thisjob(Thisjob(1).Status).setuptime
        Estcycletime.Text = Thisjob(Thisjob(1).Status).Estcycletime
        LoadUn.Text = Thisjob(Thisjob(1).Status).Estloadunloadtime
        Tasknumber.Text = Thisjob(Thisjob(1).Status).Status
        Taskname.Text = Thisjob(Thisjob(1).Status).Taskname
        InTool1required.Text = Thisjob(Thisjob(1).Status).Tool1required
        'InTool4required.Text = Thisjob(Thisjob(1).Status).Tool4required
        'InTool3required.Text = Thisjob(Thisjob(1).Status).Tool3required
        'InTool2required.Text = Thisjob(Thisjob(1).Status).Tool2required
        Finaltask.Text = Thisjob(Thisjob(1).Status).Finaltask
        Machinetype.Text = Thisjob(Thisjob(1).Status).Machinetype
        CNCfilename.Text = Thisjob(Thisjob(1).Status).CNCfilename
```

```
        Duedate.Text = Thisjob(Thisjob(1).Status).Duedate
        Fixtures.Text = Thisjob(Thisjob(1).Status).Fixturesfilename
        SetUpInstructionsFilename.Text = Thisjob(Thisjob(1).Status).SetUpInstruc
rionsFilename
        Geometric.Text = Thisjob(Thisjob(1).Status).Geometricfilename
        Forcevar = Thisjob(1).Force


        'now we are required to increment the "Status variable to reflect the fa
ct that the
        'task that is next required to be done is the next record.

        'store the old value of the status bit in a temporary holding variable
        Oldstatus = Thisjob(1).Status

        'increment counter if we are not at the end of the job
        If Thisjob(1).Status = CInt(Finaltask.Text) Then
        'This is the final task to be done reset task status to 1 so that if a n
ew
        'job is required to be done in the future then the jobfile is ready to c
.
                Thisjob(1).Status = 1
                Finaltask = 1
                Else
                'we are not yet at the end of the job therefore increment the st
tus bit
                    Thisjob(1).Status = Thisjob(1).Status + 1
        End If


        'Put the value of this variable back into the first record
        For I = 1 To Oldstatus
            Put Filenum, I, Thisjob(I)
        Next


        Close Filenum


        '******TempNoParts = CInt(Currentqueue(Incomingqueue.Instartofqueue).Bi
No_Parts)
        '*******TempBidCycleTime = CInt(Currentqueue(Incomingqueue.Instartofque
e).Bidcycletime)
        '*******TempBidLoadUnLoadTime = CInt(Currentqueue(Incomingqueue.Instart
queue).BidloadUnloadtime)
        '*******TempBidSetUpTime = CInt(Currentqueue(Incomingqueue.Instartofque
).Bidsetuptime)

    'Open file to get remaining task information from file and increment status
bit
    '******Length = Len(InComingJob)
    '******Filenum = FreeFile

    '******Open Defaultpath & JobInf.Text For Random Access Read Write Shared A
Filenum Len = Length
```

```
    '******X = X + 1    'Redimension array = 1 deep, 1 dimensional array to rea
 status bit
    '******ReDim InTask(X)
    '******Get Filenum, StartLookingPoint, InTask(X)
    'Find out what the current task is and open up the infofile
    'at that position so the  remaining information can be read and put
    'into the Incoming Queue Data frame - this is done by reading
    'the status variable(Which only has a value at the array's
    'first position)'to an intermediate variable called
    ' "Current_Task" so that the status variable can be
    'incremented independent of it's previous value as it's
    'previous value is still required to open the current task
    'and read the current task information into the "Incoming
    'Queue data " frame.A variable sized array is used as incoming
    'jobinfo files will have different sizes depending on how
    'many tasks are required for each job thus maximising memory

    '*****Current_Task = InTask(X).Status
    '*****FinalTaskNo = InTask(X).FinalTask
    '****InTask(X).Status = InTask(X).Status + 1
    '****If Current_Task = FinalTaskNo + 1 Then
            '****MsgBox ("Job already Complete,FixDmacs software program corrup
ed!")
          '****'Goto next task in queue
          '****Incomingqueue.Instartofqueue = Incomingqueue.Instartofqueue + 1

          '****'Test pointer status
          '****If Incomingqueue.Instartofqueue = Incomingqueue.InfinishofQueue
Then
          '****GoSub EmptyMessage
          '****End If

          '****If Incomingqueue.Instartofqueue = (InComingFinOfQueue + 1) And
ncomingqueue.InfinishofQueue = InComingBeginOfQueue Then
          '****GoSub EmptyMessage
          '****Incomingqueue.Instartofqueue = InComingBeginOfQueue
          '****End If

          '****If Incomingqueue.Instartofqueue = (InComingFinOfQueue + 1) And
ncomingqueue.InfinishofQueue > InComingBeginOfQueue Then
          '****Incomingqueue.Instartofqueue = InComingBeginOfQueue
          '****End If

          '****Queuestart.Text = Incomingqueue.Instartofqueue

          'Save new queue pointer state to file
          '****Position = 1
          '****Filenum = FreeFile
          '****Filelength = Len(Incomingqueue)
          '****Fname = "IPV" & VMDNo.Text & Nodename.Text & ".Txt"
          '****Open Defaultpath & Fname For Random As Filenum Len = Filelength
          '****Put Filenum, Position, Incomingqueue
          '****Close Filenum

    '****Else    'This job has task to do,save new status bit to file
```

```
            '****Put Filenum, StartLookingPoint, InTask(X)

            '****ReDim InTask(FinalTaskNo) 'ReDimension array to suit length of
            'Complete JobInfo file
            '****For i = 1 To FinalTaskNo
            'Put all task info into memory
            '****Get Filenum, i, InTask(i)
            '****Next i

            'Put current task information into the InComingqueue Data Frame
            'from memory prior to transfer to the vmd

            '****Tasknumber.Text = InTask(Current_Task).Tasknumber
            '****Taskname.Text = InTask(Current_Task).Taskname
            '****InTool1required.Text = InTask(Current_Task).Tool1required
            '****InTool4Required.Text = InTask(Current_Task).Tool4required
            '****InTool3Required.Text = InTask(Current_Task).Tool3required
            '****InTool2Required.Text = InTask(Current_Task).Tool2required
            '****FinalTask.Text = InTask(Current_Task).FinalTask
            '****Machinetype.Text = InTask(Current_Task).Machinetype
            '****CNCfilename.Text = InTask(Current_Task).CNCfilename
            '****Duedate.Text = InTask(Current_Task).Duedate
            '****Fixtures.Text = InTask(Current_Task).Fixturesfilename
            '****SetUpInstructionsFilename.Text = InTask(Current_Task).SetUpIns
ructionsFilename
            '****Geometric.Text = InTask(Current_Task).Geometricfilename
            '****Close Filenum
          'Now we must continue and load all the pertinent information
          'from the "Incoming Queue Data" frame textboxes into the vmd
          'as specified in the "VMD Status" frame.The pertinent data
          'includes Tasksetuptime,Number of parts required to be made,
          'Estimated Cycletime,LoadUnLoadTime and the CNC Filename
          'required to manufacture the part

          'Send the CNC Filename from the queueing algorithm to the vmd


          'node = Nodename.text
          'Blocktyp = "V" & Trim(VMDNo.text) & "RN" & Trim(Nodename.text) & "S
"
          'field = "A_CV"
          'Texts = CStr(Tasksetuptime.text)
          'Call SendInfo(node, Blocktyp, field, Texts)

          'Send no. parts info to the vmd
          Node = Trim(Nodename.Text)
          Field = "A_CV"
          Blocktyp = "V" & Trim(VMDNo.Text) & "RN" & Trim(Nodename.Text) & "No

          texts = CStr(No_OfParts.Text)
          Call SendInfo(Node, Blocktyp, Field, texts)


         'set the current part block to 1
          Blocktyp = "V" & Trim(VMDNo.Text) & "CURRPART"
          texts = "1"
```

```
        Call SendInfo(Node, Blocktyp, Field, texts)

        'Send the Estimated Cycletime to the vmd
        'Blocktyp = "V" & Trim(VMDNo.text) & "RN" & Trim(Nodename.text) & "C
m"
        'Texts = CStr(Estcycletime.text)
        'Call SendInfo(node, Blocktyp, field, Texts)

        'Send the Load & Unload time from the queueing algorithm
        'to the vmd
        'Blocktyp = "V" & Trim(VMDNo.text) & "RN" & Trim(Nodename.text) & "I
n"
        'Texts = CStr(LoadUn.text)
        'Call SendInfo(node, Blocktyp, field, Texts)

        'Send the cnc Filename to the VMD

        'Blocktyp = "V" & Trim(VMDNo.text) & "RN" & Trim(Nodename.text) & "C
C"
        'Texts = CNCfilename.text
        'Call SendInfo(node, Blocktyp, field, Texts)

        '*************this next part is only for simulation purposes delete
or the real thing

        'CurrTaskcst = (Val(No_OfParts.text) * (Val(Estcycletime.text) + 2 *
Val(LoadUn.text))) + Val(Tasksetuptime.text)

         'curr task cost is the cost of one cycle only
         CurrTaskcst = (Val(Estcycletime.Text) + 2 * Val(LoadUn.Text)) + Val
Tasksetuptime.Text)

        ' MsgBox ("The number of parts are " & CStr(No_OfParts.Text))
        ' MsgBox ("The number of cycles are " & CStr(Estcycletime.Text))
         'MsgBox ("The time for unloading is " & CStr(LoadUn.Text))
        ' MsgBox ("The time for setup is " & CStr(Tasksetuptime.Text))


         'MsgBox (CStr(Currtaskcost))
        'the calculated value is of course in minutes - we need to convert it
 into seconds so
        'that the FixDmacs VMD simulator can understand it

         'Therefore -
         '****Currtaskcost = Currtaskcost * 60

        'now send this information to the relevant tasktime block
        Blocktyp = "V" & VMDNo.Text & "Tasktime"
        texts = CStr(CurrTaskcst)
        Call SendInfo(Node, Blocktyp, Field, texts)



        'The task has been loaded into the vmd and this fact must now
        'be conveyed to the vmd status textboxes
        Currentjob.Text = Jobname.Text
```

```
            VMDJobinfo.Text = Jobinf.Text
            VMDTotalNo.Text = No_OfParts.Text
            VMDPartNo.Text = 1



            'Now that the task has been loaded into the vmd we must alter the
            'InComingQueue pointer so that the next task to be done in future
            'is the next task in the queue and not the one that we have just done
            'Also we must recalculate the incoming queue Cost and write a program
            'that recalculates the queue cost after each part is done in the
            'vmd as a task is still considered to be queued until all of it's cyc
es are
            'complete- this is done through monitoring the vmd status textboxes.I
            'might be imagined that when the job's task is complete we will have
othing
            'to refer to when we load the next task into the outgoing Queue,howev
            'this can be achieved by looking at the "InComing Queue Data" frames
            'parameters as we will load the job just worked on's next task into t

            'outgoing queue before loading the task off the top of the incoming
            'queue into the "InComing Queue Data" frame textboxes

            '****Incomingqueue.Instartofqueue = Incomingqueue.Instartofqueue + 1


            'Must now test new pointer state for queue emptyness and need to wrap
round
            'InStartOfQueue pointer



            '****If Incomingqueue.Instartofqueue = Incomingqueue.InfinishofQueue
hen
            '****GoSub EmptyMessage
            '****End If

          '****If Incomingqueue.Instartofqueue = (InComingFinOfQueue + 1) And Inc
omingqueue.InfinishofQueue = InComingBeginOfQueue Then
            '****GoSub EmptyMessage

            '****Incomingqueue.Instartofqueue = InComingBeginOfQueue
            '****End If


            '****If Incomingqueue.Instartofqueue = (InComingFinOfQueue + 1) And Inc
mingqueue.InfinishofQueue > InComingBeginOfQueue Then
            '****Incomingqueue.Instartofqueue = InComingBeginOfQueue
            '****End If

            'Update queue posititon textbox
            '****Queuestart.Text = Incomingqueue.Instartofqueue

            'Now we must save the new incoming queue queue pointer state to file
            '****Position = 1
```

```
        '****Filenum = FreeFile
        '****Filelength = Len(Incomingqueue)
        '****Fname = "IPV" & VMDNo.Text & Nodename.Text & ".Txt"
        '****Open Defaultpath & Fname For Random As Filenum Len = Filelength
        '****Put Filenum, Position, Incomingqueue
        '****Close Filenum

      'but we are also required to remove the job from the incoming queue and
then save the new queue
      'status to the incoming queue file.
      Queue_Status.Inqueue.RemoveItem 0

      'it is also required that we delete the tool reservation file fpr this
ob being loaded
      'because if we dont the next job will not be able to compete for tool a
ctions as the
      'system will say that the job's tools have already been reserved.

      'GoTo Nextline
      Kill Netdrive & "Cellcont\Toolbid\" & Trim(Nodename.Text) & "V" & Trim(
MDNo.Text) & "Rst.res"
'Nextline:
'       MsgBox (Netdrive & "Cellcont\Toolbid\" & Trim(Nodename.Text) & "V" & T
im(VMDNo.Text) & "Rst.res")

      'now call the procedure that updates the queue

      CurrentVMD = VMDNo.Text

      NumberOftasks = Queue_Status.Inqueue.ListCount


      '*********************************************8
      Dim Inqueuedat As String * 20

      'Kill the old copy of the file and create a new copy

      Kill "C:\Cellcont\Queue\Inqueue\V" & Trim(CurrentVMD) & "Inqueu.txt"

      If NumberOftasks = 0 Then 'if the queue is empty then just kill the file
 and exit the procedure
          GoTo ConFigVMD
      End If

      Filenum = FreeFile

      Open "C:\Cellcont\Queue\Inqueue\V" & Trim(CurrentVMD) & "Inqueu.txt" For
 Random Access Write Shared As Filenum Len = 20

      For I = 1 To NumberOftasks
          Inqueuedat = Queue_Status.Inqueue.List(I - 1)
          Put Filenum, I, Inqueuedat
      Next

      Close Filenum
      '*********************************************
```

```
        'now we are also required to tell the correct VMD how many jobs are now
eft in the queue

ConFigVMD:

        'readjust block telling the size of inqueue
        Blocktyp = "V" & VMDNo.Text & "inqusize"
        texts = CStr(NumberOftasks)
        Call SendInfo(Node, Blocktyp, Field, texts)


        'prompt the vmd telling it that all data has been downloaded

        Blocktyp = "V" & VMDNo.Text & "DWNCOMPL"
        texts = "1"
        Call SendInfo(Node, Blocktyp, Field, texts)

        '******THIS IS ANOTHER EXCEL BIT AND IS USED TO RECORD THE JOB LOAD EVE
T AS REQUIRED

        '*TSelect Case Val(VMDNo.Text)

        '*TCase Is = 1

        '*TColumnfocus = 1

        '*TCall WriteToExcel(VMD1Rowpointer, Columnfocus)

        '*TVMD1Rowpointer = VMD1Rowpointer + 1

        'VMD1Rowpointer = 4
        ' VMD2Rowpointer = 4
        ' VMD3Rowpointer = 4
        ' VMD4Rowpointer = 4


        '*TCase Is = 2
        '*TColumnfocus = 8

        '*TCall WriteToExcel(VMD2Rowpointer, Columnfocus)

        '*TVMD2Rowpointer = VMD2Rowpointer + 1


        '*TCase Is = 3

        '*TColumnfocus = 15

        '*TCall WriteToExcel(VMD3Rowpointer, Columnfocus)

        '*TVMD3Rowpointer = VMD3Rowpointer + 1


       '*T Case Is = 4

        '*TColumnfocus = 22
```

```
        '*T Call WriteToExcel(VMD4Rowpointer, Columnfocus)

        '*T VMD4Rowpointer = VMD4Rowpointer + 1


        '*T End Select



End If
'Bidobjectcontrol.Text = 0


Case Is = 4#
'Lets take the circumstance whereby the task has just been
'completed by the VMD - what we are required to do now is to
'effectively take the job out of the vmd and queue up the
'next task to be performed and put that task into the
'outgoing queue at a position governed by the force variable
'of the job.In FixDmacs a program recognises when the last part
'has been completed by comparing the V1RNCC1CCN(For VMD1,Cell Controller 1)
'database block(which details the current cycle number for vmd1
'Cell Controller 1 for the current task) with the V1RNCC1NOP Database Block
'(Which details the number of parts to be made)and when V1RNCC1CCN is greater
'by "1" changes the "V1QCONTROL" database block to "4" which automatically
'changes this textbox to a "4" which loads the just finished job's next task
'into the outgoing queue

'Firstly transfer as much information about the task as possible
'from the vmd status frame

'Length = Len(InComingJob)
'filenum = FreeFile

'Open Defaultpath & VMDJobinfo.Text For Random Access Read Write Shared As file
um Len = Length

'X = X + 1    'Redimension array = 1 deep, 1 dimensional array to read status b
t
'ReDim OutTask(X)
'Get filenum, StartLookingPoint, OutTask(X)

'We must open up the Job Information File which represents the task just
'completed and find out what the next task to be done is - the next task to
'be done will be that indicated by the status field at position 1 of the file
'so this is the field that must be interrogated. Once the file is opened we
'read the status field to a temporary store variable so that the status field
'can be incremented independent of it's current value.We test the value of the
'Current Task to see if the final task has already been done - if it has we
'simply exit the procedure,if not we save the new status field to file and
're-dimension the variable array to take the whole jobfile to memory.Next we
'take the whole job file to memory and test the force variable at the current
'position (Which will be the same value at all positions).If the force variable
'is a "1" the the task must be put at the beginning of the queue - this is
'done in the following steps:
```

```
'1)We decrement the OutGoingQueue.OutStartOfQueue pointer and the re-assess
'it's status in respect to the other queue parameters

'2)We pass the values of the fields representing the current task into the
'beginning of the queue
'eg:Nextqueue(OutGoingQueue.OutStartOfQueue).JobName = OutTask(Current_Task).Jo
Name

'3)The new value of the OutGoingQueue.OutStartOfQueue is passed into the
'OutQueueStart.Text textbox

'If the force variable is not equal to "1" the we must put the task at the end
 of the queue - this is done in the following steps:

'1)We simply put the current task fields into the OutGoingQueue.OutFinishOfQueu
'pointer position in the outgoing queue of the vmd
'eg:Nextqueue(OutGoingQueue.OutFinishOfQueue).JobName = OutTask(Current_Task).J
bName
'- this is because when we add a job to the end of a queue we put the job in
'then jump over the job (increment the end of queue pointer),but when we add
'a job to the beginning we jump back and then add the job

'2)Next we increment the OutGoingQueue.OutFinishOfQueue queue pointer and
'then re-evaluate it's status as compared to the other queueing variables

'3)The new queue pointer status is written to the OutFinishQueue.Text textbox

'Finally the Queue Pointer status and the actual queue status are saved to file


'Current_Task = OutTask(X).Status
'FinalTaskNo = OutTask(X).Finaltask
'OutTask(X).Status = OutTask(X).Status + 1
'If Current_Task = FinalTaskNo + 1 Then
'   MsgBox ("This job is now complete")

  '  Else    'This job has more tasks to do,save new status bit to file
   '         Put filenum, StartLookingPoint, OutTask(X)
    '
     '       ReDim OutTask(FinalTaskNo) 'ReDimension array to suit length of
            'JobInfo file up to the current task
      '      For i = 1 To FinalTaskNo
            'Get all task info into memory
       '     Get filenum, i, OutTask(i)
        '    Next i
         '


          'Check to see what the force variable is - if it is a
          '"1" then put the job at the beginning of the queue
          'otherwise put it at the end of the queue


          'Put the next task to be done in the queue,adjust
          'the outqueue pointers 'and save new outgoing queue
          'parameters to file
```

```
        '   If OutTask(Current_Task).Force = 1 Then
                ' bidjob.Text = OutTask(Current_Task).Force '******This works
             '     BidInformationFilename.Text = OutTask(Current_Task).Jobname
                  'then decrement the outgoing queue OutStartOfQueue pointer
                  'by one to indicate that a job has been added to the beginning
of
                  'the outgoing queue
              '     OutGoingQueue.OutStartOfQueue = OutGoingQueue.OutStartOfQueu
 - 1
                        'Now test the OutQueue Pointer status
               '     If OutGoingQueue.OutStartOfQueue = 0 And OutGoingQueue.OutFi
ishOfQueue = OutGoingEndOfQueue Then
                '               GoSub Fullmess
                 '              OutGoingQueue.OutStartOfQueue = 1
                  '   End If
                   '
                    '  If OutGoingQueue.OutStartOfQueue = 0 And OutGoingQueue.OutFi
ishOfQueue < OutGoingEndOfQueue Then
                     '          OutGoingQueue.OutStartOfQueue = OutGoingEndOfQueue
                  '    End If

                   '   If OutGoingQueue.OutStartOfQueue = OutGoingQueue.OutFinishOf
ueue Then
                    '           GoSub Fullmess
                     ' End If


                     'Put task at the beginning of the outgoing queue
    '                Nextqueue(OutGoingQueue.OutStartOfQueue).Jobname = OutTask(C
rrent_Task).Jobname
     '               Nextqueue(OutGoingQueue.OutStartOfQueue).InFoFilename = OutT
sk(Current_Task).InFoFilename
      '              Nextqueue(OutGoingQueue.OutStartOfQueue).Bid_No_Of_Parts = Ou
tTask(Current_Task).NoOfCycles
       '             Nextqueue(OutGoingQueue.OutStartOfQueue).Bidsetuptime = OutT
sk(Current_Task).setuptime
        '            Nextqueue(OutGoingQueue.OutStartOfQueue).BidMachTyp = OutTas
(Current_Task).Machinetype
         '           Nextqueue(OutGoingQueue.OutStartOfQueue).Bidcycletime = OutTa
sk(Current_Task).Estcycletime
          '          Nextqueue(OutGoingQueue.OutStartOfQueue).BidTool1Required =
utTask(Current_Task).Tool1required
                    'Nextqueue(OutGoingQueue.OutStartOfQueue).BidTool2Required = O
utTask(Current_Task).Tool2required
                    'Nextqueue(OutGoingQueue.OutStartOfQueue).BidTool3Required = O
utTask(Current_Task).Tool3required
                    'Nextqueue(OutGoingQueue.OutStartOfQueue).BidTool4Required = O
utTask(Current_Task).Tool4required
           '        Nextqueue(OutGoingQueue.OutStartOfQueue).BidloadUnloadtime =
OutTask(Current_Task).Estloadunloadtime
            '       Nextqueue(OutGoingQueue.OutStartOfQueue).Force = OutTask(Cur
ent_Task).Force
```

```
                'Update visual representation of start of Nextqueue pointer
            '     Outqueuestart.Text = OutGoingQueue.OutStartOfQueue

        ' Else
                'The job to be done is not urgent therefore put it at the
                'end of the outgoing queue

          '       Nextqueue(OutGoingQueue.OutFinishOfQueue).Jobname = OutTask(
urrent_Task).Jobname
            '     Nextqueue(OutGoingQueue.OutFinishOfQueue).InFoFilename = Out
ask(Current_Task).InFoFilename
              '   Nextqueue(OutGoingQueue.OutFinishOfQueue).Bid_No_Of_Parts =
tTask(Current_Task).NoOfCycles
              '   Nextqueue(OutGoingQueue.OutFinishOfQueue).Bidsetuptime = Out
ask(Current_Task).setuptime
             ''    Nextqueue(OutGoingQueue.OutFinishOfQueue).BidMachTyp = OutT
sk(Current_Task).Machinetype
               ' Nextqueue(OutGoingQueue.OutFinishOfQueue).Bidcycletime = Out
 sk(Current_Task).Estcycletime
                'Nextqueue(OutGoingQueue.OutFinishOfQueue).BidTool1Required =
OutTask(Current_Task).Tool1required
                'Nextqueue(OutGoingQueue.OutFinishOfQueue).BidTool2Required =
utTask(Current_Task).Tool2required
                'Nextqueue(OutGoingQueue.OutFinishOfQueue).BidTool3Required =
ttTask(Current_Task).Tool3required
                'Nextqueue(OutGoingQueue.OutFinishOfQueue).BidTool4Required =
OutTask(Current_Task).Tool4required
    '             Nextqueue(OutGoingQueue.OutFinishOfQueue).BidloadUnloadtime
 OutTask(Current_Task).Estloadunloadtime
     '            Nextqueue(OutGoingQueue.OutFinishOfQueue).Force = OutTask(Cu
rent_Task).Force
       '
        '           OutGoingQueue.OutFinishOfQueue = OutGoingQueue.OutFinishOfQu
ue + 1

              'Test new Queue Status
       '                      If OutGoingQueue.OutFinishOfQueue > OutGoingEnd(
Queue And OutGoingQueue.OutStartOfQueue > OutGoingBeginOfQueue Then
                            'Wrap Queue around to beginning of array as firs
position in array is empty
        '                     OutGoingQueue.OutFinishOfQueue = OutGoingBeginO
ueue
         '                    End If

          '                   If OutGoingQueue.OutFinishOfQueue > OutGoingEnd(
fQueue And OutGoingQueue.OutStartOfQueue = OutGoingBeginOfQueue Then
            '                  GoSub Fullmess
          '                   End If

              '               If OutGoingQueue.OutFinishOfQueue = OutGoingQue
.OutStartOfQueue Then
            '                 GoSub Fullmess
              '               End If
                ' OutFinishQueue.Text = OutGoingQueue.OutFinishOfQueue
      '    End If
```

```
                    'and now save latest Outqueue pointer status to file
          '           filenum = FreeFile
           '          Filelength = Len(OutGoingQueue)
            '         Fname = "OPV" & VMDNo.Text & Nodename.Text & ".Txt"
             '        position = 1
              '        Open Defaultpath & Fname For Random As filenum Len = Filel
ngth
                '     Put filenum, position, OutGoingQueue
                      '    Close filenum

                    'Now save latest Out queue state to file
          '           filenum = FreeFile
           '          position = 1
            '         Fname = "OQV" & VMDNo.Text & Nodename.Text & ".Txt"
             '        Filelength = Len(OutGoingQue)
              '       Open Defaultpath & Fname For Random As filenum Len = Filele
gth
                '     position = 1
   '                  Do Until position = 101
    '                 Put filenum, position, Nextqueue(position)
       '                 position = position + 1
         '           Loop
          '          Close filenum


'End If
'Close filenum

Case Is = 5#

'This particular procedure is designed so that when a Cell Controller wants
'to hold an auction and the job to be auctioned is on the top of this VMD'S
'outgoing queue the cell controller places a 5 in this textbox via the
'"V" & VmdNo.text & "QCONTROL" Cell Controller database block which has an
'automatic link to this textbox - this will send the bidding objects from
'the top of this VMD'S outgoing queue to the Factory Controllers bidding
'blocks.The Cell Controller's database blocks of type "V" & VmdNo.text &
'"QCONTROL"  are used when transfer of data is required in terms of only
'one vmd eg:movement of bidding objects to the Factory Controller whereas
'the bidcontrol block is used for control of all the vmd's within the one
'Cell Controller eg: Retrieval of bidding objects from Factory Controller

'Node = Nodename.Text
'Field = "A_CV"
'Blocktyp = "BidJob"
'BidObj = Nextqueue(OutGoingQueue.OutStartOfQueue).Jobname
'Call SendInfo(Node, Blocktyp, Field, BidObj)

'Blocktyp = "BidForce"
'BidObj = Nextqueue(OutGoingQueue.OutStartOfQueue).Force
'Call SendInfo(Node, Blocktyp, Field, BidObj)

'Blocktyp = "BidInfo"
'BidObj = Nextqueue(OutGoingQueue.OutStartOfQueue).InFoFilename
'Call SendInfo(Node, Blocktyp, Field, BidObj)
```

```
 Blocktyp = "BidNoParts"
'BidObj = Nextqueue(OutGoingQueue.OutStartOfQueue).Bid_No_Of_Parts
'Call SendInfo(Node, Blocktyp, Field, BidObj)

'Blocktyp = "BidSupTm"
'BidObj = Nextqueue(OutGoingQueue.OutStartOfQueue).Bidsetuptime
 Call SendInfo(Node, Blocktyp, Field, BidObj)

'Blocktyp = "BidCycTm"
 BidObj = Nextqueue(OutGoingQueue.OutStartOfQueue).Bidcycletime
'Call SendInfo(Node, Blocktyp, Field, BidObj)

 Blocktyp = "BidLoadUn"
 BidObj = Nextqueue(OutGoingQueue.OutStartOfQueue).BidloadUnloadtime
'Call SendInfo(Node, Blocktyp, Field, BidObj)

 increment OutGoingQueue.OutStartOfQueue pointer to indicate that the job
'has been removed from the OutGoing Queue

 OutGoingQueue.OutStartOfQueue = OutGoingQueue.OutStartOfQueue + 1


          'Test pointer status
  '         If OutGoingQueue.OutStartOfQueue = OutGoingQueue.OutFinishOfQueue Th
n
  '         GoSub EmptyMessage
    '       End If

     '       If OutGoingQueue.OutStartOfQueue = (OutGoingEndOfQueue + 1) And OutG
oingQueue.OutFinishOfQueue = OutGoingBeginOfQueue Then
      '       GoSub EmptyMessage
       '      OutGoingQueue.OutStartOfQueue = OutGoingBeginOfQueue
        '     End If

         '    If OutGoingQueue.OutStartOfQueue = (OutGoingEndOfQueue + 1) And OutG
oingQueue.OutFinishOfQueue > OutGoingBeginOfQueue Then
         '    OutGoingQueue.OutStartOfQueue = OutGoingBeginOfQueue
         '    End If

'update outqueuepointer on screen
 Outqueuestart.Text = OutGoingQueue.OutStartOfQueue

'Now we are required to save the new outgoing queue pointer status to file

 Filenum = FreeFile
'Filelength = Len(OutGoingQueue)
 Fname = "OPV" & VMDNo.Text & Nodename.Text & ".Txt"
'position = 1
'Open Defaultpath & Fname For Random As Filenum Len = Filelength
 Put Filenum, position, OutGoingQueue
 Close Filenum

End Select
```

```
GoTo Skip
EmptyMessage:
Msg = "You Have just removed the last job from the queue,remove no more"
Typ = MB_ICONEXCLAMATION
title = "Queue Status"
QueueFull = MsgBox(Msg, Typ, title)
Return

Queue_is_empty:
Msg = "The queue is empty,donot attempt to load a task"
Typ = MB_ICONEXCLAMATION
title = "Queue Status"
QueueEmp = MsgBox(Msg, Typ, title)
Return

Fullmess:
FullOutQFlag = 1
Msg = "The OutGoing Queue of Cell Controller " & Nodename.Text & " VMD Number "
& VMDNo.Text & "is completely full, so this job cannot be added"
title = "OutQueue Status"
Typ = MB_ICONEXCLAMATION
Dummy = MsgBox(Msg, Typ, title)
Return

Skip:

Exit Sub


checkthis:

If Err = 9 Then
Resume Next
    Else
    If Err = 70 Then
    MsgBox ("The error is permission denied")
        Else
        MsgBox ("The error is " & Error(Err) & "is at the " & Trim(Fileconcerned
) & "File")
        Resume Next
    End If
End If


End Sub

Sub BidWin_Change ()

Dim ReceptQueue As Inqueue
Dim Job As String * 20
Dim JobIn As String * 20
Dim Force As String * 20
Dim Msg As String * 113
Dim Typ As Integer
```

```
EWQUEUE.FRM - 38

Dim title As String * 14
 im Dummy As Integer
Dim Filenum As Variant
Dim Filelength As Variant
 im Fname As String * 30
Dim NoParts As String * 10
Dim Bidsetuptime As String * 10
 im Bidcycletime As String * 10
Dim BidloadUnloadtime As String * 10
Dim InComingQueueEndNewTaskCost As Single
 im Incomingqueuestartnewtaskcost As Single
Dim cost As Single
Dim Columnfocus As Integer


 This is Part Of the Queue Control application. When this
 Textbox turns to a 1 it means that this VMD has won the
'Auction and therefore gets the Bidding objects from the
 Factory Controller,tests the Force variable and if the
 variable is a 1 then the job is put at the beginning of
'the queue otherwise the job is put at the end of the queue


On Error GoTo Checktheerr

 MsgBox (Bidwin.Text)
If CStr(Bidwin.Text) Like ("*E230*") Or CStr(Bidwin.Text) Like ("*E175*") Then
   Exit Sub
 nd If

If CInt(Bidwin.Text) = 0 Then
   Exit Sub
End If


When this textbox changes to a 1 it is telling the queueing application to get
the bidding
 information from the bidfile and to put it in the VMD's incoming queue.

'Type Bid_data_To_Be_sent
 Jobname As String * 10
 Machinetype As Single
'EstCycletime As Integer
 Setuptime As Integer
 LoadUnloadtime As Integer
''NoofCycles As Integer
 Tool1required As Single
 Activate As Integer
'Force as Integer
'End Type

Dim ThisbidFile As Bid_Data_to_Be_sent
 im Thisfilename As String
```

```
If CInt(Bidwin.Text) = CInt(VMDNo.Text) Then 'if this vmd has won the auction
en get it to queue
'the job
    'denote a filenumber for the new file.
    Filenum = FreeFile
    Filelength = Len(ThisbidFile)
    Thisfilename = Netdrive & "Factory\Bidding\Bidfile.bdf"
    'MsgBox (Thisfilename)

    'open the bidfile which has all the information required for the ingoing q
ue
    Open Netdrive & "Factory\Bidding\Bidfile.bdf" For Random Access Read Shared
As Filenum Len = Filelength

    'now we are required to get the relevant information from the file
    Get Filenum, 1, Thisbidding(1)


    Job = Thisbidding(1).Jobname

    '*****This bit deals with the results form
    Res_Form.Res_Job.AddItem Trim(Thisbidding(1).Jobname)'pass the name of the
ob to this textbox so it can be poked to excell
    Res_Form.Res_Event.AddItem "Add" 'tell excell that we are adding a task
    Res_Form.Res_time.AddItem Time$
    '*****************************************************************************
**************

    'MsgBox (Job)
    Force = CStr(Thisbidding(1).Force)
'    MsgBox (Force)
    JobIn = Trim(Thisbidding(1).Jobname) & ".iff"
 '   MsgBox (JobIn)
    NoParts = CStr(Thisbidding(1).NoOfCycles)
  '  MsgBox (NoParts)
    Bidsetuptime = CStr(Thisbidding(1).setuptime)
   ' MsgBox (Bidsetuptime)
    Bidcycletime = CStr(Thisbidding(1).Estcycletime)
  '  MsgBox (Bidcycletime)
    BidloadUnloadtime = CStr(Thisbidding(1).Loadunloadtime)
   ' MsgBox (BidloadUnloadtime)

    Close Filenum


'Node = "FIX"
'Tags = "BidJob"
'Field = "A_CV"
'job = eda_requestinfolink(Node, Tags, Field)

'Tags = "BidForce"
'Force = eda_requestinfolink(Node, Tags, Field)

'Tags = "BidInfo"
'JobIn = eda_requestinfolink(Node, Tags, Field)
```

```
'Tags = "BidNoparts"
NoParts = eda_requestinfolink(Node, Tags, Field)

'Tags = "Bidsuptm"
BidSetUpTime = eda_requestinfolink(Node, Tags, Field)

'Tags = "BidCycTm"
BidCycleTime = eda_requestinfolink(Node, Tags, Field)

'Tags = "Bidloadun"
BidLoadUnLoadTime = eda_requestinfolink(Node, Tags, Field)

'Type Inqueue
Jobname As String * 20
InFoFilename As String * 20
'Bid_No_Parts As String * 10
BidSetUpTime As String * 10
BidLoadUnLoadTime As String * 10
'BidCycleTime As String * 10
End Type

    Dim NumberOftasks As Integer
    Dim Qsize As Integer
    Dim Inqueuedata As String * 20
    Dim I As Integer
    Dim Incomingqueuenewtaskcost As Single
    Dim CurrentVMD As String

    If Val(Force) = 1 Then
    'we have to put the task not at the top of the listbox but one down from the
top because the
    'task at the top is reserving tools (but we only put it one down if there is
already a task at
    'the top of the queue - so the first thing that we are required to do is to
see whether or not
    'there is a task at the top of the queue

        If Len(Trim(Queue_Status.Inqueue.List(0))) = 0 Then 'this is the first j
ob to be added to the
            'queue therefore we can put it at the top of the queue, and not one
down.
            Queue_Status.Inqueue.AddItem Job
            Else 'put the job one down in the queue
                Queue_Status.Inqueue.AddItem Job, 1
        End If
    Else
        'put it at the bottom of the list
        Queue_Status.Inqueue.AddItem Job, CInt(Queue_Status.Inqueue.ListCount)
    End If

    'get the number of items in the listbox
    NumberOftasks = Queue_Status.Inqueue.ListCount

    'MsgBox CStr("THE NUMBER OF TASKS ARE " & NumberOftasks)
    'now we are also required to tell the correct VMD how many jobs are now left
in the queue
```

```
    Field = "A_CV"
    Node = Trim(Nodename.Text)
    Blocktyp = "V" & VMDNo.Text & "inqusize"
    texts = CStr(NumberOftasks)
    Call SendInfo(Node, Blocktyp, Field, texts)


    '    MsgBox (CStr(Numberoftasks))
    'Now open up the name of a file so that we can save this to the file
    Filenum = FreeFile

    CurrentVMD = VMDNo.Text
'    MsgBox (CurrentVMD)
    'Call Updatequeue(Filenum, CurrentVMD, NumberOfTasks)


    Open "C:\Cellcont\Queue\Inqueue\V" & Trim(VMDNo.Text) & "Inqueu.txt" For Ra
dom Access Read Write Shared As Filenum Len = 20

    For I = 1 To NumberOftasks
        Inqueuedata = Queue_Status.Inqueue.List(I - 1)
'        MsgBox (Inqueuedata)
        Put Filenum, I, Inqueuedata
    Next

    Close Filenum


    'If Force is not equal to "1" the put job at the end of the Queue
        '
        '    Currentqueue(Incomingqueue.InfinishofQueue).Jobname = Job
        ''MsgBox (Currentqueue(IncomingQueue.InFinishOfQueue).Jobname)
        '   Currentqueue(Incomingqueue.InfinishofQueue).InFoFilename = JobIn
        ''''''MsgBox (Currentqueue(IncomingQueue.InFinishOfQueue).InFoFilename)
        '   Currentqueue(Incomingqueue.InfinishofQueue).Bid_No_Parts = Noparts
        ''''MsgBox (Currentqueue(IncomingQueue.InFinishOfQueue).Bid_No_Parts)
        '  Currentqueue(Incomingqueue.InfinishofQueue).Bidsetuptime = Bidsetupti
me
      ''''''''''MsgBox (Currentqueue(IncomingQueue.InFinishOfQueue).Bidsetupti
e)
        ' Currentqueue(Incomingqueue.InfinishofQueue).Bidcycletime = Bidcycleti
e
        '''''''MsgBox (Currentqueue(IncomingQueue.InFinishOfQueue).Bidcycletime
)
        '  Currentqueue(Incomingqueue.InfinishofQueue).BidloadUnloadtime = Bidl
adUnloadtime
        '''''MsgBox (Currentqueue(IncomingQueue.InFinishOfQueue).BidloadUnloadt
ime)

    'Recalculate Current Queue Cost
    Incomingqueuenewtaskcost = (CDbl(NoParts) * (CDbl(Bidcycletime) + 2 * CDbl(
idloadUnloadtime))) + CDbl(Bidsetuptime)
'    MsgBox (CStr(IncomingQueueNewtaskcost))
    cost = Val(QueueCost.Text) + Incomingqueuenewtaskcost
    QueueCost.Text = cost
```

```
    '**Update the results form to show the new queue size
    Res_Form.Res_Cost.AddItem Trim(QueueCost.Text)

'    MsgBox (CStr(cost))
    Call updateqcost

    'this is fine but we must also send a message to tell the Factory Controller
that the objects have
    'been recieved and that everything is sweet.
    Node = "Fact"
    Tags = "BidAckNowl"
    Field = "A_CV"
    texts = "1"

    Call SendInfo(Node, Tags, Field, texts)

    'we are also required to reset the bidwin block otherwise everytime this for
 loads up
    ' a new job will be automatically awarded also if another job is awarded to
this vmd it will
    'not respond as no change event will occur.
    Node = Trim(CStr(Nodename.Text))
    Tags = "Bidwin"
    Field = "A_CV"
    texts = "0"
    Call SendInfo(Node, Tags, Field, texts)

    '*******IN HERE GOES THE DATA COLLECTION PART THAT COLLECTS THE DATA AND PUT
 IT IN THE
    '*******SPREADSHEET FOR THIS VMD*****************************************
**********

    'we are required to find out what time it is so that we can shunt this infor
ation to the
    'spreadsheet
    'Thistime.text = Time

    'we are also required to findout what the current date is as well
    'Thisdate.text = Date


    'the second thing that we are required to do is to find out what VMD we are
orking with and then
    'set the column accordingly
'T    Select Case Val(CurrentVMD)

'*T   Case Is = 1

'T    Columnfocus = 1

'*T   Call WriteToExcel(VMD1Rowpointer, Columnfocus)

'*T   VMD1Rowpointer = VMD1Rowpointer + 1

    'VMD1Rowpointer = 4
  ' VMD2Rowpointer = 4
```

```
   ' VMD3Rowpointer = 4
   ' VMD4Rowpointer = 4


'*T    Case Is = 2
 '*T   Columnfocus = 8

'*T    Call WriteToExcel(VMD2Rowpointer, Columnfocus)

'*T    VMD2Rowpointer = VMD2Rowpointer + 1


 '*T   Case Is = 3

'*T    Columnfocus = 15

'*T    Call WriteToExcel(VMD3Rowpointer, Columnfocus)

 '*T   VMD3Rowpointer = VMD3Rowpointer + 1


'*T    Case Is = 4

'*T    Columnfocus = 22

 '*T   Call WriteToExcel(VMD4Rowpointer, Columnfocus)

 '*T   VMD4Rowpointer = VMD4Rowpointer + 1


'*T    End Select

    Else
        Exit Sub
End If

Exit Sub
Checktheerr:

If Err = 55 Then
    MsgBox ("the error is file is already open")
    Resume Next
    Else
    MsgBox ("The error is " & Error(Err))
    Resume Next
End If


End Sub


Sub Exit_Click ()
Unload Newqueue
End Sub

Sub Form_Click ()
```

```
        Node = "fact"
        Blocktyp = "updateMTBC" 'updates the Mean Time between Completion
        Field = "A_CV"
        texts = "1"
        Call SendInfo(Node, Blocktyp, Field, texts)

End Sub

Sub Form_Load ()
'Sim_Start.Text = "1"
 This application is the Queueing Application

'Queueing Application:
 Designed and Written by Matthew John Jolly, Masters Student
'Massey University
'Palmerston North
 New Zealand
Dim rap As Integer
Dim Koop As Integer
 im Msg As String * 80
Dim I As Integer
Dim response As Integer
 im Repeat As String * 6
Dim Repair As String * 23
Dim ans As Integer
 im Wrong As Variant
Dim Evaluate As Integer
Dim again As String * 5
 im Loopy As Integer
 im Filenum As Variant
 im Filenumb As Variant
  m ReceptQueue As Inqueue
 im Filelength As Variant
Dim Fname As String * 30
 im OutGoingQue As Outqueue
  m Begin As Integer
Dim Newcost As Single
Dim InFinishOfQueueTemp As Integer
 im InStartOfQueuetemp As Integer
 im temp As Integer
 im nodenum As Integer
 m First As Integer
 m Last As Integer
 im Thisnode As Integer
 m nexts  As Variant
 im Count As Integer
Dim Noderecord() As nodesrecord
 im Noderecordlength As Variant
Dim tryagain As Variant
Dim Nodrecord As nodesrecord
 m Mess As String * 100
Dim title As String * 40
Dim DgDef As Integer
 m Nodrec As noodesrecord
Dim j As Integer
Dim cost As Single
```

```
On Error GoTo Checkerror


'Sim_Start.Text = 1



Firstup = 1
'This application handles the queueing of jobs to be done
'as they arrive at a Cell Controller's VMD . It is also
'responsible for the transfer of job information
Queuestart.Text = 1
QueueFin.Text = 1


'******** CONFIGURE LOCAL NODE ***********
'Check to see if node has been configured before
Filelength = Len(Nodrec)
Filenum = FreeFile
Open "C:\Cellcont\queue\config\Noderec.txt" For Random Access Read Write Shared
As Filenum Len = Filelength
Get Filenum, 1, Nodrecs(1)


If Nodrecs(1).Index = 1 Then
'This node has had at least one queuing application configured for one
'of the vmd's therefore transfer the name present in this record into the
'nodename textbox
Nodename.Text = Nodrecs(1).Nodename

        Else
        'This node has had no queuing applications configured for it therefore
        'we must state the name of the node

        'Open the file containing the names of all nodes being used
        ReDim Noderecord(1)
        Filenumb = FreeFile
        Noderecordlength = Len(Nodrecord)
        Open "C:\Cellcont\queue\config\Nodescon.txt" For Random Access Read Writ
e Shared As Filenumb Len = Noderecordlength
        Get Filenumb, 1, Noderecord(1)

              If Noderecord(1).Recordsize = 0 Then
              Noderecord(1).Recordsize = 1
              End If

        ReDim Preserve Noderecord(Noderecord(1).Recordsize)
        For I = 1 To Noderecord(1).Recordsize
        Get Filenumb, I, Noderecord(I)
        Next I



        'Request the name of the node being used
        'Call Function which asks user to enter  NodeName
```

```
again:
        Msg = "Please Enter the name of node being used"
        NodeNam = CheckNodeName(Msg)
        'Check to see if valid Node Name entered
        If Trim(NodeNam) Like "c##" Then
             Nodename.Text = UCase(NodeNam)
             'and send the nodename to the "CCNAME" FixDmacs Text Block

             'Node = NodeName.Text
             'BlockTyp = "CCNAME"
             'Field = "A_CV"
             'texts = NodeName.Text
             'Call SendInfo(Node, BlockTyp, Field, texts)

             Else
             'If NodeName not valid Re-Enter
             MsgBox ("Invalid Nodename Please Re-Enter")
             GoTo again
             End If

        I = 1
        For I = 1 To Noderecord(1).Recordsize + 1
             If Nodename.Text = Noderecord(I).Nodename Then
             Msg = "This nodename already used input another"
             MsgBox (Msg)
             rap = rap + 1
               If rap = 3 Then
               Unload Newqueue
               Exit Sub
               End If
             GoTo again
             End If
        Next I
        'Now save new nodename to file
        'Increment placeholder to next position

        Noderecord(1).Recordsize = Noderecord(1).Recordsize + 1
        Noderecord(Noderecord(1).Recordsize).Nodename = Nodename.Text

        j = 1
        For j = 1 To Noderecord(1).Recordsize
        Put Filenumb, j, Noderecord(I)
        Next j
        Close Filenumb


        'Now must save this nodename to this node's private "noderecord.txt"
        'file
        Nodrecs(1).Nodename = Nodename.Text
        Nodrecs(1).Index = 1
        Put Filenum, 1, Nodrecs(1)


End If
Close Filenum
```

```
'and write the name to a file on a local directory so that other applications o
 this cell can
'get the name
Filenum = FreeFile
Open "c:\Cellcont\Cellname.txt" For Output Shared As Filenum Len = 10
Print #Filenum, Nodename.Text
Close Filenum


'************ ASK FOR THE NUMBER OF THE VMD *************
VMDNo.Text = "4"


'************ OPEN THE INCOMING QUEUE FILE *************
'Opening the file representing the ingoing Queue and
'loading the jobs to be done into the vmd's ingoing
'queue

'Filenum = FreeFile
'Filelength = Len(ReceptQueue)
'Fname = "InV" & VMDNo.Text & Nodename.Text & ".Txt"
'Open "c:\Cellcont\queue\inqueue\" & Fname For Random As Filenum Len = Fileleng
h
 '   If position = 0 Then position = 1
  ' Do Until position = 101
   '      Get Filenum, position, Currentqueue(position)
    '     position = position + 1
   'Loop
'MsgBox (CStr(Currentqueue(100).Jobname))
'Close Filenum

'********* OPEN THE INCOMING QUEUE POINTERS FILE *******
'position = 1
'Filenum = FreeFile
'Filelength = Len(Incomingqueue)
'Fname = "IPV" & VMDNo.Text & Nodename.Text & ".Txt"
'Open "c:\Cellcont\queue\inqueue\" & Fname For Random As Filenum Len = Filelengt
h
'Get Filenum, position, Incomingqueue


'When the system starts up for the very first time the
'Incoming queue pointers will equal zero - this will also
'occur if the VMD's incoming queue file is deleted via
'D.O.S.,the File Manager or XTGold. Therefore as the queue
'has been configured to start at "1" then we must set the
'InComingQueue.InStartOfQueue and InComingQueue.InFinishOfQueue
'variables to equal "1".

'If Incomingqueue.Instartofqueue = 0 Then
'Incomingqueue.Instartofqueue = 1
'End If
'If Incomingqueue.InfinishofQueue = 0 Then
'Incomingqueue.InfinishofQueue = 1
'End If
```

```
 Queuestart.Text = Incomingqueue.Instartofqueue
'QueueFin.Text = Incomingqueue.InfinishofQueue

 Close Filenum
'Exit Sub
'*************** Calculate Current Queue Cost ************
 This Loop calculates the cost of the incoming queue

'*****************************************
 Dim tempfinish As Integer
 Dim tempstart As Integer

 If Incomingqueue.Instartofqueue = Incomingqueue.InfinishofQueue Then
 '   Cost = 0
  '  Else
   '     If Incomingqueue.InfinishofQueue > Incomingqueue.Instartofqueue Then
    '        tempfinish = Incomingqueue.InfinishofQueue
     '       tempstart = Incomingqueue.Instartofqueue
      ''       Do Until tempfinish = tempstart
       '          NewCost = ((CDbl(Currentqueue(tempfinish).Bid_No_Parts)) * ((CI
bl(Currentqueue(tempfinish).Bidcycletime) + 2 * (CDbl(Currentqueue(tempfinish).B
dloadUnloadtime)))) + CDbl(Currentqueue(tempfinish).Bidsetuptime)
         '       Cost = NewCost + Cost
          '      tempfinish = tempfinish - 1
           ' Loop
           ' Else
         '       If Incomingqueue.Instartofqueue > Incomingqueue.InfinishofQueue
 Then
          '          tempfinish = Incomingqueue.InfinishofQueue
           '         MsgBox (CStr(tempfinish))
            '        tempstart = Incomingqueue.Instartofqueue
             '       MsgBox (CStr(tempstart))
              '      Do Until tempstart = tempfinish
               '         NewCost = ((CDbl(Currentqueue(tempstart).Bid_No_Parts))
 * ((CDbl(Currentqueue(tempstart).Bidcycletime) + 2 * (CDbl(Currentqueue(tempsta
rt).BidloadUnloadtime)))) + CDbl(Currentqueue(tempstart).Bidsetuptime)
                '        MsgBox (CStr(tempstart))
                 '       MsgBox (CStr(tempfinish))
                  '      Cost = NewCost + Cost
                   '     MsgBox (CStr(Cost))
                    '    tempstart = tempstart - 1
                    'L'oop
                'End If
        End If
 End If


 '*****************************************

'load the queueing form
 oad Queue_Status
```

```
Dim NumberOftasks As Integer
Dim Inqueuedata As String * 20
Dim Incomingqueuenewtaskcost As Single
Dim position As Integer

'open up the name of the incoming queue file and load the queue data into the i
coming queue
'listbox

Filenum = FreeFile
Open "C:\Cellcont\Queue\Inqueue\V" & Trim(VMDNo.Text) & "Inqueue.txt" For Rando
 Access Read Write Shared As Filenum Len = 20
'MsgBox ("C:\Cellcont\Queue\Inqueue\V" & Trim(VMDNo.Text) & "Inqueue.txt")
position = 0

Queue_Status.Inqueue.Clear

Do Until EOF(Filenum)
    position = position + 1
    Get Filenum, position, Inqueuedata
 '   MsgBox (Inqueuedata)
    If Len(Trim(Inqueuedata)) = 20 Then 'there is a space in the file, donot add
        GoTo Nextname
    End If
    Queue_Status.Inqueue.AddItem Trim(Inqueuedata)
Nextname:
Loop

'set a variable to the number of jobs that are in this VMD's ingoing queue
Numberofjobs = Queue_Status.Inqueue.ListCount

'now tell the vmd how many jobs there are to do

'Dim Node As String * 15
'Dim Tags As String * 15
'Dim Field As String * 15
'Dim Texts As String * 6

Node = Nodename.Text
Field = "A_CV"
Tags = "V" & VMDNo.Text & "INQUSIZE"
texts = CStr(Numberofjobs)

Call SendInfo(Node, Tags, Field, texts)


Thispoint:

Close Filenum

'Declare queuecost file variables
Dim recordlength As Integer
Dim QueueCosts As VMDQueueCost
Dim filename As String
Dim Directoryname As String
Dim Thislength As Integer
```

```
'Type VMDQueueCost
    '  Cellname As String * 8
  '  VMD1_Queue_Cost As Single
   ' VMD2_Queue_Cost As Single
    ' VMD3_Queue_Cost As Single
    ' VMD4_Queue_Cost As Single
 End Type

Filenum = FreeFile 'denote a freefile number
ecordlength = Len(QueueCosts) 'get array storage size required

Directoryname = "c:\Cellcont\Queue\" 'set directory name

ilename = "Queucost.cst" 'set filename

 open the file
pen Trim(Directoryname) & Trim(filename) For Random Access Read Write Shared As
 Filenum Len = recordlength

 et Filenum, 1, Thisqueuecost(1) 'get the array into memory

 elect Case Val(VMDNo.Text)

Case Is = 1 'if this queueing application belongs to vmd1 then
 ost = Thisqueuecost(1).VMD1_Queue_Cost

Case Is = 2  'if this queueing application belongs to vmd2 then
ost = Thisqueuecost(1).VMD2_Queue_Cost

Case Is = 3 'if this queueing application belongs to vmd3 then
ost = Thisqueuecost(1).VMD3_Queue_Cost

Case Is = 4 'if this queueing application belongs to vmd4 then
ost = Thisqueuecost(1).VMD4_Queue_Cost

nd Select


 f Matchposition = 0 Then
    ' a record has not been found that matches the current node therefore we hav
  to increase the
    'size of the array by 1 and then add the new record.
  '   MsgBox (CStr(Fileisempty))
  '  MsgBox (CStr(Thislength))

'    If Fileisempty = 1 And Thislength = 1 Then
       'the file is new and has only just been opened
        Thisqueuecost(1).NumberOfRecords = Thislength
  '      Else
   '         Thisqueuecost(1).NumberOfRecords = Thisqueuecost(1).NumberOfRecords
 1
    '         Thislength = Thisqueuecost(1).NumberOfRecords
    'End If

    'MsgBox (CStr(Thislength))
```

```
    'ReDim Preserve Thisqueuecost(Thislength)
    'MsgBox (CStr(Thisqueuecost(1).NumberOfRecords))
    'If Fileisempty = 1 Then
     '   Thisqueuecost(1).NumberOfRecords = 1
      '  Fileisempty = 0
    'End If

    'Thisqueuecost(Thislength).Cellname = Nodename.Text


    'Else
        'it is not a new record so just use the matchposition
     '   Thisqueuecost(Matchposition).Cellname = Nodename.Text
      '  Thisqueuecost(Matchposition).NumberOfRecords = Thislength
       ' Select Case Val(VMDNo.Text)

      '  Case Is = 1
       ' Thisqueuecost(Matchposition).VMD1_Queue_Cost = Cost
       '
       ' Case Is = 2
        'Thisqueuecost(Matchposition).VMD2_Queue_Cost = Cost

        'Case Is = 3
        'Thisqueuecost(Matchposition).VMD3_Queue_Cost = Cost

        'Case Is = 4
        'Thisqueuecost(Matchposition).VMD4_Queue_Cost = Cost
        '
        'End Select

'End If

'now put everything back in the file
'For i = 1 To Thislength
 '   Put Filenum, i, Thisqueuecost(i)
'N'ext

QueueCost.Text = cost
Skipit:
Close Filenum

'********* OPEN THE OUTGOING QUEUE FILE ********
'This procedure opens the outgoing queue file which contains
'the names of jobs to be done and bidding objects for the
'current tasks to be bidded for and loads them into the
'outgoing queue of the current VMD.

'Filenum = FreeFile
'Filelength = Len(OutGoingQue)
'Fname = "OQV" & VMDNo.Text & Nodename.Text & ".Txt"
'Open "c:\Cellcont\queue\outqueue\" & Fname For Random As Filenum Len = Fileleng
th
'Do Until Position = 101
'Get Filenum, Position, Nextqueue(Position)
'Position = Position + 1
'Loop
```

```
NEWQUEUE.FRM - 52

'Close Filenum

'********** OPEN THE OUTGOING QUEUE POINTERS FILE *******
'Position = 1
'Filenum = FreeFile
'Filelength = Len(OutGoingQueue)
'Fname = "OPV" & VMDNo.Text & Nodename.Text & ".Txt"
Open "c:\Cellcont\queue\outqueue\" & Fname For Random As Filenum Len = Filelen
gth

Get Filenum, Position, OutGoingQueue
Close Filenum

When the system starts up for the very first time the
Incoming queue pointers will equal zero - this will also
'occur if the VMD's incoming queue file is deleted via
'D.O.S.,the File Manager or XTGold. Therefore as the queue
has been configured to start at "1" then we must set the
'InComingQueue.InStartOfQueue and InComingQueue.InFinishOfQueue
'variables to equal "1".
If OutGoingQueue.OutStartOfQueue = 0 Then
'OutGoingQueue.OutStartOfQueue = 1
'End If

'If OutGoingQueue.OutFinishOfQueue = 0 Then
'OutGoingQueue.OutFinishOfQueue = 1
End If

'Outqueuestart.Text = OutGoingQueue.OutStartOfQueue
OutFinishQueue.Text = OutGoingQueue.OutFinishOfQueue

'** SET UP AUTOMATIC LINKS WITH FACTORY & CELL CONTROLLER **

We are required to setup the automatic dde links between
'the Queueing application and the Factory Controller and
between the Queueing application and the Cell Controller.
These links are required so as to inform the user of the
'state of the VMD and for control purposes

Create a link between the bidwin textbox and the "bidwin" database block
Call CommDDELink(Bidwin, Nodename.Text & "." & "bidwin" & "." & "A_CV")


Create link between "CNC Device Status" Textbox and
Call CommDDELink(VMDBusYIdle, NodeName.text & "." & "V" & VMDNo.text & "MACHBUS
Y" & "." & "A_CV")


'Create link between "BIDWIN" Textbox and
'"VMD1WIN" Factory Controller Database block

'Call CommDDELink(BidWin, Factory & "." & Trim(NodeName.Text) & "VMD" & Trim(VMD
No.Text) & "WIN" & "." & "F_CV")

'Create the link between the "Bid Cost" textblock and the
```

```
'"V" & VMDNo.Text & "BidCost" Cell Controller Database block
'Call CommDDELink(BidCost, NodeName.text & "." & "V" & VMDNo.text & "BIDCOST" &
"." & "F_CV")

'Create link between "Bid Request" Textbox and
'"FBIDREQUEST" Factory Controller Database block

'Call CommDDELink(Bidrequest, Factory & "." & "FBIDRQUEST" & "." & "F_CV")

'Create Link between "Bid Object Control " textbox in the
'Queueing application and the "BidControl" block in the
'Cell Controller

'Call CommDDELink(BidObjectControl, NodeName.text & "." & "BIDCONTROL" & "." &
F_CV")

'Create Automatic Link between this VMD's "Queue Control"
'textbox and the "V1QCONTROL" database block in the Cell
'Controller
Call CommDDELink(Bidobjectcontrol, Nodename.Text & "." & "V" & VMDNo.Text & "QC
NTROL" & "." & "F_CV")

'Create an automatic link between the  "VMDPartno" textbox in the "VMD Status"
'frame and the database block called "V1CURRPART".This link automatically tells
'the operator what part is currently being manufactured.The idea is that in
'the current VMD  when the "V1RNCC1CCN" database block is one unit larger than
'the "V1RNFIXNOP" database block a program will see this and change the
'"V1QCONTROL" Database block to a "4" which loads that jobs next task into
'that VMD's outgoing queue
Call CommDDELink(VMDPartNo, Nodename.Text & "." & "V" & VMDNo.Text & "CURRPART"
& "." & "F_CV")

'create a link between the "StartTime" FixDmacs block and the Sim_Start textbox
so that when the
'factory controller sends a message to the starttime block the queueing applica
ion responsible
'for calculating the average queue length for its queue can calculate the start
time for the
'simulation and use that as a means for gauging the event driven queue alterati
n times
Call CommDDELink(Sim_Start, Nodename.Text & "." & "StartTime" & "." & "F_CV")




'reset the bidwin database block
Node = Nodename.Text
Tags = "Bidwin"
Field = "A_CV"
texts = "0"

Call SendInfo(Node, Tags, Field, texts)

'***LOAD THE FORM THAT DISPLAYS AND COLLECTS THE RESULTS OF THE EXPERIMENTS RUN
Load Res_Form
```

```
'****************** Load Event Complete ****************
Exit Sub

Checkerror:

If Err = 55 Then
    MsgBox ("the error is file is already open")
    Resume Next
    Else
    Resume Next
End If


End Sub


Sub Form_Unload (Cancel As Integer)
UnloadFlag = 1
Unload Queue_Status
Unload Res_Form


'send a prompt to the vmd insinuating that this VMD's queue is empty and theref
re stop the
'vmd's execution.
Nodes = Nodename.Text
Tags = "V" & VMDNo.Text & "INQUSIZE"
Field = "A_CV"
texts = "0"
Call SendInfo(Nodes, Tags, Field, texts)

Startform.Show
End Sub

Sub QCBidObjSent_Change ()
Dim BidToolSpec() As Toolspec
Dim RqFileLength As Toolspec
Dim Filelength As Single
Dim X As Integer
Dim position As Integer
Dim Filenum As Variant

Select Case QCBidObjSent

Case Is = 1#
Filenum = FreeFile
Filelength = Len(RqFileLength)
Open Defaultpath & "V1CC1ToolSpec.txt" For Random Access Write Shared As Filenu
 Len = Filelength

ReDim BidToolSpec(8)
```

```
BidToolSpec(1).NoOfToolsNeeded = 8
BidToolSpec(1).ToolTypesize = 1.2
BidToolSpec(2).ToolTypesize = 3.4
BidToolSpec(3).ToolTypesize = 4.5
BidToolSpec(4).ToolTypesize = 6.3
BidToolSpec(5).ToolTypesize = 5.3
BidToolSpec(6).ToolTypesize = 10.2
BidToolSpec(7).ToolTypesize = 8.2
BidToolSpec(8).ToolTypesize = 6.2

position = 1

For position = 1 To 8
Put Filenum, position, BidToolSpec(position)
Next

Close Filenum
End Select

End Sub

Sub QueueCost_Change ()

'Timer1.Enabled = False

'Select Case Val(Queuecost.Text)


'Case Is >= 1

Call AqlCalc


'End Select
'Timer1.Enabled = True

End Sub

Sub ResView_Click ()
Res_Form.Show
End Sub

Sub Sim_Start_Change ()

Select Case Val(Sim_Start.Text)

Case Is = 1

'reset results variables
Oldnumber = 0
OldArea = 0
OldQlength = 0

'The first thing that we are required to do is to clear all the result listbox
'of previous results so that we can record the results of the current Expt.
```

```
Res_Form.Res_Job.Clear
 es_Form.Res_time.Clear
Res_Form.Res_Event.Clear
Res_Form.Res_Cost.Clear
 es_Form.Aql_Time.Clear
Res_Form.Aql_Aql.Clear


'send the simulation start time to the spreadsheet
 es_Form.Res_Job.AddItem "Start Simulation"
Res_Form.Res_time.AddItem Time$
Res_Form.Res_Event.AddItem "Simulation Start"
 es_Form.Res_Cost.AddItem "0"


 urr_Time.Text = Time$


 *TConst MANUAL = 2, none = 0

'*TCurr_Time.LinkMode = none
'*TCurr_Time.LinkTopic = "EXCEL|c:\CELL.XLS"
 *TCurr_Time.LinkItem = "R" & CStr(Rowpointer) & "C30"
'*TCurr_Time.LinkMode = MANUAL

 *TCurr_Time.LinkPoke


CurrDate.Text = Date$

 *TCurrDate.LinkMode = none
'*TCurrDate.LinkTopic = "EXCEL|c:\CELL.XLS"
'*TCurrDate.LinkItem = "R" & CStr(Rowpointer) & "C32"
 *TCurrDate.LinkMode = MANUAL

'*TCurrDate.LinkPoke



 calculate the start time and assign it to a form level variable so that it can
 e preserved
Starttime = CalcStartTime()

'MsgBox CStr(Starttime)
'now reset the database block so that it is able to respond once again
Blocktyp = "StartTime"
Field = "A_CV"
texts = "0"
Call SendInfo(Node, Blocktyp, Field, texts)

Case Else
```

```
Exit Sub

End Select



End Sub

Sub ToolAvailability_Change ()
Dim Thistoolspec() As Toolspec
Dim Thistool As Toolspec
'Dim BidToolSpec() As ToolSpec
'Dim RqFileLength As ToolSpec
Dim Filelength As Single
Dim Tool1Available As Integer
Dim Tool2Available As Integer
Dim Tool3Available As Integer
Dim Tool4Available As Integer
Dim X As Integer
Dim Count As Integer
Dim Filenum As Integer
Dim I As Integer
Dim filename As String
Dim Thisbit As String * 6
Dim Thistext As String * 8
Dim recordlength As Integer
Dim ThisFile As String * 70
Dim TheDirectory As String
Dim ThisFileslength As Integer
Dim Thisstool As Single

'On Error Resume Next
'Select Case ToolAvailability

'Case Is = 3#

'Type ToolSpec
'Toolname As String * 40
'ToolTypeSize As Single
'NoOfToolsNeeded As Integer
'End Type
'Thistext = Trim(Nodename.Text)


'If (Trim(Thistext)) Like "C0#" Then
 '    Thisbit = Mid$(Trim(Thistext), 3, 3)
  '   MsgBox ("Its not fucked")
 '    Else
  '       Thisbit = Mid$(Trim(Thistext), 2, 3)
   '      MsgBox ("Its Fucked")
'End If

'MsgBox (Thisbit)
```

```
filename = "V" & Trim(VMDNo.Text) & "C" & Trim(Thisbit) & "Spc.txt"
TheDirectory = "C:\Factory\Config\Cells\"      '"C:\CellCont\Specs\Tools\"

filenum = FreeFile
'recordlength = Len(Thistool)
'
ThisFile = Trim(TheDirectory) & Trim(filename)

'MsgBox (ThisFile)

Open the file for read\write random access
'Open Trim(ThisFile) For Random Access Read Write Shared As filenum Len = recor
ength

'redimension the array to take the first record
ReDim Thistoolspec(1)

'get the length of the file
'Get filenum, 1, Thistoolspec(1)

this is the current length of the file however we are required to make enough
om to store all
the check box information that has been checked.
'ThisFileslength = Thistoolspec(1).NoOfToolsNeeded
'MsgBox (CStr(ThisFileslength))

'Type ToolSpec
'Toolname As String * 40
ToolTypeSize As Single
NoOfToolsNeeded As Integer
End Type

ReDim Preserve Thistoolspec(ThisFileslength)

For i = 1 To ThisFileslength
'   Get filenum, i, Thistoolspec(i)
 '  MsgBox (CStr(Thistoolspec(i).ToolTypesize))
Next

'For i = 1 To Thistoolspec(1).NoOfToolsNeeded
If CSng(Tool1required.Text) = Thistoolspec(i).ToolTypesize Then
Tool1Available = 1  'Set tool1available flag
'End If
'Next

'For i = 1 To Thistoolspec(1).NoOfToolsNeeded
'If CSng(Tool2required.Text) = Thistoolspec(i).ToolTypesize Then
Tool2Available = 1  'Set tool2available flag
End If
'Next

'For i = 1 To Thistoolspec(1).NoOfToolsNeeded
' If CSng(Tool3required.Text) = Thistoolspec(i).ToolTypesize Then
To ol3Available = 1  'Set tool3available flag
'End If
```

```
 'Next

'For i = 1 To Thistoolspec(1).NoOfToolsNeeded
'I f CSng(Tool4required.Text) = Thistoolspec(i).ToolTypesize Then
'Tool4Available = 1 'Set tool4available flag
'End If
'Next


'If Tool1Available = 1 And Tool2Available = 1 And Tool3Available = 1 And Tool4A
ailable = 1 Then
'ToolAvailability.Text = 1
'Else ToolAvailability.Text = 2
'
'End If

'Close filenum

'End Select
'node = "Fact"
'Blocktyp = "V1TOOLAVLE"
'Field = "A_CV"
'texts = ToolAvailability.Text
'Call SendInfo(Trim(node), Trim(Blocktyp), Trim(Field), Trim(texts))


End Sub

Sub View_Queues_Click ()
Queue_Status.Show
End Sub

Sub VMDPartno_Change ()
'This event procedure calculates the cost of the task
'currently being done in the vmd and as each part cycle
'is completed(as each part within a task is done) the
'total queue cost is re-evaluated as a part waiting to be
'processed within a vmd is considered to still be part of
'that vmd's queue.
Dim cost As Single
Dim Msg1 As String * 65
Dim Msg2 As String * 75
Dim MsgTot As String * 141
Dim VMDCost As Double
Dim PartNo_Error As Variant
Static OldCost As Double 'Require OldCost to be global but only accessible
Dim Newcost As Double   'by this procedure
Dim Columnfocus As Integer

Dim Thisdata As String * 20
Dim Filenum As Integer
Dim Thislength As Integer
Dim firsttime As Single
Dim Secondtime As Single
Dim LoopCount As Integer
```

```
On Error GoTo Faultfind

'TheTaskIsFinishedOnAllParts = 1 - is set to "1" when all parts have had the
'task completed on them - in reality this check will not be required as the
'FixDmacs program will not allow the next part to be loaded if there is not a
'next part to be loaded, however this check is here so that during program
'simulation procedures the user dosn't accidentally put something stupid in
'there i.e: Tries to load part number 6 of a set of 5 parts!!!!!!!

'******In Here we will have to request New estimate of the Estimated Cycle
'Time in response to that calculated in the vmd and use it to calculate true
'Queue Cost*******************

If Val(VMDPartNo.Text) = 0 Then
    Exit Sub
End If

If Val(VMDPartNo.Text) = 1# Then 'as no parts have been made yet we are just st
rting
    Exit Sub
End If

'calculate the cost of doing the entire task
Newcost = (Val(VMDTotalNo.Text) * (Val(Estcycletime.Text) + 2 * Val(LoadUn.Text
)) + Val(Tasksetuptime.Text)


If CInt(VMDPartNo.Text) > 1 And CInt(VMDPartNo.Text) <= CInt(VMDTotalNo.Text) T
en
        'At least one part has already been made but the last part is not
         'about to be made - i.e we are about to make the second part,third pa
rt,
         'fourth part or (Final Part - 1)
         cost = Val(QueueCost.Text) 'set the cost variable equal to the curren
t value of the
         'queuecost textbox
         QueueCost.Text = (cost - (Newcost / CInt(VMDTotalNo.Text)))
  '      MsgBox (Queuecost.Text)
        'Else are about to produce the final part
         Call updateqcost

         'now update the cycle data to excel so that we know when the first cy
cle was complete

           '*******RESULT COLLECTION BIT"
           Res_Form.Res_Event.AddItem "Cycle " & CStr(Val(VMDPartNo.Text) - 1)
& " Complete."
           Res_Form.Res_Job.AddItem Jobname.Text

           'we are required to find out what time it is so that we can shunt t
his information to the
           'results sheet and have a time orientated result
           Res_Form.Res_time.AddItem Time$
           Res_Form.Res_Cost.AddItem QueueCost.Text
```

```
            '*********************************

            Exit Sub
End If

If CInt(VMDPartNo.Text) > CInt(VMDTotalNo.Text) Then 'all cycles of this task a
e complete
    cost = Val(QueueCost.Text)
    QueueCost.Text = (cost - (Newcost / CInt(VMDTotalNo.Text)))

    '********RESULT COLLECTION BIT
    Res_Form.Res_Cost.AddItem QueueCost.Text


    'all cycles of this task is just finished so put the name of the job onto t
e outgoing queue
    'but only if the job has more tasks to be done.
    Call updateqcost 'update the file that contains the queuecost for this vmd


    Dim ToolToSend As Single
    Dim I As Integer
    'we are also required to return the tools that we used, we do this by sendi
g the name of the tool
    'to the factory Controller

    ToolToSend = CSng(InTool1required.Text)

    Field = "A_CV"
    Node = "Fact"
    Blocktyp = Trim(Nodename.Text) & "V" & VMDNo.Text & "tool"
    texts = CStr(ToolToSend)
    Call SendInfo(Node, Blocktyp, Field, texts)


    If Finaltask = 1 Then 'the job is completed as all required tasks have been
done
        'all tasks have been completed on this job therefore dont put it on the
outgoing queue

        '*******RESULT COLLECTION BIT"
        Res_Form.Res_Event.AddItem "Whole job Completed"
        Res_Form.Res_Job.AddItem Jobname.Text

        'we are required to find out what time it is so that we can shunt this
nformation to the
        'spreadsheet and have a time orientated result
        Res_Form.Res_time.AddItem Time$

        '******END OF COLLECTION BIT**********************************

        '****IN HERE WE ARE REQUIRED TO WRITE A PIECE OF CODE THAT RECORDS THE
IME THAT THE
        '*****JOB WAS FINISHED AND SEND THE TIME THE JOB WAS COMPLETED TO THE F
CTORY CONTROLLER
        '*****SO THAT THE FACTORY CONTROLLER IS ABLE TO UPDATE THE MTBC(Mean ti
```

e between completion)

'with this piece of code we simply tell the factory controller to update the MTBC based
'on it's own computer clock by sending it a flag

Node = "fact"
Blocktyp = "updateMTBC" 'updates the Mean Time between Completion
Field = "A_CV"
texts = "1"
Call SendInfo(Node, Blocktyp, Field, texts)

GoTo Nextpart
Else

If Forcevar = 1 Then 'this job has further tasks required to be done and it is an urgent
'job so put it at the top of the outgoing queue

'*****RESULT COLLECTION BIT"
Res_Form.Res_Job.AddItem Jobname.Text
Res_Form.Res_Event.AddItem "Top Queue add"
'we are required to find out what time it is so that we can shunt this information to the
'Results sheet and have a time orientated result
Res_Form.Res_time.AddItem Time$

'**********

Queue_Status.Outqueue.AddItem Jobname.Text, 0

Else
If Forcevar = 2 Then 'add the job at the bottom of the queue

'*******RESULT COLLECTION BIT"
Res_Form.Res_Job.AddItem Jobname.Text

'we are required to find out what time it is so that we can shunt this information to the
'Results sheet and have a time orientated result
Res_Form.Res_time.AddItem Time$

Res_Form.Res_Event.AddItem "Bottom Queue Add"

'**********
Queue_Status.Outqueue.AddItem Jobname.Text
End If

'here we are required to save the outgoing queue names to file so that if the system
'crashes or is shut down then the current state is preserved.
'Dim Filenum As Integer

```
e between completion)


        'with this piece of code we simply tell the factory controller to updat
 the MTBC based
        'on it's own computer clock by sending it a flag

        Node = "fact"
        Blocktyp = "updateMTBC" 'updates the Mean Time between Completion
        Field = "A_CV"
        texts = "1"
        Call SendInfo(Node, Blocktyp, Field, texts)


        GoTo Nextpart
    Else

        If Forcevar = 1 Then 'this job has further tasks required to be done an
 it is an urgent
        'job so put it at the top of the outgoing queue

            '*****RESULT COLLECTION BIT"
            Res_Form.Res_Job.AddItem Jobname.Text
            Res_Form.Res_Event.AddItem "Top Queue add"
            'we are required to find out what time it is so that we can shunt t
is information to the
            'Results sheet and have a time orientated result
            Res_Form.Res_time.AddItem Time$


            '************************************

            Queue_Status.Outqueue.AddItem Jobname.Text, 0


            Else
                If Forcevar = 2 Then 'add the job at the bottom of the queue

                    '*******RESULT COLLECTION BIT"
                    Res_Form.Res_Job.AddItem Jobname.Text

                    'we are required to find out what time it is so that we can
shunt this information to the
                    'Results sheet and have a time orientated result
                    Res_Form.Res_time.AddItem Time$

                    Res_Form.Res_Event.AddItem "Bottom Queue Add"

                    '************************************
                    Queue_Status.Outqueue.AddItem Jobname.Text
                End If

        'here we are required to save the outgoing queue names to file so that
f the system
        'crashes or is shut down then the current state is preserved.
            'Dim Filenum As Integer
```

```
            'Dim OutqueueJob As String * 20
            'Dim Filelength As Integer
        Dim Listlength As Integer
        Dim Outqueuejob As String * 20

        Filenum = FreeFile

        Open "C:\Cellcont\Queue\Outqueue\V" & Trim(VMDNo.Text) & "outque.txt" Fo
r Random Access Write Shared As Filenum Len = 20

        'MsgBox ("C:\Cellcont\Queue\Outqueue\V" & Trim(VMDNo.Text) & "outque.txt
")

        Listlength = Queue_Status.Outqueue.ListCount

        'MsgBox (CStr(Listlength))

        For I = 1 To Listlength
            Outqueuejob = Queue_Status.Outqueue.List(I - 1)
            'MsgBox (Outqueuejob)
            Put Filenum, I, Outqueuejob
        Next

        Close Filenum
        End If
    End If


Nextpart:
 now remove all the information from the "Task being Currently Executed" frames
'textboxes
Jobname.Text = ""
Jobinf.Text = ""
No_OfParts.Text = ""
Tasksetuptime.Text = ""
Lstcycletime.Text = ""
LoadUn.Text = ""
Tasknumber.Text = ""
Taskname.Text = ""
InTool1required.Text = ""
Finaltask.Text = ""
Machinetype.Text = ""
CNCfilename.Text = ""
Duedate.Text = ""
Fixtures.Text = ""
SetUpInstructionsFilename.Text = ""
Geometric.Text = ""
 Call SaveToOutqueue


RecordtoExcel:
 **********NOW IN HERE WE ARE REQUIRED TO PUT THE EXCEL PART THAT UPDATES THE ST
ATE OF THE VMD
    'the second thing that we are required to do is to find out what VMD we are
Working with and then
    'set the column accordingly
```

```
'*TSelect Case Val(VMDNo.Text)

'*TCase Is = 1

'*TColumnfocus = 1

'*TCall WriteToExcel(VMD1Rowpointer, Columnfocus)

'*TVMD1Rowpointer = VMD1Rowpointer + 1

'VMD1Rowpointer = 4
' VMD2Rowpointer = 4
' VMD3Rowpointer = 4
' VMD4Rowpointer = 4


'*TCase Is = 2
'*TColumnfocus = 8

'*TCall WriteToExcel(VMD2Rowpointer, Columnfocus)

'*TVMD2Rowpointer = VMD2Rowpointer + 1


'*TCase Is = 3

'*TColumnfocus = 15

'*TCall WriteToExcel(VMD3Rowpointer, Columnfocus)

'*TVMD3Rowpointer = VMD3Rowpointer + 1


'*TCase Is = 4

'*TColumnfocus = 22

'*TCall WriteToExcel(VMD4Rowpointer, Columnfocus)

'*TVMD4Rowpointer = VMD4Rowpointer + 1


'*TEnd Select

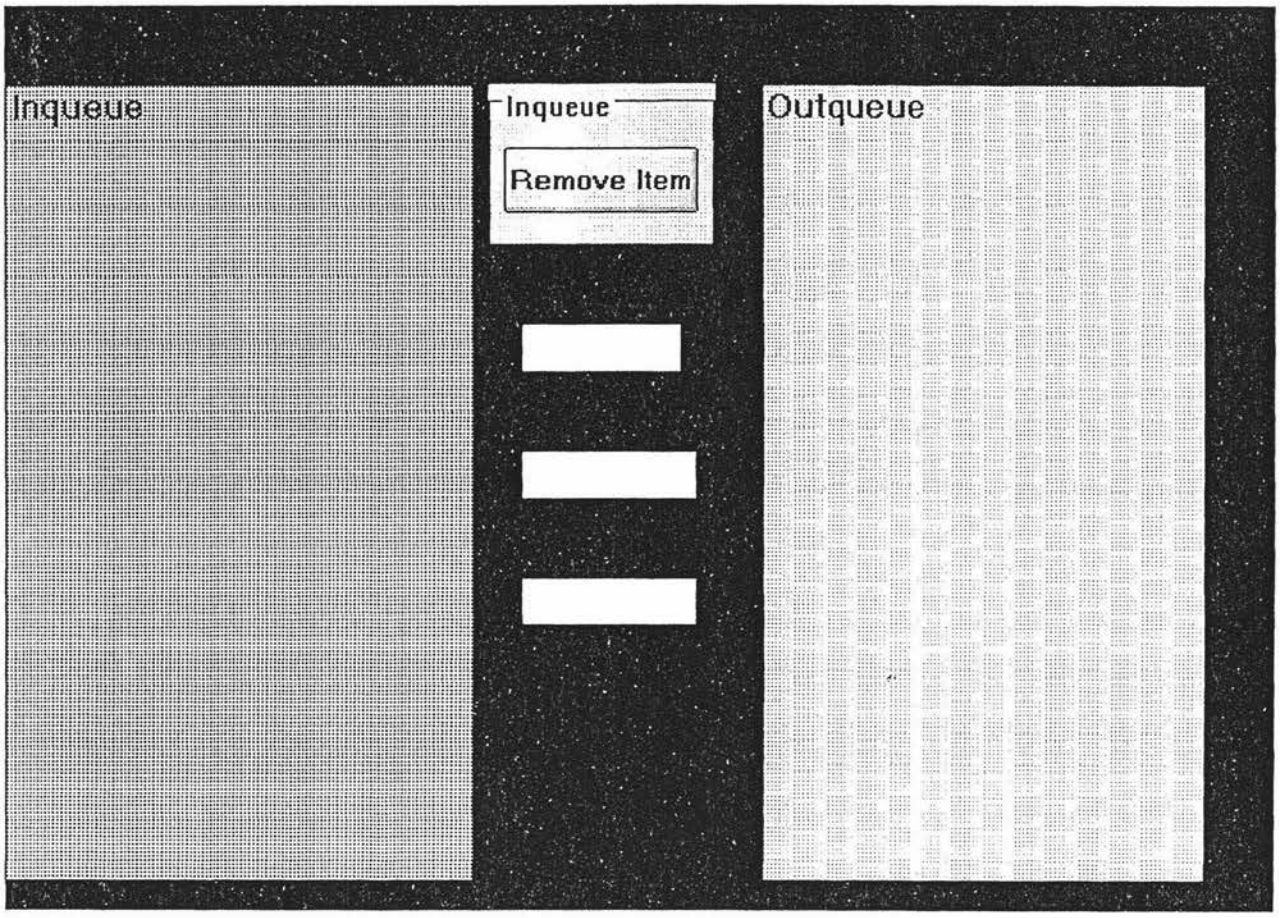

End If

Exit Sub

Faultfind:

If Err = 55 Then
    MsgBox ("the error is file is already open")
    Resume Next
    Else
        Resume Next
```

```
End If
End Sub
```


Inqueue
Inqueue
Remove Item
Outqueue

```
)UEUE_ST.FRM - 1

)ption Explicit
)ption Compare Text

'EDA Function Declarations

  functions specific to VB (not standard eda fcns)
)eclare Sub eda_version Lib "vdba.dll" (ByVal V$)
)eclare Sub eda_get1_ascii Lib "vdba.dll" (ByVal n$, ByVal t$, ByVal f$, ByVal
-)
Declare Function eda_set1_ascii% Lib "vdba.dll" (ByVal n$, ByVal t$, ByVal f$,
-/Val V$)

' Note define_group returns a long

)eclare Function eda_define_group& Lib "vdba.dll" (ByVal Count%, ByVal Detect%)
)eclare Sub eda_delete_group Lib "vdba.dll" (ByVal Handle&)

  should always pass 0 for vsp
Declare Function eda_define_ntf& Lib "vdba.dll" (ByVal Handle&, ByVal enode$, B
'al etag$, ByVal efield$, ByVal vsp&)
)eclare Sub eda_delete_ntf Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
'
)eclare Sub eda_lookup Lib "vdba.dll" (ByVal Handle&)
)eclare Sub eda_read Lib "vdba.dll" (ByVal Handle&)
Declare Sub eda_write Lib "vdba.dll" (ByVal Handle&)
)eclare Function eda_write1% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
)eclare Sub eda_wait Lib "vdba.dll" (ByVal Handle&)
Declare Function eda_query_wait% Lib "vdba.dll" (ByVal Handle&)
'
)eclare Function eda_get_error% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
Declare Function eda_get_ascii% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, ByVa
 value$, ByVal maxlen%)
)eclare Function eda_set_ascii% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, ByVa
 value$)

  note pointer to float is passed
)eclare Function eda_get_float% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, value
!)
)eclare Function eda_set_float% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, value
 )

 eda communication variables
 )im Ghandler As Long
Dim eda_err As Variant
)im tex As Integer
 )im Newnum As Variant
-Dim node As String * 15
)im Tags As String * 15
-)im Field As String * 15
Dim answer As String * 80
)im ans As String * 10
 )im defnode As Long
Dim trap As String * 10
)im texts As String * 6
 )im Nodetyp As String * 20
Dim Nodes As String * 20
```

```
Dim bidrotenable As Variant
Dim Blocktyp As String * 20
Dim reply As Variant
Dim Bidoperatestate As Variant
Dim Bidnodetext As String * 10
Dim bidtokenstate As Variant
Dim Tokenstate As Integer

'dde variables
Dim X As Variant
Dim Dummy As Variant



Sub Command1_Click ()
Dim Filenum As Integer

Filenum = FreeFile
Open netdrive & "Cellcont\Auction\Auctdata.txt" For Output As Filenum Len = 80
Print #Filenum, CStr(Newqueue.NodeName.Text)
Print #Filenum, "Part"
Close Filenum

End Sub

Sub Command2_Click ()
Inqueue.AddItem "Tanya"

End Sub

Sub Command3_Click ()
Inqueue.AddItem "Pauls"
End Sub

Sub CommDDELink (Text As TextBox, LinkItem)
'This Sub Procedure creates an Automatic link between the
'Queueing application and either the Factory Controller or
'the Cell Controller

Text.LinkMode = none  'If the link is not already
                      'created then create it
Text.LinkTopic = "DMDDE|DATA"  'Link topic is FixDmacs

Text.LinkItem = LinkItem       'This is the Node,
                               'databaseblock and field we
                               'want to link to

Text.LinkMode = Link_Automatic 'Automatic update of data


End Sub

Function eda_requestinfolink (n, t, f)
Ghandler = eda_define_group(1, 0)
defnode = eda_define_ntf(Ghandler, n, t, f, 0)
```

```
UEUE_ST.FRM - 3

Call eda_lookup(Ghandler)
all eda_wait(Ghandler)
Call eda_read(Ghandler)
Call eda_wait(Ghandler)
ewnum = eda_get_ascii(Ghandler, defnode, answer, 50)
Call eda_delete_group(Ghandler)
eda_requestinfolink = answer
nd Function

Sub RunDDEServer ()
ppActivate "Microsoft Visual Basic" 'Subject is VB
  = Shell("c:\wdmacs\dmdde.exe", 1) 'Makes sure The DDE  server is going
Dummy = DoEvents()

nd Sub

ub SendInfo (n, t, f, Text)
handler = eda_define_group(1, 0)
defnode = eda_define_ntf(Ghandler, n, t, f, 0)

all eda_lookup(Ghandler)
Call eda_wait(Ghandler)
da_err = eda_get_error(Ghandler, defnode)
ext5.Text = Str$(eda_err)
Newnum = Text
reply = eda_set_ascii(Ghandler, defnode, Newnum)
all eda_write(Ghandler)
all eda_wait(Ghandler)
all eda_delete_group(Ghandler)

nd Sub

ub bidoperate_Box_Change ()

Dim VMDtoauction As Integer
im JobToAuction As String
im Filenum As Integer
Dim Recordlength As Integer
im Thisstring As String
im Listlength As Integer
Dim i As Integer
im Outqueuejob As String * 20
im ThisVMDSType As Variant
Dim ThisNodesName As String * 20
im v1state As Integer
im v2state As Integer
Dim v3state As Integer
im v4state As Integer

On Error GoTo Findthefault

'request the state of all the VMD's

ode = Trim(Newqueue.NodeName.Text)
ield = "A_CV"
```

```
'get the state of the first machine
Tags = "v1machtyp"
v1state = Val(eda_requestinfolink(node, Tags, Field))

'get the state of the first machine

Tags = "v2machtyp"
v2state = Val(eda_requestinfolink(node, Tags, Field))

'get the state of the first machine
Tags = "v3machtyp"
v3state = Val(eda_requestinfolink(node, Tags, Field))

'get the state of the first machine
Tags = "v4machtyp"
v4state = Val(eda_requestinfolink(node, Tags, Field))


VMDtoauction = CInt(Newqueue.VMDNo.Text)

'MsgBox (Bidoperate_Box.Text)
If Val(Bidoperate_Box.Text) = 0 Then
    Exit Sub
End If

'first check to make sure whether all VMD's have been given the chance to condu
t an auction.
If Val(Bidoperate_Box.Text) = 5 Then
'all possible VMD's have been given the chance to hold an auction therefore we
re
'required to release the bidtoken by sending a zero to the "bidoperate" Block
    texts = "0"
    node = CStr(Newqueue.NodeName.Text)
    Tags = "Bidoperate"
    Field = "A_CV"
    Call SendInfo(node, Tags, Field, texts)
    Exit Sub
End If



'this particular box gives the system the opportunity to auction a job off each
VMD


'THIS PART IS ONLY A TEMPORARY MEASURE IT WILL BE REPLACED BY AN EDA REQUEST FOR
 VMD ACTIVITY
'STATUS AND A CHECK
If Val(Bidoperate_Box.Text) = 1 And v1state = 0 Or Val(Bidoperate_Box.Text) = 2
And v2state = 0 Or Val(Bidoperate_Box.Text) = 3 And v3state = 0 Or Val(Bidopera
e_Box.Text) = 4 And v4state = 0 Then
   'skip this one as it is not active
   texts = CStr(Val(Bidoperate_Box.Text) + 1)
    'MsgBox (CStr(Val(Bidoperate_Box.Text)))
    'MsgBox (texts)
```

```
    node = CStr(Newqueue.NodeName.Text)
   ' MsgBox (node)
    Tags = "VMDCurAuct"
    Field = "A_CV"
    Call SendInfo(node, Tags, Field, texts)
Exit Sub
nd If


THIS PART IS ONLY A TEMPORARY MEASURE IT WILL BE REPLACED BY AN EDA REQUEST FO
VMD ACTIVITY
'STATUS AND A CHECK
'If Val(Bidoperate_Box.Text) = 2 Or Val(Bidoperate_Box.Text) = 3 Then
  'skip this one as it is not active
 '' texts = CStr(Val(Bidoperate_Box.Text) + 1)
   'MsgBox (CStr(Val(Bidoperate_Box.Text)))
   'MsgBox (texts)
'''   node = CStr(Newqueue.Nodename.Text)
  ' MsgBox (node)
'''   Tags = "VMDCurAuct"
'''   Field = "A_CV"
'''    Call SendInfo(node, Tags, Field, texts)
'''Exit Sub
'''End If


If Val(Bidoperate_Box.Text) = VMDtoauction Then
the number in the bidoperate textbox matches this modules VMD Number

   'and request the identification number from the VMD to see if the process i
currently active
   'or not
   node = CStr(Newqueue.NodeName.Text)
   Tags = "V" & CStr(VMDtoauction) & "MACHTYP"
   Field = "A_CV"

   ThisVMDSType = eda_requestinfolink(node, Tags, Field)

   If CInt(ThisVMDSType) > 0 Then
       'this vmd is currently active therefore we can auction an item off the t
o of the list
       'if it is there.

       'check to see if there is anything in the list
       If Len(Trim(Outqueue.List(0))) <> 0 Then
           'the list is not empty therefore we are required to auction an item
ff the top of the
           'list.
           JobToAuction = Outqueue.List(0)
     '       MsgBox ("The job to auction is " & JobToAuction)

           'remove the job from the list
           Outqueue.RemoveItem 0
```

```
            'kill the file that stores the status of the outgoing queue
            Kill "C:\Cellcont\Queue\Outqueue\V" & Trim(Newqueue.VMDNo.Text) &
utque.txt"


            'denote a freefile
            Filenum = FreeFile

            'open up a new copy of the file that saves the outqueue state to fi
e
            Open "C:\Cellcont\Queue\Outqueue\V" & Trim(Newqueue.VMDNo.Text) &
utque.txt" For Random Access Write Shared As Filenum Len = 20

            'get the length of the outqueue list required to be saved
            Listlength = Queue_Status.Outqueue.Listcount


            For i = 1 To Listlength
                'get each element in the list and put it in the file
                Outqueuejob = Queue_Status.Outqueue.List(i - 1)
                'put the data in the file
                Put Filenum, i, Outqueuejob
            Next


            'close the file
            Close Filenum


            'now write this name to a file and put it on the network so that tl
 Factory Controller
            'knows what job it has to auction for the Factory Controller.

