MMPHIL 6

A Gaming Simulation of Manufacturing Organisations

by

Juan Ignacio Igartua

A thesis submitted in partial fulfilment of the requirements of the Council for National Academic Awards for the degree of Master of Philosophy.

October 1991

Nottingham Polytechnic

ACKNOWLEDGEMENTS

Firstly, I would like to express my gratitude to my supervisor Dr Graham Tranfield for his advices, constant support, and guidance throughout the fulfilment of this project and specially in the writing process.

Secondly, I would like to thank Justino Unamuno, Iñaki Elguezua, Jesus Santa Cristina, and Juan Galardi for their help and friendship since all of us met in England to extend our knowledge in the production area.

I would also like to thank Phil Moore for his technical help during the completion of this work.

Special thanks to the people in the office Farhad Fassihi, Steve Wood, and Alex Seiler, who have all become very good friends.

Most of all, I would like to thank my parents and two sisters for their continual support, patience and the sacrifices they have made throughout my education. At this stage, I would also like to remember my grandfather, who unexpectedly died during the study of this work.

ABSTRACT

Manufacturing organisations are inevitably divided into individual departments at some level. Each of these departments will have their own objectives and the success of the organisation depends on the extent to which these departmental objectives are in line with the overall objectives of the organisation. They must coordinate their efforts towards the objectives of the whole organisation, forgetting their departmental rivalries. Sadly this is not the case in many western manufacturing companies.

Many companies would therefore like to realign their departmental objectives and an important step in this process is to educate their staff. They must understand how their decisions affect other parts of the company and through this the overall company performance. This is inevitably a dynamic and cross functional process, as decisions are being made throughout an organisation in response to a continually changing environment.

Therefore, education can no longer be seen as a static and independent process, where people only understand how they work, and how others work in a stand-alone basis. What is needed is a new and more integrative approach, where people understand not only how they work and why, but also how they affect other areas of their manufacturing organisation, in other words how their policies and performances fit within the policies and performances of the whole organisation.

Computer games appear to be appropriate tools for that integrated educational approach. Players are not only able to experiment with technical variables, but also they are able to experience a fundamental aspect of organisational life, that of bargaining and negotiation in the decision-making process. They are faced with the inevitably interdepartmental rivalry so common in organisations, and with the conflict between this rivalry and the need to cooperate in a company basis.

Work therefore took place at Nottingham Polytechnic to develop a dynamic, multi-user, computer based gaming-simulation which was designed to highlight the interactions between departments.

TABLE OF CONTENTS

1 INTRODUCTION

To compete in the world markets of the 1990's it is necessary to do more than to simply seek cost efficiency as manufacturing companies have done in the past.

It is evident that market conditions have become more dynamic, more global, and more customer driven. Product price is also no longer the main factor affecting business performance. Other non-price competitive factors such as quality, design, delivery, and customer service have become equally, if not more important.

Moreover, all these market factors are dynamically changing at a rate that forces manufacturing organisations to change themselves into more flexible and dynamic organisations. They are compelling manufacturing organisations into a continuous changing process.

Manufacturing managers have realised there needs to be continuous improvements in quality, reduced cost, reduced inventory, shortened flow times, and improved customer relations. They believe these changes must be implemented to maintain or improve their competitive position.

As it is clearly stated by Hayes, Wheelwright, and Clark [1], in the book *Dynamic manufacturing, creating the learning organisation*, **"It is manufacturing management's responsibility to change."**

Indeed as an educational video [2] states in the opening minutes:

"Change - timeless as life itself, certain as tomorrow's sunrise - change and our response to it are the driving force in today's manufacturing environment. Sometimes simple, often complex, change is inevitable..."

The implementation of new technologies and new management techniques has been companies' first response to this complex changing environment in order to regain or obtain the successful competitiveness needed in this customer driven market.

But what companies will have to learn, if they have not already done so by experience, is that there is no such thing as a quick technological fix. New technology is important but is not the answer, it must not come first [3]. In the majority of cases, it is people who transform raw materials into profitable high quality products. New technology and techniques, as well as organisational change must be shaped by this fact, and by the organisational requirements and needs to achieve continuing improvement.

People and the question of how best to help, coordinate, and support them, will be some of the central manufacturing issues of the 1990s. Technology will still be important, but companies can no longer afford to use technology to minimise the role of people in the organisation because they are, in many cases, the key to business success.

Companies will have to make and implement a strategic decision to put people first, to build organisations that support these people and their functions, and to use technology as a means of making more effective use of peoples' skills and abilities. The keywords are empowering, teamworking, reduced hierarchies, decentralised control and decision making, and coordinated organisational structures.

However, before any of the above can be achieved, companies will have to "fight" against organisational problems, i.e. outdated attitudes, resistance to change, over specialised narrowly based roles, vested interests, power relationships, conflicts, company politics and cultures. But what is more important, they will have to change the outlook of company employees from that of departmental performance defenders to that of coordinated individuals and functions engaged in a common goal of ongoing improvement in company competitiveness, profitability and business performance.

To carry out this first task manufacturing organisations will have to understand how each of the functions' decisions affect other functions of the manufacturing organisation and through that, the overall company performance. This will inevitably be a dynamic educational process, as in real life, that could be carried out with the aid of computer based educational tools, more specifically computer based games and simulations.

This thesis is therefore an evaluation of computer based gaming simulations as tools for education in this area.

It starts with a review of the importance of coordination among the areas of a manufacturing organisation working in a changing environment.

Chapter 3 then reviews how computer games are used as educational tools in a variety of dynamic environments and in particular looks at the different approaches to education in relation to manufacturing management.

The main part of the work for this thesis was then the construction of a computer based gaming-simulation specifically designed to address the problems of coordination in manufacturing organisations. A paper reporting the progress of this work was presented at the seventh National Conference on Production Research [97]. A complete copy of this paper can be found in Appendix E.

The methodology for developing the game is described in chapter 4 and its software design is described in chapter 5.

The conclusion then discusses the effectiveness of this game and the future of gaming-simulations in the manufacturing area in general.

2 MANUFACTURING ORGANISATIONS: COORDINATION AND EDUCATION

2.1 INTRODUCTION

Managing organisations is of particular interest in this century. In the past large organisations were restricted to basically three types of organisations - governments, armies and religions. But this is not the case any more. From the beginning of the nineteenth century new types of large organisations have arisen, the industrial and services companies being the most clear examples [4]. Nowadays, the number of industrial companies that exceed ten thousand employees is probably at least two orders of magnitude greater than the handful of such large companies in existence during the previous century.

In the coming sections of this chapter the existence of two very important issues manufacturing organisations will have to address in order to succeed in today's dynamic environment will be shown: **coordination and education.**

In order to have a better picture when explaining the importance of these two elements, it will first be explained how manufacturing organisations are structured and the different existing alternatives.

In a second step, the importance of coordination in a changing environment, and the different ways it can be achieved will be explained. The relation between the way manufacturing organisations are coordinated and their associated success is also identified.

In a third step, the links between **coordination** and **education** will be established, and hence the importance of **education** in today's manufacturing organisations will be highlighted.

2.2 THE STRUCTURE OF MANUFACTURING ORGANISATIONS

2.2.1 INTRODUCTION

Manufacturing organisations have, since their development, been structured in a **hierarchical pyramid of command** as shown in figure [2.1]. The reasons for the utilisation of this kind of structure must be found in human's span of control, individuals are only able to directly handle relatively few people compared to the number of people in the organisation. This way of organising leads to problems related to the fact that the transactions needed to provide products and customer services are not hierarchical, they are horizontal and not vertical as it is shown in figure [2.2].

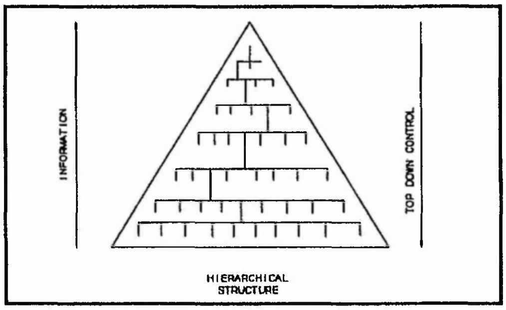


Figure 2.1 Hierarchical Structure

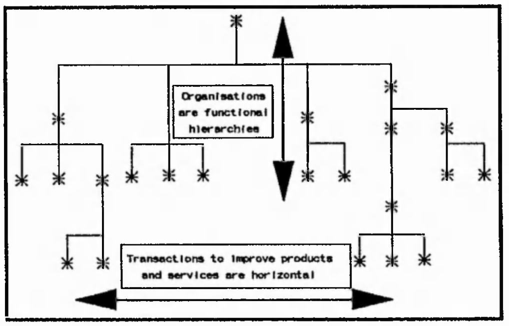


Figure 2.2 A fundamental problem.
Taken from Plowan [93]

Another element inherent in all kind of organisations, including manufacturing ones, and somehow related to the hierarchical pyramid of command is the **specialisation of labour.**

In his 1776 book, **The Wealth of Nations**, Adam Smith explained how he was able to increase the productivity of a group of pin makers by more than a thousand-fold through specialisation (division) of labour. **Specialisation of labour** means the division of a complex job into simpler tasks so that one person or group may carry out only identical or related activities [5]. The purpose of organising is, mainly, to improve productivity. Through specialisation of labour, it is possible for members of the manufacturing organisation to concentrate in a single area, resulting in increasing output. In nearly all manufacturing organisations, most of the work activities are of a specialised nature.

Another element that appears in conjunction with the idea of **specialisation of labour** is the one of **size**. Manufacturing organisations were at the start small ones and easy to manage. In a fledgling company all staff pitch in willingly and do each each other's jobs if need be. Need creates a vital team spirit. But as the company grows it becomes more complex to manage, and due to the specialisation of labour it becomes functionally differentiated.

These two reasons, **specialisation of labour** and **size** has caused manufacturing organisations to structure their organisations in departments, each of them involved with a specialised issue.

The process of dividing organisations into departments is defined as a way of grouping related work activities into manageable units in order to contribute to a more effective and efficient use of the manufacturing resources available.

The most common departments to be found in a manufacturing company are: Product Design, Production Planning, Production Control, Purchasing, Marketing, Finance, and Personnel.

Based on the same elements, manufacturing organisations have different ways of creating their departmental division, each of which is based on a particular specialisation element (function, product,..). These different ways of breaking down into departments are [5]:

- By function.
- By product.
- By customer.
- By geographic territory.
- By project.
- By a combination approach.

2.2.2 FUNCTIONAL STRUCTURE

Although the existence of all previously mentioned possible alternatives, most of manufacturing organisations are divided, at some stage, in a functional way, where different activities are performed by different functions.

A management function is defined as sets of closely related management tasks, which require similar skill for their efficient performance.

These functions form a basic classification of the science of Production Management (figure [2.3]). Each function contains sets of related tasks, each of them requiring planning, direction and control. Planning and control tend to be special to each function. The idea of direction is more general by nature.

FUNCTION	TYPE OF TASK	CONTROLS	INPUTS	OUTPUTS
PRODUCT DESIGN	Plans final form of product	Quality Control	Ideas, Market research, R&D.	Parts list, Drawings.
PRODUCTION PLANNING	Plans how product is to be made	Process Control, Maintenance.	Parts list, Drawings, Sales forecasts	Make or Buy, Plant list, Layout, Routes, Operation times, Tooling.
PRODUCTION CONTROL	Plans material supply and processing activities	Progressing, Loading, Inventory Control	Sales prog., Parts list, Plan list, Routes, Op. times. Make or Buy.	Programmes, Orders, Purchase delivery, Schedules.
PURCHASING	Finds sources Supply contracts	Purchase, progressing.	Purchase delivery. schedules, Make or Buy.	Purchase Orders.
MARKETING	Finds/develops markets, Sales and distribution	Sales control.	Sales orders	Sales records, Sales programmes.
FINANCE	Plans investment, profit and cash flow.	Budgetary Control, Standard costing.	Annual prod. progs., Invoices in/out, Bank payments in/out.	Budgets, Accounts, Balance sheet, Profit and Loss Acc.
PERSONNEL	Plans employment, conditions welfare, training, promotion.	Merit rating, Attendance.	Hired, Fired, Promoted, Retired.	Employees list, Conditions of employment.
SECRETARIAL	Plans communications and data processing.	Data control.	Production system design.	Software.

Figure 2.3 Management functions.
Taken from Burbidge [34]

This division of manufacturing organisations into separate functions was developed during the industrial revolution and based on the "scientific management" philosophy, advocated by Frederick Taylor, whereby functional experts each qualified in a different area reported to a general manager [6].

By grouping related functions, manufacturing organisations form their departments on the basis of specialised activities such as finance, marketing, production, engineering, personnel, and so on, as seen in figure [2.4].

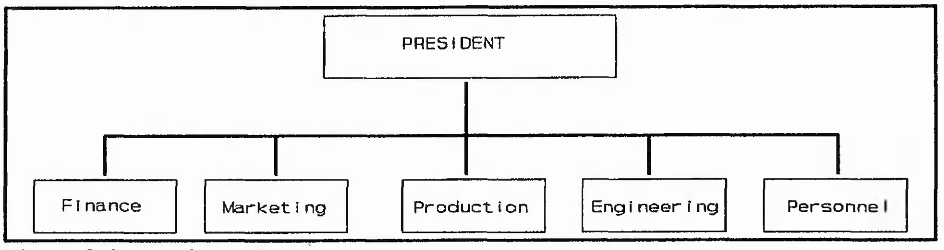


Figure 2.4 Functional departments.

This division of manufacturing organisations by function, although useful in environments where technical efficiency and quality are important, creates certain problems. Employees in specialised departments may become more concern about the functioning and performance of their own department rather than about the overall company's.

Responsibility for a particular functional or other organisational unit tends to narrow the focus of those who manage them. The resulting parochial behaviour can lead to unfortunate outcomes, affect a firm's competitiveness, unless the integration across those separate units is well managed. To paraphrase Winston Churchill, "*Each person has only to do his duty to wreck* *the world.*"

An example of this assertion is the conclusion reached by a survey conducted in the UK in May 1990, called Managing change in the 90's [7] :

"*Compartmentalisation in an organisation creates diverseness and breeds conflict which is a significant obstacle for change. Some managers' clear awareness of this does not seem to have had a major impact on the types of initiatives planned for the future.*"

Because of all this and taking into account the usually conflicting purposes of various departments in a functionally based structure, managers must ensure that an effective means of coordination exists among all the departments taking part in the organisation's functioning.

Note: Some more explicit examples of the existing functional conflicts and their financial influences can be seen in Appendix A.

2.3 COORDINATION IN MANUFACTURING ORGANISATIONS

With manufacturing organisations structured in a functional way, coordination of those different functions is of vital importance for the success of the company.

An important contribution on the matter of coordination and its importance is addressed by Galbraith is his theory of co-ordination and control [8], where he talks about the strategies organisations can use to coordinate and control the people and processes who comprise them. His starting point is the assumption that for an organisation to perform satisfactorily it must be able to coordinate the activities of its various elements or subunits, be they individuals, groups, or departments, in an effective way. The ability to bring different specialists together is and will become a critical differentiator, determining success or failure.

Another example corroborating the importance of coordination, in general terms, is the one expressed by Kiyoshi Suzaki [9], where he compares a well-performing orchestra to a well-performing factory. Each musician has skills that are well developed through extensive training. Each instrument has its one tone and is tuned and well cared for by each player. But there is also a conductor who interprets the music score and coordinates the efforts of all players into a polished performance. Only when these elements fit together does the whole orchestra produce an outstanding performance with beautiful harmony, tone and rhythm. No single element should be missing. No single player or instrument should be out of tune. To emphasise this latter idea a diagram showing the main organisational characteristics of conventional and progressive companies is shown in figure [2.5].

The "activity chain" concept is another useful approach for achieving coordination [10]. This concept is based on the importance of the integration

	Conventional Company	Progressive Company
Operational Characteristic		
Setup time	Long	Short
Lot size	Large	Small
Inventory	Large	Small
Floor space	Large	Small
Transportation	Long	Short
Lead time	Long	Short
Defect rate	High	Low
Machine trouble	High	Low
Organisational Characteristic		
Structure	Rigid	Flexible
Orientation	Local optimization	Total optimization
Communication	Long chain of command	Open communication
Agreement	Contract-based	Trust-based
Union focus	Skill-based	Company-based
Skill base	Narrow	Broad (flexible)
Suppliers	Many	Selected few
Education/training	Insignificant	Significant

Figure 2.5 Characteristics of Manufacturing in a Conventional and a Progressive Company. Taken from Suzaki [9]

between functions and departments. The concept of "activity chain" denotes a continuous chain of activities associated with the dealing of an essential task of an industrial enterprise, such as product development, production flow, and customer orders. An activity chain will cut across functions and departments, as is illustrated in figure [2.6], as opposed to traditional vertical communication. This "activity chain" concept also fosters a Production Management Concept that will give a coherent picture of the way in which production is to be managed. This latest concept includes mutual agreements between parties involved in the manufacturing process, such as sales, engineering design, production engineering, and the various production units.

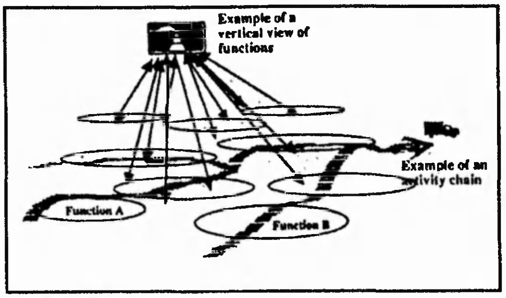


Figure 2.6 Activity Chain.
Taken from Frick [10]

In a broader way, Schonberger in his book "Building a Chain of Customers" [11], emphasises that the principles of world-class performance must now be applied to the companies as a whole, replacing stand-alone departmental strategies. Quoting Schonberger: "*The whole organisation, every single part of it, has to be part of the improvements*".

In a more specific context many other people, and leading them Burbidge [12,13,14], have addressed the importance of coordination in manufacturing organisations, by continually pointing out the benefits from breaking down shop floor activities into product based rather than functional elements. A thought that could be applied to all areas of a company. This way of dividing into departments will help to re-integrate the organisation in a new way where the importance of the coordination factor will decrease, due to the inherent coordination factor attached to this structure's nature. This product-based structure will, in his opinion, be needed if manufacturing companies want to confront the market of the 1990s where customer service and everything it implies are the competitive elements.

A related matter is illustrated in the paper written by C.A. Voss et al. [15], where they clearly identify the nature of the integration and coordination provided by the programme management organisation as vital to the achievement of the fast and effective product development needed to compete in the actual competitive market.

Another study complementing this view was by Voss [16] himself. In his study of success and failure in the implementation of advanced manufacturing technology, he found that organisational integration, coordination, was likely to be associated with successful implementation.

Therefore, the importance of coordination can, undoubtedly, be seen in the way implementation of new technology in manufacturing organisations has been carried out.

2.3.1 TECHNOLOGICAL IMPLEMENTATION

Many specialists think that the implementation of new technology, and change (in a broader context), fail to succeed because the human element is ignored. Market demands for shorter delivery times and improved quality have created a need for better integration of the many activities involve in modern manufacturing. At the same time new technology provides potentially powerful means. Nevertheless, experience indicates that a much broader approach is needed, including many aspects and many parts of the industrial organisation, in order to fully utilise modern technology to meet the competitive challenges.

As it is clearly stated in the article about MRPII "The price of ignorance" [17], the success of the companies implementing new technologies does not lie on the technological tools themselves, but in a philosophy that recognises the interdependence between activities and the importance of integration throughout the organisation. That philosophy, must not only involve new manufacturing techniques, but new techniques of management and of performance measurement.

Using "revolutionary" technologies requires a similarly radical degree of organisational change along a number of dimensions, including the skills profile, the functional and hierarchical structure, the philosophy of management and control, and the underlying culture of the organisation [18]. It is the lack of that parallel organisational change, that has caused the failure of implementations, through a persistent resistance to change as is shown in figure [2.7].

Furthermore, those organisational changes that should have been implemented along with the technological ones might fail or have already failed, (if tried to implement them), not because a lack of structural merit but because either they have not been understood or they have not been adequately coordinated. Communication, coordination and integration are the three key elements in any change, and the more manufacturing organisations understand this fact, the more effective and successful those changes will be [19].

Pressures to change →	← Resistances to change
Market environment	Lack of awareness and understanding
Technology push is catalysing change - firms have to adapt to get best out of their investments	Lack of skills
	Organisational inertia - adapt the technology to the existing organisation
	"Sedimented traditions"
	Lack of alternative models
	Supplier pressure to shape organisation to fit their technology package
	Politics - defending positions, preserving power
	Culture - "the way we do things around here" - made up of individual attitudes, beliefs, group norms, established traditions

Figure 2.7 'Force-field' analysis of technology change.
Taken from Levy [18]

For example, A.t. Kearney [20] report in the conclusions of their survey of CIM:

..to be effective as a means of achieving competitive advantage, it (CIM) must be introduced as part of a new strategic thrust in the organisation.

..it is easier to install and make operative a new software application than to introduce a cultural change which accepts shared data, multi-functional decision-making and new organisation structures.

Similarly Barrar et al. [21] in their study of MRP conclude:

Success in implementation (of MRPII) therefore, must recognise...changes in ideology, organisation and management process.

Also Buckingham and Bessant [22] :

..only by moving beyond seeing technology as a substitution device can organisations develop the flexibility necessary to fully benefit from integrated technologies.

The successful implementation of any kind of change in manufacturing companies is not based on the existence of technologically advance islands, totally independent and scattered in a sea of confusion, but on the context of a total business enterprise. A business enterprise where there are not separate elements but links in a chain, a business enterprise where all the different functions and departments are coordinated and integrated towards an overall, multi-disciplinary and ever improving compromise ("Concert building" [23]), in which neither technology nor organisational structure or process has been static, but has instead been in a state of constant evolution.

Innovation must be focused on collaborative efforts within and outside manufacturing organisations, identifying interdependent activities across the value chain that could be used to drive the organisation to a better competitive spectrum, where the customer needs are really fulfilled.

Yet another important illustration of the importance of cooperation in the implementation of Total Quality Management (TQM), an example of organisational change, is compactly summarised in one of the vertices of the Joiner Triangle: "All One Team" [24,25]. This coordination, cooperation, is a foundation stone of the Deming philosophy.

Concluding, it can be said that whichever the organisational approach taken (e.g., "activity chain", product based break down,..), manufacturing organisations will have to continue facing their challenges, improvements, and changes, in a departmental framework. The recognition of this fact, implies the acceptance for the need of coordination of all the different areas, and departments involved in the functioning of the company.

2.4 DEALING WITH COORDINATION

2.4.1 INTRODUCTION

Every manufacturing organisation struggles, intentionally or intuitively, to find a way of coordinating its functions. Therefore, in the following sub-sections some stereotype solutions used in different organisations will be reviewed. It should be stressed that no organisation uses just one solution. Actually, in Goldratt's opinion, everyone uses a blend where one solution is dominant but where at least traces of other solutions exist [4].

2.4.2 THE ARMY SOLUTION

Probably the most known and ancient solution is the army one - discipline. Discipline is a way of coordination that attempts to minimise the distortions introduced when interpreting guidelines and detailed instructions. For this sort of coordination to work, it is necessary for the lower ranks to understand exactly what is required by each order that is issued. For example, the meaning of the order "march" is not simply to walk forward, but to hold yourself upright, swing arms, hold the rifle in a specific way, etc..

In this context, the task of the lower management ranks is basically to carry out the current orders according to the respective predetermined process. This idea is summed up by the army expression "*do it by the book*".

This methodology is an attempt to neutralise the lower management levels from making any significant decision on their own, leaving the decisions to the very top of the pyramid. This discipline method tries to avoid the need to rely on local optimums. This model is the one advocated by Frederick Taylor in his "scientific management" philosophy [26].

Although this approach has a positive short term influence in manufacturing organisations (i.e. things are done) due to the need for synchronisation, it has an undesirable long term effect on the organisation (i.e. people are not involved). That lack of involvement will create conflict and unnecessary bureaucracy, that will make the manufacturing organisation unable to cope with the dynamic environment faced by manufacturing companies.

2.4.3 THE RELIGION SOLUTION

An almost antithesis of the army method is the approach used by religion-based organisations. Organisations based on the religious model do not use a predetermined, strict translation between a guideline and instructions. Rather they have developed a very detailed code of what is right and what is wrong. This code is the connection between a situation and the appropriate guideline rather than between a guideline and a detailed instruction, lower levels of the organisation can decide what to do without waiting for a decision to come from the top.

This method tries to give maximum autonomy and decision power to the lowest possible rank in an organisation, it minimises the problems for coordination because every element knows how to perform to achieve the purpose of the whole organisation.

Although this seems to be a good approach, it has some problems when dealing with organisations broken down into departments that have to synchronised mutual efforts in order to carry out their activities.

2.4.4 THE GOAL DICTATES THE SOLUTION

The basic reason for the previous two different approaches is probably based on the means that have to be used in order to achieve each organisation's goal. The goal of a religious based organisation is achieved when each individual behaves separately in a certain way. The goal of an army based organisation, however, can only be accomplished through a synchronised effort of many individuals.

Manufacturing organisations suffer from the army's basic problem even though usually not to the same degree. Based on their departmental divisions, the only way for manufacturing companies to achieve their goals is by the coordination and synchronisation of the different elements inside them.

Nevertheless, manufacturing organisations have always been aware of the advantages of the religious based approach and have tried to use this methodology by trying to define rules, **measurements**, that are supposed to define what is "right" and what is "wrong". Manufacturing organisations have continuously tried to find the balance between the army based approach and the religion based one.

The definition of general goals, and of measurements that will be used to assess how the different elements of the organisation influence that goal, will be the way ahead for companies seeking for coordination.

2.4.5 EXAMPLES OF APPLIED COORDINATION

Many Western manufacturing organisations have balanced their approach towards the army method, while Japanese companies have tend more towards the religion method.

Therefore, in the next two sections two limited examples of these two approaches will be shown, looking at the way coordination is respectively understood in Western countries and Japan.

2.4.5.1 WESTERN COUNTRIES' APPROACH

The Western countries' approach towards coordination is very much based on the army discipline and in the use of techniques as an strategy to gain effective coordination.

Western countries have been emphasising the technological aspects of coordination as a way for its achievement rather than other existing aspects (e.g. organisational,..). They have tried to achieve coordination by implementing techniques from top to bottom. They have trained people to carry out tasks rather than help them understand what is happening and why is needed to happen. They have tried to implement techniques (e.g. OPT, KANBAN, SPC,..), especially the ones used in Japanese manufacturing, but looking only to the tools provided by them rather than to the general existing philosophy and inherent overall understanding behind them.

This approach has, however, turned out to be a failure in many cases. There are many written examples that refer to failures when trying this approach in Western companies. A clear example of this fact is the study conducted by A.T.Kearney [20]. This study of manufacturing companies in Great Britain, Japan, the United States and Western Europe found that when it comes to gaining benefits from new technology, Britain is at the bottom of the league table. That study also claims that of the £ 1.9 billion spent on factory automation each year in Britain, some £ 600 million is wasted.

As it is concluded by D.M. Lascelles et al. [27] :

"Managers tend to seek instant solutions; techniques are often seen as an end in themselves, giving little proactive thought to the introduction and use of such techniques within the overall framework of managing improvement".

This frustrating experience is not a new one, having been perceived all through the 1980s. Some of them are also refer in an article by Paul Levy et al. [18]. There are some other interesting comments from Andrew Owen et al. [28] and Brenton R. Groves [29].

2.4.5.2 JAPANESE'S APPROACH

Japanese companies have approached coordination from a more general point of view. First, they have reduced their hierarchies, flattening the hierarchy as shown in figure [2.8]. This supposes that the decisions are not taken by single people but by groups, multi-disciplinary teams. In this team system, communication and decision-making occur bottom-up from a web of fellow workers, instead of top-down as in an army hierarchy. In this integrated environment, reward comes from empowering others, not by climbing over them.

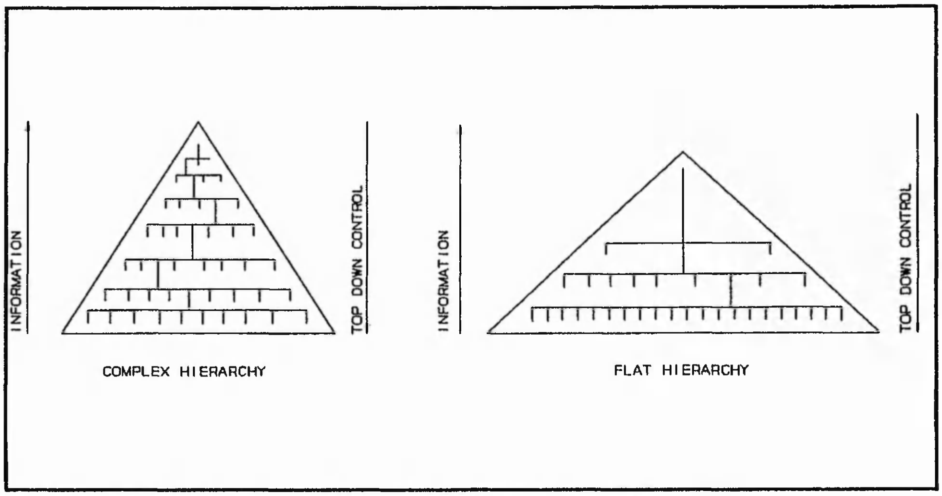


Figure 2.8 Flat hierarchy

This former idea is the one that has made Japanese organisations so successful. They see their success as the successful management of the web of highly dependent relationships existing in manufacturing organisations [30]. They emphasise harmony and cooperation among the different elements rather than individual functions and responsibilities. Each member or function is expected to do whatever is believed to be most important for the fulfilment of the goals of the collectivity at any given moment in time [31].

A second element is their view of techniques as a tool, but not as a solution to coordination. Japanese manufacturing organisations base their coordination on organisational as well as technological aspects [32]. They argue that successful coordination is achieved when the technical and human sides of the change process are jointly optimised. It is this general vision of coordination and the techniques used for it that have helped them to successfully implement philosophies like JIT, TQM,..

If there is a "secret" to Japan's success, that one lies in the continuous improvement process generated by a whole system, and not, as some Western counterparts have assumed, in specific and independent parts of that system.

2.4.6 TOWARDS THE "GOAL DICTATED" COORDINATION APPROACH

2.4.6.1 INTRODUCTION

Among the powerful forces influencing the coordination performance of manufacturing organisations, three emerge as outstanding. First is **organisational hierarchy**, second is **specialised work** and third is the **reward** system [33].

Based on the fact that the improvements in manufacturing organisations must be carried out in a hierarchy and specialised predetermined structure, there is one only way ahead towards successful coordination. This breakthrough implies the definition of a reward system, in other words, the definition of global objectives towards which the local ones should be focused. Inherent to this concept is the need to understand the dynamics and interactions of these objectives throughout the organisation. That process of understanding should involve the people that define the objectives as well as the people that try to achieve them.

2.4.6.2 DEFINE OBJECTIVES

The importance of the definition of objectives for manufacturing organisations is very much supported by people like Burbridge and Goldratt. They think that for coordination to work in a manufacturing organisation, a common goal and a way to measure its achievement has to be found. This means, that there is an implicit need to clarify how and to what extent areas, functions or departments contribute positively to the global success of the organisation.

Burbidge et al. [34,35], for example, thoroughly discuss the integration process and its importance within or between management functions in industrial enterprises. One of the conclusions that is derived from one of the studies [34] about the integration across functions is that : "*The most important need for interfunctional integration is the integration of goals*". This assertion is based on the fact that one function inevitably affects the conditions for the operations in other functions, and therefore stresses the need for focusing on the motivation to perform activities which will be beneficial for the whole organisation.

Also Goldratt [36] presents the concept of the inherent conflict. The inherent conflict states that in many cases "*the objectives of local areas are in conflict with the objective of the global organisation*", and that the way out for it, is to look for a situation where "*the objective of a local area is to contribute positively to the objective of the entire organisation*".

He also mentions the functional problems encountered when trying to apply his own "Theory of Constraints" philosophy to one function, stating : "*The lesson today is clear. Before any function can go on an ego trip, demonstrating and waving results (and by that digging its own grave) - before any function can start individual improvements, all functions should decide together in a common way.*"[37]

To summarise:

The most important element for the accomplishment of a successful coordination, integration, is the definition of a common global goal towards which all the different existing elements of the factory ,local areas, should

be directed to.

Another element related to the definition of objectives is the control system being used to measure them. This control system (rules, measurement) will have to be established to be good not only at one level or for one function, it will have to be appropriate for all management levels and for all the various tasks that managers carry out.

Deming, for example, in his study of the theme of cooperation [24,25] becomes aware of the barrier which the merit rating forms to real change and to coordination, as an example of change. The worst case is that of the effect of the performance appraisal constrained by a fixed distribution where it becomes necessary to put somebody else under in order to get a higher rating.

Some other people have also demonstrated, that the importance different functions or departments of a manufacturing company give to a certain performance measure (cost/price structure), a rating form in itself, varies greatly as it can clearly be seen in figure [2.9].

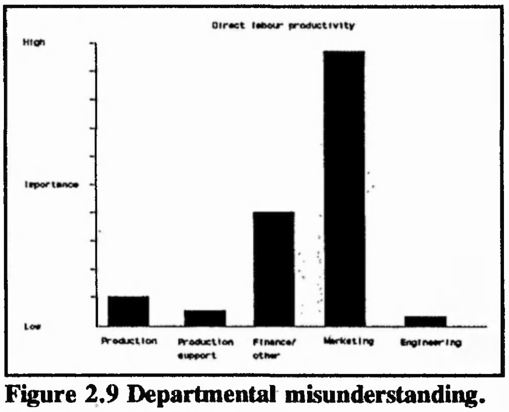


Figure 2.9 Departmental misunderstanding. Taken from Vollman [38]

There are also other references of traditional methods of accounting and finance, being seen nowadays as archaic [38,39]. The writers state on those articles that the existing methods can no longer be used in an industrial environment where so many things have and are changing as a necessity, mainly because those methods are in themselves one of the reasons for making that change fail.

Therefore, there is no point in trying to achieve common goals in an organisation that rewards its functions based on local optimums. Functional performance measures can no longer be seen as local measures in isolation, because they are not and they have never been. For example, customer service and delivery performance measures are frequently used. A manager can perform extremely well on these measures if he is not also responsible for inventory investment. Purchasing agents can score well on the cost dimension if not held accountable for inventory carrying costs, vendor lead times, or quality. Marketing personnel can perform well on total sales if not responsible for inventory investment or shop disruptions caused by continual expediting and overtime.

Throughout these previous ideas the elimination of local performance measures are not being advocated. But what is encouraged, however, is the analysis of the impact of the local measures on the more important global measures of long-term profit, market share, etc. Also it is seen as a need to study the impact that these measures have in other functional areas and production departments.

So, the point is clear, the system of measurement and reward has to change. It can no longer be based in local measures, individual results, confronted against each other and established in isolation, but in a global system of performance measurement were it can be established how functions, and different areas, influence the whole performance, objective, of the manufacturing organisation.

Attached to this idea is the need for understanding the whole of the organisation, the objectives (global,local) and the dynamics and interactions of manufacturing organisations.

2.5 THE ROLE OF EDUCATION

2.5.1 INTRODUCTION

The importance of the educational process in the success of manufacturing organisations, was addressed throughout the 1980s as it is explained by Marilyn M. Helmes [40], who also concludes that behaviourial and human relations concerns will continue being of primary importance for the successful implementation of change and that commitment to substantial training and education will be needed.

2.5.2 EDUCATING FOR CHANGE

These introductory comments are not isolated, there are many people who emphasise the huge importance of education and training as a way to introduce effective change at all levels and areas of the manufacturing organisation. When they contemplate the introduction of change, they also see the need to understand and learn, the nature of the change and, in particular, its implications on the corporate culture and values.

The reason for this consensus on the role of education is based on the fact that change can not be achieved without people. And people need to learn and understand, they have to believe that they are in fact the initiators of the change, and this can only be secured by **communication** and **training.** In fact one leads to the other. In other words change is synonymous with learning. A company in which planned proactive change is the norm is, by definition, a company in which training and learning is inherent and continuous.

As it is stated by Alan Marsden [41] the first step then is to communicate. Make everyone in the organisation aware of the impact of the change process upon culture. People should be educated to understand culture. They need to understand that it is a system of shared values (what is important) and beliefs (how things work) that interact with the company's people, organisation and controls. Other similar comments could be found in the article "Education: The key to JIT success" [42].

But this process of education, encounters an important barrier in the organisation itself, in the way it is managed and structured, as it has been stated throughout the previous sections. The so emphasised problem of functional competition and conflict avoids the completion of a cooperative management.

As it is explained in the article "Change is as good as rest... provided you train for it" [43] and based on a study conducted by Ashridge Management College, organisations are attempting to reverse or at least mitigate some of the more stultifying effects of Taylor's legacy. Manufacturing organisations are becoming flatter, more market-driven, more customer oriented, increasingly decentralised and generally more fluid.

This trend and the perception of its need, has major implications on how manufacturing organisations are managed. The ability to manage *horizontal* issues (such as quality and service) across the company, and therefore across departmental boundaries, is becoming more and more important compared to the traditional vertical management.

So, how should education be focused in this coming pro-cooperative environment? How could education emphasise the need for cross-functional involvement inside an ever changing and conflicting organisation?

2.6 CONCLUSIONS

Once having seen the importance of coordination and how organisational boundaries become less of a concern as codestiny, a word for common goal, is developed among people and functions in maintaining, looking for, a competitive position and achieving the common goals of the organisation. If people focus only on internal disputes, the organisation will most likely become extinct. And the way to the definition and understanding of that codestiny, as well as the problems associated in its search is **AWARENESS.**

As more people understand the vision of the future and the direction of the company, as well as how each individual fits into the picture, the whole system will be effectively integrated, coordinated, and become capable of responding to necessary changes. Therefore, the first task of manufacturing organisations is to fully understand their problems [44].

Organisations have to be framed in a process of learning, the "*Learning Organisation*" has to be created, an atmosphere of recognition to the role of **education** and **training** in the awareness process. That "*Learning Organisation*" has to be established through a cross-functional learning process where all will comprehend the dynamics of the process they are immersed in.

If that is the case, manufacturing organisations will have to **educate** their people in a **cross functional process** which also will have to take into account the **dynamics** of the environment companies are working in. In this context, the role of education in the development of change can not be over-stated. Indeed, the success of the different changing processes manufacturing organisations are involved in, depends upon the ownership, commitment and awareness of the people affected throughout all levels and functions of the organisation, being every of these prerequisites achievable through an effective education and training program.

Therefore, education can no longer be seen as a static and independent process, where people only understand how they work, and how others work on a stand-alone basis. What is needed is a new and more **integrative** approach, where people understand not only how they work and why, but also how they affect other areas of their manufacturing organisation, in other words how their policies and performances fit inside the policies and performances of the whole organisation. This is certainly a **dynamic** process as decisions are being made throughout the organisation in response to a continually demanding environment.

Thus, in the next chapter it will be explained the different existing educational methodologies and in particular it will be emphasised the role of computer based educational programs (e.g. computer based gaming-simulations) as tools for a more **dynamic**, **interfunctional**, and **goal integrated** approach to the learning process.

3 METHODOLOGIES FOR EDUCATION AND TRAINING

3.1 INTRODUCTION

As it has been concluded in the previous chapter, education can no longer be seen as a static and independent process but as an integrative and dynamic one.

Now that the main objectives for the educational process have been defined, there is a need to define the tools which will help carry out that educational process inside its philosophical framework.

The importance of the tools chosen for educational process is clearly stated by Robert I. Millard [45]. He concluded in his research about MRP implementation and its failures and successes related to the educational aspects that:

"*Success did seem to be related to the amount of outside assistance and to the breadth of educational effort applied during the system implementation. While these results suggest that education and training are important ingredients in the implementation process, they suggest that the type and quality of materials used may be more important than the time spent with them.*"

This assertion can be numerically seen in table [3.1], which has been taken from the previously mentioned study.

Correlation of Implementation Variables with MRP Success	Outside Assistance	Education Days per Employee	Breadth of Educ. Matls.
Increase in turnover	+ 0.50	+ 0.04	+ 0.33
Decrease in lead time	+ 0.24	+ 0.16	+ 0.10
Perceived project success	+ 0.25	+ 0.12	+ 0.13
MRP usage class	+ 0.12	+ 0.11	+ 0.24

Figure 3.1 Correlation of Implementation Variables with MRP success. Taken from Millard [45]

Nowadays, there are many well known training companies that are using different techniques and tools for educational purposes. These educational aids range from textbooks, seminars and short courses to videos; from manual games and simulations to computer based games and gaming-simulations.

Within this diversity of possible choices, this study will focus on the analysis of the computer based educational aids, and will try to establish whether these computer based aids can address the problems of **dynamics** and **goal integration** (cross-functional coordination) in the educational process.

3.2 EDUCATION AND TRAINING THROUGH COMPUTERS

Traditionally the process of education has involved lectures, discussions, and "programmed learning" approaches requiring the student to sit and read the text, answering questions at the end of each section as a way of feedback and interchange. More recently there has been a considerable number of educational videos produced on manufacturing subjects by such companies as the Oliver Wight organisation. These are useful ways to provide information but are rather limited in the way they help people understand a dynamic and real situation of the sort that exists within a manufacturing organisation. Additionally any activity that requires the student to just listen and not participate tends to get very boring. Nowadays it is the computer and its flexible structure that comes out to face the educational challenge.

The advent of the personal computer and its never ending technological improvement (e.g. speed, memory, flexibility, increase in the number of software tools,..) [46] has encouraged a explosion in the amount of educational software available. There are many written examples backing up this and explaining the use of computers for teaching purposes [47,48,49]. In one of these, for example, fourteen packages in the field of P.O.M. (Production and Operations Management) are reviewed.

The reason for the appearance and increase in the use of this sort of educational tool must be seen in the opportunity that computers provide towards the generation of an environment where theory and practice can be demonstrated in the context of real and immediate operating situations [50]. This possibility of having real experience in dealing with a "real" environment, undoubtedly, results in more effective learning. It helps to create discussion about a real question and reinforce in that way the learning process.

The topic of business games and simulation is therefore discussed in the following section. These, seem to offer an understanding of the concepts of interaction and dynamics which have already been identified as being so important in manufacturing.

3.3 BUSINESS GAMING-SIMULATIONS, WHAT ARE THEY ?

3.3.1 INTRODUCTION

Since 1960, Management Games also called Business Games have become a more widely known - but rather less widely used - technique of management education and training. They are not yet used, of course, with the same frequency and intensity as case studies but games have and will undoubtedly become another tool of the management teacher, to be selected and used whenever appropriate.

Historically, management games are directly traceable to war games. These developed with chess, continuing in 1798 with the German "war" game called "*Neues Kriegspiel*" that was a "TEWST" - Tactical Exercise Without Troops and many more broadly used during the 19th century. It was not until many years later that as a result of a visit by a member of the A.M.A. (American Management Association) to the U.S. Naval Academy, it was decided by the A.M.A. that it might be worthwhile applying the techniques of war games to management education.

Although the existence of war games was one of the stimuli to the development of management games, there were also two other factors that assisted in the development of games. These were the Development of the Theory of Games and The Advent of Digital Computers [51].

3.3.1.1 THE DEVELOPMENT OF THE THEORY OF GAMES

The publication of "The Theory of Games and Economic Behaviour" [52] by Von Neumann and Morgenstern in 1946 led economists, operational research workers and management teachers to consider, in new ways, the analysis of action in competitive situations. Although the Theory of Games offered little in the way of specific applicable analytic tools, it did illuminate the analogy between games and industrial situations. It also made people familiar with game concepts and greatly stimulated interest in problems connected with games.

3.3.1.2 THE ADVENT OF DIGITAL COMPUTERS

The advent of computers and the continuous technological advances they have been experiencing, contributed and are still contributing to the development of business games. The reasons for this were and still, partially, are based on three advantageous main elements:

1. The use of computers in a management game adds considerably to the "drama" of the game.

2. Computer manufacturers seized upon games as a good sales opportunity for their product in so far as they wished to sell computers for business, rather than scientific, purposes.

3. The existence of computers, with their speed and flexibility provided designers of games with the opportunity to develop complex mathematical models and interactions to underlie games whilst still keeping the administration of the game relatively simple.

All these factors, occurring at about the same time, accelerated interest in the transference of the gaming technique from a purely military training to that of management education.

3.3.2 BUSINESS GAMES

Management games as such started in the late 1950s, with the development of computerised management games dating from not long after this period [53].

Historically speaking, management games originate from three major design foundations: **(1) role-play, (2) gaming, and (3) computer simulation.**

1.Role-Play: In role-play, the player of the game assumes a prescribed role in a particular situation. He is given the opportunity to feel what is of interest in that position. It is hoped that players gain a greater understanding of the roles and relationships as well as a better awareness of their own activities. Moreno's [54] work with "psychodrama", a form of therapy for mental illness, is most often cited as the beginning of use of role-play as a vehicle for extending research into human behaviour in varied learning environments.

2.Gaming: Traditionally, games consist of interactions among group of players (decision makers) placed in a prescribed setting and constrained by a set of rules and procedures. The behaviour and the interaction can possibly involve competition, cooperation, conflict or even collaboration.

3.Computer Simulations: Computer simulations has roots in mathematics, probability theory, and other associated mathematical techniques. In the early days the purpose of computers simulations was to find answers quickly and accurately by following a predicted algorithm. In the corporate area, computers have been used extensively for analytical simulations to support the evaluation of implementation alternatives for new facilities and products prior to commitment to important decisions.

3.3.3 GAMING-SIMULATIONS

These three previously explained streams of development (role-play, gaming, and computer simulations) serve as a general background for **gaming-simulations**, and **computer based gaming-simulations**, words that from now on will be used indistinctively. The distinction between these three types of activities is not clear-cut; there is a great deal of overlap. Nurtured in this context **gaming-simulation** appears as a new intermediate element. **Gaming-simulation**, is an hybrid form involving the performance of game activities in simulated contexts [55].

In these **gaming-simulations**, the environment and activities of participants have the characteristics of games: players have roles to play, goals they seek to achieve, activities to perform, constraints on what can be done, and payoffs (positive and negative) as a result of their actions and the actions of other elements in the system (including chance). But also, in a **gaming-simulation,** the game roles, goals, activities, constraints and consequences, and the linkages between them are patterned from real life, they are a simulation of the real-world systems.

Gaming-simulations differ from role-play management games in several important ways. Role-playing is an instructional technique, an element of gaming-simulations, but the latter also include other components. In most role-playing exercises the participant is assigned a role and given a general outline of a situation; from there the action is freewheeling. In gaming-simulations, on the other hand, roles are defined in interactive systems. In other words, the emphasis is on the role as it interacts with other roles; the model creates the basis for the dynamic interaction, including also all the constraints, rewards and punishments that characterise games.

Gaming-simulations thus differ from role-playing exercises in the degree of structure and formalisation they entail and in their emphasis on interaction processes rather than on the playing of individuals roles. Furthermore, in many instances of classroom role-playing, several students participate while the remainder of the class watches, or the class is broken into small groups that engage in parallel role-plays. On the other hand, when gaming-simulations are used, all students are participants; none are passive observers.

3.3.4 BUSINESS GAMES AND GAMING-SIMULATIONS AS EDUCATIONAL TOOLS

The standard mode of teaching and training has historically been the lecture or the lecture and discussion method. But these methods have some limitations, weaknesses that Cathy Stein Greenblat [55] think could be overcome with the use of **gaming-simulations**.

She thinks that the most critical problem traditional teaching methods have to face, is that learners are passive recipients of information. They rarely have to put their ideas forward, and their attention frequently wanders to other goals or goals of greater interest.

Secondly, in her opinion, material is presented in a sequential way. This linear approach makes it difficult for the learner to grasp the nature of the whole. This is particularly so in instances in which one is trying to teach the appreciation of the simultaneity of events and actions.

A third shortcoming of the lecturing method is that it is a difficult task to present systems characteristics clearly. The multiple connections and consequences of a given element or action are difficult to convey verbally. While it is often impossible to understand a social system by simply immersing oneself in it, because the complexity is too confusing, the lecturing process all too often suffers from oversimplification.

In opinion of Greenblat, **Gaming-simulations** can address all these shortcomings in various ways. First, in a gaming-simulation, in contrast to a lecture situation, everybody is an active learner. They must make decisions, pay the consequences, articulate positions, and make the system work. After play, post-game focuses on what happened and why, how that relates to the counterpart, and the limitations of the gaming-simulation. Thus the learning experience builds upon the philosophy of the old Chinese proverb:

I hear and I forget

I see and I remember

I do and I understand

There are also other positive aspects of gaming-simulation as a learning tool. The active involvement of gaming-simulations in the learning process implies a experimental learning that permits discovery, experimentation, and reflection.

Second, rather than being lectured to and then engaging in discussion, students experience the topic as a whole, since many components of the system are presented simultaneously.

Third, and finally, gaming-simulations are particularly useful for conveying systems characteristics. It has long been recognised that systems are more clearly described using graphical models, such as diagrams or pictures, or using physical representations, than through using verbal models.

In Loveluck's opinion [51], gaming-simulations force participants to realise the nature and importance of effective **organisation** and **intelligent cooperation**, in relation to a "business-like" situation and problem. He also argues that participants in gaming-simulations must discover for themselves where the **problems** in this "business-like" situation lie, giving the player a superior understanding of the situation in comparison to merely being told what these problems are. Gaming-simulations also provide a focal point for thought and discussion and establishes a common basis for communication inside a feedback process. Loveluck also explains that although sometimes gaming-simulations could be seen as unrealistic, this fact is a strength rather than a weakness. The specific assumptions upon which the game is based, is an important starting point for consideration of the way in which the "real world" managerial problems evolve and impinge upon one another.

Another contribution to this matter is the one from C. Brand et al. [56] who precisely defined which is, in their opinion, the role of games which simulate real-life organisations. It is their opinion that games applied to organisations will, among other advantages, "*enable participants to understand the wider aspects of their organisation*".

With all these explanations in mind, it is concluded that while gaming-simulations will not solve all educational problems, they offer, exciting opportunities for teaching and learning about many social systems and social problems. They are an outstanding educational element when dealing with the understanding of dynamic social systems (e.g. manufacturing organisations) where the interaction, dynamics and overall perspective of the system, rather than a local one, are of a great importance. They are extremely useful when the response of the players to certain problems and dynamic situations in the context of continuous interactions, is a part of the environment to be studied.

In respect to manufacturing organisations, these are considered to be systems that interface with others in complex and uncertain ways, and also tend to become more complex as time goes on. This idea, as it is already been discussed, requires ever more sophisticated ways of managing both their internal workings and their interactions with the environment. In this complex and dynamic environment is where gaming-simulations can provide companies with an effective educational tool. They provide a relatively safe environment for the learning of a whole range of management concepts, practices and skills in the context of a world where many themes are interdependent.

Based on the fact that gaming-simulations can have a great role in the educational process of understanding and awareness of the different existing functions and interactions inside a manufacturing company, there have been and there are many different approaches to their development. The next section will look at them, seeking for a suitable approach to the already stated ideas about the role of education in a manufacturing organisation.

3.4 PAST AND FUTURE TRENDS IN BUSINESS GAMING-SIMULATIONS

3.4.1 INTRODUCTION

The use of computerised gaming-simulations for business and management education as a tool that encourages the development of an individual's wider understanding of the integrated nature of business organisations, appears to be well established.

With this notion in mind, it will then be discussed the way designers have tackled the development of these computerised gaming-simulations. Discussing, as well, the suggestions done by Tom Burgess [57] about the design of gaming-simulations.

He maintains, that the design of most, if not all, of the existing computerised gaming-simulations have hardly changed from the original implementations. It would seem that very few take advantage of the capabilities which are evident to other types of modern software (e.g. graphic user interfaces, context sensitive help, networking and so on).

In the next two subsections, the traditional and modern, future, design approaches to the development of computerised gaming-simulations will be compared, trying to establish which should, in the context of management education, be the most appropriate approach for the current manufacturing environment.

3.4.2 TRADITIONAL MODELS

The traditional computer based gaming-simulation models were and are basically based on a single factor: the engine that drives them. In other words, the fact that they are based on mathematical models, models that try to imitate a business or part of it, and its relationship within an environment.

The role of the computer in this sort of gaming-simulation is mainly a computational one. In this context computers are seen as "number crunching" calculators that will avoid the burden task of doing all the calculations involved manually. Decisions are fed into the machine which then computes the results for later distribution. In essence, the computer remains a calculating device used because it is fast or simply because it is there, and can handle more complex equations. Examples of these kind of games are the STORM [58], COMPUTER MODELS for OPERATIONS MANAGEMENT [59], TEAMSKILL [60], TOPAZ [61], EXECUTIVE [62], BISSIM [63], all of which have been thoroughly analysed as part of the research project.

Their "improvement" has also gone in that direction, trying to enrich their performance by increasing their numerical complexity (i.e. increasing the number of parameters in the mathematical model used, expanding their relationships, and making these models more robust). Quoting Ahituv and Newmann [64] :

"The model attempts to duplicate the real world under varying states of nature for a given set of parameters."

This computational approach tempts designers to construct ever more complex models which can become dangerous. The reason for that comes from the fact that they are not sufficiently complex to become a valid econometric model of a business but, on the other hand, they are too sophisticated for effective education. The added complexity does not necessarily add clarity and comprehension to the learning process.

These models assume that the real-life equivalent is able to be modelled, moreover they presume that an optimal mathematical solution (input) is available, desirable, and a reality, and that individuals and organisations work in a rational way.

But this whole idea lacks of an essential element when dealing with gaming-simulations as an educational tool: **people**. In this highly technical approach the social context of the individual is ignored; remaining unexplored aspects of group membership and social interaction (e.g. potential conflict of functional and organisational goals). In these more traditional group of games the tasks involve, consist of simulated business decision-making of various degrees of complexity in which the main focus of interest lies in the business performance of the simulated company rather than in the organisation of the group and its **interactions**.

This computational approach does not properly address "**dynamism**". Decisions, in this kind of games, are made regularly and simultaneously and in most of the cases at the start, removing from that sort of games an essential variable in managerial decision-making: **timing**.

Concluding, this traditional approach lacks two most important elements when trying to address the educational process of manufacturing organisations: **internal interactions and dynamism**. Elements that are extremely important in today's changing and cross-functional manufacturing environment.

3.4.3 MODERN MODELS

The modern computer based gaming-simulation models were and still are being developed as an answer to the weaknesses associated with the approach used by traditional models.

This approach has involved the creation of more realistic gaming-simulations rather than creating more equational complex ones. These gaming-simulations are not exercises in "beating the machine".

Their strategy is to increase the realism of such games by replacing *hard* complexity (i.e. equations) with *soft* complexity. In opinion of Robert Johnston [65], *soft* complexity is obtained by providing a situation where goals may be unclear, ambiguous and not always quantifiable, where relations are not always deterministic, where all the possible decisions choices are neither known nor limited, where the information required to choose between them may be neither complete nor available and where the consequences of the choices are not set, clear or totally consistent. Furthermore he states that to do that, the computer model should be replaced by a human process relying on a broad and not totally rational or complete understanding of the simulation situation, where choices, information, and decisions are based on human interactions and relationships that are at the heart of every business process.

This same idea of the *soft* complexity is also suggested in a study conducted by Walter E. Rohn [66]. In his study he states that there is an increasing customer demand for business games on behavioral and management-attitude problems. He also thinks that what is needed are appropriate models for the realistic simulation of all human factors influencing the effectiveness of an organisation and the quality of managerial decision making. He states that every step forward in this field allows more reality in simulating real life in gaming-simulations. Clear examples of this kind of games are the one developed by the University of Bradford called BUSGAM [67], and MOSES (Manufacturing Organization Simulation and Evaluation System) [68].

But for this to be achievable, there is an important need for advanced computer technology. As it is concluded by Nigel Bryant et al. [69] in his article "Changing technology and management games":

"Our final argument is that technology influences the type of games we have, and the types of games influence the type of management games we have. Furthermore, because there is a time lag between the development of new technology and the game developers taking advantage of this new technology, then we should be able to look at the present day technology and predict the way in which management games are going to develop."

The comparison between the traditional and modern approaches when designing gaming-simulations is summarised by Coote [94] in table [3.2].

SUMMARY OF MAIN DIFFERENCES BETWEEN COMPUTER SIMULATIONS AND COMPUTER ASSISTED SIMULATIONS

Criteria	Computer Simulations	Computer Assisted Simulations
role of computer	central: to run simulation on; for precision	peripheral; as springboard to social activity; to enhance simulation procedures
main interactions	between computer and participants; Participants face screen and respond to computer prompts	between participants who face and respond to each other
aim of simulation	"winning" against computer	exploring social situations
decision process	essentially rational (manipulation of mathematical variables)	socio-political or power bargaining (negotiation, coalition forming, bargaining)
participant roles	puzzle solvers and keyboard operators	social actors in a variety of roles

Figure 3.2 Summary of differences between computer simulations and computer assisted simulations. Taken from Coote [94]

Four of the most successful business games in Great Britain are evaluated in Appendix B. These were presented in a workshop held the 6th of February 1991 at the Management Centre, University of Bradford. The evaluation is a personal opinion about the four games, where the existence of the two design approaches formerly mentioned can also be seen.

3.5 CONCLUSION

Computers add an important element to the educational process, that is the possibility to experience "real-life" situations in a dynamic environment. Moreover, that main advantage, in respect to more traditional methods of teaching, could be used for the development of successful computer gaming-simulations that could be applied to manufacturing organisations.

But what kind of gaming-simulations? The main approaches have been shown and based on experts' opinions the relative advantages have been discussed.

In our opinion and having looked to the two approaches, the way ahead should be a compromise between the two, where computer power should be used to address specific elements of manufacturing organisations but within an integrated, interactive and dynamic framework, similar to that manufacturing organisations have to face. Those elements will certainly improve the rate of realism attached to gaming-simulations without making them more complex and rigid.

The learning opportunities inherent to this approach are undoubtedly greater. This means not only that participants are able to experiment with technical variables (product levels, marketing strategy, and so on), but also that they are able to experience a fundamental aspect of organisational life, that of bargaining and negotiation in the decision-making process. They are faced with the inevitably interdepartmental rivalry so common in organisations, and with the conflict between this rivalry and the need to cooperate on a company basis.

Inevitably, that approach to the design of games is related to the existence and effective use of the technical computer tools available. Loveluck [70] and Bryant et al. [69], for example, refer to that relation stating that business gaming-simulations will be more interactive and dynamic, focusing on the "organisational behaviour" of participants through the use of elements such as interrupt functions, graphical displays, flexible designing, and networking.

This research is directed to look at the feasibility of such games. Thus, the possibility of creating such games, based on the actual existing hardware and software, will be evaluated.

Therefore, the current work will try to develop a computer gaming-simulation based on those formerly mentioned innovative elements. A gaming-simulation will be used to understand the dynamic and interfunctional aspects of manufacturing organisations. In particular, to make people aware of the implications of their behaviour on others as well as on the whole organisation, as an essential step towards the understanding of the importance of coordination.

For this reason, in the next chapter it will be shown what has been our approach in designing Nottingham Polytechnic's gaming-simulation and which have been the problems encountered on its development.

4 NOTTINGHAM'S POLYTECHNIC GAMING-SIMULATION

4.1 INTRODUCTION

As it has been stated in the previous section, computers offer an exceptional number of educational advantages compared with other more traditional methods of teaching. It was also discussed how those educational methodologies could successfully be applied to manufacturing organisations as computer models.

It has also been suggested that for those computer models to be successful, when applied to manufacturing organisations, they should address specific elements of manufacturing companies inside an integrated, interactive and dynamic framework.

It is in this context where this chapter of the thesis will try to address the process of development of Nottingham Polytechnic's gaming-simulation, looking at the different steps of the development as well as the problems encountered in the process.

It is also the aim of this chapter to look at the feasibility of these kind of gaming-simulations in the actual technological (software and hardware) context.

4.2 TOTAL ENTERPRISE BUSINESS GAMES

Before explaining the design process of Nottingham Polytechnic's gaming-simulation, the concepts of **total enterprise business game** and the **production management game** will firstly be introduced.

These two terms although closely related diverge in the degree in which they concentrate on the production area of manufacturing organisations.

Total enterprise business game is a term used to refer to games that include all the main functions of a business enterprise as its decision inputs: marketing, production, and finance. Due to their flexibility, total enterprise business games can fulfil a great variety of purposes. They are mostly used as integrative teaching techniques to bring together all the functions of business and allow participants to view them as a whole. In management development and executive programs, games often serve to break down the conceptual walls of narrow specialised thinking and cause businessmen to better appreciate the difficulties of their counterparts employed in other business functions [71,72].

Nevertheless, most of the existing total enterprise business games are more marketing, finance, tactical, and strategy based rather than production based. Most of them can also be categorised in the **traditional model** group of computer games ,(refer to previous chapter), taking little account of the dynamics and real interactions of the situation they are resembling; although there is a view that these trends are changing [72].

At the other end of the spectrum is the **production management game** concept. Although there are many games in which the term "production" is simply used to describe a game of a total enterprise nature, **production management games** refer to those that concentrate on the organisation and control of the production function or, more specifically, they are concerned with the decision-making necessary to ensure that goods are made in accordance with the requisite quality standards, in the requisite quantities, at the requisite times, and at a minimum cost.

Based on this idea, most production management games tend to concentrate on the specific production function of an organisation in isolation, rather than integrating the production function with the financial, marketing and other elements of the organisation [73]. Although these games offer a high degree of manufacturing detail, they tend to produce suboptimum decision-making when examining the organisation as a whole.

The advantages that a half way house between the above two concepts could offer are obvious. A hybrid concept of a production function could be used to link the business-making policy of the organisation as a whole. That hybrid concept will have to simplify the production subsystem in order to show the interactions with the other subsystems of the organisation (particularly marketing, personnel, and product research and development), and also emphasise the importance of the production function as the key element of a manufacturing enterprise.

Nottingham Polytechnic's gaming-simulation is designed to be such a hybrid game.

4.3 THE DESIGN PROCESS

4.3.1 INTRODUCTION

The designing process of gaming-simulations is generally defined as being a creative one and it is therefore impossible to find precise rules about how they should be designed. However, notes and guidelines about the design and construction of gaming-simulations can be found.

Many experts associated with business games, gaming-simulations, and games in general have contributed to the game design process by publicly stating their ideas about the guidelines to be used.

Loveluck [51], for example, considers the properties of a game from the constructor, designer, point of view as follows:

- Situation being simulated.
- Model underlying the game
- Information made available

- Stress of the situation
- Decision-making required
- Reports required
- Special Assignments
- Composition of teams

Another similar approach is the one recommended by G.I. Gibbs [74] where he defines fourteen design stages:

- Aims
- Context
- Target population
- Models
- Scenario
- Information Flow
- Cast List
- Role elaboration
- Player objectives
- Statement of rules
- Description of constraints
- Materials and equipment
- Evaluation
- Reprography

The model that represents the design process defined by G.I. Gibbs is shown in figure [4.1].

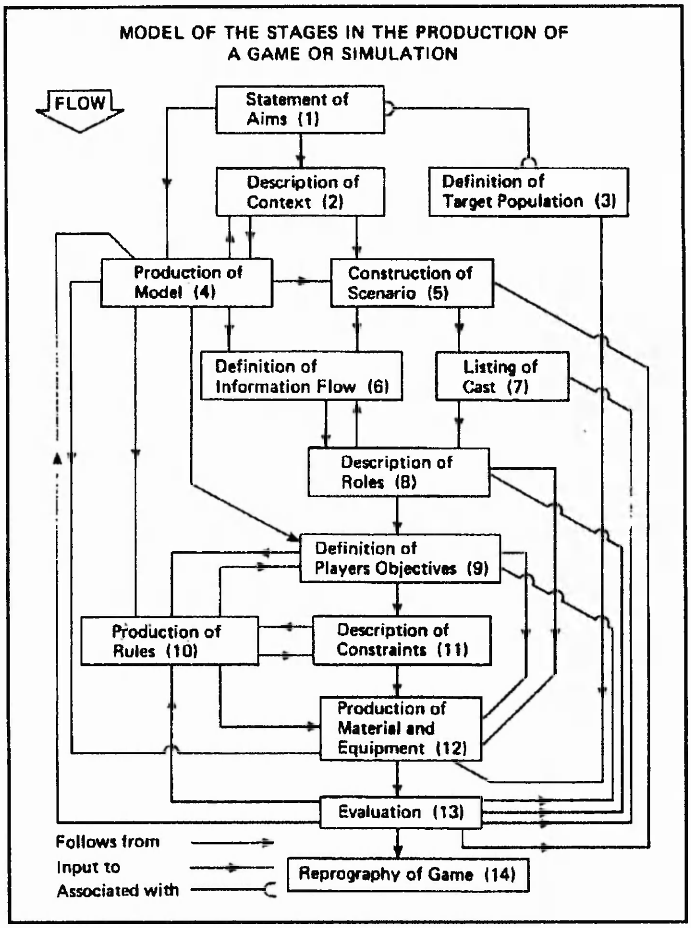


Figure 4.1 Game's design process.
Taken from Gibbs [74]

Yet another analogous contribution is the one by C. Brand and T. Walker, who in their article called: "The simulation of 'real-life' organisations within a game" [56] define a logical approach to games design. In their opinion a successful game simulating a real-life organisation must adopt a logical problem-solving approach throughout the design process. This means that the game's development can be subdivided into the following distinct stages:

- Developing the game brief
- Detailed research programme
- Production of a model of the organisation
- Formulation of the outline game
- Client review
- Detailed game design
- Play-Testing
- Production of the final game

A more recent and also more comprehensive approach to the design process is the one described by C.S. Greenblat. In her book called "Designing Games and Simulations - An illustrated handbook" [55] she states that the process of design of a gaming-simulation can be thought of as consisting of five stages.

- Stage I: Setting objectives and parameters
- Stage II: Model development
- Stage III: Decisions about representation
- Stage IV: Construction and modification of the gaming-simulation
- Stage V: Preparation for use by others

This methodology appears to include all of the steps proposed by the other authors in a coherent framework and consequently has been used in this development.

Another important element attached to this design process guideline is the fact that, although the defined sketch presents the process as unidirectional (the designer moves from Stage I to Stage II, and sequentially until Stage V), the actual process in itself is much more about moving back and forth among these different stages. In general, however, the "push" is in the forward direction - that is, the process is basically sequential, though there is considerable iteration.

With all this in mind, the coming sections will be concerned with the way Nottingham Polytechnic's gaming-simulation has been designed and constructed based on Greenblat's design process.

4.3.2 SETTING OBJECTIVES AND PARAMETERS

At the outset of the design process it is crucial that the gaming-simulation designer considers what he or she is trying to do and to what ends. This means delimiting as clearly as possible the subject matter, purpose, intended operators and participants, and the context of use for the proposed gaming-simulation.

In Greenblat's [55] opinion, this stage is too often skipped or undertaken casually, and the would-be designer's initial enthusiasm is translated into only a vague formulation, with no thought given to what the gaming-simulation is supposed to convey, to whom, and under what conditions of play. The result of this neglect is often failure or a product that is totally inappropriate for the intended audience.

Therefore, it is critical to make these decisions before anything further is done, and the good designer will spend considerable time formulating such a statement with the firm conviction it will guide him or her later in the process. Wise decisions in the coming stages will only be possible by reference to criteria that are implicit or explicit to the specifications in this stage.

One also should be aware that the **gaming-simulation's** design process has to manage two factors simultaneously, game and simulation. In this context it has to be remembered that the game roles, goals, activities, constraints and consequences, and the linkages between them are patterned from real life, in other words, they simulate these elements of the real-world systems.

4.3.2.1 THE SUBJECT MATTER

The first and more obvious consideration is that of the **subject matter**. At this early point although the designer should not be concerned with a detailed statement, he should, however, restrict the extent of the educational topic he is going to present to the players.

In Nottingham Polytechnic's game the subject matter is very much related to what has been stated in the previous two chapters of the thesis. Consequently the subject of the matter will have very much to do with manufacturing organisations, and the theme of coordination among them. That idea can be summarised in the next paragraph:

"*The gaming-simulation to be created is to simulate the way manufacturing organisations work, looking particularly at the way the different functions involved in the company process interact with each other in a dynamic, customer driven environment. Particular attention will be paid to the conflicts that arise from those interactions and the manner in which functional actions are influenced by local objectives instead of global ones.*"

4.3.2.2 PURPOSE

The question of purpose is closely connected with that of subject matter. Indeed, in some cases, the question of purpose may be more important than that of subject matter and can involve the definition of more than one purpose. If that is the case those purposes will have to be evaluated, and the most important ones will be considered.

The question of purpose consists of broad statements of educational intent. These statements express goals which outline the concepts, structures, processes, facts and attitudes, which are intended to give the participants experience.

The aims that Nottingham Polytechnic's gaming-simulation pursues are related to the importance of coordination and interfunctional harmony within manufacturing companies. This idea is fully summarised in the next paragraph:

"The gaming-simulation to be created is designed to serve as a catalyst to present issues and problems concerning the existing interactions between different functions of a manufacturing company within a dynamic, customer driven environment. In this context, an important aim of the gaming-simulation is to promote the understanding and necessary awareness of COORDINATION AND ITS IMPORTANCE IN MANUFACTURING ORGANISATIONS."

4.3.2.3 PLAYERS' CHARACTERISTICS

Another matter to be defined at this stage of the design process is the target population of the game. It has to be stated to whom the gaming-simulation is aimed, and which are the characteristics of those playing it. This topic is so important because games have to be designed to meet the needs of those being educated, and through this ensure their motivation and understanding of the process being undertaken.

As an example, the characteristics of a game about some aspects of business management should be different if it is to be played by audiences of community members to show them the pressures on business leaders than it would be if it is to be played by corporate executives to sensitise them to the problems of coordination of departments in the firm. It is important then, to state the target group of players and think carefully about their characteristics. These have implications for the degree of complexity it can be presented, the degree of formality that must be included, and the need for background information.

Nottingham Polytechnic's gaming-simulation is intended to two major potential groups of players:

> *(a) Manufacturing engineering students who desire to understand how the different functions of the manufacturing process fit together and what are the conflicts associated with their coordination in a dynamic customer driven environment.*

> *(b) People in middle management within different departmental functions of a company, that although having experience of the elements and decision-making procedures used in their respective functions, have no understanding of how other functions work and the reasons why. In other words, people wanting to understand how their functional behaviour affects other departments and consequently the company as a whole.*

4.3.2.4 OPERATOR CHARACTERISTICS

A question directly related to the characteristics of the players is that of the characteristics of the likely operators. It must be decided who these people are and what aims, interests, and abilities they have.

In Nottingham Polytechnic's case, although the game could be used by a teacher, group leader, or conference organiser; it is basically intended to be used by lecturers at the manufacturing department of the mentioned polytechnic.

4.3.2.5 CONTEXT OF USE

Specification of the context of use will give the designer the ability to determine how many players must be accommodated in play sessions. If the gaming-simulation is composed of a certain number of players, the gaming-simulation must be designed with that number of roles, or in a way players could be divided into smaller groups for simultaneous play. The answer to this point will also determine the time length of the game.

This time factor is very important for future decisions, and is closely related to the complexity of the model. As indicated by Duane Dillman [75] :

"The length of a simulation exercise should probably be governed by the amount of time necessary for the participants to uncover the governing relationships of the exercise and then develop the appropriate strategies and ideas and observe the results of their actions."

In other words, the defined time frame should be such that will allow the players to evaluate their approach to the problem being faced by a provided feedback. This common sense matter is considered by experts [71,72,73] as one of the major weaknesses of total enterprise games. Instead, these have overemphasised the decision input and model complexity factors, making the time span for game play too short to complete the learning cycle.

In Nottingham Polytechnic's model, the time frame thought to be adequate to start with, is at least two hours. That time frame could be changed based on the experience gain every time a game is played. Those two hours could be break down into some runs of above thirty minutes each, separated one from another by a analytical discussion about the first run. During that discussion the subject matter should be addressed.

4.3.2.6 RESOURCES

Another element of this stage about objectives and parameters, has to do with the resources available for the development of the gaming-simulation and the ones needed for its future use.

In the game covered in this thesis, the resources needed and available are very much related to what is needed for researching about the feasibility of multi-user, real-time gaming-simulations of manufacturing organisations. That is, the existence of a network of PCs (NOVELL [96]) and software capable of working in that environment, issues that will be discussed in the stages concerned with the representation, construction and modification of the gaming-simulation, and software design.

Coming back to the five main stages of the design process, and once all the steps regarding the setting of objectives and parameters have been clarified, it is time to move to the second stage of the design process: Developing the conceptual model.

4.3.3 MODEL DEVELOPMENT

The development of the gaming-simulation's conceptual model is extremely important as a system can not be simulated unless is understood. The task at this stage of the design process is to describe as clearly as possible the most salient aspects of the system to be simulated, that is, indicate the substantive content of the real-world system.

This description will present a simplification of the real system that is likely, however, to be far too complex to fully translate it into a gaming-simulation, particularly if play is to be relatively short (i.e., 1-2 hours). Then, the designer will have to review the model, identifying those elements and relationships that are believed to be of greatest importance for inclusion. This process involves not only the definition of concrete elements and specific linkages, but also the interconnection of their roles, goals, resources, and rules. The information processing that constitutes part of the model also has to be defined. A flow network has to be described, specifying the amount and kind of information to be handled and the timing of its presentation.

It is very much a two stage process. First a complex model is defined for the most important elements and interactions of real life systems. In a second stage, the model must then be simplified in order to convert the model into a game.

4.3.3.1 BUILD OF THEORETICAL COMPLEX MODEL

The general game model needs to be as extensive as possible, where all the different elements and interactions affecting the production process of manufacturing organisations could be seen. Its structure is based on the information flow associated to the production process, which is not relevant to this thesis and of which knowledge is assumed.

In Nottingham Polytechnic's gaming-simulation, the elements to be analysed have much to do with the functions within manufacturing organisations, their internal elements, and the existing interactions among them. All these factors were mentioned in chapter two and Appendix A, and will be the core playing ingredients of the gaming simulation.

■ Functions

The first step in creating this model was to establish the different core functions that the factory being simulated will be comprised of. Based on what was expressed about the functional structure in the second chapter of the thesis, it can be stated that there are seven functions to be considered :

- Product design
- Production Planning
- Production Control
- Purchasing
- Marketing
- Finance
- Personnel

From these seven functions, there are only five that could be consider fully related to the production process (Product design, Production Planning, Production Control, Purchasing, and Marketing). Additionally, Production Planning and Production Control may be considered as just one function from a game's point of view.

■ Elements of the model

Once the kind of manufacturing organisation and the functions that will be included in the model are established, it is time to define the core elements that, in our opinion, form part of the real interactions and conflicts within manufacturing organisations. These elements are the factors that influence and distort decision making in the functional areas of manufacturing organisations.

A summary list of the elements, divided by function, that can be found in a manufacturing organisation are :

- ♦ Product design
  - New design suggestions
  - Modifications
  - Design modification policy.
  - Design's ease of manufacture
  - Modular design

- ♦ Marketing
  - Customer orders
  - Forecasting
  - Spare parts policy
  - Lead time
  - Customer service policy (price and delivery)

♦ Production Planning & Control

- Routing
- Production lead times
- Stock policies
- Made it / Bought it
- Quality policy
- Scheduling policy

♦ Purchasing

- Purchasing batch policy
- Quality policy
- Delivery policy

Based on those functions and elements the general model defined is a highly complex one, as seen in figure [4.2] and figure [4.3].

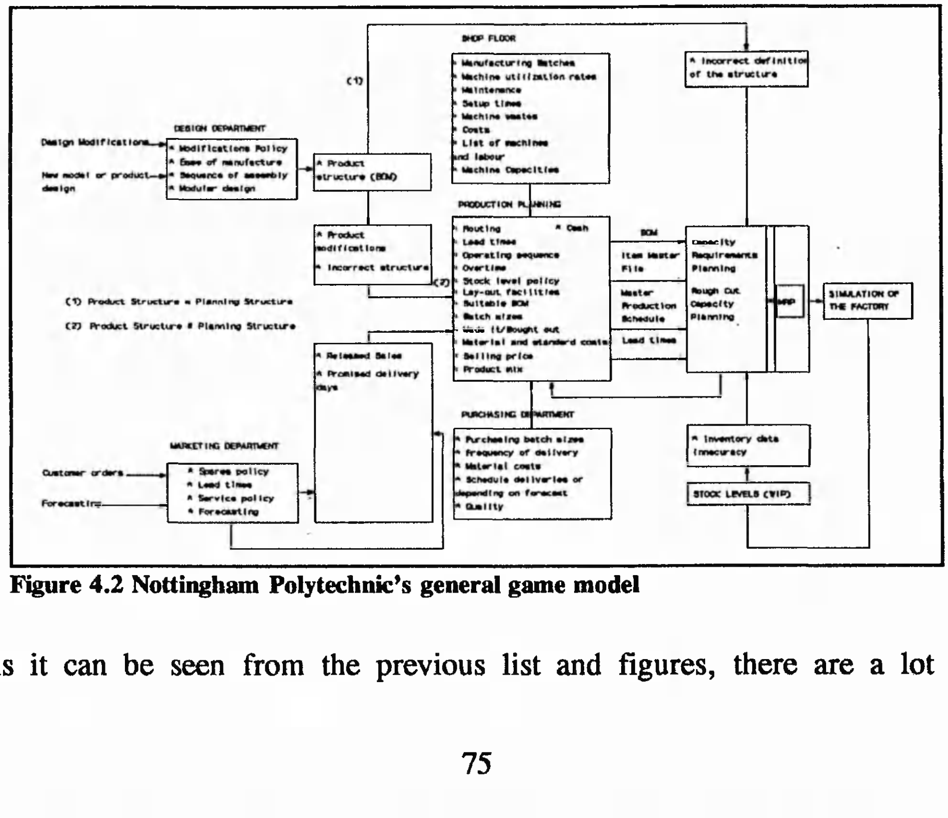


Figure 4.2 Nottingham Polytechnic's general game model

As it can be seen from the previous list and figures, there are a lot of

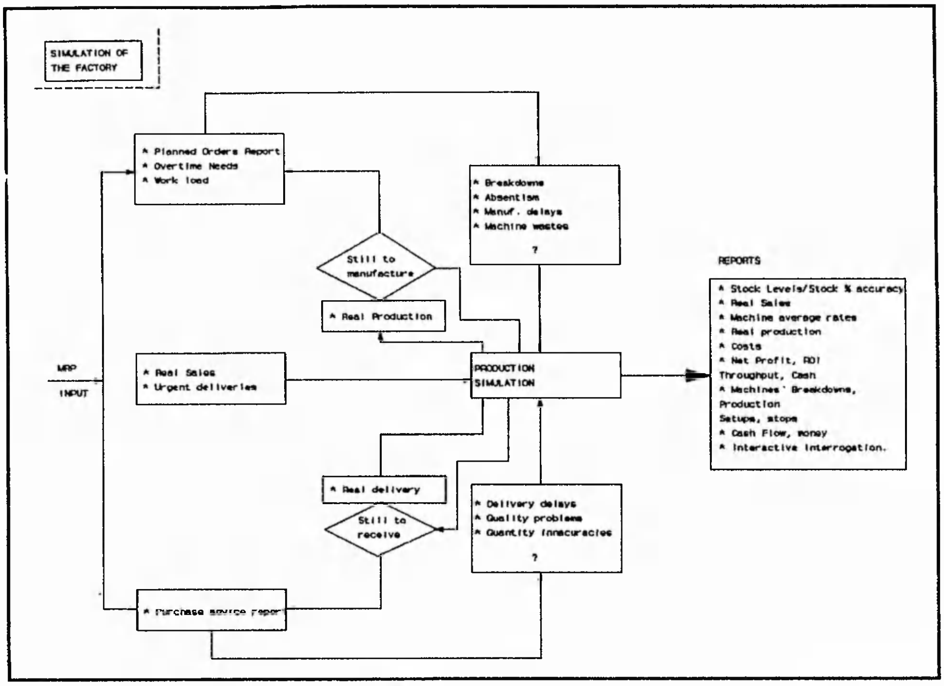


Figure 4.3 Nottingham Polytechnic's general game model.

elements that influence real life manufacturing organisations and that, due to their huge complexity, can not be addressed within a simulation game. Any attempt to do so, would imply the creation of a real simulation system capable of coping with all the internal and external elements of manufacturing organisations, something hardly possible. A model based on all those factors would not be a successful framework for a gaming-simulation's final purpose: **Learning and Experiencing.**

Hence, this general model had to be simplified in order to continue addressing the interactions and conflicts of manufacturing organisations, but within the existing technological and design constraints.

4.3.3.2 SIMPLIFY MODEL

The main problem when designing a game related to manufacturing organisations, is to simplify the real organisation sufficiently for players to handle it in gaming terms while retaining crucial relationships and patterns of interaction.

Additionally it is necessary to simplify the game from the developers point of view so that he can quickly create a prototype that will give insight into the game's behaviour, but again there must not be so much simplification that core problems are not addressed at an early stage.

In order to create the initial version of the gaming-simulation, the model was simplified in two different ways. Firstly, the number of functions was reduced to two (Production Planning and Control, and Marketing). The reason for that was based on the fact that the interaction between these two functions is one of the most important and more conflicting matters of manufacturing organisations, as was seen in Appendix A and is stated by Goldratt [37].

Secondly, the number and extent of the functional elements to be included in the gaming-simulation was reduced. That simplification was once more done with the idea of keeping the sense of the interaction between the two functions while reducing the complexity of the game in order to make it playable.

The resulting list of elements to be integrated into the model was :

- ♦ Inputs :
    - ● Marketing / Sales :
        - * Customer enquiries
    - ● Production Planning / Control :
        - * Orders to deliver
- ♦ Decisions to make :
    - ● Marketing / Sales :
        - * Price
        - * Delivery Date
    - ● Production Planning / Control :
        - * Scheduling (Quantity + Sequence)
- ♦ Outputs :
    - ● Marketing / Sales :
        - * Orders to deliver
    - ● Production Planning / Control :
        - * Stock to deliver the orders
    - ● Company :
        - * Benefits or loses
- ♦ Feedback :
    - ● Customer enquiries

Consequently, the model itself will have the structure shown in figure [4.4].

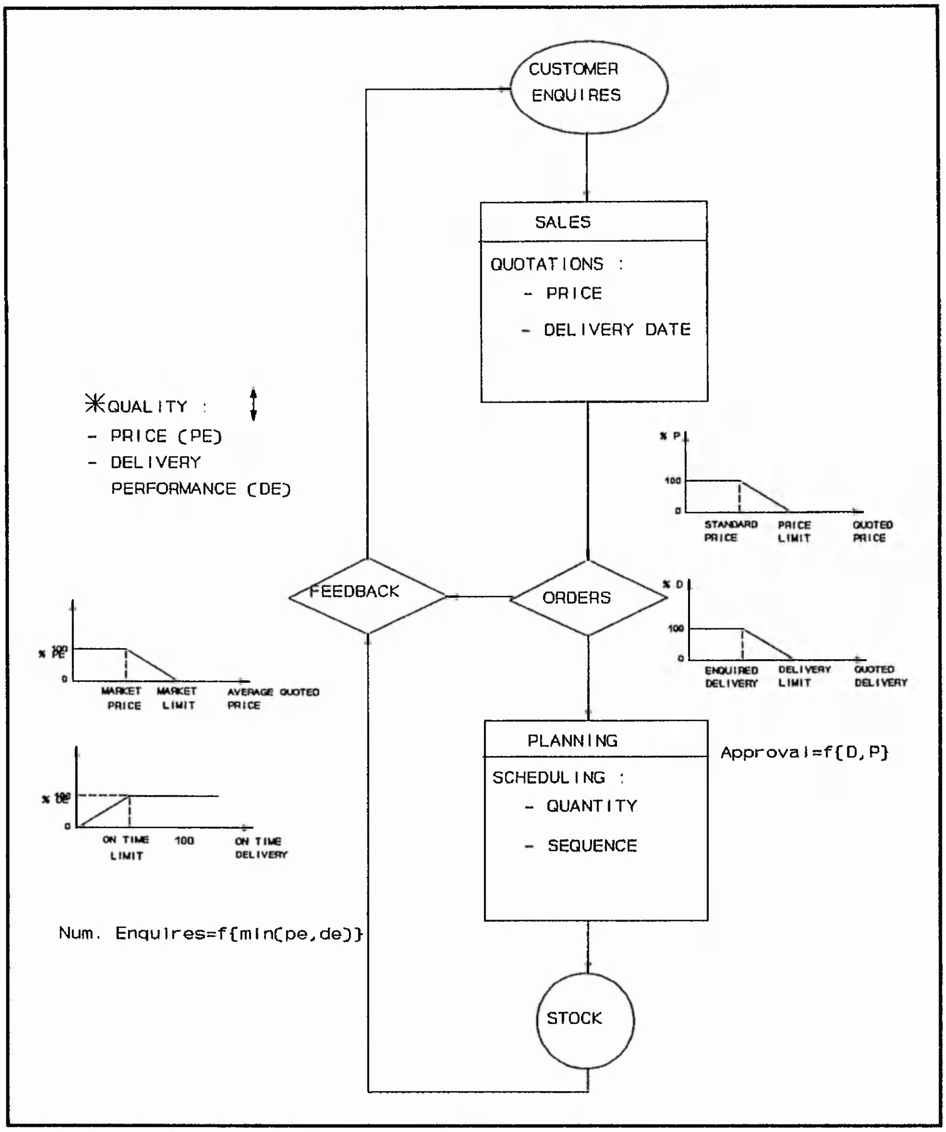


Figure 4.4 Nottingham Polytechnic's actual game model.

In this latter model the information flows in a sequential loop, from the customers that make enquiries to the people in the Marketing/Sales department who will have to quote those enquiries. Afterwards, the orders will go through a procedure which will simulate the customers' acceptance process (i.e. some quotations are accepted and some are not). From there the information about the accepted enquiries flows to the people responsible of the Production

Planning/Control department that will decide about the scheduling needs required for manufacture of the products ordered by the customers. With the defined scheduling the game will simulate the production process, which will update the information about the stock situation. Finally, the loop is closed by a preestablish relation (feedback) between the customer service performance (price + due date performance) and the amount of customer enquiries to enter the system the next period to be simulated.

Therefore, the gaming-simulation will consist of two basic, very simple, mathematical models. The one that regulates the customer quotations' approval (i.e. firm customers' orders) and the one that regulates the feedback (i.e. customers' enquiries).

4.3.3.3 ORDERS' APPROVAL

The orders' approval function is the one that simulates the acceptance of the sales department's quotation by the customer. If the quotation is accepted, it becomes a firm order that will have to be delivered to the customer and therefore, the planning department should have into account in its scheduling.

This approval function is based on two elements:

- Quotation Price
- Quoted Delivery Date.

These two elements depend on two factors : the closeness to the customer's price and delivery date expectations. The spectrum for these two factors is from 0% to 100%, depending on whether the quotation is far from what the customer expects, or equals, even improves, his expectations. The models that regulate this two factors are shown in figure [4.5].

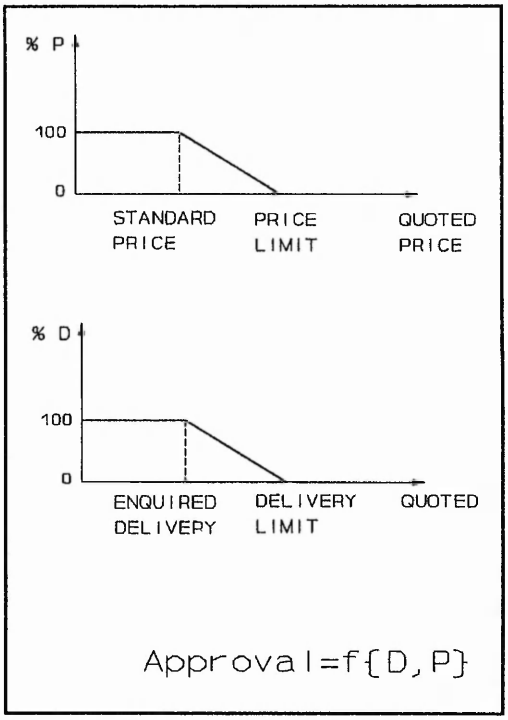


Figure 4.5 Orders' approval functions.

Hence, the ultimate model that simulates the approval process is function of a combination (multiplication) of the previous two factors and a calculated pseudo-random number (percentage). If the multiplication of the two factors associated with the quotation is greater than or equal to the generated pseudo-random number, the quotation is converted into a firm order. Otherwise the customer's quotation is rejected.

$$\text{Approval} = \text{Random Number } (\%) <= [\text{ Price factor } (\%) * \text{Delivery factor } (\%)\ ]$$

4.3.3.4 FEEDBACK

In all gaming-simulations there must be a feedback component capable of showing the dynamics of the model being played and measuring the convenience of the decisions taken. In Nottingham Polytechnic's gaming-simulation, the feedback component chosen is the *number of customer enquiries* for each gaming period.

That number of customer enquiries is function of two elements :

- Average Quoted Price
- Due Date performance.

These two elements are, themselves, associated with two factors, each of them giving a numerical estimation (percentage) of the closeness to the market's price constraints and factory's due date performance. The spectrum for these two factors is from 0% to 100%, depending on whether these two factors are far from what the market and customer expects, or equals, even improves, its expectations. The models that regulate this two factors are shown in figure [4.6].

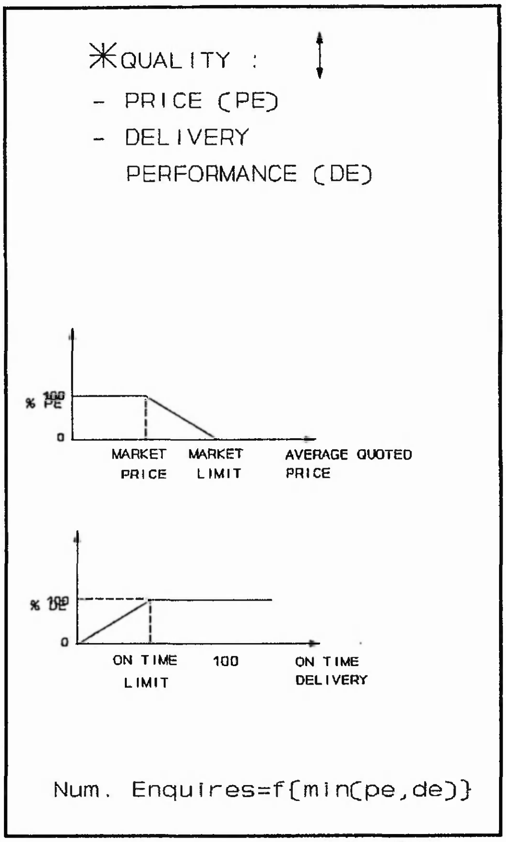


Figure 4.6 Feedback approval functions.

Hence, the ultimate model that simulates the feedback process is function of a combination of the previous two factors (minimum of the two factors) and the average number of enquiries so far. If the minimum of the two factors is low, the number of customer enquiries for the next simulation period will be reduced. On the other, hand if the two factors are more positive then the number of enquiries for the next simulation period will be increased.

The use of the average number of enquiries as an element of the feedback system is due to the fact that a malfunction in the factory performance will not affect the factory's competitiveness in a drastic and immediate way, rather in a smooth way. In other words, if the factory does not perform well today, it does not mean that this afternoon will be out of business. To implement this idea, a numerical function (depending on the number of days already simulated had to be added to the feedback system (figure [4.7]). The reason for that was that the effect of the average number of enquiries in the feedback system could not properly be seen in the defined gaming time and therefore, it had to be featured.

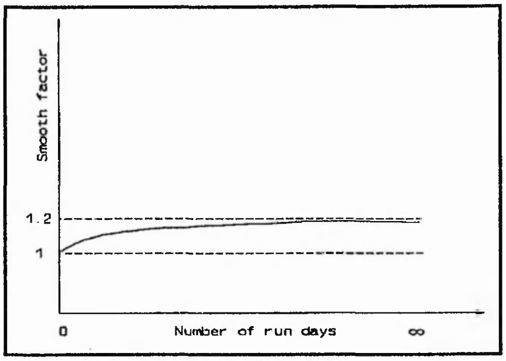


Figure 4.7 Smoothing function.

Consequently, the feedback function is :

Number of enquiries = 1 + [((6n-1)/5n)*average number of enquiries*min(pe,de)]

where "n" is the number of periods already gamed, and "min(pe,de)" is the minimum factor (percentage) associated to the market price and due date performance. It has to be said that the number of enquiries can never be zero, it will be at least one. That feedback element is calculated for each one of the final products in the factory being simulated.

There is also another important element in the model which has not been mentioned yet, but is primarily present in all aspects and stages of the model. That important element is **Time**.

4.3.3.5 THE TIME ELEMENT

Its relevance is based on the dynamic nature of games, and on the use of time as a factor in the decision-making processes embedded in a management game. As Kibbee et al. [76] expressed long ago :

"*The second major contribution of management games from the learning standpoint is the time dimension...the game has advantages over a case study in the training of a manager. It is alive. Its state is constantly changing in response to previous actions. Planning must consider the present and future simultaneously. Emergencies must be anticipated. Crisis must be met. With no other teaching technique has it been possible to demonstrate so vividly the effects of sequential decision-making in a business environment.*

Time is an important factor in games in another way. Participants are often placed under conditions of severe time limitations to simulate the stress encountered in a real management situation".

Whilst the claims for the "dynamic" aspect of gaming-simulations is an aspect to be confirmed by this thesis's work, a major characteristic which detracts from this "dynamism" is that decision-makers in most of the games are forced into making decisions on a regular cyclical basis. This characteristic forces games into an analytic framework and into an unrealistic environment in which all decisions are made regularly and simultaneously.

For a game regarded as truly "dynamic" it is vitally important, to integrate the time dimension into the structure of the game.

Based on the former assertion, Nottingham Polytechnic's gaming-simulation has tried to address the time element in a positive way. The approach taken for that, is based on the inclusion of **time** as an "endogenous" factor [77] within the game. That method, opposed to the inclusion of **time** as an "exogenous" factor [77], will permit the player effectively use **time** as part of his strategy. That approach is highly encouraged by Loveluck [77], who thinks that to reflect contemporary management practice, time must be introduced as an endogenous variable. In his opinion this must be done by developing "real-time" games which will involve the formulation of multi-user computer models, into which time compression and consumption is made a specific decision alternative. Thus, one of the objectives of this work is to study the feasibility of the aforementioned "real-time" games.

There are several possible ways in which this "real-time" games may be created. Nottingham Polytechnic's gaming-simulation has done it in two different ways.

First, **time** has been introduced as an "endogenous" factor by simulating the passing of each day (at a rate preestablished by the game manager; e.g. 3 minutes per day) regardless of the decisions made. The point in time at which decisions are made becomes a decision in its own right; timing of decisions becomes part of team strategy. The difficulty of this approach lies in the technology used (hardware and software), a limitation experienced throughout this thesis's work.

Second, the model has been developed in a way that will allow the simulation to be stopped (frozen by an interrupt function); avoiding, at the same time, the input of data from any other of the functions involved in the game. A dialogue can then be set up which allows the participants to discuss their behaviour and address their worries.

It is only under these conditions that the timing of decisions becomes a genuine option in the decision-makers' range of alternatives, and that games become models of dynamics rather than comparative statics.

4.3.4 DECISIONS ABOUT REPRESENTATION

The decisions to be made at this stage of the gaming-simulation design process are of two different natures. They either are related to the style of the game or to its form.

4.3.4.1 STYLE

The dimensions of the gaming-simulation related to the style can be divided into the following three components :

4.3.4.1.1 LEVEL OF ABSTRACTION

In Nottingham Polytechnic's gaming-simulation the discussion of the level of abstraction is very much related to the use of a computer as the mechanism for playing the game.

The game is shown to the players in a numerical way. Players are dealing with a set of data, that give them information about the status of the different areas involve in the simulation of the manufacturing organisation. Moreover, players response to the game is also a numerical one. They have to input numerical data in order to answer to the situation they are presented with.

4.3.4.1.2 TIME FRAME

Another aspect that must be considered when deciding about the game's representation is **time**. The more "realistic" the gaming-simulation is, the more important this factor is.

There are two elements related to the time aspect. One is the real time that players spend playing the game and another one is the time that is going to be represented in the game. The former has already been defined in the "Context of Use" section in the stage about **SETTING OBJECTIVES AND PARAMETERS**, now the latter element has to be defined.

The period to be represented is related to the subject matter and purpose specified in the stage about objectives and parameters.

There are actually two aspects of the time to be represented that must be considered: the length of total time to be represented by a game session, and the length of time to be represented by a cycle. The former must be long enough to encompass the full expression of the behaviour and to show the response of the system to the players' principal decision options. The latter must be fine enough to show the true shape of the dominant dynamics.

In Nottingham Polytechnic's gaming-simulation, the decided starting time frame is based on having a cycle representing at least 10 days (8 working hours per day) played twice, and separated by a discussion about the first cycle.

Anyway, the time of representation should be continually reviewed and adjusted based on the experience gained during the played gaming sessions. For that to be possible, the gaming-simulation had been constructed in a flexible way, allowing the game manager to adjust that period time to whatever he/she thinks is appropriate.

4.3.4.1.3 INTERACTION

The decision about the kind of interactional structure to be used in the gaming-simulation is also an important feature, and is inherent in the nature of the game being explained in this thesis.

Nottingham Polytechnic's game is considered to be a highly interactive one, due to the nature of the manufacturing relations that are being presented. Different players are linked to a network which manages the game's common files. This high interaction implies that the game is a more looser, more informal, and less predictable than tightly-structured games.

4.3.4.2 FORM

Another matter when making decisions about representation is the question of form and of how to incorporate each element of the conceptual model into the gaming-simulation. This question can be divided into six different elements (scenario, roles, procedures and rules, external factors, visual imagery and symbols, and accounting system).

4.3.4.2.1 SCENARIO

The players in Nottingham Polytechnic's game are situated in a manufacturing organisation structured in a functional way. The two functions the company consists off are : Marketing/Sales and Production Planning/Control.

In this scenario, it was decided that the manufacturing company to be simulated had to be quite simple to start with and within a game structure capable of handling more complex future models.

The initial limitations were :

- Five final products.
- No parts. Products are manufacture with no material.
- One operation manufacturing process.
- Two resources
  - One produces three final products.
  - The other produces the other two.
- Make to stock

Then, the problem that players have to face in this "simpler", but flexible manufacturing context, is the one of coordination between the mentioned functions in order to make the whole company to success.

4.3.4.2.2 ROLES

The kind of roles associated with the gaming-simulation are related to the two existing functions within it.

The players that undertake the tasks connected to the Marketing/Sales department, will have to answer the incoming enquiries, by quoting their prices and delivery dates.

On the other hand, the players concerned with the Production Planning/Control department will have to define the adequate schedule (quantity and sequence) required to produce enough stock for the customer orders to be delivered on time. Their decisions will be based on the existing information about the firm customers' orders.

In both simulated departments the decision-making process is subject to a situational stress. That stress pressure is due to two factors : the time pressure element (explained in the model development stage) and the inadequacy of the information they have to base their decisions on. That latter factor is a vital educational element for addressing the importance of coordination.

4.3.4.2.3 PROCEDURES AND RULES

The decisions about procedures and rules of a gaming-simulation are very important when dealing with non computer based games. But when dealing with computer games, the rules are determined by the designer's decisions about the structure and functioning of the program/s, and by the set of game parameters decided by the game manager.

4.3.4.2.4 EXTERNAL FACTORS

The Nottingham Polytechnic's gaming-simulation model has only two external factors : the customer enquiries entering the system and the firm customer orders to be delivered.

Both of them are regulated by functions that use pseudo-random numbers, as was explained in the previous stage of the design process. The information attached to those factors, concerns the quantity of the enquiry/order and its expected delivery date. Those values are calculated with pseudo-random numbers and will vary between preestablished values, that are parameters of the game that may be modified by the game manager.

4.3.4.2.5 VISUAL IMAGERY AND SYMBOLS

The definition of the visual representation to be used in the gaming-simulation is very much related to the materials and equipment used for its construction.

The gaming-simulation concerning this thesis is a computer based one, and therefore its equipment is based on two elements : Hardware and Software.

As it was explained in the design stage related to the setting of objectives and parameters, the hardware use is an existing network of PCs (NOVELL [96]). The software used to create the program is a database management system capable of handling great amounts of information in a multi-user environment. The software chosen was FOXBASE+ [78], and the reasons for that were based on the fact that the system provided a very convenient means of creating certain features of the game (screens, menus, etc) and at the same time it was one of the fastest DBMS available. The database approach also provided the means for concurrent information availability, a element of recognised importance [79].

In this context, the visual representation chosen was a "browsing system", that can be defined as a full screen display system with input and data selection capabilities. That "browsing system" can be seen in figure [4.8], and will also be referred to in the chapter about the software design.

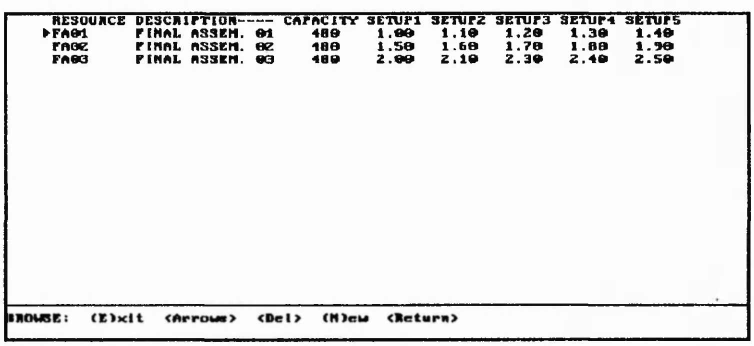


Figure 4.8 Browsing system.

4.3.4.2.6 ACCOUNTING SYSTEM

When defining the elements of Nottingham Polytechnic's gaming-simulation, the performance measure system (accounting system) was defined as one of the important elements to enter the model.

As it was explained in the second chapter of this thesis, the "reward" system is one of the elements behind the conflicts between local and global objectives, in other words between the existing functions or areas of a manufacturing organisation and the organisation itself. A conflict that is the core subject of the educational objectives to be addressed in this thesis's game.

With that in mind, the game is based on the "competition" of different functional elements within an organisation, rather than the competition for an existing market between different vendors. In that context, it can not be clearly defined who is doing well or winning. However, what can be fostered is a discussion about how the different functions and the whole company are performing.

To "measure" that performance, two kind of performance measures were defined : local and global ones. While the local performance measures were defined to be associated with the performance of the different functional departments being simulated, the global ones were defined to be related to the performance of the company, as a whole. That arrangement was thought to make players aware of the fact that although the performance of each one of the departmental functions could be very positive, the performance of the company as a whole might not. The system was also thought to make people aware of the interrelations and conflicts between the performance in the different departments.

The selected local performance measures were :

■ <u>Marketing/Sales Department</u>

- ♦ Number of enquiries received
- ♦ Number/Percentage of enquiries quoted
- ♦ Number/Percentage of firm orders
- ♦ Number of enquiries to be delivered on the current period
- ♦ Due date performance
- ♦ Average delivery delay
- ♦ Volume of sales orders

■ <u>Production Planning/Control</u>

- ♦ Number/Percentage of incomplete orders
- ♦ Number/Percentage of started orders
- ♦ Number/Percentage of finished orders
- ♦ Each product's production
- ♦ Number of set-ups scheduled
- ♦ Time/Percentage spent in set-ups
- ♦ Level of stock

On the other hand the selected global performance measures were :

- ♦ Throughput
- ♦ Inventory
- ♦ Operating Expenses

These are measures that could easily be related to standard financial measures as :

- ♦ NET PROFIT (NP) = THROUGHPUT - OPERATING EXPENSES
- ♦ RETURN ON INVESTMENT = NET PROFIT (NP) / INVENTORY

These are some of the performance measures as proposed by people like Goldratt [80], William J. Bruns et al. [81], and S. Kaplan [82].

All of the former mentioned performance measure elements, are calculated for each one of the simulated periods, also keeping a record of the average performance so far.

Coming back to the stages concerned with the design of gaming-simulations, and once the decisions about representation have been made, it is time to start constructing the game.

4.3.5 CONSTRUCTION AND MODIFICATION OF THE GAMING-SIMULATION

The construction of gaming-simulation was based on the prototyping methodology [83]. The game construction is based on building and using a model of a system for designing, implementing, testing, and installing the final game. When all game parts are assembled into a prototype, the gaming-simulation must be field tested, debugged, retested, and so on until it operates successfully. Considerable trial and retrial is usually necessary before the game generates the behaviours and outcomes characteristics of the referent system. Each error or disappointment, leads the designer to a reexamination of the conceptual model and the design decisions made, as well as to check for possible failures in the construction process; all that in search of a missing element or elements inaccurately linked to others. This whole process could be quite disappointing, for the point at which you are "almost there" but not "there yet". Greenblat [55] suggests that when the game has run smoothly 10 times, it is time to proceed to the next and final stage of the design process.

In theory, at this stage of the design process, it has to be decided whether to use a computer for game operations, and if so construct the needed programs. Due to the nature of the objectives of this research the decision about the use of computers for game operations was a component inherent to the research in itself. Nottingham Polytechnic's gaming-simulation was, by definition, to be developed on computers linked to a network.

With this in mind, and due to the complex nature of the technical matters involved in the construction and development of the programs discussed in this thesis's computer game, there is a whole section named "SOFTWARE DESIGN" dedicated to this aspect.

4.3.6 PREPARATION FOR USE BY OTHERS

After looking at the four previous stages of the gaming-simulation's design process, it is time to look at the last of the stages. The stage related to the preparation of the gaming-simulation for use by others.

This stage is an important and often complicated one. This stage of the design process is carried out when the designer is satisfied with the way the gaming-simulation works (i.e. technically) and corresponds to the "real-life" situations.

The stage about preparation of the game for use by others, involves the creation of the operator's manual and the dissemination of the gaming-simulation. The operator's manual should have a technical overview, a explanation of the game's conceptual model, a description of the game in operation, how the game should be run, and a guide to post-play discussion. Additionally, the dissemination involves the preparation for the distribution of the gaming-simulation and its review, both informally and formally.

This preparation stage has not been accomplished during the design of Nottingham's Polytechnic gaming-simulation. The reason for that is based on the fact that the result achieved is not completely satisfactory. Although it is thought that the educational purposes could be accomplished by a basically similar model (user friendlier), the existing technical limitations make the game non-playable at an advance stage of the game (e.g. too much information to handle, the game slows down at a great pace, etc).

5 SOFTWARE DESIGN

5.1 GENERAL ARCHITECTURE

The architecture of the system is based on a network of PCs (NOVELL [96]) and database management system (FOXBASE+ [78]). The system uses three personal computers and the network's file server. While two of the personal computers will respectively simulate input/output tasks related to the Marketing/Sales and Production Planning/Control departments, the third personal computer is used to control the game (simulate the time system) as well as simulate the run of the factory (generate enquiries, simulate the approval of quotations, simulate the production process, and simulate the delivery of customer orders). On the other hand, the file server will store the data files being used in the gaming-simulation, allowing their access from any of the personal computers. The actual hardware architecture of the system is shown in figure [5.1].

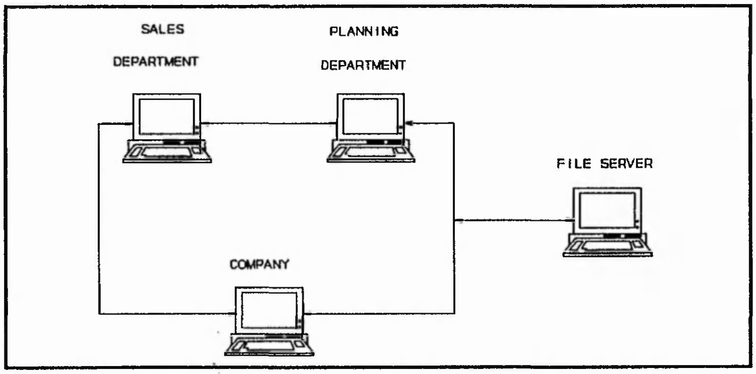


Figure 5.1 Game's computer architecture.

The flow-chart of the whole game can be seen in figure [5.2]. That figure shows the different data files used in the game as well as the relations among them.

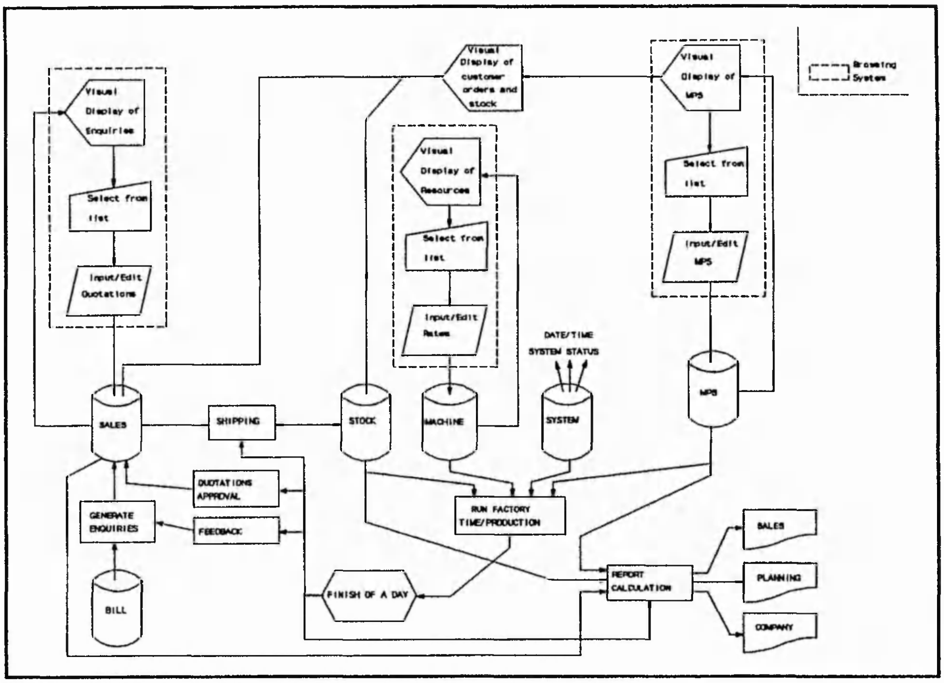


Figure 5.2 Game's flow-chart.

Each computer runs its own specific program [Appendix C], the important elements of which are described in the section about "technical problems".

5.2 USERS' VIEW OF THE GAME

Nottingham Polytechnic's gaming-simulation is based on four screen handling facilities : menus, browsing system, report system and messages.

The menu systems gives the users the opportunity to decide which kind of activity they are going to perform. The browsing system on the other hand, is related to the more specific data view and edition activity.

The report system will show the information concerning each department's daily and overall performance, as well as the global company's one. Finally, the messages are used to inform the players about the status of the game.

5.2.1 MENU SYSTEM

The menu system is used in different computers to perform different activities. In the computer simulating the factory run there are different menus that will allow the game manager to :

- Configure the resources of the factory (figure [5.3], figure [5.4])

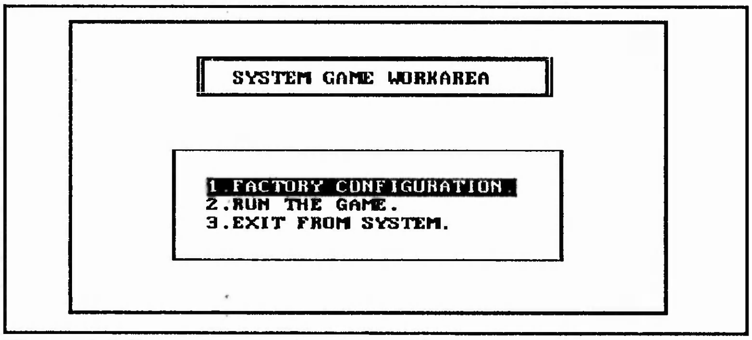


Figure 5.3 Game manager's menu option.

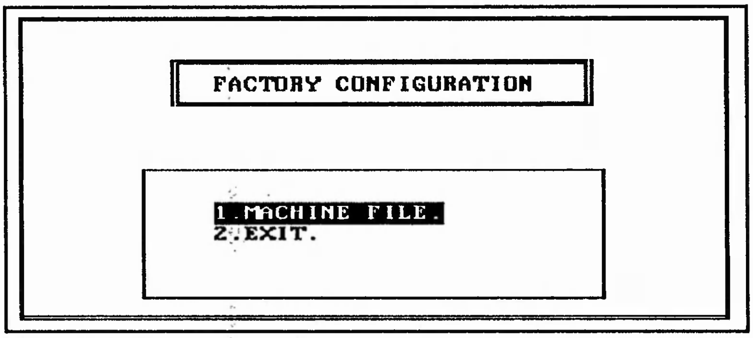


Figure 5.4 Game manager's menu option.

- Start the game (figure [5.5]).

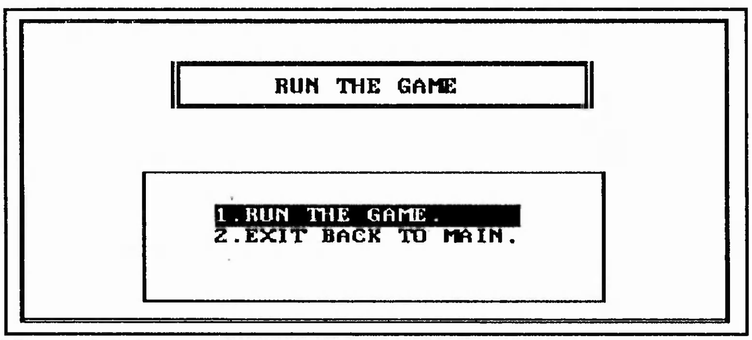


Figure 5.5 Game manager's menu option.

- Save a day's final data, run another day, or exit the game (figure [5.6])

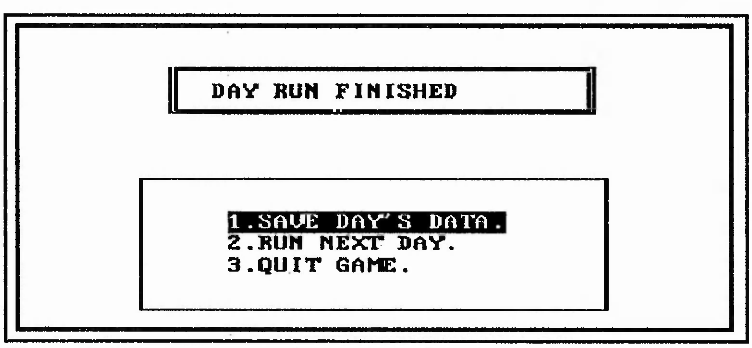


Figure 5.6 Game manager's menu option.

● To continue or quit a "frozen" game (figure [5.7])

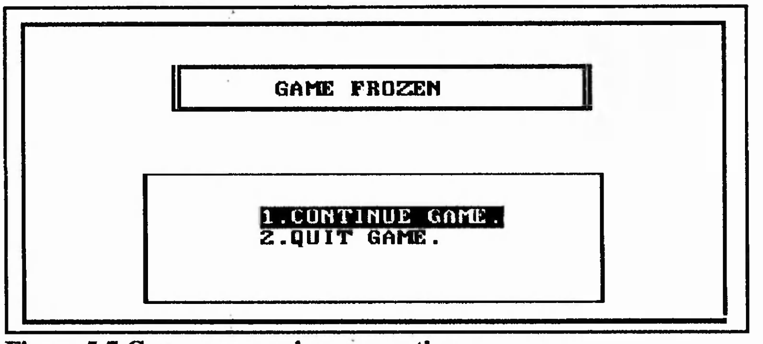


Figure 5.7 Game manager's menu option.

The program simulating the Planning area works with only one menu system. That one, allows the user to select the printer as the reports' output device and display the reports (figure [5.8]).

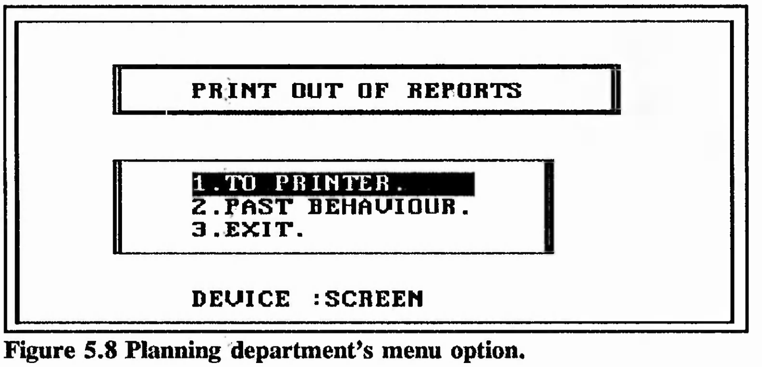


Figure 5.8 Planning department's menu option.

The program simulating the Marketing/Sales area uses the previous report menu system, as well as another one that allows the player in this area to input the standard price for each one of the final products, or select the display/input of quotations (figure [5.9]).

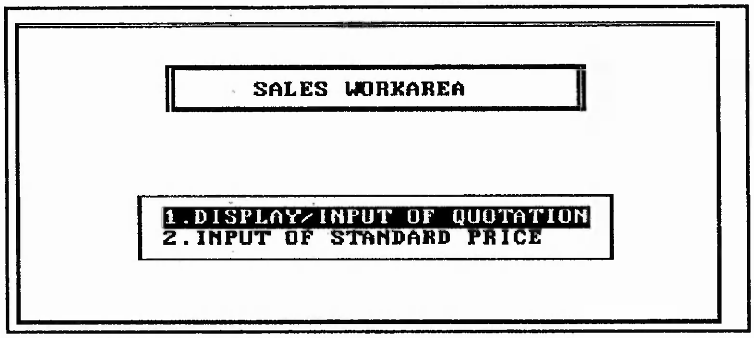


Figure 5.9 Marketing/Sales department's menu option.

5.2.2 BROWSING SYSTEM

The "browsing" system, as it has already been defined in the design stage about representation, is a full screen display system with input and data selection capabilities.

The information displayed on the browsing system depends on the sub-program that the browsing system is affecting to.

The browsing system for the Marketing/Sales department, for example, displays information about the coming enquiries (figure [5.10],[5.11]), giving the players the possibility to :

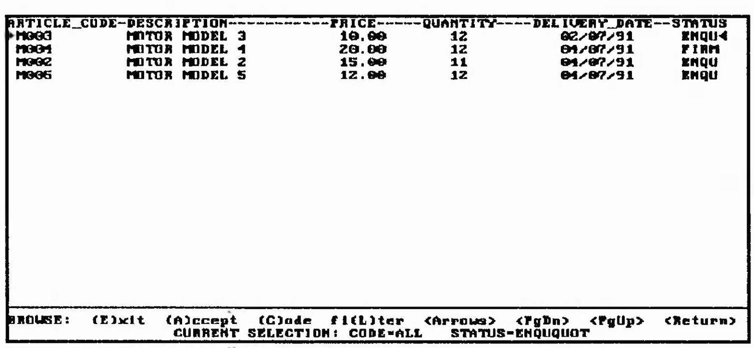


Figure 5.10 Enquiries' browsing.

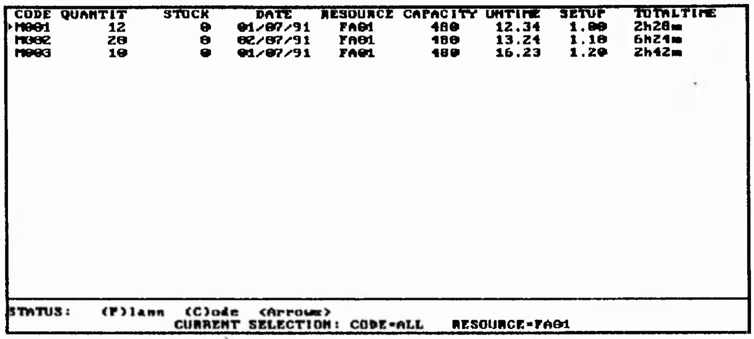


Figure 5.11 Enquiries' browsing.