            'Define a file reference number
            Filenum = FreeFile

            'get the name of the node holding the auction
            Thisstring = CStr(Newqueue.NodeName.Text)

            'open the auctiondata file that will store the auction data
            Open netdrive & "Cellcont\Auction\Auctdata.txt" For Output Shared A
 Filenum Len = 50

            Print #Filenum, JobToAuction 'print to file the name of the job to
uction to file
            Print #Filenum, Thisstring   'print the name of the node holding tl
 auction to file
            Print #Filenum, CStr(VMDtoauction)'print the number of the vmd hol
ng the auction to file

            'close the file
            Close Filenum

            'now we are required to prompt the factory Controller to auction tl
s particular item
            'using EDA.
```

```
            texts = "1"
            node = "Fact"
            'Exit Sub
            Tags = "CellisAuct"
            Field = "A_CV"
            Call SendInfo(node, Tags, Field, texts)

        Else
                'this VMD's queue is empty therefore
                'we are required to let the next VMD have a go at conducting an
uction therefore send
                'this information to the bidoperate textbox.
                texts = CStr(VMDtoauction + 1)
            '    MsgBox (texts)
                node = CStr(Newqueue.NodeName.Text)
                Tags = "VMDCurAuct"
                Field = "A_CV"
                Call SendInfo(node, Tags, Field, texts)
        End If
    'else this VMD is not active so increment Auction pointer
    Else
        texts = CStr(VMDtoauction + 1)
        node = CStr(Newqueue.NodeName.Text)
        Tags = "VMDCurAuct"
        Field = "A_CV"
        Call SendInfo(node, Tags, Field, texts)
    End If
'Else 'increment the VMDto auction pointer

'   texts = CStr(VMDtoauction + 1)
  '  node = CStr(Newqueue.NodeName.Text)
   ' Tags = "VMDCurAuct"
    'Field = "A_CV"
    'Call SendInfo(node, Tags, Field, texts)

nd If

Exit Sub

indthefault:

f Err = 55 Then
    MsgBox ("the error is file is already open")
    Resume Next
    Else
    If Err = 13 Then
        Resume Next
    Else
        MsgBox ("The error is " & Error)
        Resume Next
    End If
nd If


nd Sub
```

```
Sub Form_Load ()

Dim NodeName As String
Dim Filenum As Integer
Dim Listlength As Integer
Dim Outqueuejob As String * 20
Dim Position As Integer

'no error recovery actions required
On Error Resume Next


NodeName = CStr(Newqueue.NodeName)

Call RunDDEServer 'run the fixdmacs dde server module

'create an automatic link between the Bidoperate textbox and the VMDCurrauct db
se block
Call CommDDELink(Bidoperate_Box, Newqueue.NodeName.Text & "." & "VMDCurAuct.F_C
")

'Create an automatic link between the "AuctionInitiate" textbox and the 'Auctio
Std"
'(Auction Started) database block in the Factory controller
Call CommDDELink(RequeJob, Newqueue.NodeName.Text & "." & "REQUEUEJOB" & "." &
F_CV")

'Now create a link between this VMD and "Machtyp" database block so that if th
'block changes to a zero we are able requeue the tasks on the queue
Call CommDDELink(Process_Has_Failed, Newqueue.NodeName.Text & "." & "V" & Trim
ewqueue.VMDNo.Text) & "MACHTYP" & "." & "F_CV")


'now write the contents of the outgoing queue file to the outgoing queue
Position = 0

Filenum = FreeFile

Open "C:\Cellcont\Queue\Outqueue\V" & Trim(Newqueue.VMDNo.Text) & "outque.txt"
or Random Access Read As Filenum Len = 20

'MsgBox ("C:\Cellcont\Queue\Outqueue\V" & Trim(Newqueue.VMDNo.Text) & "outque.t
t")


Do While Not EOF(Filenum)
    If Position = 0 GoTo Thisbit:
    Queue_Status.Outqueue.List(Position - 1) = Outqueuejob
Thisbit:
    Position = Position + 1
    Get Filenum, Position, Outqueuejob
Loop

Close Filenum
```

```
End Sub

Sub Form_Unload (Cancel As Integer)

If Unloadflag = 1 Then
    Exit Sub
    Else
        Cancel = -1
        Queue_Status.Hide
End If

End Sub

Sub Process_Has_Failed_Change ()
'This particular procedure is written so that when a process
'is isolated the ingoing queue items are requed on the
'outgoing queue

Dim InqueueTask() As String * 20
Dim i, s As Integer


Select Case Val(Process_Has_Failed.Text)

Case Is = 0


    'redimension array to hold all records
    ReDim InqueueTask(Inqueue.Listcount)

    s = Inqueue.Listcount


    For i = 1 To Inqueue.Listcount
        InqueueTask(i) = Inqueue.List(i - 1)
    Next



    Inqueue.Clear


    For i = 1 To s
        Outqueue.AddItem InqueueTask(i)
    Next


    'Ok fine, but we still need to save the contents of the outgoing queue
    'to file

    'Kill the old version of the outqueue file
    Kill "C:\Cellcont\Queue\Outqueue\V" & Trim(Newqueue.VMDNo.Text) & "outque.t
```

```
t"

    'now save the altered queue to file
    Dim Listlength As Integer
    Dim Outqueuejob As String * 20
    Dim Filenum As Integer


    Filenum = FreeFile

    Open "C:\Cellcont\Queue\Outqueue\V" & Trim(Newqueue.VMDNo.Text) & "outque.t
t" For Random Access Write As Filenum Len = 20

    Listlength = Queue_Status.Outqueue.Listcount

    'MsgBox (CStr(Listlength))

    For i = 1 To Listlength
        Outqueuejob = Queue_Status.Outqueue.List(i - 1)
        Put Filenum, i, Outqueuejob
    Next

    Close Filenum

    'Kill the old ingoing queue file
    Kill "C:\Cellcont\Queue\Inqueue\V" & Trim(Newqueue.VMDNo.Text) & "Inqueu.tx
"

Case Else
Exit Sub

End Select


End Sub

Sub Remove_item_Click ()

On Error Resume Next

Dim Numberoftasks As Integer
Dim Inqueuedata As String * 20
Dim i As Integer
Dim InComingQueueNewTaskCost As Single
Dim reply As Integer
Dim msg As String
Dim Typ As Integer
Dim Title As String * 30
Dim Filenum As Integer


'the first thing that is required to be done is to make sure that an item has be
en selected
If Len(Inqueue.List(Inqueue.ListIndex)) = 0 Then
    MsgBox "You haven't selected anything from the list, please do so.", MB_ICO
STOP, "No selection made"
```

```
    Exit Sub
    Else
        reply = MsgBox("Are you sure you want to delete the selected task?", MB
ICONQUESTION + MB_YESNo, "Delete Selection?")
        If reply = IDNO Then
            Exit Sub
            Else 'remove the item
                Inqueue.RemoveItem (Inqueue.ListIndex)
                'now kill the oldfile and open a new copy of the file and save
he new contents
                'to it.
                Kill "C:\Cellcont\Queue\Inqueue\V" & Trim(Newqueue.VMDNo.Text)
 "Inqueue.txt"

                'Open up a new copy of the file and save information to it.
                Numberoftasks = Inqueue.Listcount

                Filenum = FreeFile
                Open "C:\Cellcont\Queue\Inqueue\V" & Trim(Newqueue.VMDNo.Text)
 "Inqueue.txt" For Random Access Read Write As Filenum Len = 20

                For i = 1 To Numberoftasks
                    Inqueuedata = Queue_Status.Inqueue.List(i - 1)
                    Put Filenum, i, Inqueuedata
                Next

                Close Filenum
        End If
nd If



End Sub

ub RequeJob_Change ()
'When this textbox changes to the number that is the same as the VMD number that
 this queueing
 application has been given it means that it is required to get the name of the
job currently being
'auctioned and place it at the bottom of this queueing application's outgoing qu
 ue as although
 an auction was held for it the winning process had it's ingoing queue full. The
 job's name is
 only put at the bottom of the queue if it is not urgent otherwise it is put at
he bottom
'of the "Jobs Busy" queue in the factory Controller.

declare variables
Dim Filenum As Integer
 im CellToGiveBackTo As String
 im Jobname As String

n Error Resume Next 'if there is an error keep going
```

```
If Val(Newqueue.VMDNo.Text) = Val(RequeJob.Text) Then
    'we are required to open up the "R:\Factory\Bidding\Bidfile.bdf" file and g
t the name of the
    'jobfile to be requeued and then put it on the bottom of the queue and ther
resave the queue
    'data to file
    Filenum = FreeFile 'denote a freefile

    'open a network file that this information can be written to and put the
    'information in it and then prompt the relevant cell controller to
    'requeue the job
    Open netdrive & "Cellcont\Auction\Rejectjb.job" For Input As Filenum Len =
0

    Line Input #Filenum, CellToGiveBackTo 'read from the file the name of the c
ll
    'that the job must be returned to

    Line Input #Filenum, Jobname 'read from the file the name of the job that m
st be retured to
    'this VMDs outgoing queue

    Close Filenum 'now close the file

    'now that we have the name of the required job to be requeued we can put it
at the bottom of the queue

    Outqueue.AddItem Trim(Jobname)

    'Kill the old version of the outqueue file
    Kill "C:\Cellcont\Queue\Outqueue\V" & Trim(Newqueue.VMDNo.Text) & "outque.t
t"

    'now save the altered queue to file
    Dim Listlength As Integer
    Dim i As Integer
    Dim Outqueuejob As String * 20

    Filenum = FreeFile
    Open "C:\Cellcont\Queue\Outqueue\V" & Trim(Newqueue.VMDNo.Text) & "outque.t
t" For Random Access Write As Filenum Len = 20

    Listlength = Queue_Status.Outqueue.Listcount

    'MsgBox (CStr(Listlength))

    For i = 1 To Listlength
        Outqueuejob = Queue_Status.Outqueue.List(i - 1)
        Put Filenum, i, Outqueuejob
    Next

    Close Filenum


    'its one thing to respond to a change that occurs as a result of this textb
x being initiated
```

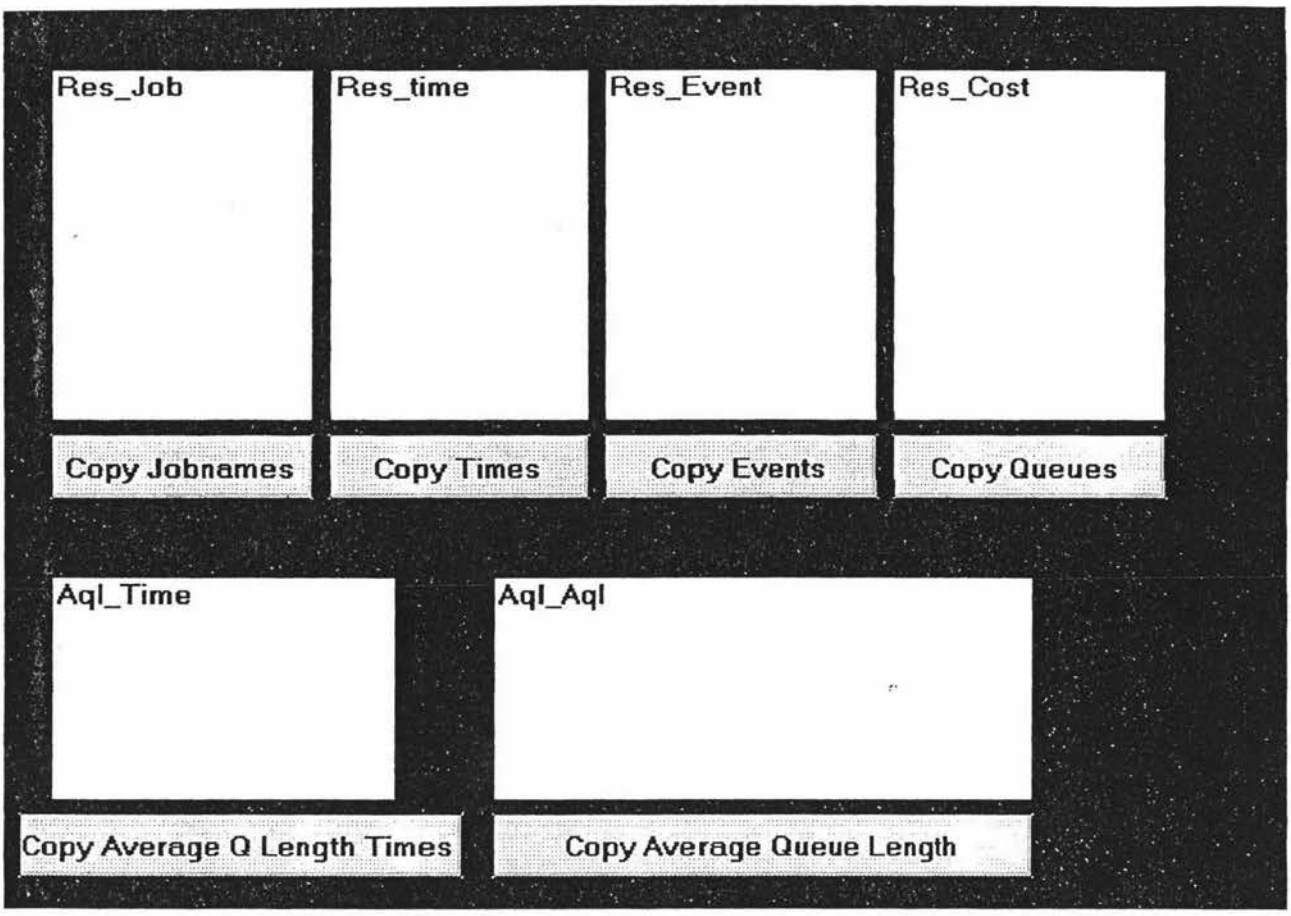
```
    'by a database block but we must also reset the database block so that it i
 able to respond
    'to the next change event.

    texts = "0"
    node = Trim(Newqueue.NodeName.Text)
    Field = "A_CV"
    Blocktyp = "Requeuejob"

    Call SendInfo(node, Blocktyp, Field, texts)

End If


End Sub
```


Res_Job
Res_time
Res_Event
Res_Cost
Copy Jobnames
Copy Times
Copy Events
Copy Queues
Aql_Time
Aql_Aql
Copy Average Q Length Times
Copy Average Queue Length

```
RES_FORM.FRM - 1

Option Explicit
Dim Oldtextrequired As String
Dim IsFirstpass As Integer

Sub Av_Q_Click ()
Dim Textrequired As String
Dim i As Integer
Dim Msg As String
Dim DefVal As String
Dim title As String

Oldtextrequired = ""
IsFirstpass = 0
ClipBoard.Clear


or i = 1 To Aql_Time.ListCount

    If IsFirstpass = 0 Then
        Textrequired = Aql_Time.List(0)
        Oldtextrequired = Textrequired
        IsFirstpass = 1
        Else
            Textrequired = Trim(Oldtextrequired) & Chr(10) & Aql_Time.List(i - 1
            Oldtextrequired = Textrequired
    End If

Next

ClipBoard.SetText Textrequired


End Sub

Sub Cop_AvQ_Click ()
Dim Textrequired As String
m i As Integer
Dim Msg As String
Dim DefVal As String
m title As String

Oldtextrequired = ""
Firstpass = 0
ipBoard.Clear


r i = 1 To Aql_Aql.ListCount

    If IsFirstpass = 0 Then
        Textrequired = Aql_Aql.List(0)
        Oldtextrequired = Textrequired
        IsFirstpass = 1
        Else
            Textrequired = Trim(Oldtextrequired) & Chr(10) & Aql_Aql.List(i - 1)
```

```
            Oldtextrequired = Textrequired
    End If

Next

ClipBoard.SetText Textrequired


End Sub

Sub Copy_Events_Click ()
Dim Textrequired As String
Dim i As Integer
Dim Msg As String
Dim DefVal As String
Dim title As String

Oldtextrequired = ""
IsFirstpass = 0
ClipBoard.Clear


For i = 1 To Res_Event.ListCount

    If IsFirstpass = 0 Then
        Textrequired = Res_Event.List(0)
        Oldtextrequired = Textrequired
        IsFirstpass = 1
        Else
            Textrequired = Trim(Oldtextrequired) & Chr(10) & Res_Event.List(i -
1)
            Oldtextrequired = Textrequired
    End If

Next

ClipBoard.SetText Textrequired


End Sub

Sub Copy_Jobnames_Click ()

Dim Textrequired As String
Dim i As Integer
Dim Msg As String
Dim DefVal As String
Dim title As String

Oldtextrequired = ""
IsFirstpass = 0
ClipBoard.Clear


For i = 1 To Res_Job.ListCount
```

```
RES_FORM.FRM - 3

    If IsFirstpass = 0 Then
        Textrequired = Res_Job.List(0)
        Oldtextrequired = Textrequired
        IsFirstpass = 1
        Else
            Textrequired = Trim(Oldtextrequired) & Chr(10) & Res_Job.List(i - 1
            Oldtextrequired = Textrequired
    End If

Next

ClipBoard.SetText Textrequired

nd Sub

Sub Copy_Queues_Click ()
)im Textrequired As String
Dim i As Integer
Dim Msg As String
im DefVal As String
im title As String

ldtextrequired = ""
sFirstpass = 0
ClipBoard.Clear


For i = 1 To Res_Cost.ListCount

    If IsFirstpass = 0 Then
        Textrequired = Res_Cost.List(0)
        Oldtextrequired = Textrequired
        IsFirstpass = 1
        Else
            Textrequired = Trim(Oldtextrequired) & Chr(10) & Res_Cost.List(i - 1

            Oldtextrequired = Textrequired
    End If

lext

lipBoard.SetText Textrequired


id Sub

Sub Copy_times_Click ()
im Textrequired As String
im i As Integer
Dim Msg As String
im DefVal As String
m title As String

ldtextrequired = ""
sFirstpass = 0
ClipBoard.Clear
```

```
For i = 1 To Res_time.ListCount

    If IsFirstpass = 0 Then
        Textrequired = Res_time.List(0)
        Oldtextrequired = Textrequired
        IsFirstpass = 1
        Else
            Textrequired = Trim(Oldtextrequired) & Chr(10) & Res_time.List(i -
)
            Oldtextrequired = Textrequired
    End If

Next

ClipBoard.SetText Textrequired

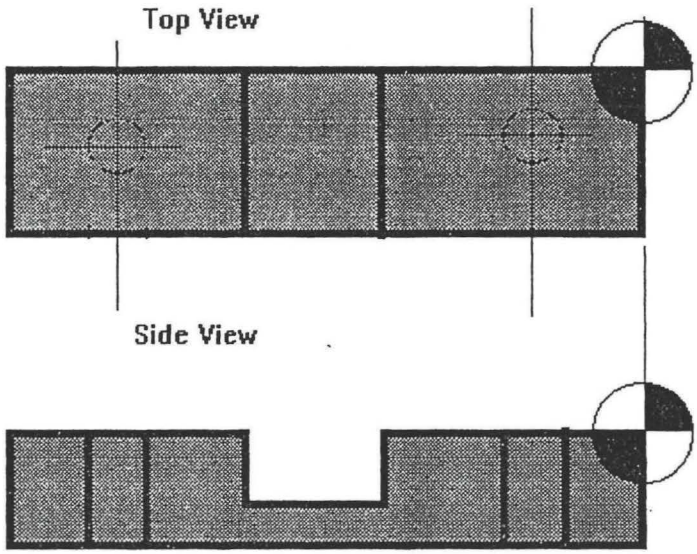

End Sub

Sub Form_Unload (Cancel As Integer)

If Unloadflag = 1 Then
    Exit Sub
    Else
        Cancel = -1
        Res_Form.Hide
End If

End Sub
```

The diagram as below illustrates the datum point of the part that is required to be manufactured.



```
Sub Exit_Click ()
On Error Resume Next
Unload SetupForm
End Sub
```

```
NEWLYQUE.BAS - 1

Option Explicit
Option Base 1
Option Compare Text


' MsgBox parameters
Global Const MB_OK = 0                    ' OK button only
Global Const MB_OKCANCEL = 1              ' OK and Cancel buttons
Global Const MB_ABORTRETRYIGNORE = 2      ' Abort, Retry, and Ignore buttons
Global Const MB_YESNOCANCEL = 3           ' Yes, No, and Cancel buttons
Global Const MB_YESNO = 4                 ' Yes and No buttons
Global Const MB_RETRYCANCEL = 5           ' Retry and Cancel buttons

Global Const MB_ICONSTOP = 16             ' Critical message
Global Const MB_ICONQUESTION = 32         ' Warning query
Global Const MB_ICONEXCLAMATION = 48      ' Warning message
Global Const MB_ICONINFORMATION = 64      ' Information message

Global Const MB_APPLMODAL = 0             ' Application Modal Message Box
Global Const MB_DEFBUTTON1 = 0            ' First button is default
Global Const MB_DEFBUTTON2 = 256          ' Second button is default
Global Const MB_DEFBUTTON3 = 512          ' Third button is default
Global Const MB_SYSTEMMODAL = 4096         'System Modal

' MsgBox return values
Global Const IDOK = 1                     ' OK button pressed
Global Const IDCANCEL = 2                 ' Cancel button pressed
Global Const IDABORT = 3                  ' Abort button pressed
Global Const IDRETRY = 4                  ' Retry button pressed
Global Const IDIGNORE = 5                 ' Ignore button pressed
Global Const IDYES = 6                    ' Yes button pressed
Global Const IDNO = 7                     ' No button pressed

'***** QUEUEING USER DEFINED VARIABLE TYPE DECLARATIONS ***

 Incoming Queue user defined variable declarations
Type Inqueue
Jobname As String * 20
nFoFilename As String * 20
Bid_No_Parts As String * 10
BidSetUpTime As String * 10
idLoadUnLoadTime As String * 10
BidCycleTime As String * 10
End Type

 Define node\Cell controller config user defined variable
Type Nodevmdconfig
Recordsize As Integer
Nodename As String * 4
VMDNo As Integer
Recordposi As Integer
End Type

 Noderecord required in Cell Configuration
Type Nodesrecord
Recordsize As Integer
```

```
Nodename As String * 6
End Type

Type Noodesrecord
Nodename As String * 8
Index As Integer
End Type

'Outgoing Queue user defined variable declarations
Type OutQueue
Jobname As String * 20
InFoFilename As String * 20
Bid_No_Of_Parts As Integer
BidSetUpTime As Integer
BidMachTyp As Single
BidCycleTime As Integer
BidTool1Required As Single
BidTool2Required As Single
BidTool3Required As Single
BidTool4Required As Single
BidLoadUnLoadTime As Integer
Force As Integer
End Type


'Declare Ingoing Queue pointers
Type InQueuePointers
    InStartOfQueue As Integer
    InFinishOfQueue As Integer
End Type

'Declare Outgoing Queue Pointers
Type OutQueuePointers
    OutStartOfQueue As Integer
    OutFinishOfQueue As Integer
End Type

'declare variable type who's array describes a VMD's tools that are available
Type ToolSpec
'Toolname As String * 40
    ToolTypeSize As Single
    NoOfToolsNeeded As Integer
End Type

'Define a user defined variable that is used to hold the value of each VMD's qu
uecost in each
'Cell Controller

Type VMDQueueCost
    Cellname As String * 8
    VMD1_Queue_Cost As Single
    VMD2_Queue_Cost As Single
    VMD3_Queue_Cost As Single
    VMD4_Queue_Cost As Single
End Type
```

```
NEWLYQUE.BAS - 3

'define a user defined variable type that is used to store the bidding data as
' is recieved
Type Bid_data_To_Be_sent
    Jobname As String * 10
    Machinetype As Single
    EstCycletime As Integer
    Setuptime As Integer
    LoadUnloadtime As Integer
    NoofCycles As Integer
    Tool1required As Single
'   Tool2required As Single
'   Tool3required As Single
'   Tool4required As Single
    Activate As Integer
    Force As Integer
End Type

'define a user defined variable type that is used to store the value of the too
'availability
'status in memory

Type Tool_AvleRec
    NumberOfRecords As Integer
    ToolType As String * 30
    ProcessType As String * 30
    Availability As String * 15
    Location As String * 10
    LifeUsed As Integer
    Toolsize As Integer
    Reserved As String * 10
    Toolcode As Single
End Type


'Queue handle Variables
Global position As Integer
'*******************************************************

Global Const Defaultpath = "C:\"
Global Const Netdrive = "r:\"

'****************** DDE Constants **********************
' LinkMode (forms and controls)
Global Const NONE = 0          ' 0 - None
Global Const LINK_SOURCE = 1     ' 1 - Source (forms only)
Global Const LINK_AUTOMATIC = 1 ' 1 - Automatic (controls only)
Global Const LINK_MANUAL = 2     ' 2 - Manual (controls only)
Global Const LINK_NOTIFY = 3     ' 3 - Notify (controls only)

'***************** Global Factory Node Constant***********
Global Const Factory = "Fact"

Type JobRecord
    Jobname As String * 13
    JaFoFilename As String * 20
    FinalTask As Integer
```

```
Tasknumber As Integer
Machinetype As Single 'Type as 3.1 the ".1" represents capacity
Taskname As String * 22
EstCycletime As Integer
Setuptime As Integer
Estloadunloadtime As Integer
NoofCycles As Integer
CNCfilename As String * 13
Geometricfilename As String * 13
Tool1required As Single
'Tool2required As Single
'Tool3required As Single
'Tool4required As Single
Fixturesfilename As String * 13
Status As Integer
StartTime As String * 5
Duedate As String * 15
RequiredCompDate As String * 15
RequiredComptime As String * 5
Force As Integer
SetUpInstructionsFilename As String * 20 'Shows datum graphically
End Type



Global Job As JobRecord

'***** Current Queue Cost Calculation Variable *******
'Global Cost As Single
Global Lo_GCost As Double

'******NEW THINGS REQUIRED TO BE DONE IN RESPONSE TO DEVELOPING THE VMD****

'1)Request new value of estimated Cycle Time from VMD and use it to calculate
'new queue time based on that new estimate


Global Firstup As Integer
Global Unloadflag As Integer
Global ForceVar As Integer
Global FinalTask As Integer


'define a variable that keeps track of the number of jobs in the incoming queue
Global NumberofJobs As Integer
Global Inqueuescost As Single

Sub openForInput (Thisfilesnum As Integer, filename As String, recordlength As
nteger)

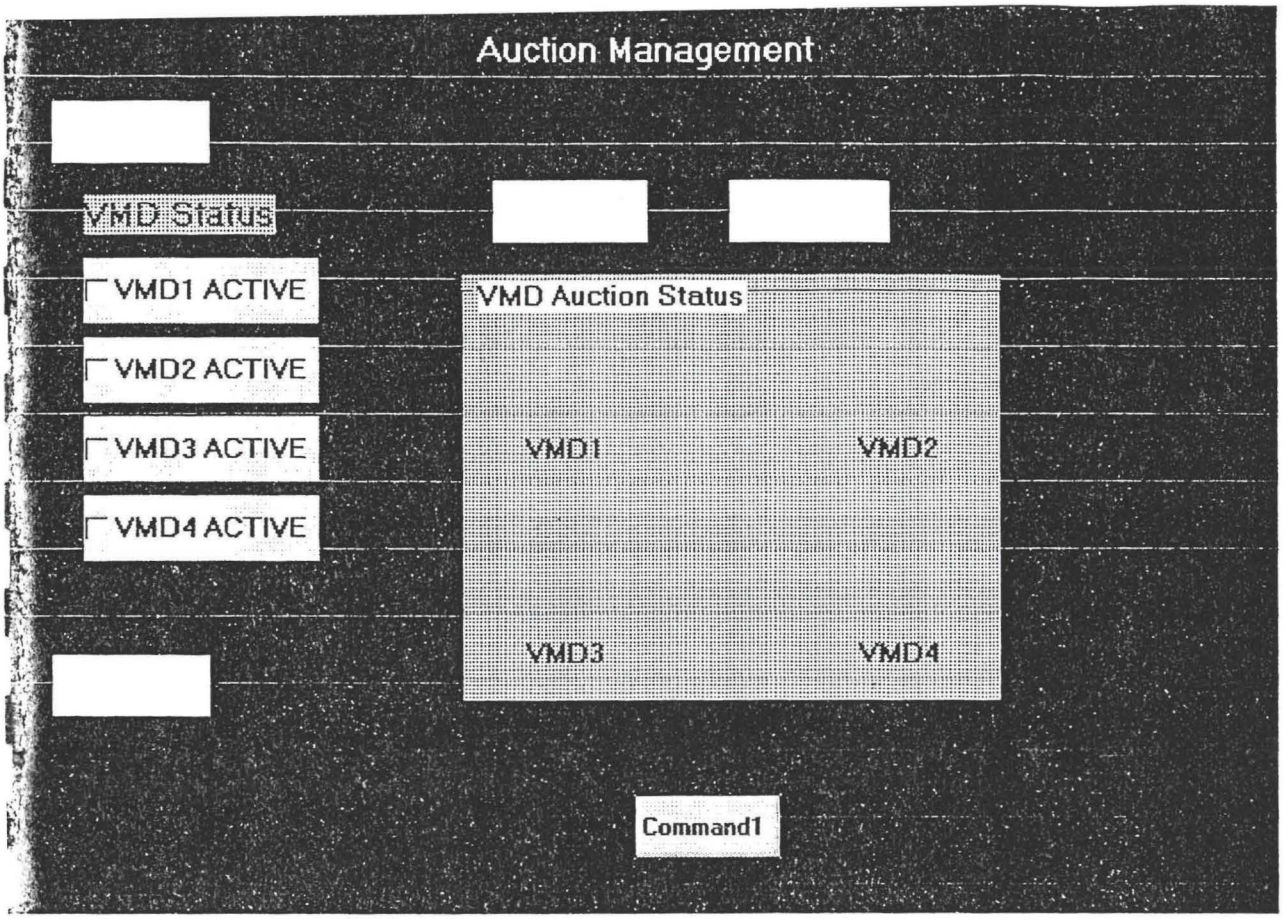
Open Trim(filename) For Input As Thisfilesnum Len = recordlength


End Sub
```

A.2 Implementation of the Auction Manager Application

The Auction Manager Application is resident within the Cell Controller and is responsible for instigating the securing of the bid token and the polling of the various devices controlled by the cell. As such it gives each device in the cell the opportunity to hold a task auction. Once each device within the cell has completed its attempt to auction it informs the Auction Manager application. The application then either gives the next device an opportunity to hold an auction or releases the bid token to the next cell.

The following material documents the Microsoft Visual Basic programming written for the implementation of the application.


Auction Management
VMD Status
VMD1 ACTIVE
VMD2 ACTIVE
VMD3 ACTIVE
VMD4 ACTIVE
VMD Auction Status
VMD1
VMD2
VMD3
VMD4
Command1

```
Option Explicit
Option Compare Text
Option Base 1

'EDA Function Declarations

' functions specific to VB (not standard eda fcns)
Declare Sub eda_version Lib "vdba.dll" (ByVal V$)
Declare Sub eda_get1_ascii Lib "vdba.dll" (ByVal n$, ByVal t$, ByVal f$, ByVal
$)
Declare Function eda_set1_ascii% Lib "vdba.dll" (ByVal n$, ByVal t$, ByVal f$,
yVal V$)
'
' Note define_group returns a long
'
Declare Function eda_define_group& Lib "vdba.dll" (ByVal Count%, ByVal Detect%)
Declare Sub eda_delete_group Lib "vdba.dll" (ByVal Handle&)
'
' should always pass 0 for vsp
Declare Function eda_define_ntf& Lib "vdba.dll" (ByVal Handle&, ByVal enode$, B
Val etag$, ByVal efield$, ByVal vsp&)
Declare Sub eda_delete_ntf Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
'
Declare Sub eda_lookup Lib "vdba.dll" (ByVal Handle&)
Declare Sub eda_read Lib "vdba.dll" (ByVal Handle&)
Declare Sub eda_write Lib "vdba.dll" (ByVal Handle&)
Declare Function eda_write1% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
Declare Sub eda_wait Lib "vdba.dll" (ByVal Handle&)
Declare Function eda_query_wait% Lib "vdba.dll" (ByVal Handle&)
'
Declare Function eda_get_error% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
Declare Function eda_get_ascii% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, ByVa
 Value$, ByVal maxlen%)
Declare Function eda_set_ascii% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, ByVa
 Value$)

' note pointer to float is passed
Declare Function eda_get_float% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, Valu
!)
Declare Function eda_set_float% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, Val
!)

'eda communication variables
Dim Ghandler As Long
Dim eda_err As Variant
Dim tex As Integer
Dim Newnum As Variant
Dim Node As String * 15
Dim Tags As String * 15
Dim Field As String * 15
Dim answer As String * 80
Dim ans As String * 10
Dim defnode As Long
Dim trap As String * 10
Dim Texts As String * 6
Dim Nodetyp As String * 20
```

```
OCTRON.FRM - 2

Dim nodes As String * 20
Dim bidrotenable As Variant
Dim BlockTyp As String * 20
Dim reply As Variant
Dim Bidoperatestate As Variant
Dim Bidnodetext As String * 10
Dim bidtokenstate As Variant
Dim Tokenstate As Integer
Dim X As Variant
Dim Dummy As Variant

Sub Commddelink (Text As TextBox, LinkItem)
'This Sub Procedure creates an Automatic link between the
'Queueing application and either the Factory Controller or
 the Cell Controller

Text.LinkMode = none  'If the link is not already
                      'created then create it
Text.LinkTopic = "DMDDE|DATA"  'Link topic is FixDmacs

Text.LinkItem = LinkItem       'This is the Node,
                               'databaseblock and field we
                               'want to link to

Text.LinkMode = Link_Automatic 'Automatic update of data


End Sub

Function eda_requestinfolink (n, t, f)
Ghandler = eda_define_group(1, 0)
defnode = eda_define_ntf(Ghandler, n, t, f, 0)

Call eda_lookup(Ghandler)
Call eda_wait(Ghandler)
Call eda_read(Ghandler)
Call eda_wait(Ghandler)
Newnum = eda_get_ascii(Ghandler, defnode, answer, 50)
Call eda_delete_group(Ghandler)
eda_requestinfolink = answer
End Function

Sub RunDDEServer ()
AppActivate "Microsoft Visual Basic" 'Subject is VB
X = Shell("dmdde.exe", 1) 'Makes sure The DDE  server is going
Dummy = DoEvents()

End Sub

Sub SendInfo (n, t, f, Text)
Ghandler = eda_define_group(1, 0)
defnode = eda_define_ntf(Ghandler, n, t, f, 0)

Call eda_lookup(Ghandler)
Call eda_wait(Ghandler)
eda_err = eda_get_error(Ghandler, defnode)
```

```
Text5.Text = Str$(eda_err)
Newnum = Text
reply = eda_set_ascii(Ghandler, defnode, Newnum)
Call eda_write(Ghandler)
Call eda_wait(Ghandler)
Call eda_delete_group(Ghandler)

End Sub

Sub BidControl_Change ()

'this textbox prompts the program to let each VMD have a chance to hold an auct
on, when a
'particular VMD has finished holding an auction then it sends a message to this
application
'telling it that it has finished and to either let the next VMD have a go or if
all VMDs have
'tried then release the bidtoken and let it go to another Cell Controller or th
 Factory Controller.



Select Case Val(Bidcontrol.Text)



Case Is = 1

'*********** Now we want each Vmd to be given the chance to hold an auction****
**********]

''If VMD1.Value = True Then
    VMD1Auctionstate.Picture = LoadPicture("C:\CellCont\Bitmaps\Trffc10a.ico")
    VMD2Auctionstate.Picture = LoadPicture("C:\CellCont\Bitmaps\Trffc10c.ico")
    VMD3Auctionstate.Picture = LoadPicture("C:\CellCont\Bitmaps\Trffc10c.ico")
    VMD4Auctionstate.Picture = LoadPicture("C:\CellCont\Bitmaps\Trffc10c.ico")
    'now we are required to send a prompt to the VMD1 telling it to start the a
ctioning process

   ' Node = "C01"'CStr(Auctrun.Nodename.Text)
    'MsgBox (Node)
   ' Tags = "VMDCurAuct"
   ' texts = "1"
   ' Field = "A_CV"
   ' Call SendInfo(Node, Tags, Field, texts)

'Dim Node As String * 15
'Dim Tags As String * 15
'Dim Field As String * 15


''End If

Case Is = 2
```

```
'If VMD2.Value = True Then
    VMD1Auctionstate.Picture = LoadPicture("C:\CellCont\Bitmaps\Trffc10c.ico")
    VMD2Auctionstate.Picture = LoadPicture("C:\CellCont\Bitmaps\Trffc10a.ico")
    VMD3Auctionstate.Picture = LoadPicture("C:\CellCont\Bitmaps\Trffc10c.ico")
    VMD4Auctionstate.Picture = LoadPicture("C:\CellCont\Bitmaps\Trffc10c.ico")

    'now we are required to send a prompt to the VMD2 telling it to start the a
ctioning process

''    Node = CStr(Auctrun.Nodename.Text)
    'MsgBox (Node)
''    Tags = "VMDCurAuct"
''    Texts = "2"
''    Field = "A_CV"
''    Call SendInfo(Node, Tags, Field, Texts)

''End If
'

'Tags = "VMDCurAuct"
'Texts = "2"
 Call SendInfo(node, Tags, field, Texts)

Case Is = 3

 'If VMD3.Value = True Then
    VMD1Auctionstate.Picture = LoadPicture("C:\CellCont\Bitmaps\Trffc10c.ico")
    VMD2Auctionstate.Picture = LoadPicture("C:\CellCont\Bitmaps\Trffc10c.ico")
    VMD3Auctionstate.Picture = LoadPicture("C:\CellCont\Bitmaps\Trffc10a.ico")
    VMD4Auctionstate.Picture = LoadPicture("C:\CellCont\Bitmaps\Trffc10c.ico")

    'now we are required to send a prompt to the VMD3 telling it to start the a
ctioning process

  ''  Node = CStr(Auctrun.Nodename.Text)
    'MsgBox (Node)
  ''  Tags = "VMDCurAuct"
''    Texts = "3"
''    Field = "A_CV"
  ''  Call SendInfo(Node, Tags, Field, Texts)


''End If



Case Is = 4
''If VMD4.Value = True Then

    VMD1Auctionstate.Picture = LoadPicture("C:\CellCont\Bitmaps\Trffc10c.ico")
    VMD2Auctionstate.Picture = LoadPicture("C:\CellCont\Bitmaps\Trffc10c.ico")
    VMD3Auctionstate.Picture = LoadPicture("C:\CellCont\Bitmaps\Trffc10c.ico")
    VMD4Auctionstate.Picture = LoadPicture("C:\CellCont\Bitmaps\Trffc10a.ico")
```

```
    'now we are required to send a prompt to the VMD3 telling it to start the a
ctioning process

 ''    Node = CStr(Auctrun.Nodename.Text)
    'MsgBox (Node)
  ''   Tags = "VMDCurAuct"
   '' Texts = "4"
   '' Field = "A_CV"
   '' Call SendInfo(Node, Tags, Field, Texts)
''End If
'

Case Is = 5
'at this stage all thge vmd's have been given the opportunity to hold an auctio
. Therefore
'we must release the bidtoken.
    VMD1Auctionstate.Picture = LoadPicture("C:\CellCont\Bitmaps\Trffc10c.ico")
    VMD2Auctionstate.Picture = LoadPicture("C:\CellCont\Bitmaps\Trffc10c.ico")
    VMD3Auctionstate.Picture = LoadPicture("C:\CellCont\Bitmaps\Trffc10c.ico")
    VMD4Auctionstate.Picture = LoadPicture("C:\CellCont\Bitmaps\Trffc10c.ico")

    Tags = "Bidoperate"
    Texts = "0"
    Call SendInfo(Node, Tags, Field, Texts)

    'we are also required to reset this block
    Node = CStr(Auctrun.Nodename.Text)
    Tags = "VMDCurAuct"
    Texts = "0"
    Field = "A_CV"
    Call SendInfo(Node, Tags, Field, Texts)


End Select



End Sub

Sub BidToken_Change ()

Select Case Val(BidToken.Text)

Case Is = 1

'if this textbox turns into a "1" then this means that this cell controller has
recieved the bid
'token and that it must now prompt each VMD to try and auction an item off it'
outgoing queue.

'However the first thing that it is required to do is to stop the bid rotataio
application from
'rotating the token any further so that the cell controller can hold the auctio
.
```

```
'therefore we must send a "1" to the bidoperate database block to lock it.

'define the node that this application resides on
Node = Nodename.Text

 the block to send to is the "Bidoperate" database block
Tags = "Bidoperate"
 the field is ascii current value
ield = "A_CV"
'the item to be sent is a "1"
 exts = "1"

Call SendInfo(Node, Tags, Field, Texts)

 set the bidcontrol textbox to "1" to allow this application to give each VMD th
e opportunity to
'hold an auction.
 ommand1_Click


 idToken.Text = 0
Case Else
'if any other number is entered a mistake has occured so dont allow any programm
 ng to be
'executed.
    Exit Sub
 nd Select


 nd Sub

Sub Command1_Click ()
 ode = Trim(Nodename.Text)
lockTyp = "vmdcurauct"
Field = "A_CV"
 exts = "1"
all SendInfo(Node, BlockTyp, Field, Texts)

 nd Sub

Sub Form_Load ()
'When this form loads we are required to find out what VMD's are active so that
 e know
'what VMD's we should give the chance to conduct an auction for.
im Filenum As Integer
 im Thisstring As String
Dim VMD1activity As String
im VMD2Activity As String
 im VMD3Activity As String
Dim VMD4Activity As String

n Error GoTo Checkerror:

'first find out what node this  program is running on
ilenum = FreeFile
pen "c:\Cellcont\Cellname.txt" For Input Shared As Filenum Len = 10
```

```
Line Input #Filenum, Thisstring
Close Filenum

'pass the name of the node to the nodename textbox.
Nodename.Text = Trim(Thisstring)


'******REQUEST ACTIVITY STATUS OF VMD1******
Node = Nodename.Text
Tags = "V1MACHTYP"
Field = "A_CV"

VMD1activity = eda_requestinfolink(Node, Tags, Field)

MsgBox (VMD1activity)
If Val(VMD1activity) <> 0 Then
'check the checkbox to say that it is active.
    VMD1.Value = True
End If

'******REQUEST ACTIVITY STATUS OF VMD2******
Tags = "V2MACHTYP"
Field = "A_CV"

VMD2Activity = eda_requestinfolink(Node, Tags, Field)

If Val(VMD2Activity) <> 0 Then
'check the checkbox to say that it is active.
    VMD2.Value = True
End If
'
'******REQUEST ACTIVITY STATUS OF VMD3******
Tags = "V3MACHTYP"
Field = "A_CV"

VMD3Activity = eda_requestinfolink(Node, Tags, Field)

If Val(VMD3Activity) <> 0 Then
'check the checkbox to say that it is active.
    VMD3.Value = True
End If


'******REQUEST ACTIVITY STATUS OF VMD4******
Tags = "V4MACHTYP"
Field = "A_CV"
''
VMD4Activity = eda_requestinfolink(Node, Tags, Field)

If Val(VMD4Activity) <> 0 Then
'check the checkbox to say that it is active.
    VMD4.Value = True
End If
```

```
OCTRON.FRM - 6

'***************NOW FORM A LINK BETWEEN THE BIDTOKEN TEXTBOX AND THE BIDTOKEN BL(
 K***************

'We are required to form a link between these so when this cell is passed the b.
 token then this
 application can tell and then prompt each VMD to hold an auction and then pass
the BidToken to
 the next Cell Controller or Factory Controller once all VMD's have been given
..e chance to hold
'an auction.


'first we have to run the DDE Server to allow Dynamic Data Exchange to occur.
 all RunDDEServer

'now we are required to form the link between the bidtoken textbox and the bidt
 en database
 block
'ndf

 all Commddelink(BidToken, Nodename.Text & "." & "Bidtoken.A_CV")


 now we are required to create a link between the bidcontrol textbox and the
 "VMDCurAuct" block
 all Commddelink(Bidcontrol, Nodename.Text & "." & "VMDCurAuct.A_CV")

 xit Sub

 heckerror:

f Err = 5 Then
 esume Next
    Else
    MsgBox ("The error is " & Err)
    Resume Next
 nd If

nd Sub
```

```
'Written by Matthew J Jolly
'PhD Student
'Production Technology Department
'Massey University
'Palmerston North

'State all arrays begin at 1
Option Base 1
'State that all variables must be declared explictly
Option Explicit
'State that text comparison is not case-sensitive
Option Compare Text



' MsgBox parameters
Global Const MB_OK = 0                    ' OK button only
Global Const MB_OKCANCEL = 1              ' OK and Cancel buttons
Global Const MB_ABORTRETRYIGNORE = 2      ' Abort, Retry, and Ignore buttons
Global Const MB_YESNOCANCEL = 3           ' Yes, No, and Cancel buttons
Global Const MB_YESNO = 4                 ' Yes and No buttons
Global Const MB_RETRYCANCEL = 5           ' Retry and Cancel buttons

Global Const MB_ICONSTOP = 16             ' Critical message
Global Const MB_ICONQUESTION = 32         ' Warning query
Global Const MB_ICONEXCLAMATION = 48      ' Warning message
Global Const MB_ICONINFORMATION = 64      ' Information message

Global Const MB_APPLMODAL = 0             ' Application Modal Message Box
Global Const MB_DEFBUTTON1 = 0            ' First button is default
Global Const MB_DEFBUTTON2 = 256          ' Second button is default
Global Const MB_DEFBUTTON3 = 512          ' Third button is default
Global Const MB_SYSTEMMODAL = 4096         'System Modal

' MsgBox return values
Global Const IDOK = 1                     ' OK button pressed
Global Const IDCANCEL = 2                 ' Cancel button pressed
Global Const IDABORT = 3                  ' Abort button pressed
Global Const IDRETRY = 4                  ' Retry button pressed
Global Const IDIGNORE = 5                 ' Ignore button pressed
Global Const IDYES = 6                    ' Yes button pressed
Global Const IDNO = 7                     ' No button pressed



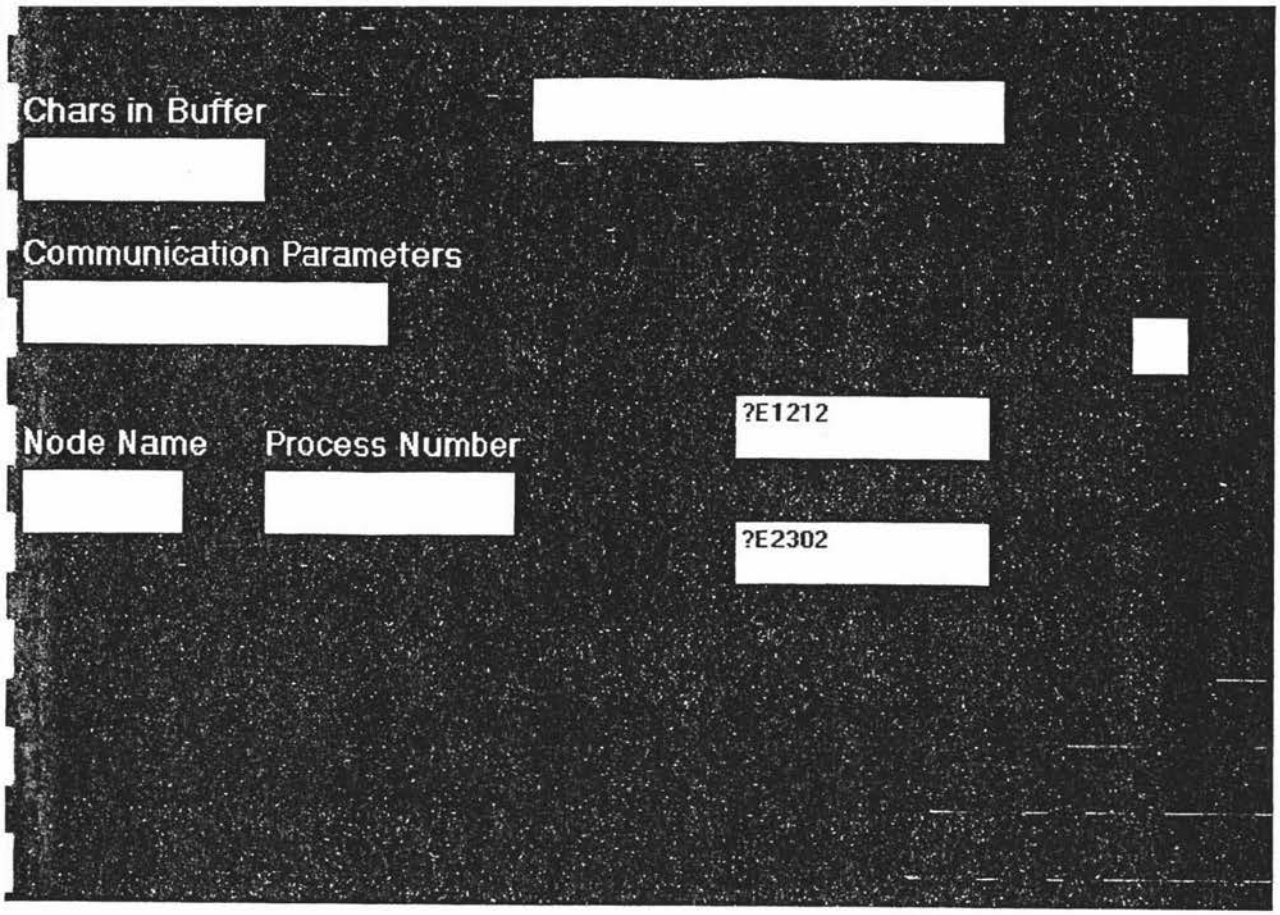
'******************** DDE Constants ***********************
' LinkMode (forms and controls)
Global Const NONE = 0           ' 0 - None
Global Const LINK_SOURCE = 1    ' 1 - Source (forms only)
Global Const LINK_AUTOMATIC = 1 ' 1 - Automatic (controls only)
Global Const LINK_MANUAL = 2    ' 2 - Manual (controls only)
Global Const LINK_NOTIFY = 3    ' 3 - Notify (controls only)


'*********** Declare Variables specific to the Auction Management Application**
***********
```

A.3 Implementation of the Hybrid Program Download Application

This applications main responsibility is to ensure that the correct NC program is downloaded to the correct device for the correct task being considered. A copy of the application exists for each device(controlled by the cell concerned) and data is transmitted in accordance with the required communications requirements of the device being communicated with.

The material following outlines the Microsoft Visual Basic programming written to implement the application.


Chars in Buffer
Communication Parameters
Node Name
Process Number
?E1212
?E2302

```
DDE1.FRM - 1

Option Explicit
Option Compare Text

Dim Msg As String
Dim Firsttime As Double
Dim Filename As String
Dim Checkfilename As String
Const Close_DDE = 5, Open_DDE = 1, Flow_Halt = 1, Flow_Go = 2
Dim Title As String * 30
Dim CommparametersPort1  As String
Dim CommHandshakePort1  As Integer
Dim Dummy


'EDA Function Declarations

' functions specific to VB (not standard eda fcns)
Declare Sub eda_version Lib "vdba.dll" (ByVal V$)
Declare Sub eda_get1_ascii Lib "vdba.dll" (ByVal n$, ByVal t$, ByVal f$, ByVal
$)
Declare Function eda_set1_ascii% Lib "vdba.dll" (ByVal n$, ByVal t$, ByVal f$,
ByVal V$)

' Note define_group returns a long
'
Declare Function eda_define_group& Lib "vdba.dll" (ByVal Count%, ByVal Detect%)
Declare Sub eda_delete_group Lib "vdba.dll" (ByVal Handle&)
'
' should always pass 0 for vsp
Declare Function eda_define_ntf& Lib "vdba.dll" (ByVal Handle&, ByVal enode$, B
Val etag$, ByVal efield$, ByVal vsp&)
Declare Sub eda_delete_ntf Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
'
Declare Sub eda_lookup Lib "vdba.dll" (ByVal Handle&)
Declare Sub eda_read Lib "vdba.dll" (ByVal Handle&)
Declare Sub eda_write Lib "vdba.dll" (ByVal Handle&)
Declare Function eda_write1% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
Declare Sub eda_wait Lib "vdba.dll" (ByVal Handle&)
Declare Function eda_query_wait% Lib "vdba.dll" (ByVal Handle&)
'
Declare Function eda_get_error% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
Declare Function eda_get_ascii% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, ByVa
 value$, ByVal maxlen%)
Declare Function eda_set_ascii% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, ByVa
 value$)

' note pointer to float is passed
Declare Function eda_get_float% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, valu
!)
Declare Function eda_set_float% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, valu
!)

'eda communication variables
Dim Ghandler As Long
Dim eda_err As Variant
```

```
Dim tex As Integer
Dim Newnum As Variant
Dim Node As String * 15
Dim Tags As String * 15
Dim Field As String * 15
Dim Answer As String * 80
Dim ans As String * 10
Dim defnode As Long
Dim trap As String * 10
Dim texts As String * 6
Dim Nodetyp As String * 20
Dim nodes As String * 20
Dim bidrotenable As Variant
Dim BlockTyp As String * 20
Dim Reply As Variant
Dim Bidoperatestate As Variant
Dim Bidnodetext As String * 10
Dim bidtokenstate As Variant
Dim Tokenstate As Integer

Function CheckNodeName (Msg)

Title = "Configure Node Name"
CheckNodeName = InputBox(Msg, Title)
End Function

Sub CommOutPort1 (Filename, CommHandshakePort1)
Dim Stringcount As Integer, TransmitChar, Msg, x
Dim Firsttime, SecondTime As Double

Comm1.Handshaking = CommHandshakePort1
Comm1.CommPort = 1
Comm1.PortOpen = True
Comm1.Settings = Trim(CStr(Text4.text))


Open Filename For Input As #1

Do While Not EOF(1)
DoEvents
TransmitChar = Input(1, #1)
Comm1.Output = TransmitChar
Stringcount = Stringcount + 1

If Stringcount = 3000 Or CInt(Text2.text) = 2 Then
    If Stringcount = 3000 Then
        Text5.text = Comm1.OutBufferCount
        Firsttime = Timer
        Do Until Timer > Firsttime + 15
        Loop
        Comm1.OutBufferCount = 0
        Else
            'the Program Invocation loop has stopped transmission
            MsgBox ("Transmission Halted")
            Stringcount = 0
            Close #1
```

```
            Comm1.PortOpen = False
            Stringcount = 0

            'Reset the interupt block and exit
            Node = Nodename.text
            BlockTyp = "V" & Trim(VMD_Number.text) & "STOPDOWN"
            Field = "A_CV"
            texts = "1"
            Call SendInfo(Node, BlockTyp, Field, texts)

            Exit Sub
    End If
    Stringcount = 0
End If
Loop

SecondTime = Timer
Do Until Timer > SecondTime + 10
Loop
Msg = MsgBox("Transmission Complete,48,Communication Status")
Close #1
Comm1.PortOpen = False
Stringcount = 0
'Now we must indicate that the transmission of data from
'the cnc filename is complete as we must wait until it is
'complete before we try to run the program.We do this by
'sending a "1" to a block called "V1LCNCComp"(V1 Load CNC
'File is Complete) using eda communications

Node = Nodename.text
BlockTyp = "V" & Trim(VMD_Number.text) & "DOWNCOMP"
Field = "A_CV"
texts = "1"
Call SendInfo(Node, BlockTyp, Field, texts)

Node = Nodename.text
BlockTyp = "V" & Trim(VMD_Number.text) & "CNCSTART"
Field = "F_CV"
texts = "0"
Call SendInfo(Node, BlockTyp, Field, texts)

End Sub

Function eda_requestinfolink (n, t, f)
Ghandler = eda_define_group(1, 0)
defnode = eda_define_ntf(Ghandler, n, t, f, 0)

Call eda_lookup(Ghandler)
Call eda_wait(Ghandler)
Call eda_read(Ghandler)
Call eda_wait(Ghandler)
Newnum = eda_get_ascii(Ghandler, defnode, Answer, 50)
Call eda_delete_group(Ghandler)
eda_requestinfolink = Answer
End Function
```

```
Sub SendInfo (n, t, f, text)
Ghandler = eda_define_group(1, 0)
defnode = eda_define_ntf(Ghandler, n, t, f, 0)

Call eda_lookup(Ghandler)
Call eda_wait(Ghandler)
eda_err = eda_get_error(Ghandler, defnode)
Newnum = text
Reply = eda_set_ascii(Ghandler, defnode, Newnum)
Call eda_write(Ghandler)
Call eda_wait(Ghandler)
Call eda_delete_group(Ghandler)

End Sub

Sub Transfer_Data (LinkItem1, Linkitem2)
Const Automatic = 1, none = 0
Dim x As Integer

On Error Resume Next
Text1.LinkMode = none
Text1.LinkTopic = "DMDDE|DATA"
Text1.LinkItem = LinkItem1
Text1.LinkMode = Automatic      'Automatic update of data
On Error Resume Next
Text2.LinkMode = none
Text2.LinkTopic = "DMDDE|DATA"
Text2.LinkItem = Linkitem2
Text2.LinkMode = Automatic



x = Shell("Clipbrd.exe", 2)
AppActivate "Microsoft Visual Basic"
SendKeys "%F", True
SendKeys "{Down}", True
SendKeys "{Down}", True
SendKeys "{Down}", True
SendKeys "{Enter}", True

Dummy = DoEvents()

MsgBox "DDE Link Activated"
AppActivate "Microsoft Visual Basic"

End Sub

Sub Comm1_OnComm ()
Select Case Comm1.CommEvent
Case MSCOMM_EV_SEND
Case MSCOMM_EV_RECEIVE
Case MSCOMM_EV_CTS
Case MSCOMM_EV_DSR
Case MSCOMM_EV_CD
Case MSCOMM_EV_RING
```

```
DE1.FRM - 5

Case MSCOMM_EV_EOF

'Error code constants
Case MSCOMM_ER_BREAK
ase MSCOMM_ER_CTSTO
ase MSCOMM_ER_DSRTO
Case MSCOMM_ER_FRAME
ase MSCOMM_ER_OVERRUN
ase MSCOMM_ER_CDTO
Case MSCOMM_ER_RXOVER
ase MSCOMM_ER_RXPARITY
ase MSCOMM_ER_TXFULL
MsgBox ("Transmit Buffer is full!")
esume Next
nd Select
End Sub

ub Form_Load ()
Dim x, i As Integer
im Firsttime As Double
im NodeNam As Variant
Dim Answer As String * 5
im DefVal, Msg, Title  ' Declare variables.
eda stuff
im Thisbaudrate As String * 10
im DeviceParity As String * 10
im DeviceStopNDatabits As String * 10
im Fullstring As String * 30


n Error Resume Next


 Request the name of the node being used
Again:
 Call Function which asks user to enter  NodeName
sg = "Please Enter the name of the node being used"
NodeNam = CheckNodeName(Msg)
 Check to see if valid Node Name entered

Nodename.text = UCase(NodeNam)


'request the number of the process this is to be used with

sg = "Enter the a value from 1 to 4."  ' Set prompt.
Title = "InputBox Demo" ' Set title.
efVal = "1"    ' Set default return value.

Answer = InputBox(Msg, Title, DefVal)   ' Get user input.

MD_Number.text = Trim(Answer)

orm1.Show
ppActivate "Microsoft Visual Basic" 'Subject is VB
 = Shell("dmdde.exe", 1) 'Makes sure The DDE  server is going
```

```
Counter = 1
Dummy = DoEvents()

'set up links that controls transmission

Call Transfer_Data(Trim(Nodename.text) & "." & "V" & Trim(Answer) & "CNCSTART.F_
CV", Trim(Nodename.text) & "." & "V" & Trim(Answer) & "STOPDOWN.F_CV")



'get communication parameters the device uses

'get the device baudrate
Node = Trim(Nodename.text)
Tags = "V" & Trim(VMD_Number.text) & "BAUDRATE"

Field = "A_CV"

Thisbaudrate = eda_requestinfolink(Trim(Node), Tags, Field)


'get the device parity rating
Node = Trim(Nodename.text)
Tags = "VMD" & Trim(VMD_Number.text) & "PARITY"

Field = "A_CV"

DeviceParity = eda_requestinfolink(Trim(Node), Tags, Field)


'get the stop/data bit communication configuration
Node = Trim(Nodename.text)
Tags = "V" & Trim(VMD_Number.text) & "STPSTART"

Field = "A_CV"

DeviceStopNDatabits = eda_requestinfolink(Trim(Node), Tags, Field)


'combine parts to give full rating
Fullstring = Trim(Thisbaudrate) & "," & Trim(DeviceParity) & "," & Trim(DeviceSt
opNDatabits)


'place trimmed string in communication parameter textbox

Text4.text = Trim(Fullstring)



End Sub

Sub Text1_Change ()
Dim x As Integer
Dim Thisonesfilename As String * 50
```

```
DDE1.FRM - 7

Loopy:
If CStr(Text1.text) = "?E1212" Or CStr(Text1.text) = "?E2302" Then
AppActivate "Microsoft Visual Basic"
x = Shell("Clipbrd.exe", 2)
SendKeys "%F", True
SendKeys "{Down}", True
SendKeys "{Down}", True
SendKeys "{Enter}", True
Dummy = DoEvents()
GoTo Loopy
End If

If CInt(Text1.text) = Open_DDE And Counter <> 1 Then

'get the name of the NC filename to download using eda

Node = Trim(Nodename.text)
Tags = "VMD" & Trim(VMD_Number.text) & "NCFILE"

Field = "A_CV"

Thisonesfilename = eda_requestinfolink(Trim(Node), Tags, Field)

Text3.text = Trim(Thisonesfilename)

Filename = Trim(Thisonesfilename)

Call CommOutPort1(Filename, MSCOMM_HANDSHAKE_XONXOFF)
Else
End If
Counter = 0
End Sub

Sub Text3_Change ()
If Text3.text <> " " Then
Filename = Text3.text
End If
End Sub

Sub Text5_Click ()
Text5.text = Comm1.OutBufferCount
End Sub
```

```
Global Const MSCOMM_EV_SEND = 1
Global Const MSCOMM_EV_RECEIVE = 2
Global Const MSCOMM_EV_CTS = 3
Global Const MSCOMM_EV_DSR = 4
Global Const MSCOMM_EV_CD = 5
Global Const MSCOMM_EV_RING = 6
Global Const MSCOMM_EV_EOF = 7

'Error code constants
Global Const MSCOMM_ER_BREAK = 1001
Global Const MSCOMM_ER_CTSTO = 1002
Global Const MSCOMM_ER_DSRTO = 1003
Global Const MSCOMM_ER_FRAME = 1004
Global Const MSCOMM_ER_OVERRUN = 1006
Global Const MSCOMM_ER_CDTO = 1007
Global Const MSCOMM_ER_RXOVER = 1008
Global Const MSCOMM_ER_RXPARITY = 1009
Global Const MSCOMM_ER_TXFULL = 1010
Global Const MSCOMM_HANDSHAKE_XONXOFF = 1

Global Counter
```

```
QUEUE1.BAS - 1

Option Explicit

'**For queue program

 I don't know why this works but it does, variant's should not work in a type d
claration

Type queue

    Job_Name As Variant
    IF_Filename As Variant
    No_ofparts As Variant
    Setup_Time As Variant
    Tools_Required As Variant
    Machine_Type As Variant
    Cycle_Time As Variant
    Loadunload_time As Integer

End Type

' MsgBox parameters
Global Const MB_OK = 0                    ' OK button only
Global Const MB_OKCANCEL = 1              ' OK and Cancel buttons
Global Const MB_ABORTRETRYIGNORE = 2      ' Abort, Retry, and Ignore buttons
Global Const MB_YESNOCANCEL = 3           ' Yes, No, and Cancel buttons
Global Const MB_YESNO = 4                 ' Yes and No buttons
Global Const MB_RETRYCANCEL = 5           ' Retry and Cancel buttons

Global Const MB_ICONSTOP = 16             ' Critical message
Global Const MB_ICONQUESTION = 32         ' Warning query
Global Const MB_ICONEXCLAMATION = 48      ' Warning message
Global Const MB_ICONINFORMATION = 64      ' Information message

Global Const MB_APPLMODAL = 0             ' Application Modal Message Box
Global Const MB_DEFBUTTON1 = 0            ' First button is default
Global Const MB_DEFBUTTON2 = 256          ' Second button is default
Global Const MB_DEFBUTTON3 = 512          ' Third button is default
Global Const MB_SYSTEMMODAL = 4096         'System Modal

' MsgBox return values
Global Const IDOK = 1                     ' OK button pressed
Global Const IDCANCEL = 2                 ' Cancel button pressed
Global Const IDABORT = 3                  ' Abort button pressed
Global Const IDRETRY = 4                  ' Retry button pressed
Global Const IDIGNORE = 5                 ' Ignore button pressed
Global Const IDYES = 6                    ' Yes button pressed
Global Const IDNO = 7                     ' No button pressed


'** For file IF control


' Define job record type and global file management variables

  Status variables for file manipulation
```

```
Function loadApplication (Appname, Path) As Variant

'Written by Michael Butler
'Department of Production Technology
'Massey University
'
'This function takes an application name and path then attempts to start it
'if any errors occur it aborts, returns loadapplication yes or no ie it has bee
 loaded
'
'
'Improvement can be made here by testing if the application is open before it
 opened using an API call
'

Dim response, x                                         'response from message
x
Const IDYES = 6, IDNO = 7                               'return from message box

    On Error GoTo ErrorHandel   'if an error occurs goto errorhandel
    x = Shell(Path & Appname)
    loadApplication = True

Exit Function

ErrorHandel:

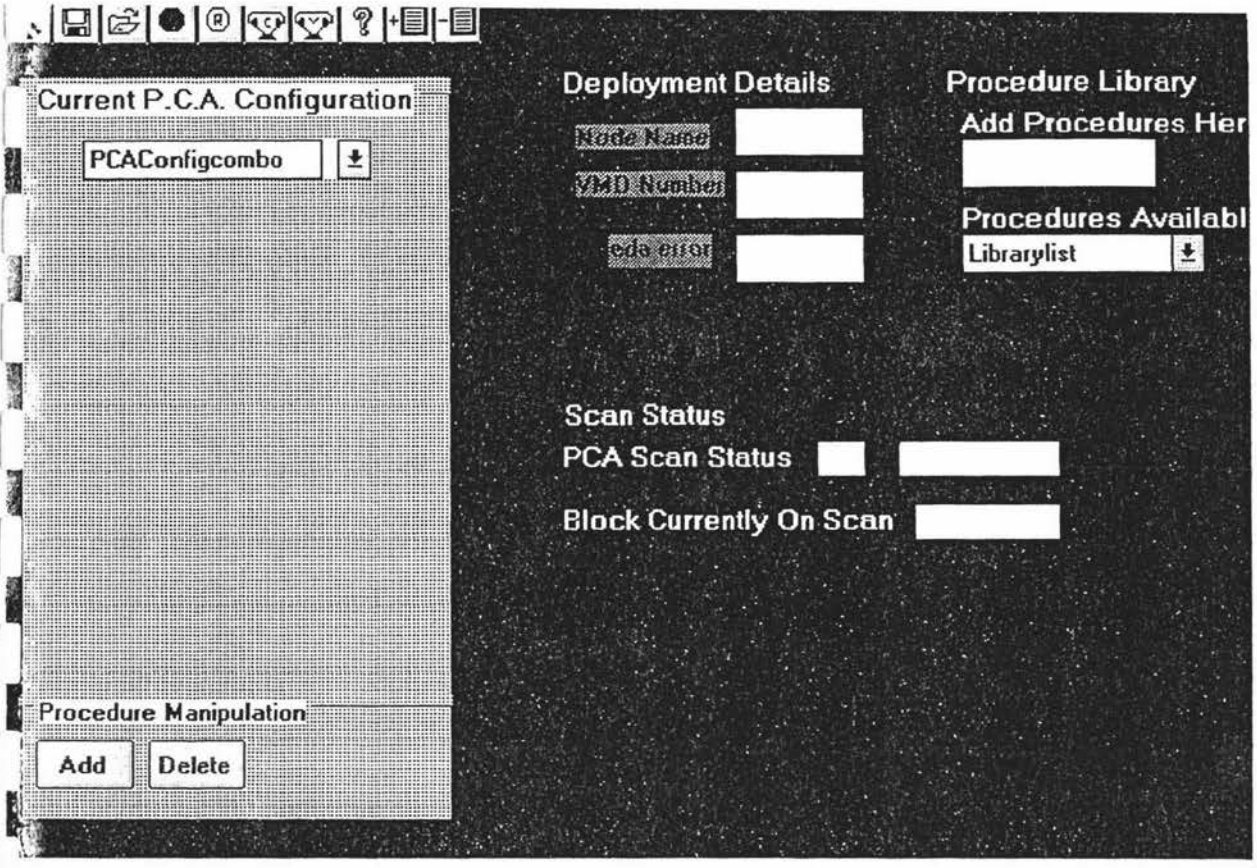
    response = MsgBox("Is " & Appname & " currently running?", MB_YESNOCANCEL,
Application Status")      'Ask if want to start application
    If response = IDYES Then
        loadApplication = True
    Else
        MsgBox "Error in opening application, this operation may not continue",
48, "Application Status"
        loadApplication = False
    End If
    Exit Function

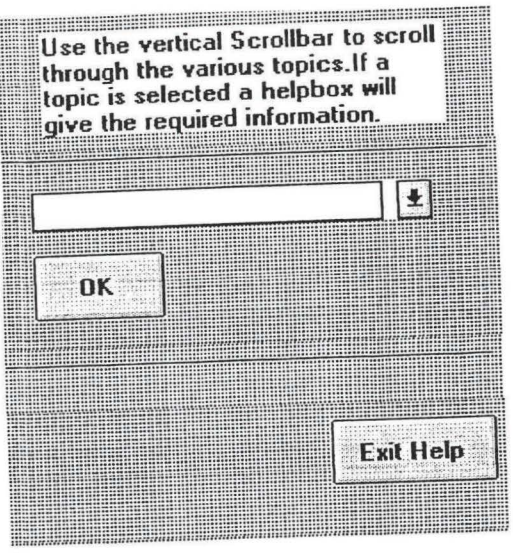
End Function
```

A.4 Implementation of the Hybrid Process Control Application

The Hybrid Process Control Application is responsible for the ordering (and execution of) the FIXDMACS database blocks which emulate the Process Behaviour Algorithm. As these blocks have a link to the real world manufacturing processes they directly control the series of steps the process goes through to complete a task.

The material following outlines the Microsoft Visual Basic programming written to implement the application.


Current P.C.A. Configuration
PCAConfigcombo
Procedure Manipulation
Add
Delete
Deployment Details
Node Name
VMID Number
cda error
Procedure Library
Add Procedures Her
Procedures Availabl
Librarylist
Scan Status
PCA Scan Status
Block Currently On Scan


Use the vertical Scrollbar to scroll through the various topics.If a topic is selected a helpbox will give the required information.
OK
Exit Help

```
CACONFG.BAS - 1

ption Explicit
This application is responsible for the development and
'coordination of the P.C.A.(Process Control Algorithm) and
has three main areas of responsibilty:

'1)The updating of libraries of FixDmacs procedures
'2)The saving of P.C.A. Configurations
 3)The co-ordination of P.C.A. execution

ption Base 1

' MsgBox parameters
Global Const MB_OK = 0                    ' OK button only
lobal Const MB_OKCANCEL = 1               ' OK and Cancel buttons
Global Const MB_ABORTRETRYIGNORE = 2      ' Abort, Retry, and Ignore buttons
Global Const MB_YESNOCANCEL = 3           ' Yes, No, and Cancel buttons
lobal Const MB_YESNO = 4                  ' Yes and No buttons
Global Const MB_RETRYCANCEL = 5           ' Retry and Cancel buttons

lobal Const MB_ICONSTOP = 16              ' Critical message
Global Const MB_ICONQUESTION = 32         ' Warning query
Global Const MB_ICONEXCLAMATION = 48      ' Warning message
lobal Const MB_ICONINFORMATION = 64       ' Information message

Global Const MB_APPLMODAL = 0             ' Application Modal Message Box
lobal Const MB_DEFBUTTON1 = 0             ' First button is default
lobal Const MB_DEFBUTTON2 = 256           ' Second button is default
Global Const MB_DEFBUTTON3 = 512          ' Third button is default
lobal Const MB_SYSTEMMODAL = 4096         'System Modal

' MsgBox return values
lobal Const IDOK = 1                      ' OK button pressed
lobal Const IDCANCEL = 2                  ' Cancel button pressed
Global Const IDABORT = 3                  ' Abort button pressed
Global Const IDRETRY = 4                  ' Retry button pressed
lobal Const IDIGNORE = 5                  ' Ignore button pressed
Global Const IDYES = 6                    ' Yes button pressed
Global Const IDNO = 7                     ' No button pressed

Type Nodevmdconfig
Recordsize As Integer
lodename As String * 4
VMDNo As Integer
Recordposi As Integer
nd Type

'Noderecord required in Cell Configuration
ype Nodesrecord
Recordsize As Integer
Nodename As String * 6
nd Type

Type Noodesrecord
lodename As String * 8
Index As Integer
End Type
```

```
'Declare Factory Controller path
Global Const Defaultpath = "C:\"

'Declare the procedure library array variable
Type Procedurelibrary
Libsize As Integer
BlockTagname As String * 12
End Type

'Declare PCA File Configuration storage array variable
Type PCARecordtyp
Recordsize As Integer
BlockTagname As String * 12
End Type

'Declare help file array variable
Type HelpFileType
Recordnumber As Integer
HelpText As String * 120
Helprecordname As String * 10
End Type



'Declare combo list modification variable
Dim Temp() As String * 14

'Declare file saved check exit variable
Global Filealtered As Integer

'Declare PCA currently running variable
Global PCARunning As Integer

'******************** DDE Constants **********************
' LinkMode (forms and controls)
Global Const NONE = 0          ' 0 - None
Global Const LINK_SOURCE = 1     ' 1 - Source (forms only)
Global Const LINK_AUTOMATIC = 1 ' 1 - Automatic (controls only)
Global Const LINK_MANUAL = 2     ' 2 - Manual (controls only)
Global Const LINK_NOTIFY = 3     ' 3 - Notify (controls only)

'Procedure fire position variable
Global Currentproc As Integer

'Declare global PCA State variable
Global PCAStopped As Integer
```

```
CACONFG.FRM - 1

ption Explicit


im pcaprocedures() As String * 10

General Declarations
im X As Variant
Dim Dummy As Variant
im Text As Variant
im NodeNam As String * 10
im Msg As String * 40
Dim Title As String * 35
im UpdateInQueueCost As Double
im Justidle As Integer


Nullify Case sensitivity in "like" Functions
Option Compare Text



EDA Function Declarations

 functions specific to VB (not standard eda fcns)
eclare Sub eda_version Lib "vdba.dll" (ByVal V$)
Declare Sub eda_get1_ascii Lib "vdba.dll" (ByVal n$, ByVal t$, ByVal f$, ByVal V
)
eclare Function eda_set1_ascii% Lib "vdba.dll" (ByVal n$, ByVal t$, ByVal f$, B
yVal V$)

 Note define_group returns a long

eclare Function eda_define_group& Lib "vdba.dll" (ByVal Count%, ByVal Detect%)
eclare Sub eda_delete_group Lib "vdba.dll" (ByVal Handle&)

' should always pass 0 for vsp
eclare Function eda_define_ntf& Lib "vdba.dll" (ByVal Handle&, ByVal enode$, By
val etag$, ByVal efield$, ByVal vsp&)
Declare Sub eda_delete_ntf Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)

eclare Sub eda_lookup Lib "vdba.dll" (ByVal Handle&)
Declare Sub eda_read Lib "vdba.dll" (ByVal Handle&)
eclare Sub eda_write Lib "vdba.dll" (ByVal Handle&)
eclare Function eda_write1% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
Declare Sub eda_wait Lib "vdba.dll" (ByVal Handle&)
eclare Function eda_query_wait% Lib "vdba.dll" (ByVal Handle&)

Declare Function eda_get_error% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
eclare Function eda_get_ascii% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, ByVa
value$, ByVal maxlen%)
Declare Function eda_set_ascii% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, ByVa
value$)

' note pointer to float is passed
```

```
Declare Function eda_get_float% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, value
!)
Declare Function eda_set_float% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, value
!)

'eda communication variables
Dim Ghandler As Long
Dim eda_err As Variant
Dim tex As Integer
Dim Newnum As Variant
Dim Node As String * 15
Dim Tags As String * 15
Dim Field As String * 15
Dim answer As String * 80
Dim ans As String * 10
Dim defnode As Long
Dim trap As String * 10
Dim texts As String * 6
Dim Nodetyp As String * 20
Dim nodes As String * 20
Dim bidrotenable As Variant
Dim BlockTyp As String * 20
Dim Reply As Variant
Dim Bidoperatestate As Variant
Dim Bidnodetext As String * 10
Dim bidtokenstate As Variant
Dim Tokenstate As Integer

Dim Nodrecs(1) As noodesrecord

Sub Combo1_Change ()

End Sub

Sub CommDDELink (Text As TextBox, LinkItem)
'This Sub Procedure creates an Automatic link between the
'Queueing application and either the Factory Controller or
'the Cell Controller

Text.LinkMode = none  'If the link is not already
                              'created then create it
Text.LinkTopic = "DMDDE|DATA"  'Link topic is FixDmacs

Text.LinkItem = LinkItem       'This is the Node,
                              'databaseblock and field we
                              'want to link to

Text.LinkMode = Link_Automatic 'Automatic update of data


End Sub

Function eda_requestinfolink (n, t, f)
Ghandler = eda_define_group(1, 0)
defnode = eda_define_ntf(Ghandler, n, t, f, 0)
```

```
all eda_lookup(Ghandler)
all eda_wait(Ghandler)
Call eda_read(Ghandler)
all eda_wait(Ghandler)
ewnum = eda_get_ascii(Ghandler, defnode, answer, 50)
Call eda_delete_group(Ghandler)
da_requestinfolink = answer
nd Function

ub mnu_timer (ThisTime)
hisTime = Timer
Do Until Timer > ThisTime + .15
oop

nd Sub

ub Opengetfilelength (TheFilesname, Recordtype, VarLengthArrayname())

End Sub

ub Picture1_Click ()
esetpic.Picture = LoadPicture("C:\visual\bitmaps\toolbar3\rst_dwn.bmp")
nd Sub

Sub RunDDEServer ()
ppActivate "Microsoft Visual Basic" 'Subject is VB
 = Shell("dmdde.exe", 1) 'Makes sure The DDE  server is going
Dummy = DoEvents()

nd Sub

ub Runpca (Thisproc)

'Notify the user of which block is being scanned
Block_Scanned.Text = PCAConfigcombo.List(Thisproc)

Node = pcanode.Text
BlockTyp = PCAConfigcombo.List(Thisproc)
ield = "A_SCAN"
exts = "on"
Call SendInfo(Node, BlockTyp, Field, texts)

nd Sub

ub SendInfo (n, t, f, Text)
handler = eda_define_group(1, 0)
defnode = eda_define_ntf(Ghandler, n, t, f, 0)

all eda_lookup(Ghandler)
Call eda_wait(Ghandler)
eda_err = eda_get_error(Ghandler, defnode)
ext5.Text = Str$(eda_err)
Newnum = Text
Reply = eda_set_ascii(Ghandler, defnode, Newnum)
Call eda_write(Ghandler)
Call eda_wait(Ghandler)
```

```
Call eda_delete_group(Ghandler)

End Sub

Sub Add_Click ()
addpic_Click
End Sub

Sub Add_proc_Click ()
Dim Msg
Dim DgDef As Integer
Dim response As Integer
Dim I As Integer

'This is the event that occurs whereby the user has selected
'a procedure to be added from the "procedure update" listbox
'and wants to add it to the currently opened library at
'a specified position.The position at which the procedure
'is added determines when that procedure will be fired.The
'procedure is added by selecting a new procedure from the
'Librarylist combo box and then selecting a procedure from
'the PCAConfigcombo combo box such that the new procedure
'being added will be added after the one selected from the
'PCAConfigcombo combo box


'Check to see if an item has been selected from the Librarylist
'combo box
If Librarylist.ListIndex = -1 Then
MsgBox ("You didn't select a position for the procedure to be added")
Exit Sub
End If

'Check to see if the procedure insertion position has been selected
If PCAConfigcombo.ListIndex = -1 Then
      'When a new procedure is being added to the "PCAConfigcombo" combo list
      'and the list is empty it is not possible to select insertion position
      'therefore PCAConfigcombo.ListIndex = -1 therefore the user must be
      'asked whether or not they are trying to add a procedure to an empty
      'list - if they answer yes then the procedure is added to the
      'beginning of the list but if they answer no then the assumption is
      'made that they have simply not selected an insertion point

      Title = "Configuration Query"
      Msg = "Are you adding a procedure to a PCA "
      Msg = Msg & "configuration that currently has"
      Msg = Msg & " no procedures associated with it? "
      DgDef = MB_ICONQUESTION + MB_YESNOCANCEL
      response = MsgBox(Msg, DgDef, Title)
      If response = IDYES Then
          PCAConfigcombo.List(0) = Librarylist.List(Librarylist.ListIndex)
          'there is a gap before the first item on the list - remove it
          PCAConfigcombo.RemoveItem ((PCAConfigcombo.NewIndex))
      End If

      If response = IDCANCEL Then
```

```
            Exit Sub
        End If

        If response = IDNo Then
            MsgBox ("You didn't select a position for the procedure to be added")
            Exit Sub
        End If
    nd If

'otherwise the situation exists where the new procedure can be added
so add


A variable array called "Temp" is going to be used to store the current list
'contents in whilst those contents are being shuffled about - this is required
'so that none of the contents of the list are lost


'Redimension the temp array variable so that all the contents of the list are
able to be held

ReDim Temp(PCAConfigcombo.ListCount + 1)


For I = 0 To PCAConfigcombo.ListCount

emp(I) = PCAConfigcombo.List((PCAConfigcombo.ListIndex) + I) 'Store list item
to be overwritten in temporary variable to avoid losing it

f I = 0 Then
'Transfer new procedure item
CAConfigcombo.List(PCAConfigcombo.ListIndex) = Librarylist.List(Librarylist.Lis
Index)
lse
'The new item has already been added so must reorder the remaining items
CAConfigcombo.List((PCAConfigcombo.ListIndex) + I) = Temp(I - 1)
End If
Next

'Set flag to indicate that the combo box has altered and therefore is
'required to be saved before exiting this application
ilealtered = 1

End Sub

ub addpic_Click ()
On Error Resume Next
im timing As Integer

'This particular event allows the user to add a particular FixDmacs "procedure"
to the library of procedures that already exist - the procedure being the name
of a particular FixDmacs program block

Configure visual menu display
ddpic.Picture = LoadPicture("C:\visual\bitmaps\toolbar3\aproc-dn.bmp")
Call mnu_timer(timing)
```

```
addpic.Picture = LoadPicture("C:\visual\bitmaps\toolbar3\aproc-up.bmp")

'Enable Maskedit box
LibInput.Enabled = True

'Give focus to MaskedEdit box
LibInput.SetFocus
'Set input at start of MaskedEd box
LibInput.SelStart = 0
'Clear current LibInput box value
LibInput.Text = ""

Librarylist.Enabled = False



End Sub

Sub conpic_Click ()
On Error Resume Next
Dim timing As Integer

conpic.Picture = LoadPicture("C:\visual\bitmaps\toolbar3\diagv_dn.bmp")
Call mnu_timer(timing)
conpic.Picture = LoadPicture("C:\visual\bitmaps\toolbar3\diagv_up.bmp")

End Sub

Sub del_proc_Click ()


'What we require here is for the procedure to be selected from the PCAconfigcomb
o
'combo box and then once this button is selected for the procedure to be
'removed from the combo box and all other procedures in that combo box to be
'rearranged about it

'Check to see if the procedure deletion position has been selected
If PCAConfigcombo.ListIndex = -1 Then
MsgBox ("You didn't select a position for the procedure to be added")
Exit Sub
End If

'Delete item selected and all the rest of the procedures are automatically
'reordered

PCAConfigcombo.RemoveItem PCAConfigcombo.ListIndex

End Sub

Sub Delete_Click ()
PCAConfigcombo.Clear
End Sub

Sub Deletepic_Click ()
On Error Resume Next
```

```
CACONFG.FRM - 7

im timing As Integer

Deletepic.Picture = LoadPicture("C:\visual\bitmaps\toolbar3\del_dwn.bmp")
all mnu_timer(timing)
eletepic.Picture = LoadPicture("C:\visual\bitmaps\toolbar3\del_up.bmp")

elete_Click

End Sub

ub Form_Load ()


'This application is the P.C.A Configuration and execution application and
'was designed and written by:

'Matthew John Jolly, Masters Student
'Massey University
'Palmerston North
'New Zealand

im rap As Integer
im Koop As Integer
Dim Msg As String * 80
im I As Integer
im response As Integer
Dim repeat As String * 6
Dim Repair As String * 23
im ans As Integer
Dim Wrong As Variant
Dim Evaluate As Integer
im again As String * 5
Dim Loopy As Integer
Dim Filelength As Variant
im Fname As String * 30
Dim Begin As Integer
Dim NewCost As Single
im InFinishOfQueueTemp As Integer
Dim InStartOfQueuetemp As Integer
Dim Temp As Integer
im nodenum As Integer
Dim First As Integer
Dim Last As Integer
im Thisnode As Integer
Dim nexts  As Variant
Dim Count As Integer
im Noderecord() As nodesrecord
Dim Noderecordlength As Variant
Dim Tryagain As Variant
im Nodrecord As nodesrecord
im Mess As String * 100
Dim Title As String * 40
Dim DgDef As Integer
im Nodrec As noodesrecord
Dim j As Integer
```

```
Dim Proclist() As Procedurelibrary
Dim proclisttype As Procedurelibrary
Dim Proclibmemrq As Variant
Dim Filenum As Integer


 On Error Resume Next


'******** OPEN THE FILE WHICH CONTAINS THE NAME OF THE NODE THIS APPLICATION IS
WORKING ON ***********

'Open the file containing the name of the node

Filelength = Len(Nodrec)
Filenum = FreeFile
Open "c:\Noderec.txt" For Random Access Read Write Shared As Filenum Len = File
ength
Get Filenum, 1, Nodrecs(1)

pcanode.Text = Nodrecs(1).NodeName

Close Filenum

'***STIPULATE THE VMD NUMBER FOR THIS APPLICATION TO BE USED ON**
VMDNo.Text = "1"

'***DEACTIVATE RELEVANT CONTROLS*************
LibInput.Enabled = False
Librarylist.Enabled = False


'*******OPEN UP CURRENT FILE CONTAINING CURRENT LIST OF AVAILABLE PROCEDURES**

Proclibmemrq = Len(proclisttype)
Filenum = FreeFile


ReDim Proclist(1)
Open "C:\V1Proclb.txt" For Random Access Read Write Shared As Filenum Len = Prc
libmemrq
'Read how many records there are in the file

Get Filenum, 1, Proclist(1)

'Redimension array to contain enough memory to hold all
'available procedure names

ReDim Preserve Proclist(Proclist(1).Libsize)
For j = 1 To Proclist(1).Libsize

'Write the name of each available procedure to memory
Get Filenum, j, Proclist(j)
'Write the name of each available procedure to the Combo
'box
```

```
PCACONFG.FRM - 9

ibrarylist.AddItem Proclist(j).Blocktagname
ext

Close the file
lose Filenum


*****FORM THE CONTROL LINK BETWEEN THE PCA SCAN STATUS TEXTBOX AND V1SCANSTAT D
BASE BLOCK
all RunDDEServer

'Create link between "CNC Device Status" Textbox and
Call CommDDELink(Text3, pcanode.Text & "." & "V" & VMDNo.Text & "SCANSTAT" & "."
& "F_CV")

pca_status.Text = "IDLE"


End Sub

ub Form_Unload (Cancel As Integer)
'Declarations
im Msg
im Title As String * 20
Dim DgDef As Integer
im response As Integer

'Check to see if file that has been opened and displayed in the PCAConfigcombo
combo box has been altered - if it has ask the user if he wants to save those
changes

f Filealtered = 1 Then
    Msg = "The currently displayed configuration has been altered. Do you "
    Msg = Msg & "want to save it?"
    Title = "Configuration Modified"
    DgDef = MB_YESNO + MB_ICONQUESTION

    response = MsgBox(Msg, DgDef, Title)
        If response = IDYES Then
        save_click
           Else
           Exit Sub
        End If
End If



nd Sub

ub helppic_Click ()
n Error Resume Next

im timing As Integer

elppic.Picture = LoadPicture("C:\visual\bitmaps\toolbar3\hlp-dwn.bmp")
```

```
Call mnu_timer(timing)
helppic.Picture = LoadPicture("C:\visual\bitmaps\toolbar3\hlp-up.bmp")

PCAHelp_Click

End Sub

Sub LibInput_Change ()
Dim Proclist() As Procedurelibrary
Dim Recordtype As Procedurelibrary
Dim Proclibmemrq As Variant
Dim Filenum As Integer

'make sure that the number of characthers that can be
'entered never exceeds 10
If LibInput.MaxLength > 10 Then
LibInput.MaxLength = 10
End If


'When the name of the procedure has been fully entered into
'the maskedbox then add that name to the list
If Len(LibInput) = 10 Then
      Librarylist.AddItem LibInput, 0
      'Clear the MaskEdbox
     LibInput.Text = ""

    'Enable the Combo box containing the names of all the
    'available procedures
    Librarylist.Enabled = True


    'save the new procedure to file
    'Open the procedure file
    Proclibmemrq = Len(Recordtype)
    Filenum = FreeFile

    ReDim Proclist(1)
    Open "C:\Proclib.txt" For Random Access Read Write Shared As Filenum Len =
roclibmemrq
    Get Filenum, 1, Proclist(1)


    Proclist(1).Libsize = Proclist(1).Libsize + 1
    ReDim Preserve Proclist(Proclist(1).Libsize)

    Proclist(Proclist(1).Libsize).Blocktagname = Librarylist.List(0)
    Put Filenum, 1, Proclist(1)
    Put Filenum, Proclist(1).Libsize, Proclist(Proclist(1).Libsize)
    Close Filenum
End If


End Sub

Sub load_Click ()
```

```
Dim Filenum As Integer
Dim SinglePCARecord As PCARecordtyp
Dim PCARecord() As PCARecordtyp
Dim recordlength As Single
Dim position As Integer
Dim Firstposition As Integer

On Error GoTo Errorhandler

'Clear original combo box
PCAConfigcombo.Clear

'Define filter for dialog box
OpenConfigfile.Filter = "P.C.A. Config Files(*.cfg)|*.cfg |All Files(*.*)|*.*"

'State that the type of dialog box required is the "Open"
'dialog box
OpenConfigfile.Action = 1

'Define file specs
Filenum = FreeFile

'Define file memory variable storage size
Recordlength = Len(SinglePCARecord)

Position = 1


'Redimension array so that the length of the file can be ascertained
ReDim PCARecord(position)

'Open the file that has been selected from the dialog box
Open OpenConfigfile.Filename For Random Access Read Shared As Filenum Len = rec
ordlength

'Retrieve the variable containing details of how many records
'currently exist so that the array can be modified accordingly
Get Filenum, position, PCARecord(position)

Firstposition = position

'Redefine the length of the array according to file length
ReDim Preserve PCARecord(PCARecord(Firstposition).Recordsize)

'Retrieve all procedures involved in this configuration and
'place in the combo box

For position = Firstposition To PCARecord(Firstposition).Recordsize
    Get Filenum, position, PCARecord(position)
    PCAConfigcombo.AddItem PCARecord(position).Blocktagname
Next

'Close file
Close Filenum
```

```
Errorhandler:
'User pressed cancel button so exit
    Exit Sub

End Sub

Sub loadpic_Click ()
On Error Resume Next
Dim timing As Integer
loadpic.Picture = LoadPicture("C:\visual\bitmaps\toolbar3\OPEN-mds.bmp")
Call mnu_timer(timing)
loadpic.Picture = LoadPicture("C:\visual\bitmaps\toolbar3\open-up.bmp")

load_Click
End Sub

Sub pca_status_Change ()

On Error Resume Next
'If the reset button has been pressed the stop the pca
If PCAStopped = 1 Then
'Must reset the V1SCANSTAT database block to reflect the now
'idle state of the device control algorithm
PCAStopped = 0
Node = pcanode.Text
BlockTyp = "V1SCANSTAT"
Field = "A_CV"
texts = "3"
Justidle = 1
Call SendInfo(Node, BlockTyp, Field, texts)

Exit Sub
End If



If pca_status.Text = "OFFSCAN" Then
    If CurrentProc = (PCAConfigcombo.ListCount) - 1 Then
    GoTo repeat
        Else
        CurrentProc = CurrentProc + 1
        Call Runpca(CurrentProc)

   End If
   Exit Sub
End If

If pca_status.Text = "IDLE" Then
'The PCA has been stopped and we are require to set the
'currentproc variable to "0" to indicate that when we
'restart the system we start again
If Justidle = 1 Then
    Justidle = 0
    Exit Sub
    Else
    CurrentProc = 0
```

```
nd If

xit Sub
epeat:
CurrentProc = 0
all Runpca(CurrentProc)

nd Sub

Sub PCAConfigcombo_Click ()
'When a particular procedure is clicked this event takes
 the last selected procedure from the "Librarylist" combo
'box and adds it to the "PCAConfigcombo" combo box at the
'position after that clicked which caused this event
 procedure in the first place.All the other procedures from
'that point on in the "PCAConfigcombo" combo box


On Error Resume Next




nd Sub

Sub PCAHelp_Click ()
elpform.Show

End Sub

ub Remove_Click ()
Dim Msg
Dim response As Integer
im DgDef As Integer
Dim Title As String * 30
Dim Proclist() As Procedurelibrary
im Recordtype As Procedurelibrary
Dim Proclibmemrq As Variant
Dim Filenum As Integer
im Delposition As Integer
Dim Filealreadyremoved As Integer
Dim I As Integer
im j As Integer

On Error Resume Next

ibrarylist.Enabled = True
Title = "Select Procedure to be Removed"
```

```
Msg = "Have you selected the procedure to be removed from"
Msg = Msg & " the procedure update listbox ? If you haven't"
Msg = Msg & " you may remove the wrong procedure."
DgDef = MB_ICONQUESTION + MB_YESNOCANCEL

response = MsgBox(Msg, DgDef, Title)

If response = IDNo Or response = IDCANCEL Then
       Exit Sub
   Else
   Title = "Confirm Procedure Removal"
   Msg = "Do you really want to remove the selected procedure?"
   DgDef = MB_YESNO + MB_ICONQUESTION
   response = MsgBox(Msg, DgDef, Title)

          If response = IDYES Then

         'save changes to file
         'Open the procedure file
          Proclibmemrq = Len(Recordtype)
          Filenum = FreeFile

          ReDim Proclist(1)
          Open "C:\Proclib.txt" For Random Access Read Write Shared As Filenum
Len = Proclibmemrq
          Get Filenum, 1, Proclist(1)

                If Librarylist.List(Librarylist.ListIndex) = "0" Then
                     Title = "Procedure List Empty"
                     Msg = "There are no more procedures to remove"
                     DgDef = MB_ICONSTOP + MB_OK
                     response = MsgBox(Msg, DgDef, Title)
                    'Else there are procedures available that can
                    'be removed therefore continue
                    Else
                         ReDim Preserve Proclist(Proclist(1).Libsize)
                         For I = 1 To Proclist(1).Libsize
                         Get Filenum, I, Proclist(I)
                         Next
                         For I = 1 To Proclist(1).Libsize
                                If Proclist(I).Blocktagname = Librarylist.List
(Librarylist.ListIndex) And Filealreadyremoved <> 1 Then
                                     Delposition = I
                                     Filealreadyremoved = 1
                                     ' Decrease the number of records by one
                                      For j = Delposition To (Proclist(1).Lib
ize - 1)
                                           Proclist(j).Blocktagname = Procli
t(j + 1).Blocktagname
                                      Next
                                      Proclist(1).Libsize = Proclist(1).Libsi
e - 1
                                End If
                         Next

                    'Save new specs to file
```

```
CACONFG.FRM - 15

                    Put Filenum, 1, Proclist(1)

                    For I = 1 To Proclist(1).Libsize
                        Put Filenum, I, Proclist(I)
                    Next
                    Librarylist.RemoveItem (Librarylist.ListIndex)
                End If
            Close Filenum

            End If

Lnd If
End Sub

ub removepic_Click ()
On Error Resume Next

im timing As Integer

emovepic.Picture = LoadPicture("C:\visual\bitmaps\toolbar3\s_procdn.bmp")
all mnu_timer(timing)
removepic.Picture = LoadPicture("C:\visual\bitmaps\toolbar3\s_procup.bmp")

Now remove the library procedure selected
Remove_Click



nd Sub

Sub Reset_Click ()
esetpic_Click
Lnd Sub

ub resetpic_Click ()

On Error Resume Next
This particular event stops the PCA from operating and resets it back to its
idle state so it can be restarted as required
Dim timing As Integer

esetpic.Picture = LoadPicture("C:\visual\bitmaps\toolbar3\rst_dwn.bmp")
Call mnu_timer(timing)
esetpic.Picture = LoadPicture("C:\visual\bitmaps\toolbar3\rst_up.bmp")

Startpca.Picture = LoadPicture("C:\visual\bitmaps\toolbar3\start_up.bmp")
'Set stopped flag to 1
'CAStopped = 1


nd Sub

Sub save_click ()
)im Filenum As Integer
Dim SinglePCARecord As PCARecordtyp
```

```
Dim PCARecord() As PCARecordtyp
Dim recordlength As Single
Dim position As Integer
Dim Firstposition As Integer
Dim I As Integer
On Error GoTo Errorhandle

'Define filter for dialog box
OpenConfigfile.Filter = "P.C.A. Config Files(*.cfg)|*.cfg |All Files(*.*)|*.*"

'State that the type of dialog box required is the "Open"
'dialog box
OpenConfigfile.Action = 2

'Define file specs
Filenum = FreeFile

'Define file memory variable storage size
recordlength = Len(SinglePCARecord)

'Redimension array to hold all data in the PCAConfigcombo combo box before
'saving to file
ReDim PCARecord(PCAConfigcombo.ListCount)

'Note PCAConfigcombo's size to memory variable before writing that variable
'to file
PCARecord(1).Recordsize = PCAConfigcombo.ListCount

'Transfer PCAConfigcombo combo list data to blocktagname memory variables
'before writing those variables to file

For I = 1 To PCAConfigcombo.ListCount
PCARecord(I).Blocktagname = PCAConfigcombo.List(I - 1)
Next


'Open the file that has been selected from the dialog box
Open OpenConfigfile.Filename For Random Access Write Shared As Filenum Len = re
ordlength


'Now write the variable array to file
For position = 1 To PCAConfigcombo.ListCount
Put Filenum, position, PCARecord(position)
Next

Close Filenum


Errorhandle:
'User pressed cancel button so exit
    Exit Sub

Filealtered = 0
End Sub
```

```
CACONFG.FRM - 17

ub savepic_Click ()
n Error Resume Next
Dim timing As Integer

avepic.Picture = LoadPicture("C:\visual\bitmaps\toolbar3\save-dwn.bmp")
Call mnu_timer(timing)
avepic.Picture = LoadPicture("C:\visual\bitmaps\toolbar3\save-up.bmp")

save_click


End Sub

ub start_Click ()
startpca_Click
End Sub

ub startpca_Click ()
n Error Resume Next
im Title As String * 20
Dim DgDef As Integer
Dim Msg
im response As Integer


Check to see if pca is running

If PCARunning = 1 Then
    MsgBox ("PCA already running,donot attempt to start")
xit Sub
End If

Check to see if there is a configuration loaded to be run
If PCAConfigcombo.ListCount = 0 Then
MsgBox ("You have not loaded a PCA Configuration,therefore the PCA cannot be ru
')
Exit Sub
End If

'check with user to make sure he wants the pca to run
Title = "Run Check"
Msg = "Do you want to run the PCA?"
DgDef = MB_ICONQUESTION + MB_YESNO

esponse = MsgBox(Msg, DgDef, Title)

If response = IDNo Then
     Exit Sub
nd If
'Set Running flag
PCAStopped = 0

'Load picture that indicates that pca is running
Startpca.Picture = LoadPicture("C:\visual\bitmaps\toolbar3\start_dn.bmp")
```

```
'Therefore the user wants to run the PCA so therefore run


CurrentProc = 0
'Put first procedure in the PCAConfigcombo combo box on
'scan
Call Runpca(CurrentProc)


End Sub

Sub Text3_Change ()

On Error Resume Next
Select Case CInt(Text3.Text)

Case Is = 0
pca_status.Text = "OFFSCAN"

Case Is = 1
pca_status.Text = "ONSCAN"

Case Is = 2
pca_status.Text = "REPEAT"

Case Is = 3
pca_status.Text = "IDLE"
End Select

End Sub

Sub viewpic_Click ()
On Error Resume Next
Dim timing As Integer

viewpic.Picture = LoadPicture("C:\visual\bitmaps\toolbar3\diagv_dn.bmp")
Call mnu_timer(timing)
viewpic.Picture = LoadPicture("C:\visual\bitmaps\toolbar3\diagv_up.bmp")

End Sub
```

```
Sub Exit_Help_Click ()
Unload Helpform
End Sub

Sub Form_Load ()
Combo1.AddItem "Configuring P.C.A."