- Quote them as enquired » (A)ccept
- Quote something different » <Enter>
- Select the enquiries by code » (C)ode
- Select the enquiries by resource code » (R)esource
- Select the enquiries to quote » <Arrows, PgUp, PgDn>

The one associated with the Planning Department, on the other hand, displays information about the customers' firm orders and about the scheduled orders (figure [5.12], [5.13]). The key choices for these two displays are :

- ■ Customers firm orders' display :

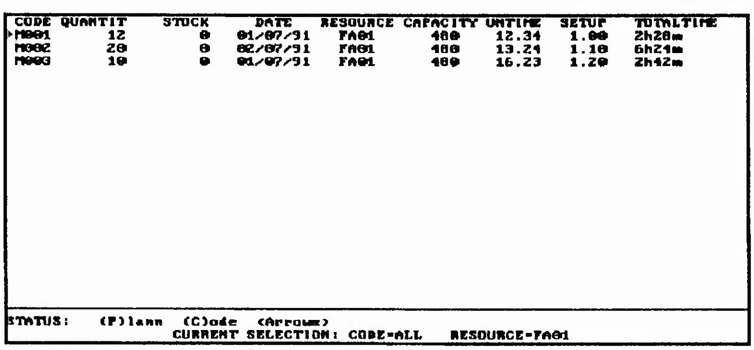


Figure 5.12 Customer firm orders' browsing.

  - Display the scheduled orders » (P)lan
  - Select the firm orders by code » (C)ode

- Select the firm orders by resource code » (R)esource
- Select the customers' firms orders » <Arrows, PgUp, PgDn>

■ Scheduled orders' display :

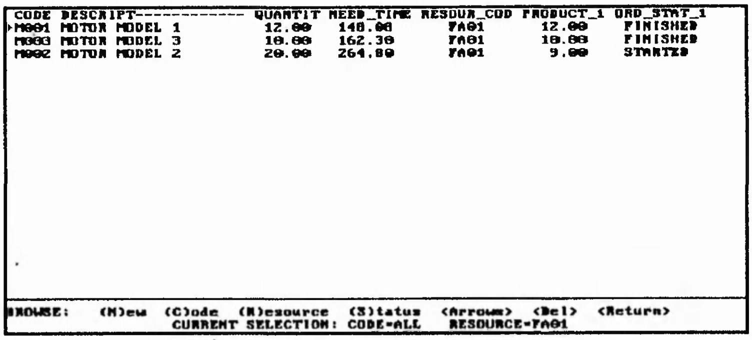


Figure 5.13 Scheduled orders' browsing.

- Display the customers' firm orders » (S)tatus
- Create a new scheduled order » (N)ew
- Select the scheduled orders by code » (C)ode
- Select the scheduled orders by resource code » (R)esource
- Select the scheduled orders to be re-scheduled » <Return>
- Delete the selected scheduled order » <Del>
- Move through the scheduled orders » <Arrows, PgUp, PgDn>

There is also another browsing system associated to the computer simulating the run of the factory. That one is used to input information about the resources within the company to be simulated. That information is related to set-ups and capacities (figure [5.14]). The browsing system gives the game managers the possibility to :

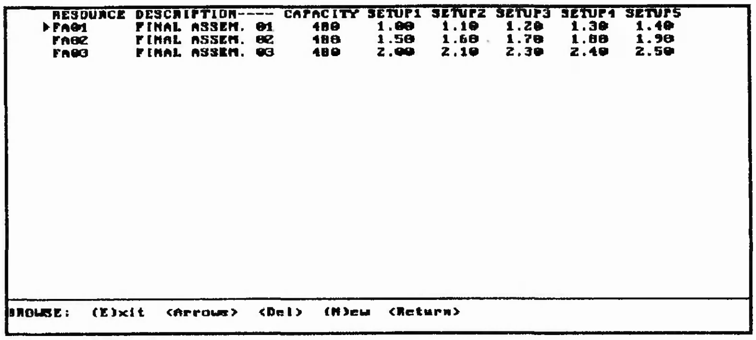


Figure 5.14 Resources' browsing.

- Exit this display » (E)xit
- Move through the existing list of resources » <Arrows>
- Delete an existing resource from the list » <Del>
- Create a new resource » (N)ew
- Edit an existing resource » <Return>

Because the game is played interactively, the browsing screens are refreshed as enquiries are received and quoted, and orders are produced. After a day has finished they will also be refreshed.

5.2.3 REPORT SYSTEM

The report system used in all program areas (Factory, Planning, and Marketing/Sales) is a full screen display of the performance measures associated to each one of the areas. Those reports can also be diverted to the printer.

The factory reports, for example, show the global performance measures associated to the whole of the factory (figure [5.15]).

FACTORY'S PAST PERFORMANCE

THROUGHPUT :	£ 1,236
STOCK :	£ 1,035
OPERATING EXPENSES :	£ 1,000
NET PROFIT :	£ 236
RETURN ON INVESTMENT :	£ 12.86 %

Figure 5.15 Factory's performance.

On the other hand, the reports associated to the Planning area display information about the WIP, production, set-ups, stock levels, etc...(figure [5.16],[5.17],[5.18],[5.19]).

PLANNING DEPARTMENT'S PAST PERFORMANCE (Page 1)

Week 0 Date 01/07/91

Code M003 Descript MOTOR MODEL 3

	DAY	OVERALL		DAY	OVERALL
Num. of incomplete orders	0	0	% over total orders	0.00	0.00
Num. of started orders	0	0	% over total orders	0.00	0.00
Num. of finished orders	1	1	% over total orders	100.00	100.00

Figure 5.16 Planning reports.

PLANNING DEPARTMENT'S PAST PERFORMANCE (Page 2)

Week 0 Date 01/07/91

Code M005 Descript MOTOR MODEL 5

	DAY	OVERALL		DAY	OVERALL
Production	10.00	10.00	Num. of setups	1	1
Batch size	10.00	10.00	Lead-time	196.00	196.00
Wip	0.00	0.00	Standard cost	269.52	269.52

Figure 5.17 Planning reports.

```
            PLANNING DEPARTMENT'S PAST
              PERFORMANCE (Page 4)

Week                0   Date      01/07/91
Code             M001   Descript  MOTOR NUMBER 1

                            DAY      OVERALL
Level of stocks               0            0
```

Figure 5.18 Planning reports.

```
            PLANNING DEPARTMENT'S PAST
              PERFORMANCE (Page 3)

Week                0   Date      01/07/91
Resource         FA01   Descript  FINAL ASSEM. 01          Machine Capacity   400

                            DAY      OVERALL
Number of Set-Ups             3         1                                    DAY     OVERALL
Machine Production       476.70    238.35   Machine Productivity         59.31     49.66
Time spend in Set-Ups      3.30      1.65   % of Set-Ups                  0.69      0.34
```

Figure 5.19 Planning reports.

Finally, the reports concerning the Marketing/Sales area will show information about the enquiries received and quotations made (figure [5.20],[5.21]).

```
            SALES DEPARTMENT'S PAST
             PERFORMANCE (Page 1)

Week                0   Date      01/07/91
Code             M001   Descript  MOTOR MODEL 1

                              DAY     OVERALL
Number of enquires           1.00      1.00                            DAY     OVERALL
Number of quotations         0.00      0.00   % over enquires         0.00      0.00
Number of sales orders       0.00      0.00   % over quotations       0.00      0.00
To be delivered                 1         1
Shippings                       1         1   % shipped on time     100.00    100.00
                                              DAY           OVERALL
              Volume of Sales orders (£)       0.00            0.00
              Average Delivery Delay          0.00            0.00
```

Figure 5.20 Marketing/Sales reports.

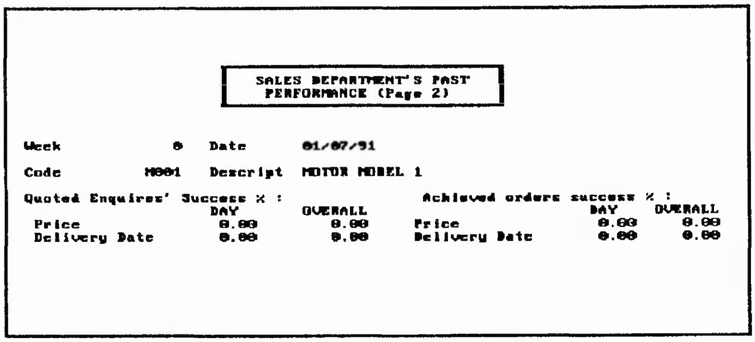


Figure 5.21 Marketing/Sales reports.

5.2.4 MESSAGES

The programs in the game use two different kind of messages. Some of them they are simple information messages, which give information about the status of the system or different kind of warnings (figure [5.22], [5.23], [5.24], [5.25], [5.26], [5.27], [5.28], [5.29], [5.30], [5.31]).

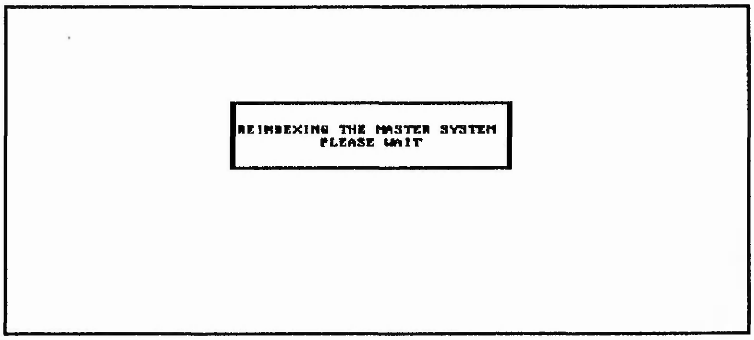


Figure 5.22 Messages.

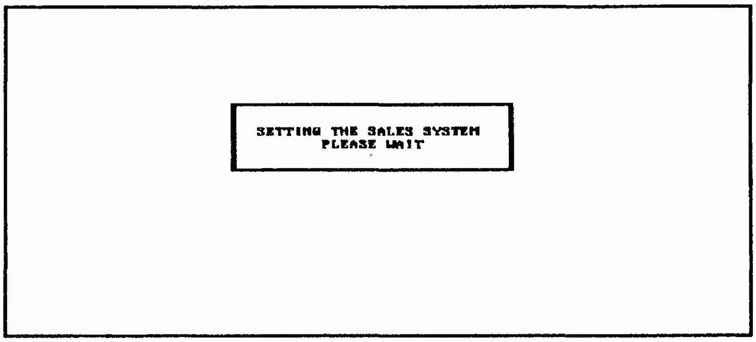


Figure 5.23 Messages.

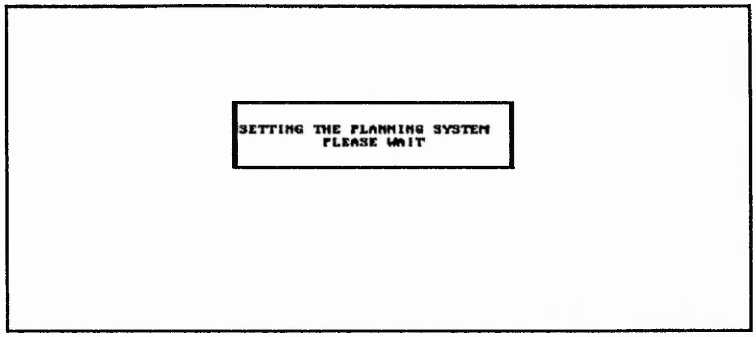


Figure 5.24 Messages

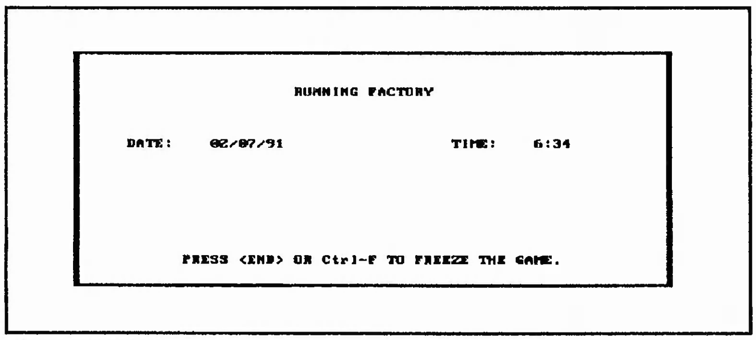


Figure 5.25 Messages.

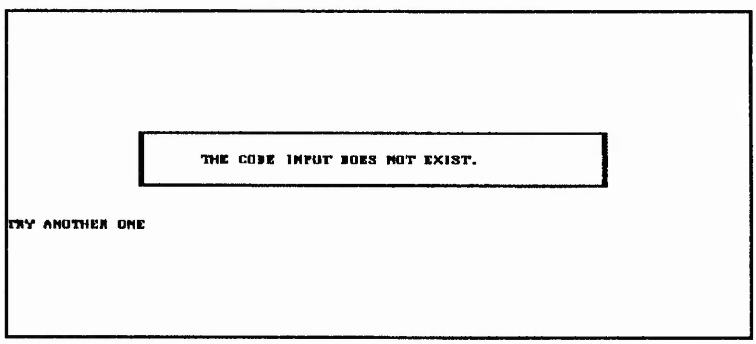


Figure 5.26 Messages.

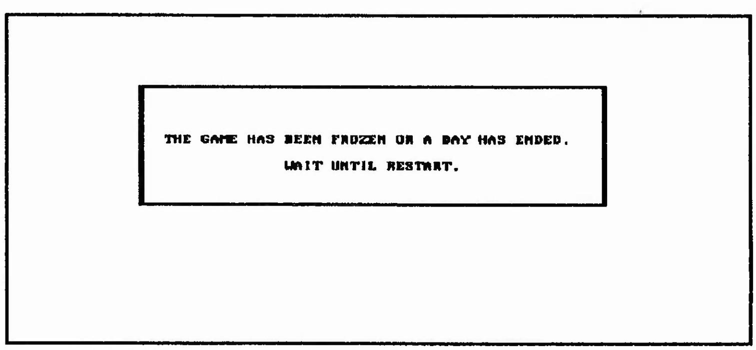


Figure 5.27 Messages.

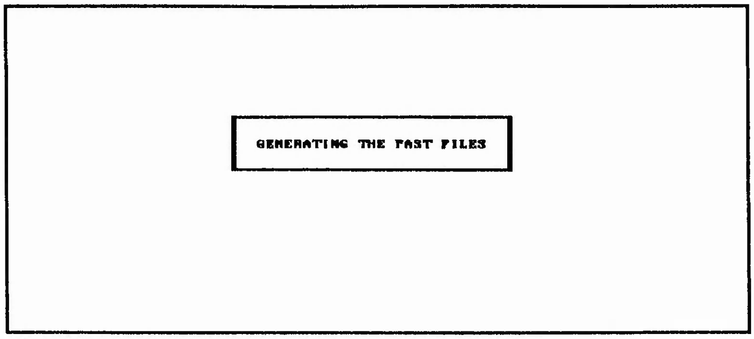


Figure 5.28 Messages.

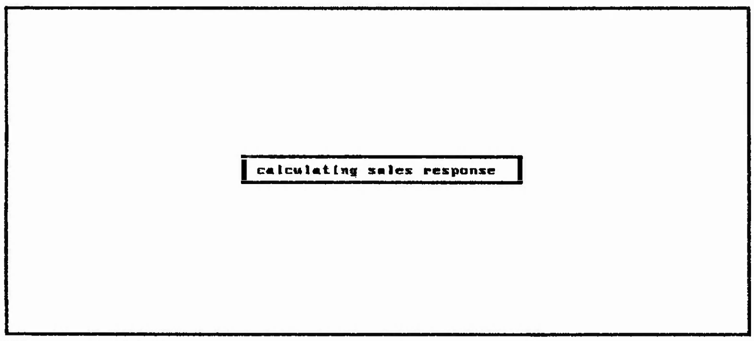


Figure 5.29 Messages.

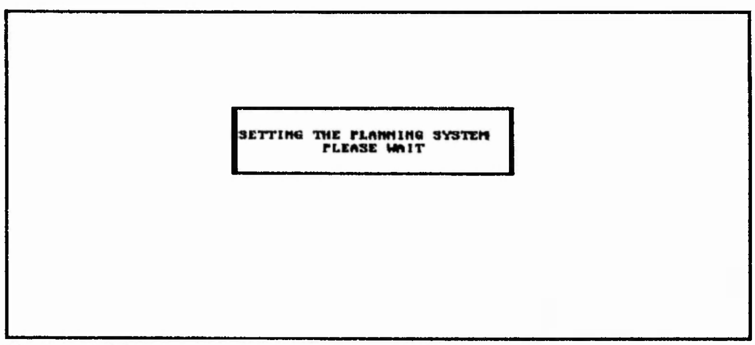


Figure 5.30 Messages.

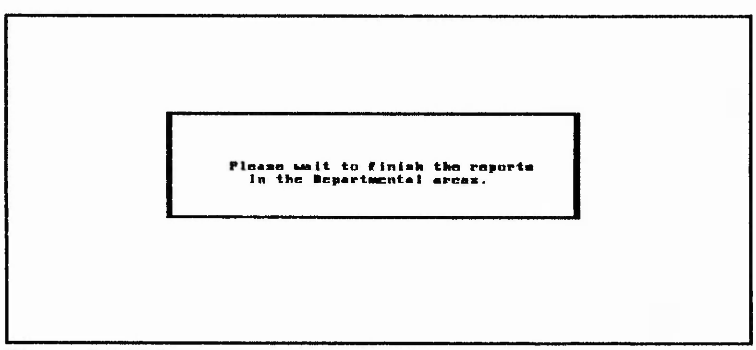


Figure 5.31 Messages.

There are some other messages that are input screens, which are used to input information regarding different elements of the game as input of quotations, configuration of dates, input of standard price, and so on (figure [5.32], [5.33], [5.34], [5.35], [5.36]).

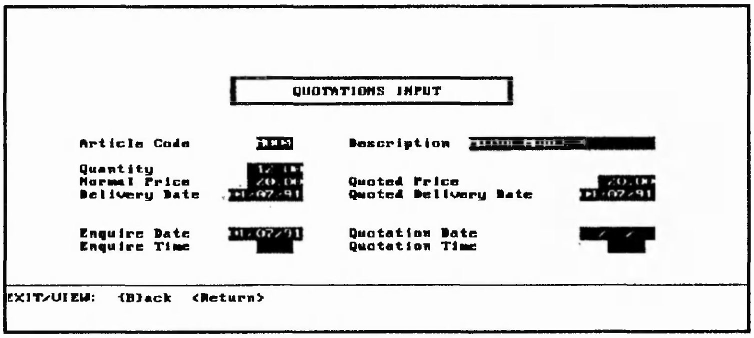


Figure 5.32 Input screen.

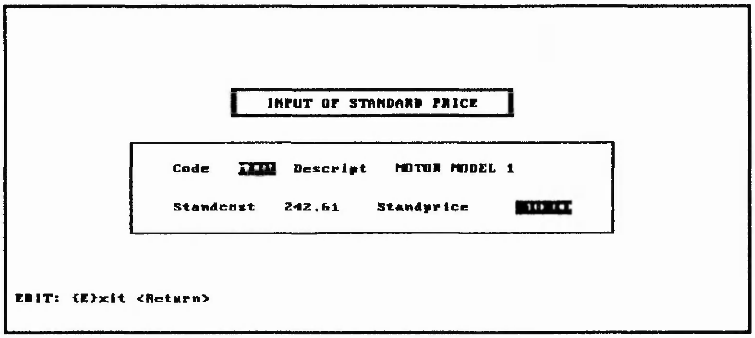


Figure 5.33 Input screen

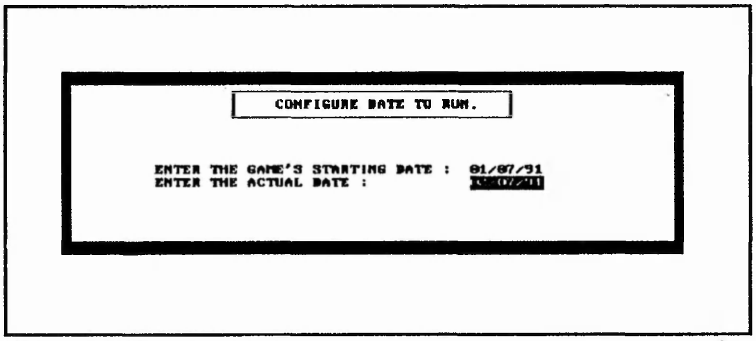


Figure 5.34 Input screen.

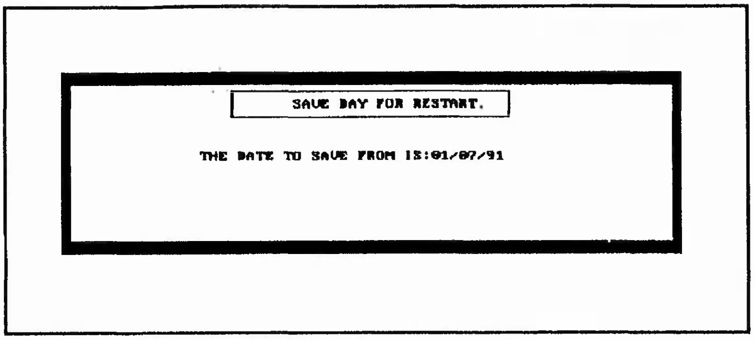


Figure 5.35 Input screen.

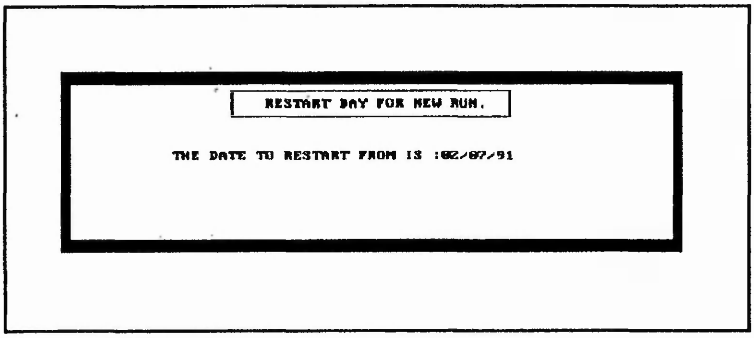


Figure 5.36 Input screen

5.3 DEALING WITH TECHNICAL PROBLEMS

The source code and structure of each of the sub-programs, as well as the data files names, structures, and associated index files can be found in [Appendix C].

Although FOXBASE+ [78] offers many good programming features, it is not a system specifically designed to cope with simulations. Moreover, things get complicated when trying to simulate "real-time" simulations. Therefore, there are some technical problems that need to be solved by the creation of suitable programs and procedures.

5.3.1 FILE CONFLICTS

Nottingham Polytechnic's gaming-simulation is based on a multi-user environment. In this context it is necessary to have a procedure to deal with the data access conflicts, i.e. one user attempts to access a record or file which is currently being accessed by another user.

FOXBASE+ [78] offers the existence of two locking functions, RLOCK() and FLOCK(), that deny the use by others of a record or a file respectively. Those two functions together with the "ON ERROR" command line offer the possibility to make the program behave in one way or another when a data conflict arises. Therefore, and based on those elements it was created a procedure called "err_fix" [Appendix C] for handling the data conflicts in each one of the sub-programs.

5.3.2 COORDINATION

When trying to simulate a "real-time" gaming-simulation in a personal computer under the DOS operating system [84], there is one major limitation to be aware of. That limitation is the impossibility of multitasking (performance of different instructions at the same time).

That restriction restrains the creation of gaming-simulations based on a continuous algorithm. In other words, the computer simulation will perform tasks on a sequential loop, in contrast to real-life situations, where multiple events happen at the same time. That implies that the information concerning the simulation will not be continually updated (i.e. real-time system). The information will only be updated every time the loop enters a new cycle in which all the different calculations will be sequentially calculated. The cycle time is variable and dependent on the time needed to carry out each of the main program tasks.

That loop will continually run until the game is stopped by the game's manager or a simulated period has come to its end. The loop will always start checking for the former situation to happen. In second place the program will calculate the absolute simulation time (i.e. simulation time elapsed since the start of the game). If that absolute simulation time has changed since the last loop, then a new calculation cycle will be started. That calculation cycle will involve the transformation of the simulation absolute time into simulation time (i.e. simulation's running day, month, year, hour, and minutes), and the simulation of the production system. Finally the program will check for the gaming period to finish (i.e. a days's simulation has finished). If that is the case, the program will force the run of tasks associated to the finish of a period : the quotation approval, shipping, feedback, and enquiries simulation systems. The cyclical algorithm of the simulation is shown in figure [5.37], and the procedure that performs that algorithm is called "RUNFACT" in the factory program [Appendix C].

```
Loop indefinitely during game until quit (FACSTOP and DAYREPORT)

        Display the running screen (Date and Time of the game)
        IF <End> or <Crtl-F> pressed
        THEN
             > Record system stop time (STARTSTOP)
             > Allow the game to continue, restart a new day or quit (Proc. FACTSTOP)
             > Record ammount of time the game was frozen (DIFSTOP)
        ENDIF
        Calculate new absolute game time
        IF absolute game time has changed since last loop
        THEN
             > Update game's playing date and time
             > Calculate production achieved during the period
             > IF day has changed
             > THEN
                   >> Record the time the game stopped (STARTSTOP)
                   >> Simulate the quotation approval
                   >> Simulate the shipping of the customers' orders
                   >> Update the Planning's Status file
                   >> Generate the day's summary reports
                   >> Simulate the Feedback system
                   >> Generate the enquiries file for the coming day
                   >> Record amount of time the game was frozen (DIFSTOP)
             > ENDIF
        ENDIF
End of Loop
<END>
```

Figure 5.37 Nottingham Polytechnic's cyclical algorithm.

The information calculated in each one of the algorithm's cycles is sent to the different data files for updating. That information will also be used to coordinate the two departmental programs with respect to the running of the simulation.

The only way information can migrate from one computer to another, within a multi-user database management system, is by data files. Therefore, the program created to operate Nottingham Polytechnic's gaming-simulation has been constructed with a control data file called "SYSTEM.DBF" [Appendix C]. That data file is a one record file and contains all the relevant variables needed to control the simultaneous run of the three departmental programs (e.g. running date and time, game's status [frozen, period conclusion, game finished,..], game's global settings, etc.). This enables the game manager to control the other players. For example, if the game is stopped (frozen or a day has ended) the departmental players will not be able to input data. If the day has finished, reports will appear in each one of the departmental computers,etc..

5.3.3 TIME SIMULATION SYSTEM (RUN FACTORY)

The time simulation procedure is based on a function of the FOXBASE+ [78] database management system. That function, called "sys(2)", returns the number of seconds elapsed since midnight.

The first value returned by that function will defined the simulation's absolute starting time. Based on that starting absolute time, the time simulation procedure will calculate the consequent relative times. These relative times will later be affected by an amplification factor that will finally return the actual simulation time. That simulation time will again be transformed to a suitable form for displaying and processing (i.e. "DAY/MONTH/YEAR" «» "HOURS:MINUTES").

The procedure used to carry out all this calculations is called "TIMECAL" in the factory sub-program (SYS11.PRG), and can be seen in [Appendix C].

5.3.4 PRODUCTION SIMULATION SYSTEM (RUN FACTORY)

One assumption of the production simulation system is that a manufacturing resource can not cope with more than one production order at a given time. It also takes into account the influence of the set-up times for each product.

Another important element of this calculation is the aforementioned sequential execution of the programs under the DOS operating system [84]. The sequential execution implies, that the production simulation will be carried out by calculating the situation at the end of each simulation cycle period. The stock and production situation will only be known at the end of each cycle, but not during it.

If that is the case, there are three kind of orders and two set-up situations that had to be taken into account, when making the calculations :

» Orders :

- Order starting and finishing in the current cycle period.
- Order not starting but finishing in the current cycle period.
- Order starting but not finishing in the current cycle period.

» Set-up situation :

- Order with set-up equal to the previous order.
- Order with set-up different to the previous order.

Those elements can be combined into six different calculational situations that the production simulation program will have to cope with. These six different situations are shown in figure [5.38].

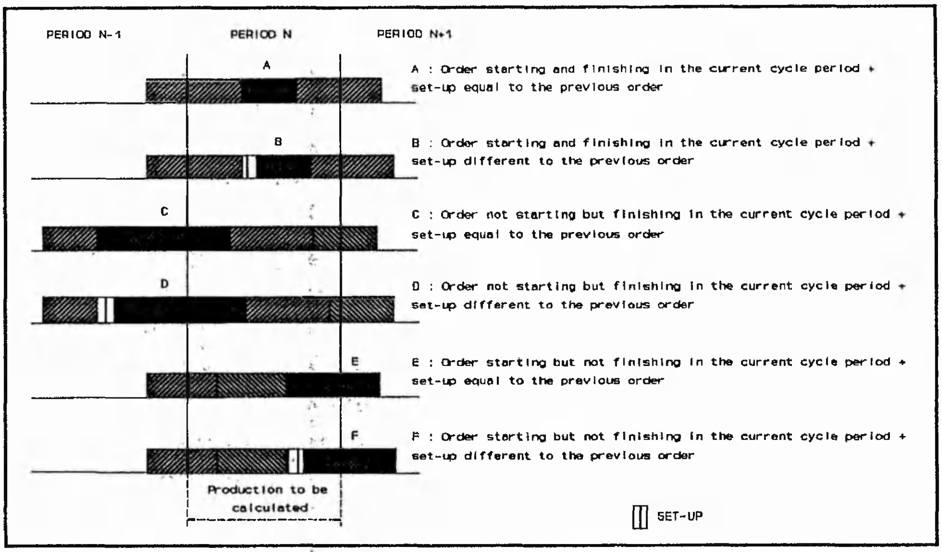


Figure 5.38 Production calculation.

Another characteristic of the production simulation system is the fact that it is based on "batch" transfer policy, by which the manufactured products will be transferred to stock only when the whole scheduled batch has been processed.

The procedure used to carry out this calculations is called "RUNPROD" in the factory sub-program (SYS11.PRG), and can be seen in [Appendix C].

5.3.5 QUOTATION APPROVAL SIMULATION SYSTEM (FINISH OF A DAY)

This procedure is defined to simulate the customers' acceptance to what the marketing department quoted for the enquiries received. The procedure is called "GOODSERV" [Appendix C] and it operates according to the rules defined in the model of "orders' approval" (see section 4.3.3.3).

5.3.6 SHIPPING SIMULATION SYSTEM (FINISH OF A DAY)

The procedure "SHIPPING" [Appendix C] is used to calculate which orders will be delivered to the customers. The manufacturing company to be modelled is a "make to stock" one.

If sufficient stock has been produced at the end of a period to completely cover an order then, that order is shipped. The order is marked as completed and stock levels adjusted.

If stock levels do not exist to completely cover an order no partial orders are shipped, the order will be delayed, and this will then affect the feedback in the game.

5.3.7 FEEDBACK SIMULATION SYSTEM (FINISH OF A DAY)

The feedback procedure "FREQUEN", in the factory sub-program [Appendix C], is the one that will control the feedback process of the gaming-simulation.

Feedback occurs because at the end of each day a calculation of the number of enquiries to enter the game the next day is made. That number is a function of the average price quoted, and due date performance (see "Feedback" in section 4.3.3.4). The day being simulated is then divided up into the appropriate number of periods (variable called "frequency"). An enquire will enter the game every period. The actual time of arrival of each one of the enquiries within its own period is based on a random number generation.

The feedback effect can be seen in figure [5.39] and figure [5.40].

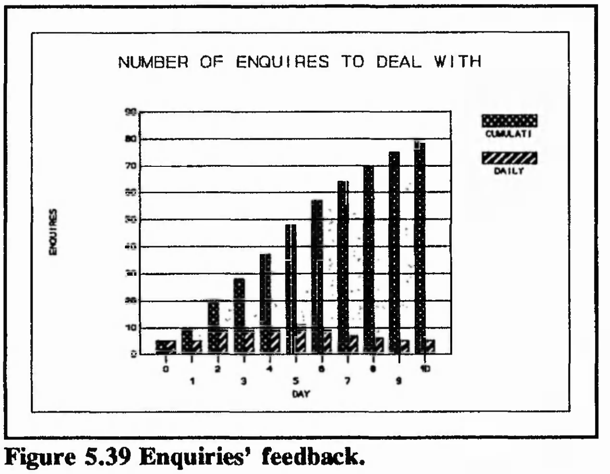


Figure 5.39 Enquiries' feedback.

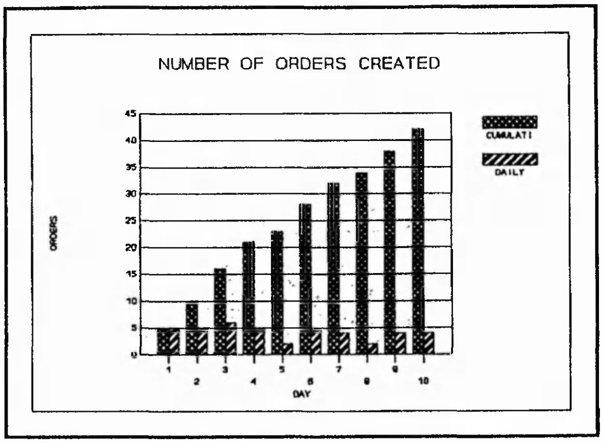


Figure 5.40 Schedule orders' situation.

5.3.8 ENQUIRIES SIMULATION SYSTEM (FINISH OF A DAY)

The procedure that controls the generation of those enquiries is called "ENQGEN" which is within the factory sub-program [Appendix C].

In generating the enquiries there are two elements to be calculated : the delivery date and price. These two elements are calculated by a pseudo-random number which fluctuates between a predetermined low and high limit. Separate limits may be specified for each of the final products.

6 CONCLUSION

In researching the background to this thesis it has become clear that there are few games available that deal with the specific problem of interdepartmental coordination in manufacturing companies, despite its central importance to the success of such companies. Even these games tend to ignore the dynamic aspects of this interaction.

The use of computer based gaming-simulations based on networked personal computers appears to be an appropriate tool for use in education in this areas.

A simplified model has been created which provides the basis for developing such a gaming-simulation and a game has been developed using standard database management system : -FOXBASE+ [78]. This system provided a very convenient means of developing many aspects of the game such as the menus, data input screens, and browsing lists. However it was difficult to use in a number of aspects.

Firstly, it was very difficult to control the progression of time. This resulted in considerable programming effort in this area and even then this slowed the game down to a point where it was difficult to play. Detailed timing tests were carried out[1] and these are shown in Appendix D. From these tests it can be

[1] The evaluation of the actual gaming-simulation was based on the run of several games. Those games involved the set-up of the whole system, and the dynamic input of data (i.e. quotation of enquiries and schedule of orders). Decisions had to be made and input in the system while the gaming-simulation was running. The games run consisted of 10 simulations days each.

seen that, using this approach to controlling time in the game, the game slows down as more orders are loaded in. This is because there is more data to process in each cycle of the time control loop.

Secondly, there were considerable problems in dealing with simultaneous updating of data by different users. The FOXBASE+ software [78] did provide both record locking and file locking facilities but there was considerable programming effort needed to give appropriate responses to users where conflicts were identified.

Finally, the means of data representation provided by this software, i.e. menus, input screens, and browsing lists, provides the user with only a limited view of what the simulation is trying to represent, although it could be argued that this is in many ways an accurate simulation of manufacturing organisations where it is often difficult to see what is actually happening.

Although these problems did in fact mean that the game produced was not successful in the sense that it could not be used in the planned role, they are in a sense superficial and the speed of current developments in software and hardware means that it is only a matter of time before they are overcome. In fact the development of a number of windows based database management systems (including FOXPRO 2.0 [95], the current development of the software used in this research) would make much more sophisticated representation of the data possible than with software that was available at the beginning of the research.

As a consequence this research supports the idea that gaming-simulations can be developed in this area and they are likely to be useful educational tools.

REFERENCES

[1] Hayes, Robert H., Wheelwright, Steven C., and Clark, Kim B.; "Dynamic manufacturing"; New York; 1988.

[2] Advanced Manufacturing Application Product Suite Modular Series Introductory Video. Taken from Edwards, J.; "Educating for Change"; BPICS Control, October/November 1990.

[3] Kidd, P. T.;"Organisation, People and Technology : Towards Continuing Improvement in Manufacturing"; Proceedings of the sixth CIM-Europe Annual Conference; May 1990.

[4] Goldratt, E. M. and Fox, R. E.; "The Theory of Constraints Journal, volume 1-1"; Tonic Enterprises Ltd. London, England; 1987.

[5] Mondy, R. Wayne et al.; "Management and Organizational Behaviour"; Allyn and Bacon, 1990.

[6] Twiss, Brian C. et al.; "Managing Industrial Organizations"; Pitman, 1980.

[7] Managing Change in the 90's - People in Business. Survey conducted in the UK in May 1990. Taken from Churchill, L.; "Beyond Functional Excellence"; Total Quality Management, February 1991.

[8] Galbraith, J.; "Organization Design: An Information Processing View"; Interfaces, vol 4, no 3, pp 28-36, 1974.

[9] Kiyoshi, Suzaki; " The New Manufacturing Challenge - The Technique for Continuous Improvement"; The Free Press, 1987.

[10] Frick, Jan et al.; "Activity chains: A method for identifying and evaluating key areas of integration in SME's"; Proceedings of the Sixth CIM-Europe Annual Conference, May 1990.

[11] Schonberger, Richard J.; "Building a Chain of Customers"; Free Press, April 1991.

[12] Burbidge, J. L.; "The Introduction of Group Technology"; London: Heinemann, 1975.

[13] Burbidge, J. L.; "Group Technology in the Engineering Industry"; Mechanical Engineering Publication, 1979.

[14] Burbidge, J. L.; " Production Control: The future choice"; 5th National Conference of Production Research, 1989.

[15] Voss, C. A. et al.; "Managerial Integration and CIM - a strategic response."; Computer-Integrated Manufacturing Systems, May 1991.

[16] Voss, C. A.; "Organisational change and CAD/CAM effectiveness"; Working Paper MATT-88W-006, Graduate centre for the Management of Technology, University of Cincinatti, USA, June 1988.

[17] Interview to Tony Kelly, Principal Consultant of Hoskyns Group's; "The price of ignorance"; Computerised Manufacturing, June 1989.

[18] Levy, Paul et al.; "Organizational strategy for CIM"; Computer-Integrated Manufacturing Systems, May 1991.

[19] Helms, Marilyn M.; "Communication: The Key to JIT Success"; Production and Inventory Management Journal, 1990.

[20] Kearney, A. T.; "CIM: Competitive Advantage or Technological Dead End?"; Consultant Report, 1989.

[21] Barrar, P. et al.; "Decision Processes in the Design; Implementation and Use of CAPM Systems in Medium-sized Organisations"; Final report to SERC/ACME Directorate Grant GR/E/21278, 1989.

[22] Buckingham, J. et al.; "Beyond Substitution, Organisational Implications for the Successful Use of Integrated Technology"; A Flexible Future? Prospects for Employment and Organisation in the 1990s, Business School Conference, 1989.

[23] Brown, David S.; "Concert Building: Management's New Goal"; Journal of Systems Management, February 1990.

[24] Neave, Henry R.; "Deming's 14 Points for Management"; Salisbury, British Deming Association, 1989.

[25] Neave, Henry R.; "The Deming Dimension"; Tennessee, SPC Press, 1990.

[26] Taylor, Frederick; "Scientific Management: comprising shop management, the principles of scientific management [and] Testimony before the special House committee / With a foreword by Harlow S. Person.", Westport [conn.]: Greenwood Press, 1972.

[27] Lascelles, D. M.; "The key issues of a quality improvement process"; Int. Jnl. Prod. Res., 1990.

[28] Owen, Andrew et al.; "Implementing MRP packages - The lessons learned"; Coopers & Lybrand Associates.

[29] Groves, Brenton R.; "The missing element in MRP - People"; Production and Inventory Management Journal, 1990.

[30] Oliver, Nick et al.; "The Japanization of British Industry"; Basil Blackwell Ltd, 1988.

[31] Manden, Yosuhiro et al.; "Innovations in Management - The Japanese Corporation"; Industrial Engineering and Management Press, 1985.

[32] Burnes, Bernard; "Barriers to employee involvement in technical change: 'more than a case of the good guys and the bad guys"; Adv. Manuf. Eng., April 1990.

[33] Pascale, Richard Tanner et al.; "The Art of Japanese Management - Applications for American Executives"; Simon and Schuster, 1981.

[34] Burbidge, John L. et al.; "Integration in Manufacturing"; Computers in Industry, No 9, 1987 p 297-305.

[35] Burbidge, John L.; "Production Flow Analysis, For Planning Group Technology"; Oxford University Press, 1989.

[36] Goldratt, E. M. and Fox, R. E.; "The Theory of Constraints Journal, volume 1-2"; Tonic Enterprises Ltd. London, England, 1987.

[37] Goldratt, E. M. and Fox, R. E.; "What is this thing called The Theory of Constraints"; Tonic Enterprises Ltd. London, England, 1990.

[38] Vollman, Thomas E.; "Cutting the Gordian Knot of Misguided Performance Measurement"; Industrial Management & Data Systems, 1991.

[39] Waldron, David; "Accounting for manufacture - adequate or archaic?"; Logistics, February 1991.

[40] Helms, Marilyn M.; "Meeting the Human Resource Challenges of JIT through Management Development"; Jnl. Management Development 9, 1990.

[41] Marsden, Alan; "Training for Change"; The Plant Engineer, March/April, 1990.

[42] Heard, Julie A.; "Education: The Key to JIT Success"; Proceedings from the 1985 Conference of American Production & Inventory Control Society.

[43] Tiernan, Tony; "Change is as good as rest...Provided you train for it"; Works Management, August 1989.

[44] Wedberg, George H.; "...But First, Understand The Problem"; Journal of Systems Management, June 1990.

[45] Millard, Robert I.; "Is MRP training aimed in the right direction?"; Production and Inventory Management Journal, 1989.

[46] Batley, T. W.; "Microcomputer Simulation for Teaching Operations Management"; Int. Jnl. of Operations and Production Management, 1991.

[47] Chong, Philip S.; "Using Microcomputer-based MRP Software to Teach Materials Management"; Production and Inventory Management Journal, 1989.

[48] De Lurgio, Stephen A. et al.; "Teaching Integrated Production and Information Control System Principles Using a Spreadsheet Simulator"; Production and Inventory Management Journal, 1989.

[49] Wieters, C. David et al.; "Software Supporting Introductory Courses in Production/Operations Management"; Production and Inventory Management Journal, 1989.

[50] Franklin II, Carter L.; "Are Microcomputers Useful in Management Education?"; Journal of Systems Management, 1990.

[51] Loveluck, Clive; "Notes on the construction, operation and evaluation of Management Games"; Management Games Ltd., 1969.

[52] Neumann John Von et al.;"Theory of Games and Economic Behaviour"; London: Wiley, 1953 [1967 reprint].

[53] Tansey, P. J. et al.; "Simulation and Gaming in Education"; Methuen: London, 1969.

[54] Moreno, J. L.; "Who Shall Survive?"; Boston: Beacon House, 1953.

[55] Greenblat, Cathy Stein; "Designing Games and Simulations - An Illustrated Handbook"; SAGE Publications, 1988.

[56] Brand, C. et al.; "The Simulation of 'real-life' organizations within a game"; Perspectives on Academic Gaming & Simulation 6, 1981.

[57] Burgess, Tom; "Management Simulation-Games and Artificial Intelligence"; Simulation\Games for Learning 20-3, 1990.

[58] Emmons, Hamilton et al.; "STORM: Quantitative Modelling for Decision Support"; Holden-Day, Oakland, CA 1986.

[59] Hall, Owen P.; "Computer Models for Operations Management"; Addison-Wesley, 1989.

[60] "TEAMSKILL - Production Management Exercises"; Hall Marketing, 1978.

[61] "TOPAZ"; Edit 515 Ltd.

[62] "EXECUTIVE"; April Computing Executive Ltd.

[63] Sykes, Philip; "BISSIM - Computerised Business Simulation Exercises"; Perspectives on Gaming & Simulation 9, 1984.

[64] Ahituv, N. et al.; "Principles of Information Systems for Management"; Dubuque, IA: William C. Brown, 1983.

[65] Johnston, Robert; "Complex Interactive Computer-Based Simulations Without Mathematical Modelling"; Simulation & Games 18-4; 1987.

[66] Rohn, Walter E.; "The Present State and Future Trends in Management Games For Management Development in Germany"; Simulations & Games, 1986.

[67] Gray, Carey et al.; "The Development of a 'HOT' Business Management Game"; Perspectives on Gaming and Simulation 14, 1989.

[68] Pruett, James M. et al; "MOSES: Manufacturing Organization Simulation and Evaluation System"; Simulation, January 1990.

[69] Bryant, Nigel et al.; "Changing Technology and Management Games"; Perspectives on Gaming and Simulation 14, 1989.

[70] Loveluck, Clive; "Some Conceptual Changes in Micro-Computer Based Games"; Simulation/Games for Learning 19-4, 1989.

[71] Keys, Bernard; "Total Enterprise Business Games - An Evaluation"; The Guide to Simulations/Games for Education and Training, 1980.

[72] Keys, Bernard; "Total Enterprise Business Games"; Simulation and Games 18-2, June 1987.

[73] Partridge, S. E. et al.; "Designing Management Games for Production Management Training"; Simulation and Games 5-3, September 1984.

[74] Gibbs, G. I.; "Handbook of Games and Simulation Exercises"; E. & F. N. SPON Ltd, 1974.

[75] Dillman, Duane; "The Design and Development of Simulation Exercises"; Paper presented at the meetings of the American Educational Research Association, Minneapolis, MN, March 1970.

[76] Kibbee, J. M. et al.; "Management Games"; Reinhold, New York, 1961.

[77] Loveluck, Clive; "The Function of TIME in Management Games and Simulations"; Simulation/Games for learning 20-1, March 1990.

[78] Goley, George F.;"Dynamic of FOXBASE PLUS Programming"; Dow Jones-Irwin, US, 1988.

[79] Crookall, J. R.; "Computer Integration of Advanced Manufacture"; Proceedings of Institution of Mechanical Engineers, Vol. 200, No. B4, 1986.

[80] Goldratt, E. M. and Fox, R. E.; "The Theory of Constraints Journal, volume 1-6"; Tonic Enterprises Ltd. London, England; 1987.

[81] Bruns, William J. et al.; "Accounting & Management, Field Study Perspectives"; Edited by William J. Bruns, Jr. Robert S. Kaplan, 1987.

[82] Kaplan, Robert S.; "Advanced Management Accounting"; London: Prentice Hall, 1988.

[83] Lantz, Kenneth E.; "The Prototyping Methodology", Prentice Hall, Englewood Cliffs: Prentice-Hall, 1986.

[84] Jamsa, Kris; "DOS: The Complete Reference"; Berkeley, Calif.: Osborne McGraw-Hill, 1987.

[85] Pegels, C. Carl; "Integrating Functional Areas for Improved Productivity and Quality"; Int. Jnl. of Operations & Production Management 11-2, 1991.

[86] Tuttle, H. C.; "Breaking the 'Wall' between Design and Manufacturing"; Production, May 1983, pp. 63-6.

[87] Reitzle, W.; "Confrontation or cooperation?"; The FMS Magazine 7, 1989.

[88] Shapiro, B. P.; "Can Marketing and Manufacturing Co-exist?"; Harvard Business Review, September-October 1977, pp. 104-14.

[89] Powers, Thomas L.et al.; "Marketing and Manufacturing Conflict: Sources and Resolution"; Production and Inventory Management Journal, 1988.

[90] "The Hard Sell"; Manufacturing Engineer, March 1991.

[91] Kotler, Philip; "Marketing Management"; Prentice-Hall, Englewood Cliffs, NJ 1977, p.12.

[92] Bechtold, Ronald K.; "The Available-to-Promise Tool"; APICS Annual Conference Proceedings 1981, pp. 386-389.

[93] Plowman, Brian; "Management Behaviour"; Total Quality Management, August 1990.

[94] Coote, Alan et al.; "Some Human and Machine Aspects of Computerized Simulations"; Perspectives on Gaming & Simulation 10, 1985.

[95] "FOXPRO 2.0"; Fox Software 1991.

[96] "NOVELL", Novell Incorporated, 1986.

[97] Tranfield, G. and Igartua, J. I.;"An Interactive gaming-simulation of manufacturing organisations"; Advances in Manufacturing Technology VI. Proceedings of the seventh National Conference on Production Research, 1991.

APPENDIX A

DEMING'S THOUGHTS ABOUT CO-OPERATION.

MANUFACTURING CONFLICTS

DEMING'S THOUGHTS ABOUT CO-OPERATION

The theme of genuine co-operation is compactly summarised in one of the vertices of the Joiner Triangle : "All One Team". This is a foundation-stone of the Deming philosophy. Yet again he becomes aware of the barrier which the merit rating forms to real progress. The worst case is performance appraisal constrained by a fixed distribution where it becomes necessary to put somebody else under in order to get a higher rating. Surely the better way is to have everybody working for the company rather than against each other or against other departments in the company. Departments in the company are so often managed in a competitive framework. Yet every department is a supplier or a customer, or both, to every other department. In a inner competitive environment, what matters the trouble which you cause to the other departments ? Indeed, it can then actually be *advantageous* for you to cause trouble to other departments (or at least as much as you can get away with).

Deming's Example : Huge financial advantages of co-operation

Harm that is caused by internal competition and conflict and the fear which is thereby generated, and good that is brought about by internal co-operation and teamwork, is of massive proportions. A Purchasing Manager, under pressure to reduce his figures, changes to a cheaper source, even if he buys poorer products and service as the result. Engineering design imposes unnecessarily tight tolerances to compensate for the fact that manufacturing never reaches the standards asked of it. Departments performing better than budget start spending near the end of the year because they know that otherwise their next year's budget will be reduced. As the end of the month looms, salesmen start doing everything they can to meet their quotas, which scant regard to the problems caused to manufacturing, administration and delivery, let alone to the customer. Figures get massaged, computations "redefined", so that reports show more of what senior management want to see. The CYA (cover your posterior) syndrome holds sway.

The following simple illustration indicates clearly something of what is won or lost in environments of co-operation and conflict respectively. It is based on a example that Deming used in his 1988's seminars, and that Henry R. Neave [24] modified.

For brevity and ease, the illustration is in a small scale. It is not difficult to imagine how the numbers multiply in more realistic examples.

	EFFECTS			
Areas and their options	on Area A	on Area B	on Area C	
AREA A				
AREA B				
AREA C				

Figure A1

The blank chart shows that the illustration concerns and organisation comprised of just three areas or departments. In the left-hand column will be listed options available for each area to adopt or not adopt according to their choice. The remaining columns will show the potential effects of adoption of the options.

	EFFECTS			
Areas and their options	on Area A	on Area B	on Area C	
AREA A				
(i)	+			
(ii)	+			
(iii)	+			
AREA B				
(i)		+		
(ii)		+		
AREA C				
(i)			+	
(ii)			+	
(iii)			+	

Figure A2

When there are barriers between the areas (old style management, MBO, inner competition, etc.), each area naturally adopts options which are beneficial to itself.

Areas and their options	EFFECTS on Area A	on Area B	on Area C	on the company
AREA A				
(i)	+			
(ii)	+			
(iii)	+			
AREA B				
(i)		+		
(ii)		+		
AREA C				
(i)			+	
(ii)			+	
(iii)			+	
Net effect of adopted options				

Figure A3

However, let us examine the effect of this primitive decision-making mechanism both on the individual areas and hence (by summation) on the whole company. For ease of illustration, let us assume that each + or - corresponds to gain or loss of the same amount of money throughout.

Areas and their options	EFFECTS on Area A	on Area B	on Area C	on the company
AREA A				
(i)	+	−	−	−
(ii)	+	−	+	+
(iii)	+	−	−	−
AREA B				
(i)	−	+	−	−
(ii)	+	+	−	+
AREA C				
(i)	+	+	+	+ + +
(ii)	−	−	+	−
(iii)	−	−	+	−
Net effect of adopted options	+ +	− −	0	0

Figure A4

Options locally beneficial to one area may well be quite the opposite for other areas. Suppose each are's gains or losses are as indicated. The net effect on the company happens to be zero in this case, i.e. equal to the effect of doing nothing at all (which would be a rather easier way of achieving the same result!). Depending on the details, the net effect could have been positive, zero or negative.

Areas and their options	EFFECTS on Area A	on Area B	on Area C	on the company
AREA A				
(i)	+	−	−	−
(ii)	+	−	+	+
(iii)	+	−	−	−
AREA B				
(i)	−	+	−	−
(ii)	+	+	−	+
AREA C				
(i)	+	+	+	+ + +
(ii)	−	−	+	−
(iii)	−	−	+	−
Net effect of adopted options	+ + +	+	+ + +	+ + + + +

Figure A5

If the management environment improves so that areas become aware of the effect of their actions on other areas (they may have been aware already, of course, but it was not in their interests to pay any attention) and the old inter-departmental rivalries are placed by genuine teamwork for mutual and company benefit, options are now only adopted if they produce net benefit to the company. Consequently, only three of the previous eight options are now adopted. (Entries relevant to adopted options are characterised in the Figures by bold print; the options now rejected have been shaded over.) By this more judicious and restricted choice of action (i.e. wise choices of both actions and inactions), everybody gains.

Areas and their options	EFFECTS on Area A	on Area B	on Area C	on the company
AREA A				
(i)	+	–	–	–
(ii)	+	–	+	+
(iii)	+	–	–	–
(iv)	–	+	+	+
(v)	–	+	+	+
(vi)	–	–	+	–
AREA B				
(i)	–	+	–	–
(ii)	+	+	–	+
(iii)	+	–	+	+
(iv)	+	–	+	+
AREA C				
(i)	+	+	+	+ + +
(ii)	–	–	+	–
(iii)	–	–	+	–
(iv)	+	+	–	+
(v)	+	–	–	–
Net effect of adopted options	+ + + +	+ +	+ + + +	+ + + + + + + + + +

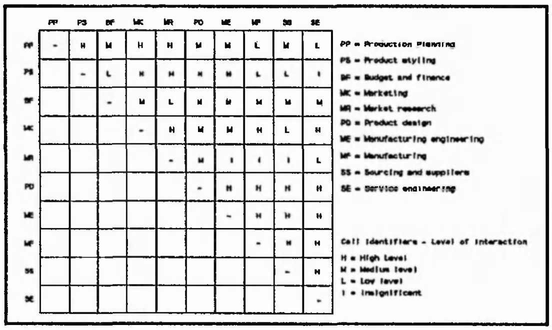
Figure A6

Further, in the improved environment, options which previously never saw the light of day are now considered. These are options that are locally disadvantageous to the area which can adopt them but which give benefit to other areas. Amongst this greater range of options, again the ones to be adopted or not adopted are now chosen according respectively to wether they are or are not of net benefit to the whole company. The bottom line results speak for themselves.

INTEGRATING FUNCTIONAL AREAS

Based on the article "Integrating Functional Areas for Improved Productivity and Quality" [85].

Figure A7 shows the existing degree of coordination between the different departments of a factory.



	PP	PS	BF	MK	MR	PD	ME	MF	SS	SE
PP	-	H	M	H	H	M	M	L	M	L
PS		-	L	H	H	H	H	L	L	I
BF			-	M	L	H	H	M	M	M
MK				-	H	M	M	H	L	H
MR					-	M	I	I	I	L
PD						-	H	H	H	H
ME							-	H	H	H
MF								-	H	H
SS									-	H
SE										-

PP = Production Planning
PS = Product styling
BF = Budget and finance
MK = Marketing
MR = Market research
PD = Product design
ME = Manufacturing engineering
MF = Manufacturing
SS = Sourcing and suppliers
SE = Service engineering

Cell identifiers - Level of interaction
H = High Level
M = Medium level
L = Low level
I = Insignificant

Figure A7

- PP = Production Planning
- PS = Production Styling
- PE = Production Engineering
- ME = Manufacturing Engineering
- MK = Marketing
- SE = Service Engineering
- PD = Product design
- MF = Manufacturing
- SS = Sourcing and Suppliers
- MR = Market Research

DESIGN AND PRODUCTION CONFLICT

In recent years the problem of conflict or separation between the product design and product engineering function, on the one hand, and manufacturing and manufacturing-process engineering, on the other hand, has become an issue of serious concern. Tuttle [86] refers to this conflict as the "wall" between product design engineering and manufacturing engineering.

There is, unfortunately, a fundamental conflict of objectives, which becomes clear when the following is taken into account [87] :

- Every designer would like to achieve a functionally superior and therefore often complex product.
- The production engineer will take fewer risks in mass production runs by using easily controllable materials and processes, so he wants to manufacture products from components which are not too numerous and complicated and which involve the simplest possible assembly processes. At the forefront of the designer's considerations is the comprehensive functional performance of the total system and all its components.

The avoidance or minimisation of the negative aspects of this dualism will lead to new solutions through cooperation alongside the introduction of modern information processing technology (CIM) and other organisational measures such as communications engineers forcing through the new "watchword" of integration.

MARKETING AND MANUFACTURING CONFLICT

Shapiro [88] was one of the early authors to question if marketing and manufacturing could co-exist. He cited situations where lack of co-operation between the two functions could lead to open warfare. It is often seen in the form of incorrect forecasting and inventory planning and a general atmosphere of mistrust that results in less than optimal performance of the company as a whole. Also other articles refer to the same problem [89,90].

Especially dangerous is the situation where either one or the other becomes dominant. If marketing dominates, the firm can become so sales-minded that manufacturing can not operate effectively. Alternatively, the firm can become so manufacturing-oriented that the needs of the customers are forgotten in the name of smooth operations. Top management must be alert to avoid conflict developing.

In order to understand sources of this conflict, motivations of the marketing and manufacturing functions must be understood. Kotler [91] has described marketing as "the analyzing, organizing, planning and controlling of the firm's customer-impinging resources, policies, and activities with a view to satisfying the needs and wants of chosen customer groups at a profit." Thus marketing is primarily concerned with forces outside of the firm. Manufacturing's domain, however, is concerned with a different set of goals, namely the on-time production of required products, the meeting of cost objectives, and maintenance of expected quality standards and targeted customer services levels [92].

Figure A8 summarises the sources of conflict between marketing and manufacturing.

Source	Example
Role orientation	Background of individual Performance evaluation criteria
Information between firm and environment	Customer demand as seen by marketing
Information within firm	Forecasts from marketing to production Build schedules from production to marketing
Product flow to environment	Loading or shortages in distribution channel
Product flow within firm	Material shortages Expedites or cancellations

Figure A8

APPENDIX B

WORKSHOP OF MANAGEMENT SIMULATIONS

Computers in Teaching Initiative (CTI)

The Management Centre, University of Bradford

WORKSHOP OF MANAGEMENT SIMULATIONS

INTRODUCTION

This workshop of Management Simulations was held on Wednesday the 6th of February 1991 at the Management Centre, University of Bradford.

The attendance to this one day workshop was possible due to the invitation received from Gillian Holmes, at the Management Centre, who is the CTI Information Officer.

The Computers into Teaching in Teaching Initiative (CTI) aims to promote a greater awareness of the potential of information technology (IT) to enhance the teaching and learning process, and to facilitate the development of skills in the use of computers in the teaching process.

Twenty one discipline-based centres have been established with the aim of assisting teachers to identify, acquire and use appropriate IT resources, course-ware and software.

The Centre for Accountancy has, in 1990, set up five regional centres, each with specialist interests. The northern regional centre for Business and Management studies is based at the Management Centre, University of Bradford, and will seek to :

- Collect and disseminate information to teachers throughout the northern region, on the use of computers in education, and materials available.

- Provide a base for the coordination of activity and information exchange among IT users in the region, including the organisation of demonstrations and workshops.

- Liaise with institutions and IT users in the region to establish a database of available software/course-ware and to publish evaluations of software/course-ware.

WORKSHOP

The workshop was based on a display of four different Management Simulations (Simulation/Games). These simulations were Network Proteus, Topaz, Executive and Bissim, which were considered by the organisers as four of the best simulation/games available now at the market place.