End Sub

Sub Help_OK_Click ()
Dim Msg As String
Dim Title As String * 20
Static ThisHelpFile(10) As Helpfiletype
Dim Helpfilelength As Single
Dim HlpFleTypRec As Helpfiletype
Dim recordnumber As Integer


'Check to see if help subject has been selected
If Combo1.ListIndex = "" Then
    Msg = "You didn't select an item of query"
    Title = "Reselect Help Item"

    Else
    'a topic was selected therefore load help information
    Helpexp.Show
    Helpexp.Caption = Combo1.List(Combo1.ListIndex)

    'Now open the help file that contains the records for each of the different
    'help options and then write the text associated with the help option
    'as selected from the help combo box

    Filenum = FreeFile
    Recordlength = Len(HlpFleTypRec)

    Open "c:\sit\Help\PCA.hlp" For Random Access Read Shared As Filenum Len = R
ecordlength

    For recordnumber = 1 To 10
    Get Filenum, recordnumber, ThisHelpFile(recordnumber)

        If Combo1.List(Combo1.ListIndex) = ThisHelpFile(recordnumber).Helprecor
dname Then
        'we have found the record we want to write to "HelpExp"'s textbox
        'therefore write it there

        Helpexp.Helptext.text = ThisHelpFile(recordnumber).Helptext
        Exit Sub
        End If

    Next
```

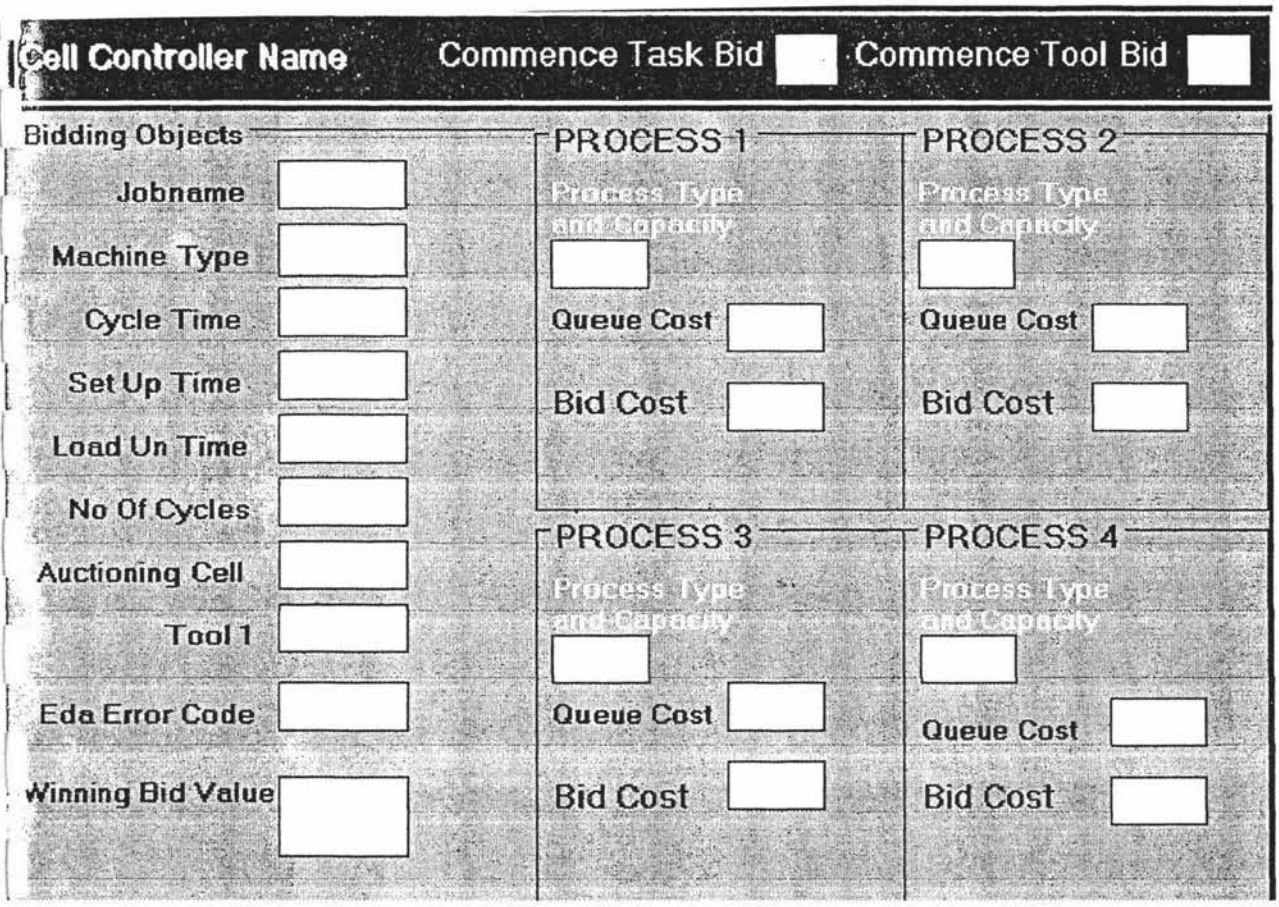
```
End If

End Sub
```

A.5 Implementation of the Bid Formulation Application

The Bid Formulation Application is responsible for the ordering (and execution of) the FIXDMACS database blocks which emulate the Process Behaviour Algorithm. As these blocks have a link to the real world manufacturing processes they directly control the series of steps the process goes through to complete a task.

The following material is a print out from Microsoft Visual Basic showing all programming carried out for the applications implementation.


Cell Controller Name
Commence Task Bid
Commence Tool Bid
Bidding Objects
Jobname
Machine Type
Cycle Time
Set Up Time
Load Un Time
No Of Cycles
Auctioning Cell
Tool 1
Eda Error Code
Winning Bid Value
PROCESS 1
Process Type and Capacity
Queue Cost
Bid Cost
PROCESS 2
Process Type and Capacity
Queue Cost
Bid Cost
PROCESS 3
Process Type and Capacity
Queue Cost
Bid Cost
PROCESS 4
Process Type and Capacity
Queue Cost
Bid Cost

```
Option Explicit

'Set all arrays to start at position 1
Option Base 1

'Nullify Case sensitivity in "like" Functions
Option Compare Text

'queuecost variable
Dim Thisqueuecost(1) As VMDQueueCost

'VMD ingoing queue maximum buffer size variable
Dim Thisbuffer(1) As InQueueBuffersize

Dim FullInQFlag As Integer
Dim FullOutQFlag As Integer
'General Declarations
Dim X As Variant
Dim Text As Variant
Dim NodeNam As String * 10
Dim Msg As String * 40
Dim title As String * 35
Dim UpdateInQueueCost As Double

'EDA Function Declarations

' functions specific to VB (not standard eda fcns)
Declare Sub eda_version Lib "vdba.dll" (ByVal V$)
Declare Sub eda_get1_ascii Lib "vdba.dll" (ByVal n$, ByVal t$, ByVal f$, ByVal
$)
Declare Function eda_set1_ascii% Lib "vdba.dll" (ByVal n$, ByVal t$, ByVal f$, B
yVal V$)
'
' Note define_group returns a long
'
Declare Function eda_define_group& Lib "vdba.dll" (ByVal Count%, ByVal Detect%)
Declare Sub eda_delete_group Lib "vdba.dll" (ByVal Handle&)
'
' should always pass 0 for vsp
Declare Function eda_define_ntf& Lib "vdba.dll" (ByVal Handle&, ByVal enode$, By
Val etag$, ByVal efield$, ByVal vsp&)
Declare Sub eda_delete_ntf Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
'
Declare Sub eda_lookup Lib "vdba.dll" (ByVal Handle&)
Declare Sub eda_read Lib "vdba.dll" (ByVal Handle&)
Declare Sub eda_write Lib "vdba.dll" (ByVal Handle&)
Declare Function eda_write1% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
Declare Sub eda_wait Lib "vdba.dll" (ByVal Handle&)
Declare Function eda_query_wait% Lib "vdba.dll" (ByVal Handle&)
'
Declare Function eda_get_error% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
Declare Function eda_get_ascii% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, ByVa
 value$, ByVal maxlen%)
Declare Function eda_set_ascii% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, ByVa
 value$)
```

```
ORMBID.FRM - 2

' note pointer to float is passed
Declare Function eda_get_float% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, valu
)
Declare Function eda_set_float% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, valu
)

'eda communication variables
Dim fart As Variant
Dim BidOpstate As Variant
Dim Dummy As Variant
Dim Ghandler As Long
Dim eda_err As Variant
Dim tex As Integer
Dim Newnum As Variant
Dim Node As String * 15
Dim Tags As String * 15
Dim Field As String * 15
Dim answer As String * 80
Dim ans As String * 10
Dim defnode As Long
Dim trap As String * 10
Dim Texts As String * 6
Dim Nodetyp As String * 20
Dim nodes As String * 20
Dim bidrotenable As Variant
Dim BlockTyp As String * 20
Dim reply As Variant
Dim Bidoperatestate As Variant
Dim Bidnodetext As String * 10
Dim bidtokenstate As Variant
Dim Tokenstate As Integer

Dim Thisbidding(1) As Bid_data_To_Be_sent
Dim Thisbid(1) As Nodesbid


'Declare form level string variables to store the specification of the processes
'configured for
'this cell
Dim VMD1machSpec As Single

'Define an array the purpose of which is to store the current data regarding whe
'ther a VMD is
'active or not.
Dim MachineSpecArray(4) As Single

'Define an array that stores the reserved status of the tools that are being res
'erved for the task
'being carried out.
Dim Toolstatus(4) As String * 10

'Define an array that stores the required data about the tools currently require
'd for the task in
'the queue.

Dim TaskToolTyp(4) As Single
```

```
'define montharray array
Dim Montharray(12) As Integer

Sub CommDDELink (Text As TextBox, LinkItem)
'This Sub Procedure creates an Automatic link between the
'Queueing application and either the Factory Controller or
'the Cell Controller

Text.LinkMode = NONE  'If the link is not already
                      'created then create it
Text.LinkTopic = "DMDDE|DATA"  'Link topic is FixDmacs

Text.LinkItem = LinkItem      'This is the Node,
                              'databaseblock and field we
                              'want to link to

Text.LinkMode = LINK_AUTOMATIC 'Automatic update of data


End Sub

Function eda_requestinfolink (n, t, f)
Ghandler = eda_define_group(1, 0)
defnode = eda_define_ntf(Ghandler, n, t, f, 0)

Call eda_lookup(Ghandler)
Call eda_wait(Ghandler)
Call eda_read(Ghandler)
Call eda_wait(Ghandler)
Newnum = eda_get_ascii(Ghandler, defnode, answer, 50)
Call eda_delete_group(Ghandler)
eda_requestinfolink = answer
End Function

Sub RunDDEServer ()
AppActivate "Microsoft Visual Basic" 'Subject is VB
X = Shell("dmdde.exe", 1) 'Makes sure The DDE  server is going
Dummy = DoEvents()

End Sub

Sub sendinfo (n, t, f, Text)
Ghandler = eda_define_group(1, 0)
defnode = eda_define_ntf(Ghandler, n, t, f, 0)

Call eda_lookup(Ghandler)
Call eda_wait(Ghandler)
eda_err = eda_get_error(Ghandler, defnode)
Text5.Text = Str$(eda_err)
'MsgBox (Text5.Text)
Newnum = Text
fart = eda_set_ascii(Ghandler, defnode, Newnum)
Call eda_write(Ghandler)
Call eda_wait(Ghandler)
Call eda_delete_group(Ghandler)
```

```
End Sub

Sub Bid_Formulation_Status_Change ()

'declare all the required variables
Dim Thisbidfile As Bid_data_To_Be_sent
Dim Filenum As Integer
Dim recordlength As Integer
Dim Proc1_OK As Integer
Dim Proc2_OK As Integer
Dim Proc3_OK As Integer
Dim Proc4_OK As Integer
Dim VMD1Bid As Single
Dim VMD2Bid As Single
Dim VMD3Bid As Single
Dim VMD4Bid As Single
Dim Taskcost As Single
Dim SystemBusy As Integer
'queuecost variable declarations

Dim QueueCosts As VMDQueueCost
Dim Matchposition As Integer
Dim i As Integer
Dim directoryname As String
Dim Filename As String
Dim Fileconcerned As String


On Error Resume Next
On Error GoTo Checkerror


If Val(Bid_Formulation_Status.Text) = 0 Then
    Exit Sub
End If

'when this changes to a "1" then the bidding process must begin

If Val(Bid_Formulation_Status.Text) = 1 Then
    'when this textbox turns to a "1" we are required to begin the bid formulation process

    'The first thing we are required to do is to open the bid file that contains the bidding objects.

    Filenum = FreeFile
    recordlength = Len(Thisbidfile)

    'open the file in shared mode so that other applications can read from it
    'strictly speaking we should lock write to it but nothing else will be
    'writing to it so dont worry
    Fileconcerned = "Bidfile"
    'Open Netdrive & "Factory\Bidding\Bidfile.bdf" For Random Access Read Shared As Filenum Len = recordlength
```

```
    Open Netdrive & "Cellcont\Bidding\BidF" & Trim(Text1.Text) & ".bdf" For Ra
om Access Read Shared As Filenum Len = recordlength
'     MsgBox (Netdrive & "Cellcont\Bidding\BidF" & Trim(Text1.Text) & ".bdf")
    Get Filenum, 1, Thisbidding(1)

    'Type Bid_data_To_Be_sent
    'Jobname As String * 10
    'Machinetype As Single
    'EstCycletime As Integer
    'Setuptime As Integer
    'LoadUnloadtime As Integer
    'NoofCycles As Integer
    'Tool1required As Single
    'Activate As Integer
    'Force As Integer
    'AuctioningCell As String * 8
    'End Type

    Jobname.Text = Thisbidding(1).Jobname
    Machtype.Text = Thisbidding(1).Machinetype
'     MsgBox (Machtype.Text)
    EstCyctime.Text = Thisbidding(1).EstCycletime
 '    MsgBox (EstCyctime)
    Setuptime.Text = Thisbidding(1).Setuptime
'      MsgBox (Setuptime.Text)
    LoadUnloadtime.Text = Thisbidding(1).LoadUnloadtime
    No_of_Cycles.Text = Thisbidding(1).NoofCycles
    Tool1.Text = Thisbidding(1).Tool1required
'     MsgBox (Tool1.Text)
    'Tool2.Text = Thisbidding(1).Tool2required
 '    MsgBox (Tool2.Text)
    'tool3.Text = Thisbidding(1).Tool3required
 '    MsgBox (tool3.Text)
    'tool4.Text = Thisbidding(1).Tool4required
  '    MsgBox (tool4.Text)
    Auct_Cell.Text = Trim(Thisbidding(1).AuctioningCell)
    'Calculate the cost of doing the actual task being considered
   'MsgBox (Auct_Cell.Text)
    'Jobname.Text = Thisbidding(1).Jobname
    'Machtype.Text = Thisbidding(1).Machinetype
    'EstCyctime.Text = Thisbidding(1).EstCycletime
    'Setuptime.Text = Thisbidding(1).Setuptime
    'LoadUnloadtime.Text = Thisbidding(1).LoadUnloadtime
    'No_of_Cycles.Text = Thisbidding(1).NoofCycles
    'Tool1.Text = Thisbidding(1).Tool1Required
    'Tool2.Text = Thisbidding(1).Tool2required
    'Tool3.Text = Thisbidding(1).Tool3Required
    'Tool4.Text = Thisbidding(1).Tool4Required
    Taskcost = ((Thisbidding(1).NoofCycles) * ((2 * (Thisbidding(1).LoadUnloadti
me) + (Thisbidding(1).EstCycletime)))) + (Thisbidding(1).Setuptime)
    'MsgBox ("Taskcost is equal to " & CStr(Taskcost))


    Close Filenum
```

```
    'Type VMDQueueCost
        'Cellname As String * 8
        'VMD1_Queue_Cost As Single
        'VMD2_Queue_Cost As Single
        'VMD3_Queue_Cost As Single
        'VMD4_Queue_Cost As Single
    'End Type



    'now open up the file that contains the incoming queue cost for each VMD
    Filenum = FreeFile 'denote a freefile number
    recordlength = Len(QueueCosts) 'get array storage size required

    directoryname = "c:\Cellcont\Queue\" 'set directory name

    Fileconcerned = "Queuecost"
    Filename = "Queucost.cst" 'set filename

etryHere:
    'open the file in lock write mode so that no body else can write to it whils
+ i am reading
    Open Trim(directoryname) & Trim(Filename) For Random Access Read Shared As
ilenum Len = recordlength

    Get Filenum, 1, Thisqueuecost(1) 'get the array into memory


    'now put each queue cost in the relevant textbox.
    Proc1qcost.Text = Thisqueuecost(1).VMD1_Queue_Cost 'put each bit of data in
the correct box
   ' MsgBox (Proc1qcost.Text)
    Proc2qcost.Text = Thisqueuecost(1).VMD2_Queue_Cost
   ' MsgBox (Proc2qcost.Text)
    Proc3qcost.Text = Thisqueuecost(1).VMD3_Queue_Cost
  '  MsgBox (Proc3qcost.Text)
    Proc4qcost.Text = Thisqueuecost(1).VMD4_Queue_Cost
   ' MsgBox (Proc4qcost.Text)

    Close Filenum 'close the file


    '*****FIRST CHECK TO SEE WHAT THE SIZE OF THIS NODES MAXIMUM BUFFER SIZE IS
FOR EACH VMD
    '*****TO SEE WHAT NODES ARE ELIGIBLE TO HOLD AN AUCTION**********************
*************


    'Declare variables for this procedure
    Dim Fileslength As Integer
    Dim Currentbuffer As InQueueBuffersize
    Dim VMD1BufferOk As Integer
    Dim VMD2BufferOk As Integer
    Dim VMD3BufferOk As Integer
    Dim VMD4BufferOk As Integer
```

```
    'Dim Thisbuffer(1) As Inqueuebuffersize

    'now we are required to get the data from the file particular to this node
    Filenum = FreeFile 'define a freefile
    Fileslength = Len(Currentbuffer) 'define storage variable size in Bytes
    Filename = Netdrive & "Cellcont\queue\" & Trim(Text1.Text) & "Bfsze.Buf" 'd
fine filename
    Fileconcerned = "Filename"
   ' MsgBox (Filename)
    Open Filename For Random Access Read Shared As Filenum Len = Fileslength 'o
en the file for write access

    Get Filenum, 1, Thisbuffer(1) 'get the data
 '   MsgBox CStr(Thisbuffer(1).VMD1Buffer)
    'MsgBox CStr(Thisbuffer(1).VMD4Buffer)


    If Thisbuffer(1).VMD1Buffer >= (Val(Proc1qcost.Text) + Taskcost) Then 'the b
uffer for VMD1 is not full
        VMD1BufferOk = 1    'so it can accept another job if it wins
    End If

    If Thisbuffer(1).VMD2Buffer >= (Val(Proc2qcost.Text) + Taskcost) Then  'the
buffer for VMD2 is not full
        VMD2BufferOk = 1    'so it can accept another job if it wins
    End If

    If Thisbuffer(1).VMD3Buffer >= (Val(Proc3qcost.Text) + Taskcost) Then 'the b
uffer for VMD3 is not full
        VMD3BufferOk = 1    'so it can accept another job if it wins
    End If

    If Thisbuffer(1).VMD4Buffer >= (Val(Proc4qcost.Text) + Taskcost) Then  'the
buffer for VMD3 is not full
        VMD4BufferOk = 1      'so it can accept another job if it wins
    End If

    Close Filenum
    'Dim Thisbuffer(1) As InQueueBuffersize
    'Type InQueueBuffersize
    '   VMD1Buffer As Integer
    '  VMD2Buffer As Integer
     ' VMD3Buffer As Integer
    ' VMD4Buffer As Integer
    'End Type
    'MsgBox (Trim(Thisnode))
    'MsgBox (CStr(TheVMD))



    'now we have loaded the pertinent bidding information we can begin the bidd
ng operation

    '****FIRST CHECK TO SEE WHAT PROCESSES ARE ACTIVE**********
    If Val(Process1_typ_Cap.Text) = 0 Then
    'the VMD is not active therefore indicate this
```

```
        Proc1_OK = False
    End If

    If Val(Process2_typ_Cap.Text) = 0 Then
    'the VMD is not active therefore indicate this
        Proc2_OK = False
    End If

    If Val(Process3_typ_Cap.Text) = 0 Then
    'the VMD is not active therefore indicate this
        Proc3_OK = False
    End If

    If Val(Process4_typ_Cap.Text) = 0 Then
    'the VMD is not active therefore indicate this
        Proc4_OK = False
    End If

    '


    '*******SECOND CHECK TO SEE IF EACH PROCESS IN EACH VMD IS OF THE CORRECT TY
PE*****
    '*******AND THAT IT IS OF BIG ENOUGH CAPACIY TO DO THE JOB*****************
*******


    If Int(Val(Process1_typ_Cap.Text)) - Int(Val(Machtype.Text)) = 0 Then
        'check to see if it is big enough
        If (Val(Process1_typ_Cap.Text) - Int(Val(Process1_typ_Cap.Text))) >= (Va
l(Machtype.Text) - Int(Val(Machtype.Text))) Then
            'Process 1 is of the correct type
            Proc1_OK = True
             MsgBox ("Process 1 is the right machine & big enough")
            Else
                Proc1_OK = False
        End If
    Else
        Proc1_OK = False
    End If

    If Int(Val(Process2_typ_Cap.Text)) - Int(Val(Machtype.Text)) = 0 Then
        'check to see if it is big enough
        If (Val(Process2_typ_Cap.Text) - Int(Val(Process2_typ_Cap.Text))) >= (Va
l(Machtype.Text) - Int(Val(Machtype.Text))) Then
            'Process 1 is of the correct type
            Proc2_OK = True
'            MsgBox ("Process 2 is the right machine & big enough")
            Else
                Proc2_OK = False
        End If
    Else
        Proc2_OK = False
    End If

    If Int(Val(Process3_typ_Cap.Text)) - Int(Val(Machtype.Text)) = 0 Then
```

```
        'check to see if it is big enough
        If (Val(Process3_typ_Cap.Text) - Int(Val(Process3_typ_Cap.Text))) >= (V
l(Machtype.Text) - Int(Val(Machtype.Text))) Then
            'Process 1 is of the correct type
            Proc3_OK = True
  '          MsgBox ("Process 3 is the right machine & big enough")
            Else
                Proc3_OK = False
        End If
    Else
        Proc3_OK = False
    End If


    If Int(Val(Process4_typ_Cap.Text)) - Int(Val(Machtype.Text)) = 0 Then
        'check to see if it is big enough
        If (Val(Process4_typ_Cap.Text) - Int(Val(Process4_typ_Cap.Text))) >= (V
l(Machtype.Text) - Int(Val(Machtype.Text))) Then
            'Process 1 is of the correct type
            Proc4_OK = True
   '         MsgBox ("Process 4 is the right machine & big enough")
            Else
                Proc4_OK = False
        End If
    Else
        Proc4_OK = False
    End If

    '**CHECK TO SEE IF THE TOOLS AVAILABLE TO EACH PROCESS ARE THE CORRECT TYPE
FOR THE JOB****

    Dim Thistoolspec() As Toolspec
    Dim Thistool As Toolspec
    'Dim BidToolSpec() As ToolSpec
    'Dim RqFileLength As ToolSpec
    Dim Filelength As Single
    Dim Tool1available As Integer
    'Dim Tool2available As Integer
    'Dim Tool3available As Integer
    'Dim Tool4available As Integer
    Dim X As Integer
    Dim Count As Integer
    Dim Thisbit As String * 6
    Dim Thistext As String * 8
    Dim ThisFile As String * 70
    Dim Thedirectory As String
    Dim Thisfileslength As Integer
    Dim Thisstool As Single
    Dim j As Integer
    Dim Cellname As String
    Dim Cellnumber As String
    Dim Proc1toolavle As Integer
    Dim Proc2toolavle As Integer
    Dim Proc3toolavle As Integer
    Dim Proc4toolavle As Integer
```

```
'ORMBID.FRM - 10

    'Type ToolSpec
    'Toolname As String * 40
    'ToolTypeSize As Single
    'NoOfToolsNeeded As Integer
    'End Type


    '*******************FIRST CHECK FOR PROCESS 1 TO SEE IF IT HAS THE CORRECT T
OOLS AVAILABLE****

tryitagain:
    For j = 1 To 4

    Cellname = CStr(Text1.Text)

    If Trim(Cellname) Like "C0#" Then
        Cellnumber = Mid$(Trim(Cellname), 3, 3)
        Else Cellnumber = Mid$(Trim(Cellname), 2, 3)
    End If

    Filename = "V" & CStr(j) & "C" & Trim(Cellnumber) & "tSpc.txt"
    Fileconcerned = "Toolspec"
     MsgBox (Filename)
    Thedirectory = Netdrive & "Factory\Config\Cells\"      '"C:\CellCont\Specs\T
ools\"
'
    'GoTo tryitagain
'MsgBox (Thedirectory)
    Filenum = FreeFile
    recordlength = Len(Thistool)

    ThisFile = Trim(Thedirectory) & Trim(Filename)

'    MsgBox (ThisFile)

    'Open the file for read\write random access
    Open Trim(ThisFile) For Random Access Read Shared As Filenum Len = recordlen
gth

    'redimension the array to take the first record
    ReDim Thistoolspec(1)

    'get the length of the file
    Get Filenum, 1, Thistoolspec(1)

    'this is the current length of the file however we are required to make enou
gh room to store all
    'the check box information that has been checked.
    Thisfileslength = Thistoolspec(1).NoOfToolsNeeded
    'MsgBox (CStr("The filelength is " & Thisfileslength))

    'Type ToolSpec
    'Toolname As String * 40
    'ToolTypeSize As Single
    'NoOfToolsNeeded As Integer
    'End Type
```

```
    ReDim Preserve Thistoolspec(Thisfileslength)

    For i = 1 To Thisfileslength
        Get Filenum, i, Thistoolspec(i)
'        MsgBox (CStr(Thistoolspec(i).ToolTypesize))
    Next

    For i = 1 To Thistoolspec(1).NoOfToolsNeeded


        If CSng(Tool1.Text) = Thistoolspec(i).ToolTypesize Then
        Tool1available = 1  'Set tool1available flag
        End If
    Next

    'For i = 1 To Thistoolspec(1).NoOfToolsNeeded
     '   If CSng(Tool2.Text) = Thistoolspec(i).ToolTypesize Then
      '      Tool2available = 1  'Set tool2available flag
       ' End If
    'Next

    'For i = 1 To Thistoolspec(1).NoOfToolsNeeded
     '   If CSng(tool3.Text) = Thistoolspec(i).ToolTypesize Then
      '      Tool3available = 1  'Set tool3available flag
       ' End If
    'Next

    'For i = 1 To Thistoolspec(1).NoOfToolsNeeded
     '   If CSng(tool4.Text) = Thistoolspec(i).ToolTypesize Then
      '      Tool4available = 1 'Set tool4available flag
       ' End If
   ' Next


    If Tool1available = 1 Then 'And Tool2available = 1 And Tool3available = 1 Ar
d Tool4available = 1 Then
        Select Case j

        Case Is = 1#
        Proc1toolavle = 1
      '  MsgBox ("process 1's toolavailabity is " & CStr(Proc1toolavle))

        Case Is = 2#
        Proc2toolavle = 1
       ' MsgBox ("process 2's toolavailabity is " & CStr(Proc2toolavle))

        Case Is = 3#
        Proc3toolavle = 1
'        MsgBox ("process 3's toolavailabity is " & CStr(Proc3toolavle))

        Case Is = 4#
        Proc4toolavle = 1
     '   MsgBox ("process 4's toolavailabity is " & CStr(Proc4toolavle))

        End Select
```

```
            Else
    '        MsgBox ("Tools not available for Process " & CStr(j))

    End If


    Close Filenum
    'reset flags
    Tool1available = 0
    'Tool2available = 0
    'Tool3available = 0
    'Tool4available = 0



    Next

If Proc1_OK And Proc1toolavle = 1 And VMD1BufferOk = 1 Then 'we can submit a bic
 for this VMD
    VMD1Bid = Taskcost + Val(Proc1qcost.Text)
    Else
    If Proc1_OK And Proc1toolavle = 1 And VMD1BufferOk = 0 Then 'if process avai
 able but buffer
        SystemBusy = 1                                          'full
    End If
nd If

If Proc2_OK And Proc2toolavle = 1 And VMD2BufferOk = 1 Then 'we can submit a bic
 for this VMD
    VMD2Bid = Taskcost + Val(Proc2qcost.Text)
    Else
    If Proc2_OK And Proc2toolavle = 1 And VMD2BufferOk = 0 Then 'if process avai
able but buffer
        SystemBusy = 1                                          'full
    End If
nd If

f Proc3_OK And Proc3toolavle = 1 And VMD3BufferOk = 1 Then 'we can submit a bid
 for this VMD
    VMD3Bid = Taskcost + Val(Proc3qcost.Text)
    Else
    If Proc3_OK And Proc3toolavle = 1 And VMD3BufferOk = 0 Then 'if process avai
able but buffer
        SystemBusy = 1                                          'full
    End If
End If

f Proc4_OK And Proc4toolavle = 1 And VMD4BufferOk = 1 Then 'we can submit a bid
 for this VMD
    VMD4Bid = Taskcost + Val(Proc4qcost.Text)
    Else
    If Proc4_OK And Proc4toolavle = 1 And VMD4BufferOk = 0 Then 'if process avai
able but buffer
        SystemBusy = 1                                          'full
    End If
```

```
End If


Proc1_Bidcost.Text = VMD1Bid
Proc2_Bidcost.Text = VMD2Bid
Proc3_bidcost.Text = VMD3Bid
Proc4_Bidcost.Text = VMD4Bid


'********FIND THE PART TRAVEL TIME FROM THE AUCTIONING CELL TO THIS CELL*******
********
'at this point we are required to find the part travel time portion of the bid w
hich is the
'time required for the part being auctioned to travel from its current location
to this cell
'given that this cell actually wins the auction this data is found in the netdri
ve &
'"Factory\Bidding\Nodetime\NodeDely.txt" file

Dim Winningbidsize As Single
Dim WinningVMDno As Integer
Dim Thisnodename As String * 8
'Type NodeTimedelay
 '   MatrixDim As Integer
  '  Interarrivaltime As Integer
   ' CellnameX As String * 10
   ' CellnameY As String * 10
'End Type

Dim Thesenodestimedif() As NodeTimedelay
Dim TheseNodesRecord As NodeTimedelay
Dim recordsize As Integer
Dim s As Integer
Dim Nextrecord As Integer
Dim Bidnodename As String * 8
Dim Auctnodename As String * 8
Dim Travelcost As Single

'first get the name of this node
Bidnodename = Trim(Text1.Text)'define the name of this node that the bid formula
tion app is on
Auctnodename = Trim(Auct_Cell.Text)'get the name of the node from file that is h
olding the auction

If Trim(Bidnodename) Like Trim(Auctnodename) Then 'the cell that the node origin
ated from and
    'the cell from which the bid is being formulated for is the same therefore t
he travel time is
    'zero.
    Travelcost = 0
   ' MsgBox ("Travelcost is equal to " & CStr(Travelcost))
    GoTo Nextpart
End If

Filenum = FreeFile
```

```
recordlength = Len(TheseNodesRecord)

Fileconcerned = "Nodedelay"

Open Netdrive & "Cellcont\Bidding\Nodetime\N" & Trim(Text1.Text) & "dely.txt" F
 Random Access Read Shared As Filenum Len = recordlength

'MsgBox (Netdrive & "Cellcont\Bidding\Nodetime\N" & Trim(Text1.Text) & "dely.tx
')
'firstly we only require the first record to get the matrixsize
ReDim Thesenodestimedif(1)

'get the first record that stores the value of the matrix size
Get Filenum, 1, Thesenodestimedif(1)

'redimension to take all the records that are in the file
recordsize = (Thesenodestimedif(1).MatrixDim * Thesenodestimedif(1).MatrixDim)

ReDim Thesenodestimedif(recordsize) 'Redimensioning the array

now get all the records
or i = 1 To recordsize
    Get Filenum, i, Thesenodestimedif(i)
    'MsgBox (Node_name.Text)
    'MsgBox (Thesenodestimedif(i).CellnameX)
    'Node_name.Text = "CO1"
    'Thesenodestimedif(i).CellnameX = "CO1"
    If Trim(Bidnodename) Like Trim(Thesenodestimedif(i).CellnameX) And Trim(Auct
odename) Like Trim(Thesenodestimedif(i).CellnameY) Then
        Travelcost = CSng(Thesenodestimedif(i).InterarrivalTime)
    '    MsgBox ("There is a travel cost and it is equal to " & CStr(Travelcost)
)
        GoTo Nextpart
    End If
Next

extpart:

lose Filenum

ow we must find the winning bid for this cell
f ((VMD1Bid <= VMD2Bid Or VMD2Bid = 0) And (VMD1Bid <= VMD3Bid Or VMD3Bid = 0)
d (VMD1Bid <= VMD4Bid Or VMD4Bid = 0)) And VMD1Bid <> 0 Then
    'vmd1 has the best bid
    WinningVMDno = 1
    Winningbidsize = VMD1Bid + Travelcost
    Bidwinning.Text = Winningbidsize
    GoTo Bidcomplete
d If

f ((VMD2Bid <= VMD1Bid Or VMD1Bid = 0) And (VMD2Bid <= VMD3Bid Or VMD3Bid = 0)
d (VMD2Bid <= VMD4Bid Or VMD4Bid = 0)) And VMD2Bid <> 0 Then
    'vmd2 has the best bid
    WinningVMDno = 2
    Winningbidsize = VMD2Bid + Travelcost
    Bidwinning.Text = Winningbidsize
```

```
    GoTo Bidcomplete
End If

If ((VMD3Bid < VMD1Bid Or VMD1Bid = 0) And (VMD3Bid < VMD2Bid Or VMD2Bid = 0) A
d (VMD3Bid < VMD4Bid Or VMD4Bid = 0)) And VMD3Bid <> 0 Then
    'vmd3 has the best bid
    WinningVMDno = 3
    Winningbidsize = VMD3Bid + Travelcost
    Bidwinning.Text = Winningbidsize
    GoTo Bidcomplete
End If

If ((VMD4Bid < VMD1Bid Or VMD1Bid = 0) And (VMD4Bid < VMD2Bid Or VMD2Bid = 0) A
d (VMD4Bid < VMD3Bid Or VMD3Bid = 0)) And VMD4Bid <> 0 Then
    'vmd4 has the best bid
    'vmd3 has the best bid
    WinningVMDno = 4
    Winningbidsize = VMD4Bid + Travelcost
    Bidwinning.Text = Winningbidsize
    GoTo Bidcomplete
End If


Bidcomplete:

'MsgBox ("Winning VMD Number is " & CStr(WinningVMDNo))
'MsgBox ("Winning Bid size equals " & CStr(WinningBidSize))

'This is all very well but we are also required to put this information in a fi
e and put it on
'the network so that the factory Controller is able to read all the bids - this
file however
'must be particular to a Cell so that the factory controller knows what cell the
 bid is coming
'from
Dim Cellnam As String * 5



'get the name of the Cell and put it into a string
Cellnam = Text1.Text

'define the files location
directoryname = Netdrive & "Factory\Bidding\Bids\"

'define complete file description
Filename = Trim(directoryname) & Trim(Cellname) & "Bid.Bid"

'get a freefile number
Filenum = FreeFile

Fileconcerned = Trim(Filename)
Open Trim(Filename) For Output Shared As Filenum Len = 10

'MsgBox ("Winning vmd number is " & WinningVMDno)
'now we need to input the information into the file.
```

```
Write #Filenum, WinningVMDno
Write #Filenum, Winningbidsize
Write #Filenum, SystemBusy

Close Filenum

'we are also required to tell the factory controller that this cell has formula
ed it's bid and
'that the Factory Controller can compare all the bids lodged when all other fla
s have been
'recieved from all the other nodes stating that the other nodes have also calcu
ated their bids.
'This is required for two reasons - the first is to make sure that all bids are
recieved by
'the factory controller so that an optimal bid is selected and also to prevent f
ile sharing
'problems by the Factory Controller trying to read a bidfile as it is still bei
 written

Node = "Fact"
Tags = Trim(Cellname) & "BIDFIN"
Field = "A_CV"
Texts = "1"

Call sendinfo(Node, Tags, Field, Texts)


End If


'this is fine but we must also reset the bidfrmstrt block so that again the syst
em can respond
'to a start bid formation command.

Node = CStr(Text1.Text)
Tags = "BIDFRMSTRT"
Field = "A_CV"
Texts = "0"

Call sendinfo(Node, Tags, Field, Texts)


Exit Sub

Checkerror:
If Err = 9 Then
Resume Next
    Else
    If Err = 70 Then
    MsgBox ("This is the error")
    GoTo RetryHere:
        Else
        MsgBox ("The error is " & Err & "is at the " & Trim(Fileconcerned) & " F
ile")
        Resume
```

```
    End If
End If

End Sub

Sub Form_Load ()
'This application is the bid formation application for SITAMA
'Written by Matthew John Jolly
'PhD Student
'Department of Production Technology
'Massey University
'Palmerston North

'This application is responsible for calculating the task bids for each active \
MD within the Cell
'Controller that it is housed within. It is also responsible for the formulatio
 of bids for the
'tools available in the factory enviornment.


'firstly we have to get the name of the Cell that this is working on

Dim Filenum As Integer
Dim Thisstring As String * 20

On Error GoTo Checkit:



Filenum = FreeFile

Open "c:\Cellcont\Cellname.txt" For Input Shared As Filenum Len = 10

Do Until EOF(Filenum)
    Line Input #Filenum, Thisstring

Loop
'MsgBox (Thisstring)
Text1.Text = Thisstring
Close Filenum

'run the server for Dynamic Data Exchange
Call RunDDEServer


'create the link between the "BidFrmstsrt" database block and the Bid_Formulatio
n_Status text
'box so that when this box changes to a "1" the task bid formulation process can
 begin!!

Call CommDDELink(Bid_Formulation_Status, Trim(Thisstring) & "." & "BIDFRMSTRT" &
 "." & "F_CV")

 now we are required to form a link between the Tool_Bid_Start textbox and the "
Bidtoolstrt"
 database block so that the cell controller is able to respond to a request from
```

```
 the factory
'controller to start formulating a bid for the tool being auctioned.
Call CommDDELink(Tool_Bid_Start, Trim(Thisstring) & "." & "BDTOOLSTRT" & "." &
F_CV")


'Now we are required to put together a permanent link between the textbox and th
'machine type database block in case a VMD becomes disabled

 Retrieve activity and type status for VMD1
Call CommDDELink(Process1_typ_Cap, Trim(Thisstring) & "." & "V1MACHTYP" & "." &
"F_CV")

 Retrieve activity and type status for VMD2
Call CommDDELink(Process2_typ_Cap, Trim(Thisstring) & "." & "V2MACHTYP" & "." &
"F_CV")

'Retrieve activity and type status for VMD3
Call CommDDELink(Process3_typ_Cap, Trim(Thisstring) & "." & "V3MACHTYP" & "." &
F_CV")

'Retrieve activity and type status for VMD4
all CommDDELink(Process4_typ_Cap, Trim(Thisstring) & "." & "V4MACHTYP" & "." &
F_CV")


reset the Bid_Formulation_Status textbox to something uneventfull
Bid_Formulation_Status.Text = 2

reset the Tool_Bid_Start textbox to something uneventfull
Tool_Bid_Start.Text = 2

xit Sub

heckit:

If Err = 5 Then
esume Next
lse
MsgBox ("The error is " & Err)
esume Next
d If


d Sub

b Form_Unload (Cancel As Integer)
trtform.Show

d Sub

b Tool_Bid_Start_Change ()
When this toolbox changes to a 1 it is the signal to this application that the
dding data for
he tool that is being auctioned has been sent and it is up to this application
```

```
 to formulate
'a bid for each ACTIVE VMD in the Cell

'declare general variables
Dim thisfilesname As String
Dim Thisrecordslength As Integer
Dim ThisstringToprint As String
Dim i As Integer
Dim Filenum As Integer
Dim s As Integer
Dim j As Integer

'declare queuing variables
Dim Inqueuedata As String * 20
Dim position As Integer
Dim Thisjob() As JobRecord
Dim SingleJobrecord As JobRecord
Dim Thisdata As String

'tool status\Bidding variables
Dim Tool1Status  As Single
Dim Tool2Status  As Single
Dim Tool3Status  As Single
Dim Tool4Status  As Single
Dim ToolNotReservedPosition As Integer
Dim FirstToolStatus As String
Dim SecondToolstatus As String
Dim ThirdToolstatus As String
Dim FourthToolstatus As String
Dim Toollocation As String
Dim ToolTypecode As String
Dim VMDBidding As Integer
Dim ToolBiddingPosition As Integer
Dim IsNothingInqueue As Integer
Dim Teststring As String
Dim statuscount As Integer
Dim Tooltype1 As Single
Dim Tooltype2 As Single
Dim ToolType3 As Single
Dim ToolType4 As Single
Dim TaskToolType As Single
Dim ReservedStatus As String * 10
Dim ThisDuedate As String * 15
Dim ThisdueTime As String * 5


'due date bid calc dec
Dim ThisTime As String * 5
Dim Thisdate As String * 15


On Error Resume Next


Select Case Val(Tool_Bid_Start)
```

```
Case Is = 1



 then the factory controller is asking this cell to formulate an auction for the
 tool bidfile
 'that has been distributed.

For i = 1 To 4 'for each of the vmd's in the cell controller attempt to submit a
 bid for the tool
    'being offered

    'therefore we are required to find out whether or not the current vmd in thi
  cell is
    'active or not and if it isnt it will not be allowed to submit a bid for the
 tool being
    'auctioned.

    If MachineSpecArray(i) = 0 Then 'this vmd is not active therefore try the ne
 t VMD
        GoTo Next_I
    End If

    'therefore the current VMD is active so lets open it's incoming queue file a
nd interogate
    'the first job in the queue if there is one

    thisfilesname = "C:\Cellcont\Queue\Inqueue\V" & CStr(i) & "inqueu.txt"

    Thisrecordslength = 20
    Filenum = FreeFile

    Call OpenForRead(thisfilesname, Thisrecordslength, Filenum) 'open each VMD's
 inqueue file in turn and see what,if any tasks
    'exists in each VMD's ingoing queue - if there are any tasks to be done then
 compare the tool
    'type of the tool being auctioned with the tool required off the top of that
 VMD's ingoing
    'queue

    'now that the file is open get the first record because this is the record t
at we want to
    'consider in the context of getting the required tools for it.
    Get Filenum, 1, Inqueuedata

    MsgBox (Inqueuedata)
    Close Filenum

    'now we are required to see if the queue is actually empty - in other words
f the first record
    'has nothing in it.
tryagain:
```

```
    Thisdata = Inqueuedata
    'MsgBox (Len(Trim(Thisdata)))
    If Len(Trim(Thisdata)) = 20 Then 'then there is nothing in the queue so try
the next VMD
        GoTo Next_I
        Else 'there is something in the queue therefore open its jobfile so that
 we are able to
        'see what types of tools are required to do that job's current task.

        'Type JobRecord
            'Jobname As String * 13
            'InFoFilename As String * 20
            'FinalTask As Integer
            'Tasknumber As Integer
            'Machinetype As Single 'Type as 3.1 the ".1" represents capacity
            'Taskname As String * 22
            'EstCycletime As Integer
            'Setuptime As Integer
            'Estloadunloadtime As Integer
            'NoofCycles As Integer
            'CNCfilename As String * 13
            'Geometricfilename As String * 13
            'Tool1required As Single
            'Fixturesfilename As String * 13
            'Fixturesfilename As String * 13
            'Status As Integer
            'StartTime As String * 5
            'Duedate As String * 15
            'RequiredCompDate As String * 15
            'RequiredComptime As String * 5
            'Force As Integer
            'SetUpInstructionsFilename As String * 20 'Shows datum graphically
        'End Type


            'define a file handle
            Filenum = FreeFile


            'define the name of the file to be opened
            thisfilesname = Netdrive & "Factory\Jobfiles\" & Trim(Inqueuedata) &
 ".iff"

            'define the length of the array variable
            Thisrecordslength = Len(SingleJobrecord)
            'MsgBox (Thisfilesname)

            Call OpenForRead(thisfilesname, Thisrecordslength, Filenum)

            'get the first record from the file so that we can get the status of
 the job - in
            'other words so that we can get an indication of what the current ta
sk is

            'redimension to take the first record in the array
            ReDim Thisjob(1)
```

```
            'get the first record
            Get Filenum, 1, Thisjob(1)

            'redimension the array to take all the task records up to and inclu
ing the current
            'task.
            ReDim Preserve Thisjob(Thisjob(1).Status)

            'get all the records up to and including the current task.
            For s = 1 To Thisjob(1).Status
                Get Filenum, s, Thisjob(s)
            Next


            'store the current task tools required data in event level variable
TaskToolTyp(1)
             TaskToolType = Thisjob(Thisjob(1).Status).Tool1required
             'store the current task's due date required data
             Thisdate = Thisjob(Thisjob(1).Status).DueDate'RequiredCompDate
             ThisTime = Thisjob(Thisjob(1).Status).StartTime'RequiredComptime


'MsgBox (CStr(TaskToolTyp(1)))
      ''    TaskToolTyp(2) = Thisjob(Thisjob(1).Status).Tool2required
'          MsgBox (CStr(TaskToolTyp(2)))
        ''  TaskToolTyp(3) = Thisjob(Thisjob(1).Status).Tool3required
 '        MsgBox (CStr(TaskToolTyp(3)))
      ''    TaskToolTyp(4) = Thisjob(Thisjob(1).Status).Tool4required
  '         MsgBox (CStr(TaskToolTyp(4)))
  '
            'close the file
            Close Filenum

            'open the network file that stores the reserved status of the tools
hat are required
            'for the current task

           ' Dim thisfilesname As String
           ' Dim Thisrecordslength As Integer
           ' Dim ThisstringToprint As String


            'define a file handle
            Filenum = FreeFile

            'define the filename and directory
            thisfilesname = Netdrive & "Cellcont\Toolbid\" & Trim(Text1.Text) &
V" & CStr(i) & "Rst.res"

           ' MsgBox (thisfilesname)
            'define the length of the record required to be stored
            Thisrecordslength = 10

            'open the file that defines the tools reserved status for the task a
```

```
t the top of the
            'VMD's queue
            Call openForInput(Filenum, thisfilesname, Thisrecordslength)

            'now lets get the data from the file


            'For j = 1 To 4
                'clear the old bit
            '    Toolstatus(j) = ""
            '    Line Input #filenum, Toolstatus(j)
        '        MsgBox (Toolstatus(j))
            'Next

            ReservedStatus = ""

            Line Input #Filenum, ReservedStatus

            Close Filenum

            'now we are required to open up the tool bid file and get the biddir
g data from it
            'and compare the bid data with the idle tool position tool type

            'define a file handle
            Filenum = FreeFile

            'define the name of the file to open
            thisfilesname = Netdrive & "Factory\Bidding\Bids\" & Trim(Text1.Tex
) & "BdDat.bdf"
            'define the file's length
            Thisrecordslength = 8

            'open the file for input
            Call openForInput(Filenum, thisfilesname, Thisrecordslength)

            'get the current tool's location
            Line Input #Filenum, Toollocation
            'MsgBox (Toollocation)
            'get the current tool's specification
            Line Input #Filenum, ToolTypecode
        ''    MsgBox (ToolTypecode)

            Close Filenum

            'now we are required to see if there is a match between the toolcode
 in the bidfile
            'and that in the toolcode at the "ToolNotReservedposition"

            'now we are required to see if there are any tools in the file that
have not been
            'reserved

            If CDbl(ToolTypecode) = TaskToolType Then 'the tool in the bidfile m
atches one
                'of the tools in the toolfile so maybe this VMD needs to submit
```

```
 bid for that
                'tool - but it only needs to do that if it hasn't already got o
e -if it hasnt
                'then it needs it so it should submit a bid
                   If Trim(ReservedStatus) = "Unreserved" Or Trim(ReservedStat
s) = "" Then 'a tool of this type has not been
                      'reserved for this task therefore we need to submit a b
 for the tool

                      'the first thing that we are required to do is to calc
ate a bid based
                      'on the job required to be done's due date, this bid i
based on the
                      'difference between the due date of the job on top of t
e VMD's queue
                      'and the beginning of the year by considering how many
our's difference
                      'exist between the beginning of the year and the due da
te and time of the
                      'job, the greater the difference the larger the bid
                         Dim Currentmonth As String * 2
                         Dim Daystring As String * 2
                         Dim CurrentDay As Integer
                         Dim Arrayposition As Integer
                         Dim p As Integer
                         Dim NoDays As Integer
                         Dim NumberOfHours As String
                         Dim FirstTime As Integer
                         Dim SecondTime As Single
                         Dim Totaltime As Single

                         'On Error Resume Next



                         Montharray(1) = 31
                         Montharray(2) = 29
                         Montharray(3) = 31
                         Montharray(4) = 30
                         Montharray(5) = 31
                         Montharray(6) = 30
                         Montharray(7) = 31
                         Montharray(8) = 31
                         Montharray(9) = 30
                         Montharray(10) = 31
                         Montharray(11) = 30
                         Montharray(12) = 31


                         'get the first part of the due date time which is th
 number of hours
                         'of slack time in the day that the job is due(eg:if
he job is due at
                         '09:20 hrs this time will be 9 hours obviously)
                         FirstTime = Val(ThisTime)
```

```
                        'in the 09:20 hrs example above we also want to ge
the :20 part in
                        'terms of hours to get the total hrs in the day,th
 part does this
                        SecondTime = (Val(Right$(ThisTime, 2))) / 60


                        'MsgBox ("the number of hours is " & FirstTime)
                        'MsgBox ("the number of Minutes is " & CStr(SecondT
me) & " Hours")

                        'calculate the total hours in the day
                        Totaltime = FirstTime + SecondTime
                        'MsgBox (CStr(Totaltime))

                        'get the string value of the task's due date month
                        Currentmonth = (Thisdate)

                        'transfer the current month into an integer eg:for
8-20-1996 we are getting the
                        'fact that it is month "8"
                        Arrayposition = CInt(Currentmonth)

                        'extract the right part of the "thisdate" string so
we can
                        'access the day part of it ie: the fact that it is
ay 20 for the above
                        Daystring = (Right$(Trim(Thisdate), 7))

                        'transfer the Daystring variable into an integer so
that we can use it mathematically
                        'and minus a day to say that although we are in the
day we have not used all of it
                        CurrentDay = CInt(Daystring) - 1

                        If Arrayposition = 1 Then
                        GoTo skipmonthlycount
                        End If

                        'total up the days of the months up to the current m
onth
                        For p = 1 To (Arrayposition - 1)
                            NoDays = Montharray(p) + NoDays
                        Next

skipmonthlycount:
                        'add the number of days in the current month that ha
ve been used
                        NoDays = NoDays + CurrentDay
                        'MsgBox (CStr(NoDays))

                        'calculate the number of hours this represents
                        NumberOfHours = CStr((NoDays * 24) + Totaltime)

                        'MsgBox CStr(NumberOfHours)
```

```
                        'define a file handle
                        Filenum = FreeFile
                        Thisrecordslength = 8
                        thisfilesname = Netdrive & "Cellcont\Toolbid\" & Tr:
 (Text1.Text) & "TLBid.Bid"
                        'MsgBox (Thisfilesname)

                        Call OpenForOutput(Filenum, Thisrecordslength, this:
ilesname)

                        VMDBidding = i
                        'ToolBiddingPosition = j

                        'what we are doing here is finding out whether or nc
t a tool that is
                        'required by the task on the top of the current VMD
 ieue whether it be
                        'the first, second, third or fourth tool matches tha
 being auctioned -
                        'if it does match then we are sending a "Bid" to the
 factory Controller
                        'that states what VMD is conducting a "Bid" and wher
 abouts in the
                        'jobfile the bids match - then we terminate any more
 bidding by this cell
                        'as we know that as all processes within a cell are
 eterogenous there
                        'is no possibility that any other VMD can hold a bid

                        'now output the data to the file
                        Print #Filenum, VMDBidding
                        Print #Filenum, NumberOfHours

                        'Now close the file
                        Close Filenum

                        GoTo resetblock
                        Exit Sub
                Else
                    'otherwise the tool is already reserved for this task
nd if that is
                    'the case then there will be no other VMD's in this ce
 that will need
                    'this tool because this has already reserved this tool
 nd all other
                    'processes in the cell are different and therefore can
ot bid for this
                    'tool so exit the sub
                    GoTo resetblock
```

```
                        Exit Sub
                    End If
                End If
            End If
Next_I:
Next

'if the program gets to here it means that no VMD is bidding therefore we must c
lter the
'systems files to reflect this

'define a file handle
Filenum = FreeFile
Thisrecordslength = 8
thisfilesname = Netdrive & "Cellcont\Toolbid\" & Trim(Text1.Text) & "TLBid.Bid"
'MsgBox (Thisfilesname)

'MsgBox ("no bid given - factory controller corrupted")
Call OpenForOutput(Filenum, Thisrecordslength, thisfilesname)

VMDBidding = 0
NumberOfHours = ""

Print #Filenum, VMDBidding
Print #Filenum, NumberOfHours

'Now close the file
Close Filenum


resetblock:
'but we are also required to send a 'zero' to the "bdtoolstrt" database block s
 that it is able
'to respond to the next change event that this textbox experiences
Node = CStr(Text1.Text)
Tags = "BDTOOLSTRT"
Field = "A_CV"
Texts = "0"

Call sendinfo(Node, Tags, Field, Texts)

'*************************************************************************
'in here wre are required to send some flags to the Factory Controller
'stating that the tool bidding for this cell is finished so that it can
'compile the toolbids and compare them.

Node = "Fact"
Tags = Trim(Text1.Text) & "ToolFin"
Field = "A_CV"
Texts = "1"

Call sendinfo(Node, Tags, Field, Texts)


End Select
```

```
End Sub
```

```
Option Explicit
Option Base 1
Option Compare Text



'****General Global Variables*****************
Global Thistime As Integer


'****General Global Constants*****************
Global Const Netdrive = "r:\"

'******************** DDE Constants **********************
' LinkMode (forms and controls)
Global Const NONE = 0          ' 0 - None
Global Const LINK_SOURCE = 1     ' 1 - Source (forms only)
Global Const LINK_AUTOMATIC = 1 ' 1 - Automatic (controls only)
Global Const LINK_MANUAL = 2     ' 2 - Manual (controls only)
Global Const LINK_NOTIFY = 3     ' 3 - Notify (controls only)

Type Bid_data_To_Be_sent
Jobname As String * 10
Machinetype As Single
EstCycletime As Integer
Setuptime As Integer
LoadUnloadtime As Integer
NoofCycles As Integer
Tool1required As Single
'Tool2required As Single
'Tool3required As Single
'Tool4required As Single
Activate As Integer
Force As Integer
AuctioningCell As String * 8
End Type

'declare variable type who's array describes a VMD's tools that are available
Type ToolSpec
'Toolname As String * 40
    ToolTypeSize As Single
    NoOfToolsNeeded As Integer
End Type

'this type of array stores the bid of each node in an array
Type Nodesbid
    VMD1bid As Single
    VMD2bid As Single
    VMD3bid As Single
    VMD4bid As Single
End Type

 this variable stores the current cost of each queue in each cell
ype VMDQueueCost
    Cellname As String * 8
    VMD1_Queue_Cost As Single
```

```
IDFORM.BAS - 2

    VMD2_Queue_Cost As Single
    VMD3_Queue_Cost As Single
    VMD4_Queue_Cost As Single
End Type

'Declare data type to store VMD's Maximum allowed buffer size
Type InQueueBuffersize
    VMD1Buffer As Integer
    VMD2Buffer As Integer
    VMD3Buffer As Integer
    VMD4Buffer As Integer
End Type

'define variable type that store the internodal part travel times
Type NodeTimedelay
    MatrixDim As Integer
    Interarrivaltime As Integer
    CellnameX As String * 10
    CellnameY As String * 10
End Type


'define the job record storage array

Type JobRecord
    Jobname As String * 13
    InFoFilename As String * 20
    FinalTask As Integer
    Tasknumber As Integer
    Machinetype As Single 'Type as 3.1 the ".1" represents capacity
    Taskname As String * 22
    EstCycletime As Integer
    Setuptime As Integer
    Estloadunloadtime As Integer
    NoofCycles As Integer
    CNCfilename As String * 13
    Geometricfilename As String * 13
    Tool1required As Single
    'Tool2required As Single
    'Tool3required As Single
    'Tool4required As Single
    Fixturesfilename As String * 13
    Status As Integer
    StartTime As String * 5
    Duedate As String * 15
    RequiredCompDate As String * 15
    RequiredComptime As String * 5
    Force As Integer
    SetUpInstructionsFilename As String * 20 'Shows datum graphically
End Type


b openForInput (Thisfilesnum As Integer, Filename As String, recordlength As I
teger)
```

```
Open Trim(Filename) For Input As Thisfilesnum Len = recordlength


End Sub

Sub OpenForOutput (Filenum As Integer, recordlength As Integer, Filenamed As St
ing)

Open Trim(Filenamed) For Output As Filenum Len = recordlength

End Sub

Sub OpenForRead (Filename As String, recordlength As Integer, Thisfilenum As In
eger)

Open Trim(Filename) For Random Access Read As Thisfilenum Len = recordlength


End Sub

Sub Output_To_File (Filename As String, Filelength As Integer, StringToPrint As
String)
Dim Filenum As Integer

Filenum = FreeFile

Open Trim(Filename) For Output As Filenum Len = 4

Print #Filenum, Trim(StringToPrint)

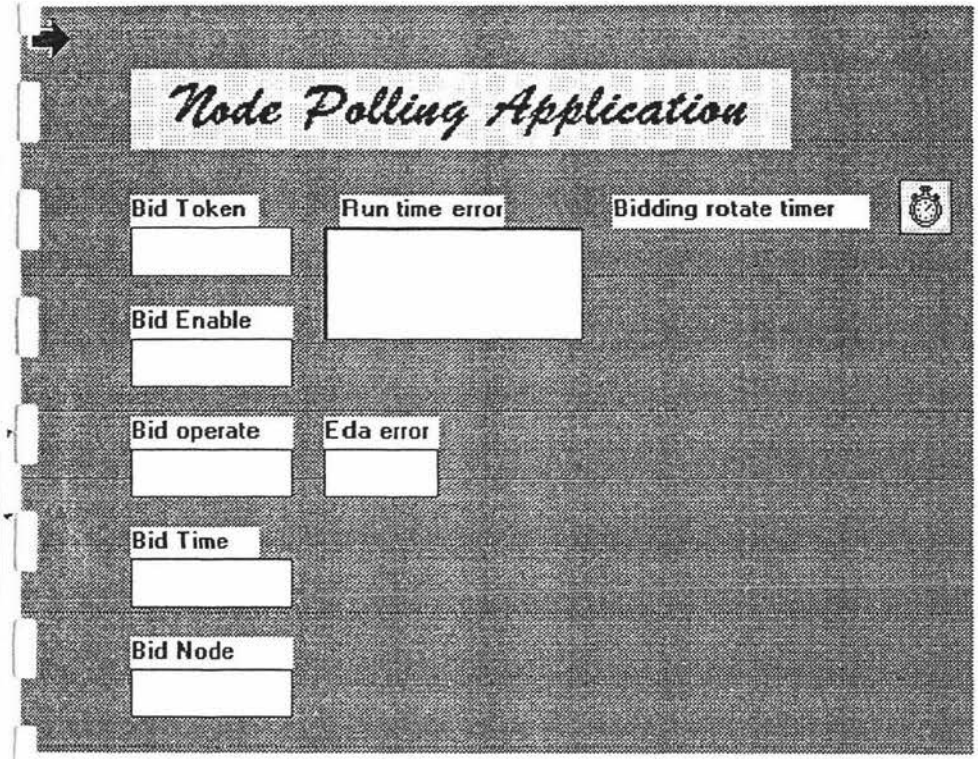
Close Filenum

End Sub
```

A.6 Implementation of the Node Polling Application

The Node Polling Application is responsible for ensuring that all nodes within the cellular environment get equal opportunity for the holding of auctions. This is carried out by the application circulating a token. Cells within the system are given the opportunity of holding an auction once the token is received and held.

The following material is a print out from Microsoft Visual Basic showing all programming carried out for the applications implementation.


Node Polling Application
Bid Token
Run time error
Bidding rotate timer
Bid Enable
Bid operate
Eda error
Bid Time
Bid Node

```
ID_ROT.FRM - 1

ption Explicit

Const Close_DDE = 5, Open_DDE = 1, none = 0, Link_Manual = 2, link_automatic =

declare record type
Dim Nodelistfile As nodelistrecord

Declare variables for file manipulation
Dim position As Integer
Dim Filenum As Integer

Dim Counter As Integer
'eda communication variables
Dim Ghandler As Long
Dim fart As Variant
Dim eda_err As Variant
Dim tex As Integer
Dim Newnum As Variant
Dim node As String * 15
Dim tags As String * 15
Dim Field As String * 15
Dim answer As String * 80
Dim ans As String * 10
Dim defnode As Long
Dim trap As String * 10
Dim texts As String * 6
Dim Nodetyp As String * 20
Dim nodes As String * 20
Dim bidrotenable As Variant
Dim BlockTyp As String * 20
Dim Bidoperatestate As Variant
Dim Bidnodetext As String * 10
Dim bidtokenstate As Variant
Dim Tokenstate As Integer
Dim BidOpstate As Variant

' functions specific to VB (not standard eda fcns)
Declare Sub eda_version Lib "vdba.dll" (ByVal v$)
Declare Sub eda_get1_ascii Lib "vdba.dll" (ByVal n$, ByVal t$, ByVal f$, ByVal
s)
Declare Function eda_set1_ascii% Lib "vdba.dll" (ByVal n$, ByVal t$, ByVal f$,
ByVal v$)

' Note define_group returns a long

Declare Function eda_define_group& Lib "vdba.dll" (ByVal Count%, ByVal Detect%)
Declare Sub eda_delete_group Lib "vdba.dll" (ByVal Handle&)

' should always pass 0 for vsp
Declare Function eda_define_ntf& Lib "vdba.dll" (ByVal Handle&, ByVal enode$, B
yVal etag$, ByVal efield$, ByVal vsp&)
Declare Sub eda_delete_ntf Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
'
Declare Sub eda_lookup Lib "vdba.dll" (ByVal Handle&)
Declare Sub eda_read Lib "vdba.dll" (ByVal Handle&)
Declare Sub eda_write Lib "vdba.dll" (ByVal Handle&)
```

```
Declare Function eda_write1% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
Declare Sub eda_wait Lib "vdba.dll" (ByVal Handle&)
Declare Function eda_query_wait% Lib "vdba.dll" (ByVal Handle&)
'
Declare Function eda_get_error% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
Declare Function eda_get_ascii% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, ByVa
 value$, ByVal maxlen%)
Declare Function eda_set_ascii% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, ByVa
 value$)

' note pointer to float is passed
Declare Function eda_get_float% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, valu
!)
Declare Function eda_set_float% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, valu
!)

'Matts add ons
Dim Firstpass As Integer

Sub Bidrequest_Change ()

'This module is the control for the bidding object transportation
'If 1 is entered then the bidrequest is transported from the source node to the
factory controller
'If 2 is entered then the bidding objects are loaded from the source node into
his application
'If 3 is entered then the data is transfered from the current source links to t
e factory links
'If 4 is entered then the bidding objects in the factory links are transported
o the factory controller

'On Error Resume Next

'Select Case bidrequest.Text

 '   Case Is = 0
        'Reset bidrequest
  '      bidrequest2.Text = 0
    '
     'Case Is = 1
        'Transport the bidrequest to the fatory controller
      '   bidrequest2.Text = 1
       '  bidrequest2.LinkPoke

    ' Case Is = 2
        'Request from source node all bidding objects except bid request
     '    bidcycletime.LinkRequest
      '   bidloadunloadtime.LinkRequest
       '  bidmachinetype.LinkRequest
        ' bidpartno.LinkRequest
        'bidsetuptime.LinkRequest
        ' bidtoolstype.LinkRequest
        'bidiffile.LinkRequest
'        bidjob.LinkRequest

 '   Case Is = 3
```

```
        'transfer these bidding objects to the outgoing factory controller text
ox's
  '      bidcycletime2.Text = bidcycletime.Text
   '     bidloadunloadtime2.Text = bidloadunloadtime.Text
    '    bidmachinetype2.Text = bidmachinetype.Text
     '   bidpartno2.Text = bidpartno.Text
      '  bidsetuptime2.Text = bidsetuptime.Text
       ' bidtoolstype2.Text = bidtoolstype.Text
        'bidiffile2.Text = bidiffile.Text
        'bidjob2.Text = bidjob.Text

'    Case Is = 4
        'poke all the outgoing bidding objects to the factory controller
'       bidcycletime2.LinkPoke
  '     bidloadunloadtime2.LinkPoke
   '    bidmachinetype2.LinkPoke
    '   bidpartno2.LinkPoke
     '  bidsetuptime2.LinkPoke
      '  bidtoolstype2.LinkPoke
       ' bidiffile2.LinkPoke
      '  bidjob2.LinkPoke

    Case Else
       'do nothing


End Select









nd Sub

Sub CommDDELink (LinkItem, DDETextName As TextBox, link_type As Integer)

'This procedure is designed to run once on startup to set up a DDE link with fi
dmacs
' The Block name is passed in as Linkitem and the DDE Text box name is passed in
as
' DDETextName



n Error Resume Next
       DDETextName.LinkTopic = "DMDDE|DATA"     'set topic
       DDETextName.LinkItem = LinkItem          'set link item, this is passed

       DDETextName.LinkMode = link_type         'sets up an link


nd Sub
```

```
Function eda_requestinfolink (n, t, f)
Ghandler = eda_define_group(1, 0)
defnode = eda_define_ntf(Ghandler, n, t, f, 0)

Call eda_lookup(Ghandler)
Call eda_wait(Ghandler)
Call eda_read(Ghandler)
Call eda_wait(Ghandler)
Newnum = eda_get_ascii(Ghandler, defnode, answer, 50)
Call eda_delete_group(Ghandler)
eda_requestinfolink = answer
End Function

Sub resetoperate ()

'This procedure is designed to reset the bidoperate blocks before this bidrotat
 application closes down
Close

' Create new file specs
Filenum = FreeFile

' Opening file machine list file
Open defaultpath & "nodefile.txt" For Random Shared As Filenum Len = Len(Nodeli
tfile)

'Do until reachs end of file
If Not EOF(Filenum) Then

    Do Until EOF(Filenum)

        If Trim(Nodelistfile.node) Like "???#" Then

           'if record is not empty then setup links for bidding enable
            Call CommDDELink(Nodelistfile.node & ".bidenable.A_CV", Bidenablete
t, Link_Manual)

            ' set the bidenable to 0 to let the nodes know that a bodrota appli
ation is closeing down
            Bidenabletext.Text = 0
            Bidenabletext.LinkPoke

            'Close DDE link to this DBB
            Bidenabletext.LinkMode = none

            End If

            position = position + 1
            Get Filenum, position, Nodelistfile

            'safety trip
            If position = 20 Then
                MsgBox "File access error in nodefile.txt"
                End
                End If
```

```
        Loop
End If

close file
close Filenum

End Sub

Sub sendinfo (n, t, f, Text)
Ghandler = eda_define_group(1, 0)
defnode = eda_define_ntf(Ghandler, n, t, f, 0)

Call eda_lookup(Ghandler)
Call eda_wait(Ghandler)
eda_err = eda_get_error(Ghandler, defnode)
Text5.Text = Str$(eda_err)
Newnum = Text
Part = eda_set_ascii(Ghandler, defnode, Newnum)
Call eda_write(Ghandler)
Call eda_wait(Ghandler)
Call eda_delete_group(Ghandler)

End Sub

Sub Bid_token_Click ()
Form100.Show
End Sub

Sub Bidenable_Click ()
Form101.Show
End Sub

Sub Bidnodes_Click ()
Form103.Show
End Sub

Sub Bidoperate_Click ()
Form102.Show
End Sub

Sub bidtime_Change ()

Timer2.Interval = bidtime.Text
End Sub

Sub Form_Load ()
errorDim Recordlength As Integer

'This application was written by
'Matthew Jolly with help from Michael Butler
'Department of Production Technology
'New Zealand
'Copyright 1994

'This application is designed to rotate a bidding token around the nodes that
'consist of the factory this is done by first testing to see if any other
```

```
'copies of the bidrotation program are operating by testing to see if the
'bidenable block is 1, if it is zero then it sets the bid operate to 1 and
'then starts moving the bidtoken around the factory.  If a node has something
'to auction then it will take the bidding token by setting the bidoperate
'block to 1 for the duration of the bidding process. When that node has
'finished bidding the bidoperate block is reeset to zero


' Create new file specs
Filenum = FreeFile
position = 1

'sets timer interval into screen
bidtime.Text = Timer2.Interval

'open DDE links between this bidrotation program and the factory controller

'Call CommDDELink(Factory_Controller & ".bidrequest.A_CV", bidrequest2, Link_M
ual)
'Call CommDDELink(Factory_Controller & ".bidcyctm.A_CV", bidcycletime2, Link_M
ual)
'Call CommDDELink(Factory_Controller & ".bidloadun.A_CV", bidloadunloadtime2, L
nk_Manual)
'Call CommDDELink(Factory_Controller & ".bidmachtyp.A_CV", bidmachinetype2, Li
_Manual)
'Call CommDDELink(Factory_Controller & ".bidpartno.A_CV", bidpartno2, Link_Man
l)
'Call CommDDELink(Factory_Controller & ".bidsetupm.A_CV", bidsetuptime2, Link_M
nual)
'Call CommDDELink(Factory_Controller & ".bidtooltyp.A_CV", bidtoolstype2, Link
anual)
'Call CommDDELink(Factory_Controller & ".bidiffile.A_CV", bidiffile2, Link_Man
l)
'Call CommDDELink(Factory_Controller & ".bidjob.A_CV", bidjob2, Link_Manual)

On Error Resume Next


Dim Thisrecord As nodelistrecord
Dim posi As Integer
Dim Firstime As Integer
Dim node As String
Dim Field As String
Dim BlockTyp As String
Dim texts As String


If FileLen("c:\Factory\Config\nodefile.txt") = 0 Then 'if there is nothing in
e file then exit
'this subprocedure
    MsgBox "No nodes are currently configured, Please configure"
    Exit Sub
End If


'Assign a free number to this file
```

```
 ilenum = FreeFile
 Calculate the recordlength in Bytes
Recordlength = Len(Thisrecord)

  Opening file machine list file
Open "C:\Factory\Config\nodefile.txt" For Random Access Read Shared As Filenum L
 n = Recordlength

'Set position to 1 to get first record
posision = 1

'Do until reachs end of file
If Not EOF(Filenum) Then

    ' Get the name of the first node from the file
    Get Filenum, posi, Nodelistfile

    Do Until EOF(Filenum)

        'display to error text box
        ERRORTEXT.Text = Error

        Bidnodetext = Nodelistfile.node
         MsgBox (Bidnodetext)
        BlockTyp = "Bidenable"
        Field = "A_CV"
        'Retrieve the value of the bidenable block on the first node
        bidrotenable = eda_requestinfolink(Bidnodetext, BlockTyp, Field)
        Bidenabletext.Text = bidrotenable
           'Test to see if another copy of the bidrotation program is
           'operating by Checking to see if another copy of it has placed
           'a "1" in this nodes Bidenable block
           If Bidenabletext.Text = 0 Then
                'Place a "1" in the current nodes bidenable block
                Bidenabletext.Text = 1
                Bidnodetext = Nodelistfile.node
                BlockTyp = "Bidenable"
                Field = "A_CV"
                tex = Bidenabletext.Text
                Call sendinfo(Bidnodetext, BlockTyp, Field, tex)

                Else
                    'another bidrota application must be operating goto end and
close all links
                    End
           End If


            posi = posi + 1
            Get Filenum, posi, Nodelistfile
            'safety trip
            If position = 20 Then
                MsgBox "File access error in nodefile.txt"
                End
                End If
```

```
        Loop
End If

'set position to 1 again
posi = 1

'close file
Close Filenum

'display to error text box
ERRORTEXT.Text = Error

Timer2.Enabled = True
End Sub

Sub mnuabout_Click ()
MsgBox "This application was Designed by Michael Butler, Copyright 1994.  It wa
 designed to pass a Token around the Virtual Factory Database Blocks allowing ea
ch cell controller to have a chance to become the center of Bid decision making
  The token is passed around in a block called BidToken.F_CV.  The Bidenable.F_
V block goes to 1 if this application is operating.  The block Bidoperate.F_CV
s used by the cell to establish if it wishes to use the bid tiken and make a bi
.", MB_ICONEXCLAMATION

End Sub

Sub mnuhowto_Click ()
MsgBox "This module is the control for the bidding object transportation.  If 1
is entered then the bidrequest is transported from the source node to the facto
y controller.  If 2 is entered then the bidding objects are loaded from the sou
ce node into this application.  If 3 is entered then the data is transfered fro
 the current source links to the factory links.  If 4 is entered then the biddi
g objects in the factory links are transported to the factory controller", MB_I
ONEXCLAMATION

End Sub

Sub Timer2_Timer ()
Dim Recordlength As Integer

'This application was written by
'Matthew Jolly
'Department of Production Technology
'New Zealand
'Copyright 1994

'This portion of the program is responsible for rotating a Bid Token around a
'series of nodes as specified in the "Nodefile.txt" textfile which is created
'in the Factory Controller. When a node is given the Bid Token the BidToken
'textbox changes to a "1"-the node then looks at its job queues. If there is
'something in a queue it holds an Auction - before it can hold an auction the
'Bid Operate block is passed the value of "1" which stops the Bid Token from
'being rotated further until the auction is complete.When the auction is
'complete or the allowable auction time has passed the BidOperate block is
'passed the value of "1" which allows the Bidtoken to be moved to the next node
```

```
'so giving it the opportunity to hold an auction

On Error Resume Next


' Create new file specs
Filenum = FreeFile
ecordlength = Len(Nodelistfile)

' Open the listfile containing the names of all the nodes in the Factory
pen "C:\Factory\Config\nodefile.txt" For Random Access Read Shared As Filenum L
en = Recordlength

display to error text box
ERRORTEXT.Text = Error

disable timer so it is unable to respond to timer event
Timer2.Enabled = False


If the timer event has been executed more than once after the load event
f Firstpass <> 0 Then
   Bidnodetext = Bidnode.Text
   BlockTyp = "Bidoperate"
   Field = "A_CV"
   'Request the value of the bidoperate block in the first node
   Bidoperatestate = eda_requestinfolink(Bidnodetext, BlockTyp, Field)
   Bidoperatetext.Text = Bidoperatestate

   BlockTyp = "Bidtoken"
   'Request the value of the bidtoken block in the first node
   bidtokenstate = eda_requestinfolink(Bidnodetext, BlockTyp, Field)
   Bidtokentext.Text = bidtokenstate

        'If the current node has nothing to auction or it hasn't got the
        'bidtoken or the auction time is up then take the bidtoken off the
        'current node and pass it to the next node

        If CInt(Bidoperatetext.Text) <> 1 Or CInt(Bidtokentext.Text) <> 1 Or Cou
nter = 9 Then
              'Reset Bidding Token
              Bidtokentext.Text = 0
              Tokenstate = Bidtokentext.Text
              Bidnodetext = Bidnode.Text
              BlockTyp = "Bidtoken"
              'Reset Counter variable
              Counter = 0

              'Take bidtoken off current node
              Call sendinfo(Bidnodetext, BlockTyp, Field, Tokenstate)

              'Setout bidoperate = 0
              BlockTyp = "Bidoperate"
              Bidoperatetext.Text = 0
              BidOpstate = Bidoperatetext.Text
              Call sendinfo(Bidnodetext, BlockTyp, Field, BidOpstate)
```

```
            position = position + 1


            'Get the next node(example Cell Controller 2) name from the
            'nodefile
            Get Filenum, position, Nodelistfile


                    If Not EOF(Filenum) Then
                            'Pass a value of "1" to the next node's bidtoker
                            'block to give it a chance to conduct an auction
                          Bidtokentext.Text = 1
                            Tokenstate = Bidtokentext.Text
                            Field = "A_CV"
                            BlockTyp = "Bidtoken"
                            Bidnodetext = Nodelistfile.node
'                            MsgBox (Bidnodetext)
                            'Send the value of "1" to the bidtoken block
                            Call sendinfo(Bidnodetext, BlockTyp, Field, Toke
state)
                            Bidnode.Text = Nodelistfile.node
                            Else
                            'end of file reached so reset firstpass variable
and the
                            'position variable
                            Firstpass = 0
                            position = 1
                            End If
              Else
            'Bidtoken has been taken & Bidding is occurring
              Counter = Counter + 1 'add "1" to timeout Counter
              ERRORTEXT.Text = Counter
         End If
     Else
   'First pass through timer event, get first node name from the nodefile.txt
   'file and pass the bidtoken to it
   Get Filenum, position, Nodelistfile
   Bidtokentext.Text = 1
   Bidnodetext = Nodelistfile.node
   BlockTyp = "Bidtoken"
   Field = "A_CV"
   Tokenstate = Bidtokentext.Text
 '  MsgBox (Bidnodetext)
   Call sendinfo(Bidnodetext, BlockTyp, Field, Tokenstate)
   Bidnode.Text = Nodelistfile.node
   Firstpass = Firstpass + 1
End If

'Close nodefile.txt file
Close Filenum
'Enable timer to allow next timer event to occur
Timer2.Enabled = True

End Sub
```

Bidenable Textbox

The Bidenable Textbox is a mechanism by which the Bid Rotation program functions. The program requests data from subsequent Nodes as in the 'Nodefile.txt' file from the bidenable block in those nodes and if a '1' is contained in a node other copies of the Bid Rotation program are closed. It is a means by which the user can see if the Bid Rotation program is functioning.

```
Sub Exit_Click ()
Unload Form101
End Sub
```

Bid Operate Textbox

The Bid Operate Textbox indicates whether or not a node in the actory has retained the Bid Token. If a node takes the Bid Token then the Bid Operate textbox changes to "1" and that node's name appears in the "Bid Node" textbox, otherwise the Bid Operate textbox has a "0" in it. When the Bid Operate textbox changes to "1" the node responsible begins an auction. Once that node has completed the Auction or the Auction time allowed is over it changes the "Bid Operate textbox back to "0" and the Bid Token moves to the next node to give it the opportunity to hold an auction.

```
Sub Exit_Click ()
Unload Form102
End Sub
```

BID TOKEN TEXTBOX

The bidtoken Textbox displays a '1' within it's frame when a Node as configured in the "Nodefile.txt" recieves the Bidtoken as it is rotated about the factory. As any number of nodes are capable of recieving the bidtoken the name of the node currently possessing the token is included in the Bidnode Textbox. When the Bid Token textbox has a '0' within its frame this indicates that the Bid Rotation program is currently in the process of passing the token.

```
Sub Exit_Click ()
Unload Form100
End Sub
```

Bid Node Textbox

The Bid Node Textbox contains the name of the Node which is currently holding the Bid Token

```
Sub Exit_Click ()
Unload Form103
End Sub
```

```
MAINPROG.BAS - 1

' This application was written By:
' Matthew Jolly
'Department of Production Technology
' Massey University
' Copyright 1994

' Define a global variable to store status of user.
' This value will be Peasant, Supervisor, or Manager.
Option Explicit
Option Compare Text

Global User

' Define a variable to determine whether or not FIX is running
'Values will be "Yes" or "No"

Global fixrunning, DDERunning, DBaseRunning

Global cell 'current cell no

'***This sets the factory controller in the factory***
Global Const Netdrive = "r:\"

'Variables for every cell position:
Global cell1TL, cell1TR, cell1BL, cell1BR
Global cell2TL, cell2TR, cell2BL, cell2BR
Global cell3TL, cell3TR, cell3BL, cell3BR
Global cell4TL, cell4TR, cell4BL, cell4BR
Global cell5TL, cell5TR, cell5BL, cell5BR
Global cell6TL, cell6TR, cell6BL, cell6BR
Global cell7TL, cell7TR, cell7BL, cell7BR
Global cell8TL, cell8TR, cell8BL, cell8BR
Global cell9TL, cell9TR, cell9BL, cell9BR

' ***Definitions for job and task selection***

' This type contains all the fields/ information required about a particular job

Type jobrecord
    jobname As String * 13
    currentindex As Integer
    Finaltask As Integer
    tasknumber As Integer
    Machinetype As Integer
    taskname As String * 13
    Estcycletime As Integer
    setuptime As Integer
    Estloadunloadtime As Integer
    Noofcycles As Integer
    Machinefilename As String * 13
    Geometricfilename As String * 13
    Toolsrequired As Integer
    Fixturesfilename As String * 13
    Status As Integer
    duedate As String * 8
    force As Integer
```

```
End Type

' Define job record type and global file management variables
Global Recordname As jobrecord  'Record structure
Global temprecord As jobrecord

' Status variables for file manipulation
Global filestatus As Integer
Global isfilenew As Integer
Global filepath As String
Global position As Integer
Global filenum As Integer

' Constants for file usage
Global Const filehasbeenopened = 1
Global Const filehasbeenclosed = 0
Global Const defaultpath = "C:\Factory\Config\"
Global Const newfile = 1
Global Const modifyfile = 2
'***End***
'
'
'***Definitions for machine selection file type and ascociated parts (*.mch) **
'
' Type definitions
Type machinerecord
    Machinename As String * 14
    currentindex As Integer
End Type
'
' file manipulation variables
Global machinefilenum As Integer
'***End***
'
'
'***Definitions for Date record file type and related variables
'
'Type definition
Type daterecord
    Filename As String * 13
    Date As String * 8
End Type
'
'File manipulation variables
Global datefilenum As Integer
Global datefile As daterecord
'***End***
'
'
'***Definitions for node record type, this type contains all the information abc
ut the cells in a particular node.
'
Type noderecord

    nodename As String * 10
    Devicename As String * 20
```

```
    Devicetype As Integer
    setuptime As Integer
    downtime As Integer
    serviceinterval As Integer
    IFFiletype As Integer
    CommunicationID As Integer
    communicationparameters As String * 40
    Partservicetype As String * 15
    feedbacksignal As String * 100
    communicationsport As Integer

End Type
'
 ***END***
'

 ***Definitions for node list record type
'
Type nodelistrecord
    node As String * 10

End Type

 ***END***



ub readfromfile (record As jobrecord)
read from file at position into recordname
If position = 0 Then
    position = 1
    End If

et filenum, position, record

End Sub

ub savetofile (record As jobrecord)

im response As Integer

' save all information to file
esponse = MsgBox("This task has been modified do you wish to save it?", MB_YES
 , "Save Job")
    If response = IDYes Then
    ' save file
     Put #filenum, position, record
  End If
    ' No save file
 End Sub

ub testfix ()

 This function is designed to warn users that Fixdmacs must be operating to per
orm some of the functions in this application
```

```
Dim response, response2 As Integer                    'Variable to see if Fix is
running
Dim Title As String                                   'Check if Fix is running

If fixrunning = IDYes Then

    'This is fine
    Exit Sub

   Else
    'fixrunning = IDNo or something else

    response = MsgBox("Is FIX DMACS currently running.", 4, Title)

    If response = IDNo Then
        response = MsgBox("This operation requires Fixdmacs to be operating.
 you wish to continue.", 4, Title)

        If response2 = IDNo Then
            'end and pass out that this operation will not continue

            fixrunning = IDNo
            Exit Sub
            End If

         MsgBox "Before continuing on with this operation you must first Switch
to program manager and start Fixdmacs"

         ElseIf response = IDYes Then
            'then fix is running

            fixrunning = IDYes
         End If

    End If

End Sub
```

```
QUEUE1.BAS - 1

ption Explicit

'**For queue program

I don't know why this works but it does, variant's should not work in a type d
claration

ype queue

    Job_Name As Variant
    IF_Filename As Variant
    No_ofparts As Variant
    Setup_Time As Variant
    Tools_Required As Variant
    Machine_Type As Variant
    Cycle_Time As Variant
    Loadunload_time As Integer

End Type

' MsgBox parameters
Global Const MB_OK = 0                    ' OK button only
Global Const MB_OKCANCEL = 1              ' OK and Cancel buttons
Global Const MB_ABORTRETRYIGNORE = 2      ' Abort, Retry, and Ignore buttons
Global Const MB_YESNOCANCEL = 3           ' Yes, No, and Cancel buttons
Global Const MB_YESNO = 4                 ' Yes and No buttons
Global Const MB_RETRYCANCEL = 5           ' Retry and Cancel buttons

Global Const MB_ICONSTOP = 16             ' Critical message
Global Const MB_ICONQUESTION = 32         ' Warning query
Global Const MB_ICONEXCLAMATION = 48      ' Warning message
Global Const MB_ICONINFORMATION = 64      ' Information message

Global Const MB_APPLMODAL = 0             ' Application Modal Message Box
Global Const MB_DEFBUTTON1 = 0            ' First button is default
Global Const MB_DEFBUTTON2 = 256          ' Second button is default
Global Const MB_DEFBUTTON3 = 512          ' Third button is default
Global Const MB_SYSTEMMODAL = 4096         'System Modal

' MsgBox return values
Global Const IDOK = 1                     ' OK button pressed
Global Const IDCANCEL = 2                 ' Cancel button pressed
Global Const IDABORT = 3                  ' Abort button pressed
Global Const IDRETRY = 4                  ' Retry button pressed
Global Const IDIGNORE = 5                 ' Ignore button pressed
Global Const IDYES = 6                    ' Yes button pressed
Global Const IDNO = 7                     ' No button pressed

'** For file IF control


' Define job record type and global file management variables

  Status variables for file manipulation
```

```
Function loadApplication (Appname, Path) As Variant

'Written by Michael Butler
'Department of Production Technology
'Massey University
'
'This function takes an application name and path then attempts to start it
'if any errors occur it aborts, returns loadapplication yes or no ie it has bee
 loaded
'
'
'Improvement can be made here by testing if the application is open before it i
 opened using an API call
'

Dim response, x                                         'response from message b
x
Const IDYES = 6, IDNO = 7                               'return from message box

    On Error GoTo ErrorHandel   'if an error occurs goto errorhandel
    x = Shell(Path & Appname)
    loadApplication = True

Exit Function

ErrorHandel:

    response = MsgBox("Is " & Appname & " currently running?", MB_YESNOCANCEL,
Application Status")      'Ask if want to start application
    If response = IDYES Then
        loadApplication = True
    Else
        MsgBox "Error in opening application, this operation may not continue",
48, "Application Status"
        loadApplication = False
    End If
    Exit Function

End Function
```

Appendix B

Factory Controller Services - Visual Basic Implementation

This Appendix details the programming employed in the implementation of the Factory Controller Services. It is intended that the programming be used in conjunction with the implementation chapter in Volume 1, in this way the reader is able to get exact knowledge as to the way various features of the system have been implemented.

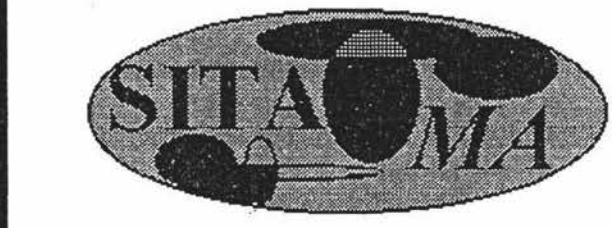
Please find as follows the Visual Basic programming implementing the Factory Controller based services.

Appendix B

Factory Controller Services - Visual Basic Implementation

This Appendix details the programming employed in the implementation of the Factory Controller Services. It is intended that the programming be used in conjunction with the implementation chapter in Volume 1, in this way the reader is able to get exact knowledge as to the way various features of the system have been implemented.

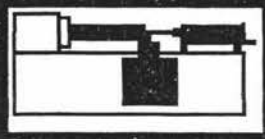
Please find as follows the Visual Basic programming implementing the Factory Controller based services.



THE SYSTEM INTEGRATION TOOL FOR ADVANCED MANUFACTURING AUTOMATION

Next

Created by Matthew John Jolly with the assistance of Dr Saeid Nahavandi, Department of Production Technology Massey University New Zealand





```
ACTINT.FRM - 1


Sub Next_Click ()

Jnload Factint
Load Factrun
'actrun.Show


;nd Sub

Sub Timer1_Timer ()

;teps = steps + 1

'f steps = 5 Then
;teps = 1
End If

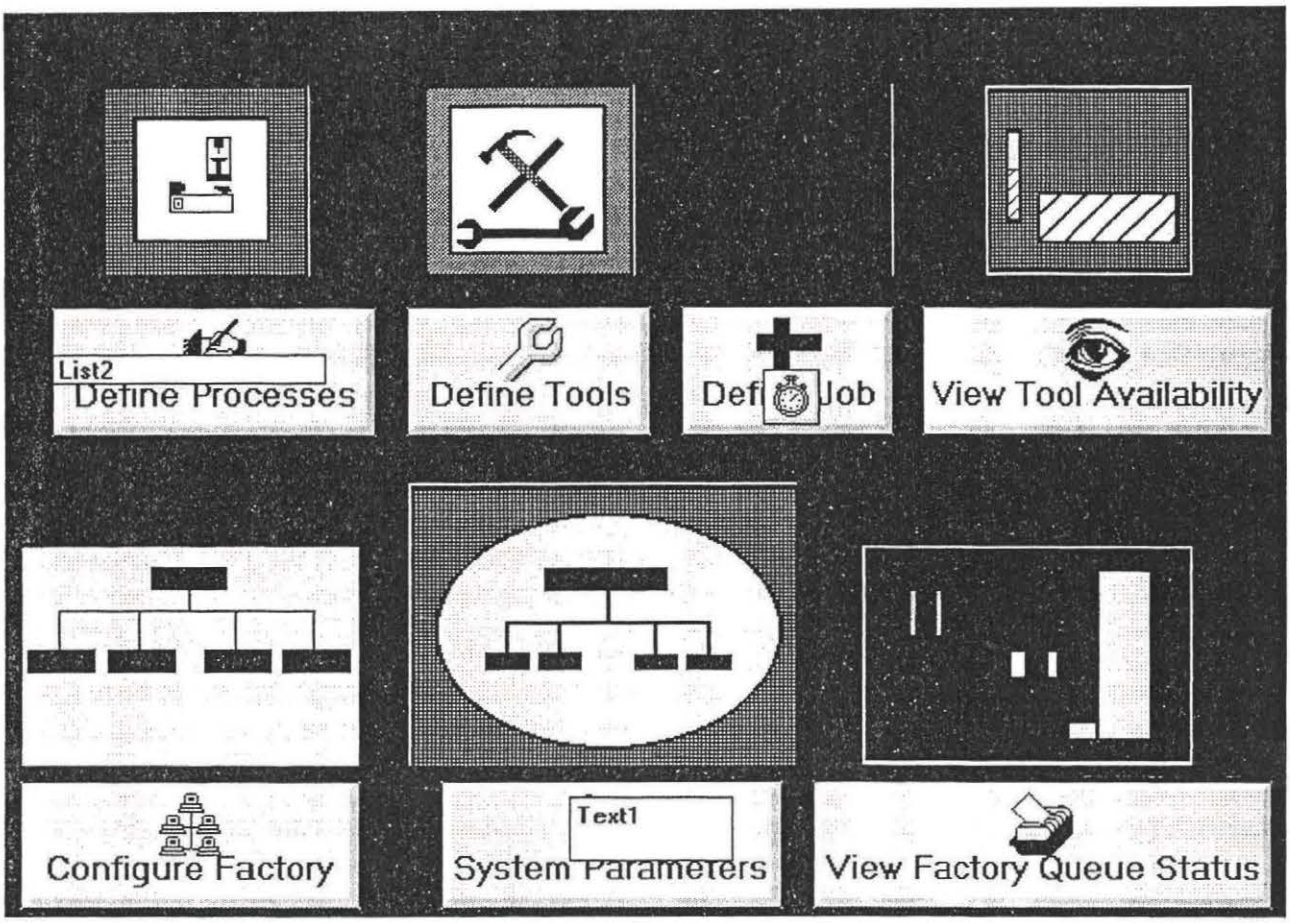

;f steps = 1 Then
Picture2.Picture = LoadPicture("c:\Factory\System\Lathe.bmp")
;nd If

If steps = 2 Then
'icture2.Picture = LoadPicture("c:\Factory\System\Lathe1.bmp")
End If

;f steps = 3 Then
'icture2.Picture = LoadPicture("c:\Factory\System\Lathe2.bmp")
End If

_f steps = 4 Then
'icture2.Picture = LoadPicture("c:\Factory\System\Lathe3.bmp")
;nd If


;nd Sub
```


List2
Define Processes
Define Tools
Def Job
View Tool Availability
Configure Factory
Text1
System Parameters
View Factory Queue Status

```
FACTRUN.FRM - 1

Option Explicit
Option Base 1

Dim Firsttime As Integer

Dim Ghandler As Long
Dim fart As Variant
Dim eda_err As Variant
Dim tex As Integer
Dim Newnum As Variant
Dim node As String * 15
Dim tags As String * 15
Dim Field As String * 15
Dim answer As String * 80
Dim ans As String * 10
Dim defnode As Long
Dim trap As String * 10
Dim texts As String * 6
Dim Nodetyp As String * 20
Dim nodes As String * 20
Dim bidrotenable As Variant
Dim BlockTyp As String * 20
Dim Bidoperatestate As Variant
Dim Bidnodetext As String * 10
Dim bidtokenstate As Variant
Dim Tokenstate As Integer
Dim BidOpstate As Variant

' functions specific to VB (not standard eda fcns)
Declare Sub eda_version Lib "vdba.dll" (ByVal v$)
Declare Sub eda_get1_ascii Lib "vdba.dll" (ByVal n$, ByVal t$, ByVal f$, ByVal v
$)
Declare Function eda_set1_ascii% Lib "vdba.dll" (ByVal n$, ByVal t$, ByVal f$, B
yVal v$)
'
' Note define_group returns a long
'
Declare Function eda_define_group& Lib "vdba.dll" (ByVal Count%, ByVal Detect%)
Declare Sub eda_delete_group Lib "vdba.dll" (ByVal Handle&)
'
' should always pass 0 for vsp
Declare Function eda_define_ntf& Lib "vdba.dll" (ByVal Handle&, ByVal enode$, By
Val etag$, ByVal efield$, ByVal vsp&)
Declare Sub eda_delete_ntf Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
'
Declare Sub eda_lookup Lib "vdba.dll" (ByVal Handle&)
Declare Sub eda_read Lib "vdba.dll" (ByVal Handle&)
Declare Sub eda_write Lib "vdba.dll" (ByVal Handle&)
Declare Function eda_write1% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
Declare Sub eda_wait Lib "vdba.dll" (ByVal Handle&)
Declare Function eda_query_wait% Lib "vdba.dll" (ByVal Handle&)

Declare Function eda_get_error% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
Declare Function eda_get_ascii% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, ByVal
 Value$, ByVal maxlen%)
Declare Function eda_set_ascii% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, ByVal
```

```
 Value$)

' note pointer to float is passed
Declare Function eda_get_float% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, Val
!)
Declare Function eda_set_float% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, Valu
!)

Dim Thisconfig(4) As whatmachinesarewhere

Sub Command3_Click ()
Timer1.Enabled = True
End Sub

Sub Command4_Click ()
Timer1.Enabled = False
End Sub

Sub sendinfo (n, t, f, Text)
Ghandler = eda_define_group(1, 0)
defnode = eda_define_ntf(Ghandler, n, t, f, 0)

Call eda_lookup(Ghandler)
Call eda_wait(Ghandler)
eda_err = eda_get_error(Ghandler, defnode)
Text5.Text = Str$(eda_err)
Newnum = Text
fart = eda_set_ascii(Ghandler, defnode, Newnum)
Call eda_write(Ghandler)
Call eda_wait(Ghandler)
Call eda_delete_group(Ghandler)

End Sub

Sub Command1_Click ()
Load Define_Processes
Define_Processes.Show
End Sub

Sub Command2_Click ()
Define_Tools.Show
End Sub

Sub ConF_Fact_Click ()
Config_Factory.Show
End Sub

Sub Define_Job_Click ()
Load Def_Job
Def_Job.Show
Timer1.Enabled = False
End Sub

Sub Form_Load ()
Dim CurrQueue As Factqueue
Dim i As Integer
```

```
:ACTRUN.FRM - 3

)im Filenum As Integer
)im Recordlength As Integer
Dim Machcode As Single
Dim Filenam As String
)im j As Integer
Dim Nodename As String * 10
Dim BlockTyp As String * 12
)im Field As String * 5
Dim NodeConfig As whatmachinesarewhere
Dim Machinecode As Single

Dim WhatNodesConfigured As Activenodes
Dim Noderecords() As Activenodes

.oad Queuestat
Load Tool_Avle

'Queuestat.Show
'When this form loads we must open the Factory Controller Job Queue and load it
.nto memory
 so that it can be accessed as required, it has been decided to conmstruct the f
:ctory Controller
'in the form of a infinite variable array.

'Free up a number for the file
''filenum = FreeFile
 define filename
''Filenam = "C:\Factory\Queue\Queuejob.que"
'Get the array record length
 'Recordlength = Len(CurrQueue)
'Get the files length
''QFileslength = GetFileslength6(Trim(Filenam), CurrQueue, ThisQueue(), filenum)

 'Text1.Text = QFileslength

 Read the contents of the file to memory
 'Call WriteToMemory6(filenum, QFileslength, ThisQueue())


 'Close filenum

 we are required to reconfigure the Factory according to our records of our node
 configurations

 firstly we have to open a file which tells us the names of the nodes that have
 already been
'configured because we have to know which configuration file data we have to sen
 .
.ilenum = FreeFile

Filenam = "C:\Factory\Config\Cells\Whatnode.sys"

'open the file that records what nodes have been configured
)pen Filenam For Random Access Read As Filenum Len = Len(WhatNodesConfigured)
```

```
ReDim Noderecords(1)'dimension array to take the first record only

Get Filenum, 1, Noderecords(1) 'get the first record to ascertain the files len
th

'MsgBox (CStr(Noderecords(1).Filelength))
If Noderecords(1).Filelength = 0 Then 'there is nothing to be added so exit the
event
    Exit Sub
    Else 'there is something to be transmitted to the Cell Controllers so get i
.

        'redimension the array to take all the records that are in the file
        ReDim Preserve Noderecords(Noderecords(1).Filelength)

        'get all the numbers of all the nodes that have been configured
        For i = 1 To Noderecords(1).Filelength
            Get Filenum, i, Noderecords(i)
    '         MsgBox (CStr(Noderecords(i).CellNodeNum))
        Next

        Close Filenum
        'now that we have got the names of all the nodes that have been configu
ed we can open
        'up the relevant configuration file and retrieve the configuration data
and send the data
        'to the relevant node.


        For i = 1 To Noderecords(1).Filelength
            Filenum = FreeFile
            Filenam = "C:\Factory\Config\Cells\C" & CStr(i) & "Confg.sys"
     '        MsgBox (Filenam)
            'open cell 1's configuration file
            Open Filenam For Random Access Read As Filenum Len = Len(NodeConfig

            For j = 1 To 4
                'Type whatmachinesarewhere
                'Filelength As Integer
                'Nodenumber As Integer
                'MachType As Single
                'Enabled as Integer
                'End Type
                Get Filenum, j, Thisconfig(j)
       '         MsgBox (CStr(ThisConfig(j).Machtype))

                'If ThisConfig(j).Machtype <> 0 Then 'the node has been configur
ed

                'End If
                If Thisconfig(j).Enabled = 1 Then
                    Machinecode = Thisconfig(j).MachType
                    Else
                        Machinecode = 0
                End If
```

```
                BlockTyp = "V" & CStr(j) & "MACHTYP"
                Field = "A_CV"

                If i >= 10 Then
                    Nodename = "C" & CStr(i)
                        Else
                        Nodename = "C0" & CStr(i)
                End If

                Call sendinfo(Trim(Nodename), Trim(BlockTyp), Trim(Field), Mach.
iecode)
                'so send the appropriate configuration data.

            Next
        Next
End If
Close Filenum

'Type Activenodes
'Filelength As Integer
 CellnodeNum As Integer
'End Type


End Sub

ub Form_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Singl
e)
On Error Resume Next
 xit Sub
nload System_Form
End Sub

Sub Form_Unload (Cancel As Integer)


 set a variable to tell other forms that the Factory Controller is unloading so
 hey will unload
 rather than just hide
 actoryIsUnloading = 1

 nload Queuestat

 nload Tool_Avle

 nload Res_Form

nd Sub

Sub Sys_parameters_Click ()
System_Form.Show
 nd Sub

Sub Timer1_Timer ()
 This little application runs when the factory controller software is first load
ed
```

```
'and runs at variable intervals after that as defined by the user

'This application is fired at the beginning each day and then every minute and
t's
'function is to look at the '"Jobs pending" file and compare todays date and ti
e with the
'release date and time of each job in that file.It was thought that it would be
best to run this
'program every 1 minute to as the sensitivity of release date setting for the s
stem is
' down to the nearest minute.

'First open the "pending jobs" file
'This button allows the user to queue the current job providing all tasks have
een defined

Dim Thistask As Jobrecord
Dim Filenum As Integer
Dim Filenums As Integer
Dim Filenam As String * 50
Dim ThisFileslength As Integer
Dim i As Integer
Dim ThisFilesname As String * 60
Dim Reply As Integer
Dim JobSpending() As Pendingjob
Dim Thisjob As Pendingjob
Dim Todaysdate As String * 10
Dim Currenttime As String * 5
Dim typ As Integer
Dim ThisString As String
Dim ThatString As String
Dim j As Integer
Dim s As Integer
Dim Removedit() As String * 30

'Define jobfile variables
Dim readtask As Jobrecord
Dim Recordlength As Integer
Dim Taskposition As Integer
Dim Loadtasks() As Jobrecord

'Define Factory Job Queueing variables

Dim FactQnum As Integer
Dim qFilename As String * 40
Dim Recordleng As Integer
Dim TempQFileslength As Integer
Dim Skip As Integer
Dim Thisqueue()  As Factqueue
Dim CurrQueue As Factqueue
Dim CounTTheLaps As Integer



'define config sys variables
Dim Scantime As String
```

```
On Error Resume Next

Timer1.Enabled = False

'The timer can only have intervals of 1 minute max so use a global counter that
counts to the value
'entered by the user in the system configuration stage

'therefore open the file and get the value to which the counter must be set as
efined by the user
Filenum = FreeFile
Open "C:\Factory\System\Config\Scantime.sys" For Input As Filenum Len = 4
Line Input #Filenum, Scantime
'MsgBox ("Scan Time is " & Scantime)
Close Filenum

Counttoscantime = Counttoscantime + 1

If Counttoscantime = Val(Scantime) Then 'execute this procedure

'     MsgBox ("Yes we are scanning")
    Timer1.Enabled = False

    Firsttime = Firsttime + 1

    ThisFilesname = "C:\Factory\Queue\jobspend.jpf"
    Filenum = FreeFile

    'get the length of the file
    ThisFileslength = getfilesLength5(ThisFilesname, Thisjob, JobSpending(), Fi
enum)

    If ThisFileslength = 0 Then
        Close Filenum
        Timer1.Enabled = True
        Counttoscantime = 0
        Exit Sub
    End If

    'Now write the contents of the file to memory.
    Call WriteToMemory7(Filenum, ThisFileslength, JobSpending())


    'Now write the contents of the file to the phantom listbox if this form has
ust been loaded
    'the phantom listbox is used as a file handling device
    List2.Clear



    For i = 1 To ThisFileslength
        CounTTheLaps = CounTTheLaps + 1

        If CounTTheLaps = 3 Then
            DoEvents
```

```
            CounTTheLaps = 0
        End If

        List2.AddItem JobSpending(i).Jobname
    Next

    Close Filenum

    ReDim Removedit(ThisFileslength)

    'Now we want to look for a match between the dates in the file and todays d
te and the current
    'time and the job release time to see if there is a match - if there is we
ill queue the job's first task.
    Todaysdate = Date$
    Currenttime = Time$

    For i = 1 To ThisFileslength
        Filenums = FreeFile
        Recordlength = Len(readtask)
        'open each job's job file and examine the due date
        Filenam = "R:\Factory\jobfiles\" & Trim(List2.List(i - 1)) & ".iff"

    '     MsgBox (Filenam)
        ReDim Loadtasks(1)
        Open Trim(Filenam) For Random Access Read Shared As Filenums Len = Rec
dlength
        'Text1.Text = List2.List(i - 1)
        ' Get the due date of the job from file
        Get Filenums, 1, Loadtasks(1)

            If Len(Trim(Loadtasks(1).Duedate)) = 0 Then

                GoTo Nextstop
            End If


        'Check for a match

        If InStr(Trim(Loadtasks(1).Duedate), Trim(Todaysdate)) = 1 And Trim(Loa
tasks(1).Starttime) Like Currenttime Then
            'both the jobs start date and starttime matches todays date and tim
 so put it in the queue

            'update the queueing listbox
            'Check to see if the job is urgent and if it is put
            'it at the top of the listbox
            Removedit(i) = List2.List(i - 1)

            If Loadtasks(1).Force = 1 Then
                Queuestat.QueueList.AddItem Loadtasks(1).Jobname, 0


                    Else 'it is not urgent so add it at the end
                        Queuestat.QueueList.AddItem Loadtasks(1).Jobname 'Queue
tat.Jobspend.ListCount - 1
```

```
            End If
            Else
        Removedit(i) = ""

        End If

        Close Filenums
Nextstop:
    Next

    'Now we must alter the listbox to reflect the changes made
    For i = 1 To ThisFileslength
        For j = 1 To ThisFileslength
            If InStr(Trim(List2.List(i - 1)), Trim(Removedit(j))) = 1 And Len(T
rim(Removedit(j))) > 0 Then 'the strings match so del
                List2.RemoveItem (i - 1)
            End If
        Next
    Next

    'Now that we have altered this listbox that displays
    'whats in the queue we must retain it via a file

    'define filename
    qFilename = "C:\Factory\Queue\Queuejob.que"

    'Kill the old file
    Kill qFilename

    'Open up a new copy of the file and save modified
    'list contents to it.

    'Free up a number for the file
    FactQnum = FreeFile

    'Get the array record length
    Recordleng = Len(CurrQueue)

    'Redimension the array to hold all the list content
    ReDim Thisqueue(Queuestat.QueueList.ListCount)

    'Open the file
    Open qFilename For Random Access Read Write Shared As FactQnum Len = Recordl
eng

    For i = 1 To Queuestat.QueueList.ListCount
        Get FactQnum, i, Thisqueue(i)
    Next


    Thisqueue(1).Filelength = Queuestat.QueueList.ListCount


    For s = 1 To (Queuestat.QueueList.ListCount)
        Thisqueue(s).Jobname = Trim(Queuestat.QueueList.List(s - 1))
    Next
```

```
    For s = 1 To (Queuestat.QueueList.ListCount)
        Put FactQnum, s, Thisqueue(s)
    Next

    Close FactQnum



'update jobspending list
Queuestat.Jobspend.Clear

For i = 1 To List2.ListCount
    Queuestat.Jobspend.AddItem List2.List(i - 1)
Next


'Next
'Now we must alter the job's pending file to reflect the fact that the job's ha
e been removed
'queued
'The contents of the "Phantom List" is the status of the new jobs pending file
'so we must write the contents of the listbox to the file but first close the o
dfile and then
'kill it


Kill "C:\Factory\Queue\jobspend.jpf"

'Now open the file again and write new contents into it
ThisFilesname = "C:\Factory\Queue\jobspend.jpf"
Filenum = FreeFile
ReDim JobSpending(List2.ListCount)

Open ThisFilesname For Random Access Read Write Shared As Filenum Len = Len(This
job)

For i = 1 To List2.ListCount
    Get Filenum, i, JobSpending(i)
Next


JobSpending(1).Filelength = List2.ListCount

For i = 1 To List2.ListCount
    JobSpending(i).Jobname = List2.List(i - 1)
    Put Filenum, i, JobSpending(i)
Next

Close Filenum

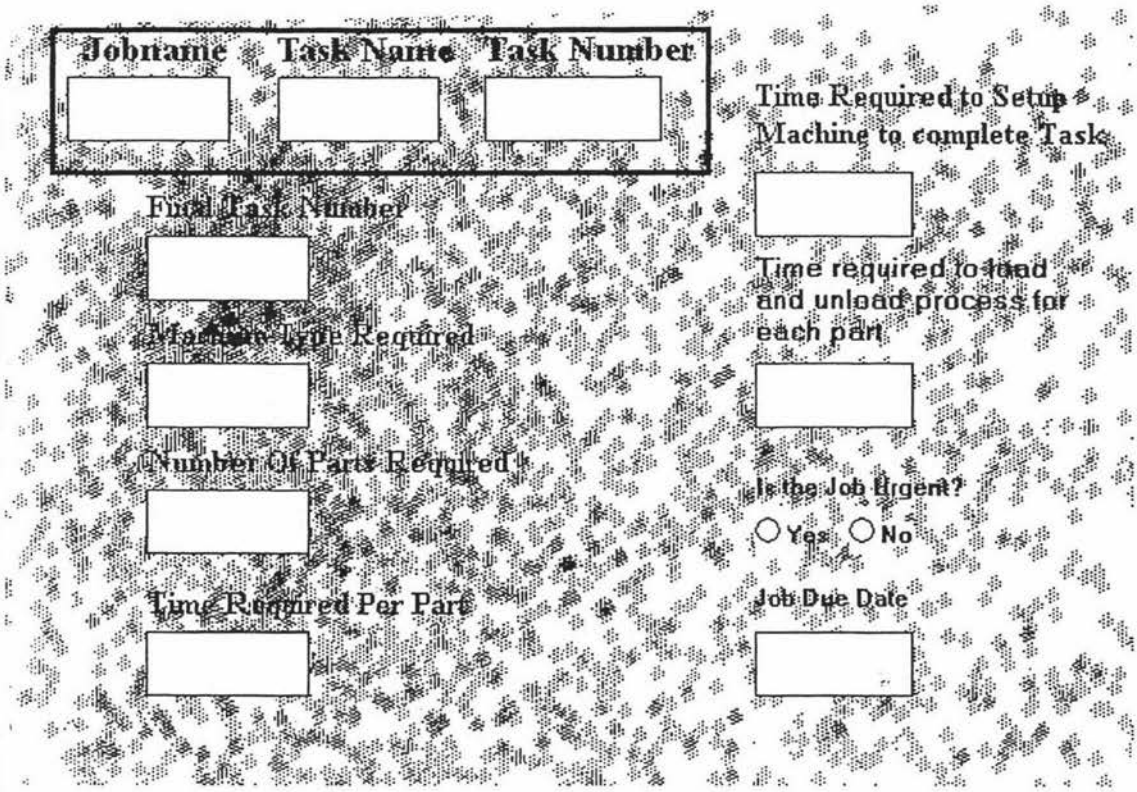

'Type Pendingjob
'Jobname As String * 13
'Filelength As Integer
'End Type
```

```
Counttoscantime = 0
End If
Timer1.Enabled = True
End Sub

Sub View_Tool_Avle_Click ()
Tool_Avle.Show
End Sub

Sub ViewQueue_Click ()
Queuestat.Show
Queuestat.WindowState = 2
'QueueStat.SetFocus

End Sub
```


Jobname
Task Name
Task Number
Final Task Number
Machine Type Required
Number Of Parts Required
Time Required Per Part
Time Required to Setup Machine to complete Task
Time required to load and unload process for each part
Is the Job Urgent?
Yes
No
Job Due Date

```
JOB_STAT.FRM - 1

Option Explicit

Sub Form_Load ()
'This event allows the user to open up the selected file and to give details
'regarding what stage the job is currently at.
'On Error Resume Next

On Error Resume Next
'Define jobfile variables
Dim Readtask As JobRecord
Dim Recordlength As Integer
Dim Taskposition As Integer
Dim Loadtasks() As JobRecord
Dim CurrentJob_Status As Integer
Dim Filenum As Integer
Dim Filenam As String

Filenum = FreeFile
Recordlength = Len(Readtask)
'open each job's job file and examine the due date
Filenam = Trim(JobFilepath) & Trim(QueueStat.QueueList.List(QueueStat.QueueList
ListIndex)) & ".iff"
CurrentJob_Status = JobStatus(Trim(Filenam), Readtask, Loadtasks(), Filenum)

Call Writetomemory4(Filenum, CurrentJob_Status, Loadtasks())
Text1.Text = Loadtasks(CurrentJob_Status).Jobname
Text2.Text = Loadtasks(CurrentJob_Status).FinalTask
Text3.Text = Loadtasks(CurrentJob_Status).Taskname
Text4.Text = Loadtasks(CurrentJob_Status).NooFCycles & " Cycles"
Text5.Text = Loadtasks(CurrentJob_Status).EstCycletime & " Minutes"
Text6.Text = Loadtasks(CurrentJob_Status).setuptime & " Minutes"
Text7.Text = Loadtasks(CurrentJob_Status).Estloadunloadtime & " Minutes"
Text8.Text = Loadtasks(CurrentJob_Status).Duedate
Text9.Text = Loadtasks(CurrentJob_Status).Taskname
Text10.Text = Loadtasks(CurrentJob_Status).Status

If Loadtasks(CurrentJob_Status).Force = 1 Then
    Option1.Value = True
Else
    Option2.Value = True
End If

'Disable both option buttons as they are for display only
Option1.Enabled = False
Option2.Enabled = False



Close Filenum

End Sub
```

```
'ACHSUP.FRM - 1

Option Explicit



Sub Form_Load ()
Dim savemode As Integer
On Error Resume Next

PicClip1.Picture = Clipboard.GetData(2)
'save the current scalemode for the picture box
savemode = Picture3.ScaleMode

'get x and y coordinates of the clipping region

PicClip1.ClipX = 100
PicClip1.ClipY = 35

'set the area of the clipping region in pixels

PicClip1.ClipHeight = 450
PicClip1.ClipWidth = 700

'set the picture box scalemode to pixels
Picture3.ScaleMode = 3

'set the destination area to fill the picture box

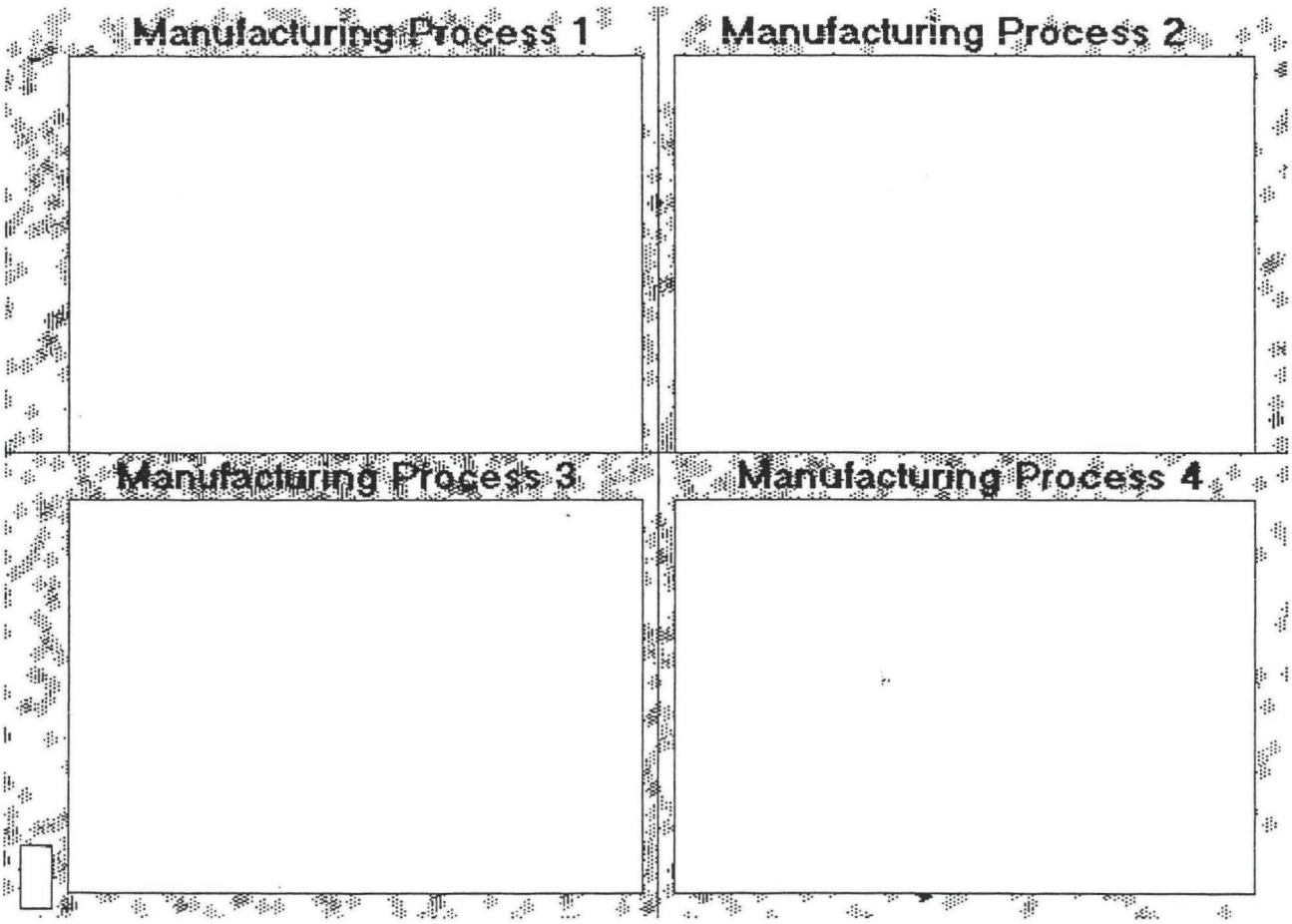
PicClip1.StretchX = Picture3.ScaleWidth
PicClip1.StretchY = Picture3.ScaleHeight

'assign the clipped bitmap to the picture box
Picture3.Picture = PicClip1.Clip

Picture3.ScaleMode = savemode

SavePicture Picture3.Picture, "C:\factory\JobFiles\Setupin\" & Trim(SetUpInsfnam
e) & ".bmp"

Unload Machsup
End Sub
```


Manufacturing Process 1
Manufacturing Process 2
Manufacturing Process 3
Manufacturing Process 4

```
NODE_DEF.FRM - 1

Option Explicit
Dim Vmd1FirstClick As Integer
Dim Thisconfig(4) As whatmachinesarewhere

Dim Ghandler As Long
Dim fart As Variant
Dim eda_err As Variant
Dim tex As Integer
Dim Newnum As Variant
Dim node As String * 15
Dim tags As String * 15
Dim field As String * 15
Dim answer As String * 80
Dim ans As String * 10
Dim defnode As Long
Dim trap As String * 10
Dim texts As String * 6
Dim Nodetyp As String * 20
Dim nodes As String * 20
Dim bidrotenable As Variant
Dim BlockTyp As String * 20
Dim Bidoperatestate As Variant
Dim Bidnodetext As String * 10
Dim bidtokenstate As Variant
Dim Tokenstate As Integer
Dim BidOpstate As Variant


' functions specific to VB (not standard eda fcns)
Declare Sub eda_version Lib "vdba.dll" (ByVal v$)
Declare Sub eda_get1_ascii Lib "vdba.dll" (ByVal n$, ByVal t$, ByVal f$, ByVal v
$)
Declare Function eda_set1_ascii% Lib "vdba.dll" (ByVal n$, ByVal t$, ByVal f$, B
yVal v$)
'
' Note define_group returns a long
'
Declare Function eda_define_group& Lib "vdba.dll" (ByVal Count%, ByVal Detect%)
Declare Sub eda_delete_group Lib "vdba.dll" (ByVal Handle&)

' should always pass 0 for vsp
Declare Function eda_define_ntf& Lib "vdba.dll" (ByVal Handle&, ByVal enode$, By
Val etag$, ByVal efield$, ByVal vsp&)
Declare Sub eda_delete_ntf Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
'
Declare Sub eda_lookup Lib "vdba.dll" (ByVal Handle&)
Declare Sub eda_read Lib "vdba.dll" (ByVal Handle&)
Declare Sub eda_write Lib "vdba.dll" (ByVal Handle&)
Declare Function eda_write1% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
Declare Sub eda_wait Lib "vdba.dll" (ByVal Handle&)
Declare Function eda_query_wait% Lib "vdba.dll" (ByVal Handle&)

Declare Function eda_get_error% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
Declare Function eda_get_ascii% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, ByVal
 Value$, ByVal maxlen%)
Declare Function eda_set_ascii% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, ByVal
```

```
 Value$)

' note pointer to float is passed
Declare Function eda_get_float% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, Val
!)
Declare Function eda_set_float% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, Valu
!)

Sub Command1_Click ()
Unload Node_Def
End Sub

Sub sendinfo (n, t, f, Text)
Ghandler = eda_define_group(1, 0)
defnode = eda_define_ntf(Ghandler, n, t, f, 0)

Call eda_lookup(Ghandler)
Call eda_wait(Ghandler)
eda_err = eda_get_error(Ghandler, defnode)
Text5.Text = Str$(eda_err)
Newnum = Text
fart = eda_set_ascii(Ghandler, defnode, Newnum)
Call eda_write(Ghandler)
Call eda_wait(Ghandler)
Call eda_delete_group(Ghandler)

End Sub

Sub Disable1_Click (index As Integer)
'this particular event procedure allows the user to disable a particular VMD sc
that
'it not allowed to participate in the following:



'** IT IS NOT ALLOWED TO PARTICIPATE IN BIDS
Dim CurrQueue As Factqueue
Dim i As Integer
Dim Filenum As Integer
Dim Recordlength As Integer
Dim Machcode As Single
Dim Filenam As String
Dim j As Integer
Dim Nodename As String * 10
Dim BlockTyp As String * 12
Dim field As String * 5
Dim NodeConfig As whatmachinesarewhere



Filenum = FreeFile
Filenam = "C:\Factory\Config\Cells\C" & CStr(NodebeingConfiged) & "Confg.sys"
Open Filenam For Random Access Read Write As Filenum Len = Len(NodeConfig)
```

```
NODE_DEF.FRM - 3

For j = 1 To 4
    'Type whatmachinesarewhere
    'Filelength As Integer
    'Nodenumber As Integer
    'MachType As Single
    'enabled as Integer
    'End Type
Get Filenum, j, Thisconfig(j)

Next

 now find the position at which the match occurs between this VMD Number and tha
t in the file
For j = 1 To 4
    If Thisconfig(j).Nodenumber = (index + 1) Then
        'a match has been found therefore change the enabled field to "0" to dis
able
        Thisconfig(j).Enabled = 0
    End If
Next

For j = 1 To 4
    Put Filenum, j, Thisconfig(j)
Next

Close Filenum

'now send the new configuration to the Cell Controller

BlockTyp = "V" & CStr(index + 1) & "MACHTYP"
field = "A_CV"
Nodename = Thisnodenam
Call sendinfo(Trim(Nodename), Trim(BlockTyp), Trim(field), 0)


End Sub

Sub Enable1_Click (index As Integer)
 this particular event procedure allows the user to enable a particular VMD so t
hat
'it  allowed to participate in the following:

On Error Resume Next

 ** IT IS ALLOWED TO PARTICIPATE IN BIDS
Dim CurrQueue As Factqueue
Dim i As Integer
Dim Filenum As Integer
Dim Recordlength As Integer
Dim Machcode As Single
Dim Filenam As String
Dim j As Integer
Dim Nodename As String * 10
Dim BlockTyp As String * 12
Dim field As String * 5
Dim NodeConfig As whatmachinesarewhere
```

```
Dim Matchposition As Integer
Dim Machinetype As Single


Filenum = FreeFile
Filenam = "C:\Factory\Config\Cells\C" & CStr(NodebeingConfiged) & "Confg.sys"
MsgBox (Filenam)
Open Filenam For Random Access Read Write As Filenum Len = Len(NodeConfig)
' MsgBox (CStr(index))
For j = 1 To 4
    'Type whatmachinesarewhere
    'Filelength As Integer
    'Nodenumber As Integer
    'MachType As Single
    'enabled as Integer
    'End Type
Get Filenum, j, Thisconfig(j)

Next

'MsgBox CStr(Thisconfig(2).Nodenumber)
'MsgBox CStr(Thisconfig(2).Machtype)
'now find the position at which the match occurs between this VMD Number and th
t in the file

For j = 1 To 4
    If Thisconfig(j).Nodenumber = (index + 1) Then
        'a match has been found therefore change the enabled field to "0" to di
able
        Matchposition = j
    End If
Next


'now change the value of the enabled variable at the position "matchposition"

Thisconfig(Matchposition).Enabled = 1


'now send the new configuration to the Cell Controller

BlockTyp = "V" & CStr(index + 1) & "MACHTYP"
field = "A_CV"
Nodename = Thisnodenam
Machinetype = Thisconfig(Matchposition).Machtype
'MsgBox (CStr(Machinetype))
Call sendinfo(Trim(Nodename), Trim(BlockTyp), field, Machinetype)

For j = 1 To 4
    Put Filenum, j, Thisconfig(j)
Next

Close Filenum
```

```
'ODE_DEF.FRM - 5


:nd Sub

Sub Form_Load ()
'this form allows the user the chance to view what machine or processes
'are associated with each VMD/Node configuration and then He/She can view
'additional information about it if required.

)im Filenum As Integer
)im Reply As String
)im Filenam  As String
)im Thisstring As String
Dim ThisIndex As Integer
)im Filenumb As Integer
)im Thisfilenam As String
Dim Again As Integer

)n Error Resume Next


 The first thing we are required to do is to open up a series of files that cont
ain the
'name of the bitmap associated with each of this Cells VMD's and then project th
)se
'bitmaps into the correct ole controls.

 THE BEAUTY OF THE SYSTEM IS THAT IF A VMD IS NOT BEING CONFIGURED IE: THE USER
)NLY WANTS
'TO USE 2 VMD'S THEN THE SYSTEM IS NOT DISRUPTED AND HE CAN DO THIS.

'**********************CARRY OUT FOR FIRST VMD*****************************

'ilenum = FreeFile

'Specify the machine configuration filename for the first VMD
'ilenam = "C:\Factory\System\V1" & "C" & CStr(NodebeingConfiged) & "Mach.sys"

'Open that file
pen Filenam For Input As Filenum Len = 100

 Read the contents of the file but only store the last variable's value
 o Until EOF(Filenum)
 f Again = 5 Then 'this again variable is to prevent the system going into a nev
)r ending
    Again = 0     'loop when the file is empty as it never reaches the end.
    Close Filenum
    Exit Sub
'nd If

 get the name of the process configured for VMD 1 on this cell
'ine Input #Filenum, Thisstring
    Again = Again + 1
Loop


Filenumb = FreeFile
```

```
'trim off the "  " on the name of the process being dealt with
Thisstring = Trim(Mid(Thisstring, 2, (Len(Thisstring)) - 2))

If Len(Trim(Thisstring)) = 0 Then
    Close Filenum
    GoTo TryNext2
End If

'define the name of the bitmap file to open using the name of the process as it
s part.
Thisfilenam = "C:\Factory\Process\Specs\" & Thisstring & ".bmp"

Open Trim(Thisfilenam) For Binary As #Filenumb 'open the bitmap file
Ole1.FileNumber = Filenumb 'define the filenumber
Ole1.Action = 12  'read the bitmap into the OLE.
Close Filenumb 'Close the bitmap file

'close the VMD configuration file.
Close Filenum

TryNext2:

Again = 0 'Reintialise again for next VMD.

'*************************CARRY OUT FOR SECOND VMD*****************************
*
Filenum = FreeFile

'Specify the machine configuration filename for the first VMD
Filenam = "C:\Factory\System\V2" & "C" & CStr(NodebeingConfiged) & "Mach.sys"

'Open that file
Open Filenam For Input As Filenum Len = 100

'Read the contents of the file but only store the last variable's value
Do Until EOF(Filenum)
If Again = 5 Then 'this again variable is to prevent the system going into a ne
er ending
    Again = 0     'loop when the file is empty as it never reaches the end.
    Close Filenum
    Exit Sub
End If

'get the name of the process configured for VMD 1 on this cell
Line Input #Filenum, Thisstring
    Again = Again + 1
Loop


Filenumb = FreeFile

'trim off the "  " on the name of the process being dealt with
Thisstring = Trim(Mid(Thisstring, 2, (Len(Thisstring)) - 2))

If Len(Trim(Thisstring)) = 0 Then
```

```
    Close Filenum
    GoTo Trynext3
End If


'define the name of the bitmap file to open using the name of the process as it
s part.
Thisfilenam = "C:\Factory\Process\Specs\" & Thisstring & ".bmp"

Open Trim(Thisfilenam) For Binary As #Filenumb 'open the bitmap file
Ole2.FileNumber = Filenumb 'define the filenumber
Ole2.Action = 12  'read the bitmap into the OLE.
Close Filenumb 'Close the bitmap file

'close the VMD configuration file.
Close Filenum


Trynext3:

Again = 0 'Reintialise again for next VMD.
'************************CARRY OUT FOR THIRD VMD*******************************
Filenum = FreeFile

'Specify the machine configuration filename for the first VMD
Filenam = "C:\Factory\System\V3" & "C" & CStr(NodebeingConfiged) & "Mach.sys"

'Open that file
Open Filenam For Input As Filenum Len = 100

'Read the contents of the file but only store the last variable's value
Do Until EOF(Filenum)
If Again = 5 Then 'this again variable is to prevent the system going into a nev
er ending
    Again = 0     'loop when the file is empty as it never reaches the end.
    Close Filenum
    Exit Sub
End If

'get the name of the process configured for VMD 1 on this cell
Line Input #Filenum, Thisstring
    Again = Again + 1
Loop


Filenumb = FreeFile

'trim off the "  " on the name of the process being dealt with
Thisstring = Trim(Mid(Thisstring, 2, (Len(Thisstring)) - 2))

If Len(Trim(Thisstring)) = 0 Then
    Close Filenum
    GoTo Trynext4
End If
```

```
'define the name of the bitmap file to open using the name of the process as it
s part.
Thisfilenam = "C:\Factory\Process\Specs\" & Thisstring & ".bmp"

Open Trim(Thisfilenam) For Binary As #Filenumb 'open the bitmap file
Ole3.FileNumber = Filenumb 'define the filenumber
Ole3.Action = 12  'read the bitmap into the OLE.
Close Filenumb 'Close the bitmap file

'close the VMD configuration file.
Close Filenum


Trynext4:

Again = 0 'Reintialise again for next VMD.

'************************CARRY OUT FOR FORTH VMD*****************************
Filenum = FreeFile

'Specify the machine configuration filename for the first VMD
Filenam = "C:\Factory\System\V4" & "C" & CStr(NodebeingConfiged) & "Mach.sys"

'Open that file
Open Filenam For Input As Filenum Len = 100

'Read the contents of the file but only store the last variable's value
Do Until EOF(Filenum)
If Again = 5 Then 'this again variable is to prevent the system going into a ne
er ending
    Again = 0     'loop when the file is empty as it never reaches the end.
    Close Filenum
    Exit Sub
End If

'get the name of the process configured for VMD 1 on this cell
Line Input #Filenum, Thisstring
    Again = Again + 1
Loop


Filenumb = FreeFile

'trim off the "  " on the name of the process being dealt with
Thisstring = Trim(Mid(Thisstring, 2, (Len(Thisstring)) - 2))

'define the name of the bitmap file to open using the name of the process as it'
s part.
Thisfilenam = "C:\Factory\Process\Specs\" & Thisstring & ".bmp"

Open Trim(Thisfilenam) For Binary As #Filenumb 'open the bitmap file
Ole4.FileNumber = Filenumb 'define the filenumber
Ole4.Action = 12  'read the bitmap into the OLE.
Close Filenumb 'Close the bitmap file

'close the VMD configuration file.
```

```
ODE_DEF.FRM - 9

Close Filenum

Again = 0


End Sub

ub NewVmd1_Click ()
heVMD = 1
Config_newVMD1.Show
nd Sub

Sub Newvmd2_Click ()
im NewVMD As New Config_newVMD1

TheVMD = 2
ewVMD.Show
ewVMD.Caption = "Configure VMD2"


nd Sub

ub NewVMD3_Click ()
im NewVMD As New Config_newVMD1

heVMD = 3
ewVMD.Show
ewVMD.Caption = "Configure VMD3"


nd Sub

ub Newvmd4_Click ()
im NewVMD As New Config_newVMD1

eVMD = 4
ewVMD.Show
ewVMD.Caption = "Configure VMD4"


nd Sub

b OLE1_Click ()
f Vmd1FirstClick = 2 Then
    Vmd1FirstClick = 1
    OLE1_DblClick
    Exit Sub
d If
gBox "he"
md1FirstClick = 2
d Sub

ub OLE1_DblClick ()
im Filenum As Integer
m Reply As String
```

```
NODE_DEF.FRM - 10

On Error Resume Next

Filenum = FreeFile

'Reply = InputBox("Enter the Filename you want to open.", "Enter Filename", " "
Open "C:\Factory\Process\Specs\Mill.bmp" For Binary As #Filenum
Ole1.FileNumber = Filenum
Ole1.Action = 12

Close Filenum
End Sub

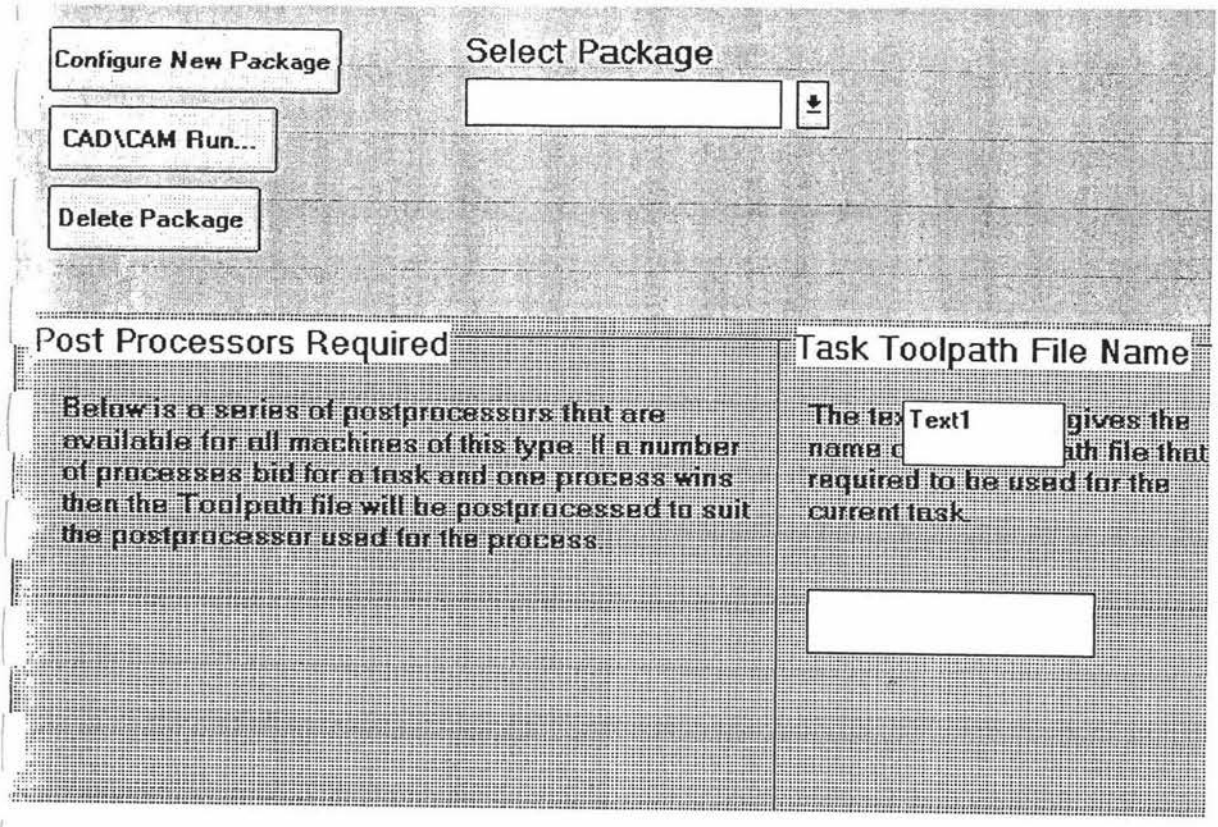
Sub Reconfigure1_Click ()
NewVmd1_Click

End Sub

Sub Reconfigure2_Click ()
Newvmd2_Click
End Sub

Sub Reconfigure3_Click ()
NewVMD3_Click
End Sub

Sub reconfigure4_Click ()
Newvmd4_Click
End Sub
```


Configure New Package
Select Package
CAD\CAM Run...
Delete Package
Post Processors Required
Below is a series of postprocessors that are available for all machines of this type. If a number of processes bid for a task and one process wins then the Toolpath file will be postprocessed to suit the postprocessor used for the process.
Task Toolpath File Name
The te
Text1
gives the
name o
ath file that
required to be used for the
current task.

```
Option Explicit
Option Base 1

Dim CadCamrecord As CadCamvariable
Dim CadCamArray() As CadCamvariable
Dim ThisFileslength As Integer

Sub Cad_Cam_Packages_Click ()
Run_Cad_Cam.Enabled = True
Del_Package.Enabled = True
End Sub

Sub Command1_Click ()
 'This procedure allows the user to give the name of the
'CAD\CAM package that he wants to integrate to the program.
'Once the name has been given this procedure adds that name
'to the listbox on this form and then asks for the fireup
'code.The Fireup code is the command line used in the windows
' icon property for that package eg:for mastercam mill the command
'line is 'Mill5.exe' so when configuring mastercam this should
'be given as the fireup code.Once the fireup string has been given
'the user is then asked for the window focus string - this is the
'caption that appears at the top of the windows application just fired
'and is used by Visual Basic to change the focus from Visual basic to
'the application just fired - in the case of mastercam this would be
```

```
the application just fired - in the case of mastercam this would be
' 'Mastercam Mill 5.0'.The last piece of data that is required to be given
'by the user is the working directory of the package - this can be gotten
 from the packages startup icon's properties.All this information as stated
'above is then saved to file so that it can be referenced later.Once all this
'data is given then the application can be fired from the system integration
 tool.


 Define localised variables........
Dim Title As String * 20
Dim msg As String
 im CadCamname As String
 im Defval As String * 20
Dim Firestring As String
 im filenum As Integer
 im ThisFilesname As String
Dim AppFocus As String
 im WorkDir As String * 20
 im thiserror As Integer

 n Error GoTo CheckTheseErrors
 firstly we must disable the 'CAD|CAM Run' button
un_Cad_Cam.Enabled = False


'Now we ask the user to give the name of the CAD\CAM package
 that they want to use with the system integration tool.

'Initialise the msg variable.
 sg = "Please enter the name of the Cad\Cam package that "
```

```
isg = msg & "you want to use with " & SystemName & ". "
:itle = "Configure Cad\Cam"
)efval = ""
:adCamname = InputBox(msg, Title, Defval)

'check to see if any reply was given
If Len(CadCamname) = 0 Then
    Exit Sub  'no reply was given so don't continue
        Else 'a valid answer was given therefore continue.
            'ask for the name of the title in the window
            'that appears when the application is first fired.
            'This title is used to fire that application in
            'the first place.
            msg = "Please give the name of the executable file(The Command Line
"
            msg = msg & " associated with the Cad\Cam package that you are"
            msg = msg & " configuring. This is available by examining the prope
ties"
            msg = msg & " of the icon used to normally fire the application, if
no icon "
            msg = msg & "exists then it is simply the name of the executable fi
e."
            Title = "Request Cad\Cam window Title"
no icon "
            msg = msg & "exists then it is simply the name of the executable fil
e."
            Title = "Request Cad\Cam window Title"
            Defval = "*.exe"
            Firestring = InputBox(msg, Title, Defval)

            'Check to see if a valid answer was given.
            If Len(Firestring) = 0 Then
            'answer was invalid
                Exit Sub
                    Else
                        'A valid answer was given therefore ask the user to
                        'give the 'getFocus' string.The 'getfocus' string is
                        'the string that is used by Visual Basic to change
                        'focus from the System Integration Tool to the Cad\Cam
                        'package being used.eg:In the case of mastercam this
                        'would be 'Mastercam mill 5.0'
                        msg = "Please give the Title of the window that first"
                        msg = msg & " appears when you fire up the Cad\Cam appl
cation."
                        Title = "Enter Cad\Cam opening window title."
                        Defval = ""
                        AppFocus = InputBox(msg, Title, Defval)

                            If Len(Trim(AppFocus)) = 0 Then 'no answer given
                            'so exit sub.
                                Exit Sub
                                    Else 'answer relevant so continue
                                        'We must ask the user for the working
                                        'directory of the cad\cam package so
                                        'that when the package is fired from
                                        'this application it knows where it has
                                        'to go to get all the relevant files.
                                        msg = "Please enter " & CadCamname & "'
 working directory. This can be found"
```

```
'CAD_CAM.FRM - 3

 tartup icon properties under windows."
                                        Title = "Enter " & CadCamname & "'s wor
ing directory"
                                        Defval = "c:\windows"
                                        WorkDir = InputBox(msg, Title, Defval)


                                        ThisFilesname = "C:\Factory\System\Cadc
m.sys"
                                        filenum = FreeFile

                                        'get the length of the file
                                        ThisFileslength = getfilesLength3(ThisF
 esname, CadCamrecord, CadCamArray(), filenum)


                                        'now write the contents of the file to r
emory.
                                        Call WriteToMemory2(filenum, ThisFilesle
 gth, CadCamArray())


                                        'now increment the file placeholder by 1
 to reflect
                                        'the fact that another record is going t
 be added.
                                        CadCamArray(1).Recordleng = CadCamArray(
1).Recordleng + 1

                                        'write the new record to memory.
                                        CadCamArray(CadCamArray(1).Recordleng).F
ackageName = Trim(CadCamname)
                                        CadCamArray(CadCamArray(1).Recordleng).F
irecommand = Trim(Firestring)
```

```
                                End If
            End If
:nd If


:xit Sub

:heckTheseErrors:

:hiserror = ChekError(Err)

:xit Sub




End Sub

Sub Del_Package_Click ()

'declare variables
Dim msg As String
Dim Title As String
Dim reply As Integer
Dim typ As Integer
Dim j As Integer
Dim filenum As Integer
Dim Filelength As Integer
Dim recordposition As Integer


'this particular event allows the user to delete individual
'packages that he no longer requires the use of, this saves
'the "cad\cam.sys" file from becoming to cluttered as well
'as the listbox containing the names of all the cad\cam
'packages.

'first as the user whether he is sure that he wants to
'remove the selected package.
msg = "Are you sure that you want to disengage the selected package and it's"
msg = msg & " associated data from the 'System Integration Tool for Advanced"
msg = msg & " Manufacturing Automation'."
typ = MB_ICONQUESTION + MB_YESNO
Title = "Disengage Selected Package?"

'get the answer
reply = MsgBox(msg, typ, Title)

'check to see what the answer is and act accordingly
If reply = IDNO Then
    Exit Sub
```

```
        Else 'We are required to write a piece of code that will take the name
        'of the package that was selected from the listbox and to get a match
        'with the same name as recorded in the 'cadcam.sys' file, this will
        'give the position of the data to be deleted, we then delete the data
        'at that position in the file and rearrange the remaining data about
        'the deleted position.

            'first find the record's position
            For j = 1 To ThisFileslength
                If Trim(CadCamArray(j).PackageName) = Trim(Cad_Cam_Packages.List
Cad_Cam_Packages.ListIndex)) Then
                'then the record deletion position has been found
                recordposition = j
                End If
            Next
            'Remove the item from the listbox
            Cad_Cam_Packages.RemoveItem Cad_Cam_Packages.ListIndex


            'Now that we have the position of the record that we awnt to delete
e must delete this
            'record and then arrange the rest of this file around the deletion p
int.

            For j = recordposition To (ThisFileslength - 1)
                CadCamArray(j).PackageName = CadCamArray(j + 1).PackageName
                CadCamArray(j).Firecommand = CadCamArray(j + 1).Firecommand
                CadCamArray(j).FocusCommand = CadCamArray(j + 1).FocusCommand
                CadCamArray(j).WorkDirectory = CadCamArray(j + 1).WorkDirectory
                'We do not save the recordlength variable from the next positior
down to the current
                'position because this only exists in the first record anyway, a
d doing so would
                'delete the recordlength record if the first packagename was del
eted.
            Next

            CadCamArray(ThisFileslength).PackageName = ""
            CadCamArray(ThisFileslength).Firecommand = ""
            CadCamArray(ThisFileslength).FocusCommand = ""
            CadCamArray(ThisFileslength).WorkDirectory = ""


            'now we must save these changes to file so that they can be retained

            filenum = FreeFile
            Filelength = Len(CadCamrecord)

            Open "c:\factory\System\cadcam.sys" For Random Access Read Write Sha
            Filelength = Len(CadCamrecord)

            Open "c:\factory\System\cadcam.sys" For Random Access Read Write Sha
red As filenum Len = Filelength
            For j = 1 To ThisFileslength
                Put filenum, j, CadCamArray(j)
            Next

            Close filenum
```

```
End If



'Dim CadcamArray() As CadCamvariable
'Dim ThisfilesLength As Integer
'Type CadCamvariable
'Packagename As String * 50
'FireCommand As String * 50
'FocusCommand As String * 100
'WorkDirectory As String * 150
'Recordleng As Integer
'End Type


End Sub

Sub Form_Load ()

'This particular form is responsible for creating the integration between
'the System Integration Tool for advanced manufacturing automation and
'any available windows CAD\CAM Package.

'Declare local variables

Dim ThisFilesname As String
Dim i As Integer
Dim filenum As Integer
Dim ThisProcessPosts As Postrecord
Dim ThisPostProcsarray() As Postrecord
Dim ProcTyp As String * 4

'Type Postrecord
'CurrentIndex As Integer
'CurrentProcId As Integer
'Recordleng As Integer
'ProcessTyp As String * 50
'Index As Integer
```

```
ackageName = Trim(CadCamname)
 recommand = Trim(Firestring)
ocusCommand = Trim(AppFocus)
orkDirectory = Trim(WorkDir)

eng

n, CadCamArray())


levant directories for this package so that
```

```
CadCamArray(CadCamArray(1).Recordleng).F
CadCamArray(CadCamArray(1).Recordleng).F
CadCamArray(CadCamArray(1).Recordleng).W

ThisFileslength = CadCamArray(1).Recordl
'now save the new data to file
Call WritetoFile2(filenum, ThisFileslenc
Close filenum
'now add the new item to the listbox
Cad_Cam_Packages.AddItem CadCamname
'now we must automatically create the re
'Workdir = CadCamname
'MkDir "c:\Factory\" & Trim(Workdir)
```

```
CurrentIndex As Integer
CurrentProcId As Integer
Recordleng As Integer
ProcessTyp As String * 50
Index As Integer
PostTyp As String * 50
End Type


'Now we must open the file that contains the names of all the post-processors
'for the chosen piece of equipment
Filenum = FreeFile
ProcTyp = Trim(Tasknam)
ThisFilesname = "c:\factory\Process\Postname\" & Trim(ProcTyp) & "proc.pps"

Text1.Text = ThisFilesname
ThisFileslength = GetFileslength4(ThisFilesname, ThisProcessPosts, ThisPostProcs
array(), filenum)

If ThisFileslength < 1 Then
    'there is nothing in the file to add to the listbox therefore exit the sub
    MsgBox "You cannot develop a Toolpath file as there are no Cad\Cam packages
```

```
onfigured yet.Please configure a package for the system.", MB_ICONINFORMATION
MB_OK, "Cad\cam Status"
    Close filenum
    Exit Sub
    Else
        'There is something in the file therefore add the information the need
 display
        'to the listbox and write remaining information to memory.
        Call ReadAndAddToList3(ThisPostProcsarray(), ThisFileslength, filenum)
        Close filenum
nd If

oolpthFname.Text = CNCncifname


'Firstly we must disable the 'Cad\Cam' run button until a relevant cad\cam
 package has been selected from the cad\cam listbox.

un_Cad_Cam.Enabled = False
el_Package.Enabled = False

'Open the file containing the names of all the Cad\Cam
 packages so far configured for integration with this
 package and read their names into the listbox on this form.

hisFilesname = "C:\Factory\System\Cadcam.sys"
ilenum = FreeFile

hisFileslength = getfilesLength3(ThisFilesname, CadCamrecord, CadCamArray(), f
enum)

f ThisFileslength = 0 Then 'there is nothing to add to the
 listbox therefore we should close the file and then exit
'this procedure.
    Close filenum
    Exit Sub
        Else 'there is something in the file so we should load
        'its contents to memory and then the listbox.

        'Write file information to memory
        Call WriteToMemory2(filenum, ThisFileslength, CadCamArray())

        'add cad\cam package names to listbox
        'Write file information to memory
        Call WriteToMemory2(filenum, ThisFileslength, CadCamArray())

        'add cad\cam package names to listbox
        For i = 1 To ThisFileslength
            If Len(Trim(CadCamArray(i).PackageName)) = 0 Then ' if there is not
ing in the
            'recordposition then don't add it
                GoTo Skip
            End If
            Cad_Cam_Packages.AddItem CadCamArray(i).PackageName
kip:
        Next
        Close filenum ' Close the file
nd If
```

```
:nd Sub

;ub Form_Unload (Cancel As Integer)

:ancel = -1
)n Error Resume Next

If Len(Def_job.SupInstructionsFname.Text) = 0 Then
    Exit Sub
End If

'Load MachSup
Cancel = 0
End Sub

Sub Run_Cad_Cam_Click ()

'This particular procedure takes the name of the CAD\CAM
'package as selected from the listbox and fires that
'particular package from Visual Basic, it then changes
'focus to that package.

On Error GoTo TrapError

'Declare Variables
Dim PackageSelected As String
Dim i As Integer
Dim Firestring As String
Dim X As Variant
Dim Pathnam As String * 100
Dim CurrentDirectory As String
Dim Errortyp As Integer
Dim ThisTime
Dim Focuscode As String
Dim typ As Integer
Dim msg As String
Dim Titles As String
Dim reply As Integer

'Create an array to keep track of currently activated cadcam packages
Static CadCamGoing(100) As String * 40

Del_Package.Enabled = False
'Firstly make sure that something has indeed been selected
'from the listbox.
If Len(Trim(Cad_Cam_Packages.List(Cad_Cam_Packages.ListIndex))) = 0 Then
    'nothing has been selected, so notify user and exit sub
```

```
'firstly make sure that something has indeed been selected
'from the listbox.
If Len(Trim(Cad_Cam_Packages.List(Cad_Cam_Packages.ListIndex))) = 0 Then
    'nothing has been selected, so notify user and exit sub
    MsgBox "You didn't select a package to start", MB_ICONSTOP, "Selection Error
"
End If

'firstly we must deduce what particular packagename has been selected from
'the listbox and when this has been done we must open up the file containing
'the names of all those packages and get a match between the packagename
'selected and a name in that file, this will give its position in the array
```

```
'and from this it is possible to get that particular packagenames firecode,
'and focuscode.

PackageSelected = Trim(Cad_Cam_Packages.List(Cad_Cam_Packages.ListIndex))

'Now we must get a match between the 'PackageSelected' variable and the name
'of the package in the file - but because the file contents have already been
'written to memory it is not necessary to reopen the file again and re-read
'the contents to memory, so just go for the direct match.
For i = 1 To ThisFileslength
    If Trim(CadCamArray(i).PackageName) = Trim(PackageSelected) Then 'If there i
s a match
        'then get the rest of the information needed to fire the program at
        'that position and write to variables.
        Firestring = Trim(CadCamArray(i).Firecommand)
        Focuscode = Trim(CadCamArray(i).FocusCommand)
        Pathnam = Trim(CadCamArray(i).WorkDirectory)
        'Pathname = "c:\mill5"
    End If
Next


'the first thing that is required to be done is that we change the working
'directory from the default directory which is probably "c:\Windows" to
'the directory needed to run the cad\cam package successfully.

'firstly record the current working directory, so that it can be changed
'back later.

CurrentDirectory = CurDir$

'Now change the directory to the one specified in the "cadcam.sys" file

ChDir Trim(Pathnam)

'fire the program that we want to use
X = Shell(Firestring, 3)

'revert to the original path
ChDir Trim(CurrentDirectory)

msg = "hello"
typ = MB_ICONSTOP
Titles = "test"

MsgBox msg, typ, Titles

'X = Shell("Hello", 3)
'AppActivate "Project1"
'AppActivate "Mastercam Mill 5.0"

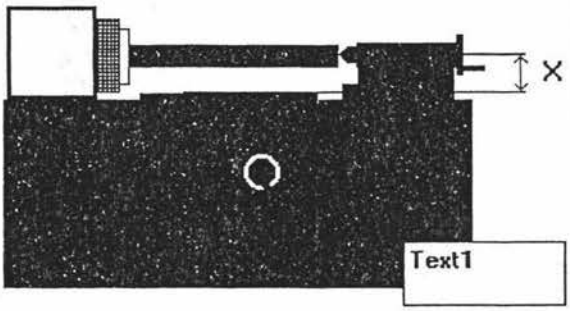


Exit Sub

TrapError:
```

This particular window allows the user to choose from a list the capacity category required to perform the particular task as required. An illustration provides a graphic representation of what this capacity means.

The illustration shows the required capacity parameter "X". Please select from the list below the minimum "X" dimension required to carry out the task.



Capacity List

Select

```
CNC_DRIL.FRM - 1

Dim Check As Integer

Sub CNCCaplist_Click ()

'Set flag = 1 to show that a capacity has been selected
Check = 1
'Calculate machine capacity selected based on item index
'as selected
Machcal = ((CNCCaplist.ListIndex) + 1) / 10

End Sub

Sub Command1_Click ()
'Initialise variables
Dim Msg As String
Dim Title As String
Dim Typ As Integer
Dim Reply As Integer

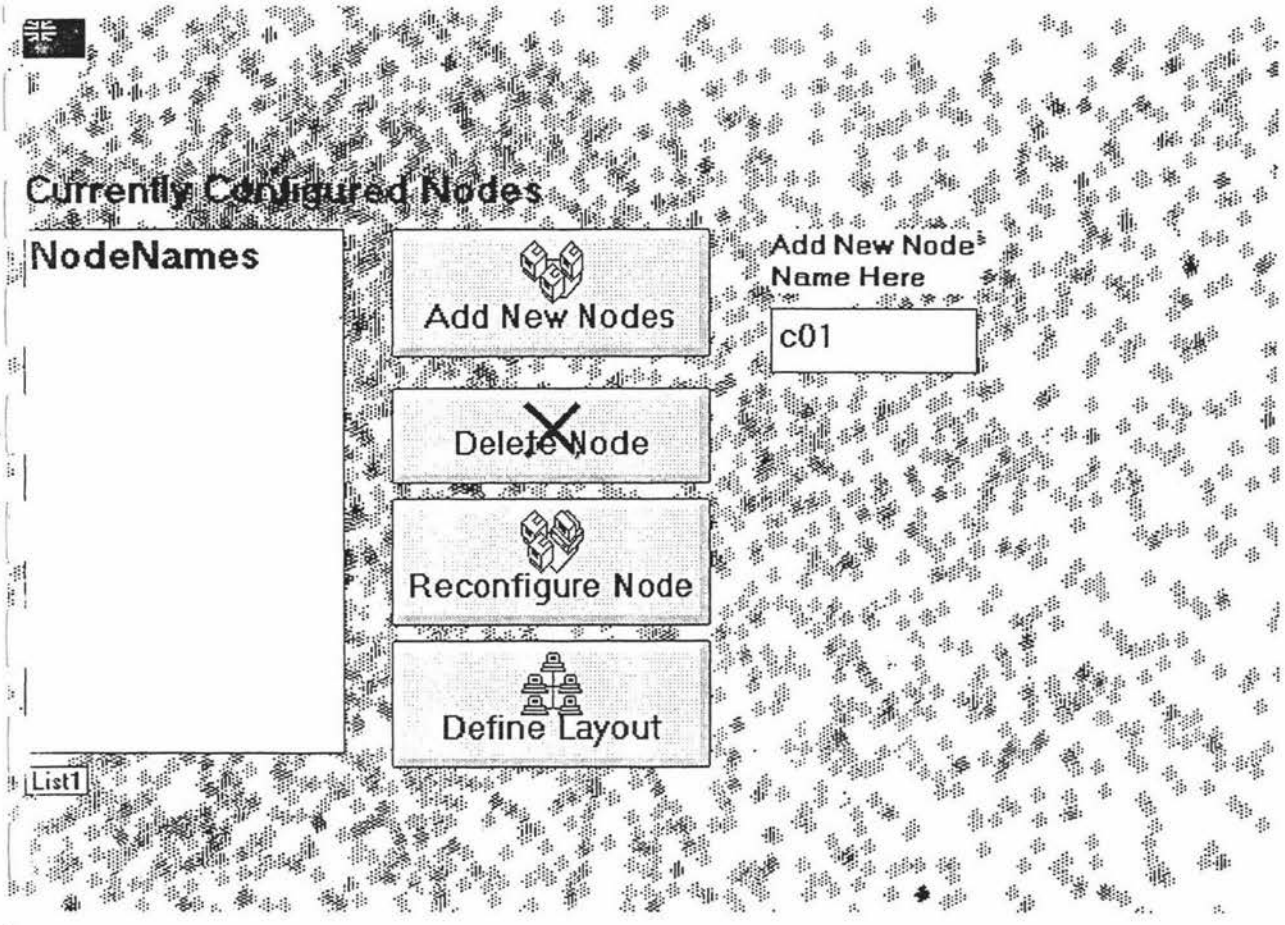
'Check to see whether or not a capacity has been selected
'from the listbox
If Check = 0 Then 'If nothing has been selected
Title = "Capacity Selection Error"
Msg = "You have not selected a capacity for this machine , please do so"
Typ = MB_ICONEXCLAMATION + MB_OKCANCEL
Reply = MsgBox(Msg, Typ, Title)
   If Reply = IDOK Then
   Exit Sub
   Else
   Typ = 0
   Title = "Machine Configuration"
   Msg = "A machine has not been configured for this task - You will have to!"
   Reply = MsgBox(Msg, Typ, Title)
   Unload CNC_DRILL
   Exit Sub
   End If
End If


DEF_JOB.MachRequired.Text = CInt(MachType.Text) + Machcal
Check = 0
Unload CNC_DRILL




End Sub

Sub Form_Load ()
MsgBox (CStr(ThisMachType))
MachType.Text = ThisMachType
'Text1.Text = MachType
End Sub
```


Currently Configured Nodes
NodeNames
Add New Nodes
Add New Node Name Here
c01
Delete Node
Reconfigure Node
Define Layout
List1

```
Option Explicit
Option Base 1
Option Compare Text


Sub Add_New_Nodes_Click ()
'This button allows the user to add new nodes to the currently configured syste
.
'The Bid Rotation Application uses this file so that it knows what nodes it has
to
'rotate the bid token around.
Dim recordlength As Integer
Dim Filenum As Integer
Dim i As Integer
Dim j As Integer
Dim ThisRecord As nodelistrecord
Dim Position As Integer
Dim Thisbit As String * 10
Dim NodebeingConfig As String

On Error Resume Next

'there is a limit of fifty cell controllers that can be defined in the factory
t any one time
'check to see that this has not been exceeded
If Nodenames.ListCount = 50 Then 'the number of cells in the factory has reache
 its maximum value
    MsgBox "The maximum number of defineable cells has been reached , no more ca
n be added", MB_ICONSTOP, "Factory at Nodal Limit"
    Exit Sub
End If

'check to see if the desired nodename has not already been entered
For i = 1 To Nodenames.ListCount
    If Len(Trim(Newnode.Text)) Like Nodenames.List(i - 1) Then
        Exit Sub
    End If
Next


If Trim(Newnode.Text) Like "Fact" Then
    Nodenames.AddItem UCase(Trim(Newnode.Text)), 0
    GoTo Nextpart
End If
'First make sure that there is something sensible to be added to the file befor
adding anything


If Len(Trim(Newnode.Text)) = 0 Then 'the user has not yet stated a node to be a
ded
    MsgBox "You have not entered any nodename!", MB_ICONSTOP + MB_OK, "Enter No
e Name"
    Exit Sub
    'Else there is something that we need to add to the file
        Else
        If Trim(Newnode.Text) Like "C##" Then
```

```
            Nodenames.AddItem UCase(Trim(Newnode.Text))
            Else MsgBox "The Nodename entered is not valid , please enter one o
 the form 'c01'", MB_ICONSTOP + MB_OK, "Enter Valid Node Name"
            Exit Sub
        End If
End If

 if the program gets to here it means that the node has been sucessfully added a
nd we need to
'prompt the user for all the othe information required namely the number of vmd'
  and whats in
'the vmd's and their capacity and what tools are at their disposal

 Now set the global variable which tells successive forms what node is currentl
 being configured
NodebeingConfig = CStr(Newnode.Text)
 hisnodenam = NodebeingConfig


 f NodebeingConfig Like "C0#" Then
    Thisbit = Mid$(Trim(NodebeingConfig), 3, 3)
    Thisnode = NodebeingConfig
    NodebeingConfiged = CInt(Thisbit)
    'MsgBox (CStr(NodebeingConfiged))
    Else
        Thisbit = Mid$(Trim(NodebeingConfig), 2, 3)
        NodebeingConfiged = CInt(Thisbit)
       ' MsgBox (CStr(NodebeingConfiged))
nd If


Nextpart:
 ilenum = FreeFile
'Calculate the recordlength in Bytes
recordlength = Len(ThisRecord)


f FileLen(Netdrive & "Factory\Config\nodefile.txt") = 0 Then 'the file doesnot
 xist so dont kill it
   GoTo Skip
    Else
    'Kill the oldfile
    Kill Netdrive & "Factory\Config\nodefile.txt"
nd If


kip:
 )pen the file for random access with allowable functions read and write
 pen Netdrive & "Factory\Config\nodefile.txt" For Random Access Read Write Share
  As Filenum Len = recordlength

  = 1

 r j = 0 To (Nodenames.ListCount - 1)
    ThisRecord.node = Nodenames.List(j)
    Put Filenum, i, ThisRecord
```

```
    i = i + 1
Next

Close Filenum

If Newnode.Text Like "Fact" Then
Newnode.Text = ""
    Exit Sub
End If


'show the form where the node' vmd's can be defined
Node_Def.Show

End Sub

Sub Def_Layout_Click ()
'declare variables
Dim i As Integer


'record the number of cell controllers that have been configured for the Factor
.
NumberOfCellControllers = (Nodenames.ListCount - 1)

'next store the contents of the list into an array so that the relevant cell co
troller names
'can be added to the correct symbols in the next form.

'Now assign each element of the cell controller portion of the list to an eleme
t of an array
'element.
For i = 1 To NumberOfCellControllers
    CellsConfigered(i) = Nodenames.List(i)
    'MsgBox (CellsConfigered(i))
Next

Define_Layout.Show
End Sub

Sub Del_NodeName_Click ()
Dim recordlength As Integer
Dim Filenum As Integer
Dim i As Integer
Dim j As Integer
Dim ThisRecord As nodelistrecord
Dim Position As Integer
Dim ItemToBeRemoved As String * 10
On Error Resume Next


'Check to see if something has been selected
If Len(Nodenames.List(Nodenames.ListIndex)) = -1 Then
    Exit Sub 'as nothing has been selected

    Else 'continue with the program
```

```
    ItemToBeRemoved = Nodenames.List(Nodenames.ListIndex)
    Nodenames.RemoveItem (Nodenames.ListIndex)

    'kill the oldfile and save the new
    Kill Netdrive & "Factory\Config\nodefile.txt"

    'Assign a free number to this file
    Filenum = FreeFile
    'Calculate the recordlength in Bytes
    recordlength = Len(ThisRecord)

    'Open the file for random access with allowable functions read and write
    Open Netdrive & "Factory\Config\nodefile.txt" For Random Access Read Write
ıared As Filenum Len = recordlength

    i = 1

    For j = 0 To (Nodenames.ListCount - 1)
        ThisRecord.node = Nodenames.List(j)
        Put Filenum, i, ThisRecord
        i = i + 1
    Next

    Close Filenum

    'however this is not the end of it we are also required to modify the intera
:rival Matrix

    'Type NodeTimedelay
    'MatrixDim As Integer
    'Interarrivaltime as integer
    'CellnameX As String * 10
    'CellnameY as String * 10
    'End Type

    Dim Thesenodestimedif() As NodeTimedelay
    Dim TheseNodesRecord As NodeTimedelay
    Dim Matrixsize  As Integer
    Dim recordsize As Integer
    Dim count As Integer
    Dim listpointer As Integer

    'redimension the 2-D array to get the first record only as we are required t
o get the size
    'of the matrix

    'this is all very fine but we must save it all to file so that it can be rec
vered as required
    Filenum = FreeFile
    recordlength = Len(TheseNodesRecord)

    Open Netdrive & "Factory\Bidding\Nodetime\NodeDely.txt" For Random Access Re
ad Write As Filenum Len = recordlength

    'firstly we only require the first record to get the matrixsize
    ReDim Thesenodestimedif(1)
```

```
    'get the first record that stores the value of the matrix size
    Get Filenum, 1, Thesenodestimedif(1)

    'redimension to take all the records that are in the file
    recordsize = (Thesenodestimedif(1).Matrixdim * Thesenodestimedif(1).Matrixd
m)
    'MsgBox CStr(Recordsize)

    ReDim Thesenodestimedif(recordsize)

    'now get all the records
    For i = 1 To recordsize
        Get Filenum, i, Thesenodestimedif(i)
        List1.AddItem CStr(Thesenodestimedif(i).Interarrivaltime)
        List1.AddItem Thesenodestimedif(i).CellnameX
        List1.AddItem Thesenodestimedif(i).Cellnamey
    Next

    'now search the list created and look for a match between the item deleted
nd the cellname in
    'the array - if there is a match then delete the record.Once all items have
been scanned
    'then redimension the array to hold the contents of the list and then write
the contents of
    'the list to file.
    listpointer = 1
    For i = 1 To (Thesenodestimedif(1).Matrixdim * Thesenodestimedif(1).Matrixdi
m)
        'If Listpointer = 1 Then
         '   Listpointer = 0
          '  i = i - 1
        'End If
        If Trim((ItemToBeRemoved)) Like Trim(Thesenodestimedif(i).CellnameX) Or
Trim((ItemToBeRemoved)) Like Trim(Thesenodestimedif(i).Cellnamey) Then
            List1.RemoveItem (3 * listpointer) - 1
            List1.RemoveItem (3 * listpointer) - 2
            List1.RemoveItem (3 * listpointer) - 3
            Else
            'MsgBox (Trim(Nodenames.ListIndex))
            listpointer = listpointer + 1
        End If

    Next

    Close Filenum
    'now resave the contents of the list back to file

    'first get the length of the file that is required - this will be equal to t
he (matrixsize of
    'the last matrix less one) squared * 4 elements

    'kill the old version of the file to get rid off overflow
    Kill Netdrive & "Factory\Bidding\Nodetime\NodeDely.txt"

    Filenum = FreeFile
```

```
    recordlength = Len(TheseNodesRecord)

    Open Netdrive & "Factory\Bidding\Nodetime\NodeDely.txt" For Random Access R
ad Write As Filenum Len = recordlength

    recordsize = (Thesenodestimedif(1).Matrixdim - 1) * (Thesenodestimedif(1).M
trixdim - 1)

    'MsgBox (CStr(recordsize))
    'redimension the array to take all the records
    ReDim Preserve Thesenodestimedif(recordsize)

    'now add all the elements of the listbox to the array
    Thesenodestimedif(1).Matrixdim = Thesenodestimedif(1).Matrixdim - 1

    count = 0
    'MsgBox (CStr(Thesenodestimedif(1).Matrixdim))
    For i = 1 To recordsize
        Thesenodestimedif(i).Interarrivaltime = CInt(List1.List(count))
        'MsgBox ("Interarrival time for " & CStr(i) & "th is " & CStr(Thesenode
imedif(i).Interarrivaltime))
        count = count + 1
        Thesenodestimedif(i).CellnameX = List1.List(count)
        'MsgBox ("CellnameX for " & CStr(i) & "th is " & CStr(Thesenodestimedif
i).CellnameX))
        count = count + 1
        Thesenodestimedif(i).Cellnamey = List1.List(count)
        'MsgBox ("Cellnamey for " & CStr(i) & "th is " & CStr(Thesenodestimedif
i).Cellnamey))
        count = count + 1
        Put Filenum, i, Thesenodestimedif(i)
    Next

    Close Filenum
    'Type NodeTimedelay
    'MatrixDim As Integer
    'Interarrivaltime as integer
    'CellnameX As String * 10
    'CellnameY as String * 10
    'End Type
'        List1.AddItem ThesenodesTimedif(i).Interarrivaltime
'        List1.AddItem ThesenodesTimedif(i).CellnameX
'        List1.AddItem ThesenodesTimedif(i).Cellnamey
End If

nd Sub

ub Form_Load ()

'When this form fires up it must open a file called "C:\Factory\Config\nodefile.
xt" which
contains the names of all the nodes for which the vmd's have been configured.If
 the file
'contains any names it writes those names into the "NodeNames" listbox.

Dim recordlength As Integer
```

```
Dim Filenum As Integer
Dim ThisRecord As nodelistrecord
Dim Position As Integer
Dim Firstime As Integer

On Error Resume Next

If FileLen(Netdrive & "Factory\Config\nodefile.txt") = 0 Then 'if there is noth
ng in the file then exit
'this subprocedure
    Exit Sub
End If


'Assign a free number to this file
Filenum = FreeFile
'Calculate the recordlength in Bytes
recordlength = Len(ThisRecord)



'Open the file for random access with allowable functions read and write
Open Netdrive & "Factory\Config\nodefile.txt" For Random Access Read Write Shar
d As Filenum Len = recordlength

Position = 1
Firstime = 0
Do Until EOF(Filenum)
    If Firstime = 1 Then
        Nodenames.AddItem Trim(ThisRecord.node)
    End If
    Get Filenum, Position, ThisRecord
    Position = Position + 1
    Firstime = 1
Loop
'********************IMPORTANT NOTE*****************************************
**********
'Note that in the loop above we donot add the name of the node to the listbox ur
til the start
'of the loop after the loop in which the nodename was read from the file.This is
 because the
'EOF function does not detect the end of the file until it has failed to read th
e last record
'and if we added it in the same loop as it was read from the file then the last
record read
'from the record would be a zero length string and this would be added to the li
st as a blank.
'In order to prevent a blank at the start of the listbox we use the variable "Fi
rstime" to
'skip the additem as no record is read at this point.
Close Filenum


End Sub

Sub ReconFig_Click ()
```

```
CONFIG_F.FRM - 8

Dim i As Integer
Dim Thisbit As String * 10
Dim NodebeingConfig As String

If Len(Nodenames.List(Nodenames.ListIndex)) = 0 Then
    'nothing has been selected so exit the procedure
    MsgBox "You have not selected a node that you want to reconfigure, please c
 so!", MB_ICONSTOP, "Selection Error"

    Exit Sub
End If

NodebeingConfig = CStr(Trim(Nodenames.List(Nodenames.ListIndex)))
Thisnodenam = NodebeingConfig

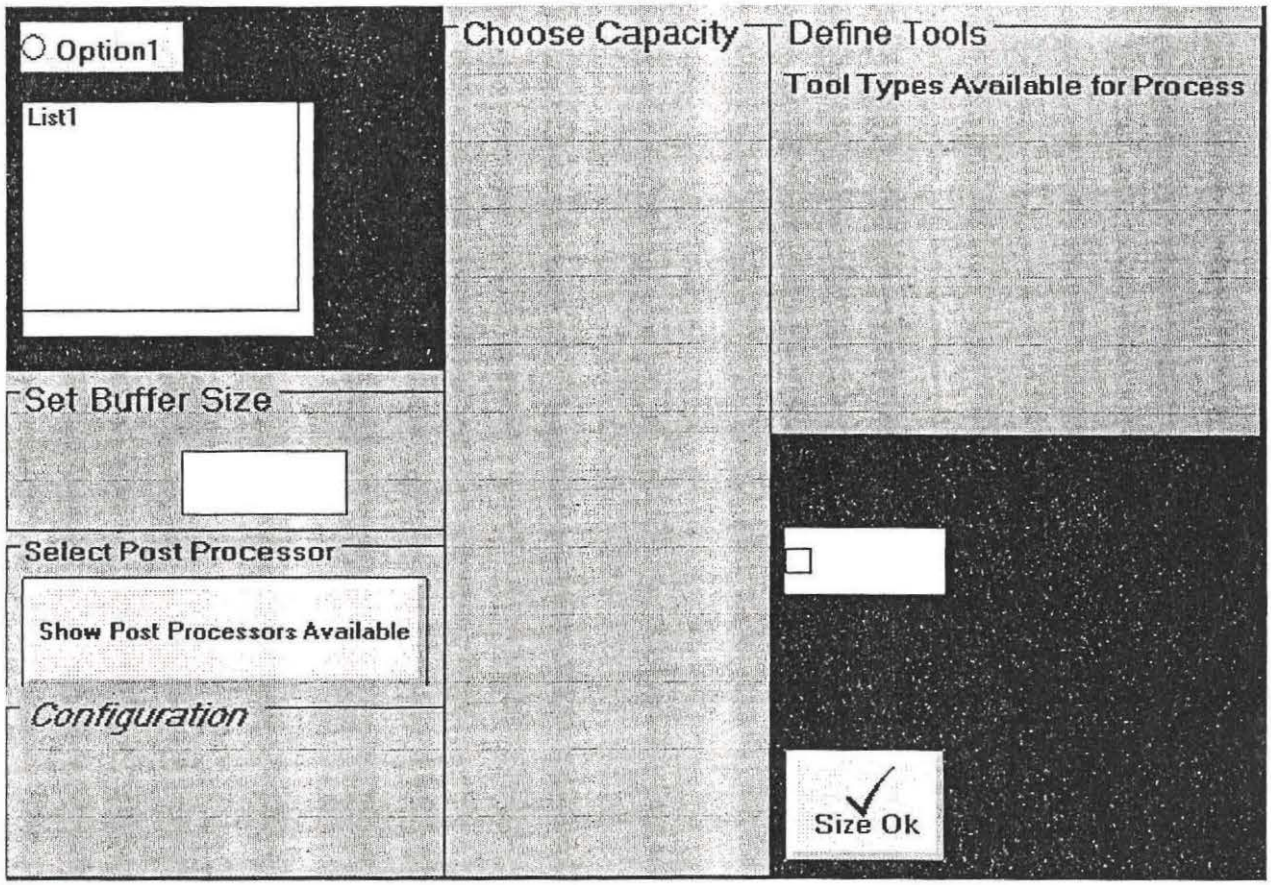
If NodebeingConfig Like "Fact" Then
    MsgBox "You cant select the Factory Controller Node for reconfiguration, pl
ase select a cell controller node.", MB_ICONSTOP, "Selection Error"
    Exit Sub
End If


If NodebeingConfig Like "C0#" Then
    Thisbit = Mid$(Trim(NodebeingConfig), 3, 3)
    Thisnode = NodebeingConfig
    NodebeingConfiged = CInt(Thisbit)
    MsgBox (CStr(NodebeingConfiged))
    Else
        Thisbit = Mid$(Trim(NodebeingConfig), 2, 3)
        NodebeingConfiged = CInt(Thisbit)
        MsgBox (CStr(NodebeingConfiged))
End If


'show the form where the node' vmd's can be defined
Node_Def.Show



End Sub
```


Option1
List1
Choose Capacity
Define Tools
Tool Types Available for Process
Set Buffer Size
Select Post Processor
Show Post Processors Available
Configuration
Size Ok

```
CONFIG_N.FRM - 1

Option Explicit
Option Compare Text
Option Base 1

Dim ThisCaption As String
Dim ThisNoOptions As Integer
Dim Machdef As Integer
Dim OptionIndex As Integer
Dim OpIndex As Integer


Dim MachineDef As Single
Dim NoChkBoxes As Integer
Dim Firstup As Integer
Dim ProcOptionIndex As Integer
Dim ChosenIndex As Integer

Dim Ghandler As Long
Dim fart As Variant
Dim eda_err As Variant
Dim tex As Integer
Dim Newnum As Variant
Dim node As String * 15
Dim tags As String * 15
Dim Field As String * 15
Dim Answer As String * 80
Dim ans As String * 10
Dim defnode As Long
Dim trap As String * 10
Dim texts As String * 6
Dim Nodetyp As String * 20
Dim nodes As String * 20
Dim bidrotenable As Variant
Dim BlockTyp As String * 20
Dim Bidoperatestate As Variant
Dim Bidnodetext As String * 10
Dim bidtokenstate As Variant
Dim Tokenstate As Integer
Dim BidOpstate As Variant

' functions specific to VB (not standard eda fcns)
Declare Sub eda_version Lib "vdba.dll" (ByVal v$)
Declare Sub eda_get1_ascii Lib "vdba.dll" (ByVal n$, ByVal t$, ByVal f$, ByVal v
$)
Declare Function eda_set1_ascii% Lib "vdba.dll" (ByVal n$, ByVal t$, ByVal f$, B
yVal v$)

' Note define_group returns a long

Declare Function eda_define_group& Lib "vdba.dll" (ByVal Count%, ByVal Detect%)
Declare Sub eda_delete_group Lib "vdba.dll" (ByVal Handle&)

' should always pass 0 for vsp
Declare Function eda_define_ntf& Lib "vdba.dll" (ByVal Handle&, ByVal enode$, By
Val etag$, ByVal efield$, ByVal vsp&)
Declare Sub eda_delete_ntf Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
```

```
'
Declare Sub eda_lookup Lib "vdba.dll" (ByVal Handle&)
Declare Sub eda_read Lib "vdba.dll" (ByVal Handle&)
Declare Sub eda_write Lib "vdba.dll" (ByVal Handle&)
Declare Function eda_write1% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
Declare Sub eda_wait Lib "vdba.dll" (ByVal Handle&)
Declare Function eda_query_wait% Lib "vdba.dll" (ByVal Handle&)
'
Declare Function eda_get_error% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
Declare Function eda_get_ascii% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, ByV
 Value$, ByVal maxlen%)
Declare Function eda_set_ascii% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, ByVa
 Value$)

' note pointer to float is passed
Declare Function eda_get_float% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, Valu
!)
Declare Function eda_set_float% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, Valu
!)

'Variable declaration for node configuration
Dim Thisconfig(4) As whatmachinesarewhere

'buffer size storage variable
Dim Thisbuffer(1) As Inqueuebuffersize


Function CheckTheList (ByVal TheToolSpec)
Dim i As Integer

'this function checks to see if there is already a tool of the same size
'and type in the listbox and if there is returns a 1 and if there isnt
'returns a 2

For i = 0 To (List1.ListCount - 1)
    If ((CSng(List1.List(i))) - TheToolSpec) = 0 Then 'there is a match
        CheckTheList = 1
        Exit Function
    End If
Next

CheckTheList = 2

End Function

Sub Command1_Click ()
'index is the number of the option button that is just selected so use this info
 to extract
'the correct capacity information from the file
Dim Msg As String
Dim typ As Integer
Dim title As String
Dim Reply As Integer
Dim ProcessSpec() As ProcessDef
Dim ProcessTyp As ProcessDef
Dim Filelength As Integer
```

```
Dim i As Integer
Dim Filenum As Integer
Dim MatchPosition As Integer
Dim Divisionsize As Single

'Type ProcessDef
'ProcessNumber As Integer
'Processname As String * 20
'CurrentProcessId As Integer
'ProcessId As Integer
'Maxcap As Integer
'End Type
Dim OptionSelected As String
Dim Request As String
Dim Answer As String
Dim Thisstate As Optionstate
Dim Recordlength As Integer
Dim Filenam As String * 40
Dim ArrayTyp As String
Dim EndOfArray As Integer
Dim OldOption As String
Dim Thatstate As OldOptions
Dim OptionList() As OldOptions
Dim NextOldoption As String
Dim ThisError As Integer
Dim j As Integer
Dim Fileslength As Integer

'Declare startup array
Dim Thisconfig As ConFigSup
Dim Position As Integer

Dim Thisstring As String * 20
Dim Thisint As Integer


Filenum = FreeFile

Recordlength = Len(Thisconfig)

Filenam = "C:\Factory\System\V" & CStr(TheVMD) & "C" & CStr(NodebeingConfiged) &
"Sup.sys"

Open Filenam For Input As Filenum Len = Recordlength

Do Until EOF(Filenum)
    Line Input #Filenum, Thisstring
    MsgBox (Thisstring)
Loop



Close Filenum

End Sub
```

```
Sub Command3D1_Click ()
Dim Nodename As String * 10
Dim Machcode As Single
Dim Field As String * 6
Dim BlockTyp As String * 14
Dim Filenum As Integer
Dim Filenam As String
Dim MatchPosition As Integer
Dim i As Integer

Dim NodeConfig As whatmachinesarewhere

'this procedure tests to see if the file is created properly

Filenum = FreeFile

Filenam = "C:\Factory\Config\Cells\C" & CStr(NodebeingConfiged) & "Confg.sys"

Open Filenam For Random Access Read Write As Filenum Len = Len(NodeConfig)

For i = 1 To 4
    Get Filenum, i, Thisconfig(i)
    MsgBox ("The machine type is " & CStr(Thisconfig(i).MachType))
    MsgBox ("The nodenumber is " & CStr(Thisconfig(i).Nodenumber))
    MsgBox ("The enabled variable is" & CStr(Thisconfig(i).Enabled))
    MsgBox ("The filelength is " & CStr(Thisconfig(i).Filelength))
Next

Close Filenum

End Sub

Sub Command3D2_Click ()
'set buffer size
End Sub

Sub Command6_Click ()
    'this procedure allows the user the ability to set the maximum buffer size a
llowed for the vmd
    'being considered.This buffer size setting could be in response to some form
 of management
    'policy such as the theory of constriants for instance.Once the length of ti
me required to do the
    'work in a VMD's incoming queue is greater than it's buffer size that VMD ca
n no longer form
    'a bid for any auction as it's ingoing queue is deemed to be full therefore
it is not allowed to
    'partake in an auction.

    'Declare variables for this procedure
    Dim Filenum As Integer
    Dim Fileslength As Integer
    Dim Filename As String
    Dim Currentbuffer As Inqueuebuffersize
```

```
    'Dim Thisbuffer(1) As Inqueuebuffersize

    'now we are required to write the data to a file particular to this node
    Filenum = FreeFile 'define a freefile
    Fileslength = Len(Currentbuffer) 'define storage variable size in Bytes
    Filename = Netdrive & "Cellcont\queue\" & Trim(Thisnode) & "Bfsze.Buf" 'def
ne filename

    Open Filename For Random Access Read Shared As Filenum Len = Fileslength 'o
en the file for write access

    Get Filenum, 1, Thisbuffer(1)

    'now put the answer variable in the appropriate part of the array
    MsgBox (CStr(Thisbuffer(1).VMD1Buffer))
    MsgBox (CStr(Thisbuffer(1).VMD2Buffer))
    MsgBox (CStr(Thisbuffer(1).VMD3Buffer))
    MsgBox (CStr(Thisbuffer(1).VMD4Buffer))

    Close Filenum
    'Dim Thisbuffer(1) As InQueueBuffersize
    'Type InQueueBuffersize
     '   VMD1Buffer As Integer
      '  VMD2Buffer As Integer
       ' VMD3Buffer As Integer
       ' VMD4Buffer As Integer
    'End Type
    'MsgBox (Trim(Thisnode))
    'MsgBox (CStr(TheVMD))

End Sub

ub sendinfo (n, t, f, Text)
Ghandler = eda_define_group(1, 0)
defnode = eda_define_ntf(Ghandler, n, t, f, 0)

all eda_lookup(Ghandler)
Call eda_wait(Ghandler)
da_err = eda_get_error(Ghandler, defnode)
ext5.Text = Str$(eda_err)
Newnum = Text
art = eda_set_ascii(Ghandler, defnode, Newnum)
all eda_write(Ghandler)
Call eda_wait(Ghandler)
all eda_delete_group(Ghandler)

End Sub

ub Sendmessage (Message As String, Typof As Integer, title As String)

MsgBox Message, Typof, title

End Sub

ub Test2_Click ()
Dim Nodename As String * 10
```

```
Dim Machcode As Single
Dim Field As String * 6
Dim BlockTyp As String * 14
Dim Filenum As Integer
Dim Filenam As String
Dim MatchPosition As Integer
Dim i As Integer
Dim Temprecord As Integer

Dim NodeConfig As whatmachinesarewhere

'Dim ThisConfig(4) As whatmachinesarewhere

'Type Activenodes
'Filelength As Integer
'CellnodeNum As Integer
'End Type
Dim WhatNodesConfigured As Activenodes
Dim Noderecords() As Activenodes

Filenum = FreeFile

Filenam = "C:\Factory\Config\Cells\Whatnode.sys"

Open Filenam For Random Access Read Write As Filenum Len = Len(WhatNodesConfigu
ed)


ReDim Noderecords(1)'dimension array to take the first record only

Get Filenum, 1, Noderecords(1) 'get the first record

Noderecords(1).Filelength = Noderecords(1).Filelength + 1

'redimension the array to take all the records plus one more in case we have to
add a new record
ReDim Preserve Noderecords(Noderecords(1).Filelength)


For i = 1 To Noderecords(1).Filelength
    Get Filenum, i, Noderecords(i)
        MsgBox (CStr((Noderecords(i).Filelength)))
        MsgBox (CStr((Noderecords(i).CellNodeNum)))
Next

'Filelength As Integer
'CellnodeNum As Integer
Close Filenum




End Sub

Sub Timer1_Timer ()
```

```
CONFIG_N.FRM - 7

'this timer event is fired once after the form has been loaded and opens files
 which contain the last selection in terms of both the machine or process type
 currently configured for this VMD and Cell as well as the currently configurec
'capacity interval for this VMD and Cell.This procedure is done in the form of
 timer
 as we cannot do it in the form load instructions as we cannot dedicate the val
es
 in the files to the option buttons as they have yet to be loaded.

'declare variables
 im Filenum As Integer
 im Filenam As String * 50

 im Machchoice As Integer
 im Thisindex As Integer
Dim capiNdex As Integer


'ThisConfig
ilenum = FreeFile

'Recordlength = Len(Thisconfig)

 ilenam = "C:\Factory\System\V" & CStr(TheVMD) & "C" & CStr(NodebeingConfiged) &
  "Mach.sys"
MsgBox (Filenam)
 pen Filenam For Input As Filenum Len = 5

'rite #Filenum, "Hello"
 nput #Filenum, Machchoice

Close Filenum

 okay now apply the correct setting for the process chosen as chosen by the user
  at a previous
 ime to the option button for the process

Option1(Machchoice).SetFocus


 lenum = FreeFile

Filenam = "C:\Factory\System\V" & CStr(TheVMD) & "C" & CStr(NodebeingConfiged) &
 "McCp.sys"

pen Filenam For Input As Filenum Len = 5

 rite #Filenum, "Hello"
nput #Filenum, capiNdex

 ose Filenum

ocap1(capiNdex).SetFocus
```

```
End Sub

Sub Choose_Click ()
Dim Msg As String
Dim typ As Integer
Dim title As String

Msg = "Choose the capacity interval that the process in this VMD falls into."
typ = 0
title = "Choose Capacity"
Call Sendmessage(Msg, typ, title)

End Sub

Sub Conn_Click ()
Dim Msg As String
Dim typ As Integer
Dim title As String

Msg = "The connect button allows the user the ability to send a message to the
MD in the correct "
Msg = Msg & "cell to configure it so that it is ready to start manufacture."
typ = 0
title = "Connecting To The Cell"
Call Sendmessage(Msg, typ, title)

End Sub

Sub Connect_Click ()
Dim Nodename As String * 10
Dim Machcode As Single
Dim Field As String * 6
Dim BlockTyp As String * 14
Dim Filenum As Integer
Dim Filenam As String
Dim MatchPosition As Integer
Dim i As Integer
Dim Temprecord As Integer

Dim NodeConfig As whatmachinesarewhere

'Type whatmachinesarewhere
'Filelength As Integer
'Nodenumber As Integer
'MachType As Single
'ProcessorType as string * 40
'Enabled As Integer
'End Type


'Type Activenodes
'Filelength As Integer
'CellnodeNum As Integer
'End Type
```

```
ONFIG_N.FRM - 9

im WhatNodesConfigured As Activenodes
 im Noderecords() As Activenodes
'When we select this button we want to send the process configuration details fr
m the factory
 controller to the Cell and VMD as required.We also want to be able to send the
tool details
'from this Factory Controllers directories to the correct Cell Controllers direc
 ories.

Machcode = CSng(Machdef + OpIndex / 10)
Nodename = Trim(Thisnode)
BlockTyp = "V" & CStr(TheVMD) & "MACHTYP"
Field = "A_CV"

Call sendinfo(Trim(Nodename), Trim(BlockTyp), Trim(Field), Machcode)


Nodename = Trim(Thisnode)
BlockTyp = "V" & CStr(TheVMD) & "CNCTYPE"
Field = "A_CV"

Call sendinfo(Trim(Nodename), Trim(BlockTyp), Trim(Field), Trim(PostprocessorSe
lected))



'this is all ok but we must also be able to save the node configuration data to
file so that when
'the Factory Controller starts up we can configure all the cells in the factory
at the beginning
'of the day.

'firstly we have to open a file which tells us the names of the nodes that have
already been
'configured because we have to know which configuration file data we have to se
d.
Filenum = FreeFile

Filenam = "C:\Factory\Config\Cells\Whatnode.sys"

Open Filenam For Random Access Read Write As Filenum Len = Len(WhatNodesConfigu
red)


ReDim Noderecords(1)'dimension array to take the first record only

Get Filenum, 1, Noderecords(1) 'get the first record

Noderecords(1).Filelength = Noderecords(1).Filelength + 1

'redimension the array to take all the records plus one more in case we have to
add a new record
ReDim Preserve Noderecords(Noderecords(1).Filelength)
```

```
'get all the records
Temprecord = Noderecords(1).Filelength
For i = 1 To Noderecords(1).Filelength
    Get Filenum, i, Noderecords(i)
    'check to see if this node has been configured before
    If Noderecords(i).CellNodeNum = NodebeingConfiged Then 'this nodes No. is a
ready recorded
        Close Filenum
        GoTo DoThis
        Exit Sub
    End If
Next

'if the program gets to here then a new record is required to be added

MsgBox (CStr(Temprecord))
Noderecords(Temprecord).CellNodeNum = NodebeingConfiged
Noderecords(Temprecord).ProcessorType = Trim(PostprocessorSelected)


'now put all the modified data back into the file.
Noderecords(1).Filelength = Temprecord
For i = 1 To Temprecord
    Put Filenum, i, Noderecords(i)
Next

Close Filenum

DoThis:
Filenum = FreeFile

Filenam = "C:\Factory\Config\Cells\C" & CStr(NodebeingConfiged) & "Confg.sys"

Open Filenam For Random Access Read Write As Filenum Len = Len(NodeConfig)

For i = 1 To 4
    Get Filenum, i, Thisconfig(i)
Next

For i = 1 To 4
'we must search the current file to see whether or not there is a record that a
ready exists
'for this node's VMD and if there is we should replace it rather than add a bra
d new record.
'we can do this by comparing this nodes number with that in the file as below
    If Thisconfig(i).Nodenumber = TheVMD Then
        MatchPosition = i
        'and replace the data at that position with the new data
        Thisconfig(MatchPosition).MachType = Machcode
        Thisconfig(MatchPosition).Enabled = 1
        Thisconfig(MatchPosition).ProcessorType = Trim(PostprocessorSelected)


    End If
Next
```

```
ONFIG_N.FRM - 11

Tf MatchPosition = 0 Then 'a new vmd is being configured so add this data to a r
w record.
    Thisconfig(1).Filelength = Thisconfig(1).Filelength + 1
    Thisconfig(Thisconfig(1).Filelength).MachType = Machcode
    Thisconfig(Thisconfig(1).Filelength).Nodenumber = TheVMD
    Thisconfig(Thisconfig(1).Filelength).Enabled = 1
    Thisconfig(MatchPosition).ProcessorType = Trim(PostprocessorSelected)


End If

now that we have done that we need to put everything back into the file
For i = 1 To 4
    Put Filenum, i, Thisconfig(i)
Next

Close Filenum

End Sub

Sub Form_Load ()
Dim Msg As String
Dim typ As Integer
Dim title As String
Dim Reply As Integer
Dim ProcessSpec() As ProcessDef
Dim ProcessTyp As ProcessDef
Dim Filelength As Integer
Dim i As Integer
Dim Filenum As Integer
Dim Filenam As String * 50
Dim Machchoice As Integer
Dim Thisindex As Integer
Dim capiNdex As Integer

On Error Resume Next

'write to a global variable the fact that the form has just been loaded
Firstup = 1

'write all the names of the available processes in this factory as defined in th
e define
processes module to memory

Filenum = FreeFile
Filelength = Len(ProcessTyp)
ReDim ProcessSpec(1)
Open "c:\Factory\Process\files\Proclist.Pfl" For Random Access Read Write Share
As Filenum Len = Filelength
Get Filenum, 1, ProcessSpec(1)

If ProcessSpec(1).Processnumber = 0 Then
    title = "File Empty"
    Msg = "You have to define the processes in the 'Define Processes' module be
ore you can "
    Msg = Msg & "configure any node."
```

```
    typ = MB_ICONEXCLAMATION + MB_OK
    Reply = MsgBox(Msg, typ, title)
    Close Filenum
    Unload Config_NewVMD1
    Exit Sub
End If

'else there is something there so add to list
ReDim Preserve ProcessSpec(ProcessSpec(1).Processnumber)

For i = 1 To ProcessSpec(1).Processnumber
    Get Filenum, i, ProcessSpec(i)
Next

'create extra check boxes to match the number of items in the file
Option1(0).Caption = ProcessSpec(1).Processname

For i = 1 To ((ProcessSpec(1).Processnumber) - 1)
    Load Option1(i)
    Option1(i).Top = Option1(i - 1).Top + 360
    Option1(i).Caption = ProcessSpec(i + 1).Processname
    Option1(i).Visible = True
Next

Close Filenum

Filenum = FreeFile

'Recordlength = Len(Thisconfig)

Filenam = "C:\Factory\System\V" & CStr(TheVMD) & "C" & CStr(NodebeingConfiged)
 "Mach.sys"
'MsgBox (Filenam)
Open Filenam For Input As Filenum Len = 5

Input #Filenum, Machchoice

Close Filenum

'okay now apply the correct setting for the process chosen as chosen by the use
 at a previous
'time to the option button for the process
Option1(Machchoice).Value = True


'now open the file that has the value of the capacity interval selected last a
 set
'the correct capacity interval in accordance with it.
Filenum = FreeFile

Filenam = "C:\Factory\System\V" & CStr(TheVMD) & "C" & CStr(NodebeingConfiged)
 "McCp.sys"

Open Filenam For Input As Filenum Len = 5

Input #Filenum, capiNdex
```

```
C )NFIG_N.FRM - 13


N sgBox (CStr(capiNdex))
Close Filenum

 ocap1(capiNdex).Value = True


 nd Sub

Sub Form_Unload (Cancel As Integer)
 im ThisFilenam As String
Dim Filenumb As Integer
Dim Thisstring As String
 im Thisint As Integer
 im Newstring As String


When we leave this form we want to update the bitmap which is in the form befor

 so as to make some sense of the new configuration


Filenumb = FreeFile
 hisstring = Trim(Option1(ChosenIndex).Caption)
ThisFilenam = "C:\Factory\Process\Specs\" & Thisstring & ".bmp"
 sgBox (ThisFilenam)

 elect Case TheVMD

 ase Is = 1

Open Trim(ThisFilenam) For Binary As #Filenumb 'open the bitmap file
 ode_Def.OLE1.FileNumber = Filenumb 'define the filenumber
 ode_Def.OLE1.Action = 12  'read the bitmap into the OLE.
 lose Filenumb 'Close the bitmap file

 ase Is = 2

 pen Trim(ThisFilenam) For Binary As #Filenumb 'open the bitmap file
Node_Def.OLE2.FileNumber = Filenumb 'define the filenumber
Node_Def.OLE2.Action = 12  'read the bitmap into the OLE.
 lose Filenumb 'Close the bitmap file

Case Is = 3

 pen Trim(ThisFilenam) For Binary As #Filenumb 'open the bitmap file
Node_Def.OLE3.FileNumber = Filenumb 'define the filenumber
Node_Def.OLE3.Action = 12  'read the bitmap into the OLE.
 lose Filenumb 'Close the bitmap file


 ase Is = 4

Open Trim(ThisFilenam) For Binary As #Filenumb 'open the bitmap file
 ode_Def.OLE4.FileNumber = Filenumb 'define the filenumber
Node_Def.OLE4.Action = 12  'read the bitmap into the OLE.
```

```
Close Filenumb 'Close the bitmap file

End Select

End Sub

Sub Opcap1_Click (Index As Integer, Value As Integer)

'Declare variables
Dim Filenam As String
Dim OptionIndex As Integer
Dim Filenum As Integer

'This event procedure allows the user the ability to choose a capacity and fron
then
'that capacity is saved to a file so that when this program is fired up once mo
e
'the correct current capacity is loaded.