The number of people who attended the workshop was around twenty, their most common position was lecturer or senior lecturer in business, at the Bradford Management Centre, and Polytechnic and Colleges around Bradford.

NETWORK PROTEUS

Forewords

The business simulation "PROTEUS" was developed at Manchester Business School about six years ago (1985). That first release was a mainframe version sponsored by IBM, because its interest in developing new software applications for its existing hardware.

The piece of software written turned to be non transportable. Based on that disadvantage and after six years of running experience at the school, they start a new project called *Project Network Proteus*.

The project Network Proteus (founded by the Training Agency, under the direction of Dr Syel Howell) was started at the very beginning of 1991, and is expected to be finished by January 1992. The objective of the project is to improve and transfer the mainframe version of Proteus to a rewritten network version. That network version is due to be available by January 1992. The network needed is a local area network of PCs with 640K of RAM.

The main architect for this project is Pat White from the Manchester Business School, and formerly IBM employee. He was the one who presented the Network Proteus at the Bradford workshop.

Technicalities

Network Proteus was described as a qualitative simulation rather than a quantitative one. The simulation is not numerically oriented.

That characteristic made the designers decided that the programming language to be used had to be more data oriented, rather than numerical. That fact together with the need to work in a multi-user environment, meant that the language program to be used had to be a Database Management System (DBMS).

Among the different DBMS software, they decided to use PARADOX from Borland. They select this software because it was thought to be most user friendly system.

The hardware requirements were a network of PCs (hard disk and 640K RAM) could be used. The PCs had to be connected to a file server, with at least four Megabytes per playing group (data files). The program was meant to run under PARADOX supported by a spreadsheet for calculations. The expected sale price is £1,000.

Context

The simulation is based on a rainwear factory, manufacturing umbrellas. Each team (factory) is formed by five people (positions) that have a different role in the business. These five roles are :

- Marketing
- Production
- Purchasing
- Finance
- Manager Director

Although there can be different teams (factories) working at the same time, it is not suppose to be a competition game, where teams compete against each other.

The simulation was described as a total business game capable of making everybody know everything. It is thought to be a game where people get involved all over the factory, seeking for collaboration inside the team.

Apart from the teams and their members, there is also another key element in

Network Proteus, the **Tutor**.

The Tutor has two main tasks :

- Generator of the external world.
- Analyzer of the teams' results.

The Tutor generates the external world by sending different messages to the players. Those notes are related to actual running problems of the factory, and concern areas like quality, stock levels, etc. Apart from standard messages (contained in a library), the Tutor can create and add his own ones. The Tutor can also decide who is he sending the notes to, giving him the option to mess up well performing teams if wanted.

Further developments could include a expert system to generate the Tutor's notes.

Simulation Description

The simulation is based on teams of five people performing five different factory's roles. Each of the roles is simulated in one computer of the network and therefore, the roles are interconnected.

The interaction between roles come from two possible sources :

- Access to different roles' data.
- MAIL system.

The MAIL system is based on a directory of names and addresses of the people may be contacted, to ask, complain, inform,.., about a certain issue related to the run of the factory. Thus, the interaction works by sending and answering to each role's mail.

Although there are five roles in the game, only two of them were described, Production and Marketing.

Production Management System (Production Director)

The player/s dealing with this role have to make decisions concerning the production area. Their task is to define the production plan and the operations scheduling.

The production plan is based on sales expectations given by the Marketing function. After defining the plan, the parts requirements will be calculated based on the existing BOM (Bill of Materials). This calculation is carried out by a MRP system, not very accurate nor complicated but convenient for the purposes of the simulation. At this stage is possible to use the existing spreadsheet.

This parts requirement will give information about the need of material, as well as resources loads. Information, that may make lead to amend the production plan.

The production director will also be able to introduce new manufacturing products in the manufacturing process. It will take three simulation months for those new products to enter the manufacturing process. In the mean time, he will also have to deal with all the internal problems (MAIL) as well as with the external ones (TUTOR'S notes).

Marketing

Due to the nature of the business being simulated, rainwear company, the role of the Marketing function in this factory is very characteristic. The marketing demand is of a seasonal nature.

The marketing people will have to decide about the expecting sales, forecasts (long and short range), competition, prices, advertising, discount rates,...To support these decisions the Marketing people can access to information about the weather, list of customers and their acquisitions, etc.

Players on this role will also have to decide about the sales strategy, deciding whether they are selling through agents or their own sale force.

Functioning Characteristics

The old version of PROTEUS was used for MBA, 2nd year Managers and Postgraduates students. The new version (Network Proteus) targets the same population.

Each run cycle is one simulation month (i.e. four week period time). The time needed to run six cycles (six simulation months) is three real full-time working weeks.

PAT WHITE'S EXPLANATIONS

UNIVERSITY OF BRADFORD

MANAGEMENT CENTRE EMM LANE BRADFORD WEST YORKSHIRE BD9 4JL UK
☎ (0274) 542299
Telex 51309 UNIBFD G
Fax (0274) 546866
E-mail ghctiaUK.AC,BRADFORD

Director and Professor of Management DAVID T H WEIR MA DipPSA FBIM

COMPUTERS IN TEACHING INITIATIVE CENTRE FOR BUSINESS AND MANAGEMENT STUDIES
Coordinator: RICHARD WELFORD BA MA PhD PGCE ABIM

14 February 1991

Mr J I Igartua
Nottingham Polytechnic

Dear Mr Igartua

I hope you enjoyed the workshop on management simulations last week, despite the cold weather! I enclose some information on Network Proteus which Pat White has sent for distribution to you. Pat has undertaken to feed information on future developments to me for wider distribution - so watch this space! If you have any further queries on the other simulations we saw, please don't hesitate to get in touch with me, or directly with the suppliers.

Yours sincerely

GEHolmes

Gillian Holmes

Sponsored by the Institute of Chartered Accountants in England and Wales
and the Computer Board for Universities and Research Councils

The Network Proteus Project

In 1985 the IBM company awarded a very substantial grant (of hardware, software and money for the employment of development staff) to the Manchester Business School. In addition two senior IBM staff were seconded to the School to support the project for its three year duration.

A key element in MBS gaining the award in competition with many other business schools was a proposal to build a management simulation of a new kind for use in the MBA programme. Like most computer projects which break new ground, the simulation project, called Proteus after the god who could assume many forms, identified a number of problems. Even so, it was clear that the novel features of the simulation added considerable value, and it is still in regular use at the IBM project partner sites, the business schools at Aston, Strathclyde and Warwick Universities.

The Proteus project was designed to make full use of the facilities provided by the IBM award, specifically the 4381 mainframe, the VM operating system, and the PROFS, AS and SQL software packages. While this contributed to the richness of the simulation, it has been a major obstacle to its wider use.

In the summer of 1989, MBS responded to an initiative by the Training Agency of the Department of Employment by submitting a proposal for rebuilding and extending the original Proteus simulation in a new, more widely accessible environment based on a network of personal computers. This proposal was accepted and a contract for a two year project, christened Network Proteus to reflect its predecessor and the new environment, was signed in January 1990. The contract is between the Training agency and the University of Manchester, is based at MBS, and has the active support of the original partner sites. The training branch of the National Health Service is also showing an interest which is also likely to lead to collaboration.

Network Proteus, like the mainframe-based development, contains features absent from the vast majority of management simulations currently in use. When playing a typical business game, the participants are presented with a limited selection of quantitative data for a hypothetical company, balance sheet, profit and loss account, sales history, etc., and possibly additional figures about the size of the market and the competition. The players, usually operating in teams, make a standard set of decisions about allocation of resources, which are fed into the model to generate a new set of data for the next operating period, the objective being to finish with a set of figures superior to the other teams. If the teams are competing with each other in the same market, the simulation takes on all the attributes of a zero-sum game. If they are not carefully controlled, these simulations can degenerate into an attempt to beat the algorithm.

Both the earlier and the current Proteus projects provide for larger and more realistic flows of numeric data, but the range and richness of the simulation is also substantially extended by the introduction of qualitative information in the form of files of memos, letters and commercial intelligence clippings which must receive attention. For example, in one of the Proteus models being developed, the simulation of a rainwear manufacturing company, the management team is faced with

a letter complaining about quality from the chief buyer of a major customer

an insurance claim for damage caused by one of the company's delivery vehicles

persistent bad timekeeping by the best worker in the general office

queues around the recently introduced vending machines

an adverse movement in the exchange rate with a country from which several products are imported

In addition they have to prepare for a forthcoming ministerial visit. Failure to cope with these issues can have unfortunate consequences. If, for example, the complaint about quality is ignored, the business will be lost leaving the team with a mountain of unsaleable stock. While coping with these qualitative problems, they still have to manage the numbers. In contrast to existing games, no decision need be taken. The team doesn't have to order raw materials, nor change production schedules. The results of not doing so may leave a lot to be desired!

A second key feature of Network Proteus is the access to information. In a conventional game the entire team is usually in possession of all the data available. In a Network Proteus model, each team member can see the data relevant to his or her own role. Thus, the marketing director can check on advertising expenditure but has no access to detailed production schedules. If he needs data from them, he must enlist the aid of his production director who, as in real life, has his own problems to contend with. Also if team members call for the same information they may get what appear to be inconsistent answers. A request for last month's sales figures from the marketing director will probably produce totals of orders taken. The same request from the production director would be satisfied by shipment figures, while the finance director could well get goods invoiced less returns credited.

Reference was made earlier to the volume of numeric data in a Network Proteus model. The project attempts to copy the real world by providing substantial amounts of detail, supported by tools for interrogation and data reduction. As an example, in the rainwear model, the players have access to a file containing all the orders received in the last two years of simulated operations, upwards of 1,000 entries. Information about the level of customer returns is part of this file. Accurate analysis of the data in the file will show that the returns rate for the complaining customer is an order of magnitude greater for the product which is the subject of the complaint than for any other customer/product combination. In this way the project should achieve a secondary objective, introducing students to some of the tools and techniques available through information technology to support management decisions.

The earlier Proteus project did not fully justify its name, since it provided only one scenario, usable in two ways. From the outset Network Proteus has been designed to assume many forms. To achieve this, a simple modelling language has been defined and a shell to interpret it has been written. The language will allow us to simulate a wide variety of business situations. These may be team exercises or case studies for analysis by individuals. The project plan has allocated resources for the production of two major team based simulations. One is the rainwear manufacturing example referred to above. The other, in complete contrast, will simulate the running of an NHS hospital under the recently introduced Resource Management Initiative. At present, the RMI is being piloted in a small number of hospitals, one of which is associated with the project. A number of individual case studies will also be produced. The first of these, already under development, requires the student to analyse the inventory problems of a personal computer dealer.

Network Proteus models will, wherever possible, employ commercially available PC software. The two key areas will be a data base package which provides flexible and easy to use storage, retrieval and manipulation facilities, and a spreadsheet to reduce the effort involved in handling numeric data. Another possibility being examined is the use of an expert system to assist in the control of the more complex models by the tutor.

The project plan calls for the delivered product to include

- the model definition language and the shell to interpret it
- the two major simulations described above
- a number of supporting case studies
- PC based tutorials to introduce the Network Proteus environment and the commercial packages used
- tools to enable Proteus users to develop their own models and case studies

In addition, although there is no contractual obligation to do so, we hope to include data capture facilities so that Network Proteus can be used as a tool for research into the learning process and the behaviour of teams using computer networks. The anticipated completion data is December 1991.

The project is directed for the University by Dr S D Howell, Lecturer in Management Accounting and Control at MBS. Computer development is managed by Mr P L White, one of the secondees from IBM to the earlier project, since retired from IBM and appointed to an honorary fellowship at MBS.

P L White
5.2.91

TOPAZ

Forewords

Topaz is one of a range of management games produced by EDIT 515 Ltd. Topaz simulates major company functions enabling between 12 and 48 players divided in teams which compete against each other. An elaboration of TOPAZ is the base for "The Scotsman Management Game" run by "The Scotsman" newspaper.

EDIT 515 Ltd. has been developing games for 20 years, being a very experienced company dealing with this sort of games.

Technicalities

TOPAZ is a game developed in FORTRAN, running in a stand alone machine. Its price is £2500 with an academic discount of 15%.

Game's description

TOPAZ is a management game where 3 or 8 teams compete against each other. The teams are formed by 3 to 6 people, each of them concerned with a different role. The game has to be played for a minimum of 5 plays and a maximum of 10.

The areas within the game are : Marketing, Finance, Personnel, Production Planning, R&D, and Purchasing, although the speaker, Bill Robertson, insisted in the importance of the existing conflict between Marketing and Production Planning. He also insisted in the general interaction that is encouraged through the run of the game.

After a discussion of one hour, each team have to take some decisions concerning different areas of the factory. Those decisions are filled in the input form and will afterwards input in the computer. The computer after computing those inputs (30 to 45 minutes), will display some numeric results.

Game's elements :

Product Description

The factory that is being gamed has three different final products with an unknown bill of materials (BOM). They are three black boxes that could mean anything.

Production Process

The production process of the factory being gamed is based on an assembly shop an a machine shop. The assembly shop is where the product is assembled and the machine shop is where the parts for that assembly process are machined.

As there is no BOM (Bill of Materials), only final products, the times considered by the game for machining and assemble are some fixed predetermined ones. The level of quality of the final products is directly proportional to the assembly time spent.

The production process defined for the game, assumes that there is no WIP (Work in Progress), that means that if there are not any final products to be produced, there will not be any parts being produced either.

About the shifts, the machine shop can be run with 1,2, or 3 shifts. On the other the assembly shop is only run with one shift, with a limited number of men and overtime available.

The game, also, allows the possibility to by machines, that will only be possible to be bought if the financial state of the factory is satisfactory. The machine will start working at a deferred time.

Marketing

The decisions to be taken in the Marketing area are related to sale price, credits given, design (fashion, technical advances,..), quality (warranties), deliveries, competitiveness,..

They also will have to decide about advertising matters, having to choose from three different kinds of advertising.

As it was explained in the introduction, the game is designed to run under the conflict between Production and Marketing, and therefore they will have to reach an agreement by adjusting their different objectives. The other departments role will be to do what comes from the adjustments agreed between Marketing and Production Planning.

Research and Development (R&D)

There is a possibility to introduce product improvements but of coarse at a certain expense. That expense is a cumulative and therefore, does not always mean that because you spend money you are going to get the improvement. The research and development can turn out to be none, minor, or major.

That improvement status is controlled by random numbers :

IF random number > cumulative money spent THEN improvement (minor or major)

Personnel

Their task is to keep people working in the factory happily, trying to avoid negative actions that could influence the work of the factory.

The happiness of the workforce and the factors that influence it are graphically shown in figure B1.

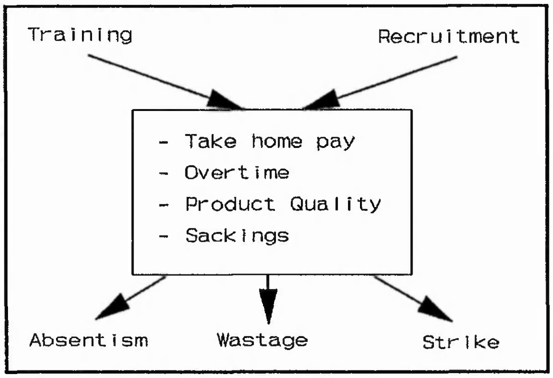


Figure B1

Finance

The factory being gamed also has a finance department, where decisions about the finance state of the factory are discussed.

Objective of the game

The objective of running the game could be explained as :

"To get, from among the different teams playing, the highest share price at the end of the game."

That share price value depends basically on the Net Assets per share, although it is also affected by the following long-term parameters :

- Dividend Performance.
- Liquidity.
- Current Market Share.
- Production Resources.
- Research and Development Effort

♦ Stock and Backlog position.

All these factors imply that the success of the gamed factory will be achieved by balancing the customer service performance and the factory's profitability (Sales Revenue, Cost, Overhead, Miscellaneous).

Game's cycle

The game's cycle is described in figure B2.

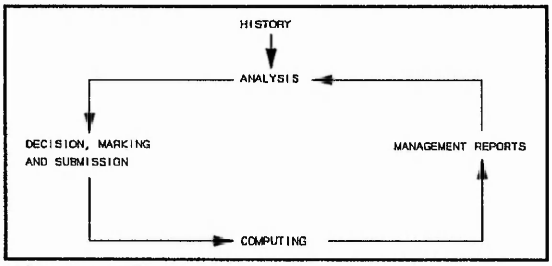


Figure B2

The forms used for playing TOPAZ are shown in the coming pages.

TOPAZ FORMS

GAME HISTORY
GAME HISTORY
GAME HISTORY
GAME HISTORY
GAME HISTORY

'THE SCOTSMAN' MANAGEMENT GAME - 1989 (Round One)

CREATED AND ADMINISTERED BY EDIT 515 Ltd.

THIS IS THE FIRST PART OF THE GAME HISTORY FOR ROUND ONE.

YEAR 1988, QTR. 1

GROUP 1 | COMPANY 1 | IDENTITY No 1 | YEAR 1988 | QUARTER 1

REPORT BASED ON DECISIONS BELOW, CHECK THEM BEFORE GOING FURTHER.

	PRODUCT 1	PRODUCT 2	PRODUCT 3
MAJOR IMPROVEMENTS TAKEN UP	0	0	1
PRICES EXPORT AREA	118	195	390
PRICES HOME AREA	108	183	378
PROMOTION IN TRADE PRESS	6	6	10
ADV SUPPORT	8	8	15
MERCHANDISING	4	4	5
FITTING TIMES (MINS)	115	170	330

SALESMEN ALLOCATED TO EXPORT 13 S 2 W 1 N 4

SALESMAN'S SALARIES	35	COMMISSION	1
SKILLED WAGE RATE	440	SHIFT LEVEL	2
MANAGEMENT BUDGET	75		
MAINTENANCE HOURS	60	MACHINES SOLD	0
DIVIDEND	4	CREDIT PERIOD	30
VANS BOUGHT	0	VANS SOLD	0
INFORMATION ON COMPANIES	1	MARKET SHARES	1

DELIVERIES REQUESTED TO (NOT DELIVERED IN FULL IF STARRED)

	PRODUCT 1	PRODUCT 2	PRODUCT 3
EXPORT	4250	2200	800
SOUTH	300	250	100
WEST	150	120	85
NORTH	700	475	180
RESEARCH EXPENDITURE	10	12	10

SALESMAN HIRED	1	DISMISSED	0	TRAINED	0
SKILLED MEN HIRED	0	DISMISSED	0	TRAINED	0

MATERIALS ORDERED	18000		
FROM SUPPLIER	2	DELIVERIES	8
MACHINES ORDERED	0		

PLANT/MATERIALS/PERSONNEL

MACHINES AVAILABLE LAST QUARTER	13
MACHINES AVAILABLE NEXT QUARTER	13
VANS AVAILABLE LAST QUARTER	21
SKILLED MAN HOURS:	
AVAILABLE LAST QUARTER	27048
ABSENTEEISM	108
USED LAST QUARTER	26169
NOTICE OF STRIKE WEEKS NEXT QUARTER	0
MACHINE HOURS:	
AVAILABLE LAST QUARTER	14196
BREAKDOWN TIME	39
MAINTENANCE	741
USED LAST QUARTER	12205
MACHINE EFFICIENCY (%)	97.5
MATERIAL UNITS.	
OPENING STOCK	1844
DELIVERED	15000
USED	15448
CLOSING STOCK	1396
ON ORDER FOR NEXT QUARTER	18000
AVAILABLE NEXT QUARTER	19396

PERSONNEL	SALESMEN	SKILLED MEN	UNSKILLED MEN
AT START OF QTR	20	46	98
RECRUITED	1	0	6
DISMISSED	0	0	0
TRAINED	0	0	
LEFT	0	0	6
AVAIL FOR NEXT QTR	21	46	98

PRODUCTS/ORDERS/STOCKS

PRODUCT QTS	PRODUCT 1	PRODUCT 2	PRODUCT 3
REQUESTED	5400	3045	1165
PRODUCED	5564	3139	1202
REJECTS	164	94	37
SERVICED	80	50	17
DELIVERED TO	PRODUCT 1	PRODUCT 2	PRODUCT 3
EXPORT	4250	2200	800
SOUTH	300	250	100
WEST	150	120	85
NORTH	700	475	180
ORDERS FROM	PRODUCT 1	PRODUCT 2	PRODUCT 3
EXPORT	3094	1737	640
SOUTH	254	201	110
WEST	101	94	70
NORTH	595	364	146
SOLD TO	PRODUCT 1	PRODUCT 2	PRODUCT 3
EXPORT	3530	1881	800
SOUTH	292	230	100
WEST	104	101	70
NORTH	640	364	146
BACKLOG OF ORDERS	PRODUCT 1	PRODUCT 2	PRODUCT 3
EXPORT	0	0	6
SOUTH	0	0	19
WEST	0	0	0
NORTH	0	0	0
PRODUCTS STOCKS IN	PRODUCT 1	PRODUCT 2	PRODUCT 3
EXPORT	720	319	0
SOUTH	8	20	0
WEST	46	19	15
NORTH	60	119	34
PRODUCT IMPROVEMENTS	MINOR	MINOR	NONE.

MARKET SHARES, BASED ON NO. OF SALES

COY.INFO.	ADVERT. EXPEND.	RESEARCH EXPEND.	CONSUMER ASSESSMENT RATINGS PROD.1	PROD.2	PROD.3	PRODUCT 1 EXP.	S.	W.	N.	PRODUCT 2 EXP.	S.	W.	N.	PRODUCT 3 EXP.	S.	W.	N.
COMPANY 1	66000	32000	****	****	****	2.7	12.5	12.5	12.5	2.4	12.5	12.5	12.5	2.2	12.5	12.5	12.
COMPANY 2	66000	32000	****	****	****	2.7	12.5	12.5	12.5	2.4	12.5	12.5	12.5	2.2	12.5	12.5	12.
COMPANY 3	66000	32000	****	****	****	2.7	12.5	12.5	12.5	2.4	12.5	12.5	12.5	2.2	12.5	12.5	12.
COMPANY 4	66000	32000	****	****	****	2.7	12.5	12.5	12.5	2.4	12.5	12.5	12.5	2.2	12.5	12.5	12.
COMPANY 5	66000	32000	****	****	****	2.7	12.5	12.5	12.5	2.4	12.5	12.5	12.5	2.2	12.5	12.5	12.
COMPANY 6	66000	32000	****	****	****	2.7	12.5	12.5	12.5	2.4	12.5	12.5	12.5	2.2	12.5	12.5	12.
COMPANY 7	66000	32000	****	****	****	2.7	12.5	12.5	12.5	2.4	12.5	12.5	12.5	2.2	12.5	12.5	12.
COMPANY 8	66000	32000	****	****	****	2.7	12.5	12.5	12.5	2.4	12.5	12.5	12.5	2.2	12.5	12.5	12.

GROSS NAT.PRODUCT(SEASONALLY ADJ) 718($M) UNEMPLOYED IN S.AREA(SEASONALLY ADJ) 6% CENTRAL BANK RATE FOR NEXT QTR. 8%

COMPANY INFORMATION	PRICES PRODUCT 1 E	H	PRODUCT 2 E	H	PRODUCT 3 E	H	TOTAL EMPLOYED	SKILLED WAGE RATE
COMPANY 1	118	108	195	183	390	378	165	440
COMPANY 3	118	108	195	183	390	378	165	440
COMPANY 5	118	108	195	183	390	378	165	440
COMPANY 7	118	108	195	183	390	378	165	440
COMPANY 2	118	108	195	183	390	378	165	440
COMPANY 4	118	108	195	183	390	378	165	440
COMPANY 6	118	108	195	183	390	378	165	440
COMPANY 8	118	108	195	183	390	378	165	440

OVERHEADS

PROMOTION & ADVERTISING	66000
SALESMEN'S SALARIES ETC	122998
SALES OFFICE	12998
GUARANTEE SERVICING	9430
TRANSPORT FLEET	121388
HIRED TRANSPORT	21872
PRODUCT RESEARCH	32000
RECRUITMENT & TRAINING	4000
MAINTENANCE	31200
WAREHOUSING & PURCHASING	6258
BUSINESS INTELLIGENCE	4500
MANAGEMENT	75000
CREDIT CONTROL	6193
OTHER COSTS	7299
TOTAL OVERHEADS	521136
TAXABLE PROFIT/LOSS ACCUMULATED	79880

PROFIT & LOSS ACCOUNT

SALES	1466651
OPENING STOCK VALUE	40665
MATERIALS PURCHASED	474352
SKILLED WAGES	136778
UNSKILLED WAGES	198633
MACHINE RUNNING	76625
LESS CLOSING STOCK VALUE	105577
COST OF SALES	821476
GROSS PROFIT/LOSS	645175
INTEREST RECEIVED	0
INTEREST PAID	2630
OVERHEADS	521136
DEPRECIATION	41529
TAX ASSESSED	0
NET PROFIT/LOSS	79880
DIVIDENDS PAID	44000
TRANSFERRED TO RESERVES	35880

BALANCE SHEET

ASSETS.-	
PROPERTY	200000
MACHINES	1247050
VANS	143302
PRODUCT STOCKS	79675
MATERIAL STOCKS	25902
DEBTORS	524624
CASH INVESTED	0
LIABILITIES:-	
TAX ASSESSED	122116
CREDITORS	608712
BANK OVERDRAFT	88400
UNSECURED LOANS	0
NET ASSETS	1401325
ORDINARY CAPITAL	1100000
RESERVES	301325
TOTAL SHAREHOLDERS FUNDS	1401325

CASH FLOW

TRADING RECEIPTS	1442862
CAPITAL RECEIPTS	0
INTEREST RECEIVED	0
INVESTMENTS SOLD	0
ADDITIONAL LOANS	0
TRADING PAYMENTS	1394985
CAPITAL PAYMENTS	0
INTEREST PAID	2630
INVESTMENTS BOUGHT	0
TAX PAID	0
LOANS REPAID	1247
DIVIDENDS PAID	44000
OVERDRAFT LIMIT FOR NEXT QUARTER	369000
PRICE OF MATERIAL ORDERED FOR NEXT QUARTER (1000 UNITS)	37934

	COMPANY 1	COMPANY 2	COMPANY 3	COMPANY 4	COMPANY 5	COMPANY 6	COMPANY 7	COMPANY 8
SHARE PRICES(p.)	144.4	144.4	144.4	144.4	144.4	144.4	144.4	144.4
DIVIDEND %	4	4	4	4	4	4	4	4

GAME HISTORY
GAME HISTORY
GAME HISTORY
GAME HISTORY

'THE SCOTSMAN' MANAGEMENT GAME - 1989 (Round One)

CREATED AND ADMINISTERED BY **EDIT 515 Ltd.**

THIS IS THE SECOND PART OF THE GAME HISTORY.

YEAR 1988, QTR.2

GROUP 1 COMPANY 1 IDENTITY No 1 YEAR 1988 QUARTER 2

REPORT BASED ON DECISIONS BELOW, CHECK THEM BEFORE GOING FURTHER.

	PRODUCT 1	PRODUCT 2	PRODUCT 3
MAJOR IMPROVEMENTS TAKEN UP	0	0	0
PRICES EXPORT AREA	118	195	400
PRICES HOME AREA	108	183	383
PROMOTION IN: TRADE PRESS	6	6	9
ADV. SUPPORT	8	8	10
MERCHANDISING	4	4	5
FITTING TIMES (MINS)	115	170	350

SALESMEN ALLOCATED TO EXPORT 13 S 3 W 1 N 4

SALESMAN'S SALARIES	35	COMMISSION	1
SKILLED WAGE RATE	440	SHIFT LEVEL	2
MANAGEMENT BUDGET	75		
MAINTENANCE HOURS	60	MACHINES SOLD	0
DIVIDEND	0	CREDIT PERIOD	30
VANS BOUGHT	0	VANS SOLD	0
INFORMATION ON COMPANIES	0	MARKET SHARES	0

DELIVERIES REQUESTED TO. (NOT DELIVERED IN FULL IF STARRED)	PRODUCT 1	PRODUCT 2	PRODUCT 3
EXPORT	3000	1700	700
SOUTH	275	200	120
WEST	100	100	75
NORTH	600	300	130
RESEARCH EXPENDITURE	10	12	10

SALESMAN HIRED	0	DISMISSED	0	TRAINED	1
SKILLED MEN HIRED	0	DISMISSED	0	TRAINED	1

MATERIALS ORDERED	16000		
FROM SUPPLIER	2	DELIVERIES	8
MACHINES ORDERED	0		

PLANT/MATERIALS/PERSONNEL

MACHINES AVAILABLE LAST QUARTER	13
MACHINES AVAILABLE NEXT QUARTER	13
VANS AVAILABLE LAST QUARTER	21
SKILLED MAN HOURS:	
AVAILABLE LAST QUARTER	27048
ABSENTEEISM	113
USED LAST QUARTER	20727
NOTICE OF STRIKE WEEKS NEXT QUARTER	0
MACHINE HOURS:	
AVAILABLE LAST QUARTER	14196
BREAKDOWN TIME	30
MAINTENANCE	750
USED LAST QUARTER	9448
MACHINE EFFICIENCY (%)	97.1
MATERIAL UNITS.	
OPENING STOCK	1396
DELIVERED	18000
USED	12006
CLOSING STOCK	7390
ON ORDER FOR NEXT QUARTER	16000
AVAILABLE NEXT QUARTER	23390

PERSONNEL:	SALESMEN	SKILLED MEN	UNSKILLED MEN
AT START OF QTR	21	46	98
RECRUITED	0	0	6
DISMISSED	0	0	0
TRAINED	1	1	
LEFT	0	0	6
AVAIL. FOR NEXT QTR	22	47	98

PRODUCTS/ORDERS/STOCKS

PRODUCT OTS	PRODUCT 1	PRODUCT 2	PRODUCT 3
REQUESTED	3975	2300	1025
PRODUCED	4096	2371	1056
REJECTS	121	71	31
SERVICED	98	58	21
DELIVERED TO	**PRODUCT 1**	**PRODUCT 2**	**PRODUCT 3**
EXPORT	3000	1700	700
SOUTH	275	200	120
WEST	100	100	75
NORTH	600	300	130
ORDERS FROM	**PRODUCT 1**	**PRODUCT 2**	**PRODUCT 3**
EXPORT	3300	1832	655
SOUTH	278	223	123
WEST	110	102	71
NORTH	639	393	155
SOLD TO	**PRODUCT 1**	**PRODUCT 2**	**PRODUCT 3**
EXPORT	3300	1832	660
SOUTH	278	220	120
WEST	110	102	71
NORTH	639	393	155
BACKLOG OF ORDERS	**PRODUCT 1**	**PRODUCT 2**	**PRODUCT 3**
EXPORT	0	0	0
SOUTH	0	1	10
WEST	0	0	0
NORTH	0	0	0
PRODUCTS STOCKS IN	**PRODUCT 1**	**PRODUCT 2**	**PRODUCT 3**
EXPORT	420	187	40
SOUTH	5	0	0
WEST	36	17	19
NORTH	21	26	9
PRODUCT IMPROVEMENTS	MINOR	MAJOR	MINOR

GROSS NAT.PRODUCT(SEASONALLY ADJ) 723(SM) UNEMPLOYED IN S.AREA(SEASONALLY ADJ) 6% CENTRAL BANK RATE FOR NEXT QTR. 8%

COMPANY INFORMATION	PRICES PRODUCT 1 E	H	PRODUCT 2 E	H	PRODUCT 3 E	H	TOTAL EMPLOYED	SKILLED WAGE RATE
COMPANY 1	118	108	195	183	400	383	167	440
COMPANY 3	118	108	195	183	400	383	167	440
COMPANY 5	118	108	195	183	400	383	167	440
COMPANY 7	118	108	195	183	400	383	167	440
COMPANY 2	118	108	195	183	400	383	167	440
COMPANY 4	118	108	195	183	400	383	167	440
COMPANY 6	118	108	195	183	400	383	167	440
COMPANY 8	118	108	195	183	400	383	167	440

OVERHEADS

PROMOTION & ADVERTISING	60000
SALESMEN'S SALARIES ETC	129346
SALES OFFICE	13846
GUARANTEE SERVICING	11300
TRANSPORT FLEET	116332
HIRED TRANSPORT	0
PRODUCT RESEARCH	32000
RECRUITMENT & TRAINING	10000
MAINTENANCE	31200
WAREHOUSING & PURCHASING	8608
BUSINESS INTELLIGENCE	0
MANAGEMENT	75000
CREDIT CONTROL	5910
OTHER COSTS	7248
TOTAL OVERHEADS	500790
TAXABLE PROFIT/LOSS ACCUMULATED	109087

PROFIT & LOSS ACCOUNT

SALES	1390259
OPENING STOCK VALUE	105577
MATERIALS PURCHASED	581990
SKILLED WAGES	94294
UNSKILLED WAGES	158634
MACHINE RUNNING	64442
LESS CLOSING STOCK VALUE	187946
COST OF SALES	816991
GROSS PROFIT/LOSS	573268
INTEREST RECEIVED	0
INTEREST PAID	3139
OVERHEADS	500790
DEPRECIATION	40132
TAX ASSESSED	0
NET PROFIT/LOSS	29207
DIVIDENDS PAID	0
TRANSFERRED TO RESERVES	29207

BALANCE SHEET

ASSETS:-	
PROPERTY	200000
MACHINES	1215874
VANS	134346
PRODUCT STOCKS	47780
MATERIAL STOCKS	140166
DEBTORS	504683
CASH INVESTED	0
LIABILITIES:-	
TAX ASSESSED	0
CREDITORS	688198
BANK OVERDRAFT	124119
UNSECURED LOANS	0
NET ASSETS	1430532
ORDINARY CAPITAL	1100000
RESERVES	330532
TOTAL SHAREHOLDERS FUNDS	1430532

CASH FLOW

TRADING RECEIPTS	1410200
CAPITAL RECEIPTS	0
INTEREST RECEIVED	0
INVESTMENTS SOLD	0
ADDITIONAL LOANS	35719
TRADING PAYMENTS	1320664
CAPITAL PAYMENTS	0
INTEREST PAID	3139
INVESTMENTS BOUGHT	0
TAX PAID	122116
LOANS REPAID	0
DIVIDENDS PAID	0
OVERDRAFT LIMIT FOR NEXT QUARTER	407000
PRICE OF MATERIAL ORDERED FOR NEXT QUARTER (1000 UNITS)	37384

	COMPANY 1	COMPANY 2	COMPANY 3	COMPANY 4	COMPANY 5	COMPANY 6	COMPANY 7	COMPANY 8
SHARE PRICES(p.)	148.3	148.3	148.3	148.3	148.3	148.3	148.3	148.3
DIVIDEND %	0	0	0	0	0	0	0	0

GAME HISTORY
GAME HISTORY
GAME HISTORY
GAME HISTORY
GAME HISTORY

'THE SCOTSMAN' MANAGEMENT GAME - 1989 (Round One)

CREATED AND ADMINISTERED BY EDIT 515 Ltd.

THIS IS THE THIRD PART OF THE GAME HISTORY.

YEAR 1988, QTR. 3

GROUP 1 COMPANY 1 IDENTITY No 1 YEAR 1988 QUARTER 3

REPORT BASED ON DECISIONS BELOW. CHECK THEM BEFORE GOING FURTHER.

	PRODUCT 1	PRODUCT 2	PRODUCT 3
MAJOR IMPROVEMENTS TAKEN UP	0	0	0
PRICES EXPORT AREA	115	195	400
HOME AREA	108	183	383
PROMOTION IN TRADE PRESS	10	6	6
ADV. SUPPORT	12	6	10
MERCHANDISING	4	4	4
FITTING TIMES (MINS)	117	170	350

SALESMEN ALLOCATED TO EXPORT 13 S 3 W 1 N 5

SALESMAN'S SALARIES	35	COMMISSION	1
SKILLED WAGE RATE	440	SHIFT LEVEL	2
MANAGEMENT BUDGET	80		
MAINTENANCE HOURS	60	MACHINES SOLD	0
DIVIDEND	5	CREDIT PERIOD	30
VANS BOUGHT	0	VANS SOLD	0
INFORMATION ON COMPANIES	0	MARKET SHARES	0

DELIVERIES REQUESTED TO	PRODUCT 1	PRODUCT 2	PRODUCT 3
EXPORT	3000	1650	600
SOUTH	260	220	125
WEST	100	100	65
NORTH	600	350	150

NOT DELIVERED IN FULL IF STARRED

	PRODUCT 1	PRODUCT 2	PRODUCT 3
RESEARCH EXPENDITURE	10	10	12

SALESMAN HIRED	0	DISMISSED	0	TRAINED	0
SKILLED MEN HIRED	0	DISMISSED	0	TRAINED	0

MATERIALS ORDERED 12000
FROM SUPPLIER 2 DELIVERIES 5
MACHINES ORDERED 0

PLANT/MATERIALS/PERSONNEL

MACHINES AVAILABLE LAST QUARTER	13
MACHINES AVAILABLE NEXT QUARTER	13
VANS AVAILABLE LAST QUARTER	21
SKILLED MAN HOURS:	
AVAILABLE LAST QUARTER	27636
ABSENTEEISM	125
USED LAST QUARTER	20377
NOTICE OF STRIKE WEEKS NEXT QUARTER	0
MACHINE HOURS:	
AVAILABLE LAST QUARTER	14196
BREAKDOWN TIME	26
MAINTENANCE	754
USED LAST QUARTER	9267
MACHINE EFFICIENCY (%)	97.2
MATERIAL UNITS:	
OPENING STOCK	7390
DELIVERED	16000
USED	11766
CLOSING STOCK	11624
ON ORDER FOR NEXT QUARTER	12000
AVAILABLE NEXT QUARTER	23624

PERSONNEL:	SALESMEN	SKILLED MEN	UNSKILLED MEN
AT START OF QTR	22	47	98
RECRUITED	0	0	6
DISMISSED	0	0	0
TRAINED	0	0	
LEFT	0	0	6
AVAIL. FOR NEXT QTR	22	47	98

PRODUCTS/ORDERS/STOCKS

PRODUCT QTS.	PRODUCT 1	PRODUCT 2	PRODUCT 3
REQUESTED	3960	2320	940
PRODUCED	4078	2392	960
REJECTS	118	72	28
SERVICED	93	55	21

DELIVERED TO	PRODUCT 1	PRODUCT 2	PRODUCT 3
EXPORT	3000	1650	600
SOUTH	260	220	125
WEST	100	100	65
NORTH	600	350	150

ORDERS FROM	PRODUCT 1	PRODUCT 2	PRODUCT 3
EXPORT	3285	1649	615
SOUTH	261	210	119
WEST	102	94	68
NORTH	597	369	146

SOLD TO	PRODUCT 1	PRODUCT 2	PRODUCT 3
EXPORT	3285	1649	615
SOUTH	261	211	125
WEST	102	94	68
NORTH	597	369	146

BACKLOG OF ORDERS	PRODUCT 1	PRODUCT 2	PRODUCT 3
EXPORT	0	0	0
SOUTH	0	0	2
WEST	0	0	0
NORTH	0	0	0

PRODUCTS STOCKS IN	PRODUCT 1	PRODUCT 2	PRODUCT 3
EXPORT	135	188	25
SOUTH	4	9	0
WEST	34	23	16
NORTH	24	7	13

PRODUCT IMPROVEMENTS	PRODUCT 1	PRODUCT 2	PRODUCT 3
	MINOR	NONE	NONE

GROSS NAT.PRODUCT(SEASONALLY ADJ) 728($M) UNEMPLOYED IN S.AREA(SEASONALLY ADJ) 6% CENTRAL BANK RATE FOR NEXT QTR. 8%

COMPANY INFORMATION	PRODUCT 1 E	PRODUCT 1 H	PRODUCT 2 E	PRODUCT 2 H	PRODUCT 3 E	PRODUCT 3 H	TOTAL EMPLOYED	SKILLED WAGE RATE
COMPANY 1	115	108	195	183	400	383	167	440
COMPANY 3	115	108	195	183	400	383	167	440
COMPANY 5	115	108	195	183	400	383	167	440
COMPANY 7	115	108	195	183	400	383	167	440
COMPANY 2	115	108	195	183	400	383	167	440
COMPANY 4	115	108	195	183	400	383	167	440
COMPANY 6	115	108	195	183	400	383	167	440
COMPANY 8	115	108	195	183	400	383	167	440

OVERHEADS

PROMOTION & ADVERTISING	62000
SALESMEN'S SALARIES ETC	133997
SALES OFFICE	12997
GUARANTEE SERVICING	10870
TRANSPORT FLEET	115638
HIRED TRANSPORT	0
PRODUCT RESEARCH	32000
RECRUITMENT & TRAINING	3000
MAINTENANCE	31200
WAREHOUSING & PURCHASING	11909
BUSINESS INTELLIGENCE	0
MANAGEMENT	80000
CREDIT CONTROL	5641
OTHER COSTS	7263
TOTAL OVERHEADS	506515
TAXABLE PROFIT/LOSS ACCUMULATED	104544

PROFIT & LOSS ACCOUNT

SALES	1307379
OPENING STOCK VALUE	187946
MATERIALS PURCHASED	510022
SKILLED WAGES	91060
UNSKILLED WAGES	158634
MACHINE RUNNING	63678
LESS CLOSING STOCK VALUE	250185
COST OF SALES	761155
GROSS PROFIT/LOSS	546224
INTEREST RECEIVED	0
INTEREST PAID	5457
OVERHEADS	506515
DEPRECIATION	38795
TAX ASSESSED	0
NET PROFIT/LOSS	-4543
DIVIDENDS PAID	55000
TRANSFERRED TO RESERVES	-59543

BALANCE SHEET

ASSETS:-	
PROPERTY	200000
MACHINES	1185476
VANS	125949
PRODUCT STOCKS	32910
MATERIAL STOCKS	217275
DEBTORS	475729
CASH INVESTED	0
LIABILITIES:-	
TAX ASSESSED	0
CREDITORS	621101
BANK OVERDRAFT	245249
UNSECURED LOANS	0
NET ASSETS	1370989
ORDINARY CAPITAL	1100000
RESERVES	270989
TOTAL SHAREHOLDERS FUNDS	1370989

CASH FLOW

TRADING RECEIPTS	1336333
CAPITAL RECEIPTS	0
INTEREST RECEIVED	0
INVESTMENTS SOLD	0
ADDITIONAL LOANS	121130
TRADING PAYMENTS	1397006
CAPITAL PAYMENTS	0
INTEREST PAID	5457
INVESTMENTS BOUGHT	0
TAX PAID	0
LOANS REPAID	0
DIVIDENDS PAID	55000
OVERDRAFT LIMIT FOR NEXT QUARTER	464000
PRICE OF MATERIAL ORDERED FOR NEXT QUARTER (1000 UNITS)	36284

	COMPANY 1	COMPANY 2	COMPANY 3	COMPANY 4	COMPANY 5	COMPANY 6	COMPANY 7	COMPANY 8
SHARE PRICES(p.)	138.8	138.8	138.8	138.8	138.8	138.8	138.8	138.8
DIVIDEND %	5	5	5	5	5	5	5	5

'THE SCOTSMAN' MANAGEMENT GAME - 1989 (Round One)

GAME HISTORY
GAME HISTORY
GAME HISTORY
GAME HISTORY
GAME HISTORY

CREATED AND ADMINISTERED BY **EDIT 515 Ltd.**

THIS IS THE FOURTH AND LAST PART OF THE GAME HISTORY.

YEAR 1988, QTR. 4

GROUP 1 COMPANY 1 IDENTITY No. 1 YEAR 1988 QUARTER 4

REPORT BASED ON DECISIONS BELOW, CHECK THEM BEFORE GOING FURTHER.

	PRODUCT 1	PRODUCT 2	PRODUCT 3
MAJOR IMPROVEMENTS TAKEN UP	0	1	0
PRICES EXPORT AREA	115	195	400
PRICES HOME AREA	108	183	385
PROMOTION IN: TRADE PRESS	6	10	7
ADV. SUPPORT	10	15	8
MERCHANDISING	4	4	5
FITTING TIMES (MINS)	115	170	340

SALESMEN ALLOCATED TO EXPORT 13 S 3 W 1 N 5

SALESMAN'S SALARIES	35	COMMISSION	1
SKILLED WAGE RATE	440	SHIFT LEVEL	2
MANAGEMENT BUDGET	80		
MAINTENANCE HOURS	60	MACHINES SOLD	0
DIVIDEND	0	CREDIT PERIOD	30
VANS BOUGHT	1	VANS SOLD	0
INFORMATION ON COMPANIES	1	MARKET SHARES	0

DELIVERIES REQUESTED TO		PRODUCT 1	PRODUCT 2	PRODUCT 3
NOT DELIVERED IN FULL IF STARRED	EXPORT	3700	2250	800
	SOUTH	300	250	150
	WEST	100	110	60
	NORTH	725	450	170
RESEARCH EXPENDITURE		13	10	12

SALESMAN HIRED	0	DISMISSED	0	TRAINED	0
SKILLED MEN HIRED	0	DISMISSED	0	TRAINED	0

MATERIALS ORDERED	10000		
FROM SUPPLIER	2	DELIVERIES	4
MACHINES ORDERED	0		

PLANT/MATERIALS/PERSONNEL

MACHINES AVAILABLE LAST QUARTER	13
MACHINES AVAILABLE NEXT QUARTER	13
VANS AVAILABLE LAST QUARTER	22
SKILLED MAN HOURS:	
AVAILABLE LAST QUARTER	27636
ABSENTEEISM	121
USED LAST QUARTER	25360
NOTICE OF STRIKE WEEKS NEXT QUARTER	0
MACHINE HOURS:	
AVAILABLE LAST QUARTER	14196
BREAKDOWN TIME	31
MAINTENANCE	749
USED LAST QUARTER	11686
MACHINE EFFICIENCY (%)	97.2
MATERIAL UNITS:	
OPENING STOCK	11624
DELIVERED	12000
USED	14930
CLOSING STOCK	8694
ON ORDER FOR NEXT QUARTER	10000
AVAILABLE NEXT QUARTER	18694

PERSONNEL:	SALESMEN	SKILLED MEN	UNSKILLED MEN
AT START OF QTR	22	47	98
RECRUITED	0	0	6
DISMISSED	0	0	0
TRAINED	0	0	
LEFT	0	0	6
AVAIL FOR NEXT QTR	22	47	98

PRODUCTS/ORDERS/STOCKS

PRODUCT QTS	PRODUCT 1	PRODUCT 2	PRODUCT 3
REQUESTED	4825	3060	1180
PRODUCED	4972	3155	1216
REJECTS	147	95	36
SERVICED	89	53	20
DELIVERED TO			
EXPORT	3700	2250	800
SOUTH	300	250	150
WEST	100	110	60
NORTH	725	450	170
ORDERS FROM			
EXPORT	3897	1976	636
SOUTH	310	242	130
WEST	122	111	71
NORTH	715	426	158
SOLD TO			
EXPORT	3835	1976	636
SOUTH	304	242	131
WEST	122	110	71
NORTH	715	426	158
BACKLOG OF ORDERS			
EXPORT	31	0	0
SOUTH	3	0	0
WEST	0	0	0
NORTH	0	0	0
PRODUCTS STOCKS IN			
EXPORT	0	274	189
SOUTH	0	8	19
WEST	12	0	5
NORTH	34	24	25
PRODUCT IMPROVEMENTS	MINOR	NONE	MINOR

COY. INFO.	ADVERT. EXPEND.	RESEARCH EXPEND.	CONSUMER ASSESSMENT RATINGS PROD.1	PROD.2	PROD.3
COMPANY 1	69000	35000	****	****	****
COMPANY 2	69000	35000	****	****	****
COMPANY 3	69000	35000	****	****	****
COMPANY 4	69000	35000	****	****	****
COMPANY 5	69000	35000	****	****	****
COMPANY 6	69000	35000	****	****	****
COMPANY 7	69000	35000	****	****	****
COMPANY 8	69000	35000	****	****	****

GROSS NAT.PRODUCT(SEASONALLY ADJ) 733($M) UNEMPLOYED IN S.AREA(SEASONALLY ADJ) 6% CENTRAL BANK RATE FOR NEXT QTR. 8%

COMPANY INFORMATION	PRODUCT 1 E	PRODUCT 1 H	PRODUCT 2 E	PRODUCT 2 H	PRODUCT 3 E	PRODUCT 3 H	TOTAL EMPLOYED	SKILLED WAGE RATE
COMPANY 1	115	108	195	183	400	385	167	440
COMPANY 3	115	108	195	183	400	385	167	440
COMPANY 5	115	108	195	183	400	385	167	440
COMPANY 7	115	108	195	183	400	385	167	440
COMPANY 2	115	108	195	183	400	385	167	440
COMPANY 4	115	108	195	183	400	385	167	440
COMPANY 6	115	108	195	183	400	385	167	440
COMPANY 8	115	108	195	183	400	385	167	440

OVERHEADS

PROMOTION & ADVERTISING	69000
SALESMEN'S SALARIES ETC	135925
SALES OFFICE	14925
GUARANTEE SERVICING	10420
TRANSPORT FLEET	127036
HIRED TRANSPORT	4127
PRODUCT RESEARCH	35000
RECRUITMENT & TRAINING	3000
MAINTENANCE	31200
WAREHOUSING & PURCHASING	12453
BUSINESS INTELLIGENCE	3000
MANAGEMENT	80000
CREDIT CONTROL	6544
OTHER COSTS	7346
TOTAL OVERHEADS	539976
TAXABLE PROFIT/LOSS ACCUMULATED	233630

PROFIT & LOSS ACCOUNT

SALES	1508647
OPENING STOCK VALUE	250185
MATERIALS PURCHASED	371096
SKILLED WAGES	127626
UNSKILLED WAGES	184802
MACHINE RUNNING	74276
LESS CLOSING STOCK VALUE	213686
COST OF SALES	794299
GROSS PROFIT/LOSS	714348
INTEREST RECEIVED	0
INTEREST PAID	7152
OVERHEADS	539976
DEPRECIATION	38134
TAX ASSESSED	70089
NET PROFIT/LOSS	58997
DIVIDENDS PAID	0
TRANSFERRED TO RESERVES	58997

BALANCE SHEET

ASSETS:-	
PROPERTY	200000
MACHINES	1155842
VANS	127449
PRODUCT STOCKS	55960
MATERIAL STOCKS	157726
DEBTORS	538302
CASH INVESTED	0
LIABILITIES:-	
TAX ASSESSED	70089
CREDITORS	496396
BANK OVERDRAFT	238808
UNSECURED LOANS	0
NET ASSETS	1429986
ORDINARY CAPITAL	1100000
RESERVES	329986
TOTAL SHAREHOLDERS FUNDS	1429986

CASH FLOW

TRADING RECEIPTS	1446074
CAPITAL RECEIPTS	0
INTEREST RECEIVED	0
INVESTMENTS SOLD	0
ADDITIONAL LOANS	0
TRADING PAYMENTS	1422481
CAPITAL PAYMENTS	10000
INTEREST PAID	7152
INVESTMENTS BOUGHT	0
TAX PAID	0
LOANS REPAID	6441
DIVIDENDS PAID	0
OVERDRAFT LIMIT FOR NEXT QUARTER	529000
PRICE OF MATERIAL ORDERED FOR NEXT QUARTER (1000 UNITS)	35734

	COMPANY 1	COMPANY 2	COMPANY 3	COMPANY 4	COMPANY 5	COMPANY 6	COMPANY 7	COMPANY 8
SHARE PRICES(p.)	147.9	147.9	147.9	147.9	147.9	147.9	147.9	147.9
DIVIDEND %	0	0	0	0	0	0	0	0

EXECUTIVE

Forewords

The Executive range of simulations model the Motor Industry, using real data. The Executive games can be used to illustrate a number of management training aspects. Their run is based on 9 teams that compete against each other and the existing market. The Executive range is use at more than 100 companies and educational institutions.

Game's purposes

The Executive game has been used by Manufacturing, Finance and Service companies. It has been employed as a run/test of the learning elements taught during a previous training program.

The main learning objectives are :

- Team Building
- Decision making,..
- Awareness of the existing functions in a factory
- Strategic Planning,..
- Finance,..

Executive is considered to be a fully interactive game. Its interaction is based on the fact that the decisions taken by one team will influence the other teams playing the game. The Executive's Master Simulation simulates the influence of non actively playing teams.

Game description

The Executive game is based on the Western car market, which is divided into 16 market sectors. The reason why this market was chosen, is because it is a fairly known business and it is also easy to understand. Game's players what it is like.

The working process is very similar to the one described in TOPAZ. Players are gathered in groups, where each of their members performs a different role. After a discussion period, the players will decide about certain aspects of the factory. Those decisions will be entered into the computer for later processing. Some numerical results will come out after. Those results are the summary reports of the performance of the factory.

The factory to be played starts from scratch, there is only money. The player has to decide about the way the money is going to be spend (workforce, facilities,..). The game could also start from an existing set-up or historic position.

The decisions could be supported by accessing to previous teams' information. The Tutor can also access that information.

Technicalities

Apart from the basic model, there are some other different versions of Executive. These models have some added modules that will allow the more specialized study of factory related issues as Quality, Finance, and Marketing.

The number of teams playing is between one and nine. If the game's purpose is teambuilding, leadership,..it is advice to work with teams of five people; and two, three or even four players if the purpose is strategy, finance,..The number of teams playing will influence the Market Place playing with. The game's cycle is one year.

The reports generated by Executive are shown in the next pages.

EXECUTIVE REPORTS

Executive-300 Business Simulation. Release 4.40.
A Product of April Computing Executive Ltd.
Purchased by : April Computing - PW

PRIVATE COMPANY REPORT

Team 1 - CAT Motors　　Exercise : STD　　End Of Year 3

Sector/ Model	Produced	Sold	Stocks	Price	Materials Cost	Market Share %
6 Felix	155806	114230	62007	9500.00	5970.57	3.17
11 Tom	155806	142475	23241	12000.00	8677.99	4.61
16 Garfield	10380	5983	8683	36000.00	13685.41	0.60

Total Workforce	40000	
Productivity	8.05	cars / man / year
Productivity Index	1.04	
Days lost to strikes	10	

Total Market	Small	L. Med	U. Med	Luxury
Sizes	2275643	3606546	3091325	990380

PROFIT AND LOSS ACCOUNT (£ millions)

Sales	3010.27	
Cost of Sales *		2345.97

Gross Profit	664.31	
Overheads:		
Fixed Overheads		249.01
Cost of Stock Upkeep		62.88
Promotion		68.10
Professional Charges		2.95
Depreciation		48.60

Operating Profit	232.77	
Interest on Current Account	52.78	
Interest on Loans		23.73

Pre-Tax Profit	261.82	
Tax		91.64
	=========	
Post-Tax Profit	170.18	
	=========	

* Cost of Sales is calculated as

Opening Stock	314.41	
plus Materials Cost	2424.39	
plus Wages	453.08	
less Closing Stock		845.91

Cost of Sales	2345.97	

CASH FLOW (£ millions)

Opening Bank Balance	586.84	

Receipts:		
Revenues	3010.27	
Net Interest	29.05	
Payments:		
Materials Cost		2424.39
Wages Cost		453.08
Total Overheads		382.94
Loan Repayments		124.97
Tax Payments		177.26
------------------------	------------	
Balance Before Loans	63.52	
------------------------	------------	
New Normal Loan	250.00	
------------------------	------------	
Closing Bank Balance	313.52	
------------------------	------------	

BALANCE SHEET (£ millions)

Fixed Assets:	
Cost	600.00
Depreciation	-162.60
Book Value	437.40
Current Assets:	
Stock	845.91
Bank Balance	313.52
Current Liabilities:	
Outstanding Emergency Loan	0.00
Tax Liabilities	-91.64
Net Current Assets	1067.80
Total Assets Less Current Liabilities	1505.20
Capital and Reserves:	
Shareholder's Equity	500.00
Retained Profit	755.20
Total Shareholders Funds	1255.20
Long Term Liabilities:	
Outstanding Normal Loan	250.00
Total Capital Employed	1505.20

FINANCIAL INDICATORS

Outstanding Debt (£m)	250.00	Return On Assets	17.39 %
Current Ratio	12.65	Gross Margin	22.07 %
Quick Ratio	3.42	Post-Tax Profit / Sales	5.65 %
Liquidity Ratio	*********	Profit / Employee (£)	6545.41

There are no R & D projects for this team.

EXTRAS INCLUDED

	MDL 1 Felix	MDL 2 Tom	MDL 3 Garfield
1. Larger Engine Option	Yes	Yes	Yes
2. 5-Speed gearbox	Yes	Yes	Yes
3. Fuel Injection	Yes	Yes	Yes
4. Alloy Wheels	Yes	Yes	Yes
5. Sunroof	Yes	Yes	Yes
6. Headlamp Washwipes			Yes
7. Spoiler Set	Yes	Yes	Yes
8. Metallic Paint	Yes	Yes	Yes
9. Electric Windows	Yes	Yes	Yes
10. Electric Aerial	Yes	Yes	Yes
11. Automatic Transmission			Yes
12. Anti-lock Braking (ABS)	Yes	Yes	Yes
13. Electric Seat Heating			Yes
14. Electric Wing Mirrors			Yes
15. Air Conditioning			Yes
16. Power Steering		Yes	Yes
17. Fog Lamps	Yes	Yes	Yes
18. Cruise Control			Yes
19. Heated Front Screen	Yes		Yes
20. Leather Upholstery			Yes
21. Run Flat Tyres			Yes
22. Electrical Adjustable Seats			Yes
23. Adjustable Steering Column			Yes
24. Central Locking	Yes	Yes	Yes
25. Self Leveling Suspension			Yes
26. Towing Package			Yes
27. Limited Slip Differential			Yes
28. Sports Steering Wheel	Yes		
29. Low Profile Tyres		Yes	Yes
30. CD Player	Yes	Yes	Yes

DATA ON COMPETITION - Year 3

1. SALES DISTRIBUTION

Team	Small	L. Med	U. Med	Luxury
1		114230 *		
1			142475 *	
1				5983 *
2		162084 *		
2			173554 *	
2				9610 *
3	110245 *			
3		115801 *		
3			122911 *	
3			137156 *	
3		122235 *		
Foreign Sales	2165398	3092196	2515229	974787

Sales By Size Group

* Models in stock.

2. MARKET DISTRIBUTION Product Prices (£) / Market Share (%)

Team	Small		L. Med		U. Med		Luxury	
1			9500	3.17				
1					12000	4.61		
1							36000	0.60
2			9250	4.49				
2					11500	5.61		
2							32000	0.97
3	5000	4.84						
3			9000	3.21				
3					11500	3.98		
3					8500	4.44		
3			8000	3.39				

3. MARKET SECTORS

Market Sector Of Model

Team	1	2	3	4	5
1.	6	11	16		
2.	6	11	16		
3.	5	6	11	7	10

4. COSTS

Basic Materials Costs (£)

Model

Team	1	2	3	4	5
1	5970.57	8677.99	13685.41		
2	5187.39	7663.20	11200.06		
3	3183.17	5187.39	7663.20	7663.20	5187.39

5. MODEL SPECIFICATIONS

Team 1 — EXTRAS SELECTED

	MDL 1 Felix	MDL 2 Tom	MDL 3 Garfield
1. Larger Engine Option	Yes	Yes	Yes
2. 5-Speed gearbox	Yes	Yes	Yes
3. Fuel Injection	Yes	Yes	Yes
4. Alloy Wheels	Yes	Yes	Yes
5. Sunroof	Yes	Yes	Yes
6. Headlamp Washwipes			Yes
7. Spoiler Set	Yes	Yes	Yes
8. Metallic Paint	Yes	Yes	Yes
9. Electric Windows	Yes	Yes	Yes
10. Electric Aerial	Yes	Yes	Yes
11. Automatic Transmission			Yes
12. Anti-lock Braking (ABS)	Yes	Yes	Yes
13. Electric Seat Heating			Yes
14. Electric Wing Mirrors			Yes
15. Air Conditioning			Yes
16. Power Steering		Yes	Yes
17. Fog Lamps	Yes	Yes	Yes
18. Cruise Control			Yes
19. Heated Front Screen	Yes		Yes
20. Leather Upholstery			Yes
21. Run Flat Tyres			Yes
22. Electrical Adjustable Seats			Yes
23. Adjustable Steering Column			Yes
24. Central Locking	Yes	Yes	Yes
25. Self Leveling Suspension			Yes
26. Towing Package			Yes
27. Limited Slip Differential			Yes
28. Sports Steering Wheel	Yes		
29. Low Profile Tyres		Yes	Yes
30. CD Player	Yes	Yes	Yes

Team 1 has no R & D projects online.