Filenum = FreeFile

OpIndex = Index

MsgBox (CStr(OpIndex))

Filenam = "C:\Factory\System\V" & CStr(TheVMD) & "C" & CStr(NodebeingConfiged)
 "McCp.sys"

Open Filenam For Output As Filenum Len = 5

'Write #Filenum, "Hello"
Write #Filenum, Index

Close Filenum



End Sub

Sub Option1_Click (Index As Integer)

'index is the number of the option button that is just selected so use this int
 to extract
'the correct capacity information from the file
Dim Msg As String
Dim typ As Integer
Dim title As String
Dim Reply As Integer
Dim ProcessSpec() As ProcessDef
Dim ProcessTyp As ProcessDef
Dim Filelength As Integer
Dim i As Integer
Dim Filenum As Integer
Dim MatchPosition As Integer
Dim Divisionsize As Single
```

```
'Type ProcessDef
'ProcessNumber As Integer
'Processname As String * 20
'CurrentProcessId As Integer
'ProcessId As Integer
'Maxcap As Integer
'End Type
Dim OptionSelected As String
Dim Request As String
Dim Answer As String
Dim Thisstate As Optionstate
Dim Recordlength As Integer
Dim Filenam As String * 40
Dim ArrayTyp As String
Dim EndOfArray As Integer
Dim OldOption As String
Dim Thatstate As OldOptions
Dim OptionList() As OldOptions
Dim NextOldoption As String
Dim ThisError As Integer
Dim j As Integer
Dim Fileslength As Integer

'Declare startup array
Dim Thisconfig As ConFigSup
Dim Position As Integer

On Error Resume Next

ChosenIndex = Index
MsgBox (CStr(ChosenIndex))

If Firstup = 1 Then
    GoSub LoadOptions
    Firstup = 2
    Exit Sub
End If

'inforHm the user that changing the option button selection will change
'the configuration of the VMD and then give him the option of quitting
Msg = "Selecting a different option button will reset the system configuration
Msg = Msg & "are you sure that you want to change it?"
typ = MB_ICONQUESTION + MB_YESNO
title = "Change Configuration?"
Reply = MsgBox(Msg, typ, title)

If Reply = IDNO Then
 Exit Sub
End If


'kill the configuration file as we are reconfiguring
Kill Netdrive & "Factory\Config\Cells\V" & CStr(TheVMD) & "C" & CStr(NodebeingC
nfiged) & "tSpc.txt"
'MsgBox (Netdrive & "Factory\Config\Cells\V" & CStr(Thevmd) & "C" & CStr(Nodebe
```

```
ngConfiged) & "tSpc.txt")
List1.Clear

'Record the number of the process that we are selecting
'ProcOptionIndex = Index

LoadOptions:

'We are required to open up a the process list file regarding the processes tha
 have
'configured for this software and try to find a match between the process chose
 from
'the option buttons and the equivalent process in the file - once this is done
e
'no the position of the process in the file and then we can draw the relevant
'information from the file.

nextbit:
Filenum = FreeFile
Filelength = Len(ProcessTyp)
ReDim ProcessSpec(1)
Open "c:\Factory\Process\files\Proclist.Pfl" For Random Access Read Write Share
 As Filenum Len = Filelength
Get Filenum, 1, ProcessSpec(1)

'else there is something there so add to list
ReDim Preserve ProcessSpec(ProcessSpec(1).Processnumber)

For i = 1 To ProcessSpec(1).Processnumber
    Get Filenum, i, ProcessSpec(i)
Next

'now we have to look for a match between the item that has been selected from t
e option lists
'and the file so that the correct information can be extracted.

ThisCaption = Option1(Index).Caption
For i = 1 To ProcessSpec(1).Processnumber
    If InStr(Trim(ThisCaption), Trim(ProcessSpec(i).Processname)) = 1 Then
    'a match has been found and mark with variable "Matchposition"
    MatchPosition = i
    End If
Next


'Now read the capacity information into the option buttons available
Divisionsize = ProcessSpec(MatchPosition).MaxCap / 10
Machdef = ProcessSpec(MatchPosition).ProcessId
MsgBox ("Mach def is " & CStr(Machdef))
For i = 1 To 10
    Opcap1(i - 1).Caption = CStr((Divisionsize) * (i)) & " mm to " & CStr((Div
ionsize) * (i + 1)) & " mm"
    If i = 10 Then
        Opcap1(9).Caption = "> " & CStr((Divisionsize) * (i + 1)) & " mm"
    End If
```

```
Next

'ReDim Preserve ProcessSpec(ProcessSpec(1).Processnumber)
'For i = 1 To ProcessSpec(1).Processnumber
' Get Filenum, i, ProcessSpec(i)
' Processlist.AddItem ProcessSpec(i).Processname
' MaxCapacity.AddItem CStr(ProcessSpec(i).Maxcap)
' Processid.AddItem CStr(ProcessSpec(i).Processid)
'Next

'CurrId = ProcessSpec(1).CurrentProcessId
Close Filenum


'Now we have to load all the types of tools that are available for this type of
process
'so that the user can choose which types of tools are available for this type o
 machine
'on this vmd

'First clear listbox
ToolsAvle.Clear

Filenum = FreeFile

Filenam = "c:\Factory\System\" & Trim(Option1(Index).Caption) & ".Opt"
EndOfArray = 4

Call ReadToMemory(Filenum, Filenam, Thisstate, EndOfArray)

For i = 1 To EndOfArray
    ToolsAvle.AddItem Optionstatearray(i).OptionCaption
Next


Close Filenum


'Now we must open the file which has the names of all the other tool types
'configured for this particular machine.

Filenum = FreeFile
Recordlength = Len(Thatstate)
Filenam = "c:\Factory\System\" & Trim(Option1(Index).Caption) & ".Old"

'Find out how many records there are in the file
Fileslength = GetFilesLength2(Filenam, Thatstate, OptionList(), Filenum)

'Write the file contents to memory
Call WriteToMemory3(Filenum, Fileslength, OptionList())

'now write the contents of the file into the listbox
For i = 1 To Fileslength
    ToolsAvle.AddItem OptionList(i).OptionName
```

```
Next

Close Filenum

'Tidy up the listbox
For i = 0 To ((ToolsAvle.ListCount) - 1)
    If Len(Trim(ToolsAvle.List(i))) = 0 Then
        ToolsAvle.RemoveItem i
    End If
Next

'Remove common names from listbox
For i = 0 To ((ToolsAvle.ListCount) - 1)
    For j = (i + 1) To ((ToolsAvle.ListCount) - 1)
        If InStr(Trim(ToolsAvle.List(i)), Trim(ToolsAvle.List(j))) = 1 And Len(
rim(ToolsAvle.List(j))) > 0 Then 'there is a match
            ToolsAvle.RemoveItem i
        End If
    Next
Next

Return


'This is all very well but we must also save the users selections to file so th
t this becomes
'the startup selection for the form.


'ThisConfig
Filenum = FreeFile

'Recordlength = Len(Thisconfig)

Filenam = "C:\Factory\System\V" & CStr(TheVMD) & "C" & CStr(NodebeingConfiged)
 "Mach.sys"

MsgBox (Filenam)
Open Filenam For Output As Filenum Len = 100

'Write #Filenum, "Hello"

Write #Filenum, Index

MsgBox (Trim(Option1(Index).Caption))
Write #Filenum, Trim(Option1(Index).Caption)
Close Filenum



Exit Sub
CheckThisError:

If Err = 64 Or 9 Then
'MsgBox (Error)
Exit Sub
```

```
End If

ThisError = ChekError(Err)

Select Case ThisError

Case Is = 2

Msg = "There are no tools of this type configured for this machine type as"
Msg = Msg & " yet. To configure tools for this machine type please goto "
Msg = Msg & "the main menu and select the 'Define Tools' button."

MsgBox Msg, MB_ICONSTOP + MB_OK, "Tools not available"
Exit Sub

Case Else
Resume Next
End Select




End Sub

Sub Proc_List_Click ()


PostprocessorSelected = Trim(Proc_List.List(Proc_List.Listindex))




End Sub

Sub Process_Click ()
Dim Msg As String
Dim typ As Integer
Dim title As String

Msg = "Select the process you want in this VMD, if the process you want is not
vailable then "
Msg = Msg & "you will have to define it in the 'Define Processes' module."
typ = 0
title = "Select Process"
Call Sendmessage(Msg, typ, title)


End Sub

Sub Redefine_tools_Click ()
'this button allows the user to kill the old toolfile associated with this VMD
rior to
```

```
'redefining the tools configured for this VMD.

Dim Msg As String
Dim Reply As Integer
Dim typ As Integer
Dim title As String
Dim Thisfilename As String
Dim TheFileTokill As String


On Error Resume Next


Msg = "Are you absolutely sure that you want to redefine the available tools fo
 this VMD"
title = "Tool Redefinition"
typ = MB_ICONQUESTION + MB_YESNO

Reply = MsgBox(Msg, typ, title)

If Reply = IDNO Then
    Exit Sub
    Else 'if the user wants to kill the file then do so
    Thisfilename = "V" & CStr(TheVMD) & "C" & CStr(NodebeingConfiged) & "TSpc.t
t"
    TheFileTokill = Netdrive & "Factory\Config\Cells\" & Trim(Thisfilename)
    Kill Trim(TheFileTokill)
End If


End Sub

Sub Set_Buff_Click ()
'this procedure allows the user the ability to set the maximum buffer size allo
ed for the vmd
'being considered.This buffer size setting could be in response to some form of
management
'policy such as the theory of constriants for instance.Once the length of time
equired to do the
'work in a VMD's incoming queue is greater than it's buffer size that VMD can n
 longer form
'a bid for any auction as it's ingoing queue is deemed to be full therefore it
s not allowed to
'partake in an auction.

'Declare variables for this procedure
Dim Prompt As String
Dim title As String
Dim DefVal As String
Dim Answer  As Integer
Dim Filenum As Integer
Dim Fileslength As Integer
Dim Filename As String
Dim Currentbuffer As Inqueuebuffersize

On Error Resume Next
```

```
title = "Input buffer size - VMD" & CStr(TheVMD)
Prompt = "Could you please input the desired time buffer size(In minutes) for P
ocess" & CStr(TheVMD)
DefVal = "1440" 'Default value for a VMD is three days of work"
Do
Answer = InputBox(Prompt, title, DefVal)
Loop Until Answer > 0

'now we are required to write the data to a file particular to this node
Filenum = FreeFile 'define a freefile
Fileslength = Len(Currentbuffer) 'define storage variable size in Bytes
Filename = Netdrive & "Cellcont\queue\" & Trim(Thisnode) & "Bfsze.Buf" 'define
ilename

Open Filename For Random Access Read Write Shared As Filenum Len = Fileslength
pen the file for write access

Get Filenum, 1, Thisbuffer(1)

'now put the answer variable in the appropriate part of the array
Select Case TheVMD
Case Is = 1
Thisbuffer(1).VMD1Buffer = Answer

Case Is = 2
Thisbuffer(1).VMD2Buffer = Answer

Case Is = 3
Thisbuffer(1).VMD3Buffer = Answer

Case Is = 4
Thisbuffer(1).VMD4Buffer = Answer


End Select

'now put the information into the file
Put Filenum, 1, Thisbuffer(1)


Close Filenum
'Dim Thisbuffer(1) As InQueueBuffersize
'Type InQueueBuffersize
  '   VMD1Buffer As String
   '  VMD2Buffer As String
    ' VMD3Buffer As String
    ' VMD4Buffer As String
'End Type
'MsgBox (Trim(Thisnode))
'MsgBox (CStr(TheVMD))


End Sub
```

```
Sub Set_Click ()
Dim Msg As String
Dim typ As Integer
Dim title As String

Msg = "Setting the inqueue size here allows the user the ability to set the max
mum size ingoing "
Msg = Msg & "queue in terms of minutes of work allowed."
typ = 0
title = "Setting Inqueue Size"
Call Sendmessage(Msg, typ, title)

End Sub

Sub Show_Processors_Click ()
'Define local variables
Dim Filenum As Integer


'Type Postrecord
'ProcessTyp As String * 50
'Index As Integer
'PostTyp As String * 50
'End Type

'firstly make sure that something sensible was selected from
'the process list

If ChosenIndex = -1 Then 'nothing chosen from tyhe list
    MsgBox "Please select a process before asking to see the Post-Processors av
ilable for it"
    Exit Sub
End If



'secondly make the list visible so that we are able to see post processors
'that are available.

Proc_List.Visible = True


'Type Postrecord
'CurrentIndex As Integer
'CurrentProcId As Integer
'Recordleng As Integer
'ProcessTyp As String * 50
'Index As Integer
'PostTyp As String * 50
'End Type

'Dim ThisProcessPosts As Postrecord
'Dim ThisPostProcsarray() As Postrecord

'Now we must open the file that contains the names of all the post-processors
'for the chosen piece of equipment
```

```
CONFIG_N.FRM - 23

Filenum = FreeFile
ProcTyp = Trim(Option1(Index).Caption)
Filenam = "c:\factory\Process\Postname\" & Trim(ProcTyp) & "proc.pps"
thisfileslength = GetFileslength4(Filenam, ThisProcessPosts, ThisPostProcsarray
 , Filenum)


If thisfileslength < 1 Then
    'there is nothing in the file to add to the listbox therefore exit the sub
    MsgBox "No Post Processors have been defined, Please do so"
    Close Filenum
    Exit Sub
    Else
        'There is something in the file therefore add the information the need
o display
        'to the listbox and write remaining information to memory.
        Call ReadAndAddToList4(ThisPostProcsarray(), thisfileslength, Filenum)
        Close Filenum
End If


End Sub

Sub Show_Sizes_Available_Click ()

'Declare variables

Dim TrimMachineName As String * 3
Dim TrimToolName As String * 4
Dim TrimMach As String * 3
Dim TrimOption As String * 4
Dim Filename As String
Dim Thistool As Toolrecord
Dim Thesetools() As Toolrecord
Dim Filesize As Variant
Dim ListBoxLocation As String
Dim ToolSizeBeingAdded As Variant
Dim Listname As String * 20
Dim ListLength As Integer
Dim IsItemInList As Variant
Dim thisfileslength As Integer
Dim Filenum As Integer
Dim ThatTool As Toolrecord
Dim ThoseTools() As Toolrecord
Dim MainToolist() As ToolIdRecord
Dim ThisToolsId As ToolIdRecord
Dim NoOfTooltypes As Integer
Dim i As Integer
Dim TheRequiredId As Integer
Dim Fileno As Integer
Dim Msg As String
Dim Lastposition As Integer

On Error Resume Next
'This event allows the user to display all the tool sizes that are avaialble fo
a
```

```
'particular type of tool associated with a particular type of process.

'Declare variables

'reset checkbox counter
NoChkBoxes = 0


'first make sure that something has been selected from the listbox before attem
ting
'to load any size data.

If Len(Trim(ToolsAvle.List(ToolsAvle.Listindex))) = 0 Then 'tell the user that h
e has
'not yet selected anything from the listbox.
    MsgBox "You haven't selected anything from the list - Please do so if you w
nt to display the relevant tool sizes.", MB_ICONSTOP, "Selection Error"
    Exit Sub
End If

'First delete any option buttons currently loaded from the last config
If ThisNoOptions > 1 Then 'we have to delete extra option buttons before adding
'new ones for the new tooltype being displayed otherwise an error will occur.
    For i = 1 To (ThisNoOptions - 1)
        Unload Check1(i)
    Next
End If

'This particular procedure opens a file containing all the sizes available of
'the tool selected from the 'ToolsAvle' listbox.

TrimMachineName = Trim(ThisCaption)
MsgBox (TrimMachineName)
TrimToolName = Trim(ToolsAvle.List(ToolsAvle.Listindex))
MsgBox (TrimToolName)

Filename = "C:\Factory\Process\ToolFile\" & TrimMachineName & "!" & TrimToolNam
 & ".tl"

'Declare local variables
Filenum = FreeFile

Filesize = getFileslength(Filename, Thistool, Thesetools(), Filenum)

ReDim Thesetools(Filesize)

For i = 1 To Filesize
    Get Filenum, i, Thesetools(i)
Next

ThisNoOptions = Filesize

Check1(0).Caption = CStr(Thesetools(1).Toolsize) & "mm"

For i = 1 To (Filesize - 1)
    Load Check1(i)
```

```
    Check1(0).SetFocus

    If i <> 13 And i <> 26 Then
        Check1(i).Top = Check1(i - 1).Top + 340
        If i > 13 And i < 26 Then
            Check1(i).Left = 7500
        End If
            If i > 26 Then
                Check1(i).Left = 8800
            End If
    End If

    If i = 13 Then
        Check1(i).Top = Check1(0).Top
        Check1(i).Left = 7500

    End If

    If i = 26 Then
        Check1(i).Top = Check1(0).Top
        Check1(i).Left = 8800
    End If

    Check1(i).Caption = CStr(Thesetools(i + 1).Toolsize) & "mm"
    Check1(i).Visible = True
Next

'record for later how many checkboxes are being used
NoChkBoxes = i
MsgBox (CStr(NoChkBoxes))
'Type Toolrecord
'Recordleng As Integer
'Toolsize As Single
'ToolId As Integer
'End Type


Close Filenum




End Sub

Sub Size_Click ()
Dim Msg As String
Dim typ As Integer
Dim title As String
```

```
Msg = "After selecting the tool sizes that are available to be used by this pro
ess for a particular "
Msg = Msg & "tool type click the 'size ok' button to confirm your selection."
typ = 0
title = "Confirm Sizes"
Call Sendmessage(Msg, typ, title)

End Sub

Sub SizeOk_Click ()
'This button allows the user the ability to say that a particular size selecti
 for a particular
'tooltype as selected 'by him/her is ok.It then saves that selection to a file.
s each tool types
'size selection is saved to file it is added to any previous data entered.

'Declare variables
Dim ThistoolSpec() As Toolspec
Dim Thistool As Toolspec
Dim Thisfilename As String * 30
Dim Position As Integer
Dim Endposition As Integer
Dim thisfileslength As Integer
Dim i As Integer
Dim Filenum As Integer
Dim Recordlength As Integer
Dim TheDirectory As String * 27
Dim ThisFile As String * 100
Dim Msg As String
Dim typ As Integer
Dim title As String
Dim Reply As Integer
Dim Filelength As Integer

'Type ProcessDef
'ProcessNumber As Integer
'Processname As String * 20
'CurrentProcessId As Integer
'ProcessId As Integer
'Maxcap As Integer
'End Type
Dim Request As String
Dim Answer As String * 20
Dim OptionNumber As Integer
Dim CheckValidity As Integer
Dim Filename As String
Dim MainToolist() As ToolIdRecord
Dim ThisToolsId As ToolIdRecord
Dim NoOfTooltypes As Integer
Dim Result1 As Integer
Dim Result2 As Integer
Dim Marker As Integer
Dim TheToolsId As Integer
Dim Needtoaddit As Integer
Dim TheToolSpec As Single
Dim j As Integer
```

```
On Error Resume Next

'First clear the listbox
List1.Clear

'When we have the process's Id(we get this from the option button when it is se
ected) we get
'the capacity of the process from the form and then this information is piped t
 the
'relevant vmd using EDA.


MachineDef = Machdef + OptionIndex / 10

'This machine definition information must be sent to the relevant VMD but will
nly be done
'when the user has confirmed that he wants it to be configured.


'First get the total number of tool types that have already been specified.
Filename = "c:\factory\system\AlToolTyps.Tl"
Filenum = FreeFile
NoOfTooltypes = GetFilesLength1(Filename, ThisToolsId, MainToolist(), Filenum)

'now write the list to memory
Call WriteToMemory(Filenum, NoOfTooltypes, MainToolist())

'Type ToolIdRecord
'CurrentToolId As Integer
'recordlength As Integer
'ToolType As String * 20
'ToolId As Integer
'End Type

For i = 1 To NoOfTooltypes 'if there is a match between the tool selected from t
he listbox
                           'and that in the file
    If InStr(Trim(MainToolist(i).ToolType), Trim(ToolsAvle.List(ToolsAvle.Listi
dex))) = 1 Then
        'then mark this position with a marker
        Marker = i
    End If
Next

'we are trying to get a match between the item selected from the listbox and tha
t in the file to
'get a sensible tool identification number

'Now that we have found a match we have found the tool id
TheToolsId = MainToolist(Marker).Toolid
MsgBox (CStr(TheToolsId))
'We have now found the type of tool being used so we must save this information
into file so we
 will create an array of type
```

```
'Type ToolSpec
'!'Toolname As String * 40
'ToolTypeSize As Single
'NoOfToolsNeeded As Integer
'End Type

Close Filenum 'Close the file

'Comment the array being worked with so that i can see it and refer to it
'without it disrupting my program.

'Type ToolSpec
'Toolname As String * 40
'ToolTypeSize As Single
'NoOfToolsNeeded As Integer
'End Type

'Define a free number to reference a new file that we require to be opened
Filenum = FreeFile

'Declare array record length
Recordlength = Len(Thistool)

'Define the name of the file to open

'opening a file that will store the available tools information for this partic
lar VMD and Cell
'combination

Thisfilename = "V" & CStr(TheVMD) & "C" & CStr(NodebeingConfiged) & "TSpc.txt"
'ie:VMD1 Cell1 Tool Specification
TheDirectory = Netdrive & "Factory\Config\Cells\"
ThisFile = Trim(TheDirectory) & Trim(Thisfilename)

'Open the file for read\write random access
Open Trim(ThisFile) For Random Access Read Write Shared As Filenum Len = Record
ength

'redimension the array to take the first record
ReDim ThistoolSpec(1)

'get the length of the file
Get Filenum, 1, ThistoolSpec(1)

'this is the current length of the file however we are required to make enough
oom to store all
'the check box information that has been checked.
thisfileslength = ThistoolSpec(1).NoOfToolsNeeded
MsgBox (CStr(thisfileslength))


'Type ToolSpec
'Toolname As String * 40
'ToolTypeSize As Single
'NoOfToolsNeeded As Integer
'End Type
```

```
ReDim Preserve ThistoolSpec(thisfileslength)

'get all of the tool specs so far accumulated
For i = 1 To thisfileslength
    Get Filenum, i, ThistoolSpec(i)
    List1.AddItem CStr(ThistoolSpec(i).ToolTypesize)
Next

'now we will see which check boxes are checked and which are not checked, if th
y are checked
'then they either needed to be added to the listbox or if they are already in t
e listbox then
'not added to it.

For i = 1 To (NoChkBoxes)
    If Check1(i - 1).Value = 1 Then 'the box has been checked
        TheToolSpec = TheToolsId + (Val(Check1(i - 1).Caption)) / 100
        Needtoaddit = CheckTheList(TheToolSpec)
        If Needtoaddit = 2 Then
            List1.AddItem CStr(TheToolSpec)
        End If
    End If
Next

'However if it is not checked but is in the list we no longer want it and must
'delete it from the list
For i = 1 To (NoChkBoxes)
   If Check1(i - 1).Value = 0 Then 'if it is not checked
      TheToolSpec = TheToolsId + (Val(Check1(i - 1).Caption)) / 100
Startagain:
         For j = 0 To (List1.ListCount - 1)
            If (CSng(List1.List(j)) - TheToolSpec) = 0 Then 'if it is not check
d but it
           'is in the list we need to remove it from the list
               List1.RemoveItem j
            GoTo Startagain
            End If
        Next
    End If
Next

Close Filenum

'now we want to kill the old version of the file and save the new one.

Kill ThisFile

'Open the file for read\write random access
Open Trim(ThisFile) For Random Access Write Shared As Filenum Len = Recordlengt

'redimension the array to take the first record
ReDim ThistoolSpec(List1.ListCount)

'Type ToolSpec
'!!Toolname As String * 40
```

```
Msg = "After selecting the tool sizes that are available to be used by this pro
ess for a particular "
Msg = Msg & "tool type click the 'size ok' button to confirm your selection."
typ = 0
title = "Confirm Sizes"
Call Sendmessage(Msg, typ, title)

End Sub

Sub SizeOk_Click ()
'This button allows the user the ability to say that a particular size selectio
 for a particular
'tooltype as selected 'by him/her is ok.It then saves that selection to a file.
s each tool types
'size selection is saved to file it is added to any previous data entered.

'Declare variables
Dim ThistoolSpec() As Toolspec
Dim Thistool As Toolspec
Dim Thisfilename As String * 30
Dim Position As Integer
Dim Endposition As Integer
Dim thisfileslength As Integer
Dim i As Integer
Dim Filenum As Integer
Dim Recordlength As Integer
Dim TheDirectory As String * 27
Dim ThisFile As String * 100
Dim Msg As String
Dim typ As Integer
Dim title As String
Dim Reply As Integer
Dim Filelength As Integer

'Type ProcessDef
'ProcessNumber As Integer
'Processname As String * 20
'CurrentProcessId As Integer
'ProcessId As Integer
'Maxcap As Integer
'End Type
Dim Request As String
Dim Answer As String * 20
Dim OptionNumber As Integer
Dim CheckValidity As Integer
Dim Filename As String
Dim MainToolist() As ToolIdRecord
Dim ThisToolsId As ToolIdRecord
Dim NoOfTooltypes As Integer
Dim Result1 As Integer
Dim Result2 As Integer
Dim Marker As Integer
Dim TheToolsId As Integer
Dim Needtoaddit As Integer
Dim TheToolSpec As Single
Dim j As Integer
```

```
'ToolTypeSize As Single
 NoOfToolsNeeded As Integer
'End Type

'histoolSpec(1).NoOfToolsNeeded = List1.ListCount

For i = 1 To ThistoolSpec(1).NoOfToolsNeeded
    ThistoolSpec(i).ToolTypesize = CSng(List1.List(i - 1))
Next

or i = 1 To ThistoolSpec(1).NoOfToolsNeeded
    Put Filenum, i, ThistoolSpec(i)
Next

lose Filenum

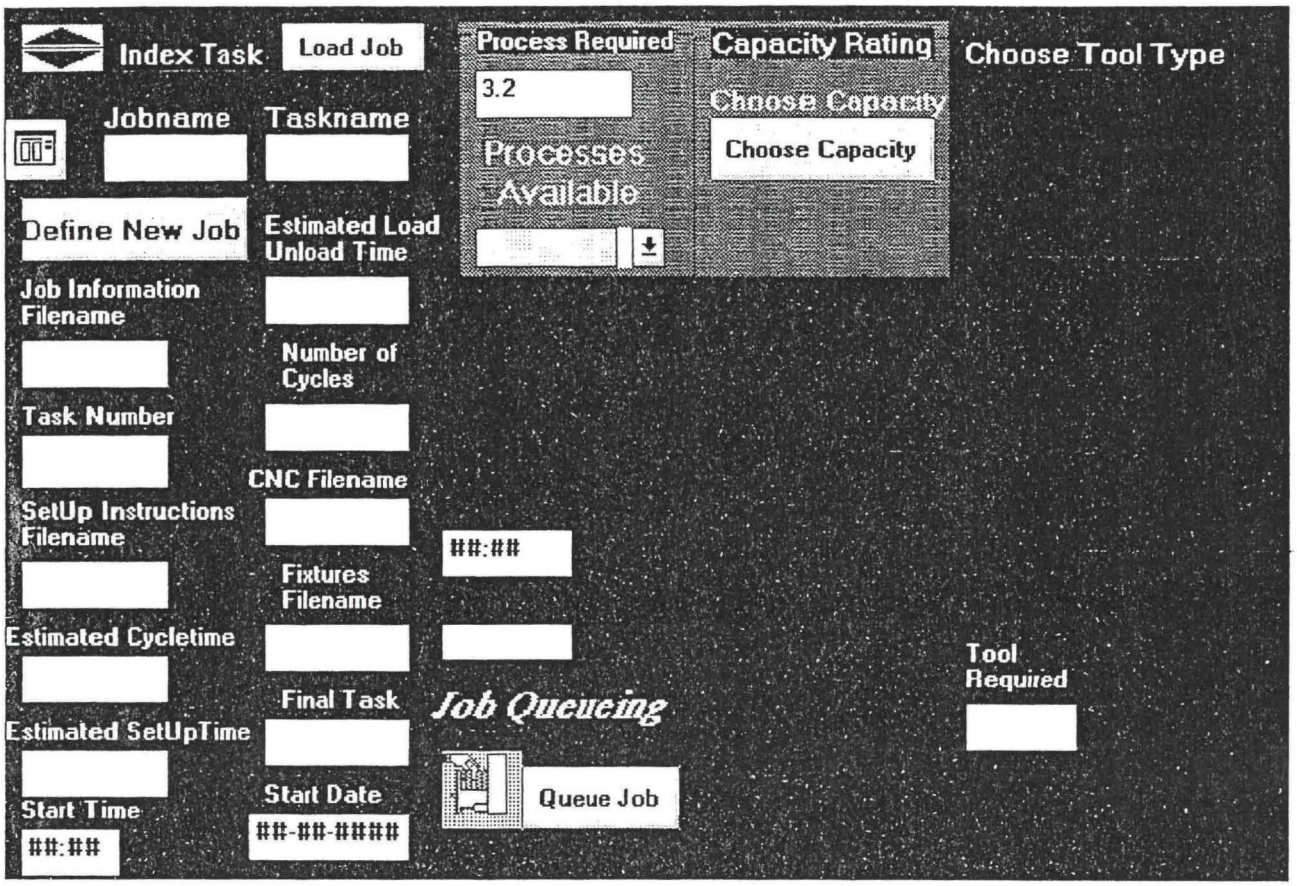
Tidy up the  listbox
or i = 0 To List1.ListCount
    If CInt(List1.List(i)) = 0 Then
        List1.RemoveItem (i)
    End If
Next


d Sub

Sub View_Click ()
im Msg As String
im typ As Integer
Dim title As String

sg = "Select a tooltype that is available for the process type setup in this VM
 and then select"
sg = Msg & " 'View Sizes' to show the sizes available for this type of process
 the Factory. "
yp = 0
itle = "View Sizes"
ll Sendmessage(Msg, typ, title)

End Sub
```


Index Task
Load Job
Jobname
Taskname
Define New Job
Estimated Load Unload Time
Job Information Filename
Number of Cycles
Task Number
CNC Filename
SetUp Instructions Filename
##:##
Fixtures Filename
Estimated Cycletime
Final Task
Job Queueing
Estimated SetUpTime
Queue Job
Start Date
##-##-####
Start Time
##:##
Process Required
3.2
Processes Available
Capacity Rating
Choose Capacity
Choose Capacity
Choose Tool Type
Tool Required

```
DEF_JOB.FRM - 1

Option Explicit
Option Base 1
'This is the form that allows the operator of the package
'to define new jobs that are required to be made
Dim Textboxchoice As Integer
Dim Force As Integer
Dim Skip As Integer
Dim ChangehasOccurred As Integer
Dim Loading As Integer

Sub AddProcess_Click ()
MachRequired.Text = Machinelist.List(Machinelist.ListIndex)
End Sub

Sub AddTool_Click ()

End Sub

Function CheckError (errtyp As Integer)
Dim types As Integer
Dim Titles As String * 20
Dim Msg As String
Dim response As Integer

types = 0

Select Case errtyp

Case Is = ERR_TYPEMISMATCH

Titles = "Task Data Incomplete"
Msg = "The task data that you have described is either incomplete or the data"
Msg = Msg & " type entered is incorrect. Please refer to the help file regardinc
"
Msg = Msg & " the correct data type entries that should be used for each field.'

Case Is = ERR_PATHNOTFOUND
Titles = "JobName not specified"
Msg = "You not entered the name of the job you are configuring"
Msg = Msg & " this task for. Please do so"

Case Is = ERR_SUBSCRIPTOUTOFRANGE
CheckError = 2
Exit Function

Case Is = ERR_FILENOTFOUND
Titles = "Bitmap in Wrong Directory"
Msg = "The Bitmap called " & Machinelist.List(Machinelist.ListIndex) & ".bmp eit
her does "
Msg = Msg & "not exist or is in the wrong directory. All bitmap files must be ho
used "
Msg = Msg & "in the 'c:\factory\process\specs\' directory"


Case Else
```

```
Msg = "An error of type " & Error & " is occurring"



End Select

response = MsgBox(Msg, types, Titles)


Select Case response

Case 1 'Ok was selected
CheckError = 1
End Select

End Function

Function ChekError (errtyp As Integer) As Integer

Dim types As Integer
Dim Titles As String * 20
Dim Msg As String
Dim response As Integer

types = 0

Select Case errtyp

Case Is = ERR_TYPEMISMATCH

Titles = "Task Data Incomplete"
Msg = "The task data that you have described is either incomplete or the data"
Msg = Msg & " type entered is incorrect. Please refer to the help file regardi
"
Msg = Msg & " the correct data type entries that should be used for each field."

Case Is = ERR_PATHNOTFOUND
Titles = "JobName not specified"
Msg = "You not entered the name of the job you are configuring"
Msg = Msg & " this task for. Please do so"

Case Is = ERR_SUBSCRIPTOUTOFRANGE
ChekError = 2
Exit Function

Case Is = ERR_BADFILENAME Or 9
Exit Function


Case Else
Msg = "An error of type " & Error & " is occurring"



End Select
```

```
)EF_JOB.FRM - 3

:esponse = MsgBox(Msg, types, Titles)


Select Case response

Case 1 'Ok was selected
ChekError = 1
nd Select

End Function

Sub Command3_Click ()
'This command button allows the user to create new capacity
 forms using an array - this is required so that new
 machines not previously considered in the design of the
'Factory Controller can be integrated into the system
'more easily and have their capacity forms custom made
Dim Msg As String
Dim Reply As Integer
)im Typ As Integer
)im Title As String


'Create form array variable
)im NewMachTyp As New CNC_Drill
lewMachTyp.Show
NewMachTyp.Caption = Machinelist.List(Machinelist.ListIndex)



'NewMachtyp.Text1.Text = "Hi"




End Sub

Sub DueDate_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As S
ngle)
'ext1.Text = "This textbox is where the user enters the date at which the job s
:uld be started - this is only entered once when the first task is defined."
Text2.Text = "Enter the data here eighth."
nd Sub

Sub MaskedEdit1_ValidationError (InvalidText As String, StartPosition As Intege

IsgBox "The data is in the wrong format enter the hour first and then the minut
 second."

nd Sub
```

```
DEF_JOB.FRM - 4

Sub Tool2rq_Change ()

On Error Resume Next

Tool3rq.Enabled = True
Textboxchoice = 3
ToolType.SetFocus


End Sub

Sub Tool2rq_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As S
ngle)
'ChangehasOccurred = 1
Textboxchoice = 2
End Sub

Sub Tool2rq_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As S
ngle)
Text1.Text = "Clicking this textbox and then selecting a toolsize from the 'Ch
se Tool Size'list will assign that tool to this box."
Text2.Text = "Never type in this box manually,System failure could result!!!"

End Sub

Sub Tool3rq_Change ()

On Error Resume Next

Tool4rq.Enabled = True
Textboxchoice = 4
ToolType.SetFocus




End Sub

Sub Tool3rq_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As S
ngle)
'ChangehasOccurred = 1
Textboxchoice = 3
End Sub

Sub Tool3rq_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Si
ngle)
Text1.Text = "Clicking this textbox and then selecting a toolsize from the 'Cho
se Tool Size'list will assign that tool to this box."
Text2.Text = "Never type in this box manually,System failure could result!!!"
End Sub

Sub Tool4rq_Change ()
On Error Resume Next
Command1.Enabled = True
```

```
Command1.SetFocus
:nd Sub

Sub Tool4rq_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As S
igle)
 ChangehasOccurred = 1
Textboxchoice = 4
:nd Sub

Sub Tool4rq_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As S
igle)
'ext1.Text = "Clicking this textbox and then selecting a toolsize from the 'Cho
se Tool Size'list will assign that tool to this box."
'ext2.Text = "Never type in this box manually,System failure could result!!!"
:nd Sub

Sub ToolList_DblClick ()
.ddTool_Click
End Sub

ub Cad_Cam_Sel_Click ()
'This particular procedure lets the user design the task
'that he wants to define in a Cad\Cam package - he does this
by selecting the name of the Cad\Cam package that he wants
'to use and then clicking 'Run'.


if Len(Trim(Jobname.Text)) = 0 Then
    MsgBox "You must state the name of the job before the Cad\Cam function can
 executed.", MB_ICONEXCLAMATION + MB_OK, "Define Jobname"
Exit Sub
End If


'First we must make sure that a  process type has been
 selected so that we know what types of Post-Processors
 we have to load into the next form, telling the operator
'what types of post-processors he has to run for his
 drawing.

If Len(Trim(Machinelist.List(Machinelist.ListIndex))) = 0 Or Len(Trim(Taskname.
xt)) = 0 Then
 nothing has been selected therefore prompt the user
    MsgBox "Please select a process type to be used for this task before select
g this button", MB_ICONSTOP + MB_OK, "Selection Error"
xit Sub
End If

 now that everything is in order we must transfer the name of the global variab
e to the
'Cad_Cam form so that it can be read by the user so that he can name the 'nci'
 le by
'the correct name when he is toolpathing.
CNCncifname = CNCFname.Text
```

```
Load Cad_cam
Cad_cam.Show
FixturesFname.Enabled = True

End Sub

Sub Cad_Cam_Sel_MouseMove (Button As Integer, Shift As Integer, X As Single, Y
s Single)
Text1.Text = "This button allows the user to select a CAD\CAM package and run i
 to define the NC code required for this task."
Text2.Text = "Enter the data here sixth."
End Sub

Sub CNCFname_Change ()

'ChangehasOccurred = 1
End Sub

Sub CNCFname_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As
ingle)
Text1.Text = "This textbox gives the name of the NC file for this task and is a
tomatically filled when the NC code is developed under 'Select Package'"
Text2.Text = "Never enter data here manually!!"
End Sub

Sub Command1_Click ()
Dim Msg As String
Dim Reply As Integer
Dim Title As String
Dim Typ As Integer
Dim ProcessSpc() As ProcessDef
Dim Processtp As ProcessDef
Dim Filelength As Integer
Dim i As Integer
Dim Filenum As Integer
Dim ThisFilenum As Integer
Dim IncrementSize As Integer
Dim Thismatches As Integer
Dim CapIncrement As Integer


On Error GoTo Errorcheck

Filenum = FreeFile

'Check how many bytes required to store array variables
Filelength = Len(Processtp)

'Redimension array to get filelength
ReDim ProcessSpc(1)

'Open the file
Open "c:\Factory\Process\files\Proclist.Pfl" For Random Access Read Write Share
 As Filenum Len = Filelength

'Get the number of the last process added so that all of the information in
```

```
)EF_JOB.FRM - 7

'the file is able to be loaded into memeory
Get Filenum, 1, ProcessSpc(1)

'Redimension the array so that all the process data defined so far can be
 recalled
ReDim Preserve ProcessSpc(ProcessSpc(1).Processnumber)

 Read all the process data so far defined into the memory array
For i = 1 To ProcessSpc(1).Processnumber
Get Filenum, i, ProcessSpc(i)
Next

'Type ProcessDef
'ProcessNumber As Integer
 Processname As String * 20
'CurrentProcessId As Integer
'ProcessId As Integer
'Maxcap As Integer
'End Type

'Now we are required to get a match between the item as selected from the listbo
x
'and that in the file so that we know what capacity information to load
For i = 1 To ProcessSpc(1).Processnumber
    If InStr(Trim(ProcessSpc(i).Processname), Trim(Machinelist.List(Machinelist.
ListIndex))) = 1 Then
        'there is a match
        Thismatches = i
    End If
Next

CapIncrement = (ProcessSpc(Thismatches).MaxCap) / 10
ThisMachtype = ProcessSpc(Thismatches).ProcessId
MsgBox (CStr(ThisMachtype))

Close Filenum


'This procedure allows for the displaying of the correct capacity information
'for the process required for the task being configured.Once this information
'is displayed the user can then select the minimum required capacity from
'that processe's capacity list in order to do the job.

'Assign the machine selected from the process list to a variable called
'MachSelected

MachSelected = Machinelist.List(Machinelist.ListIndex)

'MsgBox (MachSelected)
Select Case Machinelist.List(Machinelist.ListIndex)

'Check to see if a selection has been made
Case Is = ""
Title = "Choose a Process!"
Msg = "You cannot select a machine's capacity without defining what type of "
Msg = Msg & "machine you require for the task. Please select a machine type "
```

```
Msg = Msg & "from the 'Processes Available' list and then reselect the 'Choose
apacity' button. "
Typ = MB_ICONEXCLAMATION + MB_OK
Reply = MsgBox(Msg, Typ, Title)
Exit Sub

Case Is = "CNC Drill"
Load CNC_Drill
CNC_Drill.Show

Case Else

'Create form array variable.A form array variable allows the programmer to
'construct a program that only uses one form to display different things
'at different times thus conserving memory by not having to keep multiple
'forms in memeory
Dim NewMachTyp As New CNC_Drill

'Show new instance of the form
NewMachTyp.Show

'Assign the name of the chosen machine to the form's caption
NewMachTyp.Caption = Machinelist.List(Machinelist.ListIndex)

'Load the capacity bitmap for the machine being considered into the form's
'object linking and embedding control
ThisFilenum = FreeFile

Open "C:\Factory\Process\Specs\" & Trim(Machinelist.List(Machinelist.ListIndex)
 & ".bmp" For Binary As #ThisFilenum
NewMachTyp.OLE1.FileNumber = ThisFilenum
NewMachTyp.OLE1.Action = 12
Close ThisFilenum


For i = 1 To 10
    NewMachTyp.CNCCaplist.AddItem (CStr((i - 1) * CapIncrement) & " - " & CStr(
apIncrement * i) & "mm")
Next

End Select

Exit Sub
Errorcheck:
Reply = CheckError(Err)

Select Case Reply

Case Is = 1
''''''''''''''Unload NewMachTyp
Exit Sub

Case Is = 2
Resume Next

Case Else
```

```
ff_JOB.FRM - 9

xit Sub

nd Select


nd Sub

ub Command2_Click ()
im Msg As String * 64
Dim Reply As Integer
im Title As String * 14
im Typ As Integer
Dim Readtask As JobRecord
im Filenum As Integer
im Recordlength As Integer
Dim Taskposition As Integer
im LoadTasks() As JobRecord
onst OPEN_DIALOG = 1


n Error Resume Next
Loading = 1
'This particular button allows the user to select the name
of a jobfile and then load that job's first task into the
'appropriate textboxes

f ChangehasOccurred = 1 Then
'a change has occurred therefore tell the user that such a change has occurred
'and ask him whether or not he wants to save those changes.
  Msg = "A change has occurred in this tasks definition - Save Changes?"
  Title = "Change Status"
  Typ = MB_ICONSTOP + MB_YESNO
  Reply = MsgBox(Msg, Typ, Title)
     If Reply = Idyes And CInt(Tasknumber.Text) > 1 Then 'if the user
     'wants to save the current task and the current tasknumber is greater
     'than 1 we can spindown to save the current task that has been changed.
     Spin1_SpinDown
         Else
         If Reply = Idyes And CInt(Tasknumber.Text) = 1 Then 'if the user
         'wants to save the current task and the current task is equal to 1
         'then the program will have to spinup to save the current task.
         Spin1_Spinup
            Else
             ' if the user does not want to save the
            'changes then dont do anything.
         End If
     End If
nd If


'Set the file choice filter
.lehandler.Filter = "JobFiles (*.iff)|*.iff"
Set the common dialog box as open mode
```

```
DEF_JOB.FRM - 10

Filehandler.Action = OPEN_DIALOG


'CHECK TO SEE IF ANY FILE WAS SELECTED
If Len(Trim(Filehandler.Filename)) = 0 Then
Exit Sub
End If


'Open the file and read the first record into the textboxes
'as required
Filenum = FreeFile
Recordlength = Len(Readtask)


MachRequired.Text = Filehandler.Filename

ReDim LoadTasks(1)

Open Filehandler.Filename For Random Access Read Shared As Filenum Len = Record
ength

Get Filenum, 1, LoadTasks(1)

Tasknumber.Text = "1"
Jobname.Text = LoadTasks(1).Jobname
JobInf.Text = LoadTasks(1).Infofilename
MachRequired.Text = LoadTasks(1).Machinetype
Taskname.Text = LoadTasks(1).Taskname
EstCycTime.Text = LoadTasks(1).EstCycletime
EstSetUpM.Text = LoadTasks(1).Setuptime
EstLoadUn.Text = LoadTasks(1).EstLoadUnloadtime
NoOfCycles.Text = LoadTasks(1).NoOfCycles
CNCFname.Text = LoadTasks(1).CNCFilename
Start_Time.Text = LoadTasks(1).Starttime
Tool1rq.Text = LoadTasks(1).Tool1required
'Tool2rq.Text = Loadtasks(1).Tool2required
'Tool3rq.Text = Loadtasks(1).Tool3required
'Tool4rq.Text = Loadtasks(1).Tool4required
SupInstructionsFname.Text = LoadTasks(1).SetUpInstructionsfilename
FixturesFname.Text = LoadTasks(1).Fixturesfilename
'MsgBox (CStr(LoadTasks(1).Status))

'Text5.Enabled = True
'Text5.Visible = True
'Text5.Text = LoadTasks(1).Duedate
Duedate.Text = Trim(LoadTasks(1).Duedate)
'Text4.Text = LoadTasks(1).Duedate
Finaltask.Text = LoadTasks(1).Finaltask
Text3.Text = LoadTasks(1).Force
RqCompdate.Text = Trim(LoadTasks(1).RequiredCompDate)
RqCompTime.Text = Trim(LoadTasks(1).Requiredcomptime)


  '  StartTime As String * 5
   ' Duedate As String * 15
```

```
    '  RequiredCompDate As String * 15
     ' RequiredComptime As String * 5


Close Filenum

ChangehasOccurred = 3


End Sub

Sub Def_fix_Click ()
Dim Msg As String * 64
Dim Reply As Integer
Dim Title As String * 14
Dim Typ As Integer
Dim Readtask As JobRecord
Dim Filenum As Integer
Dim Recordlength As Integer
Dim Taskposition As Integer
Dim LoadTasks() As JobRecord
Const OPEN_DIALOG = 1


On Error Resume Next


Filenum = FreeFile
Recordlength = Len(Readtask)


ReDim LoadTasks(1)

Open "R:\Factory\Jobfiles\Tanya.iff" For Random Access Read Shared As Filenum Le
n = Recordlength

Get Filenum, 1, LoadTasks(1)

'Type JobRecord
 '    Jobname As String * 13
   '  InFoFilename As String * 20
   '  FinalTask As Integer
     ' Tasknumber As Integer
'     Machinetype As Single 'Type as 3.1 the ".1" represents capacity
  '    Taskname As String * 22
   '  EstCycletime As Integer
   '  Setuptime As Integer
'      Estloadunloadtime As Integer
'     NoofCycles As Integer
'     CNCfilename As String * 13
  '    Geometricfilename As String * 13
'     Tool1required As Single
     'Tool2required As Single
     'Tool3required As Single
     'Tool4required As Single
'     Fixturesfilename As String * 13
```

```
'    Status As Integer
'    StartTime As String * 5
'    Duedate As String * 15
'    RequiredCompDate As String * 15
'    RequiredComptime As String * 5
'    Force As Integer
'    SetUpInstructionsFilename As String * 20 'Shows datum graphically
'End Type


MsgBox ("The name of the job is " & Trim(LoadTasks(1).Jobname))
MsgBox ("The name of the infofile is " & Trim(LoadTasks(1).Infofilename))
MsgBox ("The number of the finaltask is " & Trim(CStr(LoadTasks(1).Finaltask)))
MsgBox ("The name of the tasknumber is " & Trim(CStr(LoadTasks(1).Tasknumber)))

MsgBox ("The Machinetype is " & Trim(CStr((LoadTasks(1).Machinetype))))
MsgBox ("The cycletime is " & Trim(CStr(LoadTasks(1).EstCycletime)))
MsgBox ("The setuptime is " & Trim(CStr(LoadTasks(1).Setuptime)))
MsgBox ("The load time is " & Trim(CStr(LoadTasks(1).EstLoadUnloadtime)))



MsgBox ("The number of cycles is " & Trim(CStr(LoadTasks(1).NoOfCycles)))
MsgBox ("The name of the NC File  is " & Trim(LoadTasks(1).CNCFilename))
MsgBox ("The number of the Geo Filename is " & Trim(LoadTasks(1).GeometricFilen
me))
MsgBox ("The name of the tool is " & Trim(LoadTasks(1).Tool1required))

MsgBox ("The name of the fixtures is " & Trim(LoadTasks(1).Fixturesfilename))
MsgBox ("The status is " & Trim(CStr(LoadTasks(1).Status)))
MsgBox ("The startdate of the job is " & Trim(LoadTasks(1).Duedate))
MsgBox ("The finishdate the job is " & Trim(LoadTasks(1).RequiredCompDate))

MsgBox ("The finishtime of the job is " & Trim(CStr(LoadTasks(1).Requiredcompti
e)))
MsgBox ("The force of the job is " & Trim(LoadTasks(1).Force))
MsgBox ("The setupinstructions are " & Trim(LoadTasks(1).SetUpInstructionsfilen
me))

Close Filenum

End Sub

Sub Def_fix_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As S
ngle)
Text1.Text = "This button allows the user to define the required fixtures for t
e task."
Text2.Text = "Execute this function seventh."
End Sub

Sub DefNewJob_Click ()



On Error GoTo ChekError
```

```
Dim i As Integer
Dim Reply As Integer

'This gives the user the ability to delete all of the text in the current
'task textboxes so as to re-initialise a job so as to start a job definition
'from scratch.

'Make sure that the current jobfile has been saved in it's entirity
If Val(Tasknumber.Text) = 1 Then
    Spin1_Spinup
    'If Skip = 1 Then
        'Skip = 0
        'MsgBox ("You Cannot exit until all fields are valid!")
    GoTo ClearInputs
            'End If
    Else
        If Val(Tasknumber.Text) = 0 Then
            GoTo ClearInputs

            Else
            Spin1_SpinDown
                'If Skip = 1 Then
                'Skip = 0
                'MsgBox ("You Cannot exit until all fields are valid!")
            GoTo ClearInputs
        End If
    'Unload DEF_JOB
End If


ClearInputs:

'This part of the routine is responsible for clearing all of the inputs in
'the textboxes ready for the input of new information.

'JobName.Text = ""
'TaskName.Text =

Text4.Text = Duedate.Text

'Declare a control array
Dim AnyTextbox As textbox

Set AnyTextbox = Jobname
Call ClearTextbox(AnyTextbox)

Set AnyTextbox = Taskname
Call ClearTextbox(AnyTextbox)

Set AnyTextbox = EstLoadUn
Call ClearTextbox(AnyTextbox)

Set AnyTextbox = NoOfCycles
Call ClearTextbox(AnyTextbox)

Set AnyTextbox = CNCFname
```

```
Call ClearTextbox(AnyTextbox)

Set AnyTextbox = FixturesFname
Call ClearTextbox(AnyTextbox)

Set AnyTextbox = Finaltask
Call ClearTextbox(AnyTextbox)
'
Set AnyTextbox = EstSetUpM
Call ClearTextbox(AnyTextbox)

Set AnyTextbox = EstCycTime
Call ClearTextbox(AnyTextbox)

Set AnyTextbox = SupInstructionsFname
Call ClearTextbox(AnyTextbox)

Set AnyTextbox = Tasknumber
Call ClearTextbox(AnyTextbox)

Set AnyTextbox = JobInf
Call ClearTextbox(AnyTextbox)

Set AnyTextbox = Tool1rq
Call ClearTextbox(AnyTextbox)

Set AnyTextbox = Tool2rq
Call ClearTextbox(AnyTextbox)

Set AnyTextbox = Tool3rq
Call ClearTextbox(AnyTextbox)

Set AnyTextbox = Tool4rq
Call ClearTextbox(AnyTextbox)

Duedate.Text = "__-__-____"
RqCompdate.Text = "__-__-____"

Start_Time.Text = "__:__"
RqCompTime.Text = "__:__"

Text5.Visible = False
'Now activate the first textbox that is required to be filled in so that the  us
er knows which
'textbox he must fill in first
Tasknumber.Enabled = True
Tasknumber.SetFocus
Tasknumber.Text = "1"



'Now exit the sub so that the errorchecker isnt fired
'automatically


Exit Sub
```

```
'Err_JOB.FRM - 15

ChekError:

Reply = ChekError2(Err)

Select Case Reply

Case Is = 1
Exit Sub

End Select



End Sub

Sub Duedate_Change ()
Finaltask.Enabled = True

If Loading = 1 Then
    Loading = 3
    Exit Sub
    Else
        Text5.Text = Duedate.Text
End If

End Sub

Sub DueDate_GotFocus ()
Duedate.BackColor = &H8080FF
MsgBox ("Please type in the new due date as required.")
End Sub

Sub DueDate_LostFocus ()
Duedate.BackColor = &HFFFF&
End Sub

Sub Duedate_ValidationError (InvalidText As String, StartPosition As Integer)

'When we clear the MaskedEdit box prior to defining a new job a validation erro
occurs which
causes this event to be fired therefore we must trap this error and respond to
it.

If Len(Trim(InvalidText)) = 0 Then
Dont Panic!! we are just clearing the textbox prior to defining a new job
Exit Sub
End If

MsgBox "Date Entered is not in a valid format, it should be entered in a format
such as '09-1-66'."



End Sub
```

```
Sub EstCycTime_Change ()
SupInstructionsFname.Enabled = True
'ChangehasOccurred = 1
End Sub

Sub EstCycTime_GotFocus ()
EstCycTime.BackColor = &H8080FF
End Sub

Sub EstCycTime_LostFocus ()
EstCycTime.BackColor = &HFFFF&
End Sub

Sub EstCycTime_MouseMove (Button As Integer, Shift As Integer, X As Single, Y A
 Single)
Text1.Text = "Enter the estimated time required to perform this task."
Text2.Text = "Enter the data here eleventh."

End Sub

Sub EstLoadUn_Change ()
'ChangehasOccurred = 1
NoOfCycles.Enabled = True

End Sub

Sub EstLoadUn_GotFocus ()
EstLoadUn.BackColor = &H8080FF
End Sub

Sub EstLoadUn_LostFocus ()
EstLoadUn.BackColor = &HFFFF&

End Sub

Sub EstLoadUn_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As
Single)
Text1.Text = "Fill this text box with an estimate of the workpiece loading and u
nloading time in units of minutes."
Text2.Text = "Enter the data here fourth."
End Sub

Sub EstSetUpM_Change ()
EstCycTime.Enabled = True
'ChangehasOccurred = 1
End Sub

Sub EstSetUpM_GotFocus ()
EstSetUpM.BackColor = &H8080FF
End Sub

Sub EstSetUpM_LostFocus ()
EstSetUpM.BackColor = &HFFFF&
End Sub

Sub EstSetUpM_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As
```

```
Single)
Text1.Text = "Enter the estimated time required to set up the process to perfor
 this task."
Text2.Text = "Enter the data here tenth."
End Sub

Sub Exit_Click ()

Dim Msg As String
Dim types As Integer
Dim Title As String * 21
Dim Reply As Integer

 SetUpInsfname = SupInstructionsFname.Text
 Check = 1

On Error GoTo CheckErrorss

types = MB_YESNO + MB_ICONQUESTION
Msg = "Do you really want to leave the 'Job Definition' window"
Title = "Job Definition Exit"
Reply = MsgBox(Msg, types, Title)


If Reply = IDNO Then
Exit Sub
    Else
    If Val(Tasknumber.Text) = 0 Then
            Unload DEF_JOB
            Exit Sub
                Else
                'Make sure that the current jobfile has been saved in it's enti
ty
                If Val(Tasknumber.Text) = 1 Then
                Spin1_Spinup
                        If Skip = 1 Then
                        Skip = 0
                        MsgBox ("You Cannot exit until all fields are valid!")
                        Exit Sub
                        End If
                Unload DEF_JOB

                Else
                Spin1_SpinDown
                    If Skip = 1 Then
                    Skip = 0
                    MsgBox ("You Cannot exit until all fields are valid!")
                    Exit Sub
                    End If
                Unload DEF_JOB
                End If
    End If
End If

Exit Sub
CheckErrorss:
```

```
Reply = ChekError(Err)

Select Case Reply

Case Is = 1
Exit Sub

Case Is = 2
Resume Next

Case Else
Exit Sub


End Select
End Sub

Sub Finaltask_Change ()
EstSetUpM.Enabled = True
'ChangehasOccurred = 1
End Sub

Sub Finaltask_GotFocus ()
Finaltask.BackColor = &H8080FF
End Sub

Sub Finaltask_LostFocus ()
Finaltask.BackColor = &HFFFF&
End Sub

Sub Finaltask_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As
Single)
Text1.Text = "Enter the number of tasks that you want to define overall, this i
 entered once at the first task definition only."
Text2.Text = "Enter the data here ninth."
End Sub

Sub FixturesFname_Change ()
Def_fix.Enabled = True
'ChangehasOccurred = 1
End Sub

Sub FixturesFname_GotFocus ()
FixturesFname.BackColor = &H8080FF
End Sub

Sub FixturesFname_LostFocus ()
FixturesFname.BackColor = &HFFFF&
End Sub

Sub FixturesFname_MouseMove (Button As Integer, Shift As Integer, X As Single,
 As Single)
Text1.Text = "This tells the user the name of the file that the fixture details
will be saved to."
Text2.Text = "Never enter data here manually!!"
End Sub
```

```
)EF_JOB.FRM - 19

:ub Form_Load ()

)im Msg As String
)im Typ As Integer
Dim Title As String
)im Reply As Integer
)im ProcessSpec() As ProcessDef
Dim Processtyp As ProcessDef
Dim Filelength As Integer
)im i As Integer
Dim Filenum As Integer

 The first thing that is required to be done is to open the
 file containing the names of all the different types of
'machines that are present in the Factory and load these
 into the 'Processes Available' list box

'Allocate a filenumber for the file to be opened
 ilenum = FreeFile

'Calculate array storage length
 ilelength = Len(Processtyp)

'Redimension array to hold firsy record
eDim ProcessSpec(1)

'Open the file containing the names of all the processes
)pen "c:\Factory\Process\files\Proclist.Pfl" For Random Access Read Write Shared
 As Filenum Len = Filelength

'Read the first record to memory
 et Filenum, 1, ProcessSpec(1)

Tf ProcessSpec(1).Processnumber = 0 Then
   Title = "File Empty"
   Msg = "There is nothing in this file yet it will have to be created in the "
   Msg = Msg & "'Define Processes' window"
   Typ = MB_ICONEXCLAMATION + MB_OK
   Reply = MsgBox(Msg, Typ, Title)
   Close Filenum
xit Sub
nd If

 else there is something there so add to list
eDim Preserve ProcessSpec(ProcessSpec(1).Processnumber)
For i = 1 To ProcessSpec(1).Processnumber
 Get Filenum, i, ProcessSpec(i)
 Machinelist.AddItem ProcessSpec(i).Processname
Next

 lose Filenum

f Val(Taskname.Text) = 0 Then
```

```
DEF_JOB.FRM - 20

End If



End Sub

Sub Form_Unload (Cancel As Integer)

'We must update the queuestatus form

'This application allows the user to view the current factory
'queue status and to modify it if required

Dim Thistask As JobRecord
Dim Filenum As Integer
Dim Filenam As String * 50
Dim thisfileslength As Integer
Dim i As Integer
Dim ThisFilesname As String * 60
Dim Reply As Integer
Dim Jobspending() As Pendingjob
Dim Thisjob As Pendingjob

'open the file that details all of the jobs that are waiting to be done.

ThisFilesname = "C:\Factory\Queue\jobspend.jpf"
Filenum = FreeFile

'get the length of the file
thisfileslength = getfilesLength5(ThisFilesname, Thisjob, Jobspending(), Filenu
)

'Now write the contents of the file to memory.
Call WriteToMemory5(Filenum, thisfileslength, Jobspending())

For i = 1 To thisfileslength
    Queuestat.Jobspend.AddItem (Jobspending(i).Jobname), i - 1
Next
Close Filenum


Factrun.Timer1.Enabled = True

End Sub

Sub JobInf_Change ()
'ChangehasOccurred = 1
End Sub

Sub JobInf_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Sin
gle)
Text1.Text = "This textbox contains the name that the job will be saved to and i
s entered automatically when the name of the job is defined."
```

```
Text2.Text = "Never enter data here manually!!"
End Sub

Sub JobName_Change ()

' a new job is defined so set tasknumber as 1
Tasknumber.Text = "1"

JobInf.Text = Jobname.Text & ".iff"
CNCFname.Text = Jobname.Text & Trim(Tasknumber.Text)
SupInstructionsFname.Text = Jobname.Text & Trim(Tasknumber.Text)
Machinelist.Enabled = True
FixturesFname.Text = Jobname.Text & Trim(Tasknumber.Text)
'ChangehasOccurred = 1
End Sub

Sub JobName_GotFocus ()
Jobname.BackColor = &H8080FF
End Sub

Sub JobName_LostFocus ()
Jobname.BackColor = &HFFFF&
End Sub

Sub JobName_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Si
ngle)
Text1.Text = "State the name of the job in this textbox, when the jobfile is sav
ed it is saved under this name"
Text2.Text = "Enter data here second    "
End Sub

Sub Label26_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Si
ngle)
Text1.Text = "Enter the expected completion date for the overall job here."
'Text2.Text = "Enter the data here fifth."

End Sub

Sub Label27_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Si
ngle)
Text1.Text = "Enter the expected finish time of the job here."
 Text2.Text = "Enter the data here fifth."

End Sub

Sub LoadToolTypes_Click ()

 This particular procedure only becomes available after the user has chosen the
 articular
'machine type that he wants to use. This procedure displays all the available

 tools that have been configured for the type of machine selected.
On Error Resume Next

 im FilesLength As Integer
Dim i As Integer
```

```
Dim OptionSelected As String
Dim Msg As String
Dim Reply As Integer
Dim Title As String
Dim Typ As Integer
Dim Request As String
Dim Answer As String
Dim ThisState As Optionstate
Dim Recordlength As Integer
Dim Filenam As String * 40
Dim ArrayTyp As String
Dim EndOfArray As Integer
Dim Filenum As Integer
Dim OldOption As String
Dim Thatstate As OldOptions
Dim OptionList() As OldOptions
Dim NextOldoption As String
Dim ThisError As Integer
Dim j As Integer

'Clear any residual specs from other machines
ToolType.Clear

'Load the current option button option state for the currently loaded
'process type.
Filenum = FreeFile

Filenam = "c:\Factory\System\" & Trim(Taskname.Text) & ".Opt"
EndOfArray = 4

Call ReadToMemory(Filenum, Filenam, ThisState, EndOfArray)

For i = 1 To EndOfArray
    ToolType.AddItem Optionstatearray(i).OptionCaption
Next



Close Filenum


'Now we must open the file which has the names of all the other tool types
'configured for this particular machine.

Filenum = FreeFile
Recordlength = Len(Thatstate)
Filenam = "c:\Factory\System\" & Trim(Taskname.Text) & ".Old"

'Find out how many records there are in the file
FilesLength = GetFilesLength2(Filenam, Thatstate, OptionList(), Filenum)

'Write the file contents to memory
Call WriteToMemory3(Filenum, FilesLength, OptionList())

'now write the contents of the file into the listbox
For i = 1 To FilesLength
```

```
)EF_JOB.FRM - 23

'oolType.AddItem OptionList(i).OptionName
iext

Close Filenum

'Tidy up the listbox
For i = 0 To ((ToolType.ListCount) - 1)
    If Len(Trim(ToolType.List(i))) = 0 Then
        ToolType.RemoveItem i
    End If
iext

LoadToolTypes.Enabled = False
'oolType.Enabled = True

'Remove common names from listbox
'or i = 0 To ((ToolType.ListCount) - 1)
    For j = (i + 1) To ((ToolType.ListCount) - 1)
        If InStr(Trim(ToolType.List(i)), Trim(ToolType.List(j))) = 1 And Len(Tri
(ToolType.List(j))) > 0 Then 'there is a match
            ToolType.RemoveItem i
        End If
    Next
ext

'xit Sub
heckThisError:

'f Err = 64 Or 9 Then
MsgBox (Error)
Exit Sub
'nd If

'hisError = ChekError(Err)

elect Case ThisError

Case Is = 2

Msg = "There are no tools of this type configured for this machine type as"
Msg = Msg & " yet. To configure tools for this machine type please goto "
sg = Msg & "the main menu and select the 'Define Tools' button."

MsgBox Msg, MB_ICONSTOP + MB_OK, "Tools not available"
xit Sub

Case Else
esume Next
id Select


id Sub

ub LoadToolTypes_LostFocus ()
oolType.BackColor = &H8080FF
End Sub
```

```
Sub Machinelist_Click ()
Taskname.Text = Machinelist.List(Machinelist.ListIndex)
'LoadToolTypes.Enabled = True
'ToolSize.Enabled = False
End Sub

Sub Machinelist_GotFocus ()
Machinelist.BackColor = &H8080FF
End Sub

Sub Machinelist_LostFocus ()
Machinelist.BackColor = &HFFFF&
End Sub

Sub MachRequired_Change ()
'ChangehasOccurred = 1
End Sub

Sub NoOfCycles_Change ()
CNCFname.Enabled = True
'ChangehasOccurred = 1
End Sub

Sub NoOfCycles_GotFocus ()
NoOfCycles.BackColor = &H8080FF
End Sub

Sub NoOfCycles_LostFocus ()
NoOfCycles.BackColor = &HFFFF&
End Sub

Sub NoOfCycles_MouseMove (Button As Integer, Shift As Integer, X As Single, Y A
 Single)
Text1.Text = "Enter the number of parts to be made here."
Text2.Text = "Enter the data here fifth."
End Sub

Sub Picture1_Click ()
Queuejob_Click
End Sub

Sub Queuejob_Click ()
'This button allows the user to queue the current job providing all tasks have b
een defined
Dim Thistask As JobRecord
Dim Filenum As Integer
Dim Filenam As String * 50
Dim thisfileslength As Integer
Dim i As Integer
Dim ThisFilesname As String * 60
Dim Reply As Integer
Dim Jobspending() As Pendingjob
Dim Thisjob As Pendingjob
```

```
'First check to see if all tasks have been defined
If Val(Finaltask.Text) <> Val(Tasknumber.Text) Then
    MsgBox "You have to be at the final task before you can queue this task", M
_ICONSTOP, "Queuing error"
    Exit Sub
End If

'open the file that details all of the jobs that are waiting to be done.

ThisFilesname = "C:\Factory\Queue\jobspend.jpf"
Filenum = FreeFile

' get the length of the file
thisfileslength = getfilesLength5(ThisFilesname, Thisjob, Jobspending(), Filenu
)

' Now write the contents of the file to memory.
Call WriteToMemory5(Filenum, thisfileslength, Jobspending())

' Check to see if this job has already been pended
For i = 1 To thisfileslength
    If Trim(Jobname.Text) = Trim(Jobspending(i).Jobname) Then
            Close Filenum
            Exit Sub
    End If
Next


'now increment the file placeholder by 1 to reflect
the fact that another record is going to be added.
Jobspending(1).Filelength = Jobspending(1).Filelength + 1

' write the new record to memory.
Jobspending(Jobspending(1).Filelength).Jobname = Jobname.Text


thisfileslength = Jobspending(1).Filelength

' now save the new data to file
Call WritetoFile3(Filenum, thisfileslength, Jobspending())


Close Filenum

End Sub

Sub RqCompDate_ValidationError (InvalidText As String, StartPosition As Integer)

'When we clear the MaskedEdit box prior to defining a new job a validation error
 occurs which
'causes this event to be fired therefore we must trap this error and respond to
it.
```

```
If Len(Trim(InvalidText)) = 0 Then
'Dont Panic!! we are just clearing the textbox prior to defining a new job
Exit Sub
End If

MsgBox "Date Entered is not in a valid format, it should be entered in a format
such as '09-1-66'."

End Sub

Sub RqCompTime_ValidationError (InvalidText As String, StartPosition As Integer-
'When we clear the MaskedEdit box prior to defining a new job a validation erro
 occurs which
'causes this event to be fired therefore we must trap this error and respond to
it.

If Len(Trim(InvalidText)) = 0 Then
'Dont Panic!! we are just clearing the textbox prior to defining a new job
Exit Sub
End If

MsgBox "Time Entered is not in a valid format, it should be entered in a format
such as '10:12'."

End Sub

Sub ShowTools_Click ()
Dim TrimMachineName As String * 3
Dim TrimToolname As String * 4
Dim TrimMach As String * 3
Dim TrimOption As String * 4
Dim Filename As String
Dim ThisTool As Toolrecord
Dim TheseTools() As Toolrecord
Dim Filesize As Variant
Dim ListBoxLocation As String
Dim ToolSizeBeingAdded As Variant
Dim Listname As String * 20
Dim ListLength As Integer
Dim IsItemInList As Variant
Dim thisfileslength As Integer
Dim Filenum As Integer
Dim ThatTool As Toolrecord
Dim ThoseTools() As Toolrecord
Dim MainToolist() As ToolIdRecord
Dim ThisToolsId As ToolIdRecord
Dim NoOfToolTypes As Integer
Dim i As Integer
Dim TheRequiredId As Integer
Dim Fileno As Integer
Dim Msg As String



'check to see if a tool has been selected from the 'Choose
```

```
DEF_JOB.FRM - 27

'Tool Type listbox'

If Len(ToolType.List(ToolType.ListIndex)) = 0 Then
   Msg = "You haven't selected a tool type from the 'Choose"
   Msg = Msg & " Tool Type' list, please do so."
   MsgBox Msg, MB_ICONSTOP + MB_OK, "Tool Selection Error"
   Exit Sub
End If

'This particular procedure opens a file containing all the sizes available of
''the tool selected from the 'Choose Tool Type' listbox.
TrimMachineName = Taskname.Text
TrimToolname = Trim(ToolType.List(ToolType.ListIndex))


Filename = "C:\Factory\Process\ToolFile\" & TrimMachineName & "!" & TrimToolnam
 & ".tl"


'Declare local variables
Filenum = FreeFile

Filesize = getFileslength(Filename, ThisTool, TheseTools(), Filenum)

'Clear the current list contents
ToolSize.Clear

'read toolinfo from file and write to the listbox
Call ReadAndAddToList1(TheseTools(), Filesize, Filenum)

Close Filenum

howTools.Enabled = False
oolSize.Enabled = True
End Sub

ub ShowTools_LostFocus ()
ToolSize.BackColor = &H8080FF
nd Sub

Sub ShowTools_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As
ingle)
ext1.Text = "After choosing the required tool type from the 'Choose Tool Type'
listbox choose this button to show the available tool sizes for that tooltype. F
om the list selecting toolsizes will fill the 'Tool Required' textboxes below."
ext2.Text = "Execute this button thirteenth."
End Sub

ub Spin1_SpinDown ()
'Declare variables
im Savetask As JobRecord
im Filenum As Integer
Dim Recordlength As Integer
im Taskposition As Integer
im Savetasks() As JobRecord
Dim Reply As Integer
```

```
Dim i As Integer
Dim Msg As String
Dim Typ As Integer
Dim Title As String
Dim BeginOfTaskDef As Integer

On Error GoTo CheckError

Taskposition = Val(Tasknumber.Text)
Tasknumber.Text = Val(Tasknumber.Text) - 1


If Val(Tasknumber.Text) < 1 Then
      Tasknumber.Text = CInt(Tasknumber.Text) + 1
      Msg = "You cannot go down to the next task as you are"
      Msg = Msg & " already at the first task"
      Title = "Spinbutton Error"
      Typ = MB_ICONEXCLAMATION
      Reply = MsgBox(Msg, Typ, Title)
      BeginOfTaskDef = 1
Else

BeginOfTaskDef = 2
End If

'Check to see if all textboxes containing string variables have something there
'as the error checking routine only checks to see if all textboxes containing
'other types of variables have something there(By declaring a variable type
'mismatch.This serves to check that all task parameters have been entered by
'the user before he moves onto the next task - if they have not then this
'procedure will prompt him to

'If Len(Trim(CNCFname.Text)) = 0 Or Len(Trim(TaskName.Text)) = 0 Or Len(Trim(Jc
name.Text)) = 0 Or Len(Trim(FixturesFname.Text)) = 0 Or Len(Trim(SupInstructions
Fname.Text)) = 0 Then
 '   Skip = 1
  ''   Title = "Task Data Incomplete"
   ' Msg = "The task data that you have described is either incomplete or the d
ta"
   ' Msg = Msg & " type entered is incorrect. Please refer to the help file rega
rding"
   ' Msg = Msg & " the correct data type entries that should be used for each f
eld."
   ' Reply = MsgBox(Msg, Typ, Title)
   ' tasknumber.Text = CInt(tasknumber.Text) + 1
'Exit Sub
'End If

'Save the current task information to the correct jobfile at the
'correct position and then spindown to the next task
Filenum = FreeFile
Recordlength = Len(Savetask)
'Open the jobfile who's name is in the JobInf Textbox and allow write
'only access to it

Open JobFilepath & JobInf.Text For Random Access Read Write As Filenum Len = Rec
```

```
)EF_JOB.FRM - 29

ordlength
'Redimension the array to hold only enough space as is required
ReDim Savetasks(Taskposition)

'Get array info from file
For i = 1 To Taskposition
Get Filenum, i, Savetasks(i)
Next

'Allocate the task fields to the array
.avetasks(Taskposition).Jobname = Jobname.Text
 avetasks(Taskposition).Infofilename = JobInf.Text
Savetasks(Taskposition).Tasknumber = Taskposition
 avetasks(Taskposition).Machinetype = MachRequired.Text
 avetasks(Taskposition).Taskname = Taskname.Text
Savetasks(Taskposition).EstCycletime = EstCycTime.Text
 avetasks(Taskposition).Setuptime = EstSetUpM.Text
 avetasks(Taskposition).EstLoadUnloadtime = EstLoadUn.Text
Savetasks(Taskposition).NoOfCycles = NoOfCycles.Text
 avetasks(Taskposition).CNCFilename = CNCFname.Text
 avetasks(Taskposition).Tool1required = Tool1rq.Text
Savetasks(Taskposition).Starttime = Start_Time.Text
'Savetasks(TaskPosition).Tool2required = Tool2rq.Text
 Savetasks(TaskPosition).Tool3required = Tool3rq.Text
'Savetasks(TaskPosition).Tool4required = Tool4rq.Text
Savetasks(Taskposition).Fixturesfilename = FixturesFname.Text
 avetasks(Taskposition).Duedate = Duedate.Text
avetasks(Taskposition).RequiredCompDate = RqCompdate.Text
Savetasks(Taskposition).Requiredcomptime = RqCompTime.Text
 avetasks(Taskposition).Finaltask = Finaltask.Text
avetasks(Taskposition).Force = Force
Savetasks(Taskposition).SetUpInstructionsfilename = SupInstructionsFname.Text

    Jobname As String * 13
'   InFoFilename As String * 20
'   FinalTask As Integer
    Tasknumber As Integer
''   Machinetype As Single 'Type as 3.1 the ".1" represents capacity
'   Taskname As String * 22
'   EstCycletime As Integer
  ' Setuptime As Integer
'   Estloadunloadtime As Integer
'   NoofCycles As Integer
'   CNCfilename As String * 13
'   Geometricfilename As String * 13
    Tool1required As Single
'  ' Fixturesfilename As String * 13
   'Status As Integer
'   StartTime As String * 5
  ' Duedate As String * 15
'   RequiredCompDate As String * 15
  ' RequiredComptime As String * 5
'   Force As Integer
  ' SetUpInstructionsFilename As String * 20 'Shows datum graphically
```

```
For i = 1 To Taskposition
Put Filenum, i, Savetasks(i)
Next


'Now we must Check to see if the task we are spinning down to is relevant - if
'it is then we must get the data out of the file that is opened and present
'it on the screen so that it can be viewed and modified if required, otherwise
'if it is not relevant we must exit this procedure.


If BeginOfTaskDef <> 1 Then
      'If the task spun down to is valid
      ReDim Preserve Savetasks(Taskposition - 1)

      For i = 1 To (Taskposition - 1)
      Get Filenum, i, Savetasks(i)
      Next

      'Write information to the Screen
      MachRequired.Text = Savetasks(Taskposition - 1).Machinetype
      Taskname.Text = Savetasks(Taskposition - 1).Taskname
      EstCycTime.Text = Savetasks(Taskposition - 1).EstCycletime
      EstSetUpM.Text = Savetasks(Taskposition - 1).Setuptime
      EstLoadUn.Text = Savetasks(Taskposition - 1).EstLoadUnloadtime
      NoOfCycles.Text = Savetasks(Taskposition - 1).NoOfCycles
      CNCFname.Text = Savetasks(Taskposition - 1).CNCFilename
      Tool1rq.Text = Savetasks(Taskposition - 1).Tool1required
'      Tool2rq.Text = Savetasks(TaskPosition - 1).Tool2required
 '     Tool3rq.Text = Savetasks(TaskPosition - 1).Tool3required
  '    Tool4rq.Text = Savetasks(TaskPosition - 1).Tool4required
      'Start_Time.text = Savetasks(Taskposition - 1).Starttime
      SupInstructionsFname.Text = Savetasks(Taskposition - 1).SetUpInstructions
ilename
      FixturesFname.Text = Savetasks(Taskposition - 1).Fixturesfilename
      Text3.Text = Savetasks(1).Force
     ' Text5.Text = Savetasks(Taskposition - 1).Duedate
      Close Filenum


Else
Close Filenum
BeginOfTaskDef = 2
Exit Sub
End If



Exit Sub
CheckError:
Reply = ChekError(Err)

Select Case Reply
```

```
)EF_JOB.FRM - 31

Case Is = 1
Tasknumber.Text = Val(Tasknumber.Text) + 1
Close Filenum
Exit Sub

Case Is = 2
Resume Next

Case Else
Close Filenum
Exit Sub


End Select

End Sub

Sub Spin1_Spinup ()
Dim Msg As String
Dim Typ As Integer
Dim Title As String
Dim Lasttask As Integer
Dim Savetask As JobRecord
Dim Filenum As Integer
Dim Recordlength As Integer
Dim Taskposition As Integer
Dim Savetasks() As JobRecord
Dim Reply As Integer
Dim i As Integer
Dim TaskDefEnd As Integer
Dim Urgent As Integer
Dim Origtask As Integer

On Error GoTo CheckErrors

Taskposition = Val(Tasknumber.Text)
Origtask = Val(Tasknumber.Text)
Tasknumber.Text = Val(Tasknumber.Text) + 1


'Check to see if current task has exceeded final task
If Val(Tasknumber.Text) > Val(Finaltask.Text) Then
          If Len(Trim(CNCFname.Text)) = 0 Or Len(Trim(Jobname.Text)) = 0 Or Len
Trim(Taskname.Text)) = 0 Or Len(Trim(FixturesFname.Text)) = 0 Or Len(Trim(SupIn
tructionsFname.Text)) = 0 Then
          Title = "Task Data Incomplete"
          Msg = "The task data that you have described is either incomplete or
he data"
          Msg = Msg & " type entered is incorrect. Please refer to the help fil
 regarding"
          Msg = Msg & " the correct data type entries that should be used for e
ch field."
          Reply = MsgBox(Msg, Typ, Title)
          Tasknumber.Text = Val(Tasknumber.Text) - 1
          Exit Sub
                Else
```

```
                Title = "Invalid Tasknumber"
                Typ = 0
                Msg = "You have already defined all tasks as stipulated. If yo
 want to add"
                Msg = Msg & " more tasks to this job adjust 'Final Task' box"
                Reply = MsgBox(Msg, Typ, Title)
                Tasknumber.Text = Val(Tasknumber.Text) - 1
                TaskDefEnd = 1
        End If

End If

'Check to see if all textboxes containing string variables have something there
'as the error checking routine only checks to see if all textboxes containing
'other types of variables have something there(By declaring a variable type
'mismatch.This serves to check that all task parameters have been entered by
'the user before he moves onto the next task - if they have not then this
'procedure will prompt him to
If Len(Trim(CNCFname.Text)) = 0 Or Len(Trim(Jobname.Text)) = 0 Or Len(Trim(Task
ame.Text)) = 0 Or Len(Trim(FixturesFname.Text)) = 0 Or Len(Trim(SupInstructions
name.Text)) = 0 Then
    Skip = 1
    Title = "Task Data Incomplete"
    Msg = "The task data that you have described is either incomplete or the da
a"
    Msg = Msg & " type entered is incorrect. Please refer to the help file rega
ding"
    Msg = Msg & " the correct data type entries that should be used for each fi
ld."
    Reply = MsgBox(Msg, Typ, Title)
    Tasknumber.Text = Val(Tasknumber.Text) - 1
Exit Sub
End If

'Find out whether or not this job is urgent
If Origtask = 1 Then
    Title = "Is task Urgent"
    Msg = "Is this task urgent?"
    Typ = MB_ICONQUESTION + MB_YESNO
    Reply = MsgBox(Msg, Typ, Title)
    If Reply = Idyes Then
        Urgent = 1
        Else
        Urgent = 2
    End If
End If


'Save the current task information to the correct jobfile at the
'correct position and then spinup to the next task
Filenum = FreeFile
Recordlength = Len(Savetask)
'Open the jobfile who's name is in the JobInf Textbox and allow write
'only access to it
Open JobFilepath & JobInf.Text For Random Access Read Write As Filenum Len = Re
ordlength
```

```
'Redimension the array to hold only enough space as is required
:eDim Savetasks(Taskposition)

'Get array information, write to array memory variables
'or i = 1 To Taskposition
Get Filenum, i, Savetasks(i)
Next

 Allocate the task fields to the array
Savetasks(Taskposition).Jobname = Jobname.Text
Savetasks(Taskposition).Infofilename = JobInf.Text
Savetasks(Taskposition).Tasknumber = Taskposition
Savetasks(Taskposition).Machinetype = MachRequired.Text
Savetasks(Taskposition).Taskname = Taskname.Text
Savetasks(Taskposition).EstCycletime = EstCycTime.Text
Savetasks(Taskposition).Setuptime = EstSetUpM.Text
Savetasks(Taskposition).EstLoadUnloadtime = EstLoadUn.Text
Savetasks(Taskposition).NoOfCycles = NoOfCycles.Text
Savetasks(Taskposition).CNCFilename = CNCFname.Text
Savetasks(Taskposition).Tool1required = Tool1rq.Text
Savetasks(Taskposition).Starttime = Start_Time.Text
'Savetasks(TaskPosition).Tool2required = Tool2rq.Text
'Savetasks(TaskPosition).Tool3required = Tool3rq.Text
 Savetasks(TaskPosition).Tool4required = Tool4rq.Text
Savetasks(Taskposition).Fixturesfilename = FixturesFname.Text
Savetasks(Taskposition).Duedate = Duedate.Text'Text5.Text
Savetasks(Taskposition).RequiredCompDate = RqCompdate.Text
Savetasks(Taskposition).Requiredcomptime = RqCompTime.Text
Savetasks(Taskposition).Finaltask = Finaltask.Text
Savetasks(Taskposition).Force = Urgent
Savetasks(Taskposition).SetUpInstructionsfilename = SupInstructionsFname.Text
Savetasks(1).Status = 1

If Savetasks(1).Status = 0 Then 'this is the first task
    Savetasks(1).Status = 1
End If



For i = 1 To Taskposition
Put Filenum, i, Savetasks(i)
Next

 If the current task is not the finaltask then read the next task's task
 information onto the screen from file otherwise quit the procedure.

If TaskDefEnd <> 1 Then
    ReDim Savetasks(Taskposition + 1)

    For i = 1 To (Taskposition + 1)
    Get Filenum, i, Savetasks(i)
    Next

    'Write information to the Screen
    MachRequired.Text = Savetasks(Taskposition + 1).Machinetype
```

```
      Taskname.Text = Savetasks(Taskposition + 1).Taskname
      EstCycTime.Text = Savetasks(Taskposition + 1).EstCycletime
      EstSetUpM.Text = Savetasks(Taskposition + 1).Setuptime
      EstLoadUn.Text = Savetasks(Taskposition + 1).EstLoadUnloadtime
      NoOfCycles.Text = Val(NoOfCycles.Text)
      Tool1rq.Text = Savetasks(Taskposition + 1).Tool1required
      'Tool2rq.Text = Savetasks(TaskPosition + 1).Tool2required
      'Tool3rq.Text = Savetasks(TaskPosition + 1).Tool3required
      'Tool4rq.Text = Savetasks(TaskPosition + 1).Tool4required
      Text3.Text = Savetasks(1).Force
      SupInstructionsFname.Text = Trim(Jobname.Text) & Val(Tasknumber.Text)
      CNCFname.Text = Trim(Jobname.Text) & Val(Tasknumber.Text)
      FixturesFname.Text = Trim(Jobname.Text) & Val(Tasknumber.Text)
      Close Filenum
Else
Close Filenum
TaskDefEnd = 2
Exit Sub
End If



Exit Sub
CheckErrors:
Reply = ChekError(Err)

Select Case Reply

Case Is = 1
Tasknumber.Text = Val(Tasknumber.Text) - 1
Close Filenum
Exit Sub

Case Else
Close Filenum
Exit Sub


End Select




End Sub

Sub Start_Time_ValidationError (InvalidText As String, StartPosition As Integer)
'When we clear the MaskedEdit box prior to defining a new job a validation erro
 occurs which
'causes this event to be fired therefore we must trap this error and respond to
it.

If Len(Trim(InvalidText)) = 0 Then
'Dont Panic!! we are just clearing the textbox prior to defining a new job
Exit Sub
```

```
EF_JOB.FRM - 35

nd If

MsgBox "Time Entered is not in a valid format, it should be entered in a format
such as '10:12'."

nd Sub

ub SupInstructionsFname_Change ()
upInstructionsFname.Enabled = True
'ChangehasOccurred = 1

f Check = 1 Then
    Exit Sub
    Else
        SetUpInsfname = SupInstructionsFname.Text
        Check = 0
nd If


nd Sub

Sub SupInstructionsFname_GotFocus ()
upInstructionsFname.BackColor = &H8080FF
nd Sub

ub SupInstructionsFname_LostFocus ()
upInstructionsFname.BackColor = &HFFFF&
extboxchoice = 1
ool1rq.Enabled = True
nd Sub

Sub SupInstructionsFname_MouseMove (Button As Integer, Shift As Integer, X As Si
gle, Y As Single)
ext1.Text = "This textbox is automatically filled by the system and gives the r
ame of the setup file to the jobfile. Never under any circumstances alter this b
x as the system may become corrupted!!!!"
ext2.Text = "Never enter data here manually!!"
End Sub

ub TaskName_Change ()
'ChangehasOccurred = 1

f Len(Taskname.Text) > 0 Then
LoadToolTypes.Enabled = True
nd If

asknam = Trim(Taskname.Text)
stLoadUn.Enabled = True

End Sub

b TaskName_GotFocus ()
Taskname.BackColor = &H8080FF
nd Sub

Sub TaskName_LostFocus ()
```

```
Taskname.Enabled = False
Taskname.BackColor = &HFFFF&

End Sub

Sub TaskName_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As S
ingle)
Text1.Text = "This box contains the name of the task being performed as is fill
d by selecting the required machine type from the 'Processes Available' list."
Text2.Text = "To fill select process from 'Processes Available. This is the thi
d data entry point.'"
End Sub

Sub Tasknumber_Change ()
'ChangehasOccurred = 1
Jobname.Enabled = True
End Sub

Sub Tasknumber_GotFocus ()
Tasknumber.BackColor = &H8080FF
End Sub

Sub Tasknumber_LostFocus ()
Tasknumber.BackColor = &HFFFF&
End Sub

Sub Tasknumber_MouseMove (Button As Integer, Shift As Integer, X As Single, Y A
 Single)
Text1.Text = "This textbox contains the number of the task currently being defi
ed."
Text2.Text = "Enter Data Here First     "
End Sub

Sub Text5_Click ()
Dim Msg As String
Dim Title As String
Dim Typ As Integer
Dim Reply As Integer

Msg = "Do you want to edit the due date?"
Title = "Due date change"
Typ = MB_ICONSTOP + MB_YESNO

Reply = MsgBox(Msg, Typ, Title)

If Reply = Idyes Then
    Text5.Enabled = False
    Text5.Visible = False
    Duedate.SetFocus
    Else
    Exit Sub
End If



End Sub
```

```
;ub Tool1rq_Change ()
On Error Resume Next

'ool2rq.Enabled = True
Textboxchoice = 2
'ToolType.SetFocus
;nd Sub

Sub Tool1rq_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As S
.gle)
 ChangehasOccurred = 1
Textboxchoice = 1
;nd Sub

Sub Tool1rq_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As S
gle)
'ext1.Text = "Clicking this textbox and then selecting a toolsize from the 'Cho
se Tool Size'list will assign that tool to this box."
'ext2.Text = "Never type in this box manually,System failure could result!!!"
 nd Sub

'ub ToolSize_Click ()

'Declare variables
)im Filename As String
 im Filenum As Integer
Dim NoOfToolTypes As Integer
)im MainToolist() As ToolIdRecord
 im ThisToolsId As ToolIdRecord
)im i As Integer
)im position As Integer
 im Currenttoolid As Integer
)im CurrentToolSizeId As Single
Dim ToolChoice As Single

)n Error Resume Next

 When this listbox is clicked we want the item that is selected to be placed
 in the textbox that was selected by the mousedown event.Before this event
'occurs we must also be sure that the correct tool as requested is displayed
 and it's available sizes - we do this by deactivating this list box until
 this has occurred.

 The first thing that must be done is to find out the identification code of
 the tool being considered.In order to do this we must open up the file that
'contains the names of all the tooltypes so far configured.
'First get the total number of tool types that have already been specified.

Filename = "c:\factory\system\AlToolTyps.Tl"
'ilenum = FreeFile
 )OfToolTypes = GetFilesLength1(Filename, ThisToolsId, MainToolist(), Filenum)

'now write the list to memory
 all WriteToMemory(Filenum, NoOfToolTypes, MainToolist())
```

```
'Now we must check in order to gain a match between a particular tooltype in
'the file and that selected from the 'Processes Available' listbox.
For i = 1 To NoOfToolTypes
   If Trim(MainToolist(i).ToolType) = Trim(ToolType.List(ToolType.ListIndex)) '
en
   position = i
   End If
Next

Currenttoolid = MainToolist(position).Toolid
CurrentToolSizeId = Val(ToolSize.List(ToolSize.ListIndex)) / 100

ToolChoice = Currenttoolid + CurrentToolSizeId
Close Filenum

'Now apply "ToolChoice" to the textbox that was selected

'First find out what textbox was selected and then apply
'Select Case Textboxchoice

'Case Is = 1
Tool1rq.Text = ToolChoice

'Case Is = 2
'Tool2rq.Text = ToolChoice

'Case Is = 3
'Tool3rq.Text = ToolChoice

'Case Is = 4
'Tool4rq.Text = ToolChoice

'End Select




End Sub

Sub ToolSize_GotFocus ()
ToolSize.BackColor = &H8080FF
End Sub

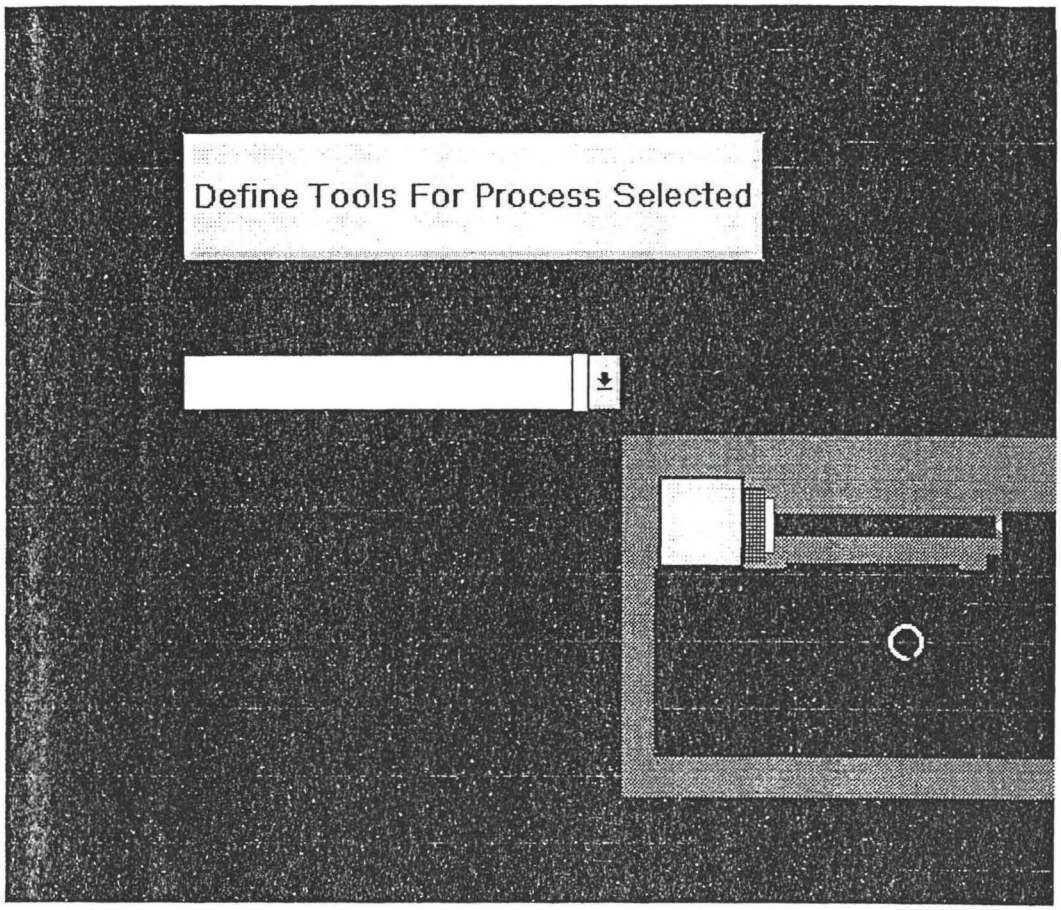
Sub ToolType_Click ()
ShowTools.Enabled = True
End Sub

Sub ToolType_GotFocus ()
ToolType.BackColor = &H8080FF
End Sub

Sub ToolType_LostFocus ()
SupInstructionsFname.BackColor = &HFFFF&
```

```
DEF_JOB.FRM - 39

End Sub
```


Define Tools For Process Selected

```
)EF_TLS.FRM - 1


Sub Def_Tools_Click ()

'This procedure allows the user to select a tool type from the list and then
'click this button which loads a form and allows the user to choose a cutter
'style and then define the cutter sizes.
)im Reply As String * 30
Dim Filenum As Integer

-)n Error Resume Next


'hisIndex = Trim(ToolList.List(ToolList.ListIndex))

'Check to see if anything has been selected
f Len(ThisIndex) = 0 Then
    MsgBox ("You are required to select a process from the process list before
-u can define the tools for it")
xit Sub
End If


StateTools.Show


'StateTools.Caption = ThisIndex & " Tools"

. ilenum = FreeFile


pen "C:\Factory\Process\Specs\" & Trim(ToolList.List(ToolList.ListIndex)) & ".b
..p" For Binary As #Filenum
StateTools.OLE1.FileNumber = Filenum
tateTools.OLE1.Action = 12
.lose Filenum


StateTools.Picture2.Picture = LoadPicture("c:\factory\process\specs\" & Trim(To
olList.List(ToolList.ListIndex)) & ".bmp")

Unload this form to save memory
Unload Define_Tools

: )d Sub

Sub Form_Load ()

'This procedure allows the user to define all the tools that he will possibly
'use in the entire Factory(Factory Tool List).The "Load" event opens the file
:ontaining the names of all the processes in the factory and uses this as
'a suffix to "Tools" allowing the user to define tools only for those
'processes that are currently in the Factory.The processes names & suffix
'Tools" are added to the listbox from which the user can choose and then
'define the tools available in the factory for that process type.
```

```
Dim Msg As String
Dim Typ As Integer
Dim Title As String
Dim Reply As Integer
Dim ProcessSpec() As Processdef
Dim Processtyp As Processdef
Dim Filelength As Integer
Dim i As Integer
Dim Filenum As Integer


'ProcessNumber As Integer
'Processname As String * 20
'CurrentProcessId As Integer
'ProcessId As Integer
'Maxcap As Integer
'End Type

'Allocate a number to the file to be opened
Filenum = FreeFile

'Free up memory for the file to be opened
Filelength = Len(Processtyp)

'Redimension storage array to take first record
ReDim ProcessSpec(1)

'open the process list file
Open "c:\Factory\Process\files\Proclist.Pfl" For Random Access Read Write Share
 As Filenum Len = Filelength

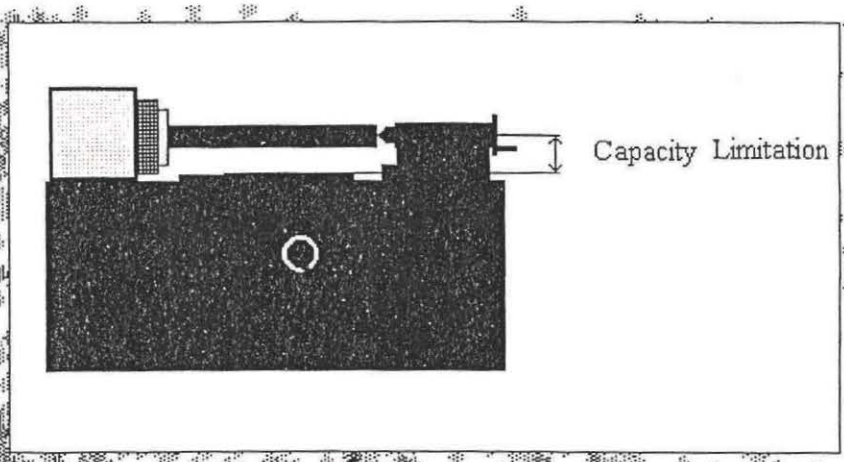
'Get the first record into memory
Get Filenum, 1, ProcessSpec(1)

If ProcessSpec(1).Processnumber = 0 Then
    Title = "File Empty"
    Msg = "There is nothing in this file yet it will have to be created"
    Typ = MB_ICONEXCLAMATION + MB_OK
    Reply = MsgBox(Msg, Typ, Title)
    Close Filenum
Exit Sub
End If

'else there is something there so add to list
ReDim Preserve ProcessSpec(ProcessSpec(1).Processnumber)
For i = 1 To ProcessSpec(1).Processnumber
 'Get each process name from the file and add each processname to the list
 'with the suffix "tools"
 Get Filenum, i, ProcessSpec(i)
 ToolList.AddItem Trim(ProcessSpec(i).Processname)
Next

'Close the file
Close Filenum
```

```
End Sub
```



Capacity Definition Box

Clear Example

A Capacity Bitmap allows the user of the system the ability to choose the capacity of machine required to perform a given job. This window allows the user to draw a sketch of the machine as well as the dimension that the capacity will apply to. For an example of such a bitmap for a lathe click this text.

```
)EFCBIT.FRM - 1

Option Explicit

Sub Command1_Click ()

)im Reply As String * 30
Dim Filenum As Integer

)n Error Resume Next

Filenum = FreeFile

Reply = InputBox("Enter the name of the file that you want this picture saved t
.", "Enter Filename", "")

)pen "C:\Factory\Process\Specs\" & Trim(Filenamed) & ".bmp" For Binary As #File
um
)le1.FileNumber = Filenum
)le1.Action = 11
Close Filenum



nd Sub

Sub Command2_Click ()
im Filenum As Integer
Dim Reply As String

n Error Resume Next
ilenum = FreeFile

eply = InputBox("Enter the Filename you want to open.", "Enter Filename", " ")

Open "C:\Factory\Process\Specs\" & Trim(Reply) & ".bmp" For Binary As #Filenum
le1.FileNumber = Filenum
le1.Action = 12
Close Filenum

nd Sub

ub ClearEx_Click ()
im Filenum As Integer
Dim Reply As String


)le1.Enabled = True

n Error Resume Next
ilenum = FreeFile

)en "C:\Factory\Process\Specs\" & "Template.bmp" For Binary As #Filenum
le1.FileNumber = Filenum
)le1.Action = 12
lose Filenum
```

```
ClearEx.Visible = False


End Sub

Sub Form_Unload (Cancel As Integer)
Dim Reply As String * 30
Dim Filenum As Integer

On Error Resume Next

Filenum = FreeFile

'Reply = InputBox("Enter the name of the file that you want this picture saved
o.", "Enter Filename", "")

Open "C:\Factory\Process\Specs\" & Trim(Filenamed) & ".bmp" For Binary As #File
um
Ole1.FileNumber = Filenum
Ole1.Action = 11
Close Filenum

End Sub

Sub Label3_Click ()
Dim Filenum As Integer
Dim Reply As String

ClearEx.Visible = True

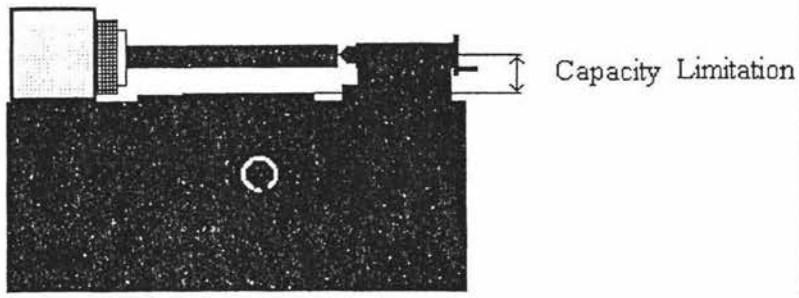
On Error Resume Next
Filenum = FreeFile

'Reply = InputBox("Enter the Filename you want to open.", "Enter Filename", " ")

Open "C:\Factory\Process\Specs\Sample.bmp" For Binary As #Filenum
Ole1.FileNumber = Filenum
Ole1.Action = 12

Close Filenum

Ole1.Enabled = False
End Sub
```



This window is used for the definition of the maximum size of the limiting factor for all processes of the type shown in the frame above.

```
DEFINE_C.FRM - 1

Option Explicit

Dim EntryIsBad As Integer
Dim Repeat As Integer
Dim CapacityStated As Integer
Dim mousemovedonce As Integer

Function ChekkError (errtyp)
Dim types As Integer
Dim Titles As String * 20
Dim Msg As String
Dim response As Integer

types = 0

Select Case errtyp

Case Is = ERR_TYPEMISMATCH

Titles = "Entry Error"
Msg = "You have either failed to enter a measurement of correct units "
Msg = Msg & "or you have not entered any data at all.Please do so. "

Case Is = ERR_PATHNOTFOUND
Titles = "JobName not specified"
Msg = "You not entered the name of the job you are configuring"
Msg = Msg & " this task for. Please do so"

Case Is = ERR_SUBSCRIPTOUTOFRANGE
ChekkError = 2
Exit Function



Case Else
Msg = "An error of type " & Error & " is occurring"



End Select

response = MsgBox(Msg, types, Titles)


Select Case response

Case 1 'Ok was selected
ChekkError = 1
End Select

End Function

Sub Form_Load ()
'This particular form shows the bitmap associated with the process that you
'want to specify the capacity for
Dim Filenum As Integer
```

```
DEFINE_C.FRM - 2

Dim Reply As String

On Error Resume Next
Filenum = FreeFile

'Reply = InputBox("Enter the Filename you want to open.", "Enter Filename", " "

)pen "C:\Factory\Process\Specs\" & Trim(Filenamed) & ".bmp" For Binary As #File
am
Ole1.FileNumber = Filenum
)le1.Action = 12
Close Filenum


End Sub

Sub Form_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Sing
e)
Dim Msg As String
)im Title As String
 )im Answer As Integer
Dim Reply As Integer
)im Defval As String
 )im Typ As Integer
Const XPosition = 5500
Const Yposition = 1200

On Error Resume Next

 ousemovedonce = mousemovedonce + 1
If mousemovedonce > 1 Then

    Exit Sub
End If

 itle = "Enter Machine Capacity"
.sg = "Of all the processes of this type in the Factory enter the Capacity Limit
ation"
 sg = Msg & " value of the machine with the largest limiting factor. "
 efval = "0"
Answer = CInt(InputBox(Msg, Title, Defval, XPosition, Yposition))


'Check that the user has input a valid entry

f Answer = 0 Then
      Msg = "Data Entered invalid.Please Re-enter data."
      Typ = MB_ok + MB_ICONEXCLAMATION
      Title = "Entry Error"
      Reply = MsgBox(Msg, Typ, Title)
      'Set flag which disallows the user the ability to unload the form until
      'something sensible has been entered for machine capacity
      mousemovedonce = 0
      Exit Sub
      Else
```

```
DEFINE_C.FRM - 3

        'something sensible has been added therefore continue

        'Assign maximum defined value to global variable
         CapacityStated = Answer

        'Re_define.Text1.Text = Answer
End If


Exit Sub
errorhandler:

Reply = ChekkError(Err)

End Sub

Sub Form_Unload (Cancel As Integer)
Dim Msg As String
Dim Reply As Integer
Dim Title As String
Dim Typ As Integer
Dim ProcessSpc() As Processdef
Dim Processtp As Processdef
Dim Filelength As Integer
Dim i As Integer
Dim Filenum As Integer
Dim IncrementSize As Integer

''If EntryIsBad = 1 Then
   'Don't unload form as the capacity properties for that machine is so far
   'undefined
  ''Cancel = -1
''End If



''Filenum = FreeFile
''Filelength = Len(Processtp)
''ReDim ProcessSpc(1)

'Open the file
''Open "c:\Factory\Process\files\Proclist.Pfl" For Random Access Read Write Sha
ed As Filenum Len = Filelength

'Get the position of the last process added so as the capacity parameters as
'defined by the user can be added
''Get Filenum, 1, ProcessSpc(1)

'Redimension the array so that all the process data defined so far can be
'recalled and the new capacity data for the last task can be recalled
''ReDim Preserve ProcessSpc(ProcessSpc(1).Processnumber)

''For i = 1 To ProcessSpc(1).Processnumber
''Get Filenum, i, ProcessSpc(i)
''Next
```

```
DEFINE_C.FRM - 4

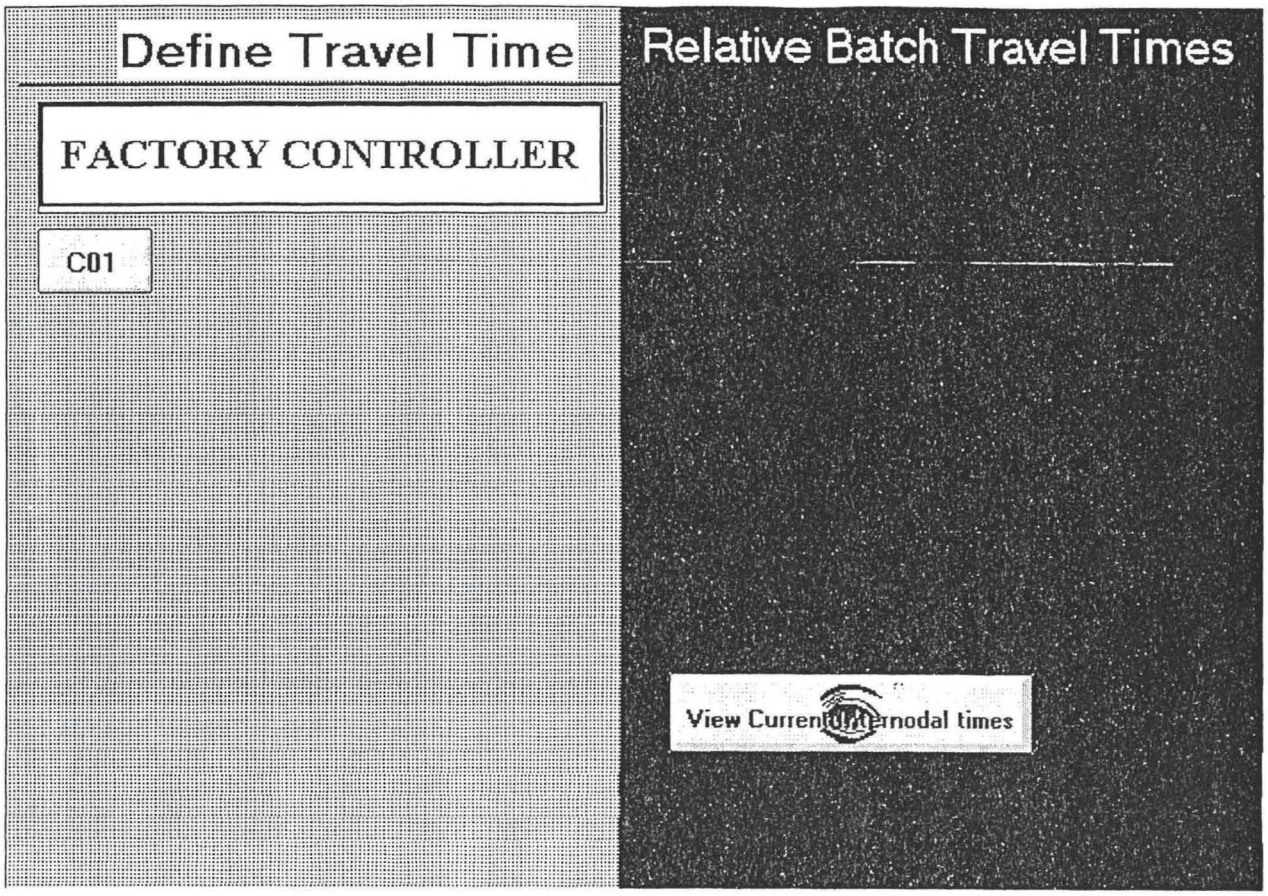
'Assign the newly defined capacity values to the array
''IncrementSize = CInt(CapacityStated / 10)

''ProcessSpc(ProcessSpc(1).Processnumber).Capindex1 = (IncrementSize)
''ProcessSpc(ProcessSpc(1).Processnumber).Capindex2 = (2 * (IncrementSize))
''ProcessSpc(ProcessSpc(1).Processnumber).Capindex3 = (3 * (IncrementSize))
''ProcessSpc(ProcessSpc(1).Processnumber).Capindex4 = (4 * (IncrementSize))
 'ProcessSpc(ProcessSpc(1).Processnumber).Capindex5 = (5 * (IncrementSize))
''ProcessSpc(ProcessSpc(1).Processnumber).Capindex6 = (6 * (IncrementSize))
''ProcessSpc(ProcessSpc(1).Processnumber).Capindex7 = (7 * (IncrementSize))
 'ProcessSpc(ProcessSpc(1).Processnumber).Capindex8 = (8 * (IncrementSize))
  ' ProcessSpc(ProcessSpc(1).Processnumber).Capindex9 = (9 * (IncrementSize))
''ProcessSpc(ProcessSpc(1).Processnumber).Capindex10 = (10 * (IncrementSize))

  'For i = 1 To ProcessSpc(1).Processnumber
''Put Filenum, i, ProcessSpc(i)
  'Next


 'Close the file
  'Close Filenum
''EntryIsBad = 1
''Define_Processes.Delete_Proc.Enabled = True
  'Define_Processes.Add_Process.Enabled = True
''Define_Processes.Def_Cap.Enabled = False

 efine_Processes.MaxCapacity.AddItem CStr(CapacityStated)
mousemovedonce = 0
End Sub
```


Define Travel Time
FACTORY CONTROLLER
C01
Relative Batch Travel Times
View Current Intermodal times

```
)EFINE_L.FRM - 1

Option Explicit
)im Thiscycle As Integer
Dim FactoryIsClicked As Integer
Dim Firstclick As Integer


Sub Command1_Click ()
 imeForTransport.RemoveItem 2

End Sub

ub Cellcont_Click (Index As Integer)
'define localised variables
 im i As Integer
 im Msg As String
Dim Title As String
 im DefVal As String
 im reply As String
 atic First_SecIndex As Integer
tatic FirstIndex As Integer 'define localised implementing but global active va
 to store first
'nodename selected as an integer
tatic SecondIndex As Integer
 tatic FirstElement As String
Static Secondelement As String

 tatic RowIncrement As Integer


 irst_SecIndex = First_SecIndex + 1
econdelement = Cellcont(Index).Caption

 FactoryIsClicked = 1 Then
        FirstElement = "FACT"
        Msg = "Please input a value in minutes for a relative time for a batch o
 components "
        Msg = Msg & "to travel from Node " & Trim(FirstElement) & " to node " &
Trim(Secondelement)
        Title = "Define Internode Travel Time"
        DefVal = "5"
        reply = InputBox$(Msg, Title, DefVal) 'ask for a reply from the user

        'now we must find out what column the secondelement clicked occupies
        TimeForTransport.Row = 1 'select the second row as this is the row that
t e factory
        'controller will occupy.
        TimeForTransport.Col = Index + 2
        TimeForTransport.Text = Trim(reply)
        FactoryIsClicked = 0

        'It is ok to put horizontally but also must put vertically
        TimeForTransport.Col = 1
        TimeForTransport.Row = Index + 2
        TimeForTransport.Text = Trim(reply)
        FactoryIsClicked = 0
```

```
        Else
            'we are configuring the distance between two cell controllers
            If First_SecIndex = 1 Then 'then this button is being pressed to de
ine the first Cell
            'Controller
                FirstIndex = Index
                Else
                    'we are defining the second index that we are required to s
ate the time delay to
                    FirstElement = Cellcont(FirstIndex).Caption
                    Secondelement = Cellcont(Index).Caption
                    Msg = "Please input a value in minutes for a relative time
or a batch of components "
                    Msg = Msg & "to travel from Node " & Trim(FirstElement) & "
to node " & Trim(Secondelement)
                    Title = "Define Internode Travel Time"
                    DefVal = "5"
                    reply = InputBox$(Msg, Title, DefVal) 'ask for a reply from
the user

                    If FirstIndex = Index Then 'then the user has a made a mist
ke as he is trying to
                    'to define the distance between the same nodes so notify hi
 of the fact
                        MsgBox "You cannot define a distance between the same no
de as none exists", MB_OK, "Selection Error"
                        First_SecIndex = 0
                        FirstIndex = -1
                        Exit Sub
                    End If

                    'now make sure that the data is put in a sensible row\colum
 arrangement
                    TimeForTransport.Row = FirstIndex + 2
                    TimeForTransport.Col = Index + 2
                    TimeForTransport.Text = reply

                    'and fill the complimentary row with the same data
                    TimeForTransport.Row = Index + 2
                    TimeForTransport.Col = FirstIndex + 2
                    TimeForTransport.Text = reply

                    'now reset any variables that are required to be reset
                    First_SecIndex = 0
            End If
End If



            'now ask for the distance between the two cells
```

```
)EFINE_L.FRM - 3


nd Sub

ub Factory_Click ()
actoryIsClicked = 1 'this variable is set to 1 if this

picture is clicked
nd Sub

ub Form_Load ()
this form allows the user the ability to load as many Cell controller characthe
rs as required
and then configure the factory According to the distance between each node in t
e factory as
part of the bidding system.
m Indexnumber As Integer
im Oldnumber As Integer
Dim TopPosition As Integer
im NumberOfNodes As Integer
im i As Integer

n Error Resume Next

If Thiscycle = 0 Then 'we are adding only the first row of cell control entitie

opPosition = Cellcont(0).Top 'store first boxe's top position in a variable
ellcont(0).Caption = Trim(CellsConfigered(1))
MsgBox (CStr(NumberOfcellControllers))
or Indexnumber = 1 To (NumberOfCellControllers - 1)
    If Indexnumber = 5 Or Indexnumber = 10 Or Indexnumber = 15 Or Indexnumber =
20 Or Indexnumber = 25 Or Indexnumber = 30 Or Indexnumber = 35 Or Indexnumber =
) Or Indexnumber = 45 Then
    'we are at the end of a line we need to drop the next line of boxes down a b
it.
        Load Cellcont(Indexnumber) 'load a new instance of the command button be
ng used.
        TopPosition = Cellcont(Indexnumber - 1).Top + 600 'indicate new top posi
ion for the control
        Cellcont(Indexnumber).Top = TopPosition 'set position of the control
        Cellcont(Indexnumber).Left = Cellcont(0).Left
        Cellcont(Indexnumber).Caption = Trim(CellsConfigered(Indexnumber + 1))'a
required to trim
        'the leading and trailing empty string spaces to fit properly on control
        Cellcont(Indexnumber).Visible = True
        GoTo NextIndex
    End If
    'otherwise we are loading a control that will fit on the current line of con
ols
    Load Cellcont(Indexnumber)
    Cellcont(Indexnumber).Top = TopPosition
    Cellcont(Indexnumber).Left = Cellcont(Indexnumber - 1).Left + 840
    Cellcont(Indexnumber).Caption = Trim(CellsConfigered(Indexnumber + 1)) 'are
required to trim
    'the leading and trailing empty string spaces to fit properly on control
    Cellcont(Indexnumber).Visible = True
```

```
NextIndex:
Next

'now we are required to set the number of required rows and columns in the grid
control in order
'to meet the number of nodes in the factory.
NumberOfNodes = NumberOfCellControllers + 1 'The number of nodes equals the num
er of cells plus
'the Factory Controller

'now reconfigure the grid to hold all data required to be defined.
TimeForTransport.Cols = (NumberOfNodes + 1) 'define number of columns.
TimeForTransport.Rows = (NumberOfNodes + 1) 'define number of rows.

'now we are required to set the column width and height to suit the number of it
ems being shown

'Set up first row and column
TimeForTransport.ColWidth(0) = 1000
TimeForTransport.RowHeight(0) = 500

'setup successive rows and columns
For i = 1 To (NumberOfNodes + 1)
    TimeForTransport.ColWidth(i) = 700
    TimeForTransport.RowHeight(i) = 325
Next

'now put a zero in each invalid Col\Row combination.
For i = 1 To NumberOfNodes
    TimeForTransport.Col = i
    TimeForTransport.Row = i
    TimeForTransport.Text = "0"
Next



'Now put a graphic in the first column and rows cell

'define row to add graphic to
TimeForTransport.Row = 0
TimeForTransport.Col = 0

'put graphic in it
TimeForTransport.Picture = GridImage.Picture

'now we are required to add the correct factory config data to the grid

'Select first row to fill first.
TimeForTransport.Row = 0
TimeForTransport.Col = 1
TimeForTransport.Text = "FACT"


For i = 1 To NumberOfCellControllers ' for each element in the row except the fi
st two elements add
```

```
'a Cell Controller name in the factory
    TimeForTransport.Col = i + 1
    TimeForTransport.Text = Trim(CellsConfigered(i))
Next

'this is fine but we must also fill the required column in row one.

'select column one as the column to fill first.
TimeForTransport.Col = 0
TimeForTransport.Row = 1
TimeForTransport.Text = "FACT"

For i = 1 To NumberOfCellControllers ' for each element in the column except the
 first two elements
 add a Cell Controller name in the factory
    TimeForTransport.Row = i + 1
    TimeForTransport.Text = Trim(CellsConfigered(i))
Next

'this is fine but we also are required to open the interarrival times and see if
 it has been
'previously configured.

Dim Thesenodestimedif() As NodeTimedelay
Dim TheseNodesRecord As NodeTimedelay
Dim Recordsize, recordlength, Filenum, j, s, Nextrecord As Integer


Filenum = FreeFile
recordlength = Len(TheseNodesRecord)

Open Netdrive & "Factory\Bidding\Nodetime\NodeDely.txt" For Random Access Read A
   Filenum Len = recordlength

'firstly we only require the first record to get the matrixsize
ReDim Thesenodestimedif(1)

'get the first record that stores the value of the matrix size
Get Filenum, 1, Thesenodestimedif(1)

'redimension to take all the records that are in the file
Recordsize = (Thesenodestimedif(1).Matrixdim * Thesenodestimedif(1).Matrixdim)
MsgBox CStr(Recordsize)

ReDim Thesenodestimedif(Recordsize)

'now get all the records
For i = 1 To Recordsize
    Get Filenum, i, Thesenodestimedif(i)
    'List1.AddItem CStr(Thesenodestimedif(i).Interarrivaltime)
    'List1.AddItem Thesenodestimedif(i).CellnameX
    'List1.AddItem Thesenodestimedif(i).Cellnamey
Next

   (Thesenodestimedif(1).Matrixdim) = 0 Then
    Close Filenum
```

```
DEFINE_L.FRM - 6

    Exit Sub
End If


Nextrecord = 0
For i = 1 To Thesenodestimedif(1).Matrixdim
        For j = 1 To Thesenodestimedif(1).Matrixdim
            TimeForTransport.Row = 0
            MsgBox CStr(i)
            MsgBox CStr(j)
            TimeForTransport.Col = i
            Nextrecord = Nextrecord + 1
            'MsgBox (CStr(TimeForTransport.Col))
            'MsgBox (Thesenodestimedif(Nextrecord).CellnameX)
            'MsgBox (TimeForTransport.Text)
            If Trim(Thesenodestimedif(Nextrecord).CellnameX) Like Trim(TimeForT
ansport.Text) Then
                    TimeForTransport.Row = j

                    TimeForTransport.Text = CStr(Thesenodestimedif(Nextrecord).
nterarrivaltime)
                    'Timefortransport.Row = j
                    'Timefortransport.Col = s
                    'Timefortransport.Text = CStr(Thesenodestimedif(s).Interarri
valtime)

            End If
        Next
Next


Close Filenum




End Sub

Sub Saveinternodal_Click ()

'Type NodeTimedelay
 '   MatrixDim As Integer
  'Interarrivaltime as integer
  'CellnameX As String * 10
  'CellnameY as String * 10
 'End Type

)im Thesenodestimedif() As NodeTimedelay
)im TheseNodesRecord As NodeTimedelay
)im Matrixsize, recordlength, Filenum, i, j  As Integer

1atrixsize = (TimeForTransport.Cols - 1)
```

```
'MsgBox (CStr(Matrixsize))
 redimension the 2-D array to hold all the records
ReDim Thesenodestimedif(Matrixsize, Matrixsize)

hesenodestimedif(1, 1).Matrixdim = Matrixsize

'now assign the fields to the array
or i = 1 To Matrixsize
    TimeForTransport.Col = i 'fix column from where you want the data to be read
 from
        For j = 1 To Matrixsize
'            MsgBox (CStr(j))
            TimeForTransport.Row = j 'define the cell that you want the data to
e read from by
            'selecting the row
            Thesenodestimedif(i, j).Interarrivaltime = Val(TimeForTransport.Text

'            MsgBox (CStr(Thesenodestimedif(i, j).Interarrivaltime))
            TimeForTransport.Row = 0
            Thesenodestimedif(i, j).CellnameX = TimeForTransport.Text
'            MsgBox (CStr(Thesenodestimedif(i, j).CellnameX))
            TimeForTransport.Row = j
            TimeForTransport.Col = 0
            Thesenodestimedif(i, j).Cellnamey = TimeForTransport.Text
'            MsgBox (CStr(Thesenodestimedif(i, j).Cellnamey))
            TimeForTransport.Col = i
        Next
Next

'this is all very fine but we must save it all to file so that it can be recover
ed as required
ilenum = FreeFile
recordlength = Len(TheseNodesRecord)

pen Netdrive & "Factory\Bidding\Nodetime\NodeDely.txt" For Random Access Write
As Filenum Len = recordlength


or i = 1 To Matrixsize
    For j = 1 To Matrixsize
            s = s + 1
            Put Filenum, s, Thesenodestimedif(i, j)
'            MsgBox (Thesenodestimedif(i, j).Cellname)
            'Thesenodestimedif(i, j).Cellname = TimeFortransport.Text
    Next
Next

lose Filenum

nd Sub

ub View_Current_Click ()
im Thesenodestimedif() As NodeTimedelay
im TheseNodesRecord As NodeTimedelay
im NumberOfElements, Matrixsize, recordlength, Filenum, i, j As Integer
```

```
Dim RootOfMatrixsize As Integer

'Type NodeTimedelay
 '   MatrixDim As Integer
  '  Cellname As String * 10
  '  ArrayXPosition As Integer
  '  ArrayYPosition As Integer
'End Type

'ReDim Thesenodestimedif(1, 1)


'this is all very fine but we must save it all to file so that it can be recover
ed as required
'Filenum = FreeFile
'recordlength = Len(TheseNodesRecord)

'Open Netdrive & "Factory\Bidding\Nodetime\NodeDely.txt" For Random Access Read
As Filenum Len = recordlength


'first we are required to get the variable that defines what the size of the or.
ginal matrix was
'Get Filenum, 1, Thesenodestimedif(1, 1)
'
'RootOfMatrixsize = Thesenodestimedif(1, 1).MatrixDim

'NumberOfElements = (RootOfMatrixsize * RootOfMatrixsize)

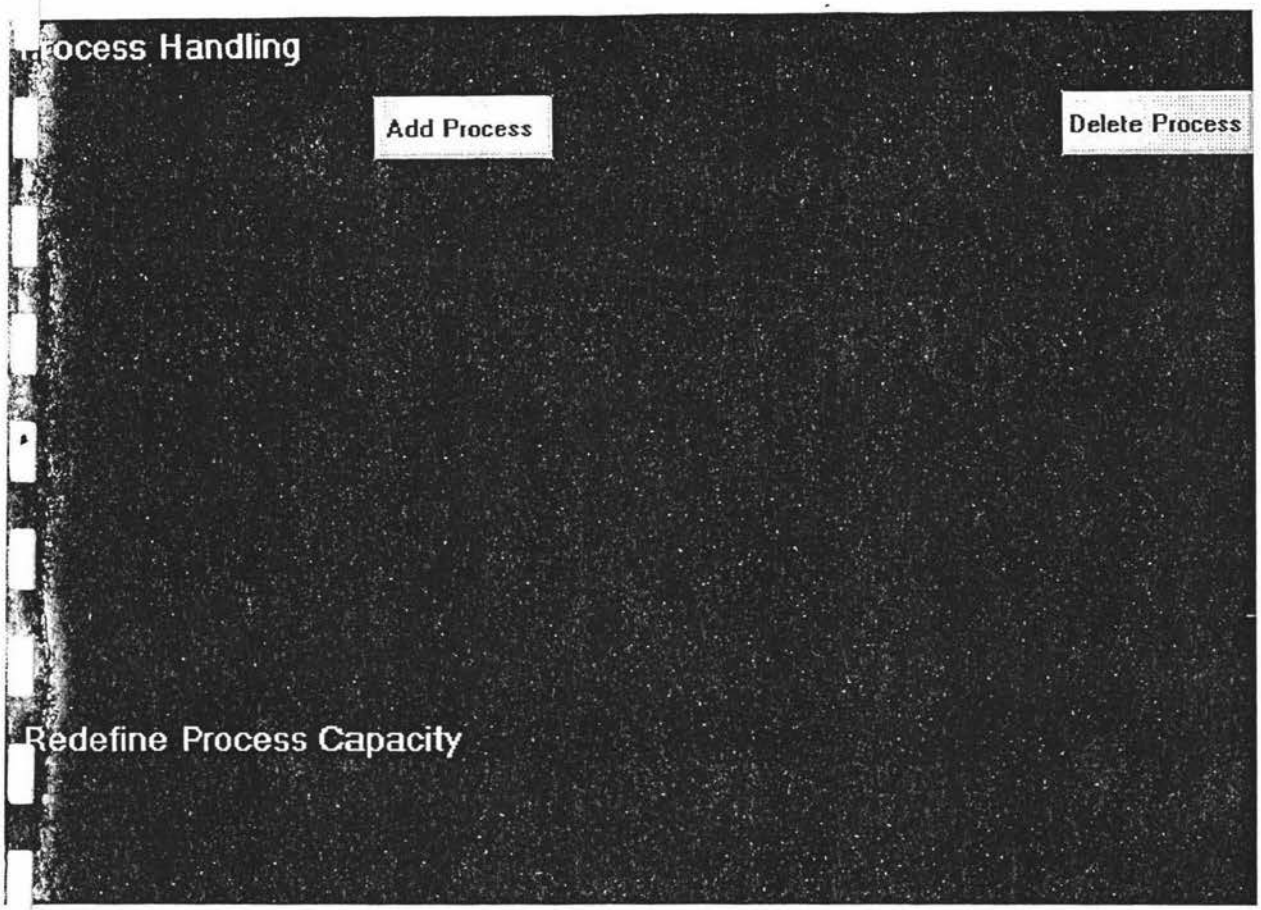
'ReDim Thesenodestimedif(RootOfMatrixsize, RootOfMatrixsize)

'For i = 1 To RootOfMatrixsize
 '   For j = 1 To RootOfMatrixsize
  '          s = s + 1
   '         Get Filenum, s, Thesenodestimedif(i, j)
    '        List1.AddItem CStr(Thesenodestimedif(i, j).ArrayXposition)
     '       List2.AddItem CStr(Thesenodestimedif(i, j).ArrayYposition)
      '      'Thesenodestimedif(i, j).Cellname = TimeFortransport.Text
    'Next
'Next

'Close Filenum




End Sub
```


rocess Handling
Add Process
Delete Process
Redefine Process Capacity

```
DEFINE_P.FRM - 1

Option Explicit

Option Base 1
Option Compare Text

Dim ThisProcessPosts As Postrecord
Dim ThisPostProcsarray() As Postrecord
Dim ProcTyp As String * 4
Dim thisfileslength As Integer
Dim i As Integer
Dim it As Integer 'Temporary variable
Dim Itaintsaved As Integer
Dim Filenam As String
Dim Firstlistcount As Integer
Dim Secondlistcount As Integer
Dim Processlistchanged As Integer


Sub Command2_Click ()
On Error Resume Next
it = it + 1

Text14.Text = ThisPostProcsarray(it).Recordleng
Text15.Text = ThisPostProcsarray(it).ProcessTyp
Text16.Text = ThisPostProcsarray(it).Index
Text17.Text = ThisPostProcsarray(it).PostTyp

End Sub

Sub Command3_Click ()
it = 0
i = 0
End Sub

Sub Add_ContName_Click ()

Dim Filenum As Integer
Dim i As Integer
Dim recordlength As Integer

'show current controllers
Def_Conts_Click

Def_CNC_Cont.Enabled = True
Save_Psts.Visible = True

'Set the flag that tells the user that a change has been made
Itaintsaved = 1


'Check to see if the user has put anything in the State_Cont_Name textbox
If Len(State_Cont_Name.Text) = 0 Then
    'There is nothing in the textbox so please tell the user to enter something
    MsgBox "You have not entered a Post-Processor type. Please do so.", MB_ICON
XCLAMATION + MB_OK, "Entry Error"
        Else
```

```
        'There is something there so add it to the file
        Def_CNC_Cont.AddItem Trim(State_Cont_Name.Text)

        'Type Postrecord
        'ProcessTyp As String * 50
        'Index As Integer
        'PostTyp As String * 50
        'End Type
        recordlength = Len(ThisProcessPosts)
        Filenum = FreeFile

        thisfileslength = thisfileslength + 1
        'redimension the array to hold all of the file and reserve
        'an extra record position for the new item being added

        ReDim Preserve ThisPostProcsarray(thisfileslength)

        'Open the file concerned
        Open Filenam For Random Access Read Write Shared As Filenum Len = recor
ength


        'Now write all file information to memory
        For i = 1 To (thisfileslength)
            Get Filenum, i, ThisPostProcsarray(i)
        Next

        'Now index the information in the first record to imply a new record is
eing add.
        ThisPostProcsarray(1).Recordleng = ThisPostProcsarray(1).Recordleng + 1
        ThisPostProcsarray(1).CurrentProcId = ThisPostProcsarray(1).CurrentProc
 + 1

        'Now add this information at the new position.
        ThisPostProcsarray(ThisPostProcsarray(1).Recordleng).ProcessTyp = ProcT

        ThisPostProcsarray(ThisPostProcsarray(1).Recordleng).Index = ThisPostPr
csarray(1).CurrentProcId
        ThisPostProcsarray(ThisPostProcsarray(1).Recordleng).PostTyp = Trim(Sta
e_Cont_Name.Text)
        'Once it is added then clear the textbox
        State_Cont_Name.Text = ""

Close Filenum

Dim ThisPostProcsarray() As Postrecord
'Type Postrecord
CurrentProcId As Integer
Recordleng As Integer
'ProcessTyp As String * 50
Index As Integer
PostTyp As String * 50
'End Type
```

```
End If


End Sub

Sub Add_Process_Click ()
'This button allows the user to add the name of a given process that will be
'operating in the Factory enviornment and then define the range of the
'capacity specification for it.
Dim Msg As String
Dim Reply As Integer
Dim Title As String
Dim Typ As Integer
Dim ProcessSpc() As ProcessDef
Dim Processtp As ProcessDef
Dim Filelength As Integer
Dim i As Integer
Dim Filenum As Integer


Def_Cap.Enabled = True

If Len(Process_Name.Text) = 0 Then
Msg = "You haven't stated the name of the process that you want to add "
Msg = Msg & "- Please do so."
Typ = MB_ICONEXCLAMATION + MB_OK
Title = "Process Name Error"
Reply = MsgBox(Msg, Typ, Title)
Exit Sub
     Else 'The name of a process is there so check to see if it hasn't been
     'added to the list already
           For i = 0 To ProcessList.ListCount
           'Trim off leading and lagging trailing string spaces of each
           'process name in the listbox and compare with the process
           'name wanting to be added
           If Trim(ProcessList.List(i)) = Process_Name.Text Then
               Msg = "The process cannot be added as it is already there."
               Typ = MB_ICONEXCLAMATION + MB_OK
               Title = "Process is Present"
               Reply = MsgBox(Msg, Typ, Title)
               Exit Sub
           End If
           Next
     'add the process to the list
     Processlistchanged = 1
     ProcessList.AddItem Process_Name.Text
     CurrId = CurrId + 1
     Processid.AddItem CStr(CurrId)

End If

End Sub

Sub Cancel_Click ()
```

```
EFINE_P.FRM - 4

im response As Integer



)ef_Conts.Visible = True
 ef_Conts.Enabled = False
Add_ContName.Visible = False
State_Cont_Name.Visible = False
.abel3.Visible = False
Cancel.Visible = False
ProcessList.Enabled = True
 ave_Psts.Visible = False
Label4.Visible = False


)ef_CNC_Cont.Clear



End Sub

 ub Def_Cap_Click ()
Dim IsFileThere As Integer
)im Msg As String
 )im Reply As Integer
Dim Title As String
)im Typ As Integer
 im X As Variant

On Error Resume Next
 Load name of process being configured into caption
Filenamed = Process_Name.Text

 Re_define.Timer1.Enabled = False

If Len(Trim(Process_Name.Text)) = 0 Then
 itle = "Redefine Error"
Msg = "You cannot redefine a processes capacity here. You must click the "
Msg = Msg & "'Redefine' button. This button is only for configuring new process
 ."
Typ = MB_ICONEXCLAMATION + MB_OK
Reply = MsgBox(Msg, Typ, Title)
 Exit Sub
 nd If

 Check to see if the
IsFileThere = FileLen("c:\Factory\Process\Specs\" & Trim(Filenamed) & ".bmp")

If Err = ERR_FILENOTFOUND Then
Msg = "Capacity bitmap does not exist for this process. Please create."
Title = "Capacity Bitmap Status"
Typ = MB_ICONEXCLAMATION + MB_OKCANCEL
Reply = MsgBox(Msg, Typ, Title)
   If Reply = IDCANCEL Then
```

```
   Exit Sub
      Else
        'open an new form and use OLE to interface with paintbrush as this is m
re reliable
        Define_CapBitmap.Show
        Define_CapBitmap.Label4.Visible = False
       'AppActivate "Microsoft Visual Basic"    'Change focus to Visual Basic.
       'X = Shell("PBrush.exe", 3)     ' Shell the Paintbrush
       'AppActivate "Paintbrush - (Untitled)"
       'SendKeys "%(F){DOWN}{ENTER}", True
       'SendKeys "{TAB}{TAB}{UP 3}{ENTER}{DOWN 5}", True
       'SendKeys "{ENTER}{DOWN 2}{ENTER}{DOWN 3}", True
       'SendKeys "{ENTER}{TAB}{TAB}{TAB}{TAB}{TAB}{TAB}{TAB}{DOWN}{ENTER}", Tru
       Exit Sub
   End If
Else
'No 'File Not Found ' error therefore the bitmap exists so load it into the
'"Define capacity" form

Load Define_Capacity
Define_Capacity.Show
End If

Exit Sub
Fixerror:
Reply = ChekError(Err)

Select Case Reply

Case Is = 1
Exit Sub

Case Is = 2
Resume Next

Case Else
Exit Sub


End Select

End Sub

Sub Def_CNC_Cont_Click ()
Del_Button.Enabled = True
End Sub

Sub Def_Conts_Click ()

'Define local variables
Dim Filenum As Integer


'Type Postrecord
'ProcessTyp As String * 50
'Index As Integer
```

```
DEFINE_P.FRM - 6

'PostTyp As String * 50
'End Type


'Firstly we must configure the available button sequence

Redefine.Enabled = False
Red_Cap.Enabled = False
Def_Cap.Enabled = False
Add_Process.Enabled = False
Delete_Proc.Enabled = False
Def_CNC_Cont.Enabled = True
'Add_ContName.Visible = True
Cancel.Visible = True
State_Cont_Name.Visible = True
Def_Conts.Visible = False


If ProcessList.ListIndex = -1 Then
    MsgBox "Sorry but you must select a process from the list for which you are
defining the controllerts for"
    Exit Sub
    Else
    Label4.Caption = "Save " & Trim(ProcessList.List(ProcessList.ListIndex)) & '
Post-Processor File."
End If


Label11.Visible = True
Del_Button.Visible = True

'Type Postrecord
'CurrentIndex As Integer
'CurrentProcId As Integer
'Recordleng As Integer
'ProcessTyp As String * 50
'Index As Integer
'PostTyp As String * 50
'End Type

'Dim ThisProcessPosts As Postrecord
'Dim ThisPostProcsarray() As Postrecord

'Now we must open the file that contains the names of all the post-processors
'for the chosen piece of equipment
Filenum = FreeFile
ProcTyp = Trim(ProcessList.List(ProcessList.ListIndex))
Filenam = "c:\factory\Process\Postname\" & Trim(ProcTyp) & "proc.pps"
thisfileslength = GetFileslength4(Filenam, ThisProcessPosts, ThisPostProcsarray
), Filenum)


If thisfileslength < 1 Then
    'there is nothing in the file to add to the listbox therefore exit the sub
    Close Filenum
```

```
    Exit Sub
    Else
        'There is something in the file therefore add the information the need
o display
        'to the listbox and write remaining information to memory.
        Call ReadAndAddToList2(ThisPostProcsarray(), thisfileslength, Filenum)
        Close Filenum
End If

End Sub


Sub Del_Button_Click ()
On Error Resume Next


'this particular button allows the user to select a post
'processor type from the list and remove it by hitting
'this delete button

'Declare local variables
Dim Itemasselected As String
Dim DeletePosition As Integer
Dim Reply As Integer

Save_Psts.Visible = True

Reply = MsgBox("Are you sure that you want to delete the selected item?", MB_IC(
NEXCLAMATION + MB_YESNOCANCEL, "Are You Sure?")


'Check the answer
If Reply = IDNO Or Reply = IDCANCEL Then
    Exit Sub
    Del_Button.Enabled = False
End If


Itemasselected = Def_CNC_Cont.List(Def_CNC_Cont.ListIndex)

'Now we check to see whether or not the item selected from
'the listbox has a match from the file - it it has, which it will
'we need to find the position of the match so we know what position
'to delete.

For i = 1 To thisfileslength
    If Trim(Itemasselected) = Trim(ThisPostProcsarray(i).PostTyp) Then 'if there
 is a match
        DeletePosition = i
    End If
Next

'Now we must delete the position from the array before saving the array to file

If DeletePosition = 1 And Def_CNC_Cont.ListCount > 1 Then
    ThisPostProcsarray(1).Recordleng = ThisPostProcsarray(1).Recordleng - 1
```

```
DEFINE_P.FRM - 8

    ThisPostProcsarray(1).ProcessTyp = ThisPostProcsarray(2).ProcessTyp
    ThisPostProcsarray(1).PostTyp = ThisPostProcsarray(2).PostTyp
    ThisPostProcsarray(1).Index = ThisPostProcsarray(2).Index

    For i = 2 To (thisfileslength - 1)
        ThisPostProcsarray(i).ProcessTyp = ThisPostProcsarray(i + 1).ProcessTyp
        ThisPostProcsarray(i).Index = ThisPostProcsarray(i + 1).Index
        ThisPostProcsarray(i).PostTyp = ThisPostProcsarray(i + 1).PostTyp
    Next

        ThisPostProcsarray(thisfileslength).ProcessTyp = ""
        ThisPostProcsarray(thisfileslength).Index = 0
        ThisPostProcsarray(thisfileslength).PostTyp = ""

        'Now remove from the listbox
        Def_CNC_Cont.RemoveItem Def_CNC_Cont.ListIndex
Itaintsaved = 1
Del_Button.Enabled = False
Exit Sub
End If

If DeletePosition = 1 And Def_CNC_Cont.ListCount = 1 Then

    ThisPostProcsarray(1).Recordleng = 0
    ThisPostProcsarray(1).ProcessTyp = ""
    ThisPostProcsarray(1).PostTyp = ""
    ThisPostProcsarray(1).Index = 0
    Def_CNC_Cont.RemoveItem Def_CNC_Cont.ListIndex
Itaintsaved = 1
Exit Sub
End If


If DeletePosition = thisfileslength Then
'we are deleting the last record from the file

    'Delete the last position from the array
    ThisPostProcsarray(thisfileslength).ProcessTyp = ""
    ThisPostProcsarray(thisfileslength).Index = 0
    ThisPostProcsarray(thisfileslength).PostTyp = ""

    'Modify the first record to reflect that a record has been removed
    ThisPostProcsarray(1).Recordleng = (ThisPostProcsarray(1).Recordleng - 1)
    Def_CNC_Cont.RemoveItem Def_CNC_Cont.ListIndex

Itaintsaved = 1
Del_Button.Enabled = False
Exit Sub
End If

If DeletePosition <> 1 And DeletePosition <> thisfileslength Then
'If the item being deleted is not at the beginning or end of the file

    'Modify the first record to reflect that a record is been removed
    ThisPostProcsarray(1).Recordleng = (ThisPostProcsarray(1).Recordleng - 1)
```

```
    'Remove the record and arrange everything else around it
        For i = DeletePosition To thisfileslength - 1
            ThisPostProcsarray(i).ProcessTyp = ThisPostProcsarray(i + 1).Proces
Typ
            ThisPostProcsarray(i).PostTyp = ThisPostProcsarray(i + 1).PostTyp
            ThisPostProcsarray(i).Index = ThisPostProcsarray(i + 1).Index
        Next

    ThisPostProcsarray(thisfileslength).ProcessTyp = ""
    ThisPostProcsarray(thisfileslength).PostTyp = ""
    ThisPostProcsarray(thisfileslength).Index = 0
    Def_CNC_Cont.RemoveItem Def_CNC_Cont.ListIndex
Itaintsaved = 1
Del_Button.Enabled = False
Exit Sub
End If

'Type Postrecord
'CurrentIndex As Integer(not used)
'CurrentProcId As Integer
'Recordleng As Integer
'ProcessTyp As String * 50
'Index As Integer
'postTyp As String * 50
'End Type




Del_Button.Enabled = False
End Sub

Sub Delete_Proc_MouseDown (Button As Integer, Shift As Integer, X As Single, Y
s Single)
'This sub procedure allows the user to delete a process as
'required
Dim Msg As String
Dim Reply As Integer
Dim Title As String
Dim Typ As Integer
Dim ProcessSpec() As ProcessDef
Dim ProcessTyp As ProcessDef
Dim Filelength As Integer
Dim i As Integer
Dim Filenum As Integer
Dim Processdeleted As String * 10
Dim Count As Integer
Dim Check As Integer
Dim Thisind As Integer
Count = 0
'Check to see if anything has been selected by checking to see if there are
'any characthers in the last selected item - if there are no characthers
'then the top of the listbox will be empty and therefore the value of it's
```

```
)EFINE_P.FRM - 10

'trimmed stringlength will be equal to zero

If Len(Trim(ProcessList.List(ProcessList.ListIndex))) = 0 Then
'Nothing has been selected so ask the user to select an item from the list
Msg = "You didnt select an item to be removed from the process list - "
Msg = Msg & "if you want to remove an item from this list please select '"
Msg = Msg & "it and click the 'Delete Process' button."
Typ = MB_ICONEXCLAMATION + MB_OK
Title = "Selection Failure"
Reply = MsgBox(Msg, Typ, Title)
Exit Sub
End If

Msg = "Do you really want to remove the selected item from the list? "
Typ = MB_ICONQUESTION + MB_YESNO
Title = "Confirm Deletion"
Reply = MsgBox(Msg, Typ, Title)

'If the user does not want the item to be deleted then exit the sub procedure
If Reply = IDNO Then
Exit Sub
End If

Processdeleted = ProcessList.List(ProcessList.ListIndex)
Thisind = ProcessList.ListIndex
ProcessList.RemoveItem (ProcessList.ListIndex)
Processid.RemoveItem (Thisind)
MaxCapacity.RemoveItem (Thisind)

'Type ProcessDef
'ProcessNumber As Integer
'Processname As String * 20
'CurrentProcessId As Integer
'ProcessId As Integer
'Maxcap As Integer
'End Type

End Sub

Sub Form_Load ()
Dim Msg As String
Dim Typ As Integer
Dim Title As String
Dim Reply As Integer
Dim ProcessSpec() As ProcessDef
Dim ProcessTyp As ProcessDef
Dim Filelength As Integer
Dim i As Integer
Dim Filenum As Integer

Type ProcessDef
ProcessNumber As Integer
'Processname As String * 20
CurrentProcessId As Integer
ProcessId As Integer
'Maxcap As Integer
```

```
'End Type


'When this form is loaded a procedure must be fired whereby a file containing
'the names of all the processes available are written into a combo box.This
'information is required in the job definition process
Filenum = FreeFile
Filelength = Len(ProcessTyp)
ReDim ProcessSpec(1)
Open "c:\Factory\Process\files\Proclist.Pfl" For Random Access Read Write Share
 As Filenum Len = Filelength
Get Filenum, 1, ProcessSpec(1)

If ProcessSpec(1).processnumber = 0 Then
Title = "File Empty"
Msg = "There is nothing in this file yet it will have to be created"
Typ = MB_ICONEXCLAMATION + MB_OK
Reply = MsgBox(Msg, Typ, Title)
 Close Filenum
Exit Sub
End If

'else there is something there so add to list
ReDim Preserve ProcessSpec(ProcessSpec(1).processnumber)
For i = 1 To ProcessSpec(1).processnumber
 Get Filenum, i, ProcessSpec(i)
 ProcessList.AddItem ProcessSpec(i).Processname
 MaxCapacity.AddItem CStr(ProcessSpec(i).Maxcap)
 Processid.AddItem CStr(ProcessSpec(i).Processid)
Next

CurrId = ProcessSpec(1).CurrentProcessId
Close Filenum


'we are using two variables called firstlistcount and secondlistcount to find c
t whether there
'have been processes added to the system during event procedures on this form. T
e then write a
'procedure in the form to make sure that all parameters are defined for the tim
 the form is open.
'If a new process has been defined and all parameters for the new process are n
t defined the
'form will not be allowed to unload until all parameters are defined.

'record number of processes defined on opening the form
'Firstlistcount = ProcessList.Listcount

End Sub

Sub Form_Unload (Cancel As Integer)
Dim Msg As String
Dim Reply As Integer
Dim Title As String
Dim Typ As Integer
Dim ProcessSpec() As ProcessDef
```

```
DEFINE_P.FRM - 12

Dim ProcessTyp As ProcessDef
Dim Filelength As Integer
Dim i As Integer
Dim Filenum As Integer
Dim response As Integer


If Processlistchanged = 1 Then ' a process has been added

    response = MsgBox("A new process has been defined - do you want to save cha
ges? ", MB_ICONQUESTION + MB_YESNO, "Save it?")

    If response = IDYES Then
        GoTo Savit1
        Else
        GoTo Savit2
    End If

End If


Savit1:

Kill "c:\Factory\Process\Files\Proclist.Pfl"
    'Open the new version of the file
Filenum = FreeFile
Filelength = Len(ProcessTyp)


If ProcessList.ListCount = 0 Then 'there is nothing to save so leave the applic
tion
    Exit Sub
End If

'Redimension array to take the whole list of processes
ReDim ProcessSpec(ProcessList.ListCount)
Open "c:\Factory\Process\Files\Proclist.Pfl" For Random Access Read Write Share
 As Filenum Len = Filelength

ProcessSpec(1).processnumber = ProcessList.ListCount
ProcessSpec(1).CurrentProcessId = CurrId

Put Filenum, 1, ProcessSpec(1)

For i = 1 To ProcessList.ListCount
    ProcessSpec(i).Processname = ProcessList.List(i - 1)
    ProcessSpec(i).Processid = Processid.List(i - 1)
    ProcessSpec(i).Maxcap = MaxCapacity.List(i - 1)
    Put Filenum, i, ProcessSpec(i)
Next

Type ProcessDef
'ProcessNumber As Integer
'Processname As String * 20
CurrentProcessId As Integer
'ProcessId As Integer
```

```
DEFINE_P.FRM - 13

'Maxcap As Integer
'End Type



Close Filenum

Saveit2:
If Itaintsaved = 1 Then
    response = MsgBox("Changes have been made to the post-processor records, do
you want to save them?", MB_ICONQUESTION + MB_YESNO, "Save it?")

    If response = IDYES Then
        Save_Psts_Click
    End If

End If


End Sub


Sub ProcessList_Click ()
'Process_Name.Text = ""

Redefine.Enabled = True
Def_Conts.Enabled = True

ProcessList.Enabled = True

Def_Conts.Visible = True
Add_ContName.Visible = False
State_Cont_Name.Visible = False
Label3.Visible = False
Cancel.Visible = False

Save_Psts.Visible = False
Label4.Visible = False
Redefine.Enabled = True
Red_Cap.Enabled = True
Add_Process.Enabled = True
Delete_Proc.Enabled = True
Def_Cap.Enabled = True
End Sub

Sub Red_Cap_Click ()
'This button allows the user to make alterations to the capacity bitmap if requ
red

'Define variables
Dim Filenum As Integer
Dim Reply As String
Dim CaptionMessage As String

On Error Resume Next
```

```
'first check to see if something has been selected
If Len(ProcessList.List(ProcessList.ListIndex)) <> 0 Then 'if something has bee
 selected
    'open the process cap bitmap construction form
    Define_CapBitmap.Show
    Filenamed = Trim(ProcessList.List(ProcessList.ListIndex))
    Filenum = FreeFile

    Open "C:\Factory\Process\Specs\" & Trim(Filenamed) & ".bmp" For Binary As #
'ilenum
    Define_CapBitmap.OLE1.FileNumber = Filenum
    Define_CapBitmap.OLE1.Action = 12
    Close Filenum

    'change the caption
    CaptionMessage = "To modify the current capacity bitmap select the bitmap w
:h the mouse"
    CaptionMessage = CaptionMessage & " this will fire the paintbrush applicati
1 where the"
    CaptionMessage = CaptionMessage & " modification can be executed."

    Define_CapBitmap.Label4.Visible = True
    Define_CapBitmap.Label4.Caption = Trim(CaptionMessage)


        Else MsgBox "Please select a process for which the bitmap is to be alte
ed from the list.", MB_OK, "Selection Error"
        Exit Sub

End If

End Sub

Sub Redefine_Click ()
'This process allows the user to select a particular machine or process type
 from the processlist.Clicking this button then shows the capacity bitmap
'for that particular process and allows the user to restate the maximum
 capacity rating for that process.This circumstance could arise whereby a new
 machine of a type already defined previously is being added to the factory
'and that machine has a larger capacity than all other machines in the factory


'Define procedure level variables
Dim Msg As String
Dim Typ As Integer
Dim Title As String
Dim Reply As Integer

'Check to see if a process to reconfigure has been selected - if not
'prompt the user to select one and the exit the procedure
If Len(Trim(ProcessList.List(ProcessList.ListIndex))) = 0 Then
   Msg = "You haven't selected a process to redefine. Please select one from the
"
   Msg = Msg & " 'Process List' and reselect the 'Redefine' button."
   Title = "Select Redefine Process"
```

```
   Typ = MB_ICONEXCLAMATION + MB_OK
   Reply = MsgBox(Msg, Typ, Title)
   Exit Sub
End If

'Give the name of the process selected to the global variable called
'filename so that it's value can be passed onto the 'Redefine Capacity
'form.
Filenamed = ProcessList.List(ProcessList.ListIndex)

Text14.Text = Filenamed

'Now we must reload the 'redefine' form and with this load the correct
'capacity bitmap conforming to the process to be reselected.It would be
'possible to use the same form as that used to define the process in the
'first place and set a global flag in this procedure(Redefine_Click) such
'that this flag is recognised in the define_form_Unload procedure and take
'the appropriate redefining action - however it was decided that this process
'would be too cumbersome and that it would be easier to load a new 'Redefine'
'form.

Load Re_Define
Re_Define.Show
Re_Define.Caption = "Redefining the " & Trim(Filenamed) & "'s" & " Capacity"


End Sub

Sub Save_Psts_Click ()

'Declare form level variables
Dim Filenum As Integer
Dim Filenamedas As String

Filenamedas = "c:\factory\Process\Postname\" & ProcTyp & "proc.pps"

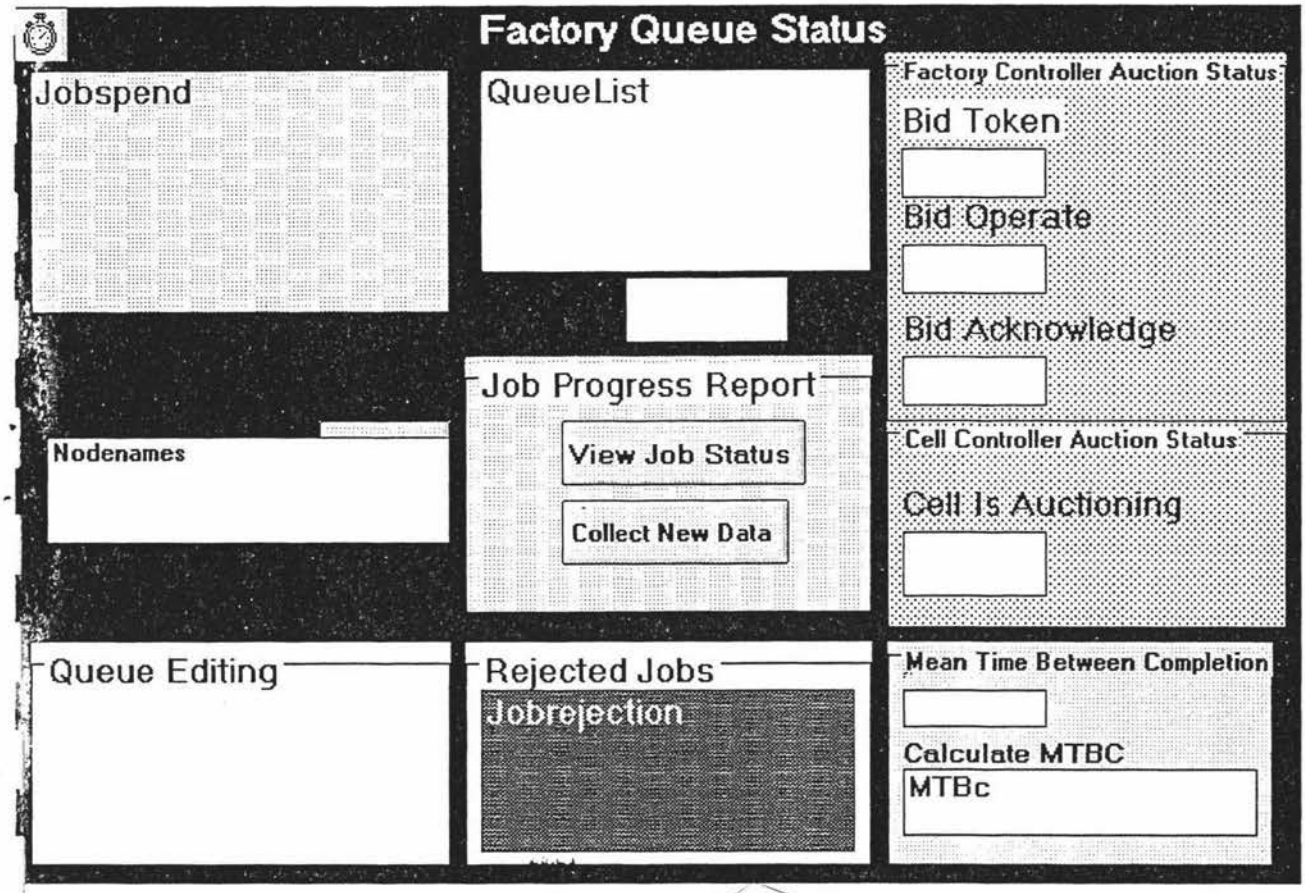
If Def_CNC_Cont.ListCount = 0 Then 'nothing left to save in the file so kill it
    Kill Filenamedas
    Exit Sub
End If

'When this button is pressed we want the contents of the
'Post-Processor definition array saved to file
Filenum = FreeFile
Filenamed = "c:\factory\Process\Postname\" & ProcTyp & "proc.pps"

Call OpenAndWrite(Filenum, Filenamedas, ThisProcessPosts, ThisPostProcsarray(),
thisfileslength)

Close Filenum
Itaintsaved = 2


End Sub
```


Factory Queue Status
Jobspend
QueueList
Factory Controller Auction Status
Bid Token
Bid Operate
Bid Acknowledge
Job Progress Report
View Job Status
Collect New Data
Nodenames
Cell Controller Auction Status
Cell Is Auctioning
Queue Editing
Rejected Jobs
Jobrejection
Mean Time Between Completion
Calculate MTBC
MTBc

QUEUESTA.FRM - 1

```
Option Explicit

Dim PositionToMoveto As Integer
Dim JobpositionToMove As Integer
Dim CheckSelected As Integer

Dim ThisBidding(1) As Bid_data_To_Be_sent
Dim Thisbid(1) As Nodesbid


Dim Ghandler As Long
Dim fart As Variant
Dim eda_err As Variant
Dim tex As Integer
Dim Newnum As Variant
Dim Node As String * 15
Dim Tags As String * 15
Dim Field As String * 15
Dim answer As String * 80
Dim ans As String * 10
Dim defnode As Long
Dim trap As String * 10
Dim Texts As String * 6
Dim Nodetyp As String * 20
Dim nodes As String * 20
Dim bidrotenable As Variant
Dim Blocktyp As String * 20
Dim Bidoperatestate As Variant
Dim Bidnodetext As String * 10
Dim bidtokenstate As Variant
Dim Tokenstate As Integer
Dim BidOpstate As Variant
Dim X As Integer
Dim Dummy As Variant

'bidding variables
Dim Nodename As String * 8
Dim VMDNumber As String * 10
Dim Jobname As String * 10
Dim Process_Busy_Auctioning As Integer
Dim Filetoopen As String * 30

' functions specific to VB (not standard eda fcns)
Declare Sub eda_version Lib "vdba.dll" (ByVal v$)
Declare Sub eda_get1_ascii Lib "vdba.dll" (ByVal n$, ByVal t$, ByVal f$, ByVal
$)
Declare Function eda_set1_ascii% Lib "vdba.dll" (ByVal n$, ByVal t$, ByVal f$, B
yVal v$)
'
' Note define_group returns a long
'
Declare Function eda_define_group& Lib "vdba.dll" (ByVal Count%, ByVal Detect%)
Declare Sub eda_delete_group Lib "vdba.dll" (ByVal Handle&)
'
' should always pass 0 for vsp
Declare Function eda_define_ntf& Lib "vdba.dll" (ByVal Handle&, ByVal enode$, By
```

```
Val etag$, ByVal efield$, ByVal vsp&)
Declare Sub eda_delete_ntf Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
'
Declare Sub eda_lookup Lib "vdba.dll" (ByVal Handle&)
Declare Sub eda_read Lib "vdba.dll" (ByVal Handle&)
Declare Sub eda_write Lib "vdba.dll" (ByVal Handle&)
Declare Function eda_write1% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
Declare Sub eda_wait Lib "vdba.dll" (ByVal Handle&)
Declare Function eda_query_wait% Lib "vdba.dll" (ByVal Handle&)
'
Declare Function eda_get_error% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
Declare Function eda_get_ascii% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, ByVal
 value$, ByVal maxlen%)
Declare Function eda_set_ascii% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, ByVal
 value$)

' note pointer to float is passed
Declare Function eda_get_float% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, value
!)
Declare Function eda_set_float% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, value
!)


Mean Time Between Completion variables
Dim K As Integer 'number of jobs just completing
Dim Tk_1 As Double 'time previous job completed
Dim MTBc_K_1 As Double 'MTBC at last time interval
Dim MTBcK As Double 'mtbc for the current time interval

Dim Montharray(12) As Integer
Dim StartTime As Double
Dim Sim_Start As Integer
Dim NumberofLoops As Integer

Function CalcStartTime ()

Dim Currentmonth As String * 2
Dim Daystring As String * 2
Dim CurrentDay As Integer
Dim Arrayposition As Integer
Dim p As Integer
Dim NoDays As Integer
Dim NumberOfHours As String
Dim FirstTime As Integer
Dim SecondTime As Double
Dim Thirdtime As Double
Dim Totaltime As Double

'Due date bid calc dec
Dim Thistime As String * 8'5
Dim Thisdate As String * 15

'tells the program where to start recording the data
'Rowpointer = 1

Montharray(1) = 31
```

```
Montharray(2) = 29
Montharray(3) = 31
Montharray(4) = 30
Montharray(5) = 31
Montharray(6) = 30
Montharray(7) = 31
Montharray(8) = 31
Montharray(9) = 30
Montharray(10) = 31
Montharray(11) = 30
Montharray(12) = 31



Thistime = Time$
'MsgBox (Thistime)

'02:07:35

Thisdate = Date$


'get the first part of the due date time which is the number of hours
'up to the present time
FirstTime = Val(Thistime)


'in the 09:20 hrs example above we also want to get the :20 part in
'terms of hours to get the total hrs in the day,this part does this
SecondTime = (Val(Right$(Thistime, 5))) / 60


Thirdtime = (Val(Right$(Thistime, 2))) / 3600

'MsgBox ("the number of hours is " & FirstTime)
'MsgBox ("the number of Minutes is " & CStr(SecondTime) & " Hours")

'calculate the total hours in the day
Totaltime = FirstTime + SecondTime + Thirdtime

'MsgBox (CStr(Totaltime))

'get the string value of the task's due date month
Currentmonth = (Thisdate)

'MsgBox CStr(Currentmonth)
'MsgBox CStr(Thisdate)
'
'transfer the current month into an integer eg:for 08-20-1996 we are getting th
'fact that it is month "8"
Arrayposition = CInt(Currentmonth)

'extract the right part of the "thisdate" string so we can
'access the day part of it ie: the fact that it is day 20 for the above
Daystring = (Right$(Trim(Thisdate), 7))
```

```
 the Daystring variable into an integer so that we can use it mathemati

us a day to say that although we are in the day we have not used all of

-y = CInt(Daystring) - 1

osition = 1 Then
monthlicount


 the days of the months up to the current month
1 To (Arrayposition - 1)
s = Montharray(p) + NoDays
ox CStr(NoDays)


licount:

 number of days in the current month that have been used
 NoDays + CurrentDay
(CStr(NoDays))

e the number of hours this represents
Hours = CStr((NoDays * 24) + Totaltime)


tTime = Totaltime + (NoDays * 24)


tion

and1_Click ()
fact"
"Updatemtbc"
"A_CV"
"0"


dinfo(Node, Tags, Field, Texts)


mand4_Click ()


mDDElink (Text As TextBox, LinkItem)
b Procedure creates an Automatic link between the
g application and either the Factory Controller or
l Controller

kMode = none   'If the link is not already
                        'created then create it
kTopic = "DMDDE|DATA"  'Link topic is FixDmacs

nkItem = LinkItem       'This is the Node,
```

```
fart = eda_set_ascii(Ghandler, defnode, Newnum)
Call eda_write(Ghandler)
Call eda_wait(Ghandler)
Call eda_delete_group(Ghandler)

End Sub

Sub Bid_Ack_Change ()
'this box is here so that it can recieve a flag from the VMD that the job was a
arded to and thus
'it can restart the rotation of the bid token
Dim ThisVMDNumber As Integer

Select Case Val(Bid_Ack.Text)

Case Is = 1

If FactoryIsAuctioning = 1 Then 'then it is the Factory Controller that is hold
ng the auction
'therefore we need to release the bidtoken from there

    Node = "fact" ' Node to talk to is the Factory Controller

    Field = "A_CV"

    Blocktyp = "Bidoperate"

    Texts = "0"

    Call sendinfo(Node, Blocktyp, Field, Texts)
    Tool_Avle.Tool_Timer.Enabled = True
    Else
        'it is a cell controller that is holding an auction therefore we are re
uired to tell the
        'cell controller that the auction is complete and that it can either re
ease the bidtoken or
        'try and auction another job


        'convert the current value of the VMD into an integer so that it can be
incremented
        ThisVMDNumber = Val(VMDNumber)

        'MsgBox (thisstring)
        'now increment the variable
        ThisVMDNumber = ThisVMDNumber + 1

        'MsgBox (node)
        'and send it back to the node that conducted the auction
        Texts = CStr(ThisVMDNumber)
        Tags = "VMDCURAUCT"
        Node = Trim(Nodename)
        Field = "A_CV"


        'MsgBox ("Resetting Cell " & Nodename)
```

```
UEUESTA.FRM - 9

        'MsgBox ("The VMDNumber is " & Texts)
        'MsgBox (Texts)
        Call sendinfo(Node, Tags, Field, Texts) 'What this will do is give the
 de that has just
        'tried to auction a job another chance to have a go with another VMD.
        Tool_Avle.Tool_Timer.Enabled = True
nd If
 now reset the BIDACKNOWL database block so that it is able to respond nexttime

exts = "0"

Tags = "BIDACKNOWL"

 ode = "Fact"

Call sendinfo(Node, Tags, Field, Texts)


Case Else 'if the box takes on any other value exit the sub
 xit Sub

End Select

 nd Sub

 b Bidtimer_Timer ()
 his timer waits for sometime for the bids to be recieved from the Factories C
ll Controller nodes
 once it times out it begins the comparison process of the bids regardless of w
 ther any bids are
'recieved or not.

 The first thing we are required to do is to open up the "nodefile.txt" file so
as to discover
'what nodes we are required to evaluate bids for.
 When this timer event occurs it must open a network file called "r:\Factory\Co
fig\nodefile.txt"
'which contains the names of all the nodes for which the vmd's have been config
 ed.

Dim recordlength As Integer
 im Filenum As Integer
im ThisRecord As nodelistrecord
 im position As Integer
 im Firstime As Integer
im Thisnode() As Bidevaluation
Dim i As Integer
 im NumberOfNodes As Integer
im Filename As String
Dim Thisdirectory As String
 Dim node As String
im Field As String
Dim Tag As String
 im Text As String
 im WinningNode As String
Dim thisstring As String
```

```
Dim ThisVMDNumber As Integer
Dim Msg As String
Dim Typ As Integer
Dim Title As String
Dim CellToGiveBackTo As String
Dim Vmdtogivebackto As String
Dim Matchposition As Integer
Dim IsThisNodeFinished As String * 10
Dim Thistime As Double
Dim Thattime As Double
Dim CellBidFin() As BidIsFinished
Dim NotFinished As Integer
Dim MaxNumBerOfLoops As Integer
Dim Thisblock As String * 20


On Error Resume Next

'MsgBox ("Bid Comparison working")
'MsgBox ("VMDNumber is " & VMDNumber)

'disable while this procedure is executing
Bidtimer.Enabled = False



'first we are required to see if all bids have been formed
MaxNumBerOfLoops = 200

NumberofLoops = 0

If FileLen("C:\Factory\Config\nodefile.txt") = 0 Then 'if there is nothing in t
e file then exit
'this subprocedure
    Exit Sub
End If



seeagain:

If NumberofLoops = MaxNumBerOfLoops Then
    MsgBox ("Contract Net Protocol System corrupted")
    Exit Sub
End If


NotFinished = 0

'Assign a free number to this file
Filenum = FreeFile
'Calculate the recordlength in Bytes
recordlength = Len(ThisRecord)

'Open the file for random access with allowable functions read and write
Open "C:\Factory\Config\nodefile.txt" For Random Access Read Shared As Filenum
```

```
UEUESTA.FRM - 11

en = recordlength

position = 1

o Until EOF(Filenum)
    'get the name of the first node from the file
    Get Filenum, position, ThisRecord
    'increment position variable so the next nodename will be gotten
    position = position + 1
'    MsgBox CStr(position)
    'now send a "1" to the "bidfrmstsrt" in the cell denoted by the record.
    Node = Trim(ThisRecord.Node)

    If Trim(Node) Like "FACT" Or Len(Trim(Node)) >= 10 Then
        'we do not need to send a prompt to the factory controller node
        GoTo NextBit
    End If


    'now we are required to request the bidfinished information for this node

    Node = "FACT"
    Tags = Trim(ThisRecord.Node) & "BIDFIN"
     MsgBox (Tags)
    Field = "A_CV"

    IsThisNodeFinished = eda_requestinfolink(Trim(Node), Tags, Field)

    'MsgBox (IsThisNodeFinished)
    ReDim Preserve CellBidFin(position - 2)

   ' Type BidIsFinished
    '    NodeName As String * 20
        'BidFin As Integer
    'End Type

    CellBidFin(position - 2).Bidfin = Val(IsThisNodeFinished)
    CellBidFin(position - 2).Nodename = Trim(ThisRecord.Node)

extBit:
Loop

lose Filenum


oEvents

'now we are required to see if all the required bids have been formed
or i = 1 To (position - 3)
    If CellBidFin(i).Bidfin = 0 Then
        NotFinished = 1
    End If
Next

f NotFinished = 1 Then 'then we are required to try again until it is
```

```
    Thistime = Timer

Loopit:

    Thattime = Timer

    If (Thattime - Thistime) > .1 Then
        NumberofLoops = NumberofLoops + 1

        'every 1 second yield to windows to allow Dmacs to talk
        Select Case NumberofLoops

        Case Is = 10, 20, 30, 40, 50, 60, 70, 80, 90
            DoEvents
        End Select

        GoTo seeagain
        Else
            GoTo Loopit
    End If

End If


'at this stage we are required to reset the "Nodename" & BidFin blocks so that
hey are able to
'respond nexttime

For i = 1 To (position - 3)

    Field = "A_CV"

    Node = "Fact"

    Blocktyp = Trim(CellBidFin(i).Nodename) & "BIDFIN"

  '  MsgBox (Blocktyp)
    Texts = "0"

    Call sendinfo(Node, Blocktyp, Field, Texts)

Next


'*****IF WE GET TO HERE ALL THE REQUIRED BIDS HAVE BEEN RECIEVED


'see if it is the cell controller auctioning - if it is skip this next part
If FactoryIsAuctioning = 0 Then
    GoTo SkipFactStuff:
End If

'now when this event occurs we may have the situation whereby another job has b
en added from
'the jobs pending list and the job that we are auctioning is actually on the Fac
```

```
tory controller
'queue, therefore we have to check to see where the new position of the job
'being auctioned is - it may well be in the same position in the list but it ma
 well not be
'too.
If Busy_Jobs_Auction = 0 Then 'then a job is being auctioned off the main Facto
y Queue therefore
    For i = 1 To QueueList.ListCount 'we are required to find a match with the
ob on this queue
        'find a match between the job being auctioned and the one in the list
        If Trim(Filetoopen) = Trim(QueueList.List(i - 1)) Then
            Matchposition = (i - 1)
        End If
    Next
    Else 'the job is being auctioned off the "Processes Busy" queue therefore w
 are required to
        'find a match between the "Filetoopen" variable and a job name on the "
rocesses Busy"
        'queue - this matching procedure is required because between the releas
 of the bid
        'objects and assessment of the bids another job could be placed on the
Processes Busy"
        'from the main Factory Queue.
        For i = 1 To Processes_Busy.ListCount
        'we are required to find a match between the job being auctioned off th
 "Processes Busy"
        'queue and the current position of the job on the queue now so that if
e have to remove
        'it we remove the correct job
            If Trim(Filetoopen) = Trim(Processes_Busy.List(i - 1)) Then ' a mat
h has been found
                Matchposition = (i - 1)
            End If
        Next
End If

DoEvents
SkipFactStuff:

If FileLen("C:\Factory\Config\nodefile.txt") = 0 Then 'if there is nothing in t
e file then exit
 this subprocedure
    Exit Sub
End If

 Assign a free number to this file
Filenum = FreeFile

 Calculate the recordlength in Bytes
recordlength = Len(ThisRecord)

 Open the file that records the names of all the nodes in the factory for rando
 access with
'allowable functions read only
)pen "C:\Factory\Config\nodefile.txt" For Random Access Read Shared As Filenum
en = recordlength
```

```
position = 1

'Type bidEvaluation
    'Nodename As String
    'WinninigBidsize As Single
    'Winning_VMD As Integer
'End Type

Do Until EOF(Filenum)
    position = position + 1
    Get Filenum, position, ThisRecord
 '   MsgBox (Thisrecord.node).
    ReDim Preserve Thisnode(position - 1)
    Thisnode(position - 1).Nodename = Trim(ThisRecord.Node)
Loop


NumberOfNodes = position - 2

'For i = 1 To (Numberofnodes)
    'MsgBox (Thisnode(i).Nodename)
'Next

Close Filenum


'we have all the names of the nodes in memory so we may as well reset the "Bidf
mstrt"
'block at each of the Cell controllers so that when the next prompt is given t
start the
'bid formulation process it is done.
For i = 1 To NumberOfNodes
    Node = Trim(Thisnode(i).Nodename)
    'MsgBox (node)
    Field = "A_CV"
    Tag = "Bidfrmstrt"
    Text = "0"
    'resetting
    Call sendinfo(Node, Tag, Field, Text)
Next



'now we have in the computer's memory the names of all the active nodes, now a
 we have
'to do is to open up a copy of each nodes bidfile that resides on the network a
d from this
'compare the bids in each file to find which VMD on which cell wins the Auctio


Dim CurrentVMD As String
Dim Currentbid As String
Dim VMDBusystatus As String
Dim Thisnodesbid() As Bid_Evaluation_In_Process
Dim Bidrecordsize As Integer
```

```
Dim Thesmallestbid As Single
Dim ThesmallestBidposition As Integer
Dim j As Integer
Dim VMDBlocked As Integer
Dim RemovedJob As String

'Type Bid_Evaluation_In_Process
    'Nodename As String * 6
    'Bidsize As Single
    'VMDNumber As String * 12
    'Vmdbusystate As String * 10
End Type

open up a bidfile for each node to get bids
For i = 1 To NumberOfNodes

    'Define a freefile number
    Filenum = FreeFile

    'define the storage length of the variable being extracted from the file
    recordlength = 10

    'define the name of the file to be opened
    Filename = Thisnode(i).Nodename

    'MsgBox (Filename)
    Thisdirectory = Netdrive & "Factory\Bidding\Bids\"

    'open the ith nodes bidfile and extract the bids for each VMD from it
    Open Trim(Thisdirectory) & Trim(Filename) & "Bid.bid" For Input Shared As F
Ilenum Len = recordlength


    'get the number of the winning bid's VMD on the node concerned.
    Line Input #Filenum, CurrentVMD
    MsgBox (CurrentVMD)
    'get the size of the winning bid.
    Line Input #Filenum, Currentbid
'    MsgBox (Currentbid)
    'get the value of the buffer full variable
    Line Input #Filenum, VMDBusystatus
'    MsgBox (VMDBusystatus)

    ReDim Preserve Thisnodesbid(i)
    ' MsgBox (VMDBusystatus)

    Thisnodesbid(i).Nodename = Trim(Filename)
    Thisnodesbid(i).Bidsize = CDbl(Currentbid)
    Thisnodesbid(i).VMDNumber = CurrentVMD
    Thisnodesbid(i).VMDBusystate = VMDBusystatus
'    MsgBox (ThisNodesbid(i).Nodename)
'    MsgBox (CStr(ThisNodesbid(i).Bidsize))
'    MsgBox (ThisNodesbid(i).VMDNumber)

    Close Filenum
```

```
Next

DoEvents

For i = 1 To NumberOfNodes

    If i = 1 And Thisnodesbid(1).Bidsize <> 0 Then
        Thesmallestbid = Thisnodesbid(i).Bidsize
        ThesmallestBidposition = i
        GoTo Next_I
        Else
            If i = 1 And Thisnodesbid(1).Bidsize = 0 Then
                Thesmallestbid = 0
                ThesmallestBidposition = 0
                GoTo Next_I
            End If
    End If

    If i > 1 And Thisnodesbid(i).Bidsize = 0 Then
        GoTo Next_I
    End If

    If i > 1 And Thisnodesbid(i).Bidsize > 0 Then
        If Thisnodesbid(i).Bidsize < Thisnodesbid(ThesmallestBidposition).Bidsi
e Or Thisnodesbid(ThesmallestBidposition).Bidsize = 0 Then
                'MsgBox CStr(Thisnodesbid(i).Bidsize)
                Thesmallestbid = Thisnodesbid(i).Bidsize
               ' MsgBox CStr(Thisnodesbid(i).Bidsize)
                Thesmallestbidposition = i
        End If
    End If
Next_I:
Next

'check to see if any vmd in the system is eligible to get the part but is busy
For j = 1 To NumberOfNodes
    If Val(Thisnodesbid(j).VMDBusystate) = 1 Then
        VMDBlocked = 1
    End If
Next

DoEvents

If Thesmallestbid = 0# And VMDBlocked <> 1 And FactoryIsAuctioning = 1 Then 'ad
 the job to the rejected jobs listbox
    'first get the name of the item to be removed and store it in a variable


    RemovedJob = QueueList.List(Matchposition)

    'now remove the job that is being auctioned
    QueueList.RemoveItem (Matchposition)

    'update event listbox
    Res_Form.Res_Job.AddItem Trim(RemovedJob)
    Res_Form.Res_time.AddItem Time$
```

```
    Res_Form.Res_Event.AddItem "Job Rejected By System"
    Res_Form.Res_Queue.AddItem "Factory Queue"


    'Define Factory Job Queueing variables
    Dim Currqueue As Factqueue
    Dim Thisqueue()  As Factqueue
    Dim FactQnum As Integer
    Dim qFilename As String * 40
    Dim Recordleng As Integer
    Dim Thislengths As Integer
    Dim Listlength As Integer

    'Now kill the old queuefile and resave the contents of the listbox to the ne
w file
    'version
    Kill "C:\Factory\Queue\Queuejob.que"

    'get a new number for the file
    FactQnum = FreeFile

    'Get the array record length
    Recordleng = Len(Currqueue)


    'open the new version of the file
     Open "C:\Factory\Queue\Queuejob.que" For Random Access Read Write Shared As
 FactQnum Len = Recordleng

    'Redimension the array to take all the list's contents
    ReDim Thisqueue(QueueList.ListCount)

    'Write the array's length to memory
    Thisqueue(1).Filelength = QueueList.ListCount

    'write the new contents of the list to the array
    For i = 1 To QueueList.ListCount
        Thisqueue(i).Jobname = Trim(QueueList.List(i - 1))
    Next

    'Put the contents of the array to file
    For i = 1 To QueueList.ListCount
        Put FactQnum, i, Thisqueue(i)
    Next

    'Now close the file
     Close FactQnum

    'this is all very fine but we must also add the job to the "Rejected Jobs" l
istbox
    Jobrejection.AddItem RemovedJob

    'kill the old version of the file that we are required to save the contents
 f this listbox
    'to.
    Kill "C:\Factory\Queue\Rejectjb.que"
```

```
    'now save this all to file
    Filenum = FreeFile

    Recordleng = 20 'recordlength = 20 bytes

    'open the file for sequential output
    Open "C:\Factory\Queue\Rejectjb.que" For Output As Filenum Len = Recordlen

    Listlength = Jobrejection.ListCount 'get the length of the list

    Print #Filenum, Listlength 'Print the listlength to file

    For i = 1 To Listlength
        Print #Filenum, Jobrejection.List(i - 1) 'write everything in the list
o file
    Next

    Close Filenum 'close the file


    Node = "fact"
    Field = "A_CV"
    Tag = "Bidoperate"
    Text = "0"
    'release the token and then exit the procedure
    Call sendinfo(Node, Tag, Field, Text)
    Exit Sub 'exit the procedure

    Else
        If Thesmallestbid = 0# And VMDBlocked = 1 And FactoryIsAuctioning = 1 T
en 'there is a process
        'in the Factory That can carry out the required task  but it's buffer i
 full - first we are required
        'to see if the job is urgent - if it is it is put on the "Processes Bus
" list(If it isn't put
        'it at the bottom of the current queue list)and whenever the 'Factory C
ntroller gets the bid
        'token it must attempt to hold an auction for each job in the '"Process
s Busy" - it does this
        'repeatedly until it is able to auction a job off this "processes 'busy
" listbox, if it cant
        'because all processes are busy(ie:Their buffers are full) then the Fac
ory Controller attempts
        'to hold an auction for the first job on it's normal queue - if it cant
because the queues
        'for that job are busy then it releases the Bidtoken and puts that job
n the "Processses Busy"
        'list, otherwise it auctions the job as usual.

        'open the job's jobfile and see whether or not it is urgent

        '*****************************************************************
****
        'Declare variables
        Dim Savetasks() As JobRecord
```

```
        Dim Savetask As JobRecord
        Dim Directorynam As String * 24
        Dim Thisfilename As String * 30
        Dim JobIsUrgent As Integer

        'declare bid file variables.
        Dim ThisbidFile As Bid_data_To_Be_sent
        Dim Filenumb As Integer


        If Busy_Jobs_Auction = 1 Then ' a job is being auctioned off the "Jobs B
usy queue" and
            'although there is a process in the factory that can do the job it
s STILL busy -
            'therefore we are required to try and auction the next job in the "
obs Busy Queue"
            'if there is a job there

'            MsgBox ("Job is still blocked")
            'Reset the 'bidtoken' textbox so it can respond to the "change event
"
            Bidtoken.Text = 0

            'now put it back to the value of one to try and auction the next job
 in the "Jobs Busy
            'Queue
            Tryagain = 1
            Bidtoken.Text = "1"


            'update event listbox
            Res_Form.Res_Job.AddItem Trim(RemovedJob)
            Res_Form.Res_time.AddItem Time$
            Res_Form.Res_Event.AddItem "Recipient Buffer Full, Factory Busy Q Au
tion"
            Res_Form.Res_Queue.AddItem "Processes Busy"


            Exit Sub
        End If

        'else it is actually a job off the factory controller queue that has bee
n auctioned
        'and its "winning processes" queue is full so we are required to transfe
 it to
        'the processes busy queue.
'        MsgBox ("Job on Factory Queue is Blocked")
        RemovedJob = QueueList.List(Matchposition)
'        MsgBox (RemovedJob)
        Filenum = FreeFile 'define a filenumber that is not being used.
        recordlength = Len(Savetask) 'define arrays storage size required.

'        MsgBox (CStr(Matchposition))
        'the name of the file to open is the name corresponding with the matchpo
sition index in
```

```
        'the list
        Filetoopen = RemovedJob
        Directorynam = Trim(Jobfilepath)
        'MsgBox (Directorynam)

        Open Trim(Directorynam) & Trim(Filetoopen) & ".iff" For Random Access R
ad Write As Filenum Len = recordlength

        ReDim Savetasks(1) 'Redimension array to hold first record only

        'Get the urgency variable of the job
        Get Filenum, 1, Savetasks(1)

        Close Filenum

            If Savetasks(1).Force = 1 Then  'then the job is urgent
                JobIsUrgent = 1
                QueueList.RemoveItem (Matchposition) 'remove the item from the p
osition in the lis
                'corresponding with the matched position.

                'add the item at the Top "processes busy" listbox
                Processes_Busy.AddItem Trim(RemovedJob), 0
                'now save this to file

                Res_Form.Res_Job.AddItem Trim(RemovedJob)
                Res_Form.Res_time.AddItem Time$
                Res_Form.Res_Event.AddItem "RBF, Urgent Job, Add - top of Proces
ses busy queue"
                Res_Form.Res_Queue.AddItem "Factory Queue"



                Filenum = FreeFile 'define a freefile

                Recordleng = 20 'get enough room to occupy file elements

                'kill the old version of the file to avoid any errors
                Kill "C:\Factory\Queue\ProcJbBs.que"

                'open the file containing the jobs for which the processes are t
oo busy to carry out.
                'it is opened for sequential output
                Open "C:\Factory\Queue\ProcJbBs.que" For Output As Filenum Len =
 Recordleng

                Listlength = Processes_Busy.ListCount 'get the number of items r
equired to be added
                'to the file.

                Print #Filenum, Listlength 'write a variable to a file saying h
w long the list of items
                'is    print

                For i = 1 To Listlength 'write each of the items in the list to
the file.
```

```
                Print #Filenum, Trim(Processes_Busy.List(i - 1))
            Next

            Close Filenum 'close the file

            'Now kill the old queuefile and resave the contents of the list
ox to the new file
            'version
            Kill "C:\Factory\Queue\Queuejob.que"

            'get a new number for the file
            FactQnum = FreeFile

            'Get the array record length
            Recordleng = Len(Currqueue)


            'open the new version of the file
            Open "C:\Factory\Queue\Queuejob.que" For Random Access Read Wri
 Shared As FactQnum Len = Recordleng

            'Redimension the array to take all the list's contents
            ReDim Thisqueue(QueueList.ListCount)

            'Write the array's length to memory
            Thisqueue(1).Filelength = QueueList.ListCount

            'write the contents of the list to the array
            For i = 1 To QueueList.ListCount
                Thisqueue(i).Jobname = Trim(QueueList.List(i - 1))
            Next

            'Put the contents of the array to file
            For i = 1 To QueueList.ListCount
                Put FactQnum, i, Thisqueue(i)
            Next

            'Now close the file
            Close FactQnum

            Else 'if the job is not urgent then put it at the bottom of the
rocesses Busy queue

                RemovedJob = QueueList.List(Matchposition) 'store the name
 the job in a temporary variable

                'now remove the item from the list and re-add it to second
ter the matchposition
                QueueList.RemoveItem (Matchposition)

                'add it to the bottom of the list
                ''QueueList.AddItem Trim(RemovedJob), Matchposition + 1

                'now resave the contents of the list to file

                'update event queue
```

```
                    Res_Form.Res_Job.AddItem Trim(RemovedJob)
                    Res_Form.Res_time.AddItem Time$
                    Res_Form.Res_Event.AddItem "RBF, Casual Job, Add - Bottom o
 Processes busy queue"
                    Res_Form.Res_Queue.AddItem "Factory Queue"

                    'Now kill the old queuefile and resave the contents of the
istbox to the new file
                    'version

                    'add the item at the Top "processes busy" listbox
                    Processes_Busy.AddItem Trim(RemovedJob)
                    'now save this to file


                    Filenum = FreeFile 'define a freefile

                    Recordleng = 20 'get enough room to occupy file elements

                    'kill the old version of the file to avoid any errors
                    Kill "C:\Factory\Queue\ProcJbBs.que"

                    'open the file containing the jobs for which the processes
re too busy to carry out.
                    'it is opened for sequential output
                    Open "C:\Factory\Queue\ProcJbBs.que" For Output As Filenum
en = Recordleng

                    Listlength = Processes_Busy.ListCount 'get the number of it
ms required to be added
                    'to the file.

                    Print #Filenum, Listlength 'write a variable to a file sayi
g how long the list of items
                    'is    print

                    For i = 1 To Listlength 'write each of the items in the lis
 to the file.
                        Print #Filenum, Trim(Processes_Busy.List(i - 1))
                    Next

                    Close Filenum 'close the file

                    'Now kill the old queuefile and resave the contents of the
istbox to the new file
                    'version
                    Kill "C:\Factory\Queue\Queuejob.que"

                    'get a new number for the file
                    FactQnum = FreeFile

                    'Get the array record length
                    Recordleng = Len(Currqueue)


                    'open the new version of the file
```

```
                    Open "C:\Factory\Queue\Queuejob.que" For Random Access Read
Write Shared As FactQnum Len = Recordleng

                    'Redimension the array to take all the list's contents
                    ReDim Thisqueue(QueueList.ListCount)

                    'Write the array's length to memory
                    Thisqueue(1).Filelength = QueueList.ListCount

                    'write the contents of the list to the array
                    For i = 1 To QueueList.ListCount
                        Thisqueue(i).Jobname = Trim(QueueList.List(i - 1))
                    Next

                    'Put the contents of the array to file
                    For i = 1 To QueueList.ListCount
                        Put FactQnum, i, Thisqueue(i)
                    Next

                    'Now close the file
                    Close FactQnum

        End If
        'if the program gets to this point it means that the Factory Controller
s incoming queue
        'has held an auction and that all possible processes to do the job are b
usy and that the
        'job has been put on either the bottom of the processes busy queue if it
is
        'not urgent or has been put on the top of the "Processes Busy" queue if
t is.
        'Next time around therefore the Bidtoken change event will see that ther
 is a job on the
        '"Processes Busy" queue and will attempt to auction it - in the mean tim
e we are required
        'to release the bidtoken to allow other nodes the opportunity to hold an
 auction.

         'therefore set the bidoperate token to "0"
        Node = "fact"
        Field = "A_CV"
        Tag = "Bidoperate"
        Text = "0"
        'release the token and then exit the procedure
        Call sendinfo(Node, Tag, Field, Text)
        Exit Sub
        End If
End If


If Thesmallestbid = 0# And VMDblocked <> 1 And Factoryisauctioning = 0 Then 'th
en a cell controller
'has instigated an auction via the factory controller but cannot give the job to
 the "winning"
 process because its queue is full.Therefore we are required to put that job on
the "Busy processes"
```

```
'job queue(bottom) if it is urgent, otherwise


If Thesmallestbid > 0# And FactoryIsAuctioning = 1 Then  'a valid bid has been
odged so we
'are required to award the job to the relevant cell's VMD.

DoEvents
    If Busy_Jobs_Auction = 1 Then ' a successful auction was carried out but t
 job being
    'auctioned was not the job off the factory queue but a job off the "Process
s Busy" queue

    'therefore we are required to remove the job off the queue at the NumberOfB
syJobsAuctioned
    'position in the "Processes Busy" listbox and then save the changes to fil

        Processes_Busy.RemoveItem (Matchposition)


        'now put some code in here to reflect the fact that a job has been auct
oned off the factory
        'controller "Processes Busy" Queue

        Res_Form.Res_Job.AddItem Trim(Filetoopen)
        Res_Form.Res_time.AddItem Time$
        Res_Form.Res_Event.AddItem "Auctioned Off"
        Res_Form.Res_Queue.AddItem "Processes Busy"

        'now save the changes to file
        Filenum = FreeFile 'define a freefile

        Recordleng = 20 'get enough room to occupy file elements

        'kill the old version of the file to avoid any errors
        Kill "C:\Factory\Queue\ProcJbBs.que"

        'open the file containing the jobs for which the processes are too busy
to carry out.
        'it is opened for sequential output
         Open "C:\Factory\Queue\ProcJbBs.que" For Output As Filenum Len = Recor
leng

        ' Listlength = Processes_Busy.ListCount 'get the number of items require
 to be added
         'to the file.

         Print #Filenum, Listlength 'write a variable to a file saying how long
the list of items
         'is

         For i = 1 To Listlength 'write each of the items in the list to the fil
e.
              Print #Filenum, Processes_Busy.List(i - 1)
         Next
```

```
        Close Filenum 'close the file
    End If


    'If Thesmallestbid = 0# And VMDblocked <> 1 And Factoryisauctioning = 2 The
    'a cell controller is holding an auction but this job cannot be distributed
because all processes
    'queues are full

    'Open Netdrive & "Cellcont\Auction\Auctdata.txt" For Input As filenum Len =
50

    'get the name of the job and store that name in the "Jobname" variable
    'Line Input #filenum, Jobname

    'get the name of the node and store it in the nodename variable
    'Line Input #filenum, Nodename
    'MsgBox (Nodename)
    'Close filenum


    'MsgBox ("The smallest bid is " & CStr(ThisNodesbid(TheSmallestBidPosition)
Bidsize))
    'MsgBox ("The winning node is " & CStr(ThisNodesbid(TheSmallestBidPosition)
Nodename))
    'MsgBox ("The winning VMD is " & CStr(ThisNodesbid(TheSmallestBidPosition).
IDNumber))


    WinningNode = Trim(Thisnodesbid(ThesmallestBidposition).Nodename)
   'MsgBox (WinningNode)


    Thisblock = "V" & Trim(Thisnodesbid(ThesmallestBidposition).VMDNumber) & "CT
CTYPE"

    'Retrieve Post Processor type for winning process
    Processortype = GetprocessorsName(Trim(WinningNode), Thisblock)

    Call RunThePostProcessor(Trim(PostProcessorType))



    'and now reset the "Bidwin" block
    Node = Trim(WinningNode)
    'MsgBox (WinningNode)
    Field = "A_CV"
    Tag = "Bidwin"
    Texts = "0"

    Call sendinfo(Node, Tag, Field, Texts)
```

```
    'set texts equal to the number of the winning VMD for this node
    Texts = Trim(Thisnodesbid(ThesmallestBidposition).VMDNumber)

    'MsgBox (Texts)
    'send the award message to the correct node indicating the number of the wi
ning VMD.
    Call sendinfo(Node, Tag, Field, Texts)


    'remove the job from the top of the incoming queue of the factory controlle
 but only if it
    'is being auctioned from there
    If Busy_Jobs_Auction <> 1 Then
        QueueList.RemoveItem 0
        'Now kill the old queuefile and resave the contents of the listbox to t
e new file
        'version
        DoEvents
        Kill "C:\Factory\Queue\Queuejob.que"


        'define filename holding the queueing data
        qFilename = "C:\Factory\Queue\Queuejob.que"

        'Free up a number for the file
        FactQnum = FreeFile

        'Get the array record length
        Recordleng = Len(Currqueue)

        'open the factroy queue file
        Open "C:\Factory\Queue\Queuejob.que" For Random Access Read Write Share
 As FactQnum Len = Recordleng

        'Get the file's length
        Thislengths = QueueList.ListCount

        'Now redimension the array to take all the records

        ReDim Thisqueue(Thislengths)


        'Type Factqueue
        'Jobname As String * 13
        'Filelength As Integer
        'End Type

        Thisqueue(1).Filelength = Thislengths

        For i = 1 To Thislengths
            Thisqueue(i).Jobname = QueueList.List(i - 1)
            Put FactQnum, i, Thisqueue(i)
        Next

        Close FactQnum
```

```
        DoEvents
        'now put some code in here to reflect the fact that a job has been auct
oned off the factory
        'controller
        Res_Form.Res_Job.AddItem Trim(Filetoopen)
        Res_Form.Res_time.AddItem Time$
        Res_Form.Res_Event.AddItem "Auctioned Off"
        Res_Form.Res_Queue.AddItem "Factory Queue"


    End If

'
 we are also required to release the bidtoken to allow other nodes the opportun
y to conduct
'an auction.
 therefore set the bidoperate token to "0"

'***WE CANNOT RELEASE THE BIDTOKEN FROM HERE BECAUSE IT SHOULDNT BE RELEASED UN
L THE WINNING VMD
 HAS ACKNOWLEDGED RECEPTION OF THE BIDDING OBJECTS
'node = "fact"
'Field = "A_CV"
 Tag = "Bidoperate"
'Text = "0"
'release the token and then exit the procedure

'Call sendinfo(node, Tag, Field, Text)

xit Sub

nd If


f Thesmallestbid = 0# And VMDBlocked <> 1 And FactoryIsAuctioning = 0 Then


    'A cell controller in the factory is attempting an auction however there is
o cell in the
    'factory capable of winning the auction as it is unable to meet the toolage,
 apacity and
    'machine requirements.