No extras have been selected by team 2.

Team 2 — RESEARCH AND DEVELOPMENT PROJECTS

	MDL 1 Hai 6	MDL 2 Hai 11	MDL 3 Hai 16
1. Aerodynamic Remodelling	Online	Online	Online
3. Facelift	Online	Online	Online
4. Turbo Charging	Online	Online	Online
8. Four Wheel Drive	Online	Online	
9. Diesel Engine	Online	Online	Online
10. Electronic Engine Control	Online	Online	Online
12. Multi Valve Engine	Online	Online	Online
13. Improved Build Quality	Online	Online	Online
14. Automatic Crash Restraint	Online	Online	Online
15. Stronger Passenger Cell	Online	Online	Online
16. Catalytic Converter	Online	Online	Online
18. Anti Engine Noise System	Online	Online	Online
20. Breathalyser Interlock	Online	Online	Online
21. Drowsiness Detection	Online	Online	
22. Automatic Window Wipers		Online	Online

No extras have been selected by team 3.

Team 3 — RESEARCH AND DEVELOPMENT PROJECTS

	MDL 1 Exp 5	MDL 2 Exp 6	MDL 3 Exp 11	MDL 4 Exp 7	MDL 5 Exp 10
3. Facelift	Online	Online	Online	Online	Online
21. Drowsiness Detection			Online	Online	

6. PRODUCTION

Team	Total Workforce	Productivity Index	Productivity (cars/man/year)	Days Lost To Strikes	Weekly Salary (£m)
1	40000	1.04	8.05	10	235
2	20000	2.07	16.05	7	270
3	60000	1.04	10.44	2	220

Team	Sales (£) / Employee	Pre Tax Profit / Employee (£)
1	75256.82	6545.41
2	190133.40	20088.95
3	85843.61	1164.26

7. PROMOTION BUDGET

Team	(£m)
1	68.10
2	53.30
3	126.00

Issue 3 of the Car Journal

Profits for team 1 were £170.18m
Profits for team 2 were £261.16m
Profits for team 3 were £45.41m

Economists predict that next year inflation will be around 4%.

TUTORS REPORT - Year 3

Team	Sales (£m)	Gross Profit (£m)	Operating Profit (£m)	Post Tax Profit (£m)	Stock Value (£m)	Shareholders Funds (£m)
1	3010.27	664.31	232.77	170.18	845.91	1255.20
2	3802.67	1265.99	453.13	261.16	118.06	930.09
3	5150.62	939.66	92.11	45.41	212.89	712.08

Team	Gross Margin (%)	Sales Margin (%)	Current Ratio	Quick Ratio	Return on Assets (%)
1	22.07	7.73	12.65	3.42	17.39
2	33.29	11.92	5.17	4.33	24.14
3	18.24	1.79	12.76	4.05	7.09

MODEL DETAILS BY TEAM

Team 1 - CAT Motors

Model Name	Felix	Tom	Garfield
Market Sector	6	11	16
Workforce	15000	15000	10000
Sale Price	9500.00	12000.00	36000.00
Basic Cost	5187.39	7663.20	11200.06
Cost of Extras	783.18	1014.80	2485.35
Gross Margin (%)	25.67	18.60	31.67
Produced	155806	155806	10380
Cars Sold	114230	142475	5983
Cars in Stock	62007	23241	8683
Market Share (%)	3.17	4.61	0.60

Team 2 - Hai Tek Motors

Model Name	Hai 6	Hai 11	Hai 16
Market Sector	6	11	16
Workforce	7500	7500	5000
Sale Price	9250.00	11500.00	32000.00
Basic Cost	5187.39	7663.20	11200.06
Cost of Extras	0.00	0.00	0.00
Gross Margin (%)	37.00	27.80	44.99
Produced	155344	155344	10350
Cars Sold	162084	173554	9610
Cars in Stock	5505	6618	2412
Market Share (%)	4.49	5.61	0.97

Team 3 - Expansion Motors Inc.

Model Name	Exp 5	Exp 6	Exp 11	Exp 7	Exp 10
Market Sector	5	6	11	7	10
Workforce	12000	12000	12000	12000	12000
Sale Price	5000.00	9000.00	11500.00	8500.00	8000.00
Basic Cost	3183.17	5187.39	7663.20	7663.20	5187.39
Cost of Extras	0.00	0.00	0.00	0.00	0.00
Gross Margin (%)	15.27	30.66	24.20	-2.55	21.99
Produced	125311	125311	125311	125311	125311
Cars Sold	110245	115801	122911	137156	122235
Cars in Stock	17851	9510	2400	4885	3076
Market Share (%)	4.84	3.21	3.98	4.44	3.39

DATA ON COMPETITION - Year 3

1. SALES DISTRIBUTION

Team	Small	L. Med	U. Med	Luxury
1		114230 *		
1			142475 *	
1				5983 *
2		162084 *		
2			173554 *	
2				9610 *
3	110245 *			
3		115801 *		
3			122911 *	
3			137156 *	
3		122235 *		
Foreign Sales	2165398	3092196	2515229	974787

Sales By Size Group

* Models in stock.

2. MARKET DISTRIBUTION Product Prices (£) / Market Share (%)

Team	Small		L. Med		U. Med		Luxury	
1			9500	3.17				
1					12000	4.61		
1							36000	0.60
2			9250	4.49				
2					11500	5.61		
2							32000	0.97
3	5000	4.84						
3			9000	3.21				
3					11500	3.98		
3					8500	4.44		
3			8000	3.39				

3. MARKET SECTORS

Market Sector Of Model

Team	1	2	3	4	5
1.	6	11	16		
2.	6	11	16		
3.	5	6	11	7	10

4. COSTS

Basic Materials Costs (£)

Model

Team	1	2	3	4	5
1	5970.57	8677.99	13685.41		
2	5187.39	7663.20	11200.06		
3	3183.17	5187.39	7663.20	7663.20	5187.39

5. MODEL SPECIFICATIONS

Team 1 EXTRAS SELECTED

	MDL 1 Felix	MDL 2 Tom	MDL 3 Garfield
1. Larger Engine Option	Yes	Yes	Yes
2. 5-Speed gearbox	Yes	Yes	Yes
3. Fuel Injection	Yes	Yes	Yes
4. Alloy Wheels	Yes	Yes	Yes
5. Sunroof	Yes	Yes	Yes
6. Headlamp Washwipes			Yes
7. Spoiler Set	Yes	Yes	Yes
8. Metallic Paint	Yes	Yes	Yes
9. Electric Windows	Yes	Yes	Yes
10. Electric Aerial	Yes	Yes	Yes
11. Automatic Transmission			Yes
12. Anti-lock Braking (ABS)	Yes	Yes	Yes
13. Electric Seat Heating			Yes
14. Electric Wing Mirrors			Yes
15. Air Conditioning			Yes
16. Power Steering		Yes	Yes
17. Fog Lamps	Yes	Yes	Yes
18. Cruise Control			Yes
19. Heated Front Screen	Yes		Yes
20. Leather Upholstery			Yes
21. Run Flat Tyres			Yes
22. Electrical Adjustable Seats			Yes
23. Adjustable Steering Column			Yes
24. Central Locking	Yes	Yes	Yes
25. Self Leveling Suspension			Yes
26. Towing Package			Yes
27. Limited Slip Differential			Yes
28. Sports Steering Wheel	Yes		
29. Low Profile Tyres		Yes	Yes
30. CD Player	Yes	Yes	Yes

Team 1 has no R & D projects online.

No extras have been selected by team 2.

Team 2 RESEARCH AND DEVELOPMENT PROJECTS

	MDL 1 Hai 6	MDL 2 Hai 11	MDL 3 Hai 16
1. Aerodynamic Remodelling	Online	Online	Online
3. Facelift	Online	Online	Online
4. Turbo Charging	Online	Online	Online
8. Four Wheel Drive	Online	Online	
9. Diesel Engine	Online	Online	Online
10. Electronic Engine Control	Online	Online	Online
12. Multi Valve Engine	Online	Online	Online
13. Improved Build Quality	Online	Online	Online
14. Automatic Crash Restraint	Online	Online	Online
15. Stronger Passenger Cell	Online	Online	Online
16. Catalytic Converter	Online	Online	Online
18. Anti Engine Noise System	Online	Online	Online
20. Breathalyser Interlock	Online	Online	Online
21. Drowsiness Detection	Online	Online	
22. Automatic Window Wipers		Online	Online

No extras have been selected by team 3.

Team 3 RESEARCH AND DEVELOPMENT PROJECTS

	MDL 1 Exp 5	MDL 2 Exp 6	MDL 3 Exp 11	MDL 4 Exp 7	MDL 5 Exp 10
3. Facelift	Online	Online	Online	Online	Online
21. Drowsiness Detection			Online	Online	

6. PRODUCTION

Team	Total Workforce	Productivity Index	Productivity (cars/man/year)	Days Lost To Strikes	Weekly Salary (£m)
1	40000	1.04	8.05	10	235
2	20000	2.07	16.05	7	270
3	60000	1.04	10.44	2	220

Team	Sales (£) / Employee	Pre Tax Profit / Employee (£)
1	75256.82	6545.41
2	190133.40	20088.95
3	85843.61	1164.26

7. PROMOTION BUDGET

Team	(£m)
1	68.10
2	53.30
3	126.00

BISSIM 6

Forewords

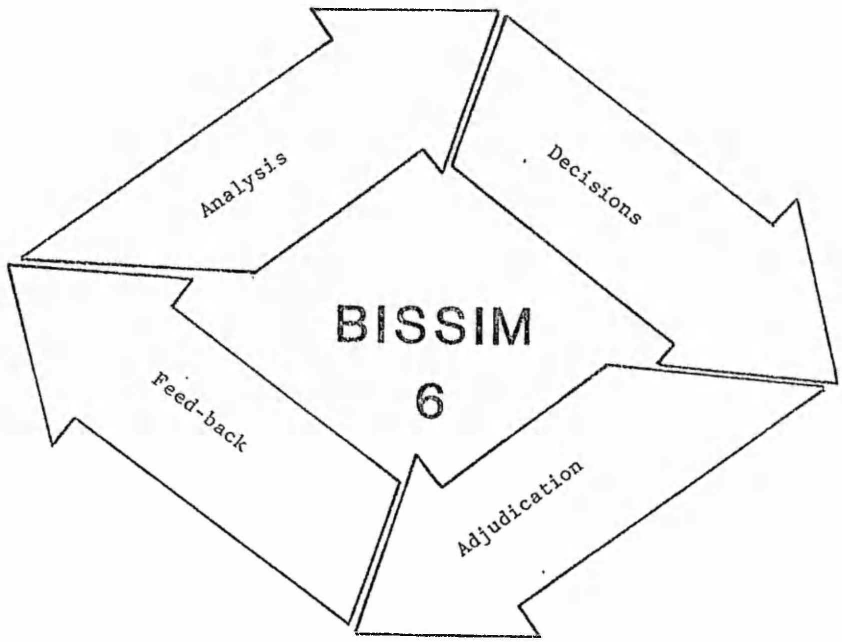
Bissim is a realistic and detailed simulation of a manufacturing company developed by Management Games Ltd. The BISSIM simulation/game allows up to 100 participants to take decisions in all the major areas of finance, production, and marketing. The structure of the game enables teams to move from the simple to the complex.

Game's description

The description of BISSIM will be done in the coming pages.

BISSIM PROSPECTUS

Management Games Limited
Methwold House, Northwold Road, Methwold,
Thetford, Norfolk IP26 4PF, England.
Telephone: 0366 728215



PROSPECTUS

Directors: Pam Sykes, Philip Sykes A.C.I.S. Barrister

Regd. London 746499

The Master Game

CONCEPTS

BISSIM is the most realistic and detailed simulation of a manufacturing company ever offered to the general public. Its unique modular structure enables participants to move from the simple to the complex and from the known to the unknown. So, for such a complex simulation, it is very easy to assimilate. Adjudication is simple and speedy through use of a computer program written in "BASIC".

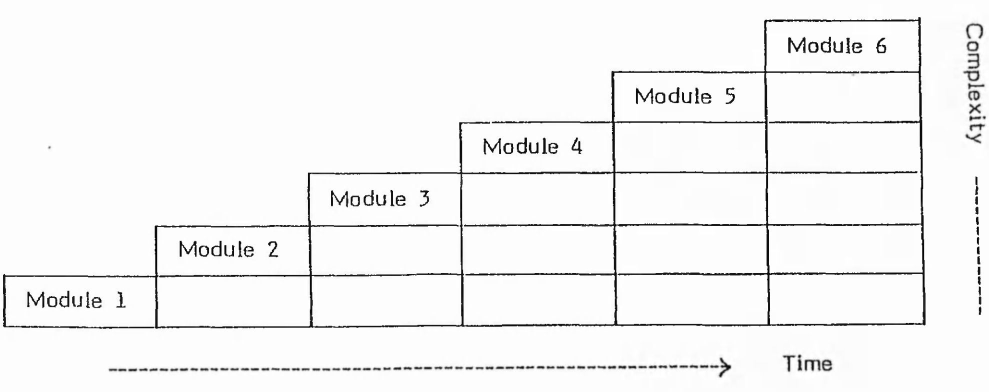
LEARNING OBJECTIVES

At the end of the simulation, participants will be able to:

* describe the relationship between supply and demand in a competitive market
* describe the relationship of investment, risk and reward (or profit)
* list the factors which can influence a company's sales and profits
* outline at least three basic business strategies
* describe the construction of the financial accounts of a company
* explain the process of business decision-making and the use of market and financial information

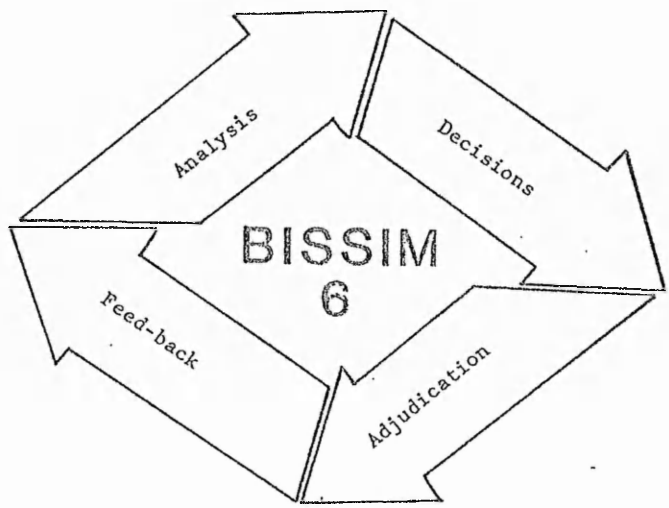
SIMULATION DESIGN

The exercise is run over 4 to 10 rounds, each representing one year's trading. In each round, each team of participants is presented with three or four new decisions in addition to their reconsideration of previous decisions. In this way, the detail is built up progressively during the exercise:



SIMULATION DESIGN (cont.)

In each round the following procedure is repeated:



1) Firstly, the situation is analysed by participants.

2) Then participants make their decisions concerning the running of their "company", and enter them on the relevant "Decision Form".

3) Those decisions are analysed by the computer model.

4) The computer prints out the financial and other results for reappraisal. A sample print-out is attached to this prospectus.

Participants do not need to make a complete set of decisions in every round. If they do nothing, the computer will interpret that as an intention to pursue the "standard" decisions which are stored in the program. The program is interactive, i.e. the decisions of one company will have an impact on its competitors.

CONTENT

BISSIM comprises six modules of increasing complexity which are cumulative in the way they are built up, each containing the following decisions:

Module One: Introduction

1. Factory Output
2. Pricing
3. Wage Rates

Module Two: Production

4. Research & development
5. Productivity improvements
6. Quality control
7. Staffing levels and overtime
8. Capital expenditure appraisal (factory extension and/or new equipment)
9. Industrial relations

Module Three: Marketing

10. Market segmentation and marketing strategies
11. Distribution channels
12. Media selection
13. Promotional expenditure

Module Four: Sales Management

14. Trade discounts
15. Salesforce size
16. Calling rates and market coverage
17. Sales training
18. Sales incentives

Moduel Five: Finance

19. Equity and loan financing
20. Share issues
21. Dividend Rates
22. Trade credit given and taken

Module Six: Customers & Society

23. After sales service (guarantees & service contracts)
24. Public relations
25. Donations
26. Pollution

CONTENT (cont.)

All modules require the analysis of financial and market research data. The interpretation of financial accounts and management information is also featured in every module. A fascinating by-product of the exercise is the light which it sheds on teamwork, leadership and other facets of human relationships.

BENEFITS

BISSIM offers the following benefits to instructors:

* "Model-based" simulations of this type have a reputation for being difficult to handle. Not so with BISSIM. This is the first such exercise which is as much fun for the instructor as for the participants!

* This is achieved through ten years of development, an unusually flexible computer program and very thorough discussion notes.

* BISSIM is flexible in its timings. A single round requires one or two hours, so as many rounds as you have time for can be worked into your programme (not necessarily consecutively). The total time occupied by the exercise can range from three to thirty hours.

* It is flexible in terms of the number of participants - from 3 to over 100 people!

* The adjudication between rounds is simple, quick and accurate (an average of 3 minutes per team is all that is required!).

BISSIM offers the following benefits to participants:

* It is an exciting, stimulating and challenging exercise (trainee motivation is never a problem).

* It is easy to get to grips with at the beginning and is "graded" (becoming more difficult as it progresses).

* It does not require high levels of numeracy to participate.

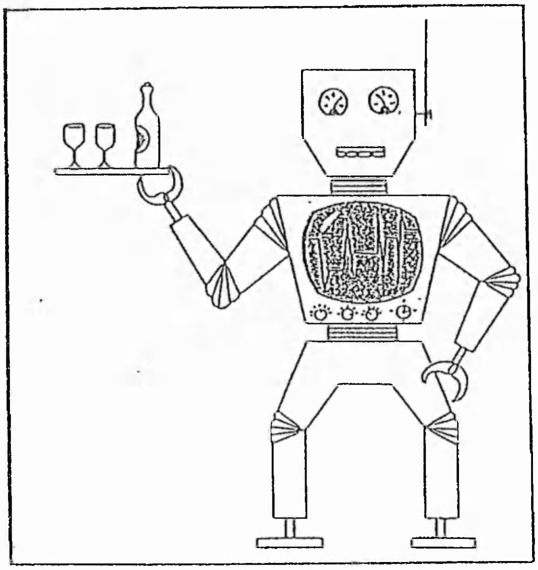
VALIDATION

BISSIM has been developed over 10 years from the practical experience gained with over 200 "live" runs of its manually operated predecessors - "Tycoon", "Finansim" and "Sellem", and from five previous editions of BISSIM itself.

The simulation is able to discriminate between teams of different abilities. Generally speaking, more able groups achieve higher profits. Users are asked to submit details of the results achieved so that a data bank of norms relating the financial results to age/salary or participants can be published. The detailed questions enable the instructor to judge the learning which has been achieved by participants.

SCENARIO

BISSIM represents a typical company manufacturing a hypothetical consumer-durable product in the year 2021. This is a DOMESTIC ROBOT of (more or less!) human appearance, which is designed to carry out a wide range of domestic chores. At the beginning of the exercise, it looks like this:



During the exercise, participants have the opportunity to develop the specification of the machine and to offer different versions of it aimed at specialised market segments.

The market in which participants operate is a typical Western European "developed" economy of about 30 million people. Over the past five years, the company has established itself in the market but its financial position is still very weak.

The sales volume of the product over that period was as follows:

2016	2017	2018	2019	2020
500	1100	1400	1600	1500

The market demand fluctuates in accordance with a 4 to 5 year economic cycle but the combined marketing effort of all competing companies will affect the size of the market to some extent.

All the competing teams are identical at the beginning of the exercise. The financial and other results which the company achieved in the year 2020 are described in the attached computer printout. The existing policies of the company can be identified from the Decision Form for 2020, which is also attached.

SIMPLIFIED EXERCISE

For those who wish to carry out financial or "business appreciation" training at a less sophisticated level, our simple computerised exercise, "TYCOON" is designed to illustrate the basic concepts of supply and demand, competition and profit or loss in a manufacturing industry. The decisions required include: pricing, market research, promotional expenditure, staffing levels, wage rates, production schedule and capital expenditure (i.e. extending factory capacity). Each team then receives a print-out of a profit and loss account and balance sheet. The exercise requires from 6 to 10 hours.

OPEN LEARNING

"BISSIM" and "TYCOON" can be run for you as a postal competition. Used this way, participants can work in teams at their own place of work and in their own time. At regular intervals (e.g. weekly or monthly) they post their decisions to MGL who process them and send back to them a printout of their results (profit and loss account, balance sheet, indices of performance, etc.). The process is repeated over 5 rounds, each of which represents one year's trading.

MODIFICATIONS

Although most of the business principles illustrated in this context apply to every business, some instructors may find that the simulation is more readily accepted by participants if the scenario is closer to their own situation. Accordingly, the publishers will modify the rules and the program to suit ANY BUSINESS. A great deal of experience has been accumulated by MGL in the design of such simulations. In fact, simulations already exist covering:

- Airlines
- Banks
- Building & Construction Companies
- Building Societies
- Computer Companies
- Consultancy-type Businesses
- Engineering Companies
- Exporters
- Food Manufacturers
- Insurance Companies (General Business and Life)
- Motor Manufacturers
- Motor Traders
- Pharmaceutical Manufacturers
- Retail Traders (Newsagencies, etc.)
- Service Businesses (e.g. Linen Supply)
- Transport Companies
- Wholesalers

Any business environment can be simulated using the BISSIM structural framework of rules and computer program. Such modifications cost less than may be imagined.

THE PACKAGE

The package comprises:

One Instructor's Manual - which provides a comprehensive guide to the use of the simulation.

Twenty Participant Manuals - containing the background, rules and decisions of the exercise. Alternatively, a copyright waiver is given.

Six Sets of Decision Forms

One Computer Program - written in "BASIC" and supplied as a disc for an IBM PC (Discs for many other machines may be supplied; please check with us for the current availability).

SUPPORT SERVICES

The publishers will provide all advice and assistance necessary to ensure that the simulation works well for you. Two days of training for your instructors OR conducting the first "live" run for you is available as part of the package.

BISSIM

DECISION FORM

Team No: Team Name: 21st CENTURY ELECTRONICS PLC. Year: 2020

DECISIONS		AMOUNT							
		Model <I>				Model <II>			
PRODUCTION DECISIONS									
Production Schedule	(Units)	1	6	0	0				0
Raw Material Quality	(1 - 4)	X	X	X	3	X	X	X	
Capital Expenditure: *									
Flow Line	(£000)								0
Flex. Manufacture	(£000)								0
R & D Key Project	(1 - 22)							1	3
R & D 2nd Project	(1 - 22)							1	4
Productivity Expenditure *	(£000)								0
Miscellaneous Expenditure * (including Market Research Donations, etc.)	(£000)								0
PERSONNEL DECISIONS									
Production - Direct								7	5
Production - Indirect (inc. Quality and Maintenance)								5	0
Sales								1	0
Research and Development								3	5
Administration (inc. Service)								3	0
Average Wage Rate	(£000)							1	0 0
Overtime Hours	(per Person per Week)								0
		Model <I>				Model <II>			
MARKETING DECISIONS									
Average Price of all Versions	(£)	5	0	0	0				0
Principal Market Segments		X	X	3	0	X	X		
Promotional Expenditure	(£000)			1	0				
Sales Incentives	(£000)			1	0				
Price (Contract)	(£)				0				
Training Expenditure	(£000)				1				
Key Outlets	(1 - 7)	X	X	X	4	X	X	X	
Trade Discount	(%)	X	X	2	0	X	X		
Guarantee Period	(Months)	X	X	0	6	X	X		
R & D Projects Allocated					0				
No. of Versions		X	X		1	X	X		

Form 2

B I S S I M

DECISION FORM (Cont.)

Team No: Team Name: 21st CENTURY ELECTRONICS PLC. Year: 2020

DECISIONS		AMOUNT			
FINANCE DECISIONS					
Shares Offered *	(000's)				0
Offer Price *	(£)				0
Dividend	(%)			1	0
Debtor Period (Customers)	(Weeks)			0	4
Creditor Period (Suppliers)	(Weeks)			1	2
Overdraft Application	(£000)	1	0	0	0
FOR ADJUDICATOR'S USE ONLY					
Marketing Index					
Staff Morale					
Productivity Index					
Quality Index					
Credit Rating					
Interest Rate					
Social Index					
Sales Management					
Cash Flow	(+/- £000)				
Government Contract Volume	(Units)				
Government Contract Product	(I or II)				

BUDGET FORM					
Forecast of Net Profit (Pre Tax) *	(£000)				
Forecast of Bank Overdraft or Cash Balance *	(+/- £000)				

Write on this form only those decisions which you wish to CHANGE. Otherwise, the computer program assumes the continuance of present policies (except for items marked with an asterisk which are of a "one-off" nature).

Form 2 (Cont.)

21st. CENTURY ELECTRONICS Plc.

R E S U L T S YEAR 2020

****

PROFIT & LOSS ACCOUNT

	#000'S	#000'S
SALES TURNOVER		6028
(after discount 1507)		
(incl. interest 0)		
EXPENSES:		
Cost of sales	2805	
Prodn. labour	1250	
Marketing	121	
Research	350	
Guarantee costs	340	
Admin & service	300	
Financial	49	
Distribution	177	
Overhead	995	
TOTAL EXPENSES		6387
NET PROFIT		-359
Taxation		0
PROFIT AFTER TAX		-359
Dividends		0
RETAINED PROFIT		-359

BALANCE SHEET

	#000'S	#000'S
SHARE CAPITAL		100
RESERVES		2332
CAP. & RESVS.		2432
FIXED ASSETS		2430
TRADE INVESTMENTS		0
CURRENT ASSETS:		
Stock	854	
Debtors	772	
Cash	0	
Total	1626	
CUR. LIABILITIES:		
Creditors	792	
Overdraft	832	
Total	1624	
NET CUR. ASSETS		2
NET ASSETS		2432

KEY RATIOS:

NET MARGIN	-6.0	%
GROSS MARGIN	54	%
R. O. C. E.	-14.8	%
MARKET SHARE		%
OUTPUT, PER PERS.	7.7	units
LIQUIDITY RATIO	47	%
STOCK TURN	6	times p.a.
EARNINGS P. SHARE	0	p.

INDICES:

MARKET IMAGE	5
PRODUCT QUALITY	3
STAFF MORALE	3
CREDIT RATING	3
PUBLIC IMAGE	3
SALES EFFICIENCY	1
AFTER-SALES SERVICE	0
VALUE FOR MONEY	10

PRODUCTION & FINANCE:

FACTORY PROD. <I>	1550	units
<II>	0	units
UNSOLD STOCK <I>	300	units
UNSOLD STOCK<II>	0	units
UNIT PROD COST<I>	1887	(#)
UNIT PRODCOST<II>	0	(#)
CAPACITY UTILN.	100	%
FUTURE CAPACITY	1550	units
SHARE PRICE	# 26.75	
R & D KEY PROJECT	13 = 95	yrs.
R & D 2nd PROJECT	14 = 1	yrs.
COMPLETED R & D		

MARKETING & PERSONNEL:

SALES <I>	1507	units
SALES <II>	0	units
LOST SALES <I>	0	units
LOST SALES <II>	0	units
CUSTMS. & PROSPTS.	1992	
CALLING RATE	600	pp pa
DIRECT PRODN. STAFF	75	
INDCT. PRODN. STAFF	50	
SALES STAFF	9	
R. $ D. STAFF	33	
ADMIN. STAFF	30	
SEGMENTS	30 14	%

****

CONCLUSION

The workshop of Management Simulations held at the Management Centre, University of Bradford, was a very interesting experienced.

Its interest was based on the possibility to see four of the most successful simulation-games.

Among the simulations shown, Network Proteus was, in my opinion, the most interesting one. The reasons for that are :

♦ Network Proteus was shown as a non-numerical based simulation (modern approach), where the main issue to experience was the existing departmental interaction. It was not, as the other three, a mere calculation game (traditional approach).

♦ Network Proteus is a dynamic interactive simulation, where interactions among roles in a factory take place while running the simulation, not before (i.e. fill in a form or enter information on a computer).

♦ Network Proteus is a "real" simulation, where the product being produced has a real bill of materials, its produced in different machines for "real" known customers, and in a real manufacturing market.

♦ Network Proteus is able to generate "real" problems, that the different teams will have to face.

There are also some disadvantages associated :

♦ Network Proteus has a very complicate interrelation structure, there two many people to contact.

♦ Network Proteus has a too long cycle time, something that leads to a very long teaching/training process. Something not many courses can afford.

♦ Although it is said to work in a real time environment were people can make decisions or not, that methodology is virtually the same as changing or not changing a decision - a capacity provided in virtually all games [77].

Other contacts

The next pages show some of the contacts held during the fulfilment of this project.

OTHER CONTACTS

Managment Workshop Programme,
Management Centre,
Emm Lane,
Heaton,
Bradford,
BD9 4JL.

Jaun Ignacio Igartua,
Nottingham Polytechnic,
Dept. Industrial &
Production Engineering,
Burton Street,
Nottingham,
NG1 4BU.

17th. January 1991

Dear Jaun,

Referring to your letter of 9th. January. Since the publication of the article in *"Perspectives on gaming and simulation 14"*, a new BUSGAM version 2 has been completed and is in use at the Management Centre on undergraduate Business Studies and Electrical Engineering, postgraduate MBA and Manufacturing Technology, and Executive Development courses. The new game is multi-product / market segment and adaptable for a number of different case studies.

In response to your questions, this type of game rapidly becomes technically difficult to design and develop once one wishes to do anything non-trivial, as the problem is to implement a real-time multi-user database, with considerations of response, locking, etc. On our network (286 PCs, 386 Server, MSNET, EtherNet) performance becomes unacceptably slow with around eight teams. Having said that, we can still put 50 or more participants through at one session (with one trainer) which is extremely resource effective compared to conventional games.

Evaluation of the game through participant questionnaire surveys, observers, etc. has lead us to conclude that generally students prefer BUSGAM to other business games they have taken part in, although this may of course, be because of its novelty value. The time dimension, with the relentless increasing clock is a strong motivating factor to maintain interest and heighten enjoyment, but we suspect this can lead to reactive behaviour; i.e. responding to events without planning. One of the key objectives of education is to enable the student to become proactive, aware and in control. Hence we now incorporate a practice session before the game, and also split up the sessions to allow reflection; i.e. thinking as well as doing. Careful working through of learning objectives is very important.

As BUSGAM is a strategic level business game and each team / company has its own computer, you may find that your plans have more in common with other games. For example, Network Proteus at Manchester Business School is being developed and will be a re-write of their mainframe version. There are four roles for participants to run a company: Production, Finance, Marketing and Managing Director, plus a trainer's interface. Another contact may be the Production and Operations Management group, here at the Management Centre. My colleague Malcolm Afferson has worked on the development of a computerised Production Management system and associated training package for local companies, funded by the Training Agency.

I hope that I have been of help.

Yours sincerely,

C.L.N. Gray

(Mr. Carey Gray, Enterprise Initiative Fellow)

UNIVERSITY OF BRADFORD
MANAGEMENT CENTRE EMM LANE BRADFORD WEST YORKSHIRE BD9 4JL UK
☎ (0274) 542299
Telex 51309 UNIBFD G
Fax (0274) 546866
Director and Professor of Management DAVID T H WEIR MA DipPSA FBIM
E-mail ghictiaUK.AC,BRADFORD

COMPUTERS IN TEACHING INITIATIVE CENTRE FOR BUSINESS AND MANAGEMENT STUDIES
Coordinator: RICHARD WELFORD BA MA PhD PGCE ABIM

1 February 1991

Mr J I Igartua
Nottingham Polytechnic
Industrial & Production Engineering Dept
Burton Street
NG1 4BU

Dear Mr Igartua

Workshop on Management Simulations - 6th February 1991

I am pleased to be able to confirm your place at the above workshop has been reserved.

I enclose a map showing the location of the Management Centre. The workshop will be held in the Heaton Mount building, which is accessed via Keighley Road.

The timing of some of some of the sessions has had to be amended slightly, which will probably mean finishing the day approximately half an hour later than publicised, and I trust this will not inconvenience you too much.

I look forward to seeing you on the 6th.

Yours sincerely

G C Holmes

Cillian Holmes
CTI Information Officer

Sponsored by the Institute of Chartered Accountants in England and Wales
and the Computer Board for Universities and Research Councils