    'the name of the rejected job is stored in the "Jobname" variable so add it
 the "Jobs
    'Rejected" listbox and then save the list's contents to file
    Jobrejection.AddItem Trim(Jobname)


    Res_Form.Res_Job.AddItem Trim(Jobname)
    Res_Form.Res_time.AddItem Time$
    Res_Form.Res_Event.AddItem "Job Rejected - Cell " & Trim(Nodename) & " Aucti
ning"
    Res_Form.Res_Queue.AddItem "Cell Outgoing Queue"
```

```
    'kill the old version of the file that we are required to save the contents
of this listbox
    'to.
    Kill "C:\Factory\Queue\Rejectjb.que"

    'now save everything that is in the "Rejected Jobs" list to file
    Filenum = FreeFile

    Recordleng = 20 'recordlength = 20 bytes

    'open the file for sequential output
    Open "C:\Factory\Queue\Rejectjb.que" For Output As Filenum Len = Recordlenc

    Listlength = Jobrejection.ListCount 'get the length of the list

    Print #Filenum, Listlength 'write the listlenght to file

    For i = 1 To Listlength
        Print #Filenum, Jobrejection.List(i - 1) 'write everything in the list
o file
    Next

    Close Filenum 'close the file

    'this auctioning process is being carried out by a cell controller and not
he factory
    'controller therefore we must tell the Cell Controller that the auction is c
omplete and
    'that it can get it's next VMD to hold an auction.We do this by getting the
current number
    'of the VMD currently holding the auction, incrementing it and sending the
ew value to the
    'node.

    'Dim Nodename As String * 8
    'Dim VMDNumber As String * 10
    'Dim Jobname As String * 10

    'convert the current value of the VMD into an integer so that it can be incr
emented
    ThisVMDNumber = Val(VMDNumber)

    'MsgBox (thisstring)
    'now increment the variable
    ThisVMDNumber = ThisVMDNumber + 1

    'MsgBox (node)
    'and send it back to the node that conducted the auction
    Texts = CStr(ThisVMDNumber)
    Tags = "VMDCURAUCT"
    Node = Trim(Nodename)
    Field = "A_CV"

    'MsgBox (Texts)
    Call sendinfo(Node, Tags, Field, Texts) 'What this will do is give the node
```

```
that has just
     'tried to auction a job another chance to have a go with another VMD.
    Exit Sub

    Else
    If Thesmallestbid = 0# And VMDBlocked = 1 And FactoryIsAuctioning = 0 Then
'       MsgBox ("A cell is Auctioning and the recipient cell is blocked")
        'a cell controller is holding an auction and although there is a proces
that is able
        'to do the job that process is busy - so we either need to put that job
t the bottom
        'of the cell controller queue that it came from if it is not urgent OR p
ut it on the
        '"processes Busy" queue in the factory controller if it is urgent.

        'therefore the first thing we must find out is whether it is urgent or n
t.
         'name of the job is stored in the "Jobname" variable which has been pas
sed to it in the
        '"Cell Auctioning" textbox.

        Filenum = FreeFile 'define a filenumber that is not being used.
        recordlength = Len(Savetask) 'define arrays storage size required.

        'the name of the file to open is the name stored in the "Jobname" Variab
e

        Filetoopen = Trim(Jobname)

        Directorynam = Trim(Jobfilepath)
        'MsgBox (Directorynam)

        Open Trim(Directorynam) & Trim(Filetoopen) & ".iff" For Random Access Re
 Write As Filenum Len = recordlength

        ReDim Savetasks(1) 'Redimension array to hold first record only

        'Get the urgency variable of the job
        Get Filenum, 1, Savetasks(1)

        Close Filenum

        If Savetasks(1).Force = 1 Then 'the job is urgent so put the name on the
actory
        'controller's "Processes Busy" queue and then save the queue to file

                Processes_Busy.AddItem Trim(Jobname) 'add the jobname at the bot
om of the queue

                Res_Form.Res_Job.AddItem Trim(Jobname)
                Res_Form.Res_time.AddItem Time$
                Res_Form.Res_Event.AddItem "RBF " & Trim(Nodename) & " Auctionin
 & "Add Factory Controller"
                Res_Form.Res_Queue.AddItem "Cell Queue"

                'now save this to file
                Filenum = FreeFile 'define a freefile
```

```
                Recordleng = 20 'get enough room to occupy file elements

                'kill the old version of the file to avoid any errors
                Kill "C:\Factory\Queue\ProcJbBs.que"

                'open the file containing the jobs for which the processes are
oo busy to carry out.
                'it is opened for sequential output
                Open "C:\Factory\Queue\ProcJbBs.que" For Output As Filenum Len
 Recordleng

                Listlength = Processes_Busy.ListCount 'get the number of items
equired to be added
                'to the file.

                Print #Filenum, Listlength 'write a variable to a file saying h
w long the list of items
                'is

                For i = 1 To Listlength 'write each of the items in the list to
the file.
                    Print #Filenum, Processes_Busy.List(i - 1)
                Next

                Close Filenum 'close the file

                'this is ok but we must also allow the next VMD in the Cell the
chance to hold an
                'auction

                'this auctioning process is being carried out by a cell control
er and not the factory
                'controller therefore we must tell the Cell Controller that the
auction is complete and
                'that it can get it's next VMD to hold an auction.We do this by
getting the current number
                'of the VMD currently holding the auction, incrementing it and
ending the new value to the
                'node.

                'convert the current value of the VMD into an integer so that i
 can be incremented
                ThisVMDNumber = Val(VMDNumber)

                'now increment the variable
                ThisVMDNumber = ThisVMDNumber + 1

                'MsgBox (node)
                'and send it back to the node that conducted the auction
                Texts = CStr(ThisVMDNumber)
                Tags = "VMDCURAUCT"
                Node = Trim(Nodename)
                Field = "A_CV"

                'MsgBox (Texts)
```

```
                Call sendinfo(Node, Tags, Field, Texts) 'What this will do is g
e the node that has just
                'tried to auction a job another chance to have a go with anothe
VMD.
                Exit Sub

                Else    'the job is not urgent therefore send the name of the r
ected job back to
                        'the cell that it came from and tell the cell to put it
ack on the VMD's
                        'outgoing queue that it came from but put it on the bot
m of the queue and
                        'not the top.

                        'Dim Nodename As String * 8
                        'Dim VMDNumber As String * 10
                        'Dim Jobname As String * 10
    '                    MsgBox ("A cell is Auctioning and the recipient cell is
blocked and the job is not urgent - I am putting back on the outgoing queue")
    '                    MsgBox ("the node to give the job back to is " & Trim(N
dename))
    '                    MsgBox ("VMD number is " & Trim(VMDNumber))

                        Res_Form.Res_Job.AddItem Trim(Jobname)
                        Res_Form.Res_time.AddItem Time$
                        Res_Form.Res_Event.AddItem "RBF " & Trim(Nodename) & " A
ctioning " & "Add Bottom of Cell Queue"
                        Res_Form.Res_Queue.AddItem "Cell Queue"


                        CellToGiveBackTo = Trim(Nodename)
                       ' MsgBox (CellToGiveBackTo)
                        Vmdtogivebackto = Trim(VMDNumber)
                       ' MsgBox (Vmdtogivebackto)

                        Filenum = FreeFile 'denote a freefile

                        'open a network file that this information can be writte
to and put the
                        'information in it and then prompt the relevant cell con
troller to
                        'requeue the job
                        Open Netdrive & "Cellcont\Auction\Rejectjb.job" For Outp
 As Filenum Len = 20

                        Print #Filenum, CellToGiveBackTo 'write to the file the
me of the cell
                        'that the job must be returned to

                        Print #Filenum, Jobname 'write the name of the job that
st be retured to
                        'the file

                        Close Filenum 'now close the file

                        'now prompt the node to get it to put the job back on th
```

```
e list

                        Texts = Trim(Vmdtogivebackto)
                        Tags = "REQUEUEJOB"
                        Node = Trim(Nodename)
                        Field = "A_CV"

                        'MsgBox (Texts)
                        Call sendinfo(Node, Tags, Field, Texts) 'This will prom
t the correct
                        'vmd to requeue the job


                        'this is ok but we must also allow the next VMD in the
ell the chance to hold an
                        'auction

                        'this auctioning process is being carried out by a cell
controller and not the factory
                        'controller therefore we must tell the Cell Controller
hat the auction is complete and
                        'that it can get it's next VMD to hold an auction.We do
this by getting the current number
                        'of the VMD currently holding the auction, incrementing
it and sending the new value to the
                        'node.

                        'convert the current value of the VMD into an integer s
 that it can be incremented
                        ThisVMDNumber = Val(VMDNumber)

                        'now increment the variable
                        ThisVMDNumber = ThisVMDNumber + 1

                        'MsgBox (node)
                        'and send it back to the node that conducted the auctio
                        Texts = CStr(ThisVMDNumber)
                        Tags = "VMDCURAUCT"
                        Node = Trim(Nodename)
                        Field = "A_CV"

                        'MsgBox (Texts)
                        Call sendinfo(Node, Tags, Field, Texts) 'What this will
do is give the node that has just
                        '   tried to auction a job another chance to have a go v
ith another VMD.
                        Exit Sub
        End If
    End If
End If


'this is fine but we must also address the situation whereby a cell in the facto
ry is conducting
'an auction and there is a winning cell in the Factory that can recieve the task
If Thesmallestbid > 0# And FactoryIsAuctioning = 0 Then
```

```
    'a cell in the factory is conducting an auction and there is a cell in th
actory that can
    'do the job and is not blocked.
    DoEvents
    WinningNode = Trim(Thisnodesbid(ThesmallestBidposition).Nodename)


    Thisblock = "V" & Trim(Thisnodesbid(ThesmallestBidposition).VMDNumber) & "(
:TYPE"

    'Retrieve Post Processor type for winning process
    Processortype = GetprocessorsName(Trim(WinningNode), Thisblock)

    'run the postprocessor according to the type of postprocessor being used by
he VMD
    Call RunThePostProcessor(Trim(PostProcessorType))


    'and now reset the "Bidwin" block of the winning node so that it can be prom
ted
    Node = Trim(WinningNode)
     MsgBox (CStr(winningNode))
    Field = "A_CV"
    Tag = "Bidwin"
    Texts = "0"

    Call sendinfo(Node, Tag, Field, Texts) 'resetting

    'set texts equal to the number of the winning VMD for this node
    Texts = Trim(Thisnodesbid(ThesmallestBidposition).VMDNumber)

     MsgBox (CStr(Texts))
    'send the award message to the correct node indicating the number of the win
g VMD.
    Call sendinfo(Node, Tag, Field, Texts)

    '******This is new be carefull*****************************

    'this auctioning process is being carried out by a cell controller and not t
 factory
    'controller therefore we must tell the Cell Controller that the auction is c
ete and
    'that it can get it's next VMD to hold an auction.We do this by getting the
nt number
     of the VMD currently holding the auction, incrementing it and sending the n
    alue to the
     node.

    'Dim Nodename As String * 8
     Dim VMDNumber As String * 10
     Dim Jobname As String * 10


      **WE ARE GOING TO CANCEL THIS OUT SO THAT THIS CAN OCCUR WHEN THE VMD HAS
     RMED THAT
     HE AWARD HAS OCCURRED
```

```
    'convert the current value of the VMD into an integer so that it can be inc
emented
    ''ThisVMDNumber = Val(VMDNumber)

    'MsgBox (thisstring)
    'now increment the variable
    ''ThisVMDNumber = ThisVMDNumber + 1

    'MsgBox (node)
    'and send it back to the node that conducted the auction
    ''texts = CStr(ThisVMDNumber)
   ''  Tags = "VMDCURAUCT"
   '' node = Trim(Nodename)
   '' Field = "A_CV"

    'MsgBox (Texts)
   '' Call sendinfo(node, Tags, Field, texts) 'What this will do is give the noc
e that has just
     'tried to auction a job another chance to have a go with another VMD.

End If



End Sub

Sub BidToken_Change ()

'Declare variables
Dim recordlength As Integer
Dim Filenum As Integer
Dim Savetasks() As JobRecord
Dim Savetask As JobRecord
Dim Directorynam As String * 24
Dim i As Integer
Dim Thisfilename As String * 30
Dim IsurgenT As Integer

'declare bid file variables.
Dim ThisbidFile As Bid_data_To_Be_sent
Dim Filenumb As Integer

'This piece of code gets the bidding objects of the task currently required to b
e done and puts
 them on the network in the form of a file. It also writes the bidding objkects
to a file on
 the network and prompts the cell controllers to get the information and to form
ulate a bid.


On Error Resume Next



Select Case Val(Bidtoken.Text)
```

```
when this turns to a "1" we want the item at the top of the list to be Auction
d.

ase Is = 1#


f Tryagain = 1 Then 'if we are trying to auction ANOTHER job off the "Busy_Job
" queue
   Firstpass = 0
lse
       Firstpass = 1
nd If

The first thing that we are required to do is to see if there are any jobs in
he
"Processes_Busy" queue
f Len(Trim(Processes_Busy.List(0))) > 0 Then

   Tool_Avle.Tool_Timer.Enabled = False
   Tool_Avle.Bid_eval.Enabled = False

   'there is something in the Processe's busy list
   'send a message to the Factory Controller FixDmacs "BidOperate" block telli
g it to stop rotating
   'the token
   If Firstpass = 1 Then
   'reset  NumberOfBusyJobsAuctioned variable
       NumberOfBusyJobsAuctioned = 0
   End If
   'otherwise increment the counter
   NumberOfBusyJobsAuctioned = NumberOfBusyJobsAuctioned + 1

   If NumberOfBusyJobsAuctioned > Processes_Busy.ListCount Then 'then we have
ried to auction
   'any one of the jobs off the jobs busy list and yet we have unable to as th
ir process's
   'ingoing queues are full therefore we have to give the factory queue a go.
       Busy_Jobs_Auction = 0
       NumberOfBusyJobsAuctioned = 0
       Tryagain = 0
       GoTo TryHere
   End If

   'otherwise we can have a go at auctioning the next job off the "Processes_B
sy" list

   Busy_Jobs_Auction = 1

   'secure the bid token
   Node = "Fact"
   Tags = "Bidoperate"
   Field = "A_CV"
   Texts = "1"

   Call sendinfo(Node, Tags, Field, Texts) 'STOP THE BIDTOKEN FROM ROTATING
```

```
    'tell the system that the factory is conducting an auction

    FactoryIsAuctioning = 1

    Filenum = FreeFile 'deffine a filenumber that is not being used.
    recordlength = Len(Savetask) 'define arrays storage size required.

    'the name of the file to open is the value of the number of jobs already att
empted an auction
    'on minus one
    Filetoopen = Processes_Busy.List(NumberOfBusyJobsAuctioned - 1)
    'MsgBox (Filetoopen)
    Directorynam = Trim(Jobfilepath)
    'MsgBox (Directorynam)
'    MsgBox ("Filename is " & Trim(Directorynam) & Trim(Filetoopen) & ".iff")

    'open the jobfile for the job being considered in the list
    Open Trim(Directorynam) & Trim(Filetoopen) & ".iff" For Random Access Read '
rite As Filenum Len = recordlength

    ReDim Savetasks(1) 'Redimension array to hold first record only

    'Get the status of the current job
    Get Filenum, 1, Savetasks(1)

    IsurgenT = Savetasks(1).Force

    'MsgBox CStr(Savetasks(1).Status)
    ReDim Preserve Savetasks(Savetasks(1).Status) 'redimension array to hold all
 job data up to
    'the current task

    'now get all the information required about the current task to be done.

    Get Filenum, Savetasks(1).Status, Savetasks(Savetasks(1).Status) 'get the cu
rrent task data

    'Text1.Text = Savetasks(Savetasks(1).Status).Jobname
    'Text1.Text = Savetasks(Savetasks(1).Status).Infofilename
    'Text1.Text = Savetasks(Savetasks(1).Status).FinalTask
    'Text1.Text = Savetasks(Savetasks(1).Status).Tasknumber
    'Text1.Text = Savetasks(Savetasks(1).Status).machinetype
    'Text1.Text = Savetasks(Savetasks(1).Status).Taskname
    'Text1.Text = Savetasks(Savetasks(1).Status).EstCycletime
    'Text1.Text = Savetasks(Savetasks(1).Status).setuptime
    'Text1.Text = Savetasks(Savetasks(1).Status).Estloadunloadtime
    'Text1.Text = Savetasks(Savetasks(1).Status).NooFCycles
    'Text1.Text = Savetasks(Savetasks(1).Status).CNCfilename
    'Text1.Text = Savetasks(Savetasks(1).Status).Geometricfilename
    'Text1.Text = Savetasks(Savetasks(1).Status).Tool1required
    'Text1.Text = Savetasks(Savetasks(1).Status).Tool2required
    'Text1.Text = Savetasks(Savetasks(1).Status).Tool3required
    'Text1.Text = Savetasks(Savetasks(1).Status).Tool4required
    'T'ext1.Text = Savetasks(Savetasks(1).Status).Fixturesfilename
    'Text1.Text = Savetasks(Savetasks(1).Status).Status
    'Text1.Text = Savetasks(Savetasks(1).Status).Duedate
```

```
    'Text1.Text = Savetasks(Savetasks(1).Status).Force
    'Text1.Text = Savetasks(Savetasks(1).Status).SetUpInstructionsFilename

    'Type Bid_data_To_Be_sent
    '   Jobname As String * 10
    'Machinetype As Single
    'EstCycletime As Integer
    'Setuptime As Integer
    'LoadUnloadtime As Integer
    'NoofCycles As Integer
    'Tool1required As Single
    'Tool2required As Single
    'Tool3required As Single
    'Tool4required As Single
    'Activate As Integer
    'force as integer
    'AuctioningCell As String * 8
    'End Type


    'First kill the oldfile that is on the network.
    Kill Netdrive & "Factory\Bidding\Bidfile.bdf"

    'denote a filenumber for the new file.
    Filenumb = FreeFile
    recordlength = Len(ThisbidFile)

    'Now we must open a new copy of the file and write the bidding objects to it

    Thisfilename = Netdrive & "Factory\Bidding\Bidfile.bdf"
    'MsgBox (Thisfilename)

    Open Netdrive & "Factory\Bidding\Bidfile.bdf" For Random Access Write As Fil
enumb Len = recordlength


    'now write the current task bidding data to the bidfile
    ThisBidding(1).Jobname = Savetasks(Savetasks(1).Status).Jobname
    ThisBidding(1).Machinetype = Savetasks(Savetasks(1).Status).Machinetype
    ThisBidding(1).EstCycletime = Savetasks(Savetasks(1).Status).EstCycletime
    ThisBidding(1).setuptime = Savetasks(Savetasks(1).Status).setuptime
    ThisBidding(1).Loadunloadtime = Savetasks(Savetasks(1).Status).Estloadunloac
Time
    ThisBidding(1).NooFCycles = Savetasks(Savetasks(1).Status).NooFCycles
    ThisBidding(1).Tool1required = Savetasks(Savetasks(1).Status).Tool1required
    'MsgBox (CStr(Savetasks(Savetasks(1).Status).Tool1required))
    'Thisbidding(1).Tool2required = Savetasks(Savetasks(1).Status).Tool2requirec
    'Thisbidding(1).Tool3required = Savetasks(Savetasks(1).Status).Tool3requirec
    'Thisbidding(1).Tool4required = Savetasks(Savetasks(1).Status).Tool4requirec
    ThisBidding(1).Activate = 1
    ThisBidding(1).Force = IsurgenT
    ThisBidding(1).AuctioningCell = "FACT"

    Put Filenumb, 1, ThisBidding(1) 'put the bidding objects into the bidfile
    Close Filenumb
```

```
    Close Filenum

    'However we must also prompt the cells to begin the Bid formation process as
 although
    'the bid formulation information is on the network files it will just sit t
ere unless
    'the cells are told to go get the information and start to formulate the bic
. We do this
    'by opening the "r:\Factory\Config\nodefile.txt" file that tells us what the
 activenodes
    'in the factory are and therefore what nodes should go get the bidding data
and then by
    'sending a "1" to each of those cells in the file "Bidfrmstrt" database bloc
k.

    'declare variables

    Dim ThisRecord As nodelistrecord
    Dim position As Integer
    Dim Firstime As Integer


    If FileLen(Netdrive & "Factory\Config\nodefile.txt") = 0 Then 'if there is
othing in the file then exit
    'this subprocedure
        Exit Sub
    End If


    'Assign a free number to this file
    Filenum = FreeFile
    'Calculate the recordlength in Bytes
    recordlength = Len(ThisRecord)

    'Open the file for random access with allowable functions read and write
    Open Netdrive & "Factory\Config\nodefile.txt" For Random Access Read Shared
As Filenum Len = recordlength

    position = 1

    Do Until EOF(Filenum)
        'get the name of the first node from the file
        Get Filenum, position, ThisRecord
        'increment position variable so the next nodename will be gotten
        position = position + 1
        'now send a "1" to the "bidfrmstsrt" in the cell denoted by the record.
        Node = Trim(ThisRecord.Node)
        If Trim(Node) Like "FACT" Or Len(Trim(Node)) >= 10 Then
            'we do not need to send a prompt to the factory controller node
            GoTo Nextstep
        End If

        'make a copy of the bidfile for each node so that a clash will not occu
 on the
        'network if it cannot fileshare
```

```
        FileCopy Netdrive & "Factory\Bidding\Bidfile.bdf", Netdrive & "Cellcont`
idding\BidF" & Trim(Node) & ".bdf"

        'also make a copy of the internodal travel time file for each node so tl
t the filesharing
        'problem does not occur

        FileCopy Netdrive & "Factory\Bidding\Nodetime\Nodedely.txt", Netdrive &
 Cellcont\Bidding\Nodetime\N" & Trim(Node) & "dely.txt"


        'prompt each node in the Factory to start bid formulation.

        Blocktyp = "Bidfrmstrt"
        Field = "A_CV"
        Texts = "1"
        Call sendinfo(Node, Blocktyp, Field, Texts)


extstep:
    Loop

    Close Filenum
    Bidtimer.Enabled = True

lse
ryHere:
        'we are required to see if there is anything in the queuelist listbox
        If Len(Trim(QueueList.List(0))) > 0 Then

        Tool_Avle.Tool_Timer.Enabled = False
        Tool_Avle.Bid_eval.Enabled = False

        'there is something in the Factory Queue list
        'send a message to the Factory Controller FixDmacs "BidOperate" block t
ling it to stop rotating
        'the token
        Busy_Jobs_Auction = 0
        Node = "Fact"
        Tags = "Bidoperate"
        Field = "A_CV"
        Texts = "1"

        Call sendinfo(Node, Tags, Field, Texts) 'STOP THE BIDTOKEN FROM ROTATIN

        'tell the system that it is an item from the Factory Queue being auctio
ed
        Process_Busy_Auctioning = 0


        'tell the system that the factory is conducting an auction

        FactoryIsAuctioning = 1

        'open the job to be auctioned jobfile
        Filenum = FreeFile 'deffine a filenumber that is not being used.
```

```
        recordlength = Len(Savetask) 'define arrays storage size required.

        'the name of the file to open is the first name in the list
        Filetoopen = QueueList.List(0)
      '  MsgBox (Filetoopen)
        Directorynam = Trim(Jobfilepath)
  '      MsgBox (Directorynam)

        Open Trim(Directorynam) & Trim(Filetoopen) & ".iff" For Random Access R
ad Write As Filenum Len = recordlength


        ReDim Savetasks(1) 'Redimension array to hold first record only

        'Get the status of the current job
        Get Filenum, 1, Savetasks(1)

        'get the urgent variable from the first record
        IsurgenT = Savetasks(1).Force

        ReDim Preserve Savetasks(Savetasks(1).Status)

        'now get all the information required about the current task to be done

        Get Filenum, Savetasks(1).Status, Savetasks(Savetasks(1).Status)

        'Text1.Text = Savetasks(Savetasks(1).Status).Jobname
        'Text1.Text = Savetasks(Savetasks(1).Status).Infofilename
        'Text1.Text = Savetasks(Savetasks(1).Status).FinalTask
        'Text1.Text = Savetasks(Savetasks(1).Status).Tasknumber
        'Text1.Text = Savetasks(Savetasks(1).Status).machinetype
        'Text1.Text = Savetasks(Savetasks(1).Status).Taskname
        'Text1.Text = Savetasks(Savetasks(1).Status).EstCycletime
        'Text1.Text = Savetasks(Savetasks(1).Status).setuptime
        'Text1.Text = Savetasks(Savetasks(1).Status).Estloadunloadtime
        'Text1.Text = Savetasks(Savetasks(1).Status).NooFCycles
        'Text1.Text = Savetasks(Savetasks(1).Status).CNCfilename
        'Text1.Text = Savetasks(Savetasks(1).Status).Geometricfilename
        'Text1.Text = Savetasks(Savetasks(1).Status).Tool1required
        'Text1.Text = Savetasks(Savetasks(1).Status).Tool2required
        'Text1.Text = Savetasks(Savetasks(1).Status).Tool3required
        'Text1.Text = Savetasks(Savetasks(1).Status).Tool4required
        'Text1.Text = Savetasks(Savetasks(1).Status).Fixturesfilename
        'Text1.Text = Savetasks(Savetasks(1).Status).Status
        'Text1.Text = Savetasks(Savetasks(1).Status).Duedate
        'Text1.Text = Savetasks(Savetasks(1).Status).Force
        'Text1.Text = Savetasks(Savetasks(1).Status).SetUpInstructionsFilename

        'Type Bid_data_To_Be_sent
        '   Jobname As String * 10
        'Machinetype As Single
        'EstCycletime As Integer
        'Setuptime As Integer
        'LoadUnloadtime As Integer
        'NoofCycles As Integer
        'Tool1required As Single
```

```
        'Tool2required As Single
        'Tool3required As Single
        'Tool4required As Single
        'Activate As Integer
        'force as Integer
        'AuctioningCell As String * 8
        'End Type


        'First kill the oldfile bidfile
        Kill Netdrive & "Factory\Bidding\Bidfile.bdf"

        'denote a filenumber for the new file.
        Filenumb = FreeFile
        recordlength = Len(ThisbidFile)

        'Now we must open a new copy of the file and write the bidding objects t
 it.
        Thisfilename = Netdrive & "Factory\Bidding\Bidfile.bdf"
'        MsgBox (Thisfilename)

        Open Netdrive & "Factory\Bidding\Bidfile.bdf" For Random Access Write Sh
red As Filenumb Len = recordlength


        ThisBidding(1).Jobname = Savetasks(Savetasks(1).Status).Jobname
'        MsgBox (Thisbidding(1).Jobname)
        ThisBidding(1).Machinetype = Savetasks(Savetasks(1).Status).Machinetype
'        MsgBox CStr(Thisbidding(1).Machinetype)
        ThisBidding(1).EstCycletime = Savetasks(Savetasks(1).Status).EstCycletim
e
'        MsgBox CStr(Thisbidding(1).EstCycletime)
        ThisBidding(1).setuptime = Savetasks(Savetasks(1).Status).setuptime
        ThisBidding(1).Loadunloadtime = Savetasks(Savetasks(1).Status).Estloadun
oadtime
        ThisBidding(1).NooFCycles = Savetasks(Savetasks(1).Status).NooFCycles
        ThisBidding(1).Tool1required = Savetasks(Savetasks(1).Status).Tool1requi
ed
'        MsgBox CStr(Thisbidding(1).Tool1required)
'        Thisbidding(1).Tool2required = Savetasks(Savetasks(1).Status).Tool2requ
ired
'        MsgBox CStr(Thisbidding(1).Tool2required)
'        Thisbidding(1).Tool3required = Savetasks(Savetasks(1).Status).Tool3requ
ired
'        MsgBox CStr(Thisbidding(1).Tool3required)
'        Thisbidding(1).Tool4required = Savetasks(Savetasks(1).Status).Tool4requ
ired
'        MsgBox CStr(Thisbidding(1).Tool4required)
        ThisBidding(1).Activate = 1
        ThisBidding(1).Force = IsurgenT
        ThisBidding(1).AuctioningCell = "FACT"

        Put Filenumb, 1, ThisBidding(1) 'put the bidding objects into the bidfil

        Close Filenumb
```

```
        Close Filenum

        DoEvents
        'However we must also prompt the cells to begin the Bid formation proce
s as although
        'the bid formulation information is on the network files it will just si
t there unless
        'the cells are told to go get the information and start to formulate th
 bid. We do this
        'by opening the "r:\Factory\Config\nodefile.txt" file that tells us what
 the activenodes
        'in the factory are and therefore what nodes should go get the bidding d
ata and then by
        'sending a "1" to each of those cells in the file "Bidfrmstrt" database
block.


        If FileLen("C:\Factory\Config\nodefile.txt") = 0 Then 'if there is noth
ng in the file then exit
        'this subprocedure
            Exit Sub
        End If

        'Assign a free number to this file
        Filenum = FreeFile
        'Calculate the recordlength in Bytes
        recordlength = Len(ThisRecord)

        'Open the file for random access with allowable functions read and writ
        Open "C:\Factory\Config\nodefile.txt" For Random Access Read Shared As
ilenum Len = recordlength

        position = 1

        Do Until EOF(Filenum)
            'get the name of the first node from the file
            Get Filenum, position, ThisRecord
            'increment position variable so the next nodename will be gotten
            position = position + 1
            'now send a "1" to the "bidfrmstsrt" in the cell denoted by the rec
rd.
            Node = Trim(ThisRecord.Node)
           ' MsgBox CStr(Len(Trim(node)))
            If Trim(Node) Like "FACT" Or Len(Trim(Node)) > 8 Then
                'we do not need to send a prompt to the factory controller node
                GoTo Nexstep
            End If

            'make a copy of the bidfile for each node so that a clash will not
ccur on the
            'network if it cannot fileshare

            FileCopy Netdrive & "Factory\Bidding\Bidfile.bdf", Netdrive & "Cellc
ont\Bidding\BidF" & Trim(Node) & ".bdf"

            'also make a copy of the internodal travel time file for each node s
```

```
 that the filesharing
            'problem does not occur


            FileCopy Netdrive & "Factory\Bidding\Nodetime\Nodedely.txt", Netdri
e & "Cellcont\Bidding\Nodetime\N" & Trim(Node) & "dely.txt"


            'prompt each node in the Factory to start bid formulation.

            Blocktyp = "Bidfrmstrt"
            Field = "A_CV"
            Texts = "1"
            Call sendinfo(Node, Blocktyp, Field, Texts)


exstep:
        Loop

        Close Filenum

        Else
            'there are no jobs to be done in the Factory Controller queue or th
 processes busy
            'queue therefore release the bidtoken and quit
            Node = "Fact"
            Tags = "Bidoperate"
            Field = "A_CV"
            Texts = "0"
            Call sendinfo(Node, Tags, Field, Texts) 'Release the bidtoken as th
re are no jobs to
            'auction and exit the event
            Exit Sub
        End If


'enable the timer that waits for sometime for the bids to be formulated on the
des in the
 factory. When the interval time is reached then the bid comparison program is
ctivated by
 the timer event.
idtimer.Enabled = True

 nd If
 BidToken.Text = 0

nd Select

End Sub

 ub CalcMTBC_Change ()
Dim Tk As Double
```

```
'The idea behind this event procedure is that we allow the Cell Controllers to
'tell the Factory Controllers everytime they finish a job so that the Factory
'Controller is able to update the MEan Time Between Completion of the Jobs in
'the Factory.

'the formula for Mean Time Between Completion is

'MtBCk = (MtBC(k-1) * (k-1)) + ((Tk - (Tk-1))/k

'where

'K = #Jobs just completing
'Tk = Time of the current job completion event
'Tk-1 = time the previous job completed processing
'MTBC(k-1) = Mean Time Between Completion for the jobs that had completed proces
sing before the
            'current one being considered

Select Case Val(CalcMTBC.Text)

Case Is = 0 'the system is just starting or the block has been reset
    Exit Sub

Case Is = 1 'then the Factory Controller has just been told to calculate the
            'new mtbc based on a new job being finished

'Dim k As Integer 'number of jobs just completing
'Dim Tk_1 As Double 'time previous job completed
'Dim MtBc_K_1 As Double 'MTBC at last time interval
'Dim MtBcK As Double 'mtbc for the current time interval
If Sim_Start = 1 Then
    Tk_1 = StartTime
    Sim_Start = 0
End If

Tk = CalcStartTime()

K = K + 1

MTBcK = ((MTBc_K_1 * (K - 1)) + (Tk - (Tk_1))) / K

Tk_1 = Tk

MTBc_K_1 = MTBcK

MTBc.AddItem CStr(MTBcK * 3600)


End Select


'now reset the database block that is responsible for polling an updated MTBC


Node = "fact"
```

```
UEUESTA.FRM - 45

ags = "Updatemtbc"
ield = "A_CV"
exts = "0"


all sendinfo(Node, Tags, Field, Texts)


nd Sub

b CellisAuct_Change ()
im Filenum As Integer

eclare variables
im Filetoopen As String * 30
im recordlength As Integer
im Savetasks() As JobRecord
im Savetask As JobRecord
im Directorynam As String * 24
m i As Integer
im Thisfilename As String * 30
m IsurgenT As Integer

eclare bid file variables.
im ThisbidFile As Bid_data_To_Be_sent
im Filenumb As Integer


n Error Resume Next

elect Case Val(CellisAuct.Text)

ase Is = 1

ool_Avle.Tool_Timer.Enabled = False
ool_Avle.Bid_eval.Enabled = False


actoryIsAuctioning = 0 'set a global variable to state that the factory control
r is not
uctioning

hen this turns to a 1 we are required to open a network file called
:\Cellcont\Auction\Auctdata.txt and read the name of the jobfile that is requi
d to be
auctioned as well as the name of the node that the auction originates from.

give the file to be opened a number
ilenum = FreeFile

open the file for input into memory variables
pen Netdrive & "Cellcont\Auction\Auctdata.txt" For Input As Filenum Len = 50

et the name of the job to be auctioned and store that name in the module level
"Jobname"
variable
```

```
Line Input #Filenum, Jobname


'get the name of the node that the auction is coming from and store it in the m
dule level
'"nodename" variable
Line Input #Filenum, Nodename

'get the number of the VMD that is holding the bid and store it in the module le
vel "VMDNumber"
'variable
Line Input #Filenum, VMDNumber



Close Filenum

DoEvents
'MsgBox (Nodename)
'MsgBox ("VMDNumber is " & VMDNumber)
'MsgBox (Jobname)


'now we have the name of the job we can open that file and distribute the biddi
g objects to all
'the nodes as required.
Filenum = FreeFile 'define a filenumber that is not being used.
recordlength = Len(Savetask) 'define arrays storage size required.

Filetoopen = Trim(Jobname)
'MsgBox (Filetoopen)
Directorynam = Trim(Jobfilepath)
'MsgBox (Directorynam)

Open Trim(Directorynam) & Trim(Filetoopen) & ".iff" For Random Access Read Write
 As Filenum Len = recordlength

ReDim Savetasks(1) 'Redimension array to hold first record only

'Get the status of the current job
Get Filenum, 1, Savetasks(1)

IsurgenT = Savetasks(1).Force

'MsgBox (CStr(IsurgenT))
ReDim Preserve Savetasks(Savetasks(1).Status)
'MsgBox (CStr(Savetasks(1).Status))
'now get all the information required about the current task to be done.

Get Filenum, Savetasks(1).Status, Savetasks(Savetasks(1).Status) 'get the data a
ssociated with
'the current task


DoEvents
'First kill the oldfile that is on the network.
```

```
UEUESTA.FRM - 47

ill Netdrive & "Factory\Bidding\Bidfile.bdf"

'denote a filenumber for the new file.
ilenumb = FreeFile

'define recordlength
ecordlength = Len(ThisbidFile)

'Now we must open a new copy of the file and write the bidding objects to it.
hisfilename = Netdrive & "Factory\Bidding\Bidfile.bdf"
MsgBox (Thisfilename)

'open the bidfile for output .
pen Netdrive & "Factory\Bidding\Bidfile.bdf" For Random Access Write As Filenu
 Len = recordlength


write the bidding data to the file
ThisBidding(1).Jobname = Savetasks(Savetasks(1).Status).Jobname
hisBidding(1).Machinetype = Savetasks(Savetasks(1).Status).Machinetype
MsgBox CStr(Thisbidding(1).Machinetype)
ThisBidding(1).EstCycletime = Savetasks(Savetasks(1).Status).EstCycletime
hisBidding(1).setuptime = Savetasks(Savetasks(1).Status).setuptime
hisBidding(1).Loadunloadtime = Savetasks(Savetasks(1).Status).Estloadunloadtime
ThisBidding(1).NooFCycles = Savetasks(Savetasks(1).Status).NooFCycles
hisBidding(1).Tool1required = Savetasks(Savetasks(1).Status).Tool1required
MsgBox CStr(Thisbidding(1).Tool1required)
'Thisbidding(1).Tool2required = Savetasks(Savetasks(1).Status).Tool2required
'MsgBox CStr(Thisbidding(1).Tool2required)
Thisbidding(1).Tool3required = Savetasks(Savetasks(1).Status).Tool3required
'MsgBox CStr(Thisbidding(1).Tool3required)
'Thisbidding(1).Tool4required = Savetasks(Savetasks(1).Status).Tool4required
MsgBox CStr(Thisbidding(1).Tool4required)
hisBidding(1).Activate = 1
ThisBidding(1).Force = IsurgenT
MsgBox cs(ThisBidding(1).Force)
hisBidding(1).AuctioningCell = Nodename

ut Filenumb, 1, ThisBidding(1)
lose Filenumb

lose Filenum 'close the nodefile


oEvents
However we must also prompt the cells to begin the Bid formation process as al
hough
the bid formulation information is on the network files it will just sit there
nless
'the cells are told to go get the information and start to formulate the bid. W
do this
by opening the "r:\Factory\Config\nodefile.txt" file that tells us what the ac
ivenodes
'in the factory are and therefore what nodes should go get the bidding data and
hen by
'sending a "1" to each of those cells in the file "Bidfrmstrt" database block.
```

```
'declare variables

Dim ThisRecord As nodelistrecord
Dim position As Integer
Dim Firstime As Integer


If FileLen("C:\Factory\Config\nodefile.txt") = 0 Then 'if there is nothing in th
e file then exit
'this subprocedure
    Exit Sub
End If


'Assign a free number to this file
Filenum = FreeFile
'Calculate the recordlength in Bytes
recordlength = Len(ThisRecord)

'Open the file for random access with allowable functions read and write
Open "C:\Factory\Config\nodefile.txt" For Random Access Read Shared As Filenum L
en = recordlength

position = 1

Do Until EOF(Filenum)
    'get the record from the file
    Get Filenum, position, ThisRecord
    position = position + 1
    'now send a "1" to the "bidfrmstsrt" in the cell denoted by the record.
    Node = Trim(ThisRecord.Node)
    If Trim(Node) Like "FACT" Or Len(Trim(Node)) = 15 Then
        GoTo Nextstepp
    End If

    'otherwise the nodename is valid

    'make a copy of the bidfile for each node so that a clash will not occur on
the
    'network if it cannot fileshare

    FileCopy Netdrive & "Factory\Bidding\Bidfile.bdf", Netdrive & "Cellcont\Bidd
ing\BidF" & Trim(Node) & ".bdf"

            'also make a copy of the internodal travel time file for each node s
o that the filesharing
            'problem does not occur


    FileCopy Netdrive & "Factory\Bidding\Nodetime\Nodedely.txt", Netdrive & "Ce
llcont\Bidding\Nodetime\N" & Trim(Node) & "dely.txt"


    Blocktyp = "Bidfrmstrt"
    Field = "A_CV"
```

```
    Texts = "1"
    Call sendinfo(Node, Blocktyp, Field, Texts)
Nextstepp:
'oop

Close Filenum

 enable the timer that waits for sometime for the bids to be formulated on the r
odes in the
'factory. When the interval time is reached then the bid comparison program is a
;tivated by
 the timer event.
Bidtimer.Enabled = True

End Select


 now reset the database block responsible for this
Node = "Fact"
 ags = "CellisAuct"
'ield = "A_CV"
Texts = "0"

 all sendinfo(Node, Tags, Field, Texts)

nd Sub

Sub Command2_Click ()
Bidtimer.Enabled = True
sgBox (CStr(Bidtimer.Interval))
End Sub

 ub Command3_Click ()

f Len(Queuestat.QueueList.List(Queuestat.QueueList.ListIndex)) = 0 Then
    MsgBox "You have not selected a job to be analysed. Please do so."
xit Sub
End If

ob_Status.Show


 nd Sub


 ub Exit_Click ()
ueuestat.Hide
End Sub

 ub Form_Activate ()
'This application allows the user to view the current factory
'queue status and to modify it if required

Dim Thistask As JobRecord
```

```
QUEUESTA.FRM - 50

Dim Filenum As Integer
Dim Filenam As String * 50
Dim ThisFileslength As Integer
Dim i As Integer
Dim Thisfilesname As String * 60
Dim Reply As Integer
Dim Jobspending() As Pendingjob
Dim Thisjob As Pendingjob

Jobspend.Clear
'open the file that details all of the jobs that are waiting to be done.

Thisfilesname = "C:\Factory\Queue\jobspend.jpf"
Filenum = FreeFile

'get the length of the file
ThisFileslength = getfilesLength5(Thisfilesname, Thisjob, Jobspending(), Filenu
)

'Now write the contents of the file to memory.
Call WriteToMemory5(Filenum, ThisFileslength, Jobspending())

For i = 1 To ThisFileslength
    Jobspend.AddItem (Jobspending(i).Jobname), i - 1
Next

Close Filenum

End Sub

Sub Form_Load ()


'This application allows the user to view the current factory
'queue status and to modify it if required

Dim Thistask As JobRecord
Dim Filenum As Integer
Dim Filenam As String * 50
Dim ThisFileslength As Integer
Dim i As Integer
Dim Thisfilesname As String * 60
Dim Reply As Integer
Dim Jobspending() As Pendingjob
Dim Thisjob As Pendingjob
Dim Thisjobname As String

'Define Factory Job Queueing variables
Dim Currqueue As Factqueue
Dim Thisqueue()  As Factqueue
Dim FactQnum As Integer
Dim qFilename As String * 40
Dim Recordleng As Integer
Dim Thislengths As Integer

'auction timer variables
```

```
UEUESTA.FRM - 51

im Aucttimeout As String
im Currentinterval As Integer


n Error Resume Next

'first set the Auction Timeout timer interval
ilenum = FreeFile
)pen "C:\Factory\System\Config\Aucttime.sys" For Input As Filenum Len = 4
Print #Filenum, Aucttimeout
lose Filenum

urrentinterval = Bidtimer.Interval
idtimer.Interval = (Val(Aucttimeout) * Currentinterval)

'initialise the textboxes so that change events can occur
idtoken.Text = ""
ellisAuct.Text = ""



open the file that details all of the jobs that are waiting to be done.

hisfilesname = "C:\Factory\Queue\jobspend.jpf"
ilenum = FreeFile

'get the length of the file
hisFileslength = getfilesLength5(Thisfilesname, Thisjob, Jobspending(), Filenu


Now write the contents of the file to memory.
Call WriteToMemory5(Filenum, ThisFileslength, Jobspending())

or i = 1 To ThisFileslength
   Jobspend.AddItem (Jobspending(i).Jobname), i - 1
Next

lose Filenum

We are also required to see the current queue status
so show it in the listbox

define filename holding the queueing data
Filename = "C:\Factory\Queue\Queuejob.que"

Free up a number for the file
actQnum = FreeFile

Get the array record length
ecordleng = Len(Currqueue)

'Get the file's length
hislengths = GetFileslength6(Trim(qFilename), Currqueue, Thisqueue(), FactQnum
'Now read the information in the file to memory
```

```
Call WriteToMemory6(FactQnum, Thislengths, Thisqueue())

'Now write the information into the listbox
For i = 1 To Thislengths
    QueueList.AddItem Thisqueue(i).Jobname
Next

Close FactQnum


'we are also required to set up automatic links between this form's bidtoken tex
tbox and
'the bidtoken database block so thta we know when this node has been given the
pportunity to
'hold an auction.

'run the dynamic data exchange server
Call RunDDEServer

'form tyhe bidtoken link as mentioned
Call CommDDElink(Bidtoken, "FACT.BIDTOKEN.f_CV")


'form the link between the BidAcknowledge block and the bid Acknowledge textbox.
Call CommDDElink(Bid_Ack, "FACT.BIDACKNOWL.F_CV")

'form link between the Cellisauct block and the Cell is Auctioning textbox
Call CommDDElink(CellisAuct, "FACT.cellisauct.F_CV")


'form a link between the updateMTBC database block and the CalcMTBC textbox
Call CommDDElink(CalcMTBC, "FACT.updateMTBC.F_CV")




'also open the system files that hold the information regarding the various que
es that are
'represented as listboxes on this form.
Dim Listlength As String

Filenum = FreeFile 'define a freefile

Recordleng = 20 'get enough room to occupy file elements


'open the file containing the jobs for which the processes are too busy to carr
 out.
'it is opened for sequential output
Open "C:\Factory\Queue\ProcJbBs.que" For Input As Filenum Len = Recordleng

Line Input #Filenum, Listlength

If Trim(Listlength) Like "" Then
    Close Filenum
```

```
xit Sub
nd If

or i = 1 To CInt(Listlength) 'write each of the items in the list to the file.
    Line Input #Filenum, Thisjobname
    Processes_Busy.AddItem Thisjobname
ext

lose Filenum 'close the file

nd Sub

Sub Form_Unload (Cancel As Integer)
ueuestat.Hide
nd Sub

ub QueueList_Click ()

If PositionToMoveto >= 1 Then 'the position to move the job to has been selecte
and we need to
specify the job to move
    CheckSelected = 2
    JobpositionToMove = QueueList.ListIndex
lse 'the position to move to has not be specified so specify
    PositionToMoveto = QueueList.ListIndex + 1
nd If


nd Sub

Sub QueueList_DblClick ()
ommand3_Click
nd Sub

Sub Re_Order_Click ()

Define Factory Job Queueing variables
im Currqueue As Factqueue
im Thisqueue()  As Factqueue
im FactQnum As Integer
Dim qFilename As String * 40
im Recordleng As Integer
im Thislengths As Integer

h Error GoTo Thiserr
This enables a user to reposition a job in the queue which would result when th
e job's priority
changes for some reason

Declare variables
im Msg As String
im Reply As Integer
Dim Typ As Integer
im Title As String
im JobTomove As String * 10
Dim i As Integer
```

```
If PositionToMoveto = 0 Then 'nothing has been selected so prompt for it
    Msg = "Please choose the position you want to move the job to."
    Title = "Reorder Procedure"
    Typ = MB_ICONSTOP + MB_OK
    MsgBox Msg, Typ, Title
    CheckSelected = 0
    PositionToMoveto = 0
    Exit Sub
End If


If CheckSelected <> 2 Then
    Msg = "Please choose the job that you want to reorder."
    Title = "Reorder Procedure"
    Typ = MB_ICONSTOP + MB_OK
    MsgBox Msg, Typ, Title
    Exit Sub
End If


'Now we have to move the job as specified

JobTomove = QueueList.List(JobpositionToMove)

QueueList.RemoveItem (JobpositionToMove)

QueueList.AddItem Trim(JobTomove), (PositionToMoveto - 1)


CheckSelected = 0
PositionToMoveto = 0

'Now we are required to save the changes to file
'Now kill the old queuefile and resave the contents of the listbox to the new f
le
'version

Kill "C:\Factory\Queue\Queuejob.que"

'get a new number for the file
FactQnum = FreeFile

'Get the array record length
Recordleng = Len(Currqueue)


'open the new version of the file
Open "C:\Factory\Queue\Queuejob.que" For Random Access Read Write Shared As Fac
Qnum Len = Recordleng

'Redimension the array to take all the list's contents
ReDim Thisqueue(QueueList.ListCount) '+1
```

```
'Write the array's length to memory
Thisqueue(1).Filelength = QueueList.ListCount '+ 1

'write the contents of the list to the array
For i = 1 To QueueList.ListCount '+ 1
    Thisqueue(i).Jobname = Trim(QueueList.List(i - 1))
Next

'Put the contents of the array to file
For i = 1 To QueueList.ListCount '+ 1
    Put FactQnum, i, Thisqueue(i)
Next

'Now close the file
Close FactQnum

Thiserr:
End Sub

Sub RemoveJob_Click ()
 Define message box variables
Dim Msg As String
Dim Reply As Integer
Dim Title As String * 20
Dim Typ  As Integer


'Define Factory Job Queueing variables
Dim Currqueue As Factqueue
Dim Thisqueue()  As Factqueue
Dim FactQnum As Integer
Dim qFilename As String * 40
Dim Recordleng As Integer
Dim Thislengths As Integer

 Define counter variables
Dim i As Integer



 First make sure that the user wants to delete the selected item
If Len(QueueList.List(QueueList.ListIndex)) = 0 Then
    Exit Sub 'as nothing has been selected
End If

'otherwise something has been selected so ask the user if he is really sure that
 he wants to
'delete the selected item
Msg = "If you select a job for deletion that is already had work done on it mate
rial may be "
Msg = Msg & "wasted. Are you sure that you want to delete the selected job?"
Title = "Deletion Verification"
Typ = MB_ICONSTOP + MB_YESNO
Reply = MsgBox(Msg, Typ, Title)
```

```
If Reply = IDYES Then
    QueueList.RemoveItem (QueueList.ListIndex)

    'Now kill the old queuefile and resave the contents of the listbox to the r
w file
    'version
    Kill "C:\Factory\Queue\Queuejob.que"

    'get a new number for the file
    FactQnum = FreeFile

    'Get the array record length
    Recordleng = Len(Currqueue)


    'open the new version of the file
    Open "C:\Factory\Queue\Queuejob.que" For Random Access Read Write Shared As
FactQnum Len = Recordleng

    'Redimension the array to take all the list's contents
    ReDim Thisqueue(QueueList.ListCount + 1)

    'Write the array's length to memory
    Thisqueue(1).Filelength = QueueList.ListCount + 1

    'write the contents of the list to the array
    For i = 1 To QueueList.ListCount + 1
        Thisqueue(i).Jobname = Trim(QueueList.List(i - 1))
    Next

    'Put the contents of the array to file
    For i = 1 To QueueList.ListCount + 1
        Put FactQnum, i, Thisqueue(i)
    Next

    'Now close the file
    Close FactQnum

End If

End Sub

Sub STart_Data_Coll_Click ()
'declare variables
Dim Filenum As Integer
Dim recordlength As Integer
'define nodefile variables
Dim ThisRecord As nodelistrecord
Dim position As Integer


'the purpose of this procedure is to reset the Average Queue Length statistic in
 the
'Cell Controller queueing applications as well as the Mean Time between completi
on between
'jobs that have been finished.
```

```
'Assign a free number to this file
Filenum = FreeFile
'Calculate the recordlength in Bytes
recordlength = Len(ThisRecord)

'Open the file for random access with allowable functions read and write
Open Netdrive & "Factory\Config\nodefile.txt" For Random Access Read Shared As F
ilenum Len = recordlength

position = 1

Do Until EOF(Filenum)
'get the name of the first node from the file
    Get Filenum, position, ThisRecord
    'increment position variable so the next nodename will be gotten
    position = position + 1
    'now send a "1" to the "bdToolstrt" in the cell denoted by the record.
    Node = Trim(ThisRecord.Node)
    If Trim(Node) Like "FACT" Then

        'we do not need to send a prompt to the factory controller node therefor
e look
        'at the next record to see if it is a cell controller
        GoTo TheNextstep
    End If

    'prompt each node in the Factory to start collecting bid info
    Blocktyp = "Starttime"
    Field = "A_CV"
    Texts = "1"

    Call sendinfo(Node, Blocktyp, Field, Texts)


heNextstep:
Loop

lose Filenum

'reset the MTBA variables

 = 0

k_1 = 0

MTBc_K_1 = 0

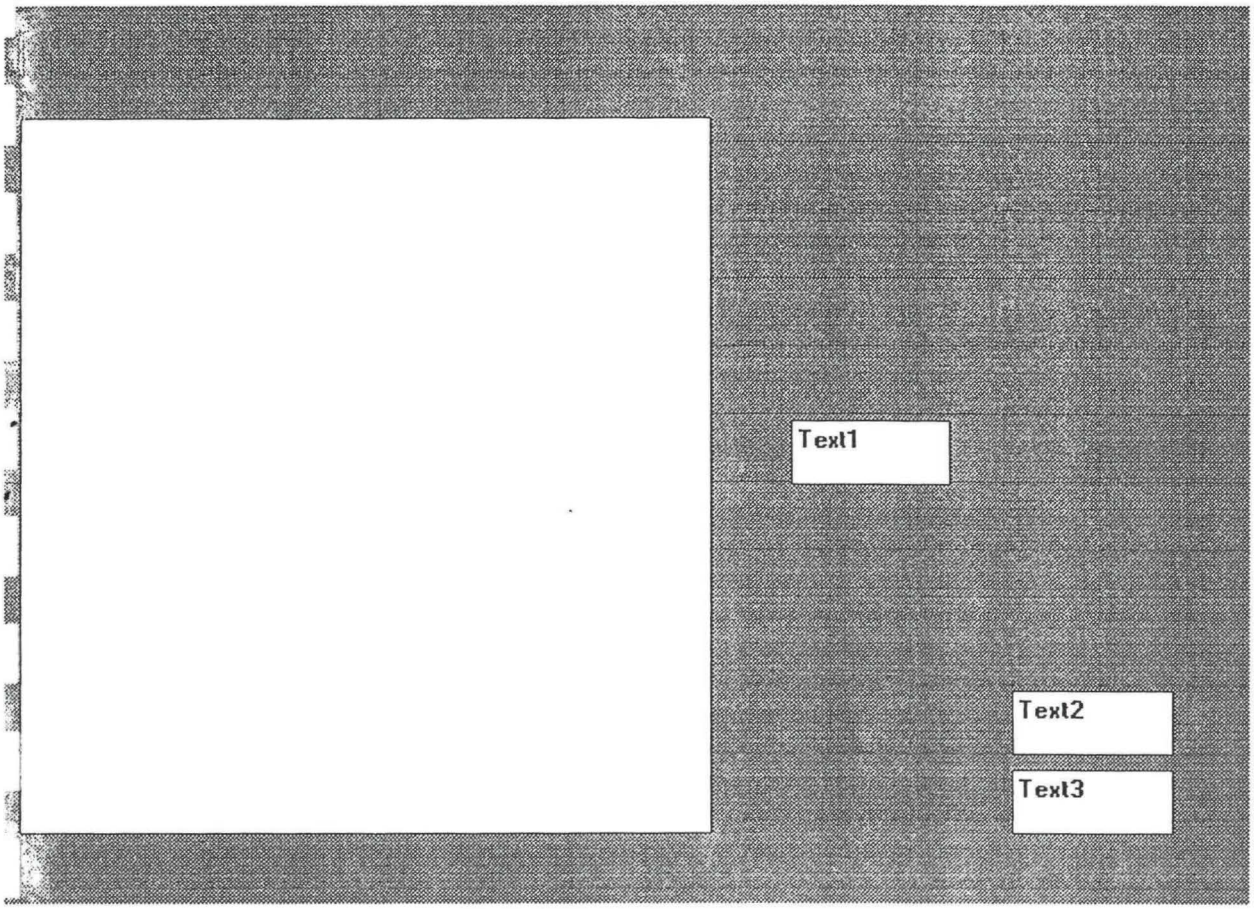
TBcK = 0

tartTime = CalcStartTime()
MsgBox (CStr(StartTime))

im_Start = 1

End Sub
```

```
Sub View_Click ()
Res_Form.Show
End Sub
```


Text1
Text2
Text3

```
RE_DEFIN.FRM - 1

Option Explicit
Dim Repeated As Integer
Dim DoNotsave As Integer


Sub Form_Load ()
'This particular form shows the bitmap associated with the process that you
'want to specify the capacity for
Dim Filenum As Integer
Dim Reply As String

Repeated = 0
Re_Define.Caption = Filenamed

On Error Resume Next
Filenum = FreeFile

'Reply = InputBox("Enter the Filename you want to open.", "Enter Filename", " "

Open "C:\Factory\Process\Specs\" & Trim(Filenamed) & ".bmp" For Binary As #File
um
Re_Define.OLE1.FileNumber = Filenum
Re_Define.OLE1.Action = 12
Close Filenum


End Sub

Sub Form_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Sing
e)
Dim Msg As String
Dim Title As String
Dim Answer As Integer
Dim Reply As Integer
Dim Defval As String
Dim Typ As Integer
Const XPosition = 5500
Const Yposition = 1200
Dim RemoveHere As Integer

Repeated = Repeated + 1
If Repeated > 1 Then
    Exit Sub
End If


On Error Resume Next

Title = "Enter Machine Capacity"
Msg = "Please enter the new maximum 'X' value(in mm) that exists for a machine o
f"
Msg = Msg & "this type in the Factory. "
Defval = "1"
Answer = CInt(InputBox(Msg, Title, Defval, XPosition, Yposition))
```

```
E_DEFIN.FRM - 2

Check that the user has input a valid entry

If Answer = 0 Then
      DoNotsave = 1
      Repeated = 1
      Exit Sub
nd If

'Assign maximum defined value to global variable
apacityStated = Answer

'Now we must replace the item's capacity that we are redefining
RemoveHere = Define_Processes.ProcessList.ListIndex

 Remove the old capacity from the list
Define_Processes.MaxCapacity.RemoveItem RemoveHere

 Now add the new item
Define_Processes.MaxCapacity.AddItem Trim(CapacityStated), RemoveHere


xit Sub
 rrorhandler:

Reply = ChekError(Err)

 elect Case Reply

ase Is = 1

'EntryIsBad = 1


xit Sub

 ase Is = 2
 esume Next

 ase Else
xit Sub


 nd Select


 nd Sub

 ub Form_Unload (Cancel As Integer)
 im Msg As String
)im Reply As Integer
 im Title As String
 im Typ As Integer
Dim ProcessSpc() As Processdef
 im Processtp As Processdef
 im Filelength As Integer
)im i As Integer
```

```
Dim Filenum As Integer
Dim IncrementSize As Integer

Cancel = -1

If DoNotsave = 1 Then 'if the user decide not to make a change then donot modify
 the proclist file
    Exit Sub
End If


Filenum = FreeFile

'Check how many bytes required to store array variables
Filelength = Len(Processtp)

'Redimension array to get filelength
ReDim ProcessSpc(1)

'Open the file
Open "c:\Factory\Process\files\Proclist.Pfl" For Random Access Read Write Shared
 As Filenum Len = Filelength

'Get the number of the last process added so that all of the information in
'the file is able to be loaded into memeory
Get Filenum, 1, ProcessSpc(1)

'Redimension the array so that all the process data defined so far can be
'recalled
ReDim Preserve ProcessSpc(ProcessSpc(1).Processnumber)

'Read all the process data so far defined into the memory array
For i = 1 To ProcessSpc(1).Processnumber
Get Filenum, i, ProcessSpc(i)
Next

'Find the array position which corresponds to the file being redefined.
For i = 1 To ProcessSpc(1).Processnumber
   Text2.Text = ProcessSpc(i).Processname
   Text3.Text = Filenamed
   If Trim(ProcessSpc(i).Processname) = Trim(Filenamed) Then
   GoTo ModifyFileAtThisSpot
   End If
Next
Cancel = 0
Exit Sub
ModifyFileAtThisSpot:

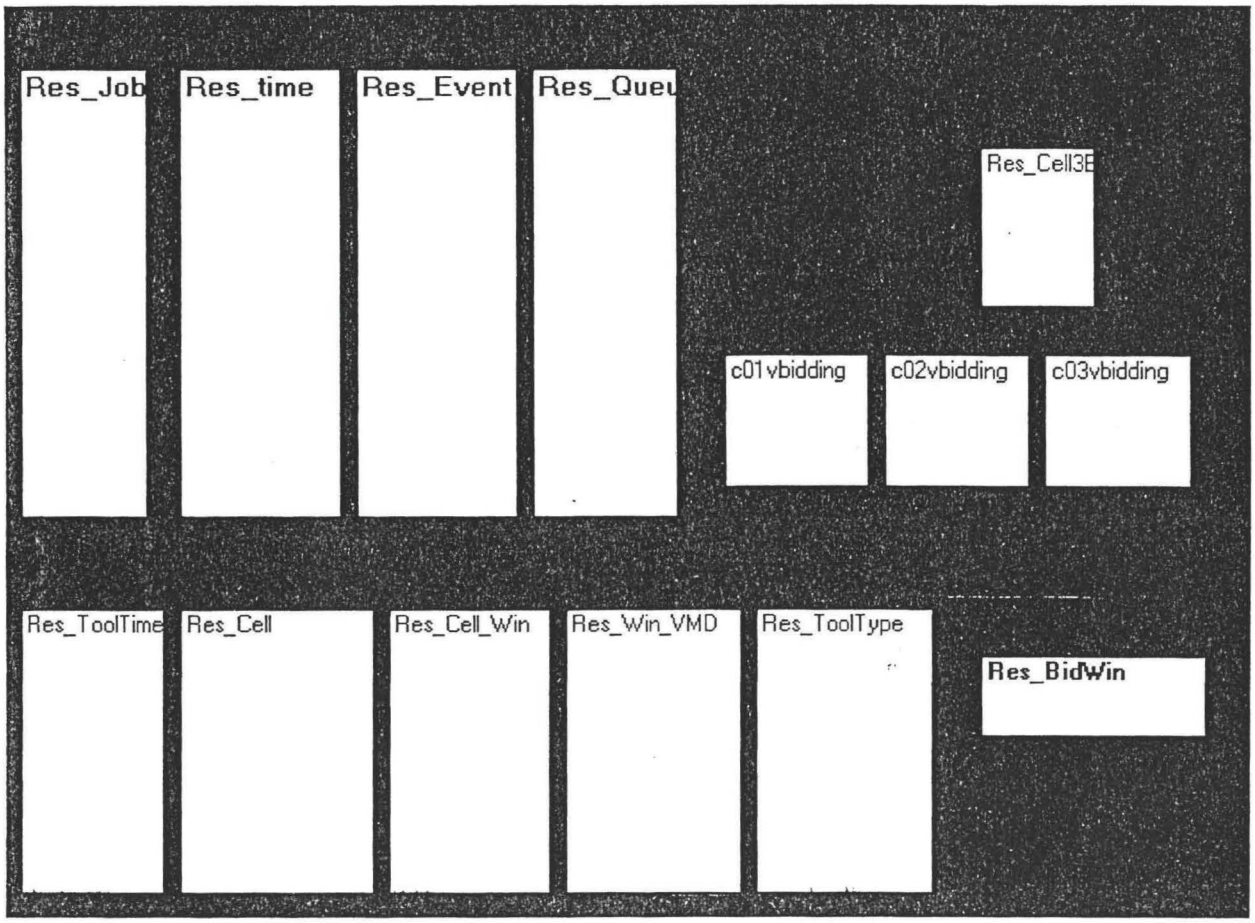
'Assign the newly defined capacity values to the array
IncrementSize = CInt(CapacityStated / 10)
Text3.Text = CInt(CapacityStated / 10)
'Assign the new values of capacity increment at the correct positions of the
'array
''ProcessSpc(i).Processname = Trim(Filenamed)
''ProcessSpc(i).Capindex1 = (IncrementSize)
''ProcessSpc(i).Capindex2 = (2 * (IncrementSize))
```

```
Text3.Text = ProcessSpc(i).Processnumber
 'ProcessSpc(i).Capindex3 = (3 * (IncrementSize))
''ProcessSpc(i).Capindex4 = (4 * (IncrementSize))
''ProcessSpc(i).Capindex5 = (5 * (IncrementSize))
 'ProcessSpc(i).Capindex6 = (6 * (IncrementSize))
''ProcessSpc(i).Capindex7 = (7 * (IncrementSize))
''ProcessSpc(i).Capindex8 = (8 * (IncrementSize))
 'ProcessSpc(i).Capindex9 = (9 * (IncrementSize))
''ProcessSpc(i).Capindex10 = (10 * (IncrementSize))
''Text2.Text = ProcessSpc(i).Capindex10
'Save the array back to file
For i = 1 To ProcessSpc(1).Processnumber
Put Filenum, i, ProcessSpc(i)
Next

'Close the file
Close Filenum

'EntryIsBad = 1

Cancel = 0
End Sub
```


Res_Job
Res_time
Res_Event
Res_Queu
Res_Cell3E
c01vbidding
c02vbidding
c03vbidding
Res_ToolTime
Res_Cell
Res_Cell_Win
Res_Win_VMD
Res_ToolType
Res_BidWin

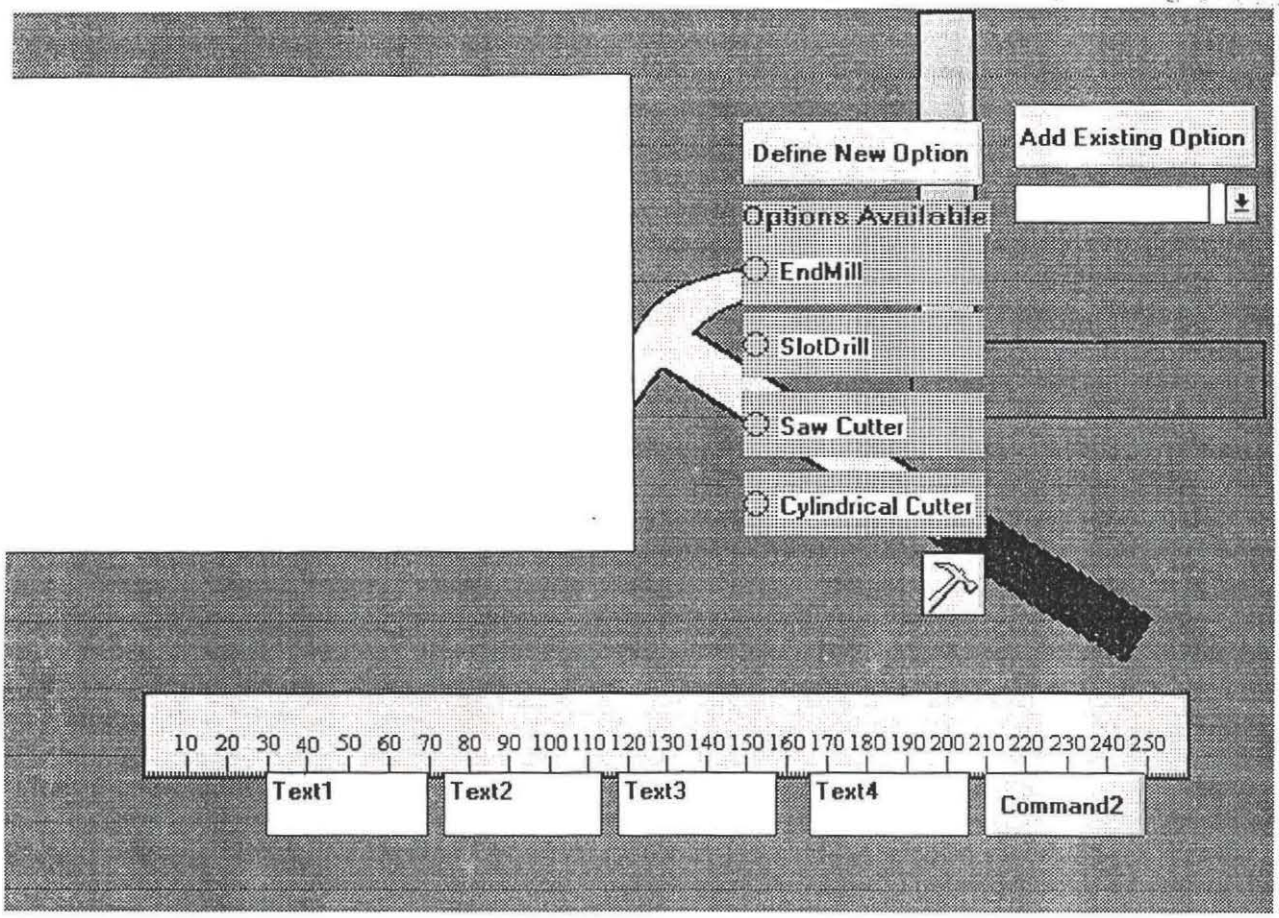
```
.ES_FORM.FRM - 1

)ption Explicit

Sub Form_Unload (cancel As Integer)

 f FactoryIsUnloading <> 1 Then
    cancel = -1
    Res_Form.Hide
    Exit Sub
    Else
        Exit Sub
 nd If

End Sub
```


Define New Option
Add Existing Option
Options Available
EndMill
SlotDrill
Saw Cutter
Cylindrical Cutter
10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240 250
Text1
Text2
Text3
Text4
Command2

```
.ATETOO.FRM - 1

 m OptionStateArray(4) As Optionstate

State that text comparison is not case-sensitive
 tion Compare Text

unction CheckOption (Options As String) As Integer

 m i As Integer


 r i = 0 To (OldOptions.ListCount) - 1
Check to see if the current option being changed is already in the
oldoptions list
 Options = Trim(OldOptions.List(i)) Then
If there is a match set the CheckOpion flag equal to 1
heckOption = 1
 d If
ext

 d Function

Sub mnu_timer (ThisTime)
 isTime = Timer
 o Until Timer > ThisTime + .13
Loop

 d Sub

ub Command1_Click ()
This procedure gives the user the opportunity to rename
an option button if another option exists that the
user wishes to use.

Declare variables to be used in this procedure
im OptionSelected As String
 im Msg As String
im Reply As Integer
Dim Title As String
 im Typ As Integer
im Request As String
Dim Answer As String * 20
 im Thisstate As Optionstate
im OldOption As String
Dim Thatstate As OldOptions
 im OptionList() As OldOptions
 im NextOldoption As String
Dim Position As Integer
 im OptionNumber As Integer
 im CheckValidity As Integer
Dim Filename As String
im MainToolist() As ToolIdRecord
 im ThisToolsId As ToolIdRecord
Dim NoOfTooltypes As Integer
im filenum As Integer
 im Result1 As Integer
Dim Result2 As Integer
```

```
'Ask the user whether or not he would like to modify an option button
Msg = "The option button as selected will be modified, do you wish to"
Msg = Msg & " continue?"
Title = "Modify Option Button"
Typ = MB_YESNO + MB_ICONQUESTION
Reply = MsgBox(Msg, Typ, Title)

'Check to see if the answer was cancel or ok
If Reply = IDNO Then
   Exit Sub 'Exit this procedure if the user doesn't want to continue
End If


'Define a generic optionbutton array
Dim ThisOption As OptionButton

'Find out which option button was selected by interrogating the array
Set ThisOption = Option1
If ThisOption.Value = True Then 'This option is selected
OptionSelected = "Option1"
   Else
   Set ThisOption = Option2
   If ThisOption.Value = True Then 'This option is selected
   OptionSelected = "Option2"
         Else
         Set ThisOption = Option3
         If ThisOption.Value = True Then 'This option is selected
         OptionSelected = "Option3"
         Else
            Set ThisOption = Option4
            If ThisOption.Value = True Then 'This option is selected
            OptionSelected = "Option4"
            End If
         End If
   End If
End If


'Ask the user to input the name of the option that he wants to use to replace
'the one selected

Request = "Enter the name of the new type of tool option you want to create."
Request = Request & "Do Not enter more than a two word description."
Title = "New Tool Option"
Answer = InputBox(Request, Title)

If Len(Trim(Answer)) = 0 Then 'Nothing was entered
Exit Sub
End If

'Trap the error in that a descriptor of more than two words was entered
'at once.
Result1 = InStr(1, Trim(Answer), " ")
Result2 = InStr((Result1 + 1), Trim(Answer), " ")
```

```
ATETOO.FRM - 3

 Result1 > 0 And Result2 > 0 Then

sg = "You have entered a three or more word descriptor rather than a one or two
ord"
g = Msg & " descriptor.Please re-enter a one or two word descriptor."

gBox Msg, MB_ICONSTOP, "Entry error"
it Sub
nd If

heck to see if this option has already been used before - if it has it will
be in the 'oldoptions' listbox
or i = 0 To (OldOptions.ListCount) - 1
  If Trim(Answer) = Trim(OldOptions.List(i)) Then 'This option has been used bef
re
  'so tell the user to use the 'Add Existing Option' key
      Msg = "This option already exists. Please click the 'Add Existing Option''
      Msg = Msg & " to view existing options. To add one of these options click'
      Msg = Msg & " the option button and the option required to be added from"
      Msg = Msg & " the list."
      Title = "Option Already Exists"
      Typ = MB_OK + MB_ICONEXCLAMATION
      Reply = MsgBox(Msg, Typ, Title)
      Exit Sub
  End If
ext


elect Case OptionSelected

ase Is = "Option1"
extOldoption = Trim(Option1.Caption)
ption1.Caption = Trim(Answer)


ase Is = "Option2"
extOldoption = Trim(Option2.Caption)
ption2.Caption = Trim(Answer)

Case Is = "Option3"
extOldoption = Trim(Option3.Caption)
ption3.Caption = Trim(Answer)

ase Is = "Option4"
extOldoption = Trim(Option4.Caption)
ption4.Caption = Trim(Answer)

nd Select

'now we must open up the file that contains all the names of the tools used
so far and add the name of the new tool to the file along with it's
identification number.

First get the total number of tool types that have already been specified.
ilename = "c:\factory\system\AlToolTyps.Tl"
filenum = FreeFile
```

```
NoOfTooltypes = GetFilesLength1(Filename, ThisToolsId, MainToolist(), filenum)

'Type ToolIdRecord
'CurrentToolId As Integer
'recordlength As Integer
'ToolType As String * 20
'ToolId As Integer
'End Type

'now write the list to memory
Call WriteToMemory(filenum, NoOfTooltypes, MainToolist())

'Now we must check to make sure that this tooltype has not been added to this
'file(The AlToolTyps File) by configuring the same tooltype for another machine
'that is we dont want to record the same tool type twice in the file and thus
'give the same tool type two different id's.

For i = 1 To NoOfTooltypes
    If Trim(MainToolist(i).Tooltype) = Trim(Answer) Then 'this tooltype has
    'already been recorded in the file that records all the different types
    'of tools , however we still need to save the new option button configurati
n.
    'therefore we do not leave this procedure rather we goto the linenumber
    'marked nextstep which will do this.
    Close filenum
    GoTo NextStep
    End If
Next


'Now We must interrogate the value of the current tool id and from this assign
'a new value for the tooltype being newly added to the file which is 1 greater
'than the current tool id.

'Increment the CurrentToolId to make sure that the new tool type being added
'has an identification number particular to it.
MainToolist(1).CurrentToolId = MainToolist(1).CurrentToolId + 1
Text1.Text = MainToolist(1).CurrentToolId
'Increment the recordlength variable at the first element of the array to
'show that the record is now one longer
MainToolist(1).recordlength = MainToolist(1).recordlength + 1

'add the new tool at its new position in the array
MainToolist(MainToolist(1).recordlength).Tooltype = Answer
MainToolist(MainToolist(1).recordlength).ToolId = MainToolist(1).CurrentToolId

Text1.Text = NoOfTooltypes
'now save all the data to file
Call WriteToFile(filenum, NoOfTooltypes, MainToolist())
Close filenum


NextStep:
'Now we must save the changes to file so that when this form is reloaded
'the new option is retained.
recordlength = Len(Thisstate)
```

```
 ilenum = FreeFile
pen "c:\Factory\System\" & ThisIndex & ".Opt" For Random Access Write Shared As
 filenum Len = recordlength
 Assign captions to file

OptionStateArray(1).OptionCaption = Option1.Caption
 ptionStateArray(2).OptionCaption = Option2.Caption
 ptionStateArray(3).OptionCaption = Option3.Caption
OptionStateArray(4).OptionCaption = Option4.Caption

 Put this data into file
For i = 1 To 4
ut filenum, i, OptionStateArray(i)
 ext

'Close the file
 lose filenum

'If the value of the caption already exists in the oldfile then there is no
 point in saving it's value to file again or writing it to the oldoptions list
 so exit the rest of this procedure which deals precisely with that

 heckValidity = CheckOption(NextOldoption)
f CheckValidity = 1 Then
Exit Sub
 nd If

'Now we must add the value of NextOldOption to the 'Oldoptions' listbox
ldOptions.AddItem NextOldoption


'open file which recalls all old options used
 ilenum = FreeFile
recordlength = Len(Thatstate)

eDim OptionList(1)
Open "c:\Factory\System\" & ThisIndex & ".old" For Random Access Read Write Sha
d As filenum Len = recordlength

'Redimension record to 1 unit to find out how many records there are in total
'in the file.
et filenum, 1, OptionList(1)

'Add one to the number of records in the file to reflect the fact that one
 more is going to be added
ptionList(1).Recordlen = OptionList(1).Recordlen + 1
Put filenum, 1, OptionList(1)

osition = OptionList(1).Recordlen
'Redimension array to hold all records

eDim Preserve OptionList(Position)
For i = 1 To Position
et filenum, i, OptionList(i)
ext
```

```
OptionList(OptionList(1).Recordlen).OptionName = NextOldoption

'Put data back into the file
For i = 1 To Position
Put filenum, i, OptionList(i)
Next

Close filenum

End Sub

Sub Command3_Click ()
Dim OptionSelected As String
Dim Msg As String
Dim Reply As Integer
Dim Title As String
Dim Typ As Integer
Dim Request As String
Dim Answer As String
Dim Thisstate As OldOptions
Dim OptionList() As OldOptions

OldOptions.Visible = True

'Clear Existing information from the listbox
OldOptions.Clear



'Load all the option captions into the listbox
recordlength = Len(Thisstate)
filenum = FreeFile
Open "c:\Factory\System\" & ThisIndex & ".Old" For Random Access Read Shared As
filenum Len = recordlength

'Redimension array to get first record which shows how many records there are
'in total in the file.
ReDim OptionList(1)

Get filenum, 1, OptionList(1)

'If there is nothing in the file
If OptionList(1).Recordlen = 0 Then
   Msg = "There are no existing options currently in existence. In order for "
   Msg = Msg & "there to be any you would to have created new options under "
   Msg = Msg & "the 'Define New Options button. The options that are replaced"
   Msg = Msg & " in the 'Options available' are saved to the *.old files in the
"
   Msg = Msg & "'System Directory'"
   Title = "Old Options Status"
   Typ = MB_ICONINFORMATION + MB_OK
   Reply = MsgBox(Msg, Typ, Title)
Exit Sub
```

```
nd If


Redimension the array so as to hold all the old options
eDim Preserve OptionList(OptionList(1).Recordlen)

or i = 1 To OptionList(1).Recordlen
Get each option from file and add to old option listbox
Get filenum, i, OptionList(i)
ldOptions.AddItem OptionList(i).OptionName
ext

nd Sub

ub Form_Load ()
im OptionSelected As String
im Msg As String
im Reply As Integer
im Title As String
im Typ As Integer
im Request As String
im Answer As String
im Thisstate As Optionstate

ption1.Value = True

Load the current option button option state for the currently loaded
process type.

ecordlength = Len(Thisstate)
ilenum = FreeFile
pen "c:\Factory\System\" & ThisIndex & ".Opt" For Random Access Read Shared As
ilenum Len = recordlength

or i = 1 To 4
et filenum, i, OptionStateArray(i)
ext

ption1.Caption = OptionStateArray(1).OptionCaption
ption2.Caption = OptionStateArray(2).OptionCaption
ption3.Caption = OptionStateArray(3).OptionCaption
ption4.Caption = OptionStateArray(4).OptionCaption

lose filenum

nd Sub

ub OldOptions_Click ()

When an item from this list is clicked with the mouse the item clicked is
sed to replace the option currently selected
Declare variables to be used in this procedure
im OptionSelected As String
im Msg As String
m Reply As Integer
im Title As String
```

```
Dim Typ As Integer
Dim Request As String
Dim Answer As String
Dim Thisstate As Optionstate
Dim OldOption As String
Dim Thatstate As OldOptions
Dim OptionList() As OldOptions
Dim NextOldoption As String
Dim Position As Integer

'Ask the user whether or not he would like to modify an option button
Msg = "The option button as selected will be replaced by the option just"
Msg = Msg & " selected. Do you wish to continue?"
Title = "Modify Option Button"
Typ = MB_YESNO + MB_ICONQUESTION
Reply = MsgBox(Msg, Typ, Title)

'Check to see if the answer was cancel or ok
If Reply = IDNO Then
   Exit Sub 'Exit this procedure if the user doesn't want to continue
End If

'Define a generic optionbutton array
Dim ThisOption As OptionButton

'Find out which option button was selected by interrogating the array
Set ThisOption = Option1
If ThisOption.Value = True Then 'This option is selected
OptionSelected = "Option1"
   Else
   Set ThisOption = Option2
   If ThisOption.Value = True Then 'This option is selected
   OptionSelected = "Option2"
         Else
         Set ThisOption = Option3
         If ThisOption.Value = True Then 'This option is selected
         OptionSelected = "Option3"
         Else
            Set ThisOption = Option4
            If ThisOption.Value = True Then 'This option is selected
            OptionSelected = "Option4"
            End If
         End If
   End If
End If


'Ask the user to input the name of the option that he wants to use to replace
'the one selected

Select Case OptionSelected

Case Is = "Option1"
NextOldoption = Trim(Option1.Caption)
Option1.Caption = OldOptions.List(OldOptions.ListIndex)
```

```
 ase Is = "Option2"
 extOldoption = Trim(Option2.Caption)
Option2.Caption = OldOptions.List(OldOptions.ListIndex)

 ase Is = "Option3"
NextOldoption = Trim(Option3.Caption)
Option3.Caption = OldOptions.List(OldOptions.ListIndex)

Case Is = "Option4"
NextOldoption = Trim(Option4.Caption)
 ption4.Caption = OldOptions.List(OldOptions.ListIndex)
End Select


 Now we must save the changes to file so that when this form is reloaded
'the new option is retained.
 ecordlength = Len(Thisstate)
 ilenum = FreeFile
Open "c:\Factory\System\" & ThisIndex & ".Opt" For Random Access Write Shared As
 filenum Len = recordlength
 Assign captions to file

 ptionStateArray(1).OptionCaption = Option1.Caption
 ptionStateArray(2).OptionCaption = Option2.Caption
OptionStateArray(3).OptionCaption = Option3.Caption
 ptionStateArray(4).OptionCaption = Option4.Caption

'Put this data into file
For i = 1 To 4
 ut filenum, i, OptionStateArray(i)
Next

 Close the file
Close filenum

 open file which recalls all old options used
Filenum = FreeFile
recordlength = Len(Thatstate)

ReDim OptionList(1)
Open "c:\Factory\System\" & ThisIndex & ".old" For Random Access Read Write Shar
 d As filenum Len = recordlength

'Redimension record to 1 unit to find out how many records there are in total
 in the file.
Get filenum, 1, OptionList(1)

 Add one to the number of records in the file to reflect the fact that one
 more is going to be added
OptionList(1).Recordlen = OptionList(1).Recordlen + 1
 ut filenum, 1, OptionList(1)

Position = OptionList(1).Recordlen
 Redimension array to hold all records

ReDim Preserve OptionList(Position)
```

```
For i = 1 To Position
Get filenum, i, OptionList(i)
Next

'Check to see whether or not current 'OldOption' is already in this file - if
'it is then do not save the current 'OldOption" to file
For i = 1 To (Position - 1)
If Trim(NextOldoption) = Trim(OptionList(i).OptionName) Then
OptionList(1).Recordlen = OptionList(1).Recordlen - 1
Put filenum, 1, OptionList(1)
Close filenum
Exit Sub
End If
Next

OldOptions.AddItem NextOldoption

OptionList(OptionList(1).Recordlen).OptionName = NextOldoption

'Put data back into the file
For i = 1 To Position
Put filenum, i, OptionList(i)
Next

Close filenum




End Sub

Sub Picture4_Click ()

Dim ThisTime As Variant
Dim ThisPicture As String

'When this image is clicked we want to show the downbutton image before we
'do anything else

'Show the button down position as the button has been pushed
ThisPicture = "C:\Factory\System\icons\hamdwn.ico"
Picture4.Picture = LoadPicture(ThisPicture)

'Put a delay in before the button goes back up
Call mnu_timer(ThisTime)

'Show the button up position as the button has returned
ThisPicture = "C:\Factory\System\icons\hammer.ico"
Picture4.Picture = LoadPicture(ThisPicture)
```

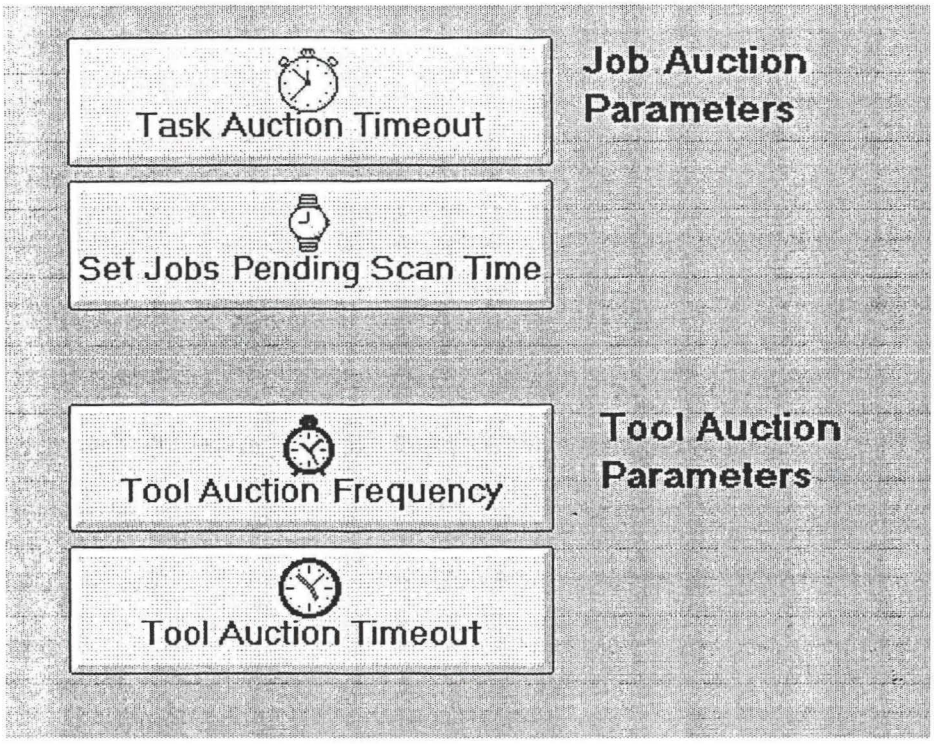
```
'Check to see what option was selected
If Option1.Value = True Then
   ActiveOption = Option1.Caption
   Else
     If Option2.Value = True Then
     ActiveOption = Option2.Caption
     Else
       If Option3.Value = True Then
       ActiveOption = Option3.Caption
       Else
         ActiveOption = Option4.Caption
       End If
     End If
End If

'Load the form which has a listbox allowing the user to configure the tools
'for his factory

Toollists.Show

'set the caption equal to the tool type being configured
Toollists.Caption = Trim(ActiveOption) & " " & StateTools.Caption

End Sub
```


Task Auction Timeout
Job Auction Parameters
Set Jobs Pending Scan Time
Tool Auction Frequency
Tool Auction Parameters
Tool Auction Timeout

```
SYSTEM_F.FRM - 1

Option Explicit

Sub Auct_Out_Click ()
Dim Msg As String
Dim Title As String
Dim typ As Integer

Msg = "The Factory Controller's 'View Tool Availability' button accesses a form
that shows the "
Msg = Msg & "current availability of all tools in the Factory. The Factory Cont
roller is "
Msg = Msg & "responsible for the auctioning of tools off the tool availability
status table in "
Msg = Msg & " this form as they become available. The 'Tool Auction Timeout' but
ton allows the "
Msg = Msg & "user the ability to set the time delay between the distribution of
the tool bidding "
Msg = Msg & "objects and the assessment of the bids recieved by the cells in the
factory."

typ = MB_ICONINFORMATION + MB_OK
Title = "Tool Auction Timeout Information"

MsgBox Msg, typ, Title

End Sub

Sub Auct_Tim_Click ()
Dim Msg As String
Dim Title As String
Dim typ As Integer

Msg = "The Auction Timeout is the time difference between the time from which th
e Factory "
Msg = Msg & "Controller distributes the Bidding Objects for a task being auction
ed and the "
Msg = Msg & " time it starts to evaluate the submitted bids from the Cell Contro
ller processes."
Msg = Msg & " Any bids submitted after the timeout will not be considered."

typ = MB_ICONINFORMATION + MB_OK
Title = "Auction Timeout Information"

MsgBox Msg, typ, Title


End Sub

Sub Auct_Time_Click ()
Dim Msg As String
Dim Title As String
Dim Defval As String
Dim Reply As String
Dim ThisFilesname As String
Dim Thisfileslength As Integer
```

```
Msg = "Please enter the desired Auction timeout value in seconds."
Defval = "3"
Title = "Enter Auction Timeout"

Reply = InputBox$(Msg, Title, Defval)

If Len(Trim(Reply)) = 0 Then
    MsgBox ("Nothing was added")
    Exit Sub
End If

If Val(Reply) = 0 Then 'something stupid has been entered so force it to the de
ault value
    Reply = Defval
End If

Thisfileslength = 3
ThisFilesname = "C:\Factory\System\Config\Aucttime.sys"

Call Output_To_File(ThisFilesname, Thisfileslength, Reply)



End Sub

Sub Jobs_pend_Click ()
Dim Msg As String
Dim Title As String
Dim typ As Integer

Msg = "The Factory Controller's 'Jobs Pending' queue contains the names of the
obs waiting to "
Msg = Msg & "be scheduled. They are scheduled by scanning their 'Job Start Date
' and comparing "
Msg = Msg & " their start date to the computer system date - when the dates are
the same the "
Msg = Msg & " Factory Controller puts the jobs that have agreeing start dates o
 the Factory "
Msg = Msg & "Controller's outgoing queue for immediate auction. The button in th
is window allows "
Msg = Msg & "the user to set the frequency(in minutes) by which the jobs in the
pending queue have their "
Msg = Msg & "due dates scanned."



typ = MB_ICONINFORMATION + MB_OK
Title = "Jobs Pending Scan Time Information"

MsgBox Msg, typ, Title

End Sub

Sub Scan_Time_Click ()
```

```
SYSTEM_F.FRM - 3

Dim Msg As String
Dim Title As String
Dim Defval As String
Dim Reply As String
Dim Filenum As Integer
Dim ThisFilesname As String
Dim Thisfileslength As Integer


Msg = "Please enter the desired Scan Time(in multiples of 10 seconds) for the j
bs in the 'Jobs Pending Queue'."
Defval = "2"
Title = "Enter Auction Timeout"

Reply = InputBox(Msg, Title, Defval)

If Len(Trim(Reply)) = 0 Then 'the cancel button was pushed so the user dosnt wa
t to do anything
    Exit Sub
End If

If Val(Reply) = 0 Then 'something stupid has been entered so force it to the de
fault value
    Reply = Defval
End If

ThisFilesname = "C:\Factory\System\Config\Scantime.sys"
Thisfileslength = 4

Call Output_To_File(ThisFilesname, Thisfileslength, Reply)

End Sub

Sub Tool_Auct_Click ()
Dim Msg As String
Dim Title As String
Dim Defval As String
Dim Reply As String
Dim Filenum As Integer
Dim ThisFilesname As String
Dim Thisfileslength As Integer

Msg = "Please enter the desired Tool Auction Frequency(in seconds)"
Defval = "5"
Title = "Enter Tool Auction Frequency"

Reply = InputBox(Msg, Title, Defval)

If Len(Trim(Reply)) = 0 Then 'the cancel button was pushed so the user dosnt wan
t to do anything
    Exit Sub
End If

If Val(Reply) = 0 Then 'something stupid has been entered so force it to the def
ault value
    Reply = Defval
```

```
End If

ThisFilesname = "C:\Factory\System\Config\ToolFreq.sys"
Thisfileslength = 4

Call Output_To_File(ThisFilesname, Thisfileslength, Reply)

End Sub

Sub Tool_AuctOut_Click ()
Dim Msg As String
Dim Title As String
Dim Defval As String
Dim Reply As String
Dim Filenum As Integer
Dim ThisFilesname As String
Dim Thisfileslength As Integer


Msg = "Please enter the desired Tool Auction Timeout(in seconds)"
Defval = "5"
Title = "Enter Tool Auction Timeout"

Reply = InputBox(Msg, Title, Defval)

If Len(Trim(Reply)) = 0 Then 'the cancel button was pushed so the user dosnt wa
t to do anything
    Exit Sub
End If

If Val(Reply) = 0 Then 'something stupid has been entered so force it to the def
ault value
    Reply = Defval
End If

ThisFilesname = "C:\Factory\System\Config\TlActOut.sys"
Thisfileslength = 4

Call Output_To_File(ThisFilesname, Thisfileslength, Reply)

End Sub

Sub ToolAuct_Click ()
Dim Msg As String
Dim Title As String
Dim typ As Integer

Msg = "The Factory Controller's 'View Tool Availability' button accesses a form
that shows the "
Msg = Msg & "current availability of all tools in the Factory. The Factory Contr
oller is "
Msg = Msg & "responsible for the auctioning of tools off the tool availability
tatus table in "
Msg = Msg & " this form as they become available. The 'Tool Auction Frequency' b
utton allows the "
Msg = Msg & "user the ability to set the frequency that this auctioning process
```

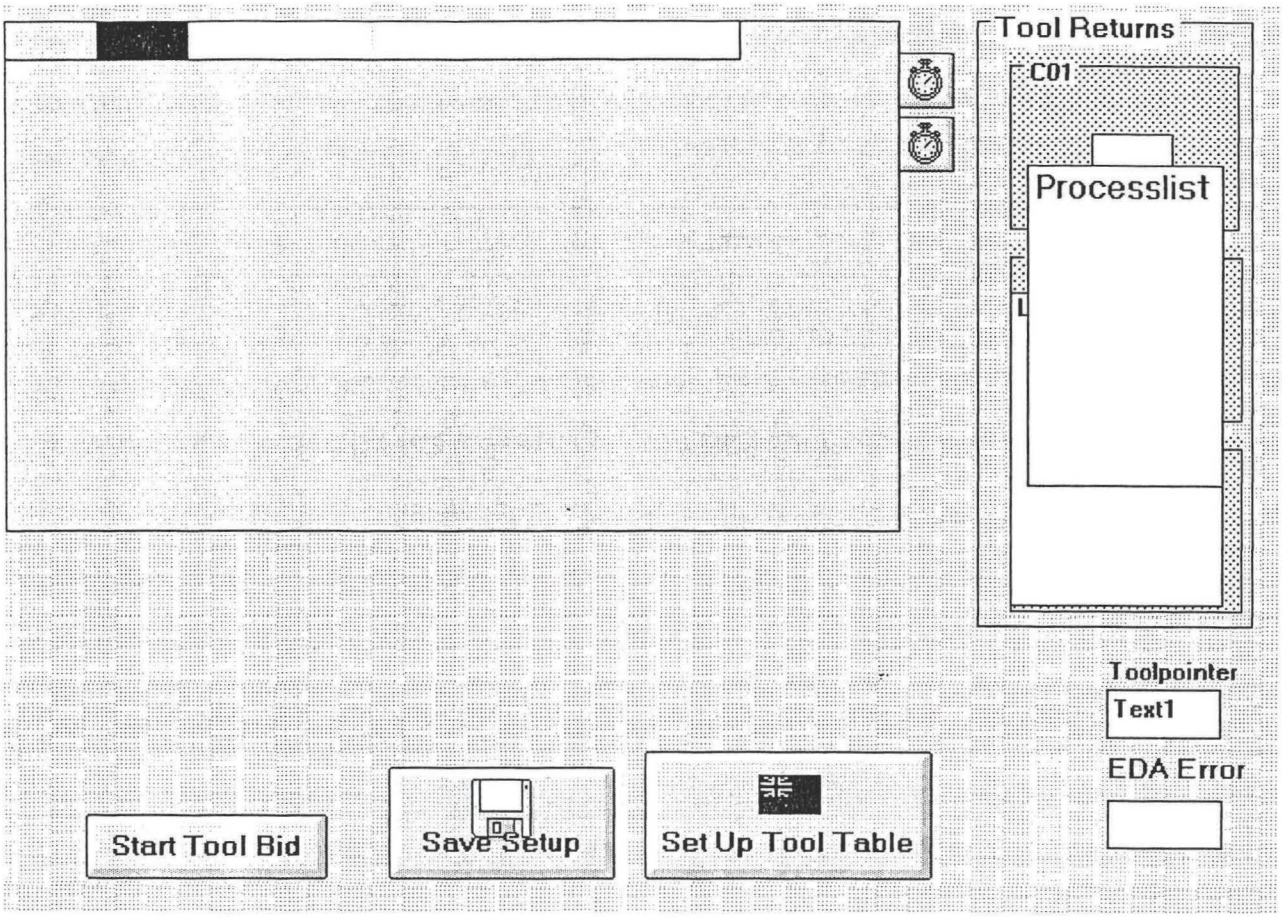
```
SYSTEM_F.FRM - 5

occurs in seconds."

typ = MB_ICONINFORMATION + MB_OK
Title = "Tool Auction Frequency Information"

MsgBox Msg, typ, Title

End Sub
```


Tool Returns
C01
Processlist
Toolpointer
Text1
EDA Error
Start Tool Bid
Save Setup
Set Up Tool Table

```
'OOL_AVL.FRM - 1

Option Explicit
Option Base 1
Option Compare Text

Dim Template_Defined As Integer
Dim NewSetUpDefined As Integer

Dim Ghandler As Long
Dim fart As Variant
Dim Node As String * 15
Dim eda_err As Variant
Dim tex As Integer
Dim Newnum As Variant
Dim Tags As String * 15
Dim Field As String * 15
Dim answer As String * 80
Dim ans As String * 10
Dim defnode As Long
Dim trap As String * 10
Dim Texts As String * 6
Dim Nodetyp As String * 20
Dim nodes As String * 20
Dim bidrotenable As Variant
Dim BlockTyp As String * 20
Dim Bidoperatestate As Variant
Dim Bidnodetext As String * 10
Dim bidtokenstate As Variant
Dim Tokenstate As Integer
Dim BidOpstate As Variant
Dim X As Integer
Dim Dummy As Variant
Dim ToolPointer As Integer



' functions specific to VB (not standard eda fcns)
Declare Sub eda_version Lib "vdba.dll" (ByVal v$)
Declare Sub eda_get1_ascii Lib "vdba.dll" (ByVal n$, ByVal t$, ByVal f$, ByVal v

Declare Function eda_set1_ascii% Lib "vdba.dll" (ByVal n$, ByVal t$, ByVal f$, B
yVal v$)

' Note define_group returns a long

Declare Function eda_define_group& Lib "vdba.dll" (ByVal Count%, ByVal Detect%)
Declare Sub eda_delete_group Lib "vdba.dll" (ByVal Handle&)

' should always pass 0 for vsp
Declare Function eda_define_ntf& Lib "vdba.dll" (ByVal Handle&, ByVal enode$, By
Val etag$, ByVal efield$, ByVal vsp&)
Declare Sub eda_delete_ntf Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)

Declare Sub eda_lookup Lib "vdba.dll" (ByVal Handle&)
```

```
Declare Sub eda_read Lib "vdba.dll" (ByVal Handle&)
Declare Sub eda_write Lib "vdba.dll" (ByVal Handle&)
Declare Function eda_write1% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
Declare Sub eda_wait Lib "vdba.dll" (ByVal Handle&)
Declare Function eda_query_wait% Lib "vdba.dll" (ByVal Handle&)
'
Declare Function eda_get_error% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&)
Declare Function eda_get_ascii% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, ByV
 Value$, ByVal maxlen%)
Declare Function eda_set_ascii% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, ByV
 Value$)

' note pointer to float is passed
Declare Function eda_get_float% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, Val
!)
Declare Function eda_set_float% Lib "vdba.dll" (ByVal Handle&, ByVal ntf&, Valu
!)