APPENDIX C

GAME'S SOFTWARE

Factory Running Sub-program » SYS11.PRG

Production Planning/Control Sub-program » PLA4.PRG

Marketing/Sales Sub-program » SAL6.PRG

FACTORY RUNNING SUB-PROGRAM

SYS11.PRG

```
*:*************************************************************************
*:                          This program is the one that simulates
*:      Program: SYS11.PRG     all the factory activities.Together with
*:                          the Sales and Planning ones conforms the
*:       System: SYS11.PRG                Nottingham Polytechnics's
*:       Author: JUAN IGNACIO IGARTUA              gaming-simulation.
*:    Copyright (c) 1991, NOTTINGHAM POLYTECHNIC
*: Last modified: 08/05/91      17:36
*:
*: Procs & Fncts: ERR_FIX
*:              : GAMCONF
*:              : OPEN
*:              : MAINMENU
*:              : FACTCONF
*:              : STARTGAM
*:              : RESET
*:              : SET
*:              : SETNEXT
*:              : NEXTDAY
*:              : EXITGAME
*:              : QUITGAME
*:              : CONTINUE
*:              : RUNFACT
*:              : FACTSTOP
*:              : SAVEDAY
*:              : RESTART
*:              : DAYREPOR
*:              : SALREPO
*:              : PLAREPO
*:              : RND
*:              : GOODSERV
*:              : RUNSCREN
*:              : SAYTIME0
*:              : SAYTIME1
*:              : TIMECAL
*:              : RUNPROD
*:              : ENQGEN
*:              : SHIPPING
*:              : REPOMENU
*:              : FREQUEN
*:              : SALPAST
*:              : SALPERF
*:              : PLAPAST
*:              : MACHBROW
*:              : MACH_DISP
*:              : NEWMACH
*:              : MACHEDIT
*:              : MACHGETS
*:              : INCLUDE
*:              : MACH_REPL
*:              : MACH_RSTR
*:              : MACH_STOR
*:              : REINDEX
*:              : OPENRUN
*:              : OPENSAL
*:              : OPENPLA
*:              : SYSTEM
*:              : GETKEY
*:              : EXITSYS
*:              : FIELDD
*:              : RESETUP
*:              : ONTIME
*:              : ENQUSET
*:              : ADDREC
*:              : RESTVAR
*:              : POSIT
*:              : CHEKFILE
*:              : CREATSTAT
*:              : RENSTAT
*:              : REPLSYS
*:              : CREATSTOR
*:              : CREATREPL
*:              : DELETC
*:
*:         Calls: ERROR()        (FOXBASE+ function)
*:              : MESSAGE()      (FOXBASE+ function)
*:              : ERR_FIX        (procedure in SYS11.PRG)
*:              : CTOD()         (FOXBASE+ function)
*:              : GAMCONF        (procedure in SYS11.PRG)
*:              : OPEN           (procedure in SYS11.PRG)
*:              : MAINMENU       (procedure in SYS11.PRG)
*:
*:          Uses: SYSTEM.DBF
*:
```

```
*:      Documented 08/05/91 at 18:22             FoxDoc  version 1.0
*:******************************************************************************
    **
    set escape on
    set talk off
    set heading off
    set bell off
    set scoreboard off
    set status off
    set color to
    set delete on
    clear
    **
    set color to bu+/n
    @ 7,24 to 12,54 double
    set color to gr+/n
    @ 9,30 say "SETTING THE SYSTEM"
    @ 10,34 say "PLEASE WAIT"
    **
    set date british
    set procedure to sys11.prg
    on error do err_fix with error(),message()
    set default to v:
    **
    public mfound,mresourcod,mdescript,mcapacity,msetup1,msetup2,msetup3,;
    msetup4,msetup5,srnlines,difclock,prtlines,pout,factclock,factclock1
    public public facthh,factdd,factmm,factyy,produce,ctdel,confdate,seed,;
    factmin,factstart,tempdd,startstop,mmfacttime,mmfactdate,number,newclock
    public difstop,slack,plow,tempdd1,facthhh,reso,actsetup,lstsetup;
    r,c,screenatr,statusatr,windowatr,promptatr,hiliteatr,factdaten,daysenqu
    public startday,startmon,startyy,enqurec,conffdate,mwaittime,oldrec,toprec,;
    botrec,tt,xx,mpla,mmar,mmac,mstk,mbil,mstu,msta,msalp,msalf,mstkp,mmacp,mplap
    public msys,pfiles,first,mloop,var,fil
    **
    screenatr="R+/N,N/W"
    statusatr="BU/N,N/W"
    windowatr="R+/N,N/W"
    promptatr="GR+/N,N/W"
    hiliteatr="N/W"
    **
    mloop=0
    var=""
    **
    srnlines=20
    prtlines=55
    pout=srnlines
    seed=0
    daysenqu=0
    oldrec=1
    botrec=1
    factdaten=ctod("00/00/00")
    first=.t.
    **
    facthh=0
    *mwaittime=0
    select 10
    set exclusive off
    use system
    **
    do gamconf
    **
    do open
    **
    close databases
    set view to run
    **
    select 10
    use system
    replace gam_stop with .f.
    unlock
    **
    do mainmenu
    **
*!******************************************************************************
*!
*!     Procedure: MAINMENU  This procedure shows the available options
*!                          when starting the program.
*!     Called by: SYS11.PRG
*!
*!         Calls: FACTCONF       (procedure in SYS11.PRG)
*!              : STARTGAM       (procedure in SYS11.PRG)
*!              : EXITSYS        (procedure in SYS11.PRG)
*!
*!******************************************************************************
    procedure mainmenu
```

```
******************
set color to
clear
do while .t.
    clear
    m_menu=0
    set color to g/n
    @ 3,16,19,62 box " ┌─┐ |┘═└| "
    set color to gr/n
    @ 5,26 to 7,53 double
    set color to bu/n
    @ 6,29 say "SYSTEM GAME WORKAREA"
    set color to br/n
    @ 10,24 to 16,54
    set color to bg/n
    @ 12,27 prompt "1.FACTORY CONFIGURATION."
    @ 13,27 prompt "2.RUN THE GAME.          "
    @ 14,27 prompt "3.EXIT FROM SYSTEM.      "
    menu to m_menu
    do case
    case m_menu=1
        do factconf
    case m_menu=2
        do startgam
    case m_menu=3
        do exitsys
v===============exit
    endcase
enddo
 *
*!*********************************************************************
*!
*!     Procedure: FACTCONF This procedure allows the game manager
*!                         modifying the machines' capacity and setup.
*!     Called by: MAINMENU       (procedure in SYS11.PRG)
*!
*!        Calls: MACHBROW        (procedure in SYS11.PRG)
*!
*!*********************************************************************
procedure factconf
******************
do while .t.
    clear
    m_menu=0
    set color to g/n
    @ 3,16,17,62 box " ┌─┐ |┘═└| "
    set color to gr/n
    @ 5,26 to 7,53 double
    set color to bu/n
    @  6,28 say " FACTORY CONFIGURATION"
    set color to br/n
    @ 10,24 to 16,54
    set color to bg/n
    @ 12,29 prompt "1.MACHINE FILE."
    @ 13,29 prompt "2.EXIT.        "
    menu to m_menu
    do case
    case m_menu=1
        do machbrow
    case m_menu=2
<===================return
    endcase
enddo
 *
*!*********************************************************************
*!
*!     Procedure: STARTGAM    This procedure allows the system manager
*!                            to start running the gaming-simulation.
*!     Called by: MAINMENU       (procedure in SYS11.PRG)
*!
*!        Calls: CONTINUE        (procedure in SYS11.PRG)
*!             : INT()          (FOXBASE+ function)
*!             : VAL()          (FOXBASE+ function)
*!             : SUBSTR()       (FOXBASE+ function)
*!             : DTOC()         (FOXBASE+ function)
*!             : LEFT()         (FOXBASE+ function)
*!             : RIGHT()        (FOXBASE+ function)
*!             : RUNFACT        (procedure in SYS11.PRG)
*!
*!*********************************************************************
procedure startgam
******************
do while .t.
    clear
    m_menu=0
```

```
        set color to g/n
        @ 3,16,17,62 box "╒═╕│╛═╘│"
        set color to gr/n
        @ 5,26 to 7,53 double
        set color to bu/n
        @  6,33 say "RUN THE GAME"
        set color to br/n
        @ 10,24 to 16,54
        set color to bg/n
        @ 12,29 prompt "1.RUN THE GAME.      "
        @ 13,29 prompt "2.EXIT BACK TO MAIN."
        menu to m_menu
        do case
        case m_menu=1
          do continue
          store int(val(substr(dtoc(confdate),4,2))) to startmon
          store int(val(left(dtoc(confdate),2)))     to startday
          store int(val(right(dtoc(confdate),2)))    to startyy
          do runfact
        case m_menu=2
                       return
        endcase
      enddo
    **
*!*************************************************************************
*!
*!      Procedure: RESET  This procedure resets the gaming-simulation's
*!                     time/date and production variables.
*!      Called by: GAMCONF        (procedure in SYS11.PRG)
*!
*!          Calls: INT()          (FOXBASE+ function)
*!               : VAL()          (FOXBASE+ function)
*!               : SUBSTR()       (FOXBASE+ function)
*!               : DTOC()         (FOXBASE+ function)
*!               : LEFT()         (FOXBASE+ function)
*!               : RIGHT()        (FOXBASE+ function)
*!
*!*************************************************************************
    procedure reset
    ***************
    store 0 to factmin,newclock,factclock1,factclock,facthh,difclock
    store 0 to difstop,startstop,produce,tempdd,tempdd1,daysenqu
    store 1 to plow
    store j->frequency to slack
    store int(val(substr(dtoc(j->factdate),4,2))) to factmm
    store int(val(left(dtoc(j->factdate),2)))     to factdd
    store int(val(right(dtoc(j->factdate),2)))    to factyy
    return
    **
*!*************************************************************************
*!
*!      Procedure: SET  This procedure resets the gaming-simualtion's
*!                    running elements when the game is exit or quit.
*!      Called by: EXITGAME       (procedure in SYS11.PRG)
*!               : QUITGAME       (procedure in SYS11.PRG)
*!               : EXITSYS        (procedure in SYS11.PRG)
*!
*!          Calls: SPACE()        (FOXBASE+ function)
*!               : EOF()          (FOXBASE+ function)
*!               : .NOT.EOF()     (FOXBASE+ function)
*!
*!*************************************************************************
    procedure set
    *************
    clear
    set color to bu+/n
    @ 8,24 to 12,54 double
    set color to gr+/n
    @ 10,29 say "THE GAME HAS FINISHED"
    **
    select 10
    replace gam_stop  with .t.
    replace sys_stop  with .t.
    replace salreport with .f.
    replace plareport with .f.
    replace newenq    with .f.
    replace newenqrec with space(9)
    replace goodserv  with .f.
    replace facttime  with "0:00"
    replace reindex   with .t.
    replace restart   with .f.
    replace restatus  with .f.
    replace statreso  with "----"
    replace statcode  with "----"
    replace waittime  with (factdate-startdate)*480
```

```
factdaten=factdate
replace factdate with factdaten
replace throughput with 0
replace systock    with 0
unlock
select 1
goto top
do while .not. eof()
   replace ord_stat_0 with ord_stat_1
   replace product_0 with product_0+product_1
   if ord_stat_1="FINISHED"
     replace product_1 with product_0
   endif
   unlock
   if .not.eof()
     skip
   endif
enddo
select 3
replace all t_avail with 0
unlock
select 5
replace all num_enqu with 0
unlock
return
**
*!*************************************************************************
*!                      This procedure sets the gaming-simulation's
*!    Procedure: SETNEXT  running elements in order to start the next
*!                      running day.
*!    Called by: NEXTDAY        (procedure in SYS11.PRG)
*!
*!       Calls: EOF()          (FOXBASE+ function)
*!            : .NOT.EOF()      (FOXBASE+ function)
*!            : SPACE()        (FOXBASE+ function)
*!
*!*************************************************************************
procedure setnext
*****************
clear
select 1
goto top
do while .not. eof()
   replace ord_stat_0 with ord_stat_1
   replace product_0 with product_0+product_1
   if ord_stat_1="FINISHED"
     replace product_1 with product_0
   endif
   unlock
   if .not.eof()
     skip
   endif
enddo
select 3
replace all t_avail with 0
unlock
select 5
replace all num_enqu with 0
unlock
select 10
replace sys_stop  with .f.
replace salreport with .f.
replace plareport with .f.
replace newenq    with .f.
replace newenqrec with space(9)
replace goodserv  with .f.
replace facttime  with "0:00"
replace reindex   with .t.
replace restart   with .f.
replace restatus  with .f.
replace statreso  with "----"
replace statcode  with "----"
replace waittime  with (factdate-startdate)*480
factdaten=factdate
replace factdate with factdaten
replace throughput with 0
replace systock    with 0
unlock
select 1
set color to
clear
return
**
*!*************************************************************************
*!
```

```
*!    Procedure: NEXTDAY   This procedure calls the procedures needed
*!                         to run the next running day.
*!    Called by: REPOMENU       (procedure in SYS11.PRG)
*!
*!        Calls: GAMCONF        (procedure in SYS11.PRG)
*!             : SETNEXT        (procedure in SYS11.PRG)
*!
*!*********************************************************************
procedure nextday
*****************
do gamconf
do setnext
return
**
*!*********************************************************************
*!                         This procedure allows the system manager to
*!    Procedure: EXITGAME exit the gaming-simulation once a day has
*!                         finished.
*!    Called by: REPOMENU       (procedure in SYS11.PRG)
*!
*!        Calls: SET            (procedure in SYS11.PRG)
*!
*!*********************************************************************
procedure exitgame
******************
do set
close all
clear all
return to master
**
*!*********************************************************************
*!                         This procedure allows the system manager
*!    Procedure: QUITGAME  to quit the gaming-simulation after it has
*!                         been frozen.
*!    Called by: FACTSTOP       (procedure in SYS11.PRG)
*!
*!        Calls: SET            (procedure in SYS11.PRG)
*!
*!*********************************************************************
procedure quitgame
******************
do set
close all
clear all
return to master
**
*!*********************************************************************
*!                         This procedure allows the system manager
*!    Procedure: CONTINUE  to continue with the gaming-simulation after
*!                         it has been frozen.
*!    Called by: STARTGAM       (procedure in SYS11.PRG)
*!             : FACTSTOP       (procedure in SYS11.PRG)
*!
*!*********************************************************************
procedure continue
******************
select 10
replace sys_stop with .f.
unlock
set color to
clear
return
**
*!*********************************************************************
*!                         This is the main "running" procedure, which controls
*!    Procedure: RUNFACT   the different tasks of the gaming-simulation. Its functioning
*!                         is thorougly explained in one of the thesis's sections.
*!    Called by: STARTGAM       (procedure in SYS11.PRG)
*!
*!        Calls: RUNSCREN       (procedure in SYS11.PRG)
*!             : SAYTIME0       (procedure in SYS11.PRG)
*!             : VAL()          (FOXBASE+ function)
*!             : SYS()          (FOXBASE+ function)
*!             : INT()          (FOXBASE+ function)
*!             : LTRIM()        (FOXBASE+ function)
*!             : STR()          (FOXBASE+ function)
*!             : TIMECAL        (procedure in SYS11.PRG)
*!             : RUNPROD        (procedure in SYS11.PRG)
*!             : GOODSERV       (procedure in SYS11.PRG)
*!             : SHIPPING       (procedure in SYS11.PRG)
*!             : RENSTAT        (procedure in SYS11.PRG)
*!             : DAYREPOR       (procedure in SYS11.PRG)
*!
*!*********************************************************************
procedure runfact
```

```
*****************
clear
do runscren
do saytime0
store val(sys(2)) to factstart
do while .t.
    newclock=int((val(sys(2))-factstart-difstop)*(160/60))
    if newclock>factclock1
        var="LOTMS"+ltrim(str(mloop))+ltrim(str(tempdd))
        public &var
        &var=val(sys(2))
        do timecal
        var="LOTMF"+ltrim(str(mloop))+ltrim(str(tempdd))
        public &var
        &var=val(sys(2))
        if tempdd1=tempdd
            do saytime0
        endif
        var="LOPDS"+ltrim(str(mloop))+ltrim(str(tempdd))
        public &var
        &var=val(sys(2))
        do runprod
        var="LOPDF"+ltrim(str(mloop))+ltrim(str(tempdd))
        public &var
        &var=val(sys(2))
        mloop=mloop+1
        if tempdd1>tempdd
            set color to gr*+/bu
            @ 10,57 say "8:00"
            set color to
            select 10
            replace sys_stop with .t.
            unlock
            var="LOGDS"+ltrim(str(tempdd))
            public &var
            &var=val(sys(2))
            do goodserv
            var="LOGDF"+ltrim(str(tempdd))
            public &var
            &var=val(sys(2))
            var="LOSPS"+ltrim(str(tempdd))
            public &var
            &var=val(sys(2))
            do shipping
            var="LOSPF"+ltrim(str(tempdd))
            public &var
            &var=val(sys(2))
            do renstat
            **
            set color to
            clear
            set color to bu+/n
            @ 8,24 to 12,54 double
            set color to gr+/n
            @ 10,27 say "GENERATING THE PAST FILES"
            **
            do dayrepor
            do runscren
            store val(sys(2)) to factstart
        endif
    endif
enddo
**
*!***************************************************************************
*!                     This procedure is called when the gaming-simulation
*!    Procedure: FACTSTOP  is frozen by the game manager, giving him two choices
*!                     (quit or continue with the game).
*!    Called by: SAYTIME0        (procedure in SYS11.PRG)
*!
*!        Calls: CONTINUE        (procedure in SYS11.PRG)
*!             : QUITGAME        (procedure in SYS11.PRG)
*!
*!***************************************************************************
procedure factstop
*****************
select 10
replace sys_stop with .t.
unlock
**
**
** SEND THIS SIGNAL TO THE DIFFERENT DEPARTMENTS.IF IT IS .T. IN EVERY
** COMPUTER OF THE GAME A BLINKING MENU WILL APPEAR SAYING THAT THE GAME
** IS TEMPORALY FROZEN IN ORDER TO DISPLAY THE DAY'S RESULTS.ALSO ALL THE
** DATABASES WILL BE CLOSED.
**
```

```
**
** LOCK THE DEPARTMENTS DATA INPUT IN ORDER NOT TO LOSE THE TIME/DATE
** SEQUENCE.
**
**
set color to
clear
do while .t.
   clear
   m_menu=0
   set color to g/n
   @ 3,16,17,62 box "╒═╕│╛═╘│"
   set color to gr/n
   @ 5,26 to 7,53 double
   set color to bu/n
   @  6,33 say "GAME FROZEN"
   set color to br/n
   @ 10,24 to 16,54
   set color to bu/bg
   @ 20,5 say "THE GAME HAS BEEN FROZEN.SELECT FROM MENU."
   set color to bg/n
   @ 12,32 prompt "1.CONTINUE GAME."
   @ 13,32 prompt "2.QUIT GAME.      "
   menu to m_menu
   do case
   case m_menu=1
      do continue
      restore screen from runscr
      exit
   case m_menu=2
      do quitgame
   endcase
enddo
return
**
*!*****************************************************************************
*!                          This procedure allows to save the data of the just
*!     Procedure: SAVEDAY   finished day to files, in order to be restored if
*!                          wanted.
*!     Called by: REPOMENU        (procedure in SYS11.PRG)
*!
*!        Calls: RESTVAR          (procedure in SYS11.PRG)
*!
*!         Uses: TEMP.DBF
*!
*!*****************************************************************************
procedure saveday
*****************
clear
set color to gr+/bu
@ 5,6 to 18,72 double
set color to g+/n
@ 6,24 to 8,54
set color to r*+/n
@ 7,31 say "SAVE DAY FOR RESTART."
set color to bg+/n
@ 11,21 say "THE DATE TO SAVE FROM IS:"
?? (mmfactdate-1)
set color to ,n/w
do restvar with (mmfactdate-1)
close databases
copy file system.dbf   to temp.dbf
select 10
use temp
copy file system.dbf   to &msys
copy file plann.dbf    to &mpla
copy file market.dbf   to &mmar
copy file machine.dbf  to &mmac
copy file stock.dbf    to &mstk
copy file bill.dbf     to &mbil
copy file setup.dbf    to &mstu
copy file status.dbf   to &msta
copy file salpast.dbf  to &msalp
copy file salperf.dbf  to &msalf
copy file stkpast.dbf  to &mstkp
copy file machpast.dbf to &mmacp
copy file plapast.dbf  to &mplap
set view to run
return
**
*!*****************************************************************************
*!                          This procedure allows the game manager to restore
*!     Procedure: RESTART   the data previously saved, in order to run the next
*!                          day with that data.
*!     Called by: GAMCONF         (procedure in SYS11.PRG)
```

```
*!
*!          Calls: DTOC()          (FOXBASE+ function)
*!               : REPLSYS         (procedure in SYS11.PRG)
*!
*!***********************************************************************
     procedure restart
     *****************
     clear
     set color to gr+/bu
     @ 5,6 to 18,72 double
     set color to g+/n
     @ 6,24 to 8,54
     set color to r*+/n
     @ 7,28 say "RESTART DAY FOR NEW RUN."
     set color to bg+/n
     @ 11,18 say "THE DATE TO RESTART FROM IS :"
     ?? dtoc(mmfactdate)
     set color to ,n/w
     select 10
     replace restart with .t.
     unlock
     do replsys
     close databases
     erase plann.dbf
     copy file &mpla to plann.dbf
     erase market.dbf
     copy file &mmar to market.dbf
     erase machine.dbf
     copy file &mmac to machine.dbf
     erase stock.dbf
     copy file &mstk to stock.dbf
     erase bill.dbf
     copy file &mbil to bill.dbf
     erase setup.dbf
     copy file &mstu to setup.dbf
     erase status.dbf
     copy file &msta to status.dbf
     erase salpast.dbf
     copy file &msalp to salpast.dbf
     erase salperf.dbf
     copy file &msalf to salperf.dbf
     erase stkpast.dbf
     copy file &mstkp to stkpast.dbf
     erase machpast.dbf
     copy file &mmacp to machpast.dbf
     erase plapast.dbf
     copy file &mplap to plapast.dbf
     return
     **
*!***********************************************************************
*!                     This procedure calls the procedures that will create
*!      Procedure: DAYREPOR and store in the report datafiles, the report variables
*!                     to be shown in the plannig department.
*!      Called by: RUNFACT         (procedure in SYS11.PRG)
*!
*!          Calls: LTRIM()         (FOXBASE+ function)
*!               : STR()           (FOXBASE+ function)
*!               : VAL()           (FOXBASE+ function)
*!               : SYS()           (FOXBASE+ function)
*!               : SALREPO         (procedure in SYS11.PRG)
*!               : PLAREPO         (procedure in SYS11.PRG)
*!               : SYSTEM          (procedure in SYS11.PRG)
*!               : .OR.SYSTEM()    (FOXBASE+ function)
*!               : REPOMENU        (procedure in SYS11.PRG)
*!
*!***********************************************************************
     procedure dayrepor
     ******************
     private scr
     scr=.t.
     var="LOSRS"+ltrim(str(tempdd))
     public &var
     &var=val(sys(2))
     do salrepo
     var="LOSRF"+ltrim(str(tempdd))
     public &var
     &var=val(sys(2))
     var="LOPRS"+ltrim(str(tempdd))
     public &var
     &var=val(sys(2))
     do plarepo
     var="LOPRF"+ltrim(str(tempdd))
     public &var
     &var=val(sys(2))
     select 10
```

```
replace salreport with .t.
replace plareport with .t.
unlock
? throughput,systock,tot_overh,assets
do while system("salreport").or.system("plareport")
    if scr
        clear
        set color to gr+/bu
        @ 7,17 clear to 15,61
        @ 7,17 to 15,61 double
        set color to r*+/bu
        @ 11,24 say "Please wait to finish the reports"
        @ 12,26 say "In the Departmental areas."
        scr=.f.
    endif
enddo
do repomenu
return
**
*!*****************************************************************************
*!                      This procedure calls the procedure that generates and
*!    Procedure: SALREPO  stores the report variables concerning the sales
*!                      department.
*!    Called by: DAYREPOR       (procedure in SYS11.PRG)
*!
*!        Calls: SALPAST        (procedure in SYS11.PRG)
*!             : ONTIME         (procedure in SYS11.PRG)
*!             : FREQUEN        (procedure in SYS11.PRG)
*!
*!*****************************************************************************
procedure salrepo
*****************
set view to sales
do salpast
do ontime
do frequen
return
**
*!*****************************************************************************
*!                      This procedure calls the procedures that generate and
*!    Procedure: PLAREPO  store the report variables concerning the planning department.
*!
*!    Called by: DAYREPOR       (procedure in SYS11.PRG)
*!
*!        Calls: PLAPAST        (procedure in SYS11.PRG)
*!
*!*****************************************************************************
procedure plarepo
*****************
set view to plann
do plapast
return
**
*!*****************************************************************************
*!                      This procedure manages the possible problems that could
*!    Procedure: ERR_FIX  arise from the fact that we are running in a multi-user
*!                      environment.
*!    Called by: SYS11.PRG
*!
*!        Calls: RLOCK()        (FOXBASE+ function)
*!
*!*****************************************************************************
procedure err_fix
*****************
parameters errnum,mess
**
** Error: File in use by another.
if errnum=108
    save screen to screen
    set color to gr+/bu
    @ 7,17 clear to 15,61
    @ 7,17 to 15,61 double
    set color to r*+/bu
    @ 11,24 say "Please wait to append a record."
    @ 12,26 say "Press any key to continue."
    read
    restore screen from screen
    retry
endif
**
** Error: Record in use by another.
if errnum=109
    save screen to screen
    time=0
    set color to gr+/bu
```

```
    @ 7,17 clear to 15,61
    @ 7,17 to 15,61 double
    set color to r*+/bu
    @ 11,26 clear to 11,54
    @ 11,26 say mess
    @ 12,26 say "Press any key to continue."
    read
    do while .not. rlock().and.time<1000
      time=time+1
    enddo
    if time<1000
      restore screen from screen
      retry
    else
      set color to gr+/bu
      @ 7,17 clear to 15,61
      @ 7,17 to 15,61 double
      set color to r*+/bu
      @ 11,19 say "Record cannot be locked.Try again later."
      @ 12,26 say "Press any key to continue."
      read
      restore screen from screen
    endif
  endif
**
** Error: Record is not locked.
  if errnum=130
    save screen to screen
    time=0
    do while .not. rlock().and.time<1000
      time=time+1
    enddo
    if time<1000
      restore screen from screen
      retry
    else
      set color to gr+/bu
      @ 7,17 clear to 15,61
      @ 7,17 to 15,61 double
      set color to r*+/bu
      @ 11,19 say "Record cannot be locked.Try again later."
      @ 12,26 say "Press any key to continue."
      read
      restore screen from screen
    endif
  endif
*
*!*************************************************************************
*!                  This procedure calculates pseudo-random numbers.
*!    Procedure: RND
*!
*!    Called by: ENQGEN       (procedure in SYS11.PRG)
*!
*!        Calls: MOD()        (FOXBASE+ function)
*!
*!*************************************************************************
procedure rnd
*************
store mod(seed*7137421+21132487,10000000)to seed
return seed/10000000
**
*!*************************************************************************
*!                  This procedure calculates the sales department's
*!    Procedure: GOODSERV quotation performance and its fail or success. Its
*!                  functioning is thorougly explained in one of the thesis's sections.
*!    Called by: RUNFACT      (procedure in SYS11.PRG)
*!
*!        Calls: EOF()        (FOXBASE+ function)
*!             : QUDELIVER>ENDEL(FOXBASE+ function)
*!
*!*************************************************************************
procedure goodserv
******************
select 10
replace goodserv with .t.
unlock
select 2
goto top
do while .not. eof()
  if status='QUOT'
    if quprice<=e->standprice
      p=100
    else
      if quprice>e->standprice.and.quprice<e->price_limt
        p=((quprice-e->price_limt)/(e->standprice-e->price_limt))*100
```

```
            else
              p=0
            endif
          endif
          dis=100
          if qudeliver<=endeliver
            d=100
          else
            if qudeliver>endeliver.and.qudeliver<(endeliver+e->deliv_limt)
              d=(((endeliver+e->deliv_limt)-qudeliver)/(e->deliv_limt))*100
            else
              d=0
            endif
          endif
          replace succsprice with p
          replace succsdeliv with d
          unlock
          if (rnd()*100)<=p*d*dis
            replace status with "FIRM"
          else
            replace status with "LOST"
          endif
          unlock
        endif
        if .not. eof()
          skip
        endif
      enddo
      select 10
      replace goodserv with .f.
      unlock
      return
      **
*!***********************************************************************
*!                    This procedure creates the running screen displayed
*!    Procedure: RUNSCREN  while the gaming-simulation is running.
*!
*!    Called by: RUNFACT        (procedure in SYS11.PRG)
*!
*!***********************************************************************
      procedure runscren
      ******************
      store 0 to check
      set color to r*+/bu
      @ 6,31 say "RUNNING FACTORY"
      set color to gr+/bu
      @ 3,7 to 21,71 double
      set color to r+/bu
      @ 10,13 say "DATE:    "
      @ 10,48 say "TIME:    "
      set color to g/w
      @ 19,19 say "PRESS <END> OR Ctrl-F TO FREEZE THE GAME."
      return
      **
*!***********************************************************************
*!                    This procedure displays the time and date values
*!    Procedure: SAYTIME0  the first time the gaming-simulation starts running.
*!
*!    Called by: RUNFACT        (procedure in SYS11.PRG)
*!
*!        Calls: INKEY()       (FOXBASE+ function)
*!             : VAL()         (FOXBASE+ function)
*!             : SYS()         (FOXBASE+ function)
*!             : FACTSTOP      (procedure in SYS11.PRG)
*!
*!***********************************************************************
      procedure saytime0
      ******************
      set color to gr*+/bu
      @ 10,22 say j->factdate picture "@E"
      @ 10,57 say j->facttime
      store inkey() to check
      if check=6
        store val(sys(2)) to startstop
        save screen to runscr
        do factstop
        difstop=difstop+(val(sys(2))-startstop)
      endif
      return
      **
*!***********************************************************************
*!                    This procedure displays the time and date values
*!    Procedure: SAYTIME1  every time these are recalculated.
*!
*!    Called by: TIMECAL        (procedure in SYS11.PRG)
```

```
*!
*!**********************************************************************
    procedure saytime1
    ******************
    set color to gr*+/bu
    @ 10,22 say mmfactdate picture "@E"
    @ 10,57 say mmfacttime
    return
    **
*!**********************************************************************
*!                         This procedure calculates the gaming-simulation's time
*!      Procedure: TIMECAL    and date values. Its functioning is thoroughly explained in one
*!                         of the thesis's sections.
*!      Called by: RUNFACT        (procedure in SYS11.PRG)
*!
*!          Calls: MOD()          (FOXBASE+ function)
*!               : INT()         (FOXBASE+ function)
*!               : STR()         (FOXBASE+ function)
*!               : CTOD()         (FOXBASE+ function)
*!               : SAYTIME1        (procedure in SYS11.PRG)
*!               : RECNO()        (FOXBASE+ function)
*!               : .AND..NOT.EOF()(FOXBASE+ function)
*!               : TRIM()         (FOXBASE+ function)
*!               : SYSTEM          (procedure in SYS11.PRG)
*!               : .NOT.TRANSFORM((FOXBASE+ function)
*!               : TRANSFORM()    (FOXBASE+ function)
*!               : SPACE()        (FOXBASE+ function)
*!
*!**********************************************************************
    procedure timecal
    *****************
    factclock=factclock1
    factclock1=newclock
    difclock=factclock1-factclock
    factmin=mod(factclock1,60)
    facthhh=facthh
    facthh=int(factclock1/60)
    tempdd=factdd
    factdd=int(facthh/8)+factdd
    tempdd1=factdd
    facthh=mod(facthh,8)
    factmm=int(factdd/31)+factmm
    factdd=mod(factdd,31)
    if factdd=0
      factdd=1
    endif
    factyy=int(factmm/13)+factyy
    factmm=mod(factmm,13)
    if factmm=0
      factmm=1
    endif
    factyy=mod(factyy,100)
    if factyy=0
      factyy=1
    endif
    if factmin<10
      mmfacttime=str(facthh,1)+":0"+str(factmin,1)
    else
      mmfacttime=str(facthh,1)+":"+str(factmin,2)
    endif
    mfactdate=str(factdd,2)+"/"+str(factmm,2)+"/"+str(factyy,2)
    mmfactdate=ctod(mfactdate)
    if tempdd1=tempdd
      do saytime1
    endif
    select 2
    set order to 2
    goto bottom
    oldrec=recno()
    mwaittime=((mmfactdate-confdate)*480)+(facthh*60)+factmin
    goto bottom
    botrec=recno()
    if (oldrec<>botrec.or.first).and..not.eof()
      if botrec=1
        tt=botrec
        first=.f.
      else
        tt=oldrec+1
      endif
      xx=trim(system("newenqrec"))
      do while tt<=botrec
        if .not.transform(tt,"999") $ xx
          xx=transform(tt,"999")+xx
        endif
        tt=tt+1
```

```
        enddo
    else
      xx=space(9)
    endif
set order to 1
select 10
replace facttime  with mmfacttime
replace factdate  with mmfactdate
replace waittime  with mwaittime
replace newenqrec with xx
unlock
return
**
*!*******************************************************************
*!                       This procedure calculates the gaming-simulation's production
*!     Procedure: RUNPROD  values for each of the existing resources. Its functioning is
*!                         thoroughly explained in one of the thesis's sections.
*!     Called by: RUNFACT         (procedure in SYS11.PRG)
*!
*!         Calls: EOF()           (FOXBASE+ function)
*!              : .NOT.EOF()      (FOXBASE+ function)
*!              : MOD()           (FOXBASE+ function)
*!              : INT()           (FOXBASE+ function)
*!              : STR()           (FOXBASE+ function)
*!              : NEWCLOCK<()     (FOXBASE+ function)
*!              : .AND..NOT.EOF()(FOXBASE+ function)
*!
*!*******************************************************************
procedure runprod
*****************
select 3
goto top
do while .not. eof()
    reso=resour_cod
    select 1
    set filter to resour_cod=reso.and.ord_stat_1<>"FINISHED"
    goto top
    if .not.eof()
        goto top
        actsetup=f->type
        do case
        case c->t_avail=0
            if actsetup=c->lastsetup
              timepro=(quantit-product_0)*e->procestime
            else
              timepro=((quantit-product_0)*e->procestime)+f->setup
            endif
        case c->t_avail<>0
            if actsetup=c->lastsetup
              timepro=quantit*e->procestime
            else
              timepro=(quantit*e->procestime)+f->setup
            endif
        endcase
        do while newclock>=(c->t_avail+timepro)
            if ord_stat_1="NO_START"
                if c->t_avail<factclock
                  select 3
                  replace t_avail with factclock
                  unlock
                  select 1
                endif
              replace start_date with j->factdate
              unlock
              mfactmin=mod(c->t_avail,60)
              mfacthh=int(c->t_avail/60)
                if mfactmin<10
                  mfacttime=str(mfacthh,1)+":0"+str(mfactmin,1)
                else
                  mfacttime=str(mfacthh,1)+":"+str(mfactmin,2)
                endif
              replace start_time with mfacttime
              replace ord_stat_1 with "STARTED"
              unlock
              select 10
              replace statreso with a->resour_cod
              replace statcode with a->code
              unlock
              select 3
              replace avail with "NO "
              unlock
              select 1
            endif
            if tempdd1>tempdd
              replace finis_date with (j->factdate)-1
```

```
      else
        replace finis_date with (j->factdate)
      endif
      unlock
      mfactmin=mod((c->t_avail+timepro),60)
      mfacthh=int((c->t_avail+timepro)/60)
      if mfactmin<10
        mfacttime=str(mfacthh,1)+":0"+str(mfactmin,1)
      else
        mfacttime=str(mfacthh,1)+":"+str(mfactmin,2)
      endif
      select 4
      replace stock with stock+(a->quantit)
      unlock
      select 3
      replace avail with "YES"
      replace t_avail with (t_avail+timepro)
      unlock
      select 1
      replace finis_time with mfacttime
      replace product_1 with quantit
      replace ord_stat_1 with "FINISHED"
      unlock
      if .not. eof()
        skip
      endif
      if eof()
        select 3
        replace lastsetup with actsetup
        unlock
        select 1
        exit
      endif
      select 3
      replace lastsetup with actsetup
      unlock
      select 1
      actsetup=f->type
      if actsetup=c->lastsetup
        timepro=quantit*e->procestime
      else
        timepro=(quantit*e->procestime)+f->setup
      endif
    enddo
    if newclock<(c->t_avail+timepro).and..not.eof()
      if ord_stat_1="NO_START"
        if c->t_avail<factclock
          select 3
          replace t_avail with factclock
          unlock
          select 1
        endif
        replace start_date with (j->factdate)
        unlock
        mfactmin=mod(c->t_avail,60)
        mfacthh=int(c->t_avail/60)
        if mfactmin<10
          mfacttime=str(mfacthh,1)+":0"+str(mfactmin,1)
        else
          mfacttime=str(mfacthh,1)+":"+str(mfactmin,2)
        endif
        replace start_time with mfacttime
        replace ord_stat_1 with "STARTED"
        unlock
        select 10
        replace statreso with a->resour_cod
        replace statcode with a->code
        unlock
        select 3
        replace avail with "NO "
        unlock
        select 1
      endif
      if actsetup=c->lastsetup
        if tempdd1>tempdd
          produce=int((480-(c->t_avail))/(e->procestime))
        else
          produce=int((newclock-(c->t_avail))/(e->procestime))
        endif
      else
        if tempdd1>tempdd
          produce=int((480-(c->t_avail)-(f->setup))/(e->procestime))
        else
          produce=int((newclock-(c->t_avail)-(f->setup))/(e->procestime))
        endif
```

```
            endif
            select 1
            replace product_1 with produce+product_0
            unlock
          endif
        endif
        set filter to
        select 3
        if .not. eof()
          skip
        endif
    enddo
return
**
*!*************************************************************************
*!                  This procedure generates the enquires that will be shown
*!    Procedure: ENQGEN   to the sales department's people the next coming day. Its
*!                  functioning is thoroughly explained in one of the thesis's sections.
*!    Called by: GAMCONF        (procedure in SYS11.PRG)
*!
*!        Calls: LEN()          (FOXBASE+ function)
*!             : VAL()          (FOXBASE+ function)
*!             : RIGHT()        (FOXBASE+ function)
*!             : TIME()         (FOXBASE+ function)
*!             : ROUND()        (FOXBASE+ function)
*!             : RND            (procedure in SYS11.PRG)
*!             : SUBSTR()       (FOXBASE+ function)
*!             : MMFACTDATE<()  (FOXBASE+ function)
*!             : MMFACTDATE>()  (FOXBASE+ function)
*!             : IIF()          (FOXBASE+ function)
*!             : INT()          (FOXBASE+ function)
*!             : STUFF()        (FOXBASE+ function)
*!
*!*************************************************************************
procedure enqgen
****************
private mindd,maxdd,gendate
do while len(enqurec)<>0
    select 5
    store val(right(time(1),2)) to seed
    store round((rnd()*(len(enqurec)-1))+1,0)to element
    store val(substr(enqurec,element,1))to select
    goto select
    if mmfactdate<(confdate+(mindelivdd-1))
        maxdd=maxdelivdd
        mindd=mindelivdd
        gendate=(confdate+(mindelivdd-1))
    else
        if mmfactdate>(confdate+(maxdelivdd-1))
            replace tot_enqu with num_enqu
        else
            maxdd=(factdd+j->enquperiod)
            maxdd=iif(maxdd>maxdelivdd,maxdelivdd,maxdd)
            mindd=factdd+1
            gendate=mmfactdate+1
        endif
    endif
    if num_enqu<tot_enqu
        store val(right(time(1),2)) to seed
        store round((rnd()*(slack-1))+plow,0)to number
        number=number+((mmfactdate-confdate)*480)
        plow=plow+slack
        replace num_enqu with (num_enqu+1)
        unlock
        store code     to mcode
        store descript to mdescript
        store standprice to mstprice
        store val(right(time(1),2)) to seed
        store round((rnd()*(maxdd-mindd)),0)+gendateto mendate
        store int((rnd()*(maxquantit-minquantit))+minquantit)to menquant
        select 9
        append blank
        replace code with mcode
        replace descript with mdescript
        replace endate with mmfactdate
        replace enquantit with menquant
        replace endeliver with mendate
        replace enprice with mstprice
        replace qudeliver with mendate
        replace quprice with mstprice
        replace status with 'ENQU'
        replace wait  with number
        unlock
    else
        enqurec=stuff(enqurec,element,1,"")
```

```
        endif
    enddo
select 9
use
return
**
*!*************************************************************************
*!                      This procedure shipes the orders with delivery date equal or
*!   Procedure: SHIPPING less than the recently finished day. Its functioning is thoroughly
*!                      explained in one of the thesis's sections.
*!   Called by: RUNFACT        (procedure in SYS11.PRG)
*!
*!       Calls: EOF()          (FOXBASE+ function)
*!            : .NOT.FOUND()   (FOXBASE+ function)
*!            : RECNO()        (FOXBASE+ function)
*!
*!*************************************************************************
procedure shipping
******************
private quantdeliv,yeskip
select 2
goto top
do while .not. eof()
   yeskip=.t.
   if ((j->factdate)-1)>=qudeliver
      if d->stock>=enquantit
         quantdeliv=enquantit
         recdeliv=qudeliver
         replace realdeliv with ((j->factdate)-1)
         replace delivered with .t.
         unlock
         select 4
         replace stock with (stock-quantdeliv)
         unlock
         select 2
         unlock
         seek recdeliv
         if .not.found()
            goto recno(0)
         endif
         yeskip=.f.
      endif
   endif
   if .not. eof().and.yeskip
      skip
   endif
enddo
return
**
*!*************************************************************************
*!                      This procedure allows the game manager to make a choice among three
*!   Procedure: REPOMENU menu choices after a day has finished (save the day's data, run the
*!                      next day or quit the game).
*!   Called by: DAYREPOR       (procedure in SYS11.PRG)
*!
*!       Calls: SAVEDAY        (procedure in SYS11.PRG)
*!            : NEXTDAY        (procedure in SYS11.PRG)
*!            : EXITGAME       (procedure in SYS11.PRG)
*!
*! Memory Files: FIL.MEM
*!
*!*************************************************************************
procedure repomenu
******************
**
**
** MENU OF THE DAY REPORT, ALLOWING TO QUIT, SAVE DAY'S RESULTS OR CONTINUE.
**
**
set view to run
set color to
do while .t.
   clear
   m_menu=0
   set color to g/n
   @ 3,16,17,62 box "╒═╕│╛═╘│"
   set color to gr/n
   @ 5,26 to 7,53 double
   set color to bu/n
   @  6,29 say "DAY RUN FINISHED"
   set color to br/n
   @ 10,24 to 16,54
   set color to bg/n
   @ 12,30 prompt "1.SAVE DAY'S DATA."
   @ 13,30 prompt "2.RUN NEXT DAY.   "
```

```
    @ 14,30 prompt "3.QUIT GAME.      "
    save to fil all like lo*
    menu to m_menu
    do case
    case m_menu=1
       do saveday
    case m_menu=2
       do nextday
       exit
    case m_menu=3
       do exitgame
    endcase
  enddo
  return
  **
*!*********************************************************************
*!                    This procedure calculates the frequency of generation
*!    Procedure: FREQUEN  of enquires. Its functioning is thoroughly explained in one
*!                    of the thesis's sections.
*!    Called by: SALREPO        (procedure in SYS11.PRG)
*!
*!       Calls: EOF()          (FOXBASE+ function)
*!            : MIN()          (FOXBASE+ function)
*!            : INT()          (FOXBASE+ function)
*!
*!*********************************************************************
  procedure frequen
  *****************
  select 1
  goto top
  nday=(mmfactdate-(j->startdate))
  do while .not. eof()
     if d->ave_qprice<=mrkt_price
        pe=1
     else
        if d->ave_qprice>mrkt_price.and.d->ave_qprice<mrkt_limt
           pe=((mrkt_limt)-(d->ave_qprice))/((mrkt_limt)-(mrkt_price))
        else
           pe=0
        endif
     endif
     if ontimedelv>=100
        de=1
     else
        if ontimedelv>ontimlimt.and.ontimedelv<100
           de=((ontimedelv)-(ontimlimt))/(100-ontimlimt)
        else
           de=0
        endif
     endif
     f=1+(((6*nday-1)/(5*nday))*(d->ave_enqu)*min(pe,de))
     replace tot_enqu with int(f)
     unlock
     if .not. eof()
        skip
     endif
  enddo
  sum tot_enqu to sum_enqu
  select 10
  replace frequency with int(480/sum_enqu)
  unlock
  select 1
  return
  **
*!*********************************************************************
*!                    This procedure calculates and stores the report variables
*!    Procedure: SALPAST  concerning the sales department's daily performance.
*!
*!    Called by: SALREPO        (procedure in SYS11.PRG)
*!
*!       Calls: EOF()          (FOXBASE+ function)
*!            : IIF()          (FOXBASE+ function)
*!            : SALPERF        (procedure in SYS11.PRG)
*!
*!*********************************************************************
  procedure salpast
  *****************
  select 1
  goto top
  do while .not. eof()
     store 0 to numenq,numquo,numfirm,mavqprice,mqprice,mqdeliv,numontime,;
     numdeliv,timdeliv,mfprice,mfdeliv,mvol,mthrough,dvol,tobedeliv
     store code to mcode
     store descript to mdescript
     store (mmfactdate-1) to mdate
```

```
select 2
goto top
do while .not. eof()
   if code=mcode
      if endate=(j->factdate)-1
         numenq=numenq+1
      endif
      if qudate=(j->factdate)-1
         if status<>"ENQU"
            numquo=numquo+1
            mavqprice=(mavqprice+quprice)
            mqprice=(mqprice+succsprice)
            mqdeliv=(mqdeliv+succsdeliv)
            if status="FIRM"
               numfirm=numfirm+1
               mfprice=(mfprice+succsprice)
               mfdeliv=(mfdeliv+succsdeliv)
               mvol=mvol+(enquantit*quprice)
            endif
         endif
      endif
      if qudeliver=(j->factdate)-1
         tobedeliv=tobedeliv+1
      endif
      if realdeliv=(j->factdate)-1
         if qudeliver=realdeliv
            numontime=numontime+1
         endif
         numdeliv=numdeliv+1
         timdeliv=(timdeliv+(realdeliv-qudeliver))
         mthrough=mthrough+(enquantit*quprice)
      endif
      perquot=iif(numenq<>0,(numquo/numenq)*100,0)
      perfirm=iif(numquo<>0,(numfirm/numquo)*100,0)
      perontime=iif(tobedeliv<>0,(numontime/tobedeliv)*100,100)
   endif
   if .not. eof()
      skip
   endif
enddo
mavqprice=iif(numquo<>0,(mavqprice/numquo),0)
mqprice=iif(numquo<>0,(mqprice/numquo),0)
mqdeliv=iif(numquo<>0,(mqdeliv/numquo),0)
mfprice=iif(numfirm<>0,(mfprice/numfirm),0)
mfdeliv=iif(numfirm<>0,(mfdeliv/numfirm),0)
timdeliv=iif(numdeliv<>0,(timdeliv/numdeliv),0)
select 3
append blank
replace code       with mcode
replace descript   with mdescript
replace date       with mdate
replace per_fprice with mfprice
replace per_fdeliv with mfdeliv
replace ave_enqu   with numenq
replace ave_quot   with numquo
replace ave_qprice with mavqprice
replace per_qprice with mqprice
replace per_qdeliv with mqdeliv
replace per_ontime with perontime
replace deliveries with numdeliv
replace todeliv    with tobedeliv
replace ovr_deltim with timdeliv
replace per_quot   with perquot
replace ave_firm   with numfirm
replace per_firm   with perfirm
replace volume     with mvol
unlock
select 10
replace throughput with (throughput+mthrough)
unlock
select 1
if .not. eof()
   skip
endif
enddo
do salperf                  && CALCULATE AND STORE THE TOTAL PERFORMACE
return                      && AVERAGES.
**
*!*********************************************************************
*!                  This procedure calculates and stores the report variables
*!    Procedure: SALPERF  concerning the sales department's overall performance.
*!
*!    Called by: SALPAST        (procedure in SYS11.PRG)
*!
*!        Calls: EOF()          (FOXBASE+ function)
```

```
*!
*!***************************************************************************
  procedure salperf
  *****************
  select 4
  goto top
  do while .not. eof()
     store code to mcode
     store (mmfactdate-1) to mdate
     select 3
     average for code=mcode to mave_enqu,mave_quot,mper_quot,mave_firm,;
     mper_firm,movr_deltim,mper_ontime,mave_qprice,mper_qprice,;
     mper_qdeliv,mper_fprice,mper_fdeliv,mdeliveries,mtodeliv,mvolume
     select 4
     replace date with mdate
     replace ave_enqu with mave_enqu
     replace ave_quot with mave_quot
     replace per_quot with mper_quot
     replace ave_firm with mave_firm
     replace per_firm with mper_firm
     replace ovr_deltim with movr_deltim
     replace per_ontime with mper_ontime
     replace ave_qprice with mave_qprice
     replace per_qprice with mper_qprice
     replace per_qdeliv with mper_qdeliv
     replace per_fprice with mper_fprice
     replace per_fdeliv with mper_fdeliv
     replace deliveries with mdeliveries
     replace todeliv    with mtodeliv
     replace volume     with mvolume
     unlock
     if .not. eof()
        skip
     endif
  enddo
  return
  **
*!***************************************************************************
*!              This procedre calculates and stores the report variables
*!     Procedure: PLAPAST  concerning the planning department's daily
*!                         and overall performance.
*!     Called by: PLAREPO         (procedure in SYS11.PRG)
*!
*!        Calls: EOF()          (FOXBASE+ function)
*!             : .NOT.EOF()     (FOXBASE+ function)
*!             : VAL()          (FOXBASE+ function)
*!             : STR()          (FOXBASE+ function)
*!             : LEFT()         (FOXBASE+ function)
*!             : RIGHT()        (FOXBASE+ function)
*!             : CTOD()         (FOXBASE+ function)
*!
*!***************************************************************************
  procedure plapast
  *****************
  select 2
  store 0 to totprod
  store 0 to mmwip
  goto top
  do while .not. eof()
     store 0 to product,util,nset,tset
     store " " to lset,sett
     store resour_cod to cod
     store capacity to cap
     store redescript to descrp
     select 1
     set filter to resour_cod=cod.and.ord_stat_0<>"FINISHED".and.ord_stat_1<>"NO_START"
     goto top
     do while .not.eof()
        sett=d->set_up
        product=product+((product_1-product_0)*d->procestime)&& production in minutes
        if lset<>sett
           nset=nset+1
           tset=tset+h->setup
        endif
        lset=sett
        if product_1<quantit
           product=480-tset
        endif
        if .not. eof()
           skip
        endif
     enddo
     totprod=totprod+product
     store (product/cap) to util
     set filter to
```

```
        select 5
        append blank
        replace resour_cod with cod
        replace descript with descrp
        replace capacity with cap
        replace production with product
        replace productiv with util*100 && Utilization in percentage.
        replace setuptime with tset
        replace setupnum with nset
        replace date with (j->factdate)-1
        unlock
        select 2
        if .not. eof()
          skip
        endif
    enddo
*
*
select 1
set relation to
*
*
select 4
goto top
do while .not. eof()
    store code to cod
    store descript to descrp
    store procestime to mprocstim
    store matcost to mmatcost
    store 0 to incomp,start,batch,prod,last,pprod
    store 0 to cost,finish,mwip,lasting,avelas
    select 1
    set filter to code=cod
    goto top
    do while .not. eof()
        if ord_stat_0="STARTED".and.ord_stat_1="STARTED"
          incomp=incomp+1
          mwip=mwip+product_1
          pprod=pprod+((product_1-product_0)*d->procestime)
        endif
        if ord_stat_0="NO_START".and.ord_stat_1="STARTED"
          start=start+1
          mwip=mwip+product_1
          pprod=pprod+((product_1-product_0)*d->procestime)
        endif
        if finis_date=(j->factdate)-1
          store val(str(start_date-finis_date,3)) to difdays
          store val(left(start_time,1))          to starthour
          store val(right(start_time,2))          to startmin
          store val(left(finis_time,1))          to finishour
          store val(right(finis_time,2))          to finismin
          store (finishour-starthour)             to difhour
          store (finismin-startmin)               to difmin
          last=(difdays*8*60)+(difhour*60)+difmin
          lasting=lasting+last
          finish=finish+1
          prod=prod+quantit
          pprod=pprod+((product_1-product_0)*d->procestime)
        endif
        if .not. eof()
          skip
        endif
    enddo
    if pprod=0
      cost=d->stand_cost
    else
      cost=((mprocstim)*(j->hour_rate))+(mmatcost)+(((j->tot_overh)*pprod)/totprod)&& overhead per day
    endif
    set filter to
    avelas=iif(finish=0,0,(lasting/finish))
    mmwip=mmwip+(mwip*mmatcost)
    mwip=iif((incomp+start)=0,0,(mwip/(incomp+start)))
    batch=iif(finish=0,0,(prod/finish))
    select 6
    append blank
    replace code with cod
    replace descript with descrp
    replace incomplete with incomp
    replace started with start
    replace finished with finish
    replace production with prod
    replace wip with mwip
    replace ave_batch with batch
    replace setupnum with (incomp+start+finish)
    replace leadtime with avelas
```

```
        replace stand_cost with cost
        replace date with (j->factdate)-1
        unlock
        select 4
        replace stand_cost with cost
        unlock
        if .not. eof()
          skip
        endif
    enddo
*
*
select 3
goto top
do while .not. eof()
        stkcode=code
        stkdes=descript
        stk=stock
        mmwip=mmwip+(stk*d->matcost)
        select 7
        append blank
        replace code with stkcode
        replace descript with stkdes
        replace stock with stk
        replace date with (j->factdate)-1 for date=ctod("  /  /  ")
        unlock
        select 3
        if .not.eof()
          skip
        endif
enddo
select 10
replace systock with mmwip
unlock
select 3
return
**
*!*********************************************************************
*!                  This procedure displays the resources' information, allowing
*!    Procedure: MACHBROW  also its edition. It's based on the "browse" algorithm which is
*!                  thoroughly explained in one of the thesis's sections.
*!    Called by: FACTCONF       (procedure in SYS11.PRG)
*!
*!        Calls: CHR()          (FOXBASE+ function)
*!             : REPLICATE()    (FOXBASE+ function)
*!             : RECNO()        (FOXBASE+ function)
*!             : MACH_DISP      (procedure in SYS11.PRG)
*!             : GETKEY         (procedure in SYS11.PRG)
*!             : BOF()          (FOXBASE+ function)
*!             : EOF()          (FOXBASE+ function)
*!             : MACH_RSTR      (procedure in SYS11.PRG)
*!             : MACHGETS       (procedure in SYS11.PRG)
*!             : UPDATED()      (FOXBASE+ function)
*!             : MACH_REPL      (procedure in SYS11.PRG)
*!             : MACH_STOR      (procedure in SYS11.PRG)
*!             : NEWMACH        (procedure in SYS11.PRG)
*!             : .NOT.DELETED() (FOXBASE+ function)
*!             : RESETUP        (procedure in SYS11.PRG)
*!
*!*********************************************************************
procedure machbrow
******************
select 3
**
private recnumtop,recnumlast,skiprecs,home,endkey,uparrow,downarrow
private row,rowtop,rowbottom,rowprompt,keystrokes,pagepaint,recnum
private promptrow,promptbar,keyst2,choice
* ---Initialize constants.
choice=" "
uparrow = chr(5)
downarrow = chr(24)
returnkey= chr(13)
pgdn= chr(3)
pgup= chr(18)
delrecord= chr(7)
keystrokes = "EN"+uparrow+downarrow+;
pgdn+pgup+delrecord+returnkey
keyst2 = "R"+delrecord+returnkey
rowtop = 1
rowbottom = 20
rowprompt = rowbottom + 3
promptrow=22
promptbar=replicate(chr(196),80)
skiprecs = rowbottom - rowtop + 1
goto top
```

```
* ---Initialize local variables.
row = rowtop
recnum = recno()
recnumtop = recnum
pagepaint = .t.
isedited = .f.
* ---Perform BROWSE.
set color to &screenatr
clear
* ---The following loop is really a "REPEAT/UNTIL <cond>".
do while .t.
    if pagepaint
        recnum = recno()
        goto recnumtop
        do mach_disp with (rowtop),skiprecs
        goto recnum
        * ---Reposition record pointer when repainting current page.
        row = rowtop
        pagepaint = .f.
    endif
    set color to &promptatr
    @ rowprompt-1,0 say promptbar

    @ rowprompt,0 say ;
    "BROWSE:  (E)xit  <Arrows>  <Del>  (N)ew  <Return> "
    @ row,4 say chr(16)
    do getkey with choice,keystrokes
    * ---Reposition record pointer.
    do while choice $ uparrow+downarrow
        @ row,4 say " "
        if choice = uparrow
            skip -1
            do case
            case bof()
                goto top
            case row > rowtop
                row = row - 1
            otherwise
                skip
            endcase
        else
            skip
            do case
            case eof()
                goto bottom
            case row < rowbottom
                row = row + 1
            otherwise
                skip -1
            endcase
        endif
        @ row,4 say chr(16)
        do getkey with choice,keystrokes
    enddo
    * ---Prompt line selections.
    do case
    case choice = "E"
                exit
    case choice = returnkey
        do mach_rstr
        do machgets with .f.
        if updated()
            do mach_repl
        endif
        do mach_disp with row,1
    case choice = "N"
        @ row,4 say " "
        do while r>20
            goto recnumtop
            skip skiprecs
            if eof()
                goto bottom
            endif
            recnumtop = recno()
            recnum = recno()
            goto recnumtop
            do mach_disp with (rowtop),skiprecs
        enddo
        goto recnum
        row=rowtop
        set color to &promptatr
        @ row,4 say chr(16)
        do mach_stor
        do newmach
        goto recnumtop
```

```
            pagepaint=.t.
        case choice = delrecord
            * ---Delete the record.
            if .not.deleted()
                delete
            endif
            recnum=recnumtop
            goto recnum
            pagepaint=.t.
        case choice = pgdn
            if .not. eof()
                goto recnumtop
                skip skiprecs
                if eof()
                    goto bottom
                endif
                recnumtop = recno()
                pagepaint = .t.
            endif
        case choice = pgup
            if .not. bof()
                goto recnumtop
                skip -skiprecs
                if bof()
                    goto top
                endif
                recnumtop = recno()
                pagepaint = .t.
            endif
        endcase
    enddo
goto top
if updated()
    set color to bu+/n
    @ 8,24 to 12,54 double
    set color to gr+*/n
    @ 10,27 say "UPDATING THE SET-UP FILE"
    set color to
    do resetup
endif
return
**
*!**************************************************************************
*!                  This procedure displays the resources' information.
*!     Procedure: MACH_DISP
*!
*!     Called by: MACHBROW        (procedure in SYS11.PRG)
*!
*!         Calls: ROW()           (FOXBASE+ function)
*!              : COL()           (FOXBASE+ function)
*!
*!**************************************************************************
procedure mach_disp
*******************
parameter row,listrecs
if listrecs > 1
    * ---Display heading when listing the entire page.
    set color to &statusatr
    @ rowtop-1,0
    ?? "      RESOURCE DESCRIPTION---- CAPACITY SETUP1 SETUP2 SETUP3 SETUP4 SETUP5"
    * ---Clear the window area.
    set color to &windowatr
    @ rowtop,0 clear to rowbottom,79
endif
* ---Display the records.
set heading off
set color to &windowatr
@ row-1,0 say ""
list next listrecs "   ",resour_cod,"   ",redescript," ",capacity,"",set_up1,set_up2,set_up3,set_up4,set_up5off
set heading on
r=row()
c=col()+5
return
**
*!**************************************************************************
*!                  This procedure calls the procedure that enable the game
*!     Procedure: NEWMACH  manager to create and edit a new resource.
*!
*!     Called by: MACHBROW        (procedure in SYS11.PRG)
*!
*!         Calls: MACHEDIT        (procedure in SYS11.PRG)
*!
*!**************************************************************************
procedure newmach
*****************
```

```
    do while .t.
        do machedit
        if choice < > "R"
            exit
        endif
        set color to &promptatr
        @ rowprompt-1,0 say promptbar
        @ rowprompt,0 say ;
        "BROWSE:  (E)xit  <Arrows>   <Del>   (N)ew   <Return> "
    enddo
    return
    **
*!**************************************************************************
*!                          This procedure allows the game manager the edition of
*!     Procedure: MACHEDIT  the input fields of a newly created resource.
*!
*!     Called by: NEWMACH         (procedure in SYS11.PRG)
*!
*!        Calls: MACHGETS        (procedure in SYS11.PRG)
*!             : GETKEY          (procedure in SYS11.PRG)
*!             : INCLUDE         (procedure in SYS11.PRG)
*!
*!**************************************************************************
    procedure machedit
    ******************
    do machgets with .t.
    set color to &promptatr
    @ rowprompt,0 clear

    @ rowprompt,0 say ;
    "NEW MACHINE:   <Del> without saving  (R)ewrite   <Return> saving "
    do getkey with choice,keyst2
    if choice=returnkey
        do include
    endif
    @ rowprompt,0 clear
    return
    **
*!**************************************************************************
*!                          This procedure gets the values input by the game manager
*!     Procedure: MACHGETS   when editing the resource's input fields.
*!
*!     Called by: MACHBROW        (procedure in SYS11.PRG)
*!              : MACHEDIT        (procedure in SYS11.PRG)
*!
*!**************************************************************************
    procedure machgets
    ******************
    parameter newget
    set color to n/w
    if newget
        @ r,c      get mresourcod picture "@!NNNN"
        @ r,c+9    get mdescript  picture "@!"
        @ r,c+27 get mcapacity  picture "9999"
        @ r,c+33 get msetup1     picture "999.99"
        @ r,c+40 get msetup2     picture "999.99"
        @ r,c+47 get msetup3     picture "999.99"
        @ r,c+54 get msetup4     picture "999.99"
        @ r,c+61 get msetup5     picture "999.99"
    else
        @ row,c     say mresourcod
        @ row,c+9   say mdescript
        @ row,c+27 get mcapacity picture "9999"
        @ row,c+33 get msetup1    picture "999.99"
        @ row,c+40 get msetup2    picture "999.99"
        @ row,c+47 get msetup3    picture "999.99"
        @ row,c+54 get msetup4    picture "999.99"
        @ row,c+61 get msetup5    picture "999.99"
    endif
    read
    return
    **
*!**************************************************************************
*!                          This procedure includes a new record in the resource
*!     Procedure: INCLUDE    datafile.
*!
*!     Called by: MACHEDIT        (procedure in SYS11.PRG)
*!
*!        Calls: MACH_REPL       (procedure in SYS11.PRG)
*!
*!**************************************************************************
    procedure include
    *****************
    append blank
    do mach_repl
```

```
return
**
*!************************************************************************
*!                        This procedure replaces in the newly included record the
*!     Procedure: MACH_REPL  values input in the resource input procedure.
*!
*!     Called by: MACHBROW       (procedure in SYS11.PRG)
*!              : INCLUDE        (procedure in SYS11.PRG)
*!
*!        Calls: EOF()          (FOXBASE+ function)
*!
*!************************************************************************
procedure mach_repl
*******************
    if .not. eof()
      * ---Replace only if there is an available record
      replace;
      resour_cod with mresourcod,;
      redescript with mdescript,;
      capacity   with mcapacity,;
      lastsetup  with lstsetup,;
      set_up1    with msetup1,;
      set_up2    with msetup2,;
      set_up3    with msetup3,;
      set_up4    with msetup4,;
      set_up5    with msetup5
      unlock
    endif
return
**
*!************************************************************************
*!                        This procedure stores the values of a resource record
*!     Procedure: MACH_RSTR   to be reedited.
*!
*!     Called by: MACHBROW       (procedure in SYS11.PRG)
*!
*!************************************************************************
procedure mach_rstr
*******************
store resour_cod to mresourcod
store redescript to mdescript
store capacity   to mcapacity
store lastsetup  to lstsetup
store set_up1    to msetup1
store set_up2    to msetup2
store set_up3    to msetup3
store set_up4    to msetup4
store set_up5    to msetup5
return
**
*!************************************************************************
*!                        This procedure stores the default values of a resource
*!     Procedure: MACH_STOR   record to be edited.
*!
*!     Called by: MACHBROW       (procedure in SYS11.PRG)
*!
*!        Calls: SPACE()        (FOXBASE+ function)
*!
*!************************************************************************
procedure mach_stor
*******************
store space(4)  to mresourcod
store space(15) to mdescript
store 0         to mcapacity
store "NO_SET " to lstsetup
store 0         to msetup1
store 0         to msetup2
store 0         to msetup3
store 0         to msetup4
store 0         to msetup5
return
**
*!************************************************************************
*!                        This procedure calls the procedures that will create the
*!     Procedure: OPEN        different necessary "views" to run the gaming-simulation.
*!
*!     Called by: SYS11.PRG
*!
*!        Calls: OPENRUN        (procedure in SYS11.PRG)
*!             : OPENSAL        (procedure in SYS11.PRG)
*!             : OPENPLA        (procedure in SYS11.PRG)
*!
*!************************************************************************
procedure open
**************
```

```
      do openrun
      do opensal
      do openpla
      return
      **
*!*************************************************************************
*!                       This procedure reindexes the used files in case
*!     Procedure: REINDEX     there has been any indexing error.
*!
*!     Called by: GAMCONF        (procedure in SYS11.PRG)
*!
*!         Calls: DELETC         (procedure in SYS11.PRG)
*!
*!          Uses: SYSTEM.DBF
*!              : PLANN.DBF
*!              : MARKET.DBF
*!              : MACHINE.DBF
*!              : STOCK.DBF
*!              : BILL.DBF
*!              : SALPAST.DBF
*!              : SALPERF.DBF
*!              : MACHPAST.DBF
*!              : PLAPAST.DBF
*!              : STKPAST.DBF
*!              : SETUP.DBF
*!
*!       Indexes: ORPLANN.IDX
*!              : NAPLANN.IDX
*!              : DEMARK.IDX
*!              : WAIT.IDX
*!              : SALDELIV.IDX
*!              : RESOUR.IDX
*!              : CODESTK.IDX
*!              : BILLNA.IDX
*!              : CDSAL.IDX
*!              : CPRFSAL.IDX
*!              : CDMACH.IDX
*!              : CDPLA.IDX
*!              : CDSTK.IDX
*!              : SETUP.IDX
*!
*!*************************************************************************
      procedure reindex
      *****************
      set exclusive off
      select 10
      use system
      replace reindex with .t.    && inform the users that the databases
      unlock                      && can not be used by them.
      set exclusive on
      use plann index orplann,naplann
      pack
      use market index demark,wait,saldeliv
      reindex
      do deletc
      pack
      use machine index resour
      reindex
      use stock index codestk
      reindex
      use bill index billna
      reindex
      use salpast index cdsal
      reindex
      use salperf index cprfsal
      reindex
      use machpast index cdmach
      reindex
      use plapast index cdpla
      reindex
      use stkpast index cdstk
      reindex
      use setup index setup
      reindex
      set exclusive off
      select 10
      use system
      replace reindex with .f.    && inform the users that the databases
      unlock                      && can be used by them.
      return
      **
*!*************************************************************************
*!                       This procedure creates the "view" to be used when
*!     Procedure: OPENRUN     running the gaming-simulation.
*!
```

```
*!    Called by: OPEN          (procedure in SYS11.PRG)
*!
*!         Uses: PLANN.DBF
*!             : MARKET.DBF
*!             : MACHINE.DBF
*!             : STOCK.DBF
*!             : BILL.DBF
*!             : SETUP.DBF
*!             : SYSTEM.DBF
*!
*!      Indexes: ORPLANN.IDX
*!             : NAPLANN.IDX
*!             : DEMARK.IDX
*!             : WAIT.IDX
*!             : SALDELIV.IDX
*!             : RESOUR.IDX
*!             : CODESTK.IDX
*!             : BILLNA.IDX
*!             : SETUP.IDX
*!
*!*****************************************************************
procedure openrun
*****************
**
** DATAFILES USED TO RUN THE FACTORY. **
**
set exclusive off
select 1
use plann index orplann,naplann
select 2
use market index demark,wait,saldeliv
set filter to wait<=mwaittime
select 3
use machine index resour
select 4
use stock index codestk
select 5
use bill index billna
select 6
use setup index setup
select 10
use system
**
** RELATIONS USED WHILE RUNNING THE FACTORY SIMULATION/GAME
**
select 2
set relation to code into d
set relation to code into e additive
select 1
set relation to code into d
set relation to code into e additive
*set relation to resour_cod into c additive
select 5
set relation to code into f
set relation to resour_cod into c additive
set safety off
create view run from environment all
set safety on
close databases
return
**
*!*****************************************************************
*!                        This procedure creates the "view" to be used when
*!    Procedure: OPENSAL   calculating the sales department's past performance
*!                        variables.
*!    Called by: OPEN          (procedure in SYS11.PRG)
*!
*!         Uses: BILL.DBF
*!             : MARKET.DBF
*!             : SALPAST.DBF
*!             : SALPERF.DBF
*!             : SYSTEM.DBF
*!
*!      Indexes: BILLNA.IDX
*!             : SALDELIV.IDX
*!             : DEMARK.IDX
*!             : WAIT.IDX
*!             : CDSAL.IDX
*!             : CPRFSAL.IDX
*!
*!*****************************************************************
procedure opensal
*****************
**
** DATAFILES USED TO CALCULATE THE SALES PERFORMANCE. **
```

```
**
set exclusive off
select 1
use bill index billna
select 2
use market index saldeliv,demark,wait
set filter to .not.delivered.or.realdeliv=conffdate
select 3
use salpast index cdsal
select 4
use salperf index cprfsal
select 10
use system
**
** RELATIONS USED WHILE CALCULATING THE SALES DEPT. PERFORMANCE.
**
select 1
set relation to code into d
select 4
set relation to code into c
set safety off
create view sales from environment all
set safety on
close databases
return
**
*!**************************************************************************
*!                        This procedure creates the "view" to be used when
*!    Procedure: OPENPLA   calculating the planning department's past performance
*!                        variables.
*!    Called by: OPEN          (procedure in SYS11.PRG)
*!
*!         Uses: PLANN.DBF
*!             : MACHINE.DBF
*!             : STOCK.DBF
*!             : BILL.DBF
*!             : MACHPAST.DBF
*!             : PLAPAST.DBF
*!             : STKPAST.DBF
*!             : SETUP.DBF
*!             : SYSTEM.DBF
*!
*!      Indexes: ORPLANN.IDX
*!             : RESOUR.IDX
*!             : CODESTK.IDX
*!             : BILLNA.IDX
*!             : CDMACH.IDX
*!             : CDPLA.IDX
*!             : CDSTK.IDX
*!             : SETUP.IDX
*!
*!**************************************************************************
procedure openpla
*****************
**
** DATAFILES USED TO CALCULATE THE PLANNING PERFORMANCE. **
**
set exclusive off
select 1
use plann index orplann
select 2
use machine index resour
select 3
use stock index codestk
select 4
use bill index billna
select 5
use machpast index cdmach
select 6
use plapast index cdpla
select 7
use stkpast index cdstk
select 8
use setup index setup
select 10
use system
**
** RELATIONS USED WHILE CALCULATING THE PLANNING DEPT. PERFORMANCE.
**
select 1
set relation to code into d
set relation to code into h additive
select 3
set relation to code into d
set safety off
```

```
create view plann from environment all
set safet on
close databases
return
**
*!***********************************************************************
*!                        This procedure obtains the most updated value of
*!    Procedure: SYSTEM    the fields in the system datafile.
*!
*!    Called by: DAYREPOR       (procedure in SYS11.PRG)
*!            : TIMECAL        (procedure in SYS11.PRG)
*!            : RENSTAT        (procedure in SYS11.PRG)
*!
*!       Calls: STR()          (FOXBASE+ function)
*!            : SELECT()        (FOXBASE+ function)
*!
*!***********************************************************************
procedure system
****************
parameters fielddd
base=str(select(),2)
set exclusive off
select 10
goto top
system=&fielddd
select &base
return system
*
*!***********************************************************************
*!                        This procedure gets the input of a key stroke
*!    Procedure: GETKEY    on the keyboard.
*!
*!    Called by: MACHBROW       (procedure in SYS11.PRG)
*!            : MACHEDIT       (procedure in SYS11.PRG)
*!
*!       Calls: INKEY()        (FOXBASE+ function)
*!            : UPPER()        (FOXBASE+ function)
*!            : CHR()          (FOXBASE+ function)
*!
*!***********************************************************************
procedure getkey
****************
parameter choice,keychars
private keycode
choice = "*"
do while .not. (choice $ keychars)
   keycode = inkey()
   if keycode > 0
      choice = upper(chr(keycode))
   endif
enddo
return
**
*!***********************************************************************
*!                        This procedure allows the user to exit the game before
*!    Procedure: EXITSYS   it has started.
*!
*!    Called by: MAINMENU       (procedure in SYS11.PRG)
*!
*!       Calls: SET            (procedure in SYS11.PRG)
*!
*!***********************************************************************
procedure exitsys
*****************
do set
set color to
clear
close all
clear all
return
**
**
*!***********************************************************************
*!                        This procedure obtains the value of the setup time
*!    Procedure: FIELDD    assign to a certain setup field.
*!
*!    Called by: RESETUP        (procedure in SYS11.PRG)
*!
*!       Calls: FCOUNT()       (FOXBASE+ function)
*!            : FIELD()        (FOXBASE+ function)
*!
*!***********************************************************************
procedure fieldd
****************
parameter fieldd
```

```
n=fcount(3)
nn=1
fiel=field(nn,3)
do while nn<n.and.fiel<>fieldd
   nn=nn+1
   fiel=field(nn,3)
enddo
return c->&fiel
**
*!**********************************************************************
*!                    This procedure updates the values of the setup datafile
*!   Procedure: RESETUP   in case there has been an update of the resource datafile.
*!
*!   Called by: MACHBROW       (procedure in SYS11.PRG)
*!
*!      Calls: EOF()          (FOXBASE+ function)
*!           : FIELDD          (procedure in SYS11.PRG)
*!
*!**********************************************************************
procedure resetup
*****************
select 5
goto top
do while .not. eof()
   select 6
   replace type with e->set_up
   replace setup with fieldd(e->set_up)
   unlock
   select 5
   if .not. eof()
      skip
   endif
enddo
return
**
*!**********************************************************************
*!                    This procedure calculates the number of deliveries
*!   Procedure: ONTIME     done on time.
*!
*!   Called by: SALREPO       (procedure in SYS11.PRG)
*!
*!      Calls: EOF()          (FOXBASE+ function)
*!
*!**********************************************************************
procedure ontime
****************
select 1
goto top
do while .not. eof()
   replace ontimedelv with d->per_ontime
   unlock
   if .not. eof()
      skip
   endif
enddo
return
**
*!**********************************************************************
*!                    This procedure calculates the variable (enqurec) where all the
*!   Procedure: ENQUSET   record numbers of the enquires to be generated will be stored.
*!
*!   Called by: GAMCONF       (procedure in SYS11.PRG)
*!
*!      Calls: EOF()          (FOXBASE+ function)
*!           : LTRIM()        (FOXBASE+ function)
*!           : STR()         (FOXBASE+ function)
*!           : RECNO()        (FOXBASE+ function)
*!
*!**********************************************************************
procedure enquset
*****************
enqurec=""
select 5
goto top
do while .not. eof()
   enqurec=enqurec+ltrim(str(recno()))
   if .not. eof()
      skip
   endif
enddo
return
**
*!**********************************************************************
*!                    This procedure configurates the gaming-simulation to
*!   Procedure: GAMCONF   a certain starting date.
```

```
*!
*!      Called by: SYS11.PRG
*!               : NEXTDAY        (procedure in SYS11.PRG)
*!
*!          Calls: CTOD()         (FOXBASE+ function)
*!               : RESTVAR        (procedure in SYS11.PRG)
*!               : CHEKFILE       (procedure in SYS11.PRG)
*!               : DTOC()         (FOXBASE+ function)
*!               : LEFT()         (FOXBASE+ function)
*!               : SUBSTR()       (FOXBASE+ function)
*!               : RESTART        (procedure in SYS11.PRG)
*!               : REINDEX        (procedure in SYS11.PRG)
*!               : RESET          (procedure in SYS11.PRG)
*!               : .NOT.FILE()    (FOXBASE+ function)
*!               : ENQUSET        (procedure in SYS11.PRG)
*!               : ENQGEN         (procedure in SYS11.PRG)
*!               : ADDREC         (procedure in SYS11.PRG)
*!
*!           Uses: SYSTEM.DBF
*!               : &MAR.DBF
*!
*!*****************************************************************************
    procedure gamconf
    *****************
    private cmffdate,mar,mffdate
    clear
    select 10
    store ctod("01/07/91") to confdate
    store factdate to conffdate
    set color to gr+/bu
    @ 5,6 to 18,72 double
    set color to g+/n
    @ 6,24 to 8,54
    set color to r*+/n
    @ 7,29 say "CONFIGURE DATE TO RUN."
    set color to bg+/n
    @ 12,16 say "ENTER THE GAME'S STARTING DATE :"
    @ 12,50 say confdate picture "@E"
    read
    set color to bg+/n
    @ 13,16 say "ENTER THE ACTUAL DATE :"
    @ 13,50 get conffdate picture "@E" valid conffdate>=confdate
    read
    mmfactdate=conffdate
    do restvar with (conffdate-1)
    do chekfile
    mffdate=dtoc(mmfactdate)
    cmffdate=left(mffdate,2)+substr(mffdate,4,2)
    mar=cmffdate+"mar.dbf"
    if updated().or.conffdate=confdate
        if pfiles
            do restart
            do reindex
            select 10
            use system
            replace restart with .f.
            unlock
            store j->waittime to mwaittime
            set view to run
            do reset
            if .not.file("&mar")
                set color to gr+*/n
                @ 14,25 say "CREATING THE ENQUIRES FILE"
                select 2
                copy stru to &mar
                select 9
                use &mar
                do enquset
                do enqgen
            endif
            do addrec
        else
            set color to
            clear
            @ 12,30 say "No existing past files."
            @ 13,28 say "Press any key to continue."
            wait
            store j->waittime to mwaittime
<=====================return
        endif
    else
        set color to
        @ 12,7 clear to 13,70
        set color to gr+/bu
        @ 5,6 to 18,72 double
```

```
    set color to g+/n
    @ 6,24 to 8,54
    set color to r*+/n
    @ 7,25 say "CONTINUE RUN FROM ACTUAL DAY."
    set color to bg+/n
    @ 11,18 say "THE DATE TO CONTINUE FROM IS :"
    ?? mffdate
    set color to ,n/w
    close databases
    select 10
    use system
    replace restart with .t.
    unlock
    do reindex
    select 10
    use system
    replace restart with .f.
    unlock
    store j->waittime to mwaittime
    set view to run
    do reset
    if .not.file("&mar")
      set color to gr+*/n
      @ 14,20 say "CREATING THE ENQUIRES FILE"
      select 2
      copy stru to &mar
      select 9
      use &mar
      do enquset
      do enqgen
    endif
    do addrec
  endif
  mloop=0
  return
  **
*!**************************************************************************
*!                     This procedure adds an empty record to a datafile.
*!   Procedure: ADDREC
*!
*!   Called by: GAMCONF       (procedure in SYS11.PRG)
*!
*!        Uses: &MAR.DBF
*!
*!**************************************************************************
  procedure addrec
  ****************
  select 2
  append from &mar
  return
  **
*!**************************************************************************
*!                     This procedure calculates the name of the files to restore
*!   Procedure: RESTVAR     from or save to by the "saveday" and "restore" procedures.
*!
*!   Called by: GAMCONF       (procedure in SYS11.PRG)
*!            : SAVEDAY       (procedure in SYS11.PRG)
*!
*!       Calls: DTOC()        (FOXBASE+ function)
*!            : CONFDATE()    (FOXBASE+ function)
*!            : LEFT()        (FOXBASE+ function)
*!            : SUBSTR()      (FOXBASE+ function)
*!
*!**************************************************************************
  procedure restvar
  *****************
  parameter vardate
  vardate=dtoc(vardate)
  if conffdate=confdate.and.(mmfactdate-1)<>confdate
    vardate="0007"
  else
    vardate=left(vardate,2)+substr(vardate,4,2)
  endif
  msys=vardate+"sys.dbf"
  mpla=vardate+"pla.dbf"
  mmar=vardate+"mar.dbf"
  mmac=vardate+"mac.dbf"
  mstk=vardate+"stk.dbf"
  mbil=vardate+"bil.dbf"
  mstu=vardate+"stu.dbf"
  msta=vardate+"sta.dbf"
  msalp=vardate+"salp.dbf"
  msalf=vardate+"salf.dbf"
  mstkp=vardate+"stkp.dbf"
  mmacp=vardate+"macp.dbf"
```

```
mplap=vardate+"plap.dbf"
return
**
*!*********************************************************************
*!                       This procedure calculates the screen position of the last
*!      Procedure: POSIT       enquire to be displayed.
*!
*!          Calls: .NOT.EOF()      (FOXBASE+ function)
*!               : EOF()           (FOXBASE+ function)
*!
*!*********************************************************************
procedure posit
***************
mposit=1
goto top
do while .not.eof()
  replace posit with mposit
  mposit=mposit+1
  if .not. eof()
    skip
  endif
enddo
return
**
*!*********************************************************************
*!                       This procedure checks the existance of previously saved
*!      Procedure: CHEKFILE    files in order to be restored.
*!
*!      Called by: GAMCONF         (procedure in SYS11.PRG)
*!
*!          Calls: FILE()          (FOXBASE+ function)
*!
*!*********************************************************************
procedure chekfile
******************
file1=file("&msys")
file2=file("&mpla")
file3=file("&mmar")
file4=file("&mmac")
file5=file("&mstk")
file6=file("&mbil")
file7=file("&mstu")
file8=file("&msalp")
file9=file("&msalf")
file10=file("&mstkp")
file11=file("&mmacp")
file12=file("&mplap")
file13=file("&msta")
if file1.and.file2.and.file3.and.file4.and.file5.and.file6.and.file7
  if file8.and.file9.and.file10.and.file11.and.file12.and.file13
    pfiles=.t.
  else
    pfiles=.f.
  endif
else
  pfiles=.f.
endif
**
*!*********************************************************************
*!                       This procedure creates the status file which is based on the
*!      Procedure: CREATSTAT  sales orders and is used by the planning department to do the
*!                       scheduling.
*!      Called by: RENSTAT         (procedure in SYS11.PRG)
*!
*!          Calls: .NOT.EOF()      (FOXBASE+ function)
*!               : CREATSTOR       (procedure in SYS11.PRG)
*!               : CREATREPL       (procedure in SYS11.PRG)
*!
*!           Uses: TEMPSTAT.DBF
*!
*!        Indexes: TEMPSTAT.IDX
*!
*!*********************************************************************
procedure creatstat
*******************
private mtottime,crcode,crquant,crstk,crdate
private crcdate,crresour,crcap,crtime,crsetup,crtottime
store 0 to mtottime,crcode,crquant,crstk,crdate
store 0 to crcdate,crresour,crcap,crtime,crsetup,crtottime
set exclusive on
select 7
use tempstat index tempstat
zap
use
set exclusive off
```

```
select 7
use tempstat index tempstat
select 2
goto top
do while .not.eof()
   if status="FIRM".and..not.delivered
      do creatstor
      do creatrepl
      select 2
   endif
   if .not.eof()
      skip
   endif
enddo
select 7
total to status on cdate fields quantit,tottime
use
return
**
*!*******************************************************************
*!                        This procedure freezes the departmental programs while the
*!     Procedure: RENSTAT     system is re-calculating the status datafile.
*!
*!     Called by: RUNFACT        (procedure in SYS11.PRG)
*!
*!         Calls: SYSTEM         (procedure in SYS11.PRG)
*!              : LTRIM()        (FOXBASE+ function)
*!              : STR()          (FOXBASE+ function)
*!              : VAL()          (FOXBASE+ function)
*!              : SYS()          (FOXBASE+ function)
*!              : CREATSTAT      (procedure in SYS11.PRG)
*!
*!*******************************************************************
procedure renstat
*****************
select 10
replace restatus with .t.
unlock
do while system("restatus")
enddo
var="LOCRS"+ltrim(str(tempdd))
public &var
&var=val(sys(2))
do creatstat
var="LOCRF"+ltrim(str(tempdd))
public &var
&var=val(sys(2))
select 10
replace restatus with .t.
unlock
return
**
*!*******************************************************************
*!                         This procedure updates the system datafile's fields
*!     Procedure: REPLSYS     after a restart from previously saved files has been done.
*!
*!     Called by: RESTART        (procedure in SYS11.PRG)
*!
*!          Uses: &MSYS
*!              : SYSTEM.DBF
*!
*!*******************************************************************
procedure replsys
*****************
set exclusive off
select 9
use &msys
select 10
replace;
startdate with i->startdate,;
factdate  with i->factdate,;
hour_rate with i->hour_rate,;
tot_overh with i->tot_overh,;
frequency with i->frequency,;
waittime  with i->waittime
replace;
throughput with i->throughput,;
assets     with i->assets,;
systock    with i->systock,;
enquperiod with i->enquperiod
unlock
select 9
use
select 10
use system
```

```
return
**
*!*********************************************************************
*!                          This procedure calculates the status datafile's fields
*!     Procedure: CREATSTOR    based on the existing sales orders.
*!
*!     Called by: CREATSTAT       (procedure in SYS11.PRG)
*!
*!         Calls: DTOC()          (FOXBASE+ function)
*!              : TRANSFORM()     (FOXBASE+ function)
*!              : INT()           (FOXBASE+ function)
*!              : MOD()           (FOXBASE+ function)
*!
*!*********************************************************************
procedure creatstor
*******************
store (enquantit*e->procestime)+f->setup to mtottime
store code to crcode
store enquantit to crquant
store d->stock to crstk
store qudeliver to crdate
store code+dtoc(qudeliver) to crcdate
store c->resour_cod to crresour
store c->capacity to crcap
store e->procestime to crtime
store f->setup to crsetup
store transform(int(mtottime/60),"9")+"h"+transform(mod(mtottime,60),"99")+"m" to crtottime
**
*!*********************************************************************
*!                          This procedure stores the values calculated by the "creatstor"
*!     Procedure: CREATREPL   procedure in the status datafile
*!
*!     Called by: CREATSTAT       (procedure in SYS11.PRG)
*!
*!*********************************************************************
procedure creatrepl
*******************
select 7
append blank
replace code with crcode
replace quantit with crquant
replace stock with crstk
replace date with crdate
replace cdate with crcdate
replace resource with crresour
replace capacity with crcap
replace time with crtime
replace setup with crsetup
replace tottime with crtottime
unlock
**
*!*********************************************************************
*!                          This procedure deletes the records which havn't been
*!     Procedure: DELETC    quoted before their delivery date.
*!
*!     Called by: REINDEX         (procedure in SYS11.PRG)
*!
*!*********************************************************************
procedure deletc
****************
goto top
delete for endeliver<mmfactdate.and.status="ENQU"
return
**
*: EOF: SYS11.ACT
```

System: SYS11.PRG
Author: JUAN IGNACIO IGARTUA
08/05/91 18:23:39
Tree Diagram

```
SYS11.PRG
|     SYSTEM.DBF (database)
|-----------ERROR() (function in ?)
|-----------MESSAGE() (function in ?)
|-----------ERR_FIX     (procedure in SYS11.PRG)
|     L----------RLOCK() (function in ?)
|-----------CTOD() (function in ?)
|-----------GAMCONF     (procedure in SYS11.PRG)
|     |     SYSTEM.DBF (database)
|     |     &MAR.DBF (database)
|     |-----------CTOD() (function in ?)
|     |-----------RESTVAR     (procedure in SYS11.PRG)
|     |     |----------DTOC() (function in ?)
|     |     |----------CONFDATE.AND.() (function in ?)
|     |     |----------LEFT() (function in ?)
|     |     L----------SUBSTR() (function in ?)
|     |-----------CHEKFILE     (procedure in SYS11.PRG)
|     |     L----------FILE() (function in ?)
|     |-----------DTOC() (function in ?)
|     |-----------LEFT() (function in ?)
|     |-----------SUBSTR() (function in ?)
|     |-----------RESTART     (procedure in SYS11.PRG)
|     |     |----------DTOC() (function in ?)
|     |     L----------REPLSYS     (procedure in SYS11.PRG)
|     |            &MSYS (database)
|     |            SYSTEM.DBF (database)
|     |-----------REINDEX     (procedure in SYS11.PRG)
|     |     |     SYSTEM.DBF (database)
|     |     |     PLANN.DBF (database)
|     |     |     MARKET.DBF (database)
|     |     |     MACHINE.DBF (database)
|     |     |     STOCK.DBF (database)
|     |     |     BILL.DBF (database)
|     |     |     SALPAST.DBF (database)
|     |     |     SALPERF.DBF (database)
|     |     |     MACHPAST.DBF (database)
|     |     |     PLAPAST.DBF (database)
|     |     |     STKPAST.DBF (database)
|     |     |     SETUP.DBF (database)
|     |     L----------DELETC     (procedure in SYS11.PRG)
|     |-----------RESET     (procedure in SYS11.PRG)
|     |     |----------INT() (function in ?)
|     |     |----------VAL() (function in ?)
|     |     |----------SUBSTR() (function in ?)
|     |     |----------DTOC() (function in ?)
|     |     |----------LEFT() (function in ?)
|     |     L----------RIGHT() (function in ?)
|     |-----------.NOT.FILE() (function in ?)
|     |-----------ENQUSET     (procedure in SYS11.PRG)
|     |     |----------EOF() (function in ?)
|     |     |----------LTRIM() (function in ?)
|     |     |----------STR() (function in ?)
|     |     L----------RECNO() (function in ?)
|     |-----------ENQGEN     (procedure in SYS11.PRG)
|     |     |----------LEN() (function in ?)
|     |     |----------VAL() (function in ?)
|     |     |----------RIGHT() (function in ?)
|     |     |----------TIME() (function in ?)
|     |     |----------ROUND() (function in ?)
|     |     |----------RND     (procedure in SYS11.PRG)
|     |     |     L----------MOD() (function in ?)
|     |     |----------SUBSTR() (function in ?)
|     |     |----------MMFACTDATE<() (function in ?)
|     |     |----------MMFACTDATE>() (function in ?)
|     |     |----------IIF() (function in ?)
|     |     |----------INT() (function in ?)
|     |     L----------STUFF() (function in ?)
|     L-----------ADDREC     (procedure in SYS11.PRG)
|           &MAR.DBF (database)
|-----------OPEN     (procedure in SYS11.PRG)
|     |-----------OPENRUN     (procedure in SYS11.PRG)
|     |     PLANN.DBF (database)
|     |     MARKET.DBF (database)
|     |     MACHINE.DBF (database)
|     |     STOCK.DBF (database)
|     |     BILL.DBF (database)
|     |     SETUP.DBF (database)
|     |     SYSTEM.DBF (database)
|     |
```

```
|  |-------OPENSAL      (procedure in SYS11.PRG)
|  |         BILL.DBF (database)
|  |         MARKET.DBF (database)
|  |         SALPAST.DBF (database)
|  |         SALPERF.DBF (database)
|  |         SYSTEM.DBF (database)
|  |-------OPENPLA      (procedure in SYS11.PRG)
|          PLANN.DBF (database)
|          MACHINE.DBF (database)
|          STOCK.DBF (database)
|          BILL.DBF (database)
|          MACHPAST.DBF (database)
|          PLAPAST.DBF (database)
|          STKPAST.DBF (database)
|          SETUP.DBF (database)
|          SYSTEM.DBF (database)
|-------MAINMENU      (procedure in SYS11.PRG)
   |-------FACTCONF      (procedure in SYS11.PRG)
   |   |-------MACHBROW      (procedure in SYS11.PRG)
   |       |-------CHR() (function  in ?)
   |       |-------REPLICATE() (function  in ?)
   |       |-------RECNO() (function  in ?)
   |       |-------MACH_DISP      (procedure in SYS11.PRG)
   |       |   |-------ROW() (function  in ?)
   |       |   |-------COL() (function  in ?)
   |       |-------GETKEY      (procedure in SYS11.PRG)
   |       |   |-------INKEY() (function  in ?)
   |       |   |-------UPPER() (function  in ?)
   |       |   |-------CHR() (function  in ?)
   |       |-------BOF() (function  in ?)
   |       |-------EOF() (function  in ?)
   |       |-------MACH_RSTR      (procedure in SYS11.PRG)
   |       |-------MACHGETS      (procedure in SYS11.PRG)
   |       |-------UPDATED() (function  in ?)
   |       |-------MACH_REPL      (procedure in SYS11.PRG)
   |       |   |-------EOF() (function  in ?)
   |       |-------MACH_STOR      (procedure in SYS11.PRG)
   |       |   |-------SPACE() (function  in ?)
   |       |-------NEWMACH      (procedure in SYS11.PRG)
   |       |   |-------MACHEDIT      (procedure in SYS11.PRG)
   |       |       |-------MACHGETS      (procedure in SYS11.PRG)
   |       |       |-------GETKEY      (procedure in SYS11.PRG)
   |       |       |   |-------INKEY() (function  in ?)
   |       |       |   |-------UPPER() (function  in ?)
   |       |       |   |-------CHR() (function  in ?)
   |       |       |-------INCLUDE      (procedure in SYS11.PRG)
   |       |           |-------MACH_REPL      (procedure in SYS11.PRG)
   |       |               |-------EOF() (function  in ?)
   |       |-------.NOT.DELETED() (function  in ?)
   |       |-------RESETUP      (procedure in SYS11.PRG)
   |           |-------EOF() (function  in ?)
   |           |-------FIELDD      (procedure in SYS11.PRG)
   |               |-------FCOUNT() (function  in ?)
   |               |-------FIELD() (function  in ?)
   |-------STARTGAM      (procedure in SYS11.PRG)
   |   |-------CONTINUE      (procedure in SYS11.PRG)
   |   |-------INT() (function  in ?)
   |   |-------VAL() (function  in ?)
   |   |-------SUBSTR() (function  in ?)
   |   |-------DTOC() (function  in ?)
   |   |-------LEFT() (function  in ?)
   |   |-------RIGHT() (function  in ?)
   |   |-------RUNFACT      (procedure in SYS11.PRG)
   |       |-------RUNSCREN      (procedure in SYS11.PRG)
   |       |-------SAYTIME0      (procedure in SYS11.PRG)
   |       |   |-------INKEY() (function  in ?)
   |       |   |-------VAL() (function  in ?)
   |       |   |-------SYS() (function  in ?)
   |       |   |-------FACTSTOP      (procedure in SYS11.PRG)
   |       |       |-------CONTINUE      (procedure in SYS11.PRG)
   |       |       |-------QUITGAME      (procedure in SYS11.PRG)
   |       |           |-------SET      (procedure in SYS11.PRG)
   |       |               |-------SPACE() (function  in ?)
   |       |               |-------EOF() (function  in ?)
   |       |               |-------.NOT.EOF() (function  in ?)
   |       |-------VAL() (function  in ?)
   |       |-------SYS() (function  in ?)
   |       |-------INT() (function  in ?)
   |       |-------LTRIM() (function  in ?)
   |       |-------STR() (function  in ?)
   |       |-------TIMECAL      (procedure in SYS11.PRG)
   |       |   |-------MOD() (function  in ?)
   |       |   |-------INT() (function  in ?)
   |       |   |-------STR() (function  in ?)
   |       |   |-------CTOD() (function  in ?)
```

```
│       │    ├──────SAYTIME1     (procedure in SYS11.PRG)
│       │    ├──────RECNO() (function  in ?)
│       │    ├──────.AND..NOT.EOF() (function  in ?)
│       │    ├──────TRIM() (function  in ?)
│       │    ├──────SYSTEM     (procedure in SYS11.PRG)
│       │    │    ├──────STR() (function  in ?)
│       │    │    └──────SELECT() (function  in ?)
│       │    ├──────.NOT.TRANSFORM() (function  in ?)
│       │    ├──────TRANSFORM() (function  in ?)
│       │    └──────SPACE() (function  in ?)
│       ├──────RUNPROD     (procedure in SYS11.PRG)
│       │    ├──────EOF() (function  in ?)
│       │    ├──────.NOT.EOF() (function  in ?)
│       │    ├──────MOD() (function  in ?)
│       │    ├──────INT() (function  in ?)
│       │    ├──────STR() (function  in ?)
│       │    ├──────NEWCLOCK<() (function  in ?)
│       │    └──────.AND..NOT.EOF() (function  in ?)
│       ├──────GOODSERV     (procedure in SYS11.PRG)
│       │    ├──────EOF() (function  in ?)
│       │    └──────QUDELIVER>ENDELIVER.AND.QUDELIVER<() (function  in ?)
│       ├──────SHIPPING     (procedure in SYS11.PRG)
│       │    ├──────EOF() (function  in ?)
│       │    ├──────.NOT.FOUND() (function  in ?)
│       │    └──────RECNO() (function  in ?)
│       ├──────RENSTAT     (procedure in SYS11.PRG)
│       │    ├──────SYSTEM     (procedure in SYS11.PRG)
│       │    │    ├──────STR() (function  in ?)
│       │    │    └──────SELECT() (function  in ?)
│       │    ├──────LTRIM() (function  in ?)
│       │    ├──────STR() (function  in ?)
│       │    ├──────VAL() (function  in ?)
│       │    ├──────SYS() (function  in ?)
│       │    └──────CREATSTAT     (procedure in SYS11.PRG)
│       │         │  TEMPSTAT.DBF (database)
│       │         ├──────.NOT.EOF() (function  in ?)
│       │         ├──────CREATSTOR     (procedure in SYS11.PRG)
│       │         │    ├──────DTOC() (function  in ?)
│       │         │    ├──────TRANSFORM() (function  in ?)
│       │         │    ├──────INT() (function  in ?)
│       │         │    └──────MOD() (function  in ?)
│       │         └──────CREATREPL     (procedure in SYS11.PRG)
│       └──────DAYREPOR     (procedure in SYS11.PRG)
│            ├──────LTRIM() (function  in ?)
│            ├──────STR() (function  in ?)
│            ├──────VAL() (function  in ?)
│            ├──────SYS() (function  in ?)
│            ├──────SALREPO     (procedure in SYS11.PRG)
│            │    ├──────SALPAST     (procedure in SYS11.PRG)
│            │    │    ├──────EOF() (function  in ?)
│            │    │    ├──────IIF() (function  in ?)
│            │    │    └──────SALPERF     (procedure in SYS11.PRG)
│            │    │         └──────EOF() (function  in ?)
│            │    ├──────ONTIME     (procedure in SYS11.PRG)
│            │    │    └──────EOF() (function  in ?)
│            │    └──────FREQUEN     (procedure in SYS11.PRG)
│            │         ├──────EOF() (function  in ?)
│            │         ├──────MIN() (function  in ?)
│            │         └──────INT() (function  in ?)
│            ├──────PLAREPO     (procedure in SYS11.PRG)
│            │    └──────PLAPAST     (procedure in SYS11.PRG)
│            │         ├──────EOF() (function  in ?)
│            │         ├──────.NOT.EOF() (function  in ?)
│            │         ├──────VAL() (function  in ?)
│            │         ├──────STR() (function  in ?)
│            │         ├──────LEFT() (function  in ?)
│            │         ├──────RIGHT() (function  in ?)
│            │         └──────CTOD() (function  in ?)
│            ├──────SYSTEM     (procedure in SYS11.PRG)
│            │    ├──────STR() (function  in ?)
│            │    └──────SELECT() (function  in ?)
│            ├──────.OR.SYSTEM() (function  in ?)
│            └──────REPOMENU     (procedure in SYS11.PRG)
│                 ├──────SAVEDAY     (procedure in SYS11.PRG)
│                 │    │  TEMP.DBF (database)
│                 │    └──────RESTVAR     (procedure in SYS11.PRG)
│                 │         ├──────DTOC() (function  in ?)
│                 │         ├──────CONFDATE.AND.() (function  in ?)
│                 │         ├──────LEFT() (function  in ?)
│                 │         └──────SUBSTR() (function  in ?)
│                 ├──────NEXTDAY     (procedure in SYS11.PRG)
│                 │    ├──────GAMCONF     (procedure in SYS11.PRG)
│                 │    │    │  SYSTEM.DBF (database)
│                 │    │    │  &MAR.DBF (database)
│                 │    │    ├──────CTOD() (function  in ?)
```

```
      RESTVAR      (procedure in SYS11.PRG)
         DTOC() (function in ?)
         CONFDATE.AND.() (function in ?)
         LEFT() (function in ?)
         SUBSTR() (function in ?)
      CHEKFILE     (procedure in SYS11.PRG)
         FILE() (function in ?)
      DTOC() (function in ?)
      LEFT() (function in ?)
      SUBSTR() (function in ?)
      RESTART      (procedure in SYS11.PRG)
         DTOC() (function in ?)
         REPLSYS      (procedure in SYS11.PRG)
       &MSYS (database)
       SYSTEM.DBF (database)
      REINDEX      (procedure in SYS11.PRG)
      SYSTEM.DBF (database)
      PLANN.DBF (database)
      MARKET.DBF (database)
      MACHINE.DBF (database)
      STOCK.DBF (database)
      BILL.DBF (database)
      SALPAST.DBF (database)
      SALPERF.DBF (database)
      MACHPAST.DBF (database)
      PLAPAST.DBF (database)
      STKPAST.DBF (database)
      SETUP.DBF (database)
         DELETC       (procedure in SYS11.PRG)
      RESET        (procedure in SYS11.PRG)
         INT() (function in ?)
         VAL() (function in ?)
         SUBSTR() (function in ?)
         DTOC() (function in ?)
         LEFT() (function in ?)
         RIGHT() (function in ?)
      .NOT.FILE() (function in ?)
      ENQUSET      (procedure in SYS11.PRG)
         EOF() (function in ?)
         LTRIM() (function in ?)
         STR() (function in ?)
         RECNO() (function in ?)
      ENQGEN       (procedure in SYS11.PRG)
         LEN() (function in ?)
         VAL() (function in ?)
         RIGHT() (function in ?)
         TIME() (function in ?)
         ROUND() (function in ?)
         RND          (procedure in SYS11.PRG)
            MOD() (function in ?)
         SUBSTR() (function in ?)
         MMFACTDATE<() (function in ?)
         MMFACTDATE>() (function in ?)
         IIF() (function in ?)
         INT() (function in ?)
         STUFF() (function in ?)
      ADDREC       (procedure in SYS11.PRG)
     &MAR.DBF (database)
   SETNEXT      (procedure in SYS11.PRG)
      EOF() (function in ?)
      .NOT.EOF() (function in ?)
      SPACE() (function in ?)
EXITGAME     (procedure in SYS11.PRG)
   SET      (procedure in SYS11.PRG)
      SPACE() (function in ?)
      EOF() (function in ?)
      .NOT.EOF() (function in ?)
EXITSYS      (procedure in SYS11.PRG)
   SET      (procedure in SYS11.PRG)
      SPACE() (function in ?)
      EOF() (function in ?)
      .NOT.EOF() (function in ?)
```

```
System: SYS11.PRG
Author: JUAN IGNACIO IGARTUA
08/05/91   18:23:25
Database Structure Summary

-------------------------------------------------------------------------

17 databases in the system
   SYSTEM.DBF
   TEMP.DBF
   PLANN.DBF
   MARKET.DBF
   MACHINE.DBF
   STOCK.DBF
   BILL.DBF
   SALPAST.DBF
   SALPERF.DBF
   MACHPAST.DBF
   PLAPAST.DBF
   STKPAST.DBF
   SETUP.DBF
   &MAR.DBF
   &MAR
   TEMPSTAT.DBF
   &MSYS
-------------------------------------------------------------------------

Structure for database : SYSTEM.DBF
Number of data records :       1
          Last updated : 07/01/91 at   9:17
Field  Field name  Type        Width    Dec    Start      End
    1  STARTDATE   Date            8               1        8
    2  FACTDATE    Date            8               9       16
    3  FACTTIME    Character       4              17       20
    4  SYS_STOP    Logical         1              21       21
    5  GAM_STOP    Logical         1              22       22
    6  HOUR_RATE   Numeric         7      2       23       29
    7  TOT_OVERH   Numeric        10      2       30       39
    8  NEWENQ      Logical         1              40       40
    9  NEWENQREC   Character       9              41       49
   10  FREQUENCY   Numeric         3              50       52
   11  GOODSERV    Logical         1              53       53
   12  REINDEX     Logical         1              54       54
   13  STATRESO    Character       4              55       58
   14  STATCODE    Character       4              59       62
   15  WAITTIME    Numeric         4              63       66
   16  RESTART     Logical         1              67       67
   17  RESTATUS    Logical         1              68       68
   18  THROUGHPUT  Numeric        10      2       69       78
   19  ASSETS      Numeric        10      2       79       88
   20  SYSTOCK     Numeric        10      2       89       98
   21  SALREPORT   Logical         1              99       99
   22  PLAREPORT   Logical         1             100      100
   23  ENQUPERIOD  Numeric         1             101      101
** Total **                      102

FoxDoc did not find any associated index files

Used by: SYS11.PRG
       : GAMCONF        (procedure in SYS11.PRG)
       : REINDEX        (procedure in SYS11.PRG)
       : OPENRUN        (procedure in SYS11.PRG)
       : OPENSAL        (procedure in SYS11.PRG)
       : OPENPLA        (procedure in SYS11.PRG)
       : REPLSYS        (procedure in SYS11.PRG)

-------------------------------------------------------------------------

Structure for database : TEMP.DBF
Number of data records :       1
          Last updated : 07/01/91 at   8:50
Field  Field name  Type        Width    Dec    Start      End
    1  STARTDATE   Date            8               1        8
    2  FACTDATE    Date            8               9       16
    3  FACTTIME    Character       4              17       20
    4  SYS_STOP    Logical         1              21       21
    5  GAM_STOP    Logical         1              22       22
    6  HOUR_RATE   Numeric         7      2       23       29
    7  TOT_OVERH   Numeric        10      2       30       39
    8  NEWENQ      Logical         1              40       40
    9  NEWENQREC   Character       9              41       49
   10  FREQUENCY   Numeric         3              50       52
   11  GOODSERV    Logical         1              53       53
```

12	REINDEX	Logical	1		54	54
13	STATRESO	Character	4		55	58
14	STATCODE	Character	4		59	62
15	WAITTIME	Numeric	4		63	66
16	RESTART	Logical	1		67	67
17	RESTATUS	Logical	1		68	68
18	THROUGHPUT	Numeric	10	2	69	78
19	ASSETS	Numeric	10	2	79	88
20	SYSTOCK	Numeric	10	2	89	98
21	SALREPORT	Logical	1		99	99
22	PLAREPORT	Logical	1		100	100
23	ENQUPERIOD	Numeric	1		101	101
** Total **			102			

FoxDoc did not find any associated index files

Used by: SAVEDAY (procedure in SYS11.PRG)

Structure for database : PLANN.DBF
Number of data records : 5
Last updated : 07/01/91 at 9:17

Field	Field name	Type	Width	Dec	Start	End
1	CODE	Character	4		1	4
2	DESCRIPT	Character	20		5	24
3	QUANTIT	Numeric	6	2	25	30
4	START_DATE	Date	8		31	38
5	START_TIME	Character	4		39	42
6	ORDER_NUM	Numeric	4		43	46
7	SEQUENCE	Numeric	4		47	50
8	RESOUR_COD	Character	4		51	54
9	PRODUCT_1	Numeric	10	2	55	64
10	PRODUCT_0	Numeric	10	2	65	74
11	ORD_STAT_1	Character	8		75	82
12	ORD_STAT_0	Character	8		83	90
13	FINIS_DATE	Date	8		91	98
14	FINIS_TIME	Character	4		99	102
15	NEED_TIME	Numeric	6	2	103	108
** Total **			109			

This database appears to be associated with index file(s):
: ORPLANN.IDX (RESOUR_COD+CHR(SEQUENCE))
: NAPLANN.IDX (CODE)

Used by: REINDEX (procedure in SYS11.PRG)
: OPENRUN (procedure in SYS11.PRG)
: OPENPLA (procedure in SYS11.PRG)

Structure for database : MARKET.DBF
Number of data records : 10
Last updated : 07/01/91 at 9:17

Field	Field name	Type	Width	Dec	Start	End
1	CODE	Character	4		1	4
2	DESCRIPT	Character	20		5	24
3	ENQUANTIT	Numeric	4		25	28
4	ENPRICE	Numeric	6	2	29	34
5	ENDATE	Date	8		35	42
6	ENTIME	Character	4		43	46
7	ENDELIVER	Date	8		47	54
8	QUPRICE	Numeric	6	2	55	60
9	QUDATE	Date	8		61	68
10	QUTIME	Character	4		69	72
11	QUDELIVER	Date	8		73	80
12	STATUS	Character	4		81	84
13	REALDELIV	Date	8		85	92
14	DELIVERED	Logical	1		93	93
15	SUCCSPRICE	Numeric	6	2	94	99
16	SUCCSDELIV	Numeric	6	2	100	105
17	WAIT	Numeric	4		106	109
** Total **			110			

This database appears to be associated with index file(s):
: DEMARK.IDX (QUDELIVER)
: WAIT.IDX (RECNO())
: SALDELIV.IDX (QUDELIVER)

Used by: REINDEX (procedure in SYS11.PRG)

: OPENRUN (procedure in SYS11.PRG)
: OPENSAL (procedure in SYS11.PRG)

Structure for database : MACHINE.DBF
Number of data records : 3
Last updated : 07/01/91 at 9:13

Field	Field name	Type	Width	Dec	Start	End
1	RESOUR_COD	Character	4		1	4
2	REDESCRIPT	Character	15		5	19
3	AVAIL	Character	3		20	22
4	T_AVAIL	Numeric	5		23	27
5	CAPACITY	Numeric	4		28	31
6	LASTSETUP	Character	7		32	38
7	SET_UP1	Numeric	6	2	39	44
8	SET_UP2	Numeric	6	2	45	50
9	SET_UP3	Numeric	6	2	51	56
10	SET_UP4	Numeric	6	2	57	62
11	SET_UP5	Numeric	6	2	63	68
** Total **			69			

This database appears to be associated with index file(s):
: RESOUR.IDX (RESOUR_COD)

Used by: REINDEX (procedure in SYS11.PRG)
: OPENRUN (procedure in SYS11.PRG)
: OPENPLA (procedure in SYS11.PRG)

Structure for database : STOCK.DBF
Number of data records : 5
Last updated : 07/01/91 at 9:13

Field	Field name	Type	Width	Dec	Start	End
1	CODE	Character	4		1	4
2	DESCRIPT	Character	20		5	24
3	STOCK	Numeric	8		25	32
** Total **			33			

This database appears to be associated with index file(s):
: CODESTK.IDX (CODE)

Used by: REINDEX (procedure in SYS11.PRG)
: OPENRUN (procedure in SYS11.PRG)
: OPENPLA (procedure in SYS11.PRG)

Structure for database : BILL.DBF
Number of data records : 5
Last updated : 07/01/91 at 9:09

Field	Field name	Type	Width	Dec	Start	End
1	CODE	Character	4		1	4
2	DESCRIPT	Character	20		5	24
3	MINDELIVDD	Numeric	2		25	26
4	MAXDELIVDD	Numeric	2		27	28
5	MAXQUANTIT	Numeric	6	2	29	34
6	MINQUANTIT	Numeric	6	2	35	40
7	STANDPRICE	Numeric	6	2	41	46
8	MATCOST	Numeric	6	2	47	52
9	TOT_ENQU	Numeric	4		53	56
10	PRICE_LIMT	Numeric	6	2	57	62
11	DISCOUNT	Numeric	4	2	63	66
12	QUANT_DISC	Numeric	6	2	67	72
13	DELIV_LIMT	Numeric	4		73	76
14	MRKT_PRICE	Numeric	6	2	77	82
15	MRKT_LIMT	Numeric	6	2	83	88
16	ONTIMEDELV	Numeric	6	2	89	94
17	ONTIMLIMT	Numeric	6	2	95	100
18	STANDHOURS	Numeric	6	2	101	106
19	OVERHEAD	Numeric	6	2	107	112
20	STAND_COST	Numeric	6	2	113	118
21	NUM_ENQU	Numeric	4		119	122
22	RESOUR_COD	Character	4		123	126
23	PROCESTIME	Numeric	5	2	127	131
24	SET_UP	Character	7		132	138
** Total **			139			

This database appears to be associated with index file(s):
: BILLNA.IDX (CODE)

Used by: REINDEX (procedure in SYS11.PRG)
: OPENRUN (procedure in SYS11.PRG)
: OPENSAL (procedure in SYS11.PRG)
: OPENPLA (procedure in SYS11.PRG)

Structure for database : SALPAST.DBF
Number of data records : 1
Last updated : 07/01/91 at 9:17

Field	Field name	Type	Width	Dec	Start	End
1	CODE	Character	4		1	4
2	DESCRIPT	Character	20		5	24
3	DATE	Date	8		25	32
4	AVE_ENQU	Numeric	6	2	33	38
5	AVE_QUOT	Numeric	6	2	39	44
6	PER_QUOT	Numeric	6	2	45	50
7	AVE_FIRM	Numeric	6	2	51	56
8	PER_FIRM	Numeric	6	2	57	62
9	OVR_DELTIM	Numeric	4	2	63	66
10	PER_ONTIME	Numeric	6	2	67	72
11	AVE_QPRICE	Numeric	6	2	73	78
12	PER_QPRICE	Numeric	6	2	79	84
13	PER_QDELIV	Numeric	6	2	85	90
14	PER_FPRICE	Numeric	6	2	91	96
15	PER_FDELIV	Numeric	6	2	97	102
16	DELIVERIES	Numeric	6		103	108
17	VOLUME	Numeric	10	2	109	118
** Total **			119			

This database appears to be associated with index file(s):
: CDSAL.IDX (CODE+DTOC(DATE))

Used by: REINDEX (procedure in SYS11.PRG)
: OPENSAL (procedure in SYS11.PRG)

Structure for database : SALPERF.DBF
Number of data records : 5
Last updated : 07/01/91 at 9:09

Field	Field name	Type	Width	Dec	Start	End
1	CODE	Character	4		1	4
2	DESCRIPT	Character	20		5	24
3	DATE	Date	8		25	32
4	AVE_ENQU	Numeric	6	2	33	38
5	AVE_QUOT	Numeric	6	2	39	44
6	PER_QUOT	Numeric	6	2	45	50
7	AVE_FIRM	Numeric	6	2	51	56
8	PER_FIRM	Numeric	6	2	57	62
9	OVR_DELTIM	Numeric	4	2	63	66
10	PER_ONTIME	Numeric	6	2	67	72
11	AVE_QPRICE	Numeric	6	2	73	78
12	PER_QPRICE	Numeric	6	2	79	84
13	PER_QDELIV	Numeric	6	2	85	90
14	PER_FPRICE	Numeric	6	2	91	96
15	PER_FDELIV	Numeric	6	2	97	102
16	DELIVERIES	Numeric	6		103	108
17	VOLUME	Numeric	10	2	109	118
** Total **			119			

This database appears to be associated with index file(s):
: CPRFSAL.IDX (CODE)

Used by: REINDEX (procedure in SYS11.PRG)
: OPENSAL (procedure in SYS11.PRG)

Structure for database : MACHPAST.DBF
Number of data records : 3
Last updated : 07/01/91 at 9:09

Field	Field name	Type	Width	Dec	Start	End
1	RESOUR_COD	Character	4		1	4
2	DESCRIPT	Character	15		5	19
3	CAPACITY	Numeric	4		20	23
4	PRODUCTION	Numeric	7	2	24	30

```
      5  PRODUCTIV    Numeric        6       2         31         36
      6  SETUPTIME    Numeric        7       2         37         43
      7  SETUPNUM     Numeric        2                 44         45
      8  DATE         Date           8                 46         53
** Total **                         54

This database appears to be associated with index file(s):
         : CDMACH.IDX   (RESOUR_COD+DTOC(DATE))

Used by: REINDEX          (procedure in SYS11.PRG)
       : OPENPLA          (procedure in SYS11.PRG)

----------------------------------------------------------------------

Structure for database : PLAPAST.DBF
Number of data records :         0
          Last updated : 07/01/91 at  9:09
Field  Field name   Type         Width     Dec     Start        End
    1  CODE         Character        4                 1          4
    2  DESCRIPT     Character       20                 5         24
    3  INCOMPLETE   Numeric          3                25         27
    4  STARTED      Numeric          3                28         30
    5  FINISHED     Numeric          3                31         33
    6  PRODUCTION   Numeric         10       2        34         43
    7  WIP          Numeric         10       2        44         53
    8  AVE BATCH    Numeric          6       2        54         59
    9  SETUPNUM     Numeric          3                60         62
   10  LEADTIME     Numeric          6       2        63         68
   11  STAND_COST   Numeric          6       2        69         74
   12  DATE         Date             8                75         82
** Total **                         83

This database appears to be associated with index file(s):
         : CDPLA.IDX   (CODE+DTOC(DATE))

Used by: REINDEX          (procedure in SYS11.PRG)
       : OPENPLA          (procedure in SYS11.PRG)

----------------------------------------------------------------------

Structure for database : STKPAST.DBF
Number of data records :         5
          Last updated : 07/01/91 at  9:09
Field  Field name   Type         Width     Dec     Start        End
    1  CODE         Character        4                 1          4
    2  DESCRIPT     Character       20                 5         24
    3  STOCK        Numeric          8                25         32
    4  DATE         Date             8                33         40
** Total **                         41

This database appears to be associated with index file(s):
         : CDSTK.IDX   (CODE+DTOC(DATE))

Used by: REINDEX          (procedure in SYS11.PRG)
       : OPENPLA          (procedure in SYS11.PRG)

----------------------------------------------------------------------

Structure for database : SETUP.DBF
Number of data records :         5
          Last updated : 07/01/91 at  9:09
Field  Field name   Type         Width     Dec     Start        End
    1  CODE         Character        4                 1          4
    2  TYPE         Character        7                 5         11
    3  SETUP        Numeric          6       2        12         17
** Total **                         18

This database appears to be associated with index file(s):
         : SETUP.IDX   (CODE)

Used by: REINDEX          (procedure in SYS11.PRG)
       : OPENRUN          (procedure in SYS11.PRG)
       : OPENPLA          (procedure in SYS11.PRG)

----------------------------------------------------------------------

&MAR.DBF is a macro unknown to FoxDoc
```

FoxDoc did not find any associated index files

Used by: GAMCONF (procedure in SYS11.PRG)
: ADDREC (procedure in SYS11.PRG)

&MAR is a macro unknown to FoxDoc

FoxDoc did not find any associated index files

Structure for database : TEMPSTAT.DBF
Number of data records : 7
Last updated : 07/01/91 at 9:13

Field	Field name	Type	Width	Dec	Start	End
1	CODE	Character	4		1	4
2	QUANTIT	Numeric	4		5	8
3	STOCK	Numeric	8		9	16
4	DATE	Date	8		17	24
5	RESOURCE	Character	4		25	28
6	CAPACITY	Numeric	4		29	32
7	TIME	Numeric	5	2	33	37
8	SETUP	Numeric	6	2	38	43
9	TOTTIME	Character	5		44	48
10	CDATE	Character	12		49	60
** Total **			61			

This database appears to be associated with index file(s):
: TEMPSTAT.IDX (CDATE)

Used by: CREATSTAT (procedure in SYS11.PRG)

&MSYS is a macro unknown to FoxDoc

FoxDoc did not find any associated index files

Used by: REPLSYS (procedure in SYS11.PRG)

System: SYS11.PRG
Author: JUAN IGNACIO IGARTUA
08/05/91 18:23:29
Data Dictionary

Field Name	Type	Len	Dec	Database
ASSETS	N	10	2	SYSTEM.DBF
				TEMP.DBF
AVAIL	C	3	0	MACHINE.DBF
AVE_BATCH	N	6	2	PLAPAST.DBF
AVE_ENQU	N	6	2	SALPERF.DBF
				SALPAST.DBF
AVE_FIRM	N	6	2	SALPAST.DBF
				SALPERF.DBF
AVE_QPRICE	N	6	2	SALPAST.DBF
				SALPERF.DBF
AVE_QUOT	N	6	2	SALPERF.DBF
				SALPAST.DBF
CAPACITY	N	4	0	TEMPSTAT.DBF
				MACHPAST.DBF
				MACHINE.DBF
CDATE	C	12	0	TEMPSTAT.DBF
CODE	C	4	0	SALPAST.DBF
				PLANN.DBF
				SETUP.DBF
				STKPAST.DBF
				BILL.DBF
				TEMPSTAT.DBF
				SALPERF.DBF
				PLAPAST.DBF
				MARKET.DBF
				STOCK.DBF
DATE	D	8	0	SALPAST.DBF
				PLAPAST.DBF
				TEMPSTAT.DBF
				MACHPAST.DBF
				SALPERF.DBF
				STKPAST.DBF
DELIVERED	L	1	0	MARKET.DBF
DELIVERIES	N	6	0	SALPAST.DBF
				SALPERF.DBF
DELIV_LIMT	N	4	0	BILL.DBF
DESCRIPT	C	20	0	STKPAST.DBF
				PLANN.DBF
				PLAPAST.DBF
				SALPAST.DBF
				STOCK.DBF
				MARKET.DBF
				BILL.DBF
				SALPERF.DBF
DESCRIPT	C	15	0	MACHPAST.DBF
DISCOUNT	N	4	2	BILL.DBF
ENDATE	D	8	0	MARKET.DBF
ENDELIVER	D	8	0	MARKET.DBF
ENPRICE	N	6	2	MARKET.DBF
ENQUANTIT	N	4	0	MARKET.DBF
ENQUPERIOD	N	1	0	SYSTEM.DBF
				TEMP.DBF
ENTIME	C	4	0	MARKET.DBF
FACTDATE	D	8	0	SYSTEM.DBF
				TEMP.DBF
FACTTIME	C	4	0	SYSTEM.DBF
				TEMP.DBF
FINISHED	N	3	0	PLAPAST.DBF
FINIS_DATE	D	8	0	PLANN.DBF
FINIS_TIME	C	4	0	PLANN.DBF
FREQUENCY	N	3	0	TEMP.DBF

				SYSTEM.DBF
GAM_STOP	L	1	0	TEMP.DBF
				SYSTEM.DBF
GOODSERV	L	1	0	TEMP.DBF
				SYSTEM.DBF
HOUR_RATE	N	7	2	TEMP.DBF
				SYSTEM.DBF
INCOMPLETE	N	3	0	PLAPAST.DBF
LASTSETUP	C	7	0	MACHINE.DBF
LEADTIME	N	6	2	PLAPAST.DBF
MATCOST	N	6	2	BILL.DBF
MAXDELIVDD	N	2	0	BILL.DBF
MAXQUANTIT	N	6	2	BILL.DBF
MINDELIVDD	N	2	0	BILL.DBF
MINQUANTIT	N	6	2	BILL.DBF
MRKT_LIMT	N	6	2	BILL.DBF
MRKT_PRICE	N	6	2	BILL.DBF
NEED_TIME	N	6	2	PLANN.DBF
NEWENQ	L	1	0	TEMP.DBF
				SYSTEM.DBF
NEWENQREC	C	9	0	TEMP.DBF
				SYSTEM.DBF
NUM_ENQU	N	4	0	BILL.DBF
ONTIMEDELV	N	6	2	BILL.DBF
ONTIMLIMT	N	6	2	BILL.DBF
ORDER_NUM	N	4	0	PLANN.DBF
ORD_STAT_0	C	8	0	PLANN.DBF
ORD_STAT_1	C	8	0	PLANN.DBF
OVERHEAD	N	6	2	BILL.DBF
OVR_DELTIM	N	4	2	SALPAST.DBF
				SALPERF.DBF
PER_FDELIV	N	6	2	SALPAST.DBF
				SALPERF.DBF
PER_FIRM	N	6	2	SALPAST.DBF
				SALPERF.DBF
PER_FPRICE	N	6	2	SALPAST.DBF
				SALPERF.DBF
PER_ONTIME	N	6	2	SALPAST.DBF
				SALPERF.DBF
PER_QDELIV	N	6	2	SALPAST.DBF
				SALPERF.DBF
PER_QPRICE	N	6	2	SALPERF.DBF
				SALPAST.DBF
PER_QUOT	N	6	2	SALPERF.DBF
				SALPAST.DBF
PLAREPORT	L	1	0	TEMP.DBF
				SYSTEM.DBF
PRICE_LIMT	N	6	2	BILL.DBF
PROCESTIME	N	5	2	BILL.DBF
PRODUCTION	N	7	2	MACHPAST.DBF
PRODUCTION	N	10	2	PLAPAST.DBF
PRODUCTIV	N	6	2	MACHPAST.DBF
PRODUCT_0	N	10	2	PLANN.DBF
PRODUCT_1	N	10	2	PLANN.DBF
QUANTIT	N	6	2	PLANN.DBF
QUANTIT	N	4	0	TEMPSTAT.DBF
QUANT_DISC	N	6	2	BILL.DBF
QUDATE	D	8	0	MARKET.DBF
QUDELIVER	D	8	0	MARKET.DBF
QUPRICE	N	6	2	MARKET.DBF
QUTIME	C	4	0	MARKET.DBF
REALDELIV	D	8	0	MARKET.DBF
REDESCRIPT	C	15	0	MACHINE.DBF
REINDEX	L	1	0	TEMP.DBF
				SYSTEM.DBF
RESOURCE	C	4	0	TEMPSTAT.DBF
RESOUR_COD	C	4	0	PLANN.DBF
				MACHPAST.DBF
				BILL.DBF
				MACHINE.DBF

RESTART	L	1	0	TEMP.DBF
				SYSTEM.DBF
RESTATUS	L	1	0	TEMP.DBF
				SYSTEM.DBF
SALREPORT	L	1	0	SYSTEM.DBF
				TEMP.DBF
SEQUENCE	N	4	0	PLANN.DBF
SETUP	N	6	2	SETUP.DBF
				TEMPSTAT.DBF
SETUPNUM	N	2	0	MACHPAST.DBF
SETUPNUM	N	3	0	PLAPAST.DBF
SETUPTIME	N	7	2	MACHPAST.DBF
SET_UP	C	7	0	BILL.DBF
SET_UP1	N	6	2	MACHINE.DBF
SET_UP2	N	6	2	MACHINE.DBF
SET_UP3	N	6	2	MACHINE.DBF
SET_UP4	N	6	2	MACHINE.DBF
SET_UP5	N	6	2	MACHINE.DBF
STANDHOURS	N	6	2	BILL.DBF
STANDPRICE	N	6	2	BILL.DBF
STAND_COST	N	6	2	PLAPAST.DBF
				BILL.DBF
STARTDATE	D	8	0	TEMP.DBF
				SYSTEM.DBF
STARTED	N	3	0	PLAPAST.DBF
START_DATE	D	8	0	PLANN.DBF
START_TIME	C	4	0	PLANN.DBF
STATCODE	C	4	0	SYSTEM.DBF
				TEMP.DBF
STATRESO	C	4	0	SYSTEM.DBF
				TEMP.DBF
STATUS	C	4	0	MARKET.DBF
STOCK	N	8	0	STKPAST.DBF
				STOCK.DBF
				TEMPSTAT.DBF
SUCCSDELIV	N	6	2	MARKET.DBF
SUCCSPRICE	N	6	2	MARKET.DBF
SYSTOCK	N	10	2	TEMP.DBF
				SYSTEM.DBF
SYS_STOP	L	1	0	SYSTEM.DBF
				TEMP.DBF
THROUGHPUT	N	10	2	SYSTEM.DBF
				TEMP.DBF
TIME	N	5	2	TEMPSTAT.DBF
TOTTIME	C	5	0	TEMPSTAT.DBF
TOT_ENQU	N	4	0	BILL.DBF
TOT_OVERH	N	10	2	SYSTEM.DBF
				TEMP.DBF
TYPE	C	7	0	SETUP.DBF
T_AVAIL	N	5	0	MACHINE.DBF
VOLUME	N	10	2	SALPERF.DBF
				SALPAST.DBF
WAIT	N	4	0	MARKET.DBF
WAITTIME	N	4	0	TEMP.DBF
				SYSTEM.DBF
WIP	N	10	2	PLAPAST.DBF