'define variables that are used to store details of the tool currently auction
g in memory
Dim Toollocation As String
Dim ToolTypeCode As String
Dim ToolStatus(4) As String * 30

Sub Command2_Click ()
Bid_Eval.Enabled = True
End Sub

Sub CommDDElink (Text As TextBox, LinkItem)
'This Sub Procedure creates an Automatic link between the
'Queueing application and either the Factory Controller or
'the Cell Controller

Text.LinkMode = none  'If the link is not already
                      'created then create it
Text.LinkTopic = "DMDDE|DATA"  'Link topic is FixDmacs

Text.LinkItem = LinkItem       'This is the Node,
                               'databaseblock and field we
                               'want to link to

Text.LinkMode = Link_Automatic 'Automatic update of data


End Sub

Function eda_requestinfolink (n, t, f)
Ghandler = eda_define_group(1, 0)
defnode = eda_define_ntf(Ghandler, n, t, f, 0)

Call eda_lookup(Ghandler)
Call eda_wait(Ghandler)
Call eda_read(Ghandler)
Call eda_wait(Ghandler)
```

```
Newnum = eda_get_ascii(Ghandler, defnode, answer, 50)
Call eda_delete_group(Ghandler)
eda_requestinfolink = answer
End Function

Sub sendinfo (n, t, f, Text)
Ghandler = eda_define_group(1, 0)
defnode = eda_define_ntf(Ghandler, n, t, f, 0)

Call eda_lookup(Ghandler)
Call eda_wait(Ghandler)
eda_err = eda_get_error(Ghandler, defnode)
Text5.Text = Str$(eda_err)
Newnum = Text
Part = eda_set_ascii(Ghandler, defnode, Newnum)
Call eda_write(Ghandler)
Call eda_wait(Ghandler)
Call eda_delete_group(Ghandler)

End Sub

Sub UpdateToolList (ToolCellNam As String, ThisToolCode As Single)

  GoTo skipitall
'declare variables
Dim i As Integer
Dim NumberOfRecords As Integer
Dim Tool_Avlearray() As Tool_AvleRec
Dim Thistoolsrec As Tool_AvleRec
Dim Nextelement As Integer
Dim recordlength As Integer
Dim Matchfound As Integer
Dim Filenum As Integer

'open up the file that holds all the tool status information and write this inf
rmation to
'memory


'define a freefile
Filenum = FreeFile

'get the variable storage length
recordlength = Len(Thistoolsrec)

'open the required file
Open "C:\Factory\Config\Tools\Toolrecd.tl" For Random Access Read Write Shared
  Filenum Len = recordlength

'redimension the array to take the first record only so we can get the length o
the file required
to be read.
ReDim Tool_Avlearray(1)

'get the first record in the array so that we are able to see how long the tota
file is.
```

```
Get Filenum, 1, Tool_Avlearray(1)

NumberOfRecords = Tool_Avlearray(1).NumberOfRecords

'redimension the array to take all the records
ReDim Tool_Avlearray(NumberOfRecords)

For i = 1 To NumberOfRecords  'for each record in the file get it and read it i
 memory
    Get Filenum, i, Tool_Avlearray(i)'get the contents of the associated record
Next

'Type Tool_AvleRec
    'NumberOfRecords As Integer
    'ToolType As String * 30
    'ProcessType As String * 30
    'Availability As String * 15
    'Location As String * 10
    'LifeUsed As Integer
    'Toolsize As Integer
    'Reserved As String * 10
    'Toolcode As Single
'End Type



'now that we have the records in memory we must look for the match between the
'tool that has finished being used and the cell used in and the tool in
'the file

For i = 1 To NumberOfRecords
    If Tool_Avlearray(i).Toolcode = ThisToolCode And Trim(ToolCellNam) = Trim(T
ol_Avlearray(i).Location) Then
        'a match has been found between a tool that is no longer needed on this
vmd and a tool
        'that was given to this vmd and is marked busy - this means that the too
l is the same
        'because only one VMD in a cell can hold a tool of a certain type becau
e each process
        'within a cell is different.

        'therefore go to that position in the array and mark it as being idle
   '     MsgBox ("Tool Match Code is " & CStr(Tool_Avlearray(i).Toolcode))
        Tool_Avlearray(i).Availability = "IDLE"
        Tool_Avlearray(i).Reserved = ""
        Matchfound = 1
   End If
Next

'now put all the information back in the file

For i = 1 To NumberOfRecords
    Put Filenum, i, Tool_Avlearray(i)
Next
```

```
.FRM - 5

se the file                                                          nfigure
ilenum

nfound = 1 Then
gBox ("It is working")
                                                                     ory and

                                                                     en a bi

eval_Timer ()

'filehandling variables
entstring As String * 20
filesname As String
berOfStrings As Integer
ngNumber As Integer

nodefile handling variables
record As nodelistrecord                                             lenum
tion As Integer

general variables
tarrayposition As Integer
enum As Integer
sFilenum As Integer
rdlength As Integer
s Integer
srecordslength As Integer
oolsavailable As Integer
ultofComp As Integer
IsNull As Integer
rvedStatus As String * 10

an array storage variable
torageArray() As ToolBid
Bid As ToolBid
rentToolLocation As String


hisNodeFinished As String * 10
time As Double
time As Double
pCounter As Integer

count As Integer
                                                                     re ope

set period of time this timer is enabled and then its job is to compare
ds as
re recieved from the external factory
Resume Next


.Enabled = False 'disable timer
                                                                     ous ha
```

```
                                'raise the alarm.
        MsgBox ("Tool Bidding Protocol Corrupted, Reset tool bidding System")
        Close Filenum
        Exit Sub
    End If


    'request bid availability
    Node = "FACT"
    Tags = Trim(Thisrecord.Node) & "TOOLFIN"
    Field = "A_CV"

    IsThisNodeFinished = eda_requestinfolink(Trim(Node), Tags, Field)

'    MsgBox CStr(IsThisNodeFinished)
    If CInt(Trim(IsThisNodeFinished)) = 0 Then 'the cell concerned has not yet f
ormulated a bid so
    'wait 0.5 secs and then try again
        Thistime = Timer

        Do Until (Thattime - Thistime) > .1
            Thattime = Timer
        Loop

        DoEvents 'release execution to windows to allow other things to occur

        GoTo CheckBidAvailability
    End If

    'if the program gets here the bid for the cell concerned has arrived and we
can now reset the
    'associated toolbid block.

    Texts = "0"
    Tags = Trim(Thisrecord.Node) & "TOOLFIN"
    Field = "A_CV"

    Call sendinfo(Node, Tags, Field, Texts)


    For i = 1 To 2
        'define the length of the variable required to store the variable-
        Thisrecordslength = 8

        'define the filename to open
        Thisfilesname = Netdrive & "Cellcont\Toolbid\" & Trim(Thisrecord.Node) &
 "TLBid.Bid"
'       MsgBox (Thisfilesname)

        Currentstring = Input_Strings(Thisfilesname, Thisrecordslength, i)

'       MsgBox (Currentstring)
        If Val(Trim(Currentstring)) = 0 Then
            Countarrayposition = Countarrayposition - 1
            GoTo TheNextStep
        End If
```

```
        ReDim Preserve BidStorageArray(Countarrayposition)

        If i = 1 Then
            BidStorageArray(Countarrayposition).CellBidding = Trim(Thisrecord.N
de)
            BidStorageArray(Countarrayposition).VMDBidding = Trim(Currentstring
                Else
                BidStorageArray(Countarrayposition).StartDateBid = Val(Trim(Cur
entstring))
        End If


    Next


TheNextStep:
Loop

Close Filenum

DoEvents

If Countarrayposition = 0 Then 'then no bids have been submitted
    GoTo Bidnexttool
End If


'GoTo restart
okay this is all very well but we must also open up the Node interarrival time
file that tells
'the program how far it is between cells


DoEvents

Dim TheseNodesTimedif() As NodeTimedelay
Dim TheseNodesRecord As NodeTimedelay
Dim Recordsize, j, S, Nextrecord   As Integer


Filenum = FreeFile
Recordlength = Len(TheseNodesRecord)

'open the file that details the internode travel times.
Open Netdrive & "Factory\Bidding\Nodetime\NodeDely.txt" For Random Access Read S
hared As Filenum Len = recordlength

'firstly we only require the first record to get the matrixsize
ReDim TheseNodesTimedif(1)

'get the first record that stores the value of the matrix size
Get Filenum, 1, TheseNodesTimedif(1)

'redimension to take all the records that are in the file
Recordsize = (TheseNodesTimedif(1).MatrixDim * TheseNodesTimedif(1).MatrixDim)
```

```
ReDim TheseNodesTimedif(Recordsize)

'Type NodeTimedelay
 '    MatrixDim As Integer
  '   Interarrivaltime As Integer
   ' CellnameX As String * 10
    ' CellnameY As String * 10
'End Type

'now get all the records
For i = 1 To Recordsize
    Get Filenum, i, TheseNodesTimedif(i)
    'MsgBox (CStr(Thesenodestimedif(i).CellnameX))
    'MsgBox (CStr(Thesenodestimedif(i).Cellnamey))
    'MsgBox (CStr(Thesenodestimedif(i).Interarrivaltime))
    'List1.AddItem Thesenodestimedif(i).CellnameX
    'List1.AddItem Thesenodestimedif(i).Cellnamey
Next

'close the file
Close Filenum


'but this is not all that we are required to do, we are also required to try to
compare the
'relative distance between the node that has currently got the tool and the nod
 that wants
'the tool and to get the distance - this is stored in a series of variables and
if any other
'cells can give a better bid then the value of the bid, the cellname and the emp
typosition in the
'current task reference file overwrites the current winning values of the stora
e variables.


'tooltypecode

Dim WinningCell As String * 10
Dim WinningVMD As String
Dim WinningBid As Double
Dim Absentposition As Integer
Dim firstbid As Integer


CurrentToolLocation = Trim(Toollocation)



'MsgBox (Currenttoollocation)


For j = 1 To Recordsize

    If j = Int(Recordsize / 2) Or j = Int(Recordsize / 4) Then
```

```
        DoEvents
    End If

    'MsgBox (Currenttoollocation)
    For S = 1 To Countarrayposition
        If S = Countarrayposition Then
            DoEvents
        End If
      ' GoTo restart
        'MsgBox (CStr(s))
       ' MsgBox (CStr(j))
        'MsgBox (Thesenodestimedif(j).CellnameX)
        'MsgBox (Currenttoollocation)
      'MsgBox (Thesenodestimedif(j).Cellnamey)
       ' MsgBox (Bidstoragearray(s).CellBidding)
       If Trim(TheseNodesTimedif(j).CellnameX) = Trim(CurrentToolLocation) And T
rim(TheseNodesTimedif(j).Cellnamey) = Trim(BidStorageArray(S).CellBidding) Then
        '    MsgBox (CStr(j))
            'a match has been found between the auctioning cell and the bidding
ell therefore
            'get the distance between them and store the distance and other data
 in a set of
            'variables.
            'firstbid = 0
            BidStorageArray(S).TotalBid = CDbl(TheseNodesTimedif(j).Interarrival
ime) / 60 + CDbl(BidStorageArray(S).StartDateBid)
       End If
    Next
Next

oEvents
'ellBidding As String * 10
VMDBidding As String * 10
StartDateBid As Single
'otalBid As Double


'e will put the results of the bids next
or S = 1 To Countarrayposition
    If "C01" Like Trim(BidStorageArray(S).CellBidding) Then
        Res_Form.Res_Cell1Bid.AddItem CStr(BidStorageArray(S).TotalBid)
        Res_Form.c01vbidding.AddItem Trim(BidStorageArray(S).VMDBidding)
        GoTo TryCell2
    End If
ext
es_Form.Res_Cell1Bid.AddItem "0"
es_Form.c01vbidding.AddItem "0"

ryCell2:


e will put the results of the bids next
or S = 1 To Countarrayposition
    If "C02" Like Trim(BidStorageArray(S).CellBidding) Then
        Res_Form.Res_Cell2Bid.AddItem CStr(BidStorageArray(S).TotalBid)
        Res_Form.c02vbidding.AddItem Trim(BidStorageArray(S).VMDBidding)
```

```
        GoTo TryCell3
    End If
Next
Res_Form.Res_Cell2Bid.AddItem "0"
Res_Form.c02vbidding.AddItem "0"

TryCell3:

'we will put the results of the bids next
For S = 1 To Countarrayposition
    If "C03" Like Trim(BidStorageArray(S).CellBidding) Then
        Res_Form.Res_Cell3Bid.AddItem CStr(BidStorageArray(S).TotalBid)
        Res_Form.c03vbidding.AddItem Trim(BidStorageArray(S).VMDBidding)
        GoTo Finished
    End If
Next
Res_Form.Res_Cell3Bid.AddItem "0"
Res_Form.c03vbidding.AddItem "0"
Finished:


For S = 1 To Countarrayposition 'compare the valid bids

    If S = 1 Then
        WinningBid = BidStorageArray(1).TotalBid
        WinningVMD = Trim(BidStorageArray(1).VMDBidding)
        WinningCell = Trim(BidStorageArray(1).CellBidding)
        'Call UpdateRes(WinningCell, WinningVMD, WinningBid)
        Else
            If BidStorageArray(S).TotalBid < BidStorageArray(S - 1).TotalBid The
n 'if this bid is better than the old one
                WinningBid = BidStorageArray(S).TotalBid
                WinningVMD = Trim(BidStorageArray(S).VMDBidding)
                WinningCell = Trim(BidStorageArray(S).CellBidding)
                'Call UpdateRes
            End If
    End If
Next



DoEvents


'by the time we get here we will have variables stating what vmd on what cell ha
s won the auction
'so we will need to change the tool_Availability table to reflect this change.

'therefore lets make the changes to the table
Tool_Availability.Row = ToolPointer
'GoTo repeatit
Tool_Availability.Col = 2

'now change the status of the tool to reflect the fact that it has been reservec
Tool_Availability.Text = "BUSY"
```

```
TOOL_AVL.FRM - 12

'change the "Col" property to reflect the fact that we are required to change th
e reserved property
'of a particular tool
Tool_Availability.Col = 6

'put the name of the winning cell in the reserved column in the table
Tool_Availability.Text = Trim(WinningCell)

'put the result in the result table

'put results of the positive auction in the results form

'Res_Form.Res_ToolTime.AddItem Time$ 'first put time
Res_Form.Res_Cell.AddItem CurrentToolLocation
Res_Form.Res_Cell_Win.AddItem Trim(WinningCell)
Res_Form.Res_Win_VMD.AddItem Trim(WinningVMD)
Res_Form.Res_ToolType.AddItem Trim(ToolTypeCode)


'put the name of the winning cell in the location column to simulate the fact th
at the tool has
'now been shifted to the winning cell position
'Tool_Availability.Col = 3

'put the name of the winning cell in the location column in the table
'Tool_Availability.Text = Trim(WinningCell)

'enable the button so that we can save the changes to file
Setup_Save.Enabled = True

'save the changes to file
Setup_Save_Click

'disable the button once more so that the user cannot change the system
Setup_Save.Enabled = False


'open the network file that stores the reserved status of the tools that are req
uired
'for the current task that has won the right to use the auctioned tool


'open the file for input
'Call openForInput(filenum, Thisfilesname, Thisrecordslength)

'now lets get the data from the file
'For j = 1 To 4
  ''  ToolStatus(j) = ""
    ''Line Input #filenum, ToolStatus(j)
    'MsgBox ("data is " & ToolStatus(j))
''Next

'Close filenum
```

```
'MsgBox (Len(Toolstatus(1)))

'now that we have loaded the reserved status of the tools for the task we must
ake the required
'modifications and save those modifications to file.
ReservedStatus = "Reserved"

'now output the modified information back to file so that the bid formulation a
plication can
'tell if this Cell Controller\VMD combination has all tools allocated for the c
rrently queued
'task - if it has the Bid Formulation application will not indicate that this V
D\Cell combo
'will bid.


'define a freefile handler
Filenum = FreeFile


'define the filename and directory
Thisfilesname = Netdrive & "Cellcont\Toolbid\" & Trim(WinningCell) & "V" & Trim
CStr(WinningVMD)) & "Rst.res"

'define the length of the record required to be stored
Thisrecordslength = 10

'open the sequential access file for output mode
Call openForOutput(Filenum, Thisrecordslength, Thisfilesname)

'now put the records back into the file
'''For j = 1 To 4
'    MsgBox ("Tool Status is " & ToolStatus(j))
Print #Filenum, ReservedStatus
 '   MsgBox CStr(Absentposition)
'''Next

'close the file
Close Filenum

'MsgBox (Thisfilesname)
'now check to see whether or not all tools have been reserved for the particular
 task on top of
'the winning cell\VMD combinations queue - if they have prompt the Cell Controll
er\VMD to tell it
'that all tools required have been reserved for it and that it can start the job
 on top of the
 queue when it able to do it.

 'For j = 1 To 4
   ' MsgBox (ToolStatus(j))
 ''    If Trim(ToolStatus(j)) <> "Reserved" Then 'further bidding is required by
 his cell\vmd combo
   ''      AllToolsavailable = 1 ' in order to secure the remaining tools
  '  End If
 Next
```

```
MsgBox (CStr(AllToolsavailable))
''If AllToolsavailable <> 1 Then 'then tell the Cell\VMD combo that it can star
the manufacture
    'of the task on top of the queue when it is ready.

'       'set the node variable as the winning cell of the auction
ode = Trim(WinningCell)
        'set the blocktyp as the winning VMD's blocktype
lockTyp = "V" & Trim(WinningVMD) & "Toolresd"
ield = "A_CV"
exts = "1"
end the information to the appropriate block in the approporiate VMD
ll sendinfo(Node, BlockTyp, Field, Texts)
'End If



idnexttool:

Check to see if the toolpointer has reached the end of the list -if it has rese
 the
toolpointer to "1" so that the bidding algorithm goes back to the beginning of
he list and
tries to auction a tool off the beginning again.
 ToolPointer = (Tool_Availability.Rows - 1) Then 'then reset the tool pointer
ariable
    ToolPointer = 1
    Else 'otherwise increment the value of the toolpointer
        ToolPointer = ToolPointer + 1
d If


oTo repeatit
e-enable the timer so that it can respond again and try and auction another it
.
ol_Timer.Enabled = True

d Sub

ub C01v1ret_Change ()

 Val(C01v1ret.Text) = 0 Then
    Exit Sub
    Else
    Call UpdateToolList("C01", CSng(C01v1ret.Text))

    'now rest the block concerned
    Node = "Fact"
    BlockTyp = "C01V1TOOL"
    'MsgBox (Tooltypecode)
    Field = "A_CV"
    Texts = "0"
```

```
    Call sendinfo(Node, BlockTyp, Field, Texts)
End If


End Sub

Sub C01v2ret_Change ()

If Val(C01v2ret.Text) = 0 Then
    Exit Sub
    Else
    Call UpdateToolList("C01", CSng(C01v2ret.Text))

    'now rest the block concerned
    Node = "Fact"
    BlockTyp = "C01V2TOOL"
    'MsgBox (Tooltypecode)
    Field = "A_CV"
    Texts = "0"

    Call sendinfo(Node, BlockTyp, Field, Texts)
End If


End Sub

Sub c01v4ret_Change ()

If Val(C01v4ret.Text) = 0 Then
    Exit Sub
    Else
    Call UpdateToolList("C01", CSng(C01v4ret.Text))

    'now rest the block concerned
    Node = "Fact"
    BlockTyp = "C01V4TOOL"
    'MsgBox (Tooltypecode)
    Field = "A_CV"
    Texts = "0"

    Call sendinfo(Node, BlockTyp, Field, Texts)
End If

End Sub

Sub c02v1ret_Change ()

If Val(C02v1ret.Text) = 0 Then
    Exit Sub
    Else
    Call UpdateToolList("C02", CSng(C02v1ret.Text))

    'now rest the block concerned
    Node = "Fact"
    BlockTyp = "C02V1TOOL"
```

```
    'MsgBox (Tooltypecode)
    Field = "A_CV"
    Texts = "0"

    Call sendinfo(Node, BlockTyp, Field, Texts)

End If

End Sub

Sub c02v2ret_Change ()

If Val(C02v2ret.Text) = 0 Then
    Exit Sub
    Else

    Call UpdateToolList("C02", CSng(C02v2ret.Text))

    'now rest the block concerned
    Node = "Fact"
    BlockTyp = "C02V2TOOL"
    'MsgBox (Tooltypecode)
    Field = "A_CV"
    Texts = "0"

    Call sendinfo(Node, BlockTyp, Field, Texts)

End If

End Sub

Sub c02v4ret_Change ()

If Val(C02v4ret.Text) = 0 Then
    Exit Sub
    Else
    Call UpdateToolList("C02", CSng(C02v4ret.Text))

    'now rest the block concerned
    Node = "Fact"
    BlockTyp = "C02V4TOOL"
    'MsgBox (Tooltypecode)
    Field = "A_CV"
    Texts = "0"

    Call sendinfo(Node, BlockTyp, Field, Texts)

End If

End Sub

Sub c03v1ret_Change ()

If Val(C03v1ret.Text) = 0 Then
    Exit Sub
    Else
```

```
    Call UpdateToolList("C03", CSng(C03v1ret.Text))

    'now rest the block concerned
    Node = "Fact"
    BlockTyp = "C03V1TOOL"
    'MsgBox (Tooltypecode)
    Field = "A_CV"
    Texts = "0"

    Call sendinfo(Node, BlockTyp, Field, Texts)

End If

End Sub

Sub c03v2ret_Change ()

If Val(C03v2ret.Text) = 0 Then
    Exit Sub
    Else
    Call UpdateToolList("C03", CSng(C03v2ret.Text))

    'now rest the block concerned
    Node = "Fact"
    BlockTyp = "C03V2TOOL"
    'MsgBox (Tooltypecode)
    Field = "A_CV"
    Texts = "0"

    Call sendinfo(Node, BlockTyp, Field, Texts)

End If

End Sub

Sub c03v4ret_Change ()

If Val(C03v4ret.Text) = 0 Then
    Exit Sub
    Else
    Call UpdateToolList("C03", CSng(C03v4ret.Text))

    'now rest the block concerned
    Node = "Fact"
    BlockTyp = "C03V4TOOL"
    'MsgBox (Tooltypecode)
    Field = "A_CV"
    Texts = "0"

    Call sendinfo(Node, BlockTyp, Field, Texts)

End If

End Sub

Sub Command1_Click ()
```

```
)L_AVL.FRM - 18

-)l_Timer.Enabled = True

nd Sub

 ) Form_Load ()
im Msg As String
im Typ As Integer
 m Title As String
im Reply As Integer
im ProcessSpec() As Processdef
 m Processtyp As Processdef
 m Filelength As Integer
im i As Integer
 m Filenum As Integer


rocessNumber As Integer
rocessname As String * 20
CurrentProcessId As Integer
rocessId As Integer
 axcap As Integer
nd Type

llocate a number to the file to be opened
:lenum = FreeFile

ree up memory for the file to be opened
ilelength = Len(Processtyp)

edimension storage array to take first record
eDim ProcessSpec(1)

'open the process list file
pen "c:\Factory\Process\files\Proclist.Pfl" For Random Access Read Write Shared
As Filenum Len = Filelength

Get the first record into memory
Get Filenum, 1, ProcessSpec(1)

f ProcessSpec(1).Processnumber = 0 Then
    Title = "File Empty"
    Msg = "There are no processes configured for the system yet - this will have
to be done."
    Typ = MB_ICONEXCLAMATION + MB_OK
    MsgBox Msg, Typ, Title
    Close Filenum
    Exit Sub
End If

Redimension the array to all the process names in the file.
ReDim Preserve ProcessSpec(ProcessSpec(1).Processnumber)

or i = 1 To ProcessSpec(1).Processnumber
 'Get each process name from the file and add each processname to the list
    Get Filenum, i, ProcessSpec(i)
'add each item in the array to the hidden process list
```

```
    Processlist.AddItem Trim(ProcessSpec(i).Processname)
Next

'Close the process file
Close Filenum


'now set up the columns and rows in the grid
For i = 1 To 8
    'make the columns the maximum width required to fill the grid border width
   Tool_Availability.ColWidth(i - 1) = Tool_Availability.Width / 8
Next

'set the row height of the first row.
Tool_Availability.RowHeight(0) = 500

'set up the headings for the grid

Tool_Availability.Col = 0 'define the cell
Tool_Availability.Row = 0  'for the first lot of text to be placed in

Tool_Availability.Text = "Tool Type"

Tool_Availability.Col = 1
Tool_Availability.Text = "Process Type"

Tool_Availability.Col = 2
Tool_Availability.Text = "Availability"

Tool_Availability.Col = 3
Tool_Availability.Text = "Location"

Tool_Availability.Col = 4
Tool_Availability.Text = "% Life Used"

Tool_Availability.Col = 5
Tool_Availability.Text = "Size(mm)"

Tool_Availability.Col = 6
Tool_Availability.Text = "Reserved"

Tool_Availability.Col = 7
Tool_Availability.Text = "Tool Code"

'now we are required to open the files that contain the timer intervals that we
are required to
'apply to the timers in this form


'open the file that contains details about the frequency with which the tools
e auctioned.

Dim Thisfilesname As String
Dim Thisfileslength As Integer
Dim ToolAuctFreq As String * 10
Dim DelayToBidAssess As String * 10
```

```
efine the file's name
hisfilesname = Defaultpath & "Factory\System\Config\toolfreq.sys"

efine the files length
nisfileslength = 4

et the tool auction frequency in seconds
oolAuctFreq = Input_From_File(Thisfilesname, Thisfileslength)

pply the frequency to the timer
ool_Timer.Interval = (CLng(ToolAuctFreq) * 1000)

pen the file that contains the time delay until the auction timeout delay.
isfilesname = "C:\Factory\System\Config\TlActOut.sys"

efine the file's length
isfileslength = 4

layToBidAssess = Input_From_File(Thisfilesname, Thisfileslength)

id_Eval.Interval = (CLng(DelayToBidAssess) * 1000)


initialise toolpointer variable - the toolpointer is a pointer that points to t
 tool availability
table and auctions each tool off the table, once it has tried and succeeded or
ried and failed
o auction a tool off the table the pointer points to the next tool in the tabl
 to be auctioned
this system is implemented so that there is no Bias in the system - that is eac
 tool is given
equal opportunity to be auctioned.
oolPointer = 1 'point to the first row in the table


'create the desired DDE links

all CommDDElink(C01v1ret, "FACT.C01V1TOOL.f_CV")
all CommDDElink(C01v2ret, "FACT.C01V2TOOL.f_CV")
all CommDDElink(C01v4ret, "FACT.C01V4TOOL.f_CV")
all CommDDElink(C02v1ret, "FACT.C02V1TOOL.f_CV")
all CommDDElink(C02v2ret, "FACT.C02V2TOOL.f_CV")
all CommDDElink(C02v4ret, "FACT.C02V4TOOL.f_CV")

all CommDDElink(C03v1ret, "FACT.C03V1TOOL.f_CV")
all CommDDElink(C03v2ret, "FACT.C03V2TOOL.f_CV")
all CommDDElink(C03v4ret, "FACT.C03V4TOOL.f_CV")


nd Sub

ub Form_Unload (Cancel As Integer)
```

```
If FactoryIsUnloading = 1 Then 'the entire factory Controller is exiting so the
fore we are
'required to unload this form
    FactoryIsUnloading = 0
    Exit Sub
    Else 'we are just hiding this form as we want the events on the form to stil
l be active
    'prevent the form from being unloaded
    Cancel = 1
    'now hide the form
    Tool_Avle.Hide
End If

End Sub

Sub Set_Up_ToolAvle_Click ()
'define variables
Dim Filenam As String
Dim EndOfArray As Integer
Dim j  As Integer
Dim Fileslength  As Integer
Dim recordlength As Integer
Dim Thisstate As Optionstate
Dim Thatstate As OldOptions
Dim OptionList() As OldOptions
Dim S As Integer
Dim Filenum As Integer
Dim i As Integer
Dim Msg As String
Dim Reply As Integer
Dim Title As String
Dim Typ As Integer


'when this button is pushed it allows the user to setup the factory in terms of
tool location
'for the first time


On Error Resume Next


Tool_Timer.Enabled = False 'set it equal to false to disable current toolbid
Bid_Eval.Enabled = False



'ask the user whether they are sure that they want to reset the tool location
atus
Msg = "Are you sure that you want to reset the tool location status of the Fact
ry Enviornment?"
Title = "Reset Tool Location"
Typ = MB_ICONSTOP + MB_YESNO

'get the reply
```

```
?eply = MsgBox(Msg, Typ, Title)

If Reply = IDNO Then 'The user does not want to reconfigure the current tool ava
ilability status
so exit the procedure
    Exit Sub
End If


for j = 1 To Processlist.ListCount

    'clear the invisible list list1
    List1.Clear

    'define a freefile number
    Filenum = FreeFile

    'define the process option file to open that hold some of this processes toc
 data
    Filenam = "c:\Factory\System\" & Trim(Processlist.list(j - 1)) & ".Opt"
    EndOfArray = 4

    'read the contents of the file to memory
    Call ReadToMemory(Filenum, Filenam, Thisstate, EndOfArray)

    'add the contents of the file to memory
    For i = 1 To EndOfArray
        List1.AddItem Optionstatearray(i).OptionCaption
    Next

    'close the file
    Close Filenum


    'Now we must open the file which has the names of all the other tool types
    'configured for this particular machine.

    'define a freefile reference number
    Filenum = FreeFile
    'define the size of the storage variable for the file
    recordlength = Len(Thatstate)
    'define the filename to open
    Filenam = "c:\Factory\System\" & Trim(Processlist.list(j - 1)) & ".Old"

    'Find out how many records there are in the file
    Fileslength = GetFilesLength2(Trim(Filenam), Thatstate, OptionList(), Filen
?)

    'Write the file contents to memory
    Call WriteToMemory3(Filenum, Fileslength, OptionList())

    now write the contents of the file into the listbox
    For i = 1 To Fileslength
        List1.AddItem OptionList(i).OptionName
    Next
```

```
    Close Filenum

    'Tidy up the listbox by removing any blanks
    For i = 0 To ((List1.ListCount) - 1)
        If Len(Trim(List1.list(i))) = 0 Then
            List1.RemoveItem i
        End If
    Next

    'Remove common names from listbox
    For i = 0 To ((List1.ListCount) - 1)
        For S = (i + 1) To ((List1.ListCount) - 1)
            If InStr(Trim(List1.list(i)), Trim(List1.list(S))) = 1 And Len(Trim'
List1.list(S))) > 0 Then 'there is a match
                List1.RemoveItem i
            End If
        Next
    Next


    'Declare variables
    Dim TrimMachineName As String * 3
    Dim TrimToolName As String * 4
    Dim TrimMach As String * 3
    Dim TrimOption As String * 4
    Dim filename As String
    Dim Thistool As Toolrecord
    Dim Thesetools() As Toolrecord
    Dim Filesize As Variant
    Dim ListBoxLocation As String
    Dim ToolSizeBeingAdded As Variant
    Dim Listname As String * 20
    Dim ListLength As Integer
    Dim IsItemInList As Variant
    Dim Thisfileslength As Integer
    Dim ThatTool As Toolrecord
    Dim ThoseTools() As Toolrecord
    Dim MainToolist() As ToolIdRecord
    Dim ThisToolsId As ToolIdRecord
    Dim NoOfTooltypes As Integer
    Dim TheRequiredId As Integer
    Dim Fileno As Integer
    Dim Lastposition As Integer
    Dim CurrentRow As Integer
    Dim Toolcode As Single
    Dim LifePercentage As Integer

    'Type Toolrecord
    'Recordleng As Integer
    'Toolsize As Single
    'ToolId As Integer
    'ToolLife As Integer
    'LifeUsed As Integer
    'End Type
```

```
   'for each tooltype specified in the list1 listbox we must open up a file for
each tool and
   'read the data for each tooltypes' sizes and place the data in the grids req
ired.

   For i = 1 To List1.ListCount

       TrimMachineName = Trim(Processlist.list(j - 1))
       TrimToolName = Trim(List1.list(i - 1))

       'define the name of the file required to be opened
       filename = "C:\Factory\Process\ToolFile\" & TrimMachineName & "!" & Trim
oolName & ".tl"

       'Declare a freefile number
       Filenum = FreeFile

       'get the size of the file
       Filesize = getFileslength(filename, Thistool, Thesetools(), Filenum)

       ReDim Thesetools(Filesize)

       For S = 1 To Filesize
           Get Filenum, S, Thesetools(S)

           'get the value of the life of the tool used as a percentage
           If Thesetools(S).LifeUsed = 0 Then  'if the tool hasnt been used bef
ore
               LifePercentage = 100
               Else
                   LifePercentage = CInt(Thesetools(S).LifeUsed / Thesetools(S)
ToolLife)
           End If

           ' Derive the toolcode for this tool
           Toolcode = (Thesetools(S).Toolid) + ((Thesetools(S).Toolsize) / 100)
           'configure a row for this particular tool in the grid and add the da
ta
           Tool_Availability.AddItem List1.list(i - 1) & Chr$(9) & Processlist.
list(j - 1) & Chr$(9) & "IDLE" & Chr$(9) & "FACT" & Chr$(9) & Trim(LifePercenta
e) & Chr$(9) & Thesetools(S).Toolsize & Chr$(9) & Chr$(9) & CStr(Toolcode)
       Next

       'close the file for the current tool\machine combination
       Close Filenum
   'repeat the process for the next tooltype in the list
   Next
'get the name of the next process in the process list and add the tool details f
or this process
'to the list.
Next

Setup_Save.Enabled = True

NewSetUpDefined = 1
```

```
End Sub

Sub Setup_Save_Click ()
'this procedure allows the user the ability to save the tools available initial
stion setup to
'file.

'declare variables
Dim i As Integer
Dim NumberOfRecords As Integer
Dim Tool_Avlearray() As Tool_AvleRec
Dim Thistoolsrec As Tool_AvleRec
Dim Rowcount As Integer
Dim j As Integer
Dim Nextelement As Integer
Dim Filenum As Integer
Dim recordlength As Integer
Dim LoopCount As Integer
Dim firsttime As Single
Dim Secondtime As Single
Dim Thislength As Integer

Dim Stepcount As Integer

On Error Resume Next

'get the number of records required to be stored
NumberOfRecords = (Tool_Availability.Rows - 1)

'redimension the array to suit the number of records that are required to be st
red
ReDim Tool_Avlearray(NumberOfRecords)

'pass the value of the number of records into the first record of the array
Tool_Avlearray(1).NumberOfRecords = NumberOfRecords

'now for each row in the grid we must store the grids contents

'set the first row to be considered as being the first row
Rowcount = 1

For i = 1 To (NumberOfRecords)
    'for each record of the array add a row's cells of data to each element of
hat record
    Stepcount = Stepcount + 1

    If Stepcount = 3 Then
        Stepcount = 0
        DoEvents
    End If

    Tool_Availability.Row = Rowcount
    Tool_Availability.Col = 0
    Tool_Avlearray(i).ToolType = Tool_Availability.Text
```

```
    Tool_Availability.Col = 1
    Tool_Avlearray(i).ProcessType = Tool_Availability.Text
    Tool_Availability.Col = 2
    Tool_Avlearray(i).Availability = Tool_Availability.Text
    Tool_Availability.Col = 3
    Tool_Avlearray(i).Location = Tool_Availability.Text
    Tool_Availability.Col = 4
    Tool_Avlearray(i).LifeUsed = Tool_Availability.Text
    Tool_Availability.Col = 5
    Tool_Avlearray(i).Toolsize = Tool_Availability.Text
    Tool_Availability.Col = 6
    Tool_Avlearray(i).Reserved = Tool_Availability.Text
    Tool_Availability.Col = 7
    Tool_Avlearray(i).Toolcode = Tool_Availability.Text
    'increment the row of focus as the next row
    Rowcount = Rowcount + 1
increment the array to the next record
ext


now open the file that is required to hold all of this tool information and wr
te the information
'to the file


'define a freefile
'ilenum = FreeFile

'get the variable storage length
ecordlength = Len(Thistoolsrec)

pen "c:\Factory\Config\Tools\Toolrecd.tl" For Random Access Write Shared As Fi
num Len = recordlength


or i = 1 To NumberOfRecords
    Put Filenum, i, Tool_Avlearray(i)
Next

lose Filenum


'Type Tool_AvleRec
    ' NumberOfRecords As Integer
'    ToolType As String * 30
  '  ProcessType As String * 30
   ' Availability As String * 15
   ' Location As String * 10
   ' LifeUsed As Integer
   ' ToolSize As Integer
   ' Reserved As String * 10
   ' Toolcode As Single
'End Type
Setup_Save.Enabled = False
```

```
End Sub

Sub Tool_Timer_Timer ()
'this particular timer is responsible for conducting auctions for the tools in
he list on this
'forms grid control. The timer when activated tries to auction a tool off the tc
p of the list
'when it's first timer event occurs by sending out details of that tool to ever
 node in the
'factory. Each node in the factory then forms a bid for that tool and a second
imer event in
'this application is activated which then compares the formulated bids. When a t
ool is awarded
'to a cell the table in this application is modified to reflect this.


'declare variables
Dim i As Integer
Dim NumberOfRecords As Integer
Dim Tool_Avlearray() As Tool_AvleRec
Dim Thistoolsrec As Tool_AvleRec
Dim Rowcount As Integer
Dim j As Integer
Dim Nextelement As Integer
Dim Filenum As Integer
Dim recordlength As Integer
Dim filename As String * 40
Dim Thisfilesname As String
Dim Thisfileslength As Integer
'define nodefile variables
Dim Thisrecord As nodelistrecord
Dim position As Integer
Dim LoopCount As Integer
Dim firsttime As Single
Dim Secondtime As Single
Dim Thisdata As String * 20
Dim Passnumber As Integer


On Error Resume Next


'disable the tool timer so that it can respond in a timely fashion
Tool_Timer.Enabled = False

'the first thing that we are required to do is to open up the file that holds a
l the tool status
'information and write this information to the table so that when this form is
ctive both the
'user and the program can observe the latest uptodate status of the system befc
e a tool auction
'is allowed to take place.

'however we can only do this when we are sure that nobody else is using it, th
efore we are required
```

```
)OL_AVL.FRM - 28

to see if the toolbidding flag file is available - if it is it can open up the
ool file, otherwise
it must loop until it can.

im Thislength As Integer


define a freefile
ilenum = FreeFile

'this is the Tool_AvleRec
 NumberOfRecords As Integer
'ToolType As String * 30
 ProcessType As String * 30
 Availability As String * 15
'Location As String * 10
 LifeUsed As Integer
 Toolsize As Integer
'Reserved As String * 10
 Toolcode As Single


'get the variable storage length
 ecordlength = Len(Thistoolsrec)

'open the required file
 pen "C:\Factory\Config\Tools\Toolrecd.tl" For Random Access Read Shared As Fil
num Len = recordlength

 redimension the array to take the first record only so we can get the length o
 the file required
'to be read.
 eDim Tool_Avlearray(1)

'get the first record in the array so that we are able to see how long the tota
 file is.
et Filenum, 1, Tool_Avlearray(1)

umberOfRecords = Tool_Avlearray(1).NumberOfRecords
MsgBox (CStr(NumberOfRecords))

 we must check to make sure whether the system has been configured or not befor
 we try to load
'records into the cells-we do this by checking the length of the file
f Tool_Avlearray(1).NumberOfRecords = 0 Then
    'tell the user that he needs to configure the tool availability module
    MsgBox ("The Tool Availability is not yet configured. Please configure imme
iately")
    'close the file
    Close Filenum
    Exit Sub
nd If

'now prepare the table to accept the data under any circumstances

'MsgBox (CStr(NumberOfRecords))
```

```
If Tool_Availability.Rows = 1 Then 'although the file exists we are required to
create the required
'grids for the data to be inserted into
    Tool_Availability.Rows = NumberOfRecords + 1
End If


Passnumber = 0
ReDim Tool_Avlearray(NumberOfRecords)

For i = 1 To NumberOfRecords  'for each record in the file get it and read it t
 memory
    Get Filenum, i, Tool_Avlearray(i)'get the contents of the associated record
    'and then write the contents of the record into the relevant part of the ta
le that has been
    'constructed, when we do this the original contents of each 'Cell' will be c
verwritten with
    'the new contents.
    'MsgBox (CStr(i))
    Passnumber = Passnumber + 1

    If Passnumber = 3 Then
        Passnumber = 0
        DoEvents
    End If

    Tool_Availability.Row = i
    Tool_Availability.Col = 0
    Tool_Availability.Text = Tool_Avlearray(i).ToolType
    Tool_Availability.Col = 1
    Tool_Availability.Text = Tool_Avlearray(i).ProcessType
    Tool_Availability.Col = 2
    Tool_Availability.Text = Tool_Avlearray(i).Availability
    Tool_Availability.Col = 3
    Tool_Availability.Text = Tool_Avlearray(i).Location
    Tool_Availability.Col = 4
    Tool_Availability.Text = Tool_Avlearray(i).LifeUsed
    Tool_Availability.Col = 5
    Tool_Availability.Text = Tool_Avlearray(i).Toolsize
    Tool_Availability.Col = 6
    Tool_Availability.Text = Tool_Avlearray(i).Reserved
    Tool_Availability.Col = 7
    Tool_Availability.Text = Tool_Avlearray(i).Toolcode
Next


Close Filenum

DoEvents
' the tool size
'Type Tool_AvleRec
 '    ToolType As String * 30
  '   ProcessType As String * 30
   ' Availability As String * 15
   ' Location As String * 10
   ' LifeUsed As Integer
```

```
)OL_AVL.FRM - 30

   ' ToolSize As Integer
   ' Reserved As String * 10
   ' Toolcode As Single
'End Type

now that we have the latest up to date information regarding the status of all
he tools in
the factory we can conduct an auction for the tools in the grid

'for each element in the grid check to see if the tool
    'is being used - if it is not then execute an auction for it.
    'reset column to make the availability column the subject of the examinatio


'*****EVERYTHING WORKS FINE UP TO THISPOINT - ANY ERROR WILL OCCUR FROM THIS PO
T ON********

Toolpointer = 4
set the tool to be considered as that pointed to by the toolpointer
ool_Availability.Row = ToolPointer
Toolpoint.Text = ToolPointer
MsgBox ("Toolpointer is " & CStr(Toolpointer))

'set the column being pointed to as column two
ool_Availability.Col = 2

If Trim(Tool_Availability.Text) = "IDLE" Then 'the algorithm has found a tool t
at is not
    'being used at present therefore we can conduct an auction for this tool
    'now we are required to open a "Tool Bid File" and send the tool data out c
o the
    'network and then prompt the cells on the network to start to formulate a b
 for
    'the tool being auctioned.

    'now conduct an auction for that tool

    'open the tool bidfile
    Thisfilesname = "C:\Factory\Bidding\bids\Toolbids.bdf"

    Thisfileslength = 8 'set variable storage length = 8 bytes

    'reset column number to pick up current tool location
    Tool_Availability.Col = 3
    'MsgBox (CStr(Toolpointer))
    Toollocation = Tool_Availability.Text

    'reset column number to pick up the tool code of the tool
    Tool_Availability.Col = 7
    ToolTypeCode = Tool_Availability.Text
    'MsgBox (Tooltypecode)
    'open the tool bidfile and write the Tool bid data to it.
    Call OutPut_Two_Strings(Thisfilesname, Thisfileslength, Toollocation, Tool
peCode)
```

```
    'now prompt the cell controllers over the network to respond to the auctioni
ng mechanism.
    If FileLen(Netdrive & "Factory\Config\nodefile.txt") = 0 Then 'if there is n
othing in the file then exit
        'this subprocedure
        Exit Sub
    End If

    DoEvents

    'Assign a free number to this file
    Filenum = FreeFile
    'Calculate the recordlength in Bytes
    recordlength = Len(Thisrecord)

    'Open the file for random access with allowable functions read
    Open Netdrive & "Factory\Config\nodefile.txt" For Random Access Read Shared
As Filenum Len = recordlength

    position = 1

    Do Until EOF(Filenum)
    'get the name of the first node from the file
        Get Filenum, position, Thisrecord
        'increment position variable so the next nodename will be gotten
        position = position + 1
        'now send a "1" to the "bdToolstrt" in the cell denoted by the record.
        Node = Trim(Thisrecord.Node)
        If Trim(Node) Like "FACT" Or Len(Trim(Node)) = 10 Then

            'we do not need to send a prompt to the factory controller node the
efore look
            'at the next record to see if it is a cell controller
            GoTo NextStep
        End If

        'make a copy of the tool bidfile for the node and put it on the network
for the node so
        'that there is no file clash
        FileCopy "C:\Factory\Bidding\bids\Toolbids.bdf", Netdrive & "Factory\Bi
ding\bids\" & Trim(Node) & "BdDat" & ".bdf"


        'prompt each node in the Factory to start bid formulation.
        BlockTyp = "BdToolstrt"
        'MsgBox (Tooltypecode)
        Field = "A_CV"
        Texts = "1"

        Call sendinfo(Node, BlockTyp, Field, Texts)

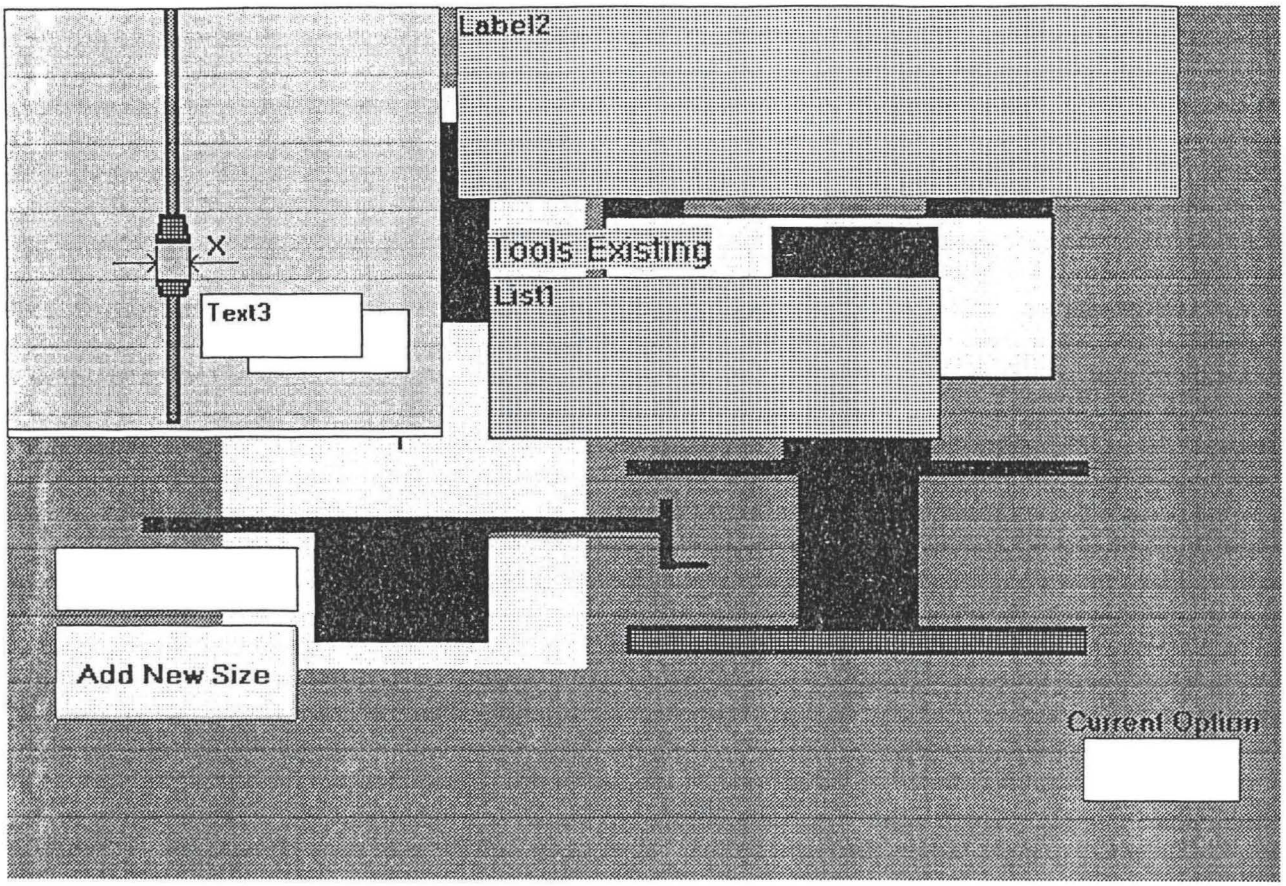

NextStep:
    Loop

    Close Filenum
```

```
   'now enable the Toolbid evaluation timer that will evaluate the bids as they
are recieved
   Bid_Eval.Enabled = True


   Else 'increment the toolpointer if the current tool is taken
       If ToolPointer = NumberOfRecords Then
           ToolPointer = 1
           Else
           ToolPointer = ToolPointer + 1
       End If
       Tool_Timer.Enabled = True
nd If

nd Sub
```


Label2
X
Text3
Tools Existing
List1
Add New Size
Current Option

```
OOLLIST.FRM - 1

Dim Countts As Integer
Dim Countss

Sub Command1_Click ()
Dim TrimMach As String * 3
Dim TrimOption As String * 4
Dim Filename As String
Dim ThisTool As Toolrecord
Dim TheseTools() As Toolrecord
Dim Filesize As Variant
Dim ListBoxLocation As String
Dim ToolSizeBeingAdded As Variant
Dim Listname As String * 20
Dim ListLength As Integer
Dim IsItemInList As Variant
Dim ThisFilesLength As Integer
Dim Filenum As Integer
Dim ThatTool As Toolrecord
Dim Thosetools() As Toolrecord
Dim MainToolist() As ToolIdRecord
Dim ThisToolsId As ToolIdRecord
Dim NoOfToolTypes As Integer
Dim i As Integer
Dim TheRequiredId As Integer
Dim Fileno As Integer

On Error Resume Next

'Trim the name of the machine for which the tools are being configured down
'to a one bit string
TrimMach = ThisIndex

'Trim the name of the type of tool being configured down to a four bit string
TrimOption = Activeoption




Countts = Countts + 1
    Filename = "c:\factory\system\AlToolTyps.Tl"
    Fileno = FreeFile
    NoOfToolTypes = getFileslength1(Filename, ThisToolsId, MainToolist(), Filer
)

   'now write the list to memory
    Call WriteToMemory(Fileno, NoOfToolTypes, MainToolist())

Text3.Text = MainToolist(Countts).ToolType
Text4.Text = MainToolist(Countts).Toolid

Close Filenum

End Sub
```

```
Sub Command2_Click ()
Dim TrimMach As String * 3
Dim TrimOption As String * 4
Dim Filename As String
Dim ThisTool As Toolrecord
Dim TheseTools() As Toolrecord
Dim Filesize As Variant
Dim ListBoxLocation As String
Dim ToolSizeBeingAdded As Variant
Dim Listname As String * 20
Dim ListLength As Integer
Dim IsItemInList As Variant
Dim ThisFilesLength As Integer
Dim Filenum As Integer
Dim ThatTool As Toolrecord
Dim Thosetools() As Toolrecord
Dim MainToolist() As ToolIdRecord
Dim ThisToolsId As ToolIdRecord
Dim NoOfToolTypes As Integer
Dim i As Integer
Dim TheRequiredId As Integer
Dim Fileno As Integer

On Error Resume Next

'Trim the name of the machine for which the tools are being configured down
'to a one bit string
TrimMach = ThisIndex

'Trim the name of the type of tool being configured down to a four bit string
TrimOption = Activeoption

Countss = Countss + 1
Filename = "C:\Factory\Process\ToolFile\" & TrimMach & "!" & TrimOption & ".tl"

'Get the current length of the file
Filenum = FreeFile

ThisFilesLength = getFileslength(Filename, ThatTool, Thosetools(), Filenum)
'Text1.Text = ThisFilesLength
'Now read the information that is in the file to memory

Call WriteToMemory1(Filenum, ThisFilesLength, Thosetools())

Text3.Text = Thosetools(Countss).Toolsize
Text4.Text = Thosetools(Countss).Toolid

Close Filenum

End Sub

Sub Add_Click ()
Dim Msg As String

Msg = "Before adding new sizes of specific tools it is extremely important"
```

```
'sg = Msg & " to examine what the size pertains to in that instance eg: For "
 sg = Msg & "an endmill the size will pertain to the diameter of the tool, "
Msg = Msg & "however for a parting tool it is the width of the tool that may"
'sg = Msg & " be important. To view an example of how a parting tool would"
 sg = Msg & " be specified select from the menu 'Help' & 'About' & 'Adding New S
izes' "
Msg = Msg & " & 'Example'. To select the size specification for the current "
 sg = Msg & "tool type select 'Tool Specification' from the menu."


'sgBox Msg, MB_ICONINFORMATION, "Size Definition"
End Sub

ub AddNewTool_Click ()

'This command button allows the user to type in the name
 of the tool that he requires to configure for the factory
 and then save that tool to file.
Dim TrimMach As String * 3
 im TrimOption As String * 4
 im Filename As String
Dim ThisTool As Toolrecord
 im TheseTools() As Toolrecord
 im Filesize As Variant
Dim ListBoxLocation As String
Dim ToolSizeBeingAdded As Variant
 im Listname As String * 20
Dim ListLength As Integer
Dim IsItemInList As Variant
 im ThisFilesLength As Integer
Dim Filenum As Integer
Dim ThatTool As Toolrecord
 im Thosetools() As Toolrecord
Dim MainToolist() As ToolIdRecord
Dim ThisToolsId As ToolIdRecord
 im NoOfToolTypes As Integer
Dim i As Integer
Dim TheRequiredId As Integer
 im Fileno As Integer
Dim Msg As String
Dim Title As String
 im ExpToolLife As String
Dim DefVal As String


Type Toolrecord
'Recordleng As Integer
 ToolSize As Single
 ToolId As Integer
'ToolLife As Integer
'LifeUsed As Integer
 End Type

'declare a user defined variable type that records all the
 different types of tools that are available in the Factory
'and their associated identification codes.
```

```
'Check to see if the user has added anything to textbox2.
If Len(Text2.Text) = 0 Then
    MsgBox "Please enter the size of the tool you want to configure in mm in the
 box under the 'Add Tool Here' caption.", MB_ICONSTOP, "Tool Size"
    Exit Sub
End If

'Trim the name of the machine for which the tools are being configured down
'to a one bit string
TrimMach = ThisIndex
'MsgBox (CStr(ThisIndex))
'Trim the name of the type of tool being configured down to a four bit string
TrimOption = Activeoption

'Combine Trimmach and trimoption to give the name of the tool file that is
'required to be opened
ToolSizeBeingAdded = Val(Text2.Text)

'Otherwise add the new item to the list
List1.AddItem CStr(ToolSizeBeingAdded) & "mm"

'this is all fine but we must also ask the user how many hours of toollife this
new tool has
Msg = "Please input a value in hours for the expected tool life of this tool."
DefVal = "2000"
Title = "State Tool Life"

ExpToolLife = InputBox$(Trim(Msg), Title, DefVal)

'Now we must save the item that has been added to the list to file so that
'when this form is reloaded the information can once again be added to the
'list and so the data from it can be used in other applications as required

Filename = "C:\Factory\Process\ToolFile\" & TrimMach & "!" & TrimOption & ".tl"

'MsgBox (Filename)
'Get the current length of the file
Filenum = FreeFile

ThisFilesLength = getFileslength(Filename, ThatTool, Thosetools(), Filenum)
'Now read the information that is in the file to memory

Call WriteToMemory1(Filenum, ThisFilesLength, Thosetools())

'Now increment the length of the file to show that a new tool is to be added
Thosetools(1).Recordleng = ThisFilesLength + 1
Thosetools(Thosetools(1).Recordleng).Toolsize = CSng(ToolSizeBeingAdded)
Thosetools(Thosetools(1).Recordleng).ToolLife = CInt(ExpToolLife)


'Now we must open the file which contains all the names of the tool types in
'the factory and their id's so that we can match the tool type being
'considered in this procedure with that in the file and once this has been
'done the tool's id can be taken from the file and stored in the file which
'details the pertinent machinetype and tooltype for that machine.
```

```
 First get the total number of tool types that have already been specified.
Filename = "c:\factory\system\AlToolTyps.Tl"
Fileno = FreeFile
 Type ToolIdRecord
'CurrentToolId As Integer
'Recordlength As Integer
 ToolType As String * 20
 ToolId As Integer
'End Type
'oOfToolTypes = getFileslength1(Filename, ThisToolsId, MainToolist(), Fileno)
    Text1.Text = NoOfToolTypes
'now write the list to memory
 all WriteToMemory(Fileno, NoOfToolTypes, MainToolist())

'Now search for a common tooltype and from that extract tool id
 or i = 1 To NoOfToolTypes
    'if the tooltype in that file position is the same as the tooltype for that
    'being dealt with in this procedure then the id associated with that positi

    'is what we want.
    If Trim(MainToolist(i).ToolType) = Trim(Activeoption) Then
        TheRequiredId = MainToolist(i).Toolid
    MsgBox (CStr(TheRequiredId))
    End If
'ext
   'Text1.Text = TheRequiredId
 Close Fileno

 Now we have found the correct id we can continue.

   'Type ToolIdRecord
   'CurrentToolId As Integer
   'Recordlength As Integer
   'ToolType As String * 20
   'ToolId As Integer
   'End Type

   'Type Toolrecord
   'recordleng As Integer
   'Toolsize As Single
   'ToolId As Integer
   'End Type
Thosetools(Thosetools(1).Recordleng).Toolid = TheRequiredId
 Text1.Text = ThoseTools(ThoseTools(1).Recordleng).ToolId

Call WriteToFile1(Filenum, ThisFilesLength, Thosetools())

Close Filenum


End Sub

Sub Example_Click ()
Dim Msg As String
```

```
Msg = "If we wanted to add a new toolsize for a parting tool we "
Msg = Msg & "would select the 'Tool Specification' title from the menu and "
Msg = Msg & "the image at left would be displayed showing the user what dimensio
n"
Msg = Msg & " has to be defined."



Picture1.Visible = True
Label2.Visible = True
Label2.Caption = Msg

End Sub

Sub Form_Deactivate ()
Dim Msg As String
Dim Typ As Integer
Dim Title As String
Dim reply As Integer

Msg = "You cannot go to another window by clicking it - you must select "
Msg = Msg & "exit from this window's menu. "
Typ = MB_ICONEXCLAMATION + MB_OK
Title = "Exiting Error"
reply = MsgBox(Msg, Typ, Title)

loopy = loopy + 1
If loopy = 10 Then
Exit Sub
End If

Toollists.SetFocus

End Sub

Sub Form_Load ()
Dim TrimMach As String * 3
Dim TrimOption As String * 4
Dim Filename As String
Dim ThisTool As Toolrecord
Dim TheseTools() As Toolrecord
Dim Filesize As Integer
Dim ListBoxLocation As String
Dim Filenum As Integer
Dim Toolname, temptoolname As String * 20
Dim start As Integer
Dim i As Integer
Dim FinalString As String
Dim Noreplace As Integer

On Error Resume Next
'Type Toolrecord
'Recordleng As Integer
'Toolsize As Single
'ToolId As Integer
'End Type
```

```
OLLIST.FRM - 7


eposition the picture box
icture1.Left = 0
cture1.Top = 0


Define the Caption of this form
olsexisting.Caption = Trim(Activeoption) & " Tools - Sizes Already Existing"

oolname = Trim(Activeoption)
emptoolname = Toolname

start = InStr(Toolname, " ")
preplace = Len(Trim(Toolname)) - (start - 1)
id(Toolname, start, Noreplace) = "                    "
oolname = Trim(Toolname)


Length = Len(Trim(Toolname)) + 1
id(temptoolname, 1, Length) = "                    "
inalString = Trim(Toolname) & Trim(temptoolname)


rrOpt.Text = FinalString
'Trim the name of the machine for which the tools are being configured down
'to a one bit string
rimMach = ThisIndex

'Trim the name of the type of tool being configured down to a four bit string
rimOption = Activeoption

'Combine Trimmach and trimoption to give the name of the tool file that is
required to be opened
ilename = "C:\Factory\Process\ToolFile\" & TrimMach & "!" & TrimOption & ".tl"

Text1.Text = Filename
ilenum = FreeFile

ilesize = getFileslength(Filename, ThisTool, TheseTools(), Filenum)

'Text1.Text = Filesize
Clear the current list contents
oollists.List1.Clear

read toolinfo from file and write to the listbox
all ReadAndAddToList(TheseTools(), Filesize, Filenum)

lose Filenum

End Sub

ub Picture1_Click ()
Picture1.Visible = False
abel2.Visible = False
nd Sub
```

```
Sub View_Click ()
Dim Msg As String
On Error Resume Next

Picture1.Picture = LoadPicture("C:\FACTORY\PROCESS\TOOLFILE\TOOLBITS\" & CurrOp
.Text & ".bmp")
Picture1.Visible = True

Msg = "The dimension that is required to be stated when defining the toolsize"
Msg = Msg & " for the tool shown at left is marked with an 'X' as shown."
Label2.Caption = Msg
Label2.Visible = True



If Err = ERR_FILENOTFOUND Then
Msg = "Tool Specification bitmap does not exist for this process. Please create
"
Title = "Tool Specification Bitmap Status"
Typ = MB_ICONEXCLAMATION + MB_OKCANCEL
reply = MsgBox(Msg, Typ, Title)

  If reply = IDCANCEL Then
   Exit Sub
      Else
      'Fire the paintbrush program and let the user create it
       AppActivate "Microsoft Visual Basic"    'Change focus to Visual Basic.
       X = Shell("PBrush.exe", 3)    ' Shell the Paintbrush and maximise
       AppActivate "Paintbrush - (Untitled)"
       SendKeys "%(F){Down}{Enter}{tab 2}{up}{Enter}{Down 5}{Enter}{Down 2}{Ent
r}{Down 4}{Enter}{Down}{enter}{Down}{Enter}{tab 7}{Down}{enter}  "
       Exit Sub
   End If

End If

End Sub
```

```
'reposition the picture box
Picture1.Left = 0
Picture1.Top = 0


'Define the Caption of this form
Toolsexisting.Caption = Trim(Activeoption) & " Tools - Sizes Already Existing"

Toolname = Trim(Activeoption)
temptoolname = Toolname

start = InStr(Toolname, " ")
Noreplace = Len(Trim(Toolname)) - (start - 1)
Mid(Toolname, start, Noreplace) = "                    "
Toolname = Trim(Toolname)


Length = Len(Trim(Toolname)) + 1
Mid(temptoolname, 1, Length) = "                    "
FinalString = Trim(Toolname) & Trim(temptoolname)


CurrOpt.Text = FinalString
'Trim the name of the machine for which the tools are being configured down
'to a one bit string
TrimMach = ThisIndex

'Trim the name of the type of tool being configured down to a four bit string
TrimOption = Activeoption

'Combine Trimmach and trimoption to give the name of the tool file that is
'required to be opened
Filename = "C:\Factory\Process\ToolFile\" & TrimMach & "!" & TrimOption & ".tl"

'Text1.Text = Filename
Filenum = FreeFile

Filesize = getFileslength(Filename, ThisTool, TheseTools(), Filenum)

'Text1.Text = Filesize
'Clear the current list contents
Toollists.List1.Clear

'read toolinfo from file and write to the listbox
Call ReadAndAddToList(TheseTools(), Filesize, Filenum)

Close Filenum

End Sub

Sub Picture1_Click ()
Picture1.Visible = False
Label2.Visible = False
End Sub
```

```
FACTORY.BAS - 1

'This is the bid comparison application and it is responsible for the
'comparison of all bids in the Factory as lodged by the Cell Controller's
'VMD's

'Written by Matthew J Jolly
'PhD Student
'Production Technology Department
'Massey University
'Palmerston North

'State all arrays begin at 1
Option Base 1
'State that all variables must be declared explictly
Option Explicit
'State that text comparison is not case-sensitive
Option Compare Text


'MsgBox parameters
Global Const MB_OK = 0                    ' OK button only
Global Const MB_OKCANCEL = 1              ' OK and Cancel buttons
Global Const MB_ABORTRETRYIGNORE = 2      ' Abort, Retry, and Ignore buttons
Global Const MB_YESNOCANCEL = 3           ' Yes, No, and Cancel buttons
Global Const MB_YESNO = 4                 ' Yes and No buttons
Global Const MB_RETRYCANCEL = 5           ' Retry and Cancel buttons

Global Const MB_ICONSTOP = 16             ' Critical message
Global Const MB_ICONQUESTION = 32         ' Warning query
Global Const MB_ICONEXCLAMATION = 48      ' Warning message
Global Const MB_ICONINFORMATION = 64      ' Information message

Global Const MB_APPLMODAL = 0             ' Application Modal Message Box
Global Const MB_DEFBUTTON1 = 0            ' First button is default
Global Const MB_DEFBUTTON2 = 256          ' Second button is default
Global Const MB_DEFBUTTON3 = 512          ' Third button is default
Global Const MB_SYSTEMMODAL = 4096         'System Modal

'MsgBox return values
Global Const IDOK = 1                     ' OK button pressed
Global Const IDCANCEL = 2                 ' Cancel button pressed
Global Const IDABORT = 3                  ' Abort button pressed
Global Const IDRETRY = 4                  ' Retry button pressed
Global Const IDIGNORE = 5                 ' Ignore button pressed
Global Const IDYES = 6                    ' Yes button pressed
Global Const IDNO = 7                     ' No button pressed


'******************** DDE Constants **********************
'LinkMode (forms and controls)
Global Const NONE = 0           ' 0 - None
Global Const LINK_SOURCE = 1    ' 1 - Source (forms only)
Global Const LINK_AUTOMATIC = 1 ' 1 - Automatic (controls only)
Global Const LINK_MANUAL = 2    ' 2 - Manual (controls only)
Global Const LINK_NOTIFY = 3    ' 3 - Notify (controls only)
```

```
'General Constants and variables
Global Const FACTORY = "FIX"
Global Filenamed As String * 30
Global steps As Integer
Global ActiveOption As String
Global thisIndex As String
Global s As Integer
Global ThisMachtype As Integer
Global Const DefaultPath = "C:\"
Global Const JobFilepath = "r:\Factory\Jobfiles\"
Global Machcal As Single
Global MachSelected As String
Global CapacityStated As Integer
Global NoOfToolTypes As Integer
Global Const Systemname = "the System Integration Tool for Advanced Manufacturi
g Automation"
Global tasknam As String
Global CNCncifname As String * 6
Global SetUpInsfname As String
Global check As Integer
Global Unloading As Integer
Global NodebeingConfiged As Integer
Global TheVmd As Integer
Global ThisNode As String * 10
Global Thisnodenam As String * 10
Global Factoryisauctioning As Integer
Global NumberOfBusyJobsAuctioned As Integer
Global Busy_Jobs_Auction As Integer
Global Firstpass As Integer
Global TryAgain As Integer
Global NumberOfCellControllers As Integer
Global FactoryIsUnloading As Integer
Global PostProcessorSelected As String * 40


'Drive that the network database resides on
Global Const Netdrive = "r:\"

'Process definition filelength variable
Global CurrId As Integer

'Global variable for queueing timer
Global CountToScanTime As Integer

'Global queue length variable
Global QFileslength As Integer

'User defined variable declarations
Type JobRecord
    Jobname As String * 13
    InFoFilename As String * 20
    FinalTask As Integer
    Tasknumber As Integer
    Machinetype As Single 'Type as 3.1 the ".1" represents capacity
    Taskname As String * 22
```

```
    EstCycletime As Integer
    Setuptime As Integer
    Estloadunloadtime As Integer
    NoofCycles As Integer
    CNCfilename As String * 13
    Geometricfilename As String * 13
    Tool1required As Single
    'Tool2required As Single
    'Tool3required As Single
    'Tool4required As Single
    Fixturesfilename As String * 13
    Status As Integer
    StartTime As String * 5
    Duedate As String * 15
    RequiredCompDate As String * 15
    RequiredComptime As String * 5
    Force As Integer
    SetUpInstructionsFilename As String * 20 'Shows datum graphically
End Type

'**********************FACTORY CONTROLLER QUEUEING VARIABLES************
Global Job As JobRecord
Global QueueJob(100) As JobRecord

'Define a variable type which records old process tool options
Type OldOptions
    Recordlen As Integer
    OptionName As String * 20
End Type

'Define a variable type which defines process parameters
Type ProcessDef
    ProcessNumber As Integer
    Processname As String * 20
    CurrentProcessId As Integer
    ProcessId As Integer
    Maxcap As Integer
End Type

'Declare user defined type which stores the data representing the option
'button captions in the "Define Tools" application
Type Optionstate
    OptionCaption As String * 20
End Type

'Declare tool\machine storage user defined variable
Type Toolrecord
    Recordleng As Integer
    Toolsize As Single
    Toolid As Integer
    ToolLife As Integer
    LifeUsed As Integer
End Type

'declare a user defined variable type that records all the
'different types of tools that are available in the Factory
```

```
'and their associated identification codes.
Type ToolIdRecord
    CurrentToolId As Integer
    recordlength As Integer
    ToolType As String * 20
    Toolid As Integer
End Type

'Create a user-defined variable that will be used in an
'array to store data regarding the type of Cad\cam packages
'that have been configured for the 'System Integration Tool
'For Advanced Manufacturing Automation'.

Type CadCamvariable
    Packagename As String * 50
    FireCommand As String * 50
    FocusCommand As String * 100
    WorkDirectory As String * 100
    Recordleng As Integer
End Type

'Define the variable type that will be used to store information
'regarding the types of post-processors available for the
'various processes available.

Type Postrecord
    CurrentIndex As Integer
    CurrentProcId As Integer
    Recordleng As Integer
    ProcessTyp As String * 50
    Index As Integer
    PostTyp As String * 50
End Type

'Decalre pending job variable type
Type Pendingjob
    Jobname As String * 13
    Filelength As Integer
End Type

'Declare Factory Queue Variable
Type Factqueue
    Jobname As String * 13
    Filelength As Integer
End Type

'Declare Factory Config startup variables
Type ConFigSup
    Procname As String * 30
    Capinterval As Integer
End Type

Type Nodesbid
    VMD1bid As Single
    VMD2bid As Single
    VMD3bid As Single
```

```
ACTORY.BAS - 5

    VMD4bid As Single
nd Type

Type bidEvaluation
    NodeName As String
    WinningBidsize As Single
    Winning_VMD As Integer
nd Type

'define a user defined variable type that stores the time distance between diffe
ent nodes
ype NodeTimedelay
    MatrixDim As Integer
    Interarrivaltime As Integer
    CellnameX As String * 10
    CellnameY As String * 10
nd Type


'This is a variable type that is used to store information about whether or not
ll bids have
'been recieved or not.
Type BidIsFinished
    NodeName As String * 20
    BidFin As Integer
End Type



 Constants required for error checking
Global Const ERR_ILLEGALFNCALL = 5
lobal Const ERR_OVERFLOW = 6
lobal Const ERR_TYPEMISMATCH = 13
Global Const ERR_OUTOFSTRINGSPACE = 14
lobal Const ERR_BADFILENAMEORNUMBER = 52
lobal Const ERR_FILENOTFOUND = 53
Global Const ERR_FILEALREADYOPEN = 55
lobal Const ERR_DISKFULL = 61
lobal Const ERR_PATHNOTFOUND = 76
Global Const ERR_SUBSCRIPTOUTOFRANGE = 9
lobal Const ERR_BADFILENAME = 64

Global OptionStateArray(4) As Optionstate

 efine a variable type that is used to store details of the
'factory enviornment nodes
Type nodelistrecord
    node As String * 10
nd Type

 a variable type that is used to store data regarding what processes
 are in what nodes
Type whatmachinesarewhere
    Filelength As Integer
    Nodenumber As Integer
```

```
    MachType As Single
    ProcessorType As String * 40
    Enabled As Integer
End Type

'this variable stores data regarding what nodes are disabled
Type Activenodes
    Filelength As Integer
    CellnodeNum As Integer
End Type


'this variable type stores data regarding the bidding data
'to be sent over the network.
Type Bid_data_To_Be_sent
    Jobname As String * 10
    Machinetype As Single
    EstCycletime As Integer
    Setuptime As Integer
    LoadUnloadtime As Integer
    NoofCycles As Integer
    Tool1required As Single
    'Tool2required As Single
    'Tool3required As Single
    'Tool4required As Single
    Activate As Integer
    Force As Integer
    AuctioningCell As String * 8
End Type

'this variable stores data regarding details of each Cell Controllers bid as for
mulated.
Type Bid_Evaluation_In_Process
    NodeName As String * 6
    Bidsize As Single
    VMDNumber As String * 12
    Vmdbusystate As String * 10
End Type

'define a user defined variable type that can be used to store the associated b
ffer sizes for
'each VMD's ingoing queue
Type InQueueBuffersize
    VMD1Buffer As Integer
    VMD2Buffer As Integer
    VMD3Buffer As Integer
    VMD4Buffer As Integer
End Type

'this variable type stores data regarding a tool's availability, current locatic
n, etc required
'for tool distribution purposes.
Type Tool_AvleRec
    NumberOfRecords As Integer
    ToolType As String * 30
    ProcessType As String * 30
```

```
FACTORY.BAS - 7

    Availability As String * 15
    Location As String * 10
    LifeUsed As Integer
    Toolsize As Integer
    Reserved As String * 10
    Toolcode As Single
End Type

Global CellsConfigered(50) As String * 20

'define variable type that stores the values of the submitted bids
Type ToolBid
    CellBidding As String * 10
    VMDBidding As String * 10
    StartDateBid As Single
    TotalBid As Double
    'ToolBidPosition As String * 10
End Type


'define a variable type that stores the current value of the winning
'bid and the cell submitting that bid
Type BidEval
    WinningNode As String * 10
    WinningBid As Integer
    Emptytasktool As Integer
End Type

Function ChekError (errtyp As Integer) As Integer

Dim Types As Integer
Dim Titles As String * 20
Dim mSG As String
Dim Response As Integer

Types = 0

Select Case errtyp

Case Is = ERR_TYPEMISMATCH

Titles = "Task Data Incomplete"
mSG = "The task data that you have described is either incomplete or the data"
mSG = mSG & " type entered is incorrect. Please refer to the help file regarding

mSG = mSG & " the correct data type entries that should be used for each field."

Case Is = ERR_PATHNOTFOUND
Titles = "JobName not specified"
mSG = "You not entered the name of the job you are configuring"
mSG = mSG & " this task for. Please do so"

Case Is = ERR_SUBSCRIPTOUTOFRANGE
ChekError = 2
Exit Function
```

```
Case Else
mSG = "An error of type " & Error & " is occurring"


End Select

Response = MsgBox(mSG, Types, Titles)


Select Case Response

Case 1 'Ok was selected
ChekError = 1
End Select

End Function

Function ChekError1 (ByVal errtyp As Integer) As Integer
Dim Titles As String * 20
Dim mSG As String
Dim Response As Integer
Dim Types As Integer

Select Case errtyp

Case Is = ERR_TYPEMISMATCH

Titles = "Task Data Incomplete"
mSG = "The task data that you have described is either incomplete or the data"
mSG = mSG & " type entered is incorrect. Please refer to the help file regardin
"
mSG = mSG & " the correct data type entries that should be used for each field.

Case Is = ERR_FILENOTFOUND
Titles = "CadCam Link Error"
mSG = "When you configured the Cad\Cam package you are presently trying"
mSG = mSG & " to run you either did not enter the name of the executable file"
mSG = mSG & " correctly or the program you are trying to run has not been insta
led properly. "
mSG = mSG & "Please delete the item from the list, re-configure and check that t
he package"
mSG = mSG & " has been installed properly."
Types = MB_ICONSTOP + MB_OK

Case Is = ERR_SUBSCRIPTOUTOFRANGE
ChekError1 = 2
Exit Function

Case Else
mSG = "An error of type " & Error & " is occurring"


End Select
```

```
esponse = MsgBox(mSG, Types, Titles)

elect Case Response

Case 1 'Ok was selected
hekError1 = 1
nd Select

nd Function

Function ChekError2 (errtyp As Integer) As Integer

im Types As Integer
Dim Titles As String * 20
im mSG As String
im Response As Integer

ypes = 0

Select Case errtyp

ase Is = ERR_TYPEMISMATCH
ChekError2 = 1
oTo TheEnd

ase Is = ERR_PATHNOTFOUND
Titles = "JobName not specified"
SG = "You not entered the name of the job you are configuring"
SG = mSG & " this task for. Please do so"

ase Is = ERR_SUBSCRIPTOUTOFRANGE
hekError2 = 2
Exit Function

ase Else
mSG = "An error of type " & Error & " is occurring"


nd Select

Response = MsgBox(mSG, Types, Titles)


elect Case Response

ase 1 'Ok was selected
ChekError2 = 1
nd Select

heEnd:
nd Function

ub ClearTextbox (ThisText As TextBox)
```

```
ThisText.Text = ""
End Sub

Function getFileslength (Filenam As String, ThatTool As Toolrecord, Thosetools(
 As Toolrecord, Filenum As Integer) As Integer
Dim Filelength As Integer

Filelength = Len(ThatTool)

Open Filenam For Random Access Read Write Shared As Filenum Len = Filelength
ReDim Thosetools(1)

Get Filenum, 1, Thosetools(1)
'MsgBox (CStr(Thosetools(1).Recordleng))
getFileslength = Thosetools(1).Recordleng

End Function

Function GetFilesLength1 (Filenam As String, ThisToolsId As ToolIdRecord, MainTo
olist() As ToolIdRecord, Filenum As Integer) As Integer
Dim Filelength As Integer

Filelength = Len(ThisToolsId)

ReDim MainToolist(1)
Open Filenam For Random Access Read Write Shared As Filenum Len = Filelength


Get Filenum, 1, MainToolist(1)
GetFilesLength1 = MainToolist(1).recordlength
'MsgBox (CStr(MainToolist(1).recordlength))
End Function

Function GetFilesLength2 (Filenam As String, Thatstate As OldOptions, OptionLis
() As OldOptions, Filenum As Integer) As Integer

Dim Filelength As Integer

Filelength = Len(Thatstate)

Open Filenam For Random Access Read Write Shared As Filenum Len = Filelength
ReDim OptionList(1)

Get Filenum, 1, OptionList(1)
GetFilesLength2 = OptionList(1).Recordlen

End Function


Function getfilesLength3 (Filenam As String, CadCamRecord As CadCamvariable, Ca
CamArray() As CadCamvariable, Filenum As Integer)

Dim Filelength As Integer

Filelength = Len(CadCamRecord)
```

```
pen Filenam For Random Access Read Write Shared As Filenum Len = Filelength
ReDim CadCamArray(1)

et Filenum, 1, CadCamArray(1)
getfilesLength3 = CadCamArray(1).Recordleng

nd Function

Function GetFileslength4 (Filenam As String, ThisProcPosts As Postrecord, ThisP
tProcsarray() As Postrecord, Filenum As Integer) As Integer

Dim Filelength As Integer

ilelength = Len(ThisProcPosts)

pen Trim(Filenam) For Random Access Read Write Shared As Filenum Len = Filelen
h
ReDim Preserve ThisPostProcsarray(1)

et Filenum, 1, ThisPostProcsarray(1)
GetFileslength4 = ThisPostProcsarray(1).Recordleng

nd Function

unction getfilesLength5 (Filenam As String, Thisjob As Pendingjob, Jobspending
As Pendingjob, Filenum As Integer) As Integer

Dim Filelength As Integer

ilelength = Len(Thisjob)

pen Filenam For Random Access Read Write Shared As Filenum Len = Filelength
eDim Jobspending(1)

et Filenum, 1, Jobspending(1)
etfilesLength5 = Jobspending(1).Filelength

nd Function

unction GetFileslength6 (Filenam As String, ThisQ As Factqueue, Thatque() As F
tqueue, ByVal Filenu As Integer) As Integer

Dim Filelength As Integer
ilelength = Len(ThisQ)

Open Filenam For Random Access Read Write Shared As Filenu Len = Filelength
eDim Thatque(1)

et Filenu, 1, Thatque(1)
etFileslength6 = Thatque(1).Filelength

End Function

unction Input_From_File (Filename As String, Filelength As Integer)
im Filenum As Integer
```

```
Dim StringToGet As String

Filenum = FreeFile

Open Trim(Filename) For Input As Filenum Len = Filelength

Line Input #Filenum, StringToGet

Close Filenum


Input_From_File = StringToGet

End Function

Function Input_Strings (Filename As String, ByVal recordlength As Integer, s As
Integer)
'this function is designed to read any number of sequential access
'strings from a file, the number of strings read depends on the
'value of stringnumber passed to it.

'declare variables.
Dim Thisfilesnum As Integer
Dim RequiredString As String
Dim SecRequired As String

'get the file handle variable
Thisfilesnum = FreeFile

'open the file
Open Trim(Filename) For Input As Thisfilesnum Len = recordlength

'get all the strings until the only string that is recorded is the
'string that we are after
Line Input #Thisfilesnum, RequiredString
Line Input #Thisfilesnum, SecRequired

'increment the stringnumber variable so that the next string from
'the file is gotten if required

If s = 1 Then
    Input_Strings = Trim(RequiredString)
    Else
    Input_Strings = Trim(SecRequired)
End If

Close Thisfilesnum
End Function

Function IsItemInListbox (ByVal InputItem As Variant, ByVal ListLength As Varia
t) As Integer
'Declare counting variable
'Dim i As Integer
'Dim X As Variant
'Dim y As Variant
```

```
'ACTORY.BAS - 13

''If ListLength = 0 Then
 'Exit Function
''End If

 'For i = 0 To ListLength - 1

''If Val(Trim(Toollists!List1.List(i))) = InputItem Then
  ''IsItemInListbox = 1
''  End If
''Next



End Function

.unction JobStatus (Filenam As String, Thisjob As JobRecord, Jobspending() As Jo
bRecord, ByVal Filenum As Integer) As Integer
 Declare variables
_im Filelength As Integer
Dim Currenttask As Integer

 ilelength = Len(Thisjob)

 pen Trim(Filenam) For Random Access Read Shared As Filenum Len = Filelength
 eDim Jobspending(1)

 et Filenum, 1, Jobspending(1)
 obStatus = Jobspending(1).Status
End Function

 ub OpenAndWrite (Filenum As Integer, Filename As String, ThisprocessesPost As P
ostrecord, ThisPostProcsarray() As Postrecord, FilesLength As Integer)

 im i As Integer
Dim recordlength As Integer
recordlength = Len(ThisprocessesPost)

 pen Filename For Random Access Write Shared As Filenum Len = recordlength
For i = 1 To FilesLength
 ut Filenum, i, ThisPostProcsarray(i)
 ext

 nd Sub

Sub openForInput (Thisfilesnum As Integer, Filename As String, recordlength As I
 teger)

Open Trim(Filename) For Input As Thisfilesnum Len = recordlength


End Sub

 ub openForOutput (Filenum As Integer, recordlength As Integer, Filenamed As Str
ing)
```

```
Open Trim(Filenamed) For Output As Filenum Len = recordlength

End Sub

Sub Output_To_File (Filename As String, Filelength As Integer, StringToPrint As
String)
Dim Filenum As Integer

Filenum = FreeFile

Open Trim(Filename) For Output As Filenum Len = 4

Print #Filenum, Trim(StringToPrint)

Close Filenum

End Sub

Sub OutPut_Two_Strings (Filename As String, Filelength As Integer, String1 As S
ring, String2 As String)
Dim Filenum As Integer

'define a freefile handle
Filenum = FreeFile

'open the file for output
Open Trim(Filename) For Output As Filenum Len = Filelength

'print the first string to file
Print #Filenum, Trim(String1)
'print the second string to file
Print #Filenum, Trim(String2)

'close the file
Close Filenum

End Sub

Sub ReadAndAddToList (Thosetools() As Toolrecord, Filesize As Integer, Filenum
s Integer)

Dim i As Integer

If Filesize = 0 Then 'there is nothing in the file so do not continue
Exit Sub
End If

'Redimension array to take whole file
ReDim Preserve Thosetools(Filesize)

'Get information from file and write it to the listbox
For i = 1 To Filesize
Get Filenum, i, Thosetools(i)
Toollists.List1.AddItem CStr(Thosetools(i).Toolsize) & "mm"
Next
```

```
ACTORY.BAS - 15


nd Sub

Sub ReadAndAddToList1 (Thosetools() As Toolrecord, ByVal Filesize As Integer, B
al Filenum As Integer)

Dim i As Integer
f Filesize = 0 Then 'there is nothing in the file so do not continue
xit Sub
End If

Redimension array to take whole file
ReDim Preserve Thosetools(Filesize)

Get information from file and write it to the listbox
or i = 1 To Filesize
et Filenum, i, Thosetools(i)
ef_Job.Toolsize.AddItem CStr(Thosetools(i).Toolsize) & "mm"
ext

nd Sub

ub ReadAndAddToList2 (ThisPostProcsarray() As Postrecord, Filesize As Integer,
ilenum As Integer)

im i As Integer

Redimension array to take whole file
ReDim Preserve ThisPostProcsarray(Filesize)

Get information from file and write it to the listbox and
add one extra empty position so that we can add a new
item if necessary.
r i = 1 To Filesize
et Filenum, i, ThisPostProcsarray(i)
efine_Processes.Def_CNC_Cont.AddItem CStr(ThisPostProcsarray(i).PostTyp)
ext

Get Filenum, (Filesize + 1), ThisPostProcsarray(Filesize + 1)

nd Sub

ub ReadAndAddToList3 (ThisPostProcsarray() As Postrecord, Filesize As Integer,
ilenum As Integer)
im i As Integer

Redimension array to take whole file
eDim Preserve ThisPostProcsarray(Filesize)

Get information from file and write it to the listbox and
add one extra empty position so that we can add a new
item if necessary.
For i = 1 To Filesize
et Filenum, i, ThisPostProcsarray(i)
ad_Cam.post_procsreq.AddItem CStr(ThisPostProcsarray(i).PostTyp)
Next
```

```
'Get Filenum, (Filesize + 1), ThisPostProcsarray(Filesize + 1)

End Sub

Function ReadAndAddToList4 (ThisPostProcsarray() As Postrecord, Filesize As Inte
ger, Filenum As Integer)
Dim i As Integer

'Redimension array to take whole file
ReDim Preserve ThisPostProcsarray(Filesize)

'Get information from file and write it to the listbox and
'add one extra empty position so that we can add a new
'item if necessary.
For i = 1 To Filesize
Get Filenum, i, ThisPostProcsarray(i)
Config_NewVMD1.Proc_List.AddItem CStr(ThisPostProcsarray(i).PostTyp)
Next

'Get Filenum, (Filesize + 1), ThisPostProcsarray(Filesize + 1)


End Function

Sub ReadToMemory (ByVal Filenum As Integer, Filenam As String, Thisstate As Opt
onstate, EndOfArray As Integer)

'Call ReadToMemory(filenum, Trim(Filenam), Thisstate, EndOfArray)


Dim i As Integer
Dim recordlength As Integer
'This particular procedure opens a file and reads the contents
'of that file to memory

recordlength = Len(Thisstate)
Open Filenam. For Random Access Read Shared As Filenum Len = recordlength

For i = 1 To EndOfArray
Get Filenum, i, OptionStateArray(i)
Next

End Sub

Sub WriteToFile (Filenum As Integer, NoOfToolTypes As Integer, MainToolist() As
ToolIdRecord)
Dim i As Integer


For i = 1 To MainToolist(1).recordlength
Put Filenum, i, MainToolist(i)
Next
End Sub

Sub WriteToFile1 (Filenum As Integer, Filelength As Integer, Thosetools() As Too
```

```
 ACTORY.BAS - 17

 record)


 im i As Integer


 or i = 1 To (Filelength + 1)
  ut Filenum, i, Thosetools(i)
 ext


 nd Sub

  ub WritetoFile2 (ByVal Filenum As Integer, ByVal Filelength As Integer, CadCam
 ray() As CadCamvariable)


  im i As Integer


  or i = 1 To (Filelength)
 Put Filenum, i, CadCamArray(i)
 ext


 nd Sub

 Sub WritetoFile3 (ByVal Filenum As Integer, ByVal Filelength As Integer, Jobspe
 ing() As Pendingjob)

 im i As Integer


 or i = 1 To (Filelength)
 ut Filenum, i, Jobspending(i)
  ext


  nd Sub

 Sub WriteToMemory (Filenum As Integer, NoOfToolTypes As Integer, MainToolist() A
   ToolIdRecord)
  im i As Integer

  f NoOfToolTypes = 0 Then 'If there is nothing in the file
 oOfToolTypes = 1
 oTo NextOne:
 nd If

'Redimension the array to take all of the file and make room
'for one more record
  eDim Preserve MainToolist(NoOfToolTypes + 1)

 extOne:
  or i = 1 To NoOfToolTypes
'Get all file info from file to memory
```

```
Get Filenum, i, MainToolist(i)
Next

End Sub

Sub WriteToMemory1 (Filenum As Integer, Thisfileslength As Integer, Thosetools(
 As Toolrecord)
Dim i As Integer


'Redimension the array to take all of the file plus an extra one for the new re
ord
ReDim Preserve Thosetools(Thisfileslength + 1)

NextOnes:
For i = 1 To (Thisfileslength + 1)
'Get all file info from file to memory
Get Filenum, i, Thosetools(i)
MsgBox CStr(Thosetools(i).Toolid)
Next

End Sub

Sub WriteToMemory2 (ByVal Filenum As Integer, ByVal Thisfileslength As Integer,
CadCamArray() As CadCamvariable)


Dim i As Integer


'Redimension the array to take all of the file plus one more
'record in case we need it.
ReDim Preserve CadCamArray(Thisfileslength + 1)


For i = 1 To (Thisfileslength + 1)
'Get all file info from file to memory
Get Filenum, i, CadCamArray(i)
Next

End Sub

Sub WriteToMemory3 (Filenum As Integer, Thisfileslength As Integer, OptionList(
 As OldOptions)

Dim i As Integer


'Redimension the array to take all of the file
ReDim Preserve OptionList(Thisfileslength)


For i = 1 To (Thisfileslength)
'Get all file info from file to memory
Get Filenum, i, OptionList(i)
Next
```

```
End Sub

Sub Writetomemory4 (ByVal Filenum As Integer, Thisfileslength As Integer, Thisa
ray() As JobRecord)
On Error Resume Next
Dim i As Integer


'Redimension the array to take all of the file
ReDim Preserve OptionList(Thisfileslength)


'get the current task data to memory
Get Filenum, Thisfileslength, Thisarray(Thisfileslength)


End Sub

Sub WriteToMemory5 (ByVal Filenum As Integer, ByVal Thisfileslength As Integer,
Jobspending() As Pendingjob)

Dim i As Integer


'Redimension the array to take all of the file plus one more
'record in case we need it.
ReDim Preserve Jobspending(Thisfileslength + 1)


For i = 1 To (Thisfileslength + 1)
'Get all file info from file to memory
Get Filenum, i, Jobspending(i)
Next

End Sub

Sub WriteToMemory6 (ByVal Filenum As Integer, ByVal Thisfileslength As Integer,
ThisQueue() As Factqueue)

Dim i As Integer
If Thisfileslength = 0 Then
Thisfileslength = 1
End If


'Redimension the array to take all of the file
ReDim Preserve ThisQueue(Thisfileslength)

For i = 1 To (Thisfileslength)
'Get all file info from file to memory
Get Filenum, i, ThisQueue(i)
Next

End Sub
```

```
Sub WriteToMemory7 (ByVal Filenum As Integer, ByVal Thisfileslength As Integer,
Jobspending() As Pendingjob)
Dim i As Integer


'Redimension the array to take all of the file plus one more
'record in case we need it.
ReDim Preserve Jobspending(Thisfileslength)


For i = 1 To (Thisfileslength)
'Get all file info from file to memory
Get Filenum, i, Jobspending(i)
Next

End Sub
```

```
NEWLYQUE.BAS - 1

'All variables must be declared
Option Explicit
'All arrays start at position 1
Option Base 1
'State that text comparison is not case-sensitive
Option Compare Text


'***** QUEUEING USER DEFINED VARIABLE TYPE DECLARATIONS ***

'Incoming Queue user defined variable declarations
Type Inqueue
JobName As String * 20
InFoFilename As String * 20
Bid_No_Parts As String * 10
BidSetUpTime As String * 10
BidLoadUnLoadTime As String * 10
BidCycleTime As String * 10
End Type

'Define V.M.D. Tool Specification user defined variable
Type ToolSpec
'Toolname As String * 40
ToolTypeSize As Single
NoOfToolsNeeded As Integer
End Type

'Define node\Cell controller config user defined variable
Type Nodevmdconfig
Recordsize As Integer
Nodename As String * 4
VMDNo As Integer
Recordposi As Integer
End Type

'Noderecord required in Cell Configuration
Type Nodesrecord
Recordsize As Integer
Nodename As String * 6
End Type

Type Noodesrecord
Nodename As String * 8
Index As Integer
End Type


'Outgoing Queue user defined variable declarations
Type OutQueue
JobName As String * 20
InFoFilename As String * 20
Bid_No_Of_Parts As Integer
BidSetUpTime As Integer
BidMachTyp As Single
BidCycleTime As Integer
```

```
BidTool1Required As Single
BidTool2Required As Single
BidTool3Required As Single
BidTool4Required As Single
BidLoadUnLoadTime As Integer
Force As Integer
End Type


'Declare Ingoing Queue pointers
Type InQueuePointers
InStartOfQueue As Integer
InFinishOfQueue As Integer
End Type

'Declare Outgoing Queue Pointers
Type OutQueuePointers
OutStartOfQueue As Integer
OutFinishOfQueue As Integer
End Type

'Number of tasks in outgoing queue variable
Global Outqueueno As Integer


'Queue handle Variables
Global Position As Integer
'*********************************************************



'***** Current Queue Cost Calculation Variable *******
Global Cost As Integer
Global Lo_GCost As Double
```