PLANNING/CONTROL SUB-PROGRAM

PLA4.PRG

```
*:*****************************************************************************
*:                     This program is the one that simulates all the activities
*:      Program: PLA4.PRG   performed by the planning department. Together with the Sales
*:                     and System ones, it conforms the Nottingham Polytechnic's
*:       System: PLA4.PRG                    gaming-simulation.
*:       Author: JUAN IGNACIO IGARTUA
*:     Copyright (c) 1991, NOTTINGHAM POLYTECHNIC
*:  Last modified: 08/05/91     18:06
*:
*:  Procs & Fncts: ERR_FIX
*:               : SYSTEM()
*:               : OPEN
*:               : CHEKSTOP
*:               : INPTMPS
*:               : MAINMENU
*:               : PLA_BROW
*:               : NEWPLAN
*:               : PLEDIT
*:               : PL_GETS
*:               : INCLUDE
*:               : PL_REPL
*:               : QUIT
*:               : PL_STOR
*:               : CALORD
*:               : COUNTDEL
*:               : GETKEY
*:               : SAM_DISP
*:               : STA_DISP
*:               : PLA_AREA0
*:               : PLA_AREA1
*:               : PLA_CODE
*:               : PLA_RESO
*:               : RESOURCE
*:               : FILT_CODE
*:               : FILT_RESO
*:               : POSITION
*:               : REPLANN
*:               : PLANREPO
*:               : PLAREPO
*:               : PLAAVE
*:               : MACHAVE
*:               : STKAVE
*:               : PRT
*:               : PLAPAST
*:               : MACHPAST
*:               : STKPAST
*:               : CREANEW
*:
*:         Calls: ERROR()        (FOXBASE+ function)
*:               : MESSAGE()      (FOXBASE+ function)
*:               : ERR_FIX        (procedure in PLA4.PRG)
*:               : REPLICATE()    (FOXBASE+ function)
*:               : CHR()          (FOXBASE+ function)
*:               : SYSTEM()       (function  in PLA4.PRG)
*:               : OPEN           (procedure in PLA4.PRG)
*:               : SELECT()       (FOXBASE+ function)
*:               : CHEKSTOP       (procedure in PLA4.PRG)
*:               : INPTMPS        (procedure in PLA4.PRG)
*:
*:          Uses: SYSTEM.DBF
*:
*:     Documented 08/05/91 at 18:26              FoxDoc  version 1.0
*:*****************************************************************************
     *
     set date british
     set procedure to pla4.prg
     on error do err_fix with error(),message()
     set default to v:
     *
     set console on
     set escape on
     set delete on
     set status off
     set talk off
     set bell off
     set scoreboard off
     set heading off
     *
     public pgdn,returnkey,pgup,delrecord,recnum,choice,promptrow,promptbar,;
     oldrecnum,screenatr,statusatr,windowatr,phrase,phras,nonew,empty
     public promptatr,hiliteatr,filtreso,filtcode,reschoice,actfilt,lphrase;
     codchoice,rowbottom,rowprompt,listlast,reccord,lfiltcode,lfiltreso
     public morder_num,mresour_co,mneedtim,mquantit,mord_stat_1,mord_stat_0,ctdel,;
     mdeliver,mcode,mdescript,mfound,pout,indsel,indmps,mpspout,stcpout,;
```

```
mproduct_1,mproduct_0,signal,srnlines,prtlines,r,c,msequence,pfilt
public sys_start,signal1,statpage,pause,start,daystat,editing
**
srnlines=6
prtlines=55
start=.f.
signal=.f.
signal1=.f.
sys_start=.t.
pause=.f.
daystat=.f.
nonew=.f.
empty=.f.
editing=.f.
*
 store "orplann"  to indmps
 store srnlines   to mpspout
 store srnlines   to stcpout
 store "namark"   to indsel
 store srnlines  to pout
 *
 screenatr="R+/N,N/W"
 statusatr="BU/N,N/W"
 windowatr="R+/N,N/W"
 promptatr="GR+/N,N/W"
 hiliteatr="N/W"
 *
 rowbottom = 20
 rowprompt = rowbottom + 3
 *
 store 0 to recnum,oldrecnum
 store " " to choice
 store "1" to reschoice
 store "0" to codchoice
 store 0 to msequence
 promptrow=22
 promptbar=replicate(chr(196),80)
 pgdn=chr(3)
 pgup=chr(18)
 returnkey=chr(13)
 delrecord=chr(7)
 signal=.f.
 **
 set exclusive off
 select 10
 use system
 **
 scr=.f.
 do while system("reindex")  && wait until the reindex is finished
    if .not.scr              && in the system program.
       set color to bu+/n
       @ 7,24 to 12,54 double
       set color to gr+/n
       @ 9,25 say "REINDEXING THE MASTER SYSTEM"
       @ 10,34 say "PLEASE WAIT"
       scr=.t.
    endif
 enddo
 **
 set color to
 clear
 set color to bu+/n
 @ 7,24 to 12,54 double
 set color to gr+/n
 @ 9,25 say "SETTING THE PLANNING SYSTEM"
 @ 10,34 say "PLEASE WAIT"
 **
 do open
 set view to pladep
 **
 do chekstop with select()
 set color to
 clear
 do inptmps
 *
*!*************************************************************************
*!                  This is the main procedure that will call to very
*!     Procedure: MAINMENU    different activities (Display/Input or Exit), depending
*!                  on the menu choice selected by the player.
*!        Calls: SELECT()        (FOXBASE+ function)
*!             : CHEKSTOP        (procedure in PLA4.PRG)
*!             : INPTMPS         (procedure in PLA4.PRG)
*!             : QUIT            (procedure in PLA4.PRG)
*!
*!*************************************************************************
```

```
procedure mainmenu
******************
do while .t.
   set color to
   clear
   m_menu=0
   set color to g/n
   @ 3,16,19,62 box "╔═╗║╝═╚║"
   set color to gr/n
   @ 5,26 to 7,53 double
   set color to bu/n
   @  6,34 say "PLANNING WORKAREA"
   set color to br/n
   @ 11,24 to 14,54
   set color to bg/n
   @ 12,27 prompt "1.DISPLAY/INPUT OF PLANN."
   @ 13,27 prompt "2.EXIT.                  "
   menu to m_menu
   do chekstop with select()
   do case
   case m_menu=1
      do inptmps
   case m_menu=2
      do quit
      set color to
      exit
   endcase
enddo
*
*!*****************************************************************************
*!                  This procedure calls the display/input procedure used to edit
*!     Procedure: INPTMPS  the scheduling, based on the information obtained from orders
*!                  accepted by the sales department.
*!     Called by: PLA4.PRG
*!              : MAINMENU       (procedure in PLA4.PRG)
*!
*!         Calls: PLA_BROW       (procedure in PLA4.PRG)
*!
*!*****************************************************************************
procedure inptmps
*****************
clear
select 5
goto top
filtreso=resour_cod
filtcode="ALL"
lfiltreso="ALL"
select 3
do pla_brow
return
*
*!*****************************************************************************
*!                  This procedure displays the scheduling, as well as the sale
*!     Procedure: PLA_BROW  information, allowing the edition of the former. It is based
*!                                on the "browse" algorithm which
*!     Called by: INPTMPS       (procedure in PLA4.PRG)  is thoroughly explained in one of
*!                                the thesis's sections.
*!         Calls: CHR()          (FOXBASE+ function)
*!              : RECNO()        (FOXBASE+ function)
*!              : FILT_RESO      (procedure in PLA4.PRG)
*!              : .NOT.EOF()     (FOXBASE+ function)
*!              : SAM_DISP       (procedure in PLA4.PRG)
*!              : EOF()          (FOXBASE+ function)
*!              : STA_DISP       (procedure in PLA4.PRG)
*!              : SYSTEM()       (function  in PLA4.PRG)
*!              : GETKEY         (procedure in PLA4.PRG)
*!              : BOF()          (FOXBASE+ function)
*!              : CREANEW        (procedure in PLA4.PRG)
*!              : .AND..NOT.SYSTE(FOXBASE+ function)
*!              : POSITION       (procedure in PLA4.PRG)
*!              : FILT_CODE      (procedure in PLA4.PRG)
*!
*!*****************************************************************************
procedure pla_brow
******************
* Notes...: BROWSE program for PLANN.DBF
*
private pancol,panmax,panlast,recnumtop,recnumlast,skiprecs
private home,endkey,uparrow,downarrow,leftarrow,rightarrow
private row,rowtop,rowbottom,rowprompt,keystrokes,pagepaint
private isedited,source,target,tarseq,sourseq,keyst1,keyst2
private keyst3,press1,press2,lastdel
* ---Initialize constants.
home = chr(1)
endkey = chr(6)
```

```
uparrow = chr(5)
downarrow = chr(24)
leftarrow = chr(19)
rightarrow = chr(4)
keystrokes = uparrow+downarrow+home+leftarrow+;
rightarrow+endkey+pgdn+pgup
keyst1="RNCS"+keystrokes+delrecord+returnkey
keyst2="R"+delrecord+returnkey
keyst3="PC"+keystrokes
rowtop = 1
rowbottom = 20
rowprompt = rowbottom + 3
skiprecs = rowbottom - rowtop + 1
goto top
* ---Initialize local variables.
press1=" "
press2="P"
row = rowtop
recnum = recno()
recnumtop = recnum
pagepaint = .t.
statpage = .f.
isedited = .f.
lastdel=.f.
pancol = 1
panlast = 1
panmax = 2
* ---Perform BROWSE.
set color to &screenatr
clear
*
do filt_reso with .t.
* ---The following loop is really a "REPEAT/UNTIL <cond>".
do while .t.
    if pagepaint
        if press2<>press1.or.statpage
            goto top
            recnumtop=recno()
            press1=press2
        endif
        recnum = recno()
        if .not.eof()
            goto recnumtop
        endif
        if press2="P"
            select 3
            do sam_disp with (rowtop),skiprecs
            goto top
            if eof()
                empty=.t.
            else
                if empty
                    empty=.f.
                endif
            endif
        else
            select 2
            do sta_disp with (rowtop),skiprecs
        endif
        if .not.eof().or.select()=2
            goto recnum
        endif
        if pancol = panlast
            * ---Reposition record pointer when repainting current page.
            row = rowtop
        endif
        panlast = pancol
        pagepaint = .f.
    endif
    set color to &promptatr
    @ rowprompt-1,0 say promptbar
    if press2="P"
        @ rowprompt,0 clear
        if empty
            @ rowprompt,0 say ;
            "NO RECORDS AVAILABLE:  (R)esource  (N)ew  (S)tatus "
        else
            if system("plareport")
                @ rowprompt,0 say ;
                "BROWSE:  (E)xit  (N)ew  (C)ode  (R)esource  (S)tatus  <Arrows>  <Del>  <Return> "
            else
                @ rowprompt,0 say ;
                "BROWSE:  (N)ew  (C)ode  (R)esource  (S)tatus  <Arrows>  <Del>  <Return> "
            endif
        endif
```

```
else
  @ rowprompt,0 clear
  if system("plareport")
    @ rowprompt,0 say ;
    "STATUS:  (E)xit  (P)lann  (C)ode  <Arrows> "
  else
    @ rowprompt,0 say ;
    "STATUS:  (P)lann  (C)ode  <Arrows> "
  endif
endif
set color to bu+,n
@ rowprompt+1,18 say "CURRENT SELECTION: CODE="
?? filtcode
@ rowprompt+1,47 say " RESOURCE="
?? filtreso
set color to &promptatr
@ row,0 say chr(16)
if press2="P"
  if empty
    do getkey with choice,"RNS"
  else
    do getkey with choice,keyst1
  endif
else
  do getkey with choice,keyst3
endif
* ---Reposition record pointer.
do while choice $ uparrow+downarrow
  @ row,0 say " "
  if choice = uparrow
    skip -1
    do case
    case bof()
      goto top
    case row > rowtop
      row = row - 1
    otherwise
      recnumtop=recno()
      * ---Scroll window down.
      scroll rowtop,0,rowbottom,79,-1
      if press2="P"
        do sam_disp with row,1
      else
        do sta_disp with row,1
      endif
    endcase
  else
    skip
    do case
    case eof()
      goto bottom
    case row < rowbottom
      row = row + 1
    otherwise
      * ---Adjust top-of-page record pointer.
      recnum=recno()
      goto recnumtop
      skip
      recnumtop=recno()
      goto recnum
      * ---Scroll window up.
      scroll rowtop,0,rowbottom,79,1
      if press2="P"
        do sam_disp with row,1
      else
        do sta_disp with row,1
      endif
    endcase
  endif
  set color to &promptatr
  @ row,0 say chr(16)
  if press2="P"
    do getkey with choice,keyst1
  else
    do getkey with choice,keyst3
  endif
enddo
* ---Prompt line selections.
do case
case choice = "N"
  do creanew
case choice = returnkey
  if ord_stat_1="NO_START".and..not.system("SYS_STOP")
    @ rowprompt,0 clear
    set color to &promptatr
```

```
        @ rowprompt,0 say ;
        "SELECT :  (E)dit/Re-edit  (R)eplan "
        do getkey with choice,"ER"
        do case
        case choice="E"
            * --- Store, Read and Replace the quantity of an existing order.
            store quantit       to mquantit
            @ row,c+26 get mquantit     picture "@Z 999" valid mquantit<>0
            read
            replace quantit with mquantit
            unlock
            do sam_disp with row,1
        case choice="R"
            do position
            pagepaint=.t.
        endcase
        endif
    case choice = delrecord
        if ord_stat_1="NO_START".and..not.system("SYS_STOP")
        * ---Delete the record.
        delete
        * ---Adjust the record position
        skip
            if eof()
            skip -1
            row=row-1
            r=r-1
            lastdel=.t.
            else
            lastdel=.f.
            endif
            if recno()=recnumtop
            recnumtop=recno()
            endif
        recnum=recno()
        * ---Scroll window up.
            if lastdel
            scroll row+1,0,rowbottom,79,1
            else
            scroll row,0,rowbottom,79,1
            endif
        goto recnumtop
        skip (skiprecs-1)
        do sam_disp with rowbottom,1
        goto recnum
        endif
    case choice = pgdn
        if .not. eof()
        goto recnumtop
        skip skiprecs
            if eof()
            goto bottom
            endif
        recnumtop = recno()
        pagepaint = .t.
        endif
    case choice = pgup
        if .not. bof()
        goto recnumtop
        skip -skiprecs
            if bof()
            goto top
            endif
        recnumtop = recno()
        pagepaint = .t.
        endif
    case choice = "C"
    lphrase=phrase
    lfiltcode=filtcode
    * ---Set FILTER for the Codes.
    do filt_code with .f.
    case choice = "R"
    lphrase=phrase
    lfiltreso=filtreso
    * ---Set FILTER for the Resources.
    do filt_reso with .f.
    case choice = "S"
    press1=press2
    press2=choice
    pfilt=actfilt
    pfiltcode=filtcode
    pcodchoice=codchoice
    select 2
    set filter to &actfilt
    goto top
```

```
          if .not.eof()
            pagepaint = .t.
          else
            press1=press2
            press2="P"
            select 3
          endif
      case choice = "P"
        press1=press2
        press2=choice
        filtcode=pfiltcode
        actfilt=pfilt
        codchoice=pcodchoice
        select 3
        set filter to &actfilt
        pagepaint = .t.
      case choice = home
        pagepaint = (pancol < > 1)
        pancol = 1
      case choice = endkey
        pagepaint = (pancol < > panmax)
        pancol = panmax
      endcase
    enddo
  set filter to
  goto top
  return
  *
*!*****************************************************************************
*!                  This procedure allows the input of the article code to be
*!     Procedure: NEWPLAN    scheduled and calls the edition procedures if that input
*!                  code exists, exiting the procedure otherwise.
*!     Called by: CREANEW        (procedure in PLA4.PRG)
*!
*!        Calls: FOUND()         (FOXBASE+ function)
*!             : PL_STOR         (procedure in PLA4.PRG)
*!             : PLEDIT          (procedure in PLA4.PRG)
*!
*!*****************************************************************************
  procedure newplan
  *****************
  mcode="     "
  editing=.t.
  do while .t.
    @ r,c get mcode picture "@!NNNN" valid mcode< >"     ".and.mcode$phrase
    read
    select 1
    find &mcode
    if found()
      store descript to mdescript
      store resour_cod to mresour_co
      select 3
      do pl_stor
      do pledit
      if choice=returnkey.or.choice=delrecord
        editing=.f.
        exit
      endif
    else
      select 3
      clear
      @ 8,33 say "INEXISTENT ARTICLE"
      wait
      exit
    endif
  enddo
  return
  **
*!*****************************************************************************
*!                  This procedure calls the procedures responsible for the
*!     Procedure: PLEDIT    input of the schedule information and the replace of it in the
*!                  <PLAN> datafile.
*!     Called by: NEWPLAN        (procedure in PLA4.PRG)
*!
*!        Calls: PL_GETS         (procedure in PLA4.PRG)
*!             : GETKEY          (procedure in PLA4.PRG)
*!             : RETURNKEY()     (FOXBASE+ function)
*!             : INCLUDE         (procedure in PLA4.PRG)
*!             : DELRECORD()     (FOXBASE+ function)
*!
*!*****************************************************************************
  procedure pledit
  ****************
  choice=" "
  do pl_gets
```

```
set color to &promptatr
@ rowprompt,0 clear
@ rowprompt,0 say ;
"NEW ORDER:   <Del> without saving  (R)ewrite   <Return> saving "
do getkey with choice,keyst2
if choice=returnkey.and..not.system("sys_stop")
   do include
else
   if choice=delrecord.and..not.system("sys_stop")
      nonew=.t.
   endif
endif
return
*
*!*****************************************************************
*!                          This procedure allows the input of the schedule
*!     Procedure: PL_GETS     information attached to each order.
*!
*!     Called by: PLEDIT          (procedure in PLA4.PRG)
*!
*!*****************************************************************
procedure pl_gets
*****************
set color to n/w
@ r,c+5  say mdescript
@ r,c+26 get mquantit    picture "@Z 999" valid mquantit<>0
read
mneedtim=mquantit*a->procestime
@ r,c+35 say mneedtim    picture "999.99"
@ r,c+47 say mresour_co  picture "@!NNNN"
@ r,c+52 say mproduct_1
@ r,c+66 say mord_stat_1
return
**
*!*****************************************************************
*!                          This procedure calls the procedures that will replace
*!     Procedure: INCLUDE     the input schedule information in the <PLAN> datafile.
*!
*!     Called by: PLEDIT          (procedure in PLA4.PRG)
*!
*!         Calls: CALORD          (procedure in PLA4.PRG)
*!              : RECNO()         (FOXBASE+ function)
*!              : PL_REPL         (procedure in PLA4.PRG)
*!
*!*****************************************************************
procedure include
*****************
do calord
if reccount()=0
   recnumtop=recno()
endif
append blank
do pl_repl
return
**
*!*****************************************************************
*!                          This procedure replaces the input schedule information
*!     Procedure: PL_REPL     in the <PLAN> datafile.
*!
*!     Called by: INCLUDE         (procedure in PLA4.PRG)
*!
*!         Calls: EOF()           (FOXBASE+ function)
*!
*!*****************************************************************
procedure pl_repl
*****************
if .not. eof()
   * ---Replace only if there is an available record
   replace;
   code        with mcode,;
   descript    with mdescript,;
   order_num   with morder_num,;
   resour_cod with mresour_co,;
   quantit     with mquantit
   replace;
   need_time   with mneedtim,;
   ord_stat_1 with mord_stat_1,;
   ord_stat_0 with mord_stat_0,;
   product_1   with mproduct_1,;
   sequence    with msequence
   unlock
endif
return
*
*!*****************************************************************
```

```
*!                          This procedure exits the Sales Department's program
*!      Procedure: QUIT      before it has been started.
*!
*!      Called by: MAINMENU        (procedure in PLA4.PRG)
*!
*!*****************************************************************
    procedure quit
    **************
    clear
    close all
    set color to
    set scoreboard on
    set bell on
    set talk on
    return
    *
*!*****************************************************************
*!                          This procedure stores default values in the schedule
*!      Procedure: PL_STOR    variables to be edit.
*!
*!      Called by: NEWPLAN         (procedure in PLA4.PRG)
*!
*!*****************************************************************
    procedure pl_stor
    *****************
    store 0                     to mquantit
    store 0                     to mproduct_1
    store 0                     to mproduct_0
    store 0                     to mneedtim
    store "NO_START"                to mord_stat_1
    store "NO_START"                to mord_stat_0
    return
    *
*!*****************************************************************
*!                          This procedure calculates the order number and the sequence
*!      Procedure: CALORD     of the new order to be edit.
*!
*!      Called by: INCLUDE         (procedure in PLA4.PRG)
*!
*!          Calls: RECCOUNT()      (FOXBASE+ function)
*!
*!*****************************************************************
    procedure calord
    ****************
    store reccount()  to morder_num
    goto bottom
    store sequence+1 to msequence
    return
    *
*!*****************************************************************
*!                          This procedure manages the possible conflicts that could
*!      Procedure: ERR_FIX    arise from the fact that we are working in a multi-user
*!                          environment. Its functioning is thoroughly explained in one
*!      Called by: PLA4.PRG   of the thesi's sections.
*!
*!          Calls: RLOCK()          (FOXBASE+ function)
*!
*!*****************************************************************
    procedure err_fix
    *****************
    parameters errnum,mess
    **
    ** Error: File in use by another.
    if errnum=108
       save screen to screen
       set color to gr+/bu
       @ 7,17 clear to 15,61
       @ 7,17 to 15,61 double
       set color to r*+/bu
       @ 11,24 say "Please wait to append a record."
       @ 12,26 say "Press any key to continue."
       read
       restore screen from screen
       retry
    endif
    **
    ** Error: Record in use by another.
    if errnum=109
       save screen to screen
       time=0
       set color to gr+/bu
       @ 7,17 clear to 15,61
       @ 7,17 to 15,61 double
       set color to r*+/bu
       @ 11,26 clear to 11,54
```

```
      @ 11,26 say mess
      @ 12,26 say "Press any key to continue."
      read
      do while .not. rlock().and.time<1000
         time=time+1
      enddo
      if time<1000
         restore screen from screen
         retry
      else
         set color to gr+/bu
         @ 7,17 clear to 15,61
         @ 7,17 to 15,61 double
         set color to r*+/bu
         @ 11,19 say "Record cannot be locked.Try again later."
         @ 12,26 say "Press any key to continue."
         read
         restore screen from screen
      endif
   endif
**
** Error: Record is not locked.
   if errnum=130
      save screen to screen
      time=0
      do while .not. rlock().and.time<1000
         time=time+1
      enddo
      if time<1000
         restore screen from screen
         retry
      else
         set color to gr+/bu
         @ 7,17 clear to 15,61
         @ 7,17 to 15,61 double
         set color to r*+/bu
         @ 11,19 say "Record cannot be locked.Try again later."
         @ 12,26 say "Press any key to continue."
         read
         restore screen from screen
      endif
   endif
*
*!*************************************************************************
*!
*!      Procedure: COUNTDEL
*!
*!          Calls: EOF()          (FOXBASE+ function)
*!               : DELETED()      (FOXBASE+ function)
*!
*!*************************************************************************
procedure countdel
******************
set delete off
store 0 to ctdel
goto top
do while .not. eof()
   if deleted()
      ctdel=ctdel+1
   endif
   if .not. eof()
      skip
   endif
enddo
goto top
set delete on
return
*
*!*************************************************************************
*!                              This procedure checks if the game manager has started,
*!      Procedure: CHEKSTOP     frozen, or quit the game. It is the procedure that checks
*!                              for the status of the running gaming-simulation and forces the
*!      Called by: PLA4.PRG     planning program to do so. Its functioning is thoroughly explained
*!               : MAINMENU       (procedure in PLA4.PRG)       in one of the thesis's sections.
*!               : GETKEY         (procedure in PLA4.PRG)
*!
*!          Calls: SYSTEM()       (function  in PLA4.PRG)
*!               : .AND..NOT.SYSTE(FOXBASE+ function)
*!               : PLANREPO       (procedure in PLA4.PRG)
*!               : PRT            (procedure in PLA4.PRG)
*!               : .AND..NOT.SHUT.(FOXBASE+ function)
*!               : .NOT.SYSTEM()  (FOXBASE+ function)
*!               : .AND.SYSTEM()  (FOXBASE+ function)
*!               : STR()          (FOXBASE+ function)
*!
```

```
*!          Uses: SYSTEM.DBF
*!
*!*************************************************************************
    procedure chekstop
    *****************
    parameters aselect
    private shut
    shut=.f.
    signal=.f.
    do while .t.
      do case
      case system("sys_stop").and..not.system("gam_stop")
        if .not. pause
          if system("plareport")
            do planrepo
            do prt
            select 10
            replace plareport with .f.
            unlock
            start=.t.
          endif
          if .not.signal
            save screen to screen1
            set color to
            clear
            set color to gr+/bu
            @ 5,14 clear to 14,64
            @ 5,14 to 14,64 double
            set color to bg+/bu
            @ 9,17 say "THE GAME HAS BEEN FROZEN OR A DAY HAS ENDED."
            @ 11,30 say "WAIT UNTIL RESTART."
            sys_start=.f.
            signal=.t.
          endif
          if system("restart").and..not.shut.and..not.system("gam_stop")
            close databases
            select 10
            use system
            shut=.t.
          else
            if .not.system("restart").and.shut
              set view to pladep
              shut=.f.
            endif
          endif
        else
          set color to gr+*/bu
          @ 24,70 say "STOPPED"
          set color to
              exit
        endif
      case .not.system("gam_stop").and..not.system("sys_stop")
        if signal.and..not.start.and..not.pause
          restore screen from screen1
        endif
        if start
          daystat=.f.
        endif
        start=.f.
        sys_start=.f.
        set color to
        @ 24,70 clear to 24,76
              exit
      case system("gam_stop").and.system("sys_stop")
        if sys_start
          if .not.signal1
            set color to
            clear
            set color to bu+/n
            @ 7,24 to 12,54 double
            set color to gr+/n
            @ 9,26 say "SETTING THE MASTER SYSTEM"
            @ 10,34 say "PLEASE WAIT"
            signal1=.t.
            start=.t.
          endif
        else
          close all
          clear all
          set color to
          clear
          set color to bu+/n
          @ 8,24 to 12,54 double
          set color to gr+/n
          @ 10,29 say "THE GAME HAS FINISHED"
```

```
            set color to
            set console off
cancel
         endif
      endcase
   enddo
   sys_start=.f.
   aselect=str(aselect)
   select &aselect
   return
   **
*!***********************************************************************
*!                      This procedure gets the keyboard input
*!     Procedure: GETKEY     done by the user.
*!
*!     Called by: PLA_BROW       (procedure in PLA4.PRG)
*!              : PLEDIT         (procedure in PLA4.PRG)
*!              : PLA_CODE       (procedure in PLA4.PRG)
*!              : PLA_RESO       (procedure in PLA4.PRG)
*!              : POSITION       (procedure in PLA4.PRG)
*!
*!         Calls: SYSTEM()       (function  in PLA4.PRG)
*!              : .NOT.SYSTEM()  (FOXBASE+ function)
*!              : CHR()          (FOXBASE+ function)
*!              : INKEY()        (FOXBASE+ function)
*!              : UPPER()        (FOXBASE+ function)
*!              : .AND.SYSTEM()  (FOXBASE+ function)
*!              : SELECT()       (FOXBASE+ function)
*!              : CHEKSTOP       (procedure in PLA4.PRG)
*!
*!          Uses: STATUS.DBF
*!
*!***********************************************************************
   procedure getkey
   ****************
   parameter choice,keychars
   private keycode,scrr
   scrr=.t.
   choice = "*"
   do while .not. (choice $ keychars)
      if system("restatus").and.scrr
         select 2
         use
         select 10
         replace restatus with .f.
         unlock
         do while .not.system("restatus")
            if scrr
               set typeahead to 0
               save screen to screen2
               set color to bu+/n
               @ 11,25 to 13,55 double
               set color to gr+*/n
               @ 12,29 say "calculating status file"
               scrr=.f.
            endif
         enddo
         set typeahead to 20
         restore screen from screen2
         set color to &promptatr
         @ row,0 say chr(16)
         select 2
         use status
         select 10
         replace restatus with .f.
         unlock
         select 3
      endif
      keycode = inkey()
      if .not.system("sys_stop").and..not.daystat
         daystat=.t.
         pagepaint=.t.
         statpage=.t.
         if press2="P"
            select 3
         else
            select 2
         endif
         set filter to &actfilt
exit
      endif
      if press2="P".and.system("STATRESO")=filtreso.and..not.editing
         if system("STATCODE")=filtcode.or.filtcode="ALL"
            pagepaint=.t.
            statpage=.t.
```

```
v═══════════════════exit
││    │    └──endif
││    └──endif
││    ┌──if keycode > 0
││    │   choice = upper(chr(keycode))
││    │   ┌──if choice $ "E".and.system("plareport")
││    │   │   pause=.f.
││    │   ├──else
││    │   │   ┌──if system("FACTTIME")<>"8:00"
││    │   │   │   pause=.t.
││    │   │   └──endif
││    │   └──endif
││    │   do chekstop with select()
││    └──endif
│└══enddo
 return
 *
*!*****************************************************************************
*!                         This procedure displays on the screen the scheduled
*!      Procedure: SAM_DISP    orders.
*!
*!      Called by: PLA_BROW        (procedure in PLA4.PRG)
*!               : POSITION      (procedure in PLA4.PRG)
*!               : CREANEW        (procedure in PLA4.PRG)
*!
*!         Calls: ROW()          (FOXBASE+ function)
*!              : COL()          (FOXBASE+ function)
*!
*!*****************************************************************************
 procedure sam_disp
 ******************
 parameter row,listrecs
 ┌──if listrecs > 1
 │   * ---Display heading when listing the entire page.
 │   set color to &statusatr
 │   @ rowtop-1,1
 │   ╒══do case
 │   ╞══case pancol = 1
 │   │   ?? "CODE DESCRIPT----------- QUANTIT NEED_TIME RESOUR_COD PRODUCT_1 ORD_STAT_1"
 │   ╘══endcase
 │   * ---Clear the window area.
 │   set color to &windowatr
 │   @ rowtop,0 clear to rowbottom,79
 └──endif
 * ---Display the records.
 set heading off
 set color to &windowatr
 @ row-1,0 say ""
 ╒══do case
 ╞══case pancol = 1
 │   list next listrecs code,descript,quantit," ",need_time,"   ",resour_cod,product_1," ",ord_stat_1," " off
 ╘══endcase
 set heading on
 ┌──if listrecs>1
 │   r=row()
 │   c=col()+1
 │   ┌──if statpage
 │   │   select 10
 │   │   replace statreso with "----"
 │   │   replace statcode with "----"
 │   │   unlock
 │   │   select 3
 │   │   statpage = .f.
 │   └──endif
 └──endif
 return
 *
*!*****************************************************************************
*!                         This procedure displays the sales orders to be
*!      Procedure: STA_DISP   delivered.
*!
*!      Called by: PLA_BROW        (procedure in PLA4.PRG)
*!
*!         Calls: ROW()          (FOXBASE+ function)
*!              : COL()          (FOXBASE+ function)
*!
*!*****************************************************************************
 procedure sta_disp
 ******************
 parameter row,listrecs
 ┌──if listrecs > 1
 │   * ---Display heading when listing the entire page.
 │   set color to &statusatr
 │   @ rowtop-1,1
 │   ╒══do case
```

```
      case pancol = 1
        ?? "CODE QUANTIT    STOCK     DATE   RESOURCE CAPACITY UNTIME  SETUP   TOTALTIME"
      endcase
      * ---Clear the window area.
      set color to &windowatr
      @ rowtop,0 clear to rowbottom,79
   endif
   * ---Display the records.
   set heading off
   set color to g+/n,n/w
   @ row-1,0 say ""
   do case
   case pancol = 1
      list next listrecs code,"  ",quantit,stock," ",date," ",resource,"   ",capacity,"  ",time,setup," ",tottime off
   endcase
   set heading on
   r=row()
   c=col()+1
   return
   *
*!**********************************************************************
*!                       This procedure calculates all the posible article codes
*!     Procedure: PLA_AREA0   to filter the <PLAN> and <STATUS> datafiles by.
*!
*!     Called by: FILT_CODE       (procedure in PLA4.PRG)
*!
*!        Calls: STR()          (FOXBASE+ function)
*!            : EOF()          (FOXBASE+ function)
*!            : PLA_CODE       (procedure in PLA4.PRG)
*!
*!**********************************************************************
   procedure pla_area0
   ******************
   parameter base
   private i,ii
   select 1
   goto top
   i=0
   ii=str(0,1)
   code&ii="ALL"
   i=i+1
   ii=str(i,1)
   do while .not. eof()
      if code<>"    "
         public code&ii
         code&ii=code
      endif
      if .not. eof()
         i=i+1
         ii=str(i,1)
         skip
      endif
   enddo
   base=str(base)
   select &base
   do pla_code with i
   return
   *
*!**********************************************************************
*!                       This procedure calculates all the possible resource codes
*!     Procedure: PLA_AREA1   to filter the <PLAN> and <STATUS> datafiles by.
*!
*!     Called by: FILT_RESO       (procedure in PLA4.PRG)
*!
*!        Calls: STR()          (FOXBASE+ function)
*!            : EOF()          (FOXBASE+ function)
*!            : PLA_RESO       (procedure in PLA4.PRG)
*!
*!**********************************************************************
   procedure pla_area1
   ******************
   parameter base1
   private i,ii
   select 5
   goto top
   i=0
   ii=str(i,1)
   do while .not. eof()
      if resour_cod<>"    "
         public reso&ii
         reso&ii=resour_cod
      endif
      if .not. eof()
         i=i+1
         ii=str(i,1)
```

```
        skip
    endif
enddo
base1=str(base1)
select &base1
do pla_reso with i
return
*
*!*****************************************************************************
*!                    This procedure calculates the filter of the <PLAN> and <STATUS>
*!    Procedure: PLA_CODE  datafiles based on the selected article code.
*!
*!    Called by: PLA_AREA0      (procedure in PLA4.PRG)
*!
*!        Calls: STR()          (FOXBASE+ function)
*!             : GETKEY         (procedure in PLA4.PRG)
*!
*!*****************************************************************************
procedure pla_code
******************
parameter numbcode
private n,nn,codekey,sayphrase
n=0
sayphrase=""
codekey=""
nn=str(n,1)
set color to &promptatr
do while n<numbcode
    sayphrase=sayphrase+"  "+nn+"-"+code&nn
    codekey=codekey+nn
    n=n+1
    nn=str(n,1)
enddo
@ rowprompt-1,0 clear
@ rowprompt-1,0 say promptbar
@ rowprompt,0 clear
@ rowprompt,0 say "SET CODE:"+sayphrase
do getkey with codchoice,codekey+returnkey
do case
case codchoice = returnkey
            return
case codchoice $ codekey
    filtcode=code&codchoice
endcase
return
*
*!*****************************************************************************
*!                    This procedure calls the procedure to calculate the filter
*!    Procedure: PLA_RESO   of the <PLAN> and <STATUS> datafiles based on the selected
*!                                              resource code.
*!    Called by: PLA_AREA1      (procedure in PLA4.PRG)
*!
*!        Calls: STR()          (FOXBASE+ function)
*!             : GETKEY         (procedure in PLA4.PRG)
*!             : RESOURCE       (procedure in PLA4.PRG)
*!
*!*****************************************************************************
procedure pla_reso
******************
parameter numbreso
private n,nn,codekey,sayphrase
n=0
sayphrase=""
codekey=""
nn=str(n,1)
set color to &promptatr
do while n<numbreso
    sayphrase=sayphrase+"  "+nn+"-"+reso&nn
    codekey=codekey+nn
    n=n+1
    nn=str(n,1)
enddo
@ rowprompt-1,0 clear
@ rowprompt-1,0 say promptbar
@ rowprompt,0 clear
@ rowprompt,0 say "SET RESOURCE:"+sayphrase
do getkey with reschoice,codekey+returnkey
if reschoice $ codekey
    filtreso=reso&reschoice
    do resource
endif
return
*
*!*****************************************************************************
*!                    This procedure calculates the filter of the <PLAN> and <STATUS>
```

```
*!      Procedure: RESOURCE     datafiles based on the selected resource code.
*!
*!      Called by: PLA_RESO        (procedure in PLA4.PRG)
*!               : FILT_RESO      (procedure in PLA4.PRG)
*!
*!         Calls: .NOT.EOF()      (FOXBASE+ function)
*!               : STR()          (FOXBASE+ function)
*!               : EOF()          (FOXBASE+ function)
*!
*!**************************************************************************
   procedure resource
   ******************
   public field
   m=0
   phras=""
   phrase=""
   select 1
   goto top
   do while .not.eof()
      if resour_cod=filtreso
         mm=str(m,1)
         public field
         field=code
         phras="&field"
         m=m+1
         if m>1
            phrase=phrase+phras
         else
            phrase=phras
         endif
      endif
      if .not. eof()
         skip
      endif
   enddo
   select 3
   return
   *
*!**************************************************************************
*!                    This procedure filters the <PLAN> and <STATUS> datafiles
*!      Procedure: FILT_CODE    based on the selected article code.
*!
*!      Called by: PLA_BROW        (procedure in PLA4.PRG)
*!
*!         Calls: SELECT()        (FOXBASE+ function)
*!               : PLA_AREA0      (procedure in PLA4.PRG)
*!               : EOF()          (FOXBASE+ function)
*!               : RECNO()        (FOXBASE+ function)
*!
*!**************************************************************************
   procedure filt_code
   *******************
   parameter start
   if .not.start
      do pla_area0 with select()
   endif
   if codchoice="0"
      actfilt="Code<>FiltCode.AND.Code$phrase"
   else
      actfilt="Code=FiltCode.AND.Code$phrase"
   endif
   set filter to &actfilt
   goto top
   if eof()
      set color to &promptatr
      @ rowprompt,0 clear
      @ rowprompt,0 say "No matching records."
      wait
      @ rowprompt,0 clear
      phrase=lphrase
      filtcode=lfiltcode
      if filtcode="ALL"
         codchoice="0"
         actfilt="Code<>FiltCode.AND.Code$phrase"
      endif
      set filter to &actfilt
      goto top
   endif
   recnumtop=recno()
   pagepaint=.t.
   set color to &screenatr
   if press2="S"
      select 2
   endif
   clear
```

```
return
*
*!************************************************************************
*!                         This procedure filters the <PLAN> and <STATUS> datafiles
*!     Procedure: FILT_RESO    based on the selected resource code.
*!
*!     Called by: PLA_BROW        (procedure in PLA4.PRG)
*!
*!         Calls: SELECT()        (FOXBASE+ function)
*!              : PLA_AREA1       (procedure in PLA4.PRG)
*!              : RESOURCE        (procedure in PLA4.PRG)
*!              : RECNO()         (FOXBASE+ function)
*!
*!************************************************************************
procedure filt_reso
*******************
parameter start
mmstart=start
if .not.start
  do pla_area1 with select()
else
  do resource
endif
if codchoice="0"
  actfilt="Code<>FiltCode.AND.code$phrase"
else
  actfilt="Code=FiltCode.AND.code$phrase"
endif
set filter to &actfilt
goto top
recnumtop=recno()
pagepaint=.t.
set color to &screenatr
clear
return
*
*!************************************************************************
*!                         This procedure allows the player to define the source
*!     Procedure: POSITION    and target positions of the order to be re-sequenced.
*!
*!     Called by: PLA_BROW        (procedure in PLA4.PRG)
*!
*!         Calls: CHR()           (FOXBASE+ function)
*!              : RECNO()         (FOXBASE+ function)
*!              : GETKEY          (procedure in PLA4.PRG)
*!              : BOF()           (FOXBASE+ function)
*!              : SAM_DISP        (procedure in PLA4.PRG)
*!              : EOF()           (FOXBASE+ function)
*!              : RLOCK()         (FOXBASE+ function)
*!              : REPLANN         (procedure in PLA4.PRG)
*!
*!************************************************************************
procedure position
******************
rsour=row
@ row,0 say " "
set color to &promptatr
@ row,0 say chr(17)
source=recno()
sourseq=sequence
@ rowprompt,0 clear
@ rowprompt,0 say ;
"SELECT SEQUENCE :  <Arrows>  <Return> "
do while .t.
  do getkey with choice,uparrow+downarrow+returnkey
  do while choice $ uparrow+downarrow
    if row<>rsour
      @ row,0 say " "
    else
      @ rsour,0 say chr(17)
    endif
    if choice=uparrow
      skip -1
      do case
      case bof()
        goto top
      case row > rowtop
        row = row - 1
      otherwise
        recnumtop=recno()
        * ---Scroll window down.
        scroll rowtop,0,rowbottom,79,-1
        do sam_disp with row,1
      endcase
    else
```

```
                skip
                do case
                case eof()
                   goto bottom
                case row < rowbottom
                   row = row + 1
                otherwise
                   * ---Adjust top-of-page record pointer.
                   recnum=recno()
                   goto recnumtop
                   skip
                   recnumtop=recno()
                   goto recnum
                   * ---Scroll window up.
                   scroll rowtop,0,rowbottom,79,1
                   do sam_disp with row,1
                endcase
             endif
          set color to &promptatr
          @ row,0 say chr(16)
          do getkey with choice,uparrow+downarrow+returnkey
       enddo
       if choice=returnkey.and.ord_stat_1="NO_START"
          if rlock() && Check that the record is locked
             target=recno()
             tarseq=sequence
             if tarseq<sourseq
                position=.t.          && UP
             else
                position=.f.          && DOWN
             endif
             do replann
          endif
 v                 exit
       endif
       @ rowprompt,0 clear
       @ rowprompt,0 say ;
       "IT IS NOT POSSIBLE TO RESEQUENCE THAT"
       wait
       @ rowprompt,0 clear
       @ rowprompt,0 say ;
       "SELECT SEQUENCE :  <Arrows>   <Return> "
    enddo
return
*
*!************************************************************************
*!                      This procedure re-schedules (re-sequence or re-edit)
*!      Procedure: REPLANN    the selected order.
*!
*!      Called by: POSITION        (procedure in PLA4.PRG)
*!
*!         Calls: SEQUENCE>TARSEQ(FOXBASE+ function)
*!              : <>SOURCE()      (FOXBASE+ function)
*!              : SEQUENCE<TARSEQ(FOXBASE+ function)
*!              : <>SOURCE()      (FOXBASE+ function)
*!              : EOF()           (FOXBASE+ function)
*!              : RECNO()         (FOXBASE+ function)
*!
*!************************************************************************
procedure replann
*****************

if target<>source
   if position
      goto source
      replace sequence with tarseq
      goto target
      replace sequence with (sequence+1)
   else
      goto source
      replace sequence with (tarseq-1)
   endif
   unlock
   if position
      filt="SEQUENCE>tarseq.AND.RECNO()<>source.AND.SEQUENCE<sourseq.AND.RECNO()<>target"
   else
      filt="SEQUENCE<tarseq.AND.RECNO()<>source.AND.SEQUENCE>sourseq.AND.RECNO()<>target"
   endif
   count for &filt to rec
   if rec<>0
      dimension rr(rec)
      goto top
      n=1
      do while .not. eof()
         if &filt
            rr(n)=recno()
```

```
            n=n+1
          endif
          if .not. eof()
            skip
          endif
        enddo
        n=1
        do while n<=rec
          goto rr(n)
          if position
            replace sequence with (sequence+1)
          else
            replace sequence with (sequence-1)
          endif
          unlock
          n=n+1
        enddo
    endif
endif
goto top
recnumtop=recno()
return
*
*!***********************************************************************
*!                  This procedure creates the "view" used during all the
*!    Procedure: OPEN      planning program.
*!
*!    Called by: PLA4.PRG
*!
*!         Uses: BILL.DBF
*!             : STATUS.DBF
*!             : PLANN.DBF
*!             : STOCK.DBF
*!             : MACHINE.DBF
*!             : PLAPAST.DBF
*!             : STKPAST.DBF
*!             : MACHPAST.DBF
*!             : SETUP.DBF
*!
*!      Indexes: BILLNA.IDX
*!             : ORPLANN.IDX
*!             : CODESTK.IDX
*!             : RESOUR.IDX
*!             : CDPLA.IDX
*!             : CDSTK.IDX
*!             : CDMACH.IDX
*!             : SETUP.IDX
*!
*!***********************************************************************
procedure open
**************
set exclusive off
select 1
use bill index billna
select 2
use status
select 3
use plann index orplann
select 4
use stock index codestk
select 5
use machine index resour
select 6
use plapast index cdpla
select 7
use stkpast index cdstk
select 8
use machpast index cdmach
select 9
use setup index setup
select 2
set safety off
create view pladep from environment all
set safety on
close databases
return
**
*!***********************************************************************
*!                    This procedure calls the report procedures for
*!    Procedure: PLANREPO    <PLAN>, <STOCK>, and <MACHINE> datafiles.
*!
*!    Called by: CHEKSTOP       (procedure in PLA4.PRG)
*!             : PRT            (procedure in PLA4.PRG)
*!
*!        Calls: PLAREPO        (procedure in PLA4.PRG)
```

```
*!
*!*******************************************************************
     procedure planrepo
     *****************
     do plarepo with 6
     do plarepo with 7
     do plarepo with 8
     signal=.f.
     return
*!*******************************************************************
*!                    This procedure manages all the report generations
*!    Procedure: PLAREPO    for the different datafiles to be report.
*!
*!    Called by: PLANREPO        (procedure in PLA4.PRG)
*!
*!        Calls: STR()           (FOXBASE+ function)
*!             : VAL()           (FOXBASE+ function)
*!             : EOF()           (FOXBASE+ function)
*!             : INT()           (FOXBASE+ function)
*!             : SUBSTR()        (FOXBASE+ function)
*!             : DTOC()          (FOXBASE+ function)
*!             : PLAAVE          (procedure in PLA4.PRG)
*!             : STKAVE          (procedure in PLA4.PRG)
*!             : MACHAVE         (procedure in PLA4.PRG)
*!             : PLAPAST         (procedure in PLA4.PRG)
*!             : STKPAST         (procedure in PLA4.PRG)
*!             : MACHPAST        (procedure in PLA4.PRG)
*!
*!*******************************************************************
     procedure plarepo
     *****************
     parameter pastbase
     set color to
     do case
     case pastbase=6
        store 0 to n,mave_inc,mper_inc,mave_star,mper_star,mave_fin
        store 0 to mper_fin,mave_prod,mave_wip,mave_batch,mave_setn
        store 0 to mave_lead,mave_cost,per_inc,per_star,per_fin,ave_star,;
        mave_tot,ave_tot
     case pastbase=7
        store 0 to n,mave_stk
     case pastbase=8
        store 0 to n,mave_cap,mave_prod,mave_prot,mave_sett,mave_setn,;
        per_sett,mper_sett
     endcase

     pastbase=str(pastbase,2)
     select &pastbase
     pastbase=val(pastbase)
     goto top
     do while .not. eof()
        do case
        case pastbase=6.or.pastbase=7
           actcode=code
           mdescript=descript
           fieldcod="CODE"
        case pastbase=8
           actcode=resour_cod
           mdescript=descript
           fieldcod="RESOUR_COD"
        endcase
        do while &fieldcod=actcode
           actweek=int(val(substr(dtoc(date),1,2))/5)
           do while int(val(substr(dtoc(date),1,2))/5)=actweek.and.&fieldcod=actcode
              n=n+1
              do case
              case pastbase=6
                 do plaave
              case pastbase=7
                 do stkave
              case pastbase=8
                 do machave
              endcase
              if .not. eof()
                 skip
              else
                          exit
              endif
           enddo
           skip -1
           set color to
           clear
           do case
           case pastbase=6
              do plapast
```

```
            case pastbase=7
              do stkpast
            case pastbase=8
              do machpast
            endcase
          skip
            do case
            case pastbase=6
              store 0 to n,mave_inc,mper_inc,mave_star,mper_star,mave_fin
              store 0 to mper_fin,mave_prod,mave_wip,mave_batch,mave_setn
              store 0 to mave_lead,mave_cost,per_inc,per_star,per_fin,ave_star,;
              mave_tot,ave_tot
            case pastbase=7
              store 0 to n,mave_stk
            case pastbase=8
              store 0 to n,mave_cap,mave_prod,mave_prot,mave_sett,mave_setn,;
              per_sett,mper_sett
            endcase
        enddo
    enddo
return
*
*!*********************************************************************
*!                   This procedure calculates the averages of the
*!    Procedure: PLAAVE     <PLAN> datafile's key fields.
*!
*!    Called by: PLAREPO         (procedure in PLA4.PRG)
*!
*!*********************************************************************
procedure plaave
****************
mave_inc=(mave_inc+incomplete)/n
mave_star=(mave_star+started)/n
mave_fin=(mave_fin+finished)/n
mave_prod=(mave_prod+production)/n
mave_wip=(mave_wip+wip)/n
mave_batch=(mave_batch+ave_batch)/n
mave_setn=(mave_setn+setupnum)/n
mave_lead=(mave_lead+leadtime)/n
mave_cost=(mave_cost+stand_cost)/n
mave_tot=(mave_inc+mave_star+mave_fin)
ave_tot=(incomplete+started+finished)
per_inc=iif(ave_tot=0,0,(incomplete/ave_tot)*100)
per_star=iif(ave_tot=0,0,(started/ave_tot)*100)
per_fin=iif(ave_tot=0,0,(finished/ave_tot)*100)
mper_inc=iif(mave_tot=0,0,(mave_inc/mave_tot)*100)
mper_star=iif(mave_tot=0,0,(mave_star/mave_tot)*100)
mper_fin=iif(mave_tot=0,0,(mave_fin/mave_tot)*100)
return
*
*!*********************************************************************
*!                   This procedure calculates the average of the <MACHINE>
*!    Procedure: MACHAVE    datafile's key fields.
*!
*!    Called by: PLAREPO         (procedure in PLA4.PRG)
*!
*!*********************************************************************
procedure machave
*****************
mave_cap=(mave_cap+capacity)/n
mave_prod=(mave_prod+production)/n
mave_prot=(mave_prot+productiv)/n
mave_sett=(mave_sett+setuptime)/n
per_sett=(setuptime/capacity)*100&& the capacity should be per day.
mper_sett=(mave_sett/capacity)*100&& the capacity should be per week.
mave_setn=(mave_setn+setupnum)/n
return
*
*!*********************************************************************
*!                   This procedure calculates the average of the <STOCK>
*!    Procedure: STKAVE    datafile's key fields.
*!
*!    Called by: PLAREPO         (procedure in PLA4.PRG)
*!
*!*********************************************************************
procedure stkave
****************
mave_stk=(mave_stk+stock)/n
return
*
*!*********************************************************************
*!                   This procedure allows the user to select the report's
*!    Procedure: PRT      output device (screen or printer).
*!
*!    Called by: CHEKSTOP        (procedure in PLA4.PRG)
```

```
*!
*!        Calls: SYS()          (FOXBASE+ function)
*!             : PLANREPO       (procedure in PLA4.PRG)
*!
*!*********************************************************************
   procedure prt
   *************
   do while .t.
      set color to
      clear
      m_menu=0
      set color to bu+/n
      @ 4,18 to 17,61
      @ 6,24 to 8,54 double
      set color to br+/n
      @  7,29 say "PRINT OUT OF REPORTS"
      set color to bu+/n
      @ 10,24,14,50 box " ╓─╖║╜─╙║ "
      set color to bg+/n
      @ 11,29 prompt "1.TO PRINTER.       "
      @ 12,29 prompt "2.PAST BEHAVIOUR."
      @ 13,29 prompt "3.EXIT.             "
      set color to bu+/n
      @ 16,29 say "DEVICE :"
      ?? sys(101)
      menu to m_menu
      do case
      case m_menu=1
         set device to print
      case m_menu=2
         do planrepo
         set device to screen
         set color to
         exit
      case m_menu=3
         set device to screen
         set color to
         exit
      endcase
   enddo
   return
   *
*!*********************************************************************
*!                     This procedure displays the reports concerning the
*!    Procedure: PLAPAST   schedule performance.
*!
*!    Called by: PLAREPO        (procedure in PLA4.PRG)
*!
*!       Calls: INKEY()         (FOXBASE+ function)
*!
*!*********************************************************************
   procedure plapast
   *****************
   set color to r/n
   @  4,23 say " ╔══════════════════════════════════╗ "
   @  5,23 say " ║  PLANNING DEPARTMENT'S PAST      ║ "
   @  6,23 say " ║     PERFORMANCE (Page 1)         ║ "
   @  7,23 say " ╚══════════════════════════════════╝ "
   @ 10, 0 say "Week"
   @ 10,15 say actweek    picture "99"
   @ 10,20 say "Date"
   @ 10,30 say date
   @ 12, 0 say "Code"
   @ 12,13 say actcode
   @ 12,20 say "Descript"
   @ 12,30 say mdescript
   @ 15,27 say "DAY"
   @ 15,34 say "OVERALL"
   @ 15,65 say "DAY"
   @ 15,72 say "OVERALL"
   @ 16, 0 say "Num. of incomplete orders"
   @ 16,27 say incomplete picture "999"
   @ 16,36 say mave_inc   picture "999"
   @ 16,43 say "% over total orders"
   @ 16,65 say per_inc    picture "999.99"
   @ 16,74 say mper_inc   picture "999.99"
   @ 17, 0 say "Num. of started orders"
   @ 17,27 say started    picture "999"
   @ 17,36 say mave_star  picture "999"
   @ 17,43 say "% over total orders"
   @ 17,65 say per_star   picture "999.99"
   @ 17,74 say mper_star  picture "999.99"
   @ 18, 0 say "Num. of finished orders"
   @ 18,27 say finished   picture "999"
   @ 18,36 say mave_fin   picture "999"
```

```
@ 18,43 say "% over total orders"
@ 18,65 say per_fin    picture "999.99"
@ 18,74 say mper_fin   picture "999.99"
*
? inkey(3)
*
set color to
clear
*
set color to r/n
@  4,23 say "╔══════════════════════════════════════╗"
@  5,23 say "║   PLANNING DEPARTMENT'S PAST         ║"
@  6,23 say "║      PERFORMANCE (Page 2)            ║"
@  7,23 say "╚══════════════════════════════════════╝"
@ 10, 2 say "Week"
@ 10,17 say actweek    picture "99"
@ 10,22 say "Date"
@ 10,32 say date
@ 12, 2 say "Code"
@ 12,15 say actcode
@ 12,22 say "Descript  "
@ 12,32 say mdescript
@ 15,22 say "DAY"
@ 15,32 say "OVERALL"
@ 15,63 say "DAY"
@ 15,70 say "OVERALL"
@ 16, 3 say "Production"
@ 16,18 say production picture "9999999.99"
@ 16,30 say mave_prod  picture "9999999.99"
@ 16,44 say "Num. of setups  "
@ 16,63 say setupnum   picture "999"
@ 16,72 say mave_setn  picture "999"
@ 17, 3 say "Batch size "
@ 17,20 say ave_batch  picture "999.99"
@ 17,32 say mave_batch picture "999.99"
@ 17,44 say "Lead-time"
@ 17,61 say leadtime   picture "999.99"
@ 17,70 say mave_lead  picture "999.99"
@ 18, 3 say "Wip        "
@ 18,18 say wip        picture "9999999.99"
@ 18,30 say mave_wip   picture "9999999.99"
@ 18,44 say "Standard cost"
@ 18,61 say stand_cost picture "999.99"
@ 18,70 say mave_cost  picture "999.99"
? inkey(3)
set color to
clear
return
*
*!***********************************************************************
*!                         This procedure displays the reports concerning the
*!      Procedure: MACHPAST    resources performance.
*!
*!      Called by: PLAREPO        (procedure in PLA4.PRG)
*!
*!          Calls: INKEY()        (FOXBASE+ function)
*!
*!***********************************************************************
procedure machpast
******************
set color to r/n
@  4,23 say "╔══════════════════════════════════════╗"
@  5,23 say "║   PLANNING DEPARTMENT'S PAST         ║"
@  6,23 say "║      PERFORMANCE (Page 3)            ║"
@  7,23 say "╚══════════════════════════════════════╝"
@ 10, 0 say "Week"
@ 10,15 say actweek    picture "99"
@ 10,20 say "Date"
@ 10,30 say date
@ 12, 0 say "Resource"
@ 12,13 say actcode
@ 12,20 say "Descript"
@ 12,30 say mdescript
@ 12,55 say "Machine Capacity"
@ 12,73 say capacity   picture "9999"
@ 15,24 say "DAY"
@ 15,32 say "OVERALL"
@ 16, 0 say "Number of Set-Ups"
@ 16,25 say setupnum   picture "99"
@ 16,34 say mave_setn  picture "99"
@ 16,66 say "DAY"
@ 16,73 say "OVERALL"
@ 17, 0 say "Machine Production"
@ 17,23 say production picture "9999.99"
@ 17,32 say mave_prod  picture "9999.99"
```

```
@ 17,42 say "Machine Productivity"
@ 17,65 say productiv  picture "999.99"
@ 17,73 say mave_prot  picture "999.99"
@ 18, 0 say "Time spend in Set-Ups"
@ 18,23 say setuptime  picture "9999.99"
@ 18,32 say mave_sett  picture "9999.99"
@ 18,42 say "% of Set-Ups"
@ 18,65 say per_sett   picture "999.99"
@ 18,73 say mper_sett  picture "999.99"
? inkey(3)
set color to
clear
return
*
*!*****************************************************************************
*!                         This procedure displays the reports concerning
*!    Procedure: STKPAST    the stock performance.
*!
*!    Called by: PLAREPO          (procedure in PLA4.PRG)
*!
*!        Calls: INKEY()          (FOXBASE+ function)
*!
*!*****************************************************************************
procedure stkpast
*****************
set color to r/n
@  4,23 say "╔══════════════════════════════════╗"
@  5,23 say "║  PLANNING DEPARTMENT'S PAST      ║"
@  6,23 say "║     PERFORMANCE (Page 4)         ║"
@  7,23 say "╚══════════════════════════════════╝"
@ 10, 0 say "Week"
@ 10,15 say actweek     picture "99"
@ 10,20 say "Date"
@ 10,30 say date
@ 12, 0 say "Code"
@ 12,13 say actcode
@ 12,20 say "Descript"
@ 12,30 say mdescript
@ 15,24 say "DAY"
@ 15,32 say "OVERALL"
@ 16, 0 say "Level of stocks"
@ 16,21 say stock       picture "99999999"
@ 16,31 say mave_stk    picture "99999999"
? inkey(3)
set color to
clear
return
**
*!*****************************************************************************
*!                         This procedure obtains the most updated value of
*!     Function: SYSTEM()   the fields of the <SYSTEM> datafile.
*!
*!    Called by: PLA4.PRG
*!             : CHEKSTOP         (procedure in PLA4.PRG)
*!             : PLA_BROW         (procedure in PLA4.PRG)
*!             : GETKEY           (procedure in PLA4.PRG)
*!
*!        Calls: STR()            (FOXBASE+ function)
*!             : SELECT()         (FOXBASE+ function)
*!
*!*****************************************************************************
procedure system
****************
parameters fieldddd
store str(select(),2) to base
select 10
goto top
system=&fieldddd
select &base
return system
**
*!*****************************************************************************
*!                         This procedure calls the procedures used to create
*!    Procedure: CREANEW    a new order in the <PLAN> datafile.
*!
*!    Called by: PLA_BROW         (procedure in PLA4.PRG)
*!
*!        Calls: .NOT.SYSTEM()    (FOXBASE+ function)
*!             : RECCOUNT()       (FOXBASE+ function)
*!             : EOF()            (FOXBASE+ function)
*!             : RECNO()          (FOXBASE+ function)
*!             : SAM_DISP         (procedure in PLA4.PRG)
*!             : CHR()            (FOXBASE+ function)
*!             : NEWPLAN          (procedure in PLA4.PRG)
*!
```

```
*!**********************************************************************
procedure creanew
*****************
    if .not.system("SYS_STOP")
        if reccount()<>0.and.r>20
            do while .not. eof()
                goto recnumtop
                skip skiprecs
                if eof()
                    goto bottom
                endif
                recnumtop = recno()
                recnum = recno()
                goto recnumtop
            enddo
            goto recnumtop
            do sam_disp with (rowtop),skiprecs
            goto recnum
            row=rowtop
            set color to &promptatr
            @ row,0 say chr(16)
        endif
        do newplan
        goto top
        recnumtop=recno()
        pagepaint = .t.
    endif
return
*: EOF: PLA4.ACT
```

System: PLA4.PRG
Author: JUAN IGNACIO IGARTUA
08/05/91 18:27:26
Tree Diagram

```
PLA4.PRG
|     SYSTEM.DBF (database)
|------------ERROR() (function in ?)
|------------MESSAGE() (function in ?)
|------------ERR_FIX     (procedure in PLA4.PRG)
|    └-----------RLOCK() (function in ?)
|------------REPLICATE() (function in ?)
|------------CHR() (function in ?)
|------------SYSTEM() (function in PLA4.PRG)
|    |-----------STR() (function in ?)
|    └-----------SELECT() (function in ?)
|------------OPEN     (procedure in PLA4.PRG)
|          BILL.DBF (database)
|          STATUS.DBF (database)
|          PLANN.DBF (database)
|          STOCK.DBF (database)
|          MACHINE.DBF (database)
|          PLAPAST.DBF (database)
|          STKPAST.DBF (database)
|          MACHPAST.DBF (database)
|          SETUP.DBF (database)
|------------SELECT() (function in ?)
|------------CHEKSTOP     (procedure in PLA4.PRG)
|    |     SYSTEM.DBF (database)
|    |-----------SYSTEM() (function in PLA4.PRG)
|    |     |-----------STR() (function in ?)
|    |     └-----------SELECT() (function in ?)
|    |-----------.AND..NOT.SYSTEM() (function in ?)
|    |-----------PLANREPO     (procedure in PLA4.PRG)
|    |     └-----------PLAREPO     (procedure in PLA4.PRG)
|    |          |-----------STR() (function in ?)
|    |          |-----------VAL() (function in ?)
|    |          |-----------EOF() (function in ?)
|    |          |-----------INT() (function in ?)
|    |          |-----------SUBSTR() (function in ?)
|    |          |-----------DTOC() (function in ?)
|    |          |-----------PLAAVE     (procedure in PLA4.PRG)
|    |          |-----------STKAVE     (procedure in PLA4.PRG)
|    |          |-----------MACHAVE     (procedure in PLA4.PRG)
|    |          |-----------PLAPAST     (procedure in PLA4.PRG)
|    |          |     └-----------INKEY() (function in ?)
|    |          |-----------STKPAST     (procedure in PLA4.PRG)
|    |          |     └-----------INKEY() (function in ?)
|    |          └-----------MACHPAST     (procedure in PLA4.PRG)
|    |               └-----------INKEY() (function in ?)
|    |-----------PRT     (procedure in PLA4.PRG)
|    |     |-----------SYS() (function in ?)
|    |     └-----------PLANREPO     (procedure in PLA4.PRG)
|    |          └-----------PLAREPO     (procedure in PLA4.PRG)
|    |               |-----------STR() (function in ?)
|    |               |-----------VAL() (function in ?)
|    |               |-----------EOF() (function in ?)
|    |               |-----------INT() (function in ?)
|    |               |-----------SUBSTR() (function in ?)
|    |               |-----------DTOC() (function in ?)
|    |               |-----------PLAAVE     (procedure in PLA4.PRG)
|    |               |-----------STKAVE     (procedure in PLA4.PRG)
|    |               |-----------MACHAVE     (procedure in PLA4.PRG)
|    |               |-----------PLAPAST     (procedure in PLA4.PRG)
|    |               |     └-----------INKEY() (function in ?)
|    |               |-----------STKPAST     (procedure in PLA4.PRG)
|    |               |     └-----------INKEY() (function in ?)
|    |               └-----------MACHPAST     (procedure in PLA4.PRG)
|    |                    └-----------INKEY() (function in ?)
|    |-----------.AND..NOT.SHUT.AND..NOT.SYSTEM() (function in ?)
|    |-----------.NOT.SYSTEM() (function in ?)
|    |-----------.AND.SYSTEM() (function in ?)
|    └-----------STR() (function in ?)
└------------INPTMPS     (procedure in PLA4.PRG)
     └-----------PLA_BROW     (procedure in PLA4.PRG)
          |-----------CHR() (function in ?)
          |-----------RECNO() (function in ?)
          |-----------FILT_RESO     (procedure in PLA4.PRG)
          |     |-----------SELECT() (function in ?)
          |     |-----------PLA_AREA1     (procedure in PLA4.PRG)
          |     |     |-----------STR() (function in ?)
          |     |     |-----------EOF() (function in ?)
          |     |     └-----------PLA_RESO     (procedure in PLA4.PRG)
          |     |
```

```
            STR() (function in ?)
            GETKEY      (procedure in PLA4.PRG)
              STATUS.DBF (database)
              SYSTEM() (function in PLA4.PRG)
                STR() (function in ?)
                SELECT() (function in ?)
              .NOT.SYSTEM() (function in ?)
              CHR() (function in ?)
              INKEY() (function in ?)
              UPPER() (function in ?)
              .AND.SYSTEM() (function in ?)
              SELECT() (function in ?)
              CHEKSTOP      (procedure in PLA4.PRG)
                SYSTEM.DBF (database)
                SYSTEM() (function in PLA4.PRG)
                  STR() (function in ?)
                  SELECT() (function in ?)
                .AND..NOT.SYSTEM() (function in ?)
                PLANREPO      (procedure in PLA4.PRG)
                  PLAREPO      (procedure in PLA4.PRG)
                    STR() (function in ?)
                    VAL() (function in ?)
                    EOF() (function in ?)
                    INT() (function in ?)
                    SUBSTR() (function in ?)
                    DTOC() (function in ?)
                    PLAAVE      (procedure in PLA4.PRG)
                    STKAVE      (procedure in PLA4.PRG)
                    MACHAVE      (procedure in PLA4.PRG)
                    PLAPAST      (procedure in PLA4.PRG)
                      INKEY() (function in ?)
                    STKPAST      (procedure in PLA4.PRG)
                      INKEY() (function in ?)
                    MACHPAST      (procedure in PLA4.PRG)
                      INKEY() (function in ?)
                PRT      (procedure in PLA4.PRG)
                  SYS() (function in ?)
                  PLANREPO      (procedure in PLA4.PRG)
                    PLAREPO      (procedure in PLA4.PRG)
                      STR() (function in ?)
                      VAL() (function in ?)
                      EOF() (function in ?)
                      INT() (function in ?)
                      SUBSTR() (function in ?)
                      DTOC() (function in ?)
                      PLAAVE      (procedure in PLA4.PRG)
                      STKAVE      (procedure in PLA4.PRG)
                      MACHAVE      (procedure in PLA4.PRG)
                      PLAPAST      (procedure in PLA4.PRG)
                        INKEY() (function in ?)
                      STKPAST      (procedure in PLA4.PRG)
                        INKEY() (function in ?)
                      MACHPAST      (procedure in PLA4.PRG)
                        INKEY() (function in ?)
                .AND..NOT.SHUT.AND..NOT.SYSTEM() (function in ?)
                .NOT.SYSTEM() (function in ?)
                .AND.SYSTEM() (function in ?)
                STR() (function in ?)
            RESOURCE      (procedure in PLA4.PRG)
              .NOT.EOF() (function in ?)
              STR() (function in ?)
              EOF() (function in ?)
        RESOURCE      (procedure in PLA4.PRG)
          .NOT.EOF() (function in ?)
          STR() (function in ?)
          EOF() (function in ?)
        RECNO() (function in ?)
    .NOT.EOF() (function in ?)
    SAM_DISP      (procedure in PLA4.PRG)
        ROW() (function in ?)
        COL() (function in ?)
    EOF() (function in ?)
    STA_DISP      (procedure in PLA4.PRG)
        ROW() (function in ?)
        COL() (function in ?)
    SYSTEM() (function in PLA4.PRG)
        STR() (function in ?)
        SELECT() (function in ?)
    GETKEY      (procedure in PLA4.PRG)
        STATUS.DBF (database)
        SYSTEM() (function in PLA4.PRG)
          STR() (function in ?)
          SELECT() (function in ?)
        .NOT.SYSTEM() (function in ?)
        CHR() (function in ?)
```

```
  |--INKEY() (function in ?)
  |--UPPER() (function in ?)
  |--.AND.SYSTEM() (function in ?)
  |--SELECT() (function in ?)
  |--CHEKSTOP     (procedure in PLA4.PRG)
       SYSTEM.DBF (database)
     |--SYSTEM() (function in PLA4.PRG)
        |--STR() (function in ?)
        |--SELECT() (function in ?)
     |--.AND..NOT.SYSTEM() (function in ?)
     |--PLANREPO     (procedure in PLA4.PRG)
        |--PLAREPO     (procedure in PLA4.PRG)
           |--STR() (function in ?)
           |--VAL() (function in ?)
           |--EOF() (function in ?)
           |--INT() (function in ?)
           |--SUBSTR() (function in ?)
           |--DTOC() (function in ?)
           |--PLAAVE     (procedure in PLA4.PRG)
           |--STKAVE     (procedure in PLA4.PRG)
           |--MACHAVE     (procedure in PLA4.PRG)
           |--PLAPAST     (procedure in PLA4.PRG)
              |--INKEY() (function in ?)
           |--STKPAST     (procedure in PLA4.PRG)
              |--INKEY() (function in ?)
           |--MACHPAST     (procedure in PLA4.PRG)
              |--INKEY() (function in ?)
     |--PRT     (procedure in PLA4.PRG)
        |--SYS() (function in ?)
        |--PLANREPO     (procedure in PLA4.PRG)
           |--PLAREPO     (procedure in PLA4.PRG)
              |--STR() (function in ?)
              |--VAL() (function in ?)
              |--EOF() (function in ?)
              |--INT() (function in ?)
              |--SUBSTR() (function in ?)
              |--DTOC() (function in ?)
              |--PLAAVE     (procedure in PLA4.PRG)
              |--STKAVE     (procedure in PLA4.PRG)
              |--MACHAVE     (procedure in PLA4.PRG)
              |--PLAPAST     (procedure in PLA4.PRG)
                 |--INKEY() (function in ?)
              |--STKPAST     (procedure in PLA4.PRG)
                 |--INKEY() (function in ?)
              |--MACHPAST     (procedure in PLA4.PRG)
                 |--INKEY() (function in ?)
     |--.AND..NOT.SHUT.AND..NOT.SYSTEM() (function in ?)
     |--.NOT.SYSTEM() (function in ?)
     |--.AND.SYSTEM() (function in ?)
     |--STR() (function in ?)
|--BOF() (function in ?)
|--CREANEW     (procedure in PLA4.PRG)
  |--.NOT.SYSTEM() (function in ?)
  |--RECCOUNT() (function in ?)
  |--EOF() (function in ?)
  |--RECNO() (function in ?)
  |--SAM_DISP     (procedure in PLA4.PRG)
     |--ROW() (function in ?)
     |--COL() (function in ?)
  |--CHR() (function in ?)
  |--NEWPLAN     (procedure in PLA4.PRG)
     |--FOUND() (function in ?)
     |--PL_STOR     (procedure in PLA4.PRG)
     |--PLEDIT     (procedure in PLA4.PRG)
        |--PL_GETS     (procedure in PLA4.PRG)
        |--GETKEY     (procedure in PLA4.PRG)
             STATUS.DBF (database)
           |--SYSTEM() (function in PLA4.PRG)
              |--STR() (function in ?)
              |--SELECT() (function in ?)
           |--.NOT.SYSTEM() (function in ?)
           |--CHR() (function in ?)
           |--INKEY() (function in ?)
           |--UPPER() (function in ?)
           |--.AND.SYSTEM() (function in ?)
           |--SELECT() (function in ?)
           |--CHEKSTOP     (procedure in PLA4.PRG)
                SYSTEM.DBF (database)
              |--SYSTEM() (function in PLA4.PRG)
                 |--STR() (function in ?)
                 |--SELECT() (function in ?)
              |--.AND..NOT.SYSTEM() (function in ?)
              |--PLANREPO     (procedure in PLA4.PRG)
                 |--PLAREPO     (procedure in PLA4.PRG)
                    |--STR() (function in ?)
```

```
                         VAL() (function  in ?)
                         EOF() (function  in ?)
                         INT() (function  in ?)
                         SUBSTR() (function  in ?)
                         DTOC() (function  in ?)
                         PLAAVE     (procedure in PLA4.PRG)
                         STKAVE     (procedure in PLA4.PRG)
                         MACHAVE     (procedure in PLA4.PRG)
                         PLAPAST     (procedure in PLA4.PRG)
                             INKEY() (function  in ?)
                         STKPAST     (procedure in PLA4.PRG)
                             INKEY() (function  in ?)
                         MACHPAST     (procedure in PLA4.PRG)
                            INKEY() (function  in ?)
                  PRT     (procedure in PLA4.PRG)
                      SYS() (function  in ?)
                      PLANREPO     (procedure in PLA4.PRG)
                         PLAREPO     (procedure in PLA4.PRG)
                            STR() (function  in ?)
                            VAL() (function  in ?)
                            EOF() (function  in ?)
                            INT() (function  in ?)
                            SUBSTR() (function  in ?)
                            DTOC() (function  in ?)
                            PLAAVE     (procedure in PLA4.PRG)
                            STKAVE     (procedure in PLA4.PRG)
                            MACHAVE     (procedure in PLA4.PRG)
                            PLAPAST     (procedure in PLA4.PRG)
                                INKEY() (function  in ?)
                            STKPAST     (procedure in PLA4.PRG)
                                INKEY() (function  in ?)
                            MACHPAST     (procedure in PLA4.PRG)
                               INKEY() (function  in ?)
                  .AND..NOT.SHUT.AND..NOT.SYSTEM() (function  in ?)
                  .NOT.SYSTEM() (function  in ?)
                  .AND.SYSTEM() (function  in ?)
                  STR() (function  in ?)
           RETURNKEY.AND..NOT.SYSTEM() (function  in ?)
           INCLUDE     (procedure in PLA4.PRG)
               CALORD     (procedure in PLA4.PRG)
                   RECCOUNT() (function  in ?)
               RECNO() (function  in ?)
               PL_REPL     (procedure in PLA4.PRG)
                  EOF() (function  in ?)
           DELRECORD.AND..NOT.SYSTEM() (function  in ?)
 .AND..NOT.SYSTEM() (function  in ?)
 POSITION     (procedure in PLA4.PRG)
     CHR() (function  in ?)
     RECNO() (function  in ?)
     GETKEY     (procedure in PLA4.PRG)
    STATUS.DBF (database)
         SYSTEM() (function  in PLA4.PRG)
             STR() (function  in ?)
             SELECT() (function  in ?)
         .NOT.SYSTEM() (function  in ?)
         CHR() (function  in ?)
         INKEY() (function  in ?)
         UPPER() (function  in ?)
         .AND.SYSTEM() (function  in ?)
         SELECT() (function  in ?)
         CHEKSTOP     (procedure in PLA4.PRG)
       SYSTEM.DBF (database)
            SYSTEM() (function  in PLA4.PRG)
                STR() (function  in ?)
                SELECT() (function  in ?)
            .AND..NOT.SYSTEM() (function  in ?)
            PLANREPO     (procedure in PLA4.PRG)
                PLAREPO     (procedure in PLA4.PRG)
                   STR() (function  in ?)
                   VAL() (function  in ?)
                   EOF() (function  in ?)
                   INT() (function  in ?)
                   SUBSTR() (function  in ?)
                   DTOC() (function  in ?)
                   PLAAVE     (procedure in PLA4.PRG)
                   STKAVE     (procedure in PLA4.PRG)
                   MACHAVE     (procedure in PLA4.PRG)
                   PLAPAST     (procedure in PLA4.PRG)
                       INKEY() (function  in ?)
                   STKPAST     (procedure in PLA4.PRG)
                       INKEY() (function  in ?)
                   MACHPAST     (procedure in PLA4.PRG)
                      INKEY() (function  in ?)
            PRT     (procedure in PLA4.PRG)
                SYS() (function  in ?)
```

```
|    |        |     `---------PLANREPO     (procedure in PLA4.PRG)
|    |        |         `---------PLAREPO     (procedure in PLA4.PRG)
|    |        |              +---------STR() (function  in ?)
|    |        |              +---------VAL() (function  in ?)
|    |        |              +---------EOF() (function  in ?)
|    |        |              +---------INT() (function  in ?)
|    |        |              +---------SUBSTR() (function  in ?)
|    |        |              +---------DTOC() (function  in ?)
|    |        |              +---------PLAAVE     (procedure in PLA4.PRG)
|    |        |              +---------STKAVE     (procedure in PLA4.PRG)
|    |        |              +---------MACHAVE     (procedure in PLA4.PRG)
|    |        |              +---------PLAPAST     (procedure in PLA4.PRG)
|    |        |              |    `---------INKEY() (function  in ?)
|    |        |              +---------STKPAST     (procedure in PLA4.PRG)
|    |        |              |    `---------INKEY() (function  in ?)
|    |        |              `---------MACHPAST     (procedure in PLA4.PRG)
|    |        |                   `---------INKEY() (function  in ?)
|    |        +---------.AND..NOT.SHUT.AND..NOT.SYSTEM() (function  in ?)
|    |        +---------.NOT.SYSTEM() (function  in ?)
|    |        +---------.AND.SYSTEM() (function  in ?)
|    |        `---------STR() (function  in ?)
|    +---------BOF() (function  in ?)
|    +---------SAM_DISP     (procedure in PLA4.PRG)
|    |    +---------ROW() (function  in ?)
|    |    `---------COL() (function  in ?)
|    +---------EOF() (function  in ?)
|    +---------RLOCK() (function  in ?)
|    `---------REPLANN     (procedure in PLA4.PRG)
|         +---------SEQUENCE>TARSEQ.AND.RECNO() (function  in ?)
|         +---------<>SOURCE.AND.SEQUENCE<SOURSEQ.AND.RECNO() (function  in ?)
|         +---------SEQUENCE<TARSEQ.AND.RECNO() (function  in ?)
|         +---------<>SOURCE.AND.SEQUENCE>SOURSEQ.AND.RECNO() (function  in ?)
|         +---------EOF() (function  in ?)
|         `---------RECNO() (function  in ?)
`---------FILT_CODE     (procedure in PLA4.PRG)
     +---------SELECT() (function  in ?)
     `---------PLA_AREA0     (procedure in PLA4.PRG)
          +---------STR() (function  in ?)
          +---------EOF() (function  in ?)
          `---------PLA_CODE     (procedure in PLA4.PRG)
               +---------STR() (function  in ?)
               `---------GETKEY     (procedure in PLA4.PRG)
                    |    STATUS.DBF (database)
                    +---------SYSTEM() (function  in PLA4.PRG)
                    |    +---------STR() (function  in ?)
                    |    `---------SELECT() (function  in ?)
                    +---------.NOT.SYSTEM() (function  in ?)
                    +---------CHR() (function  in ?)
                    +---------INKEY() (function  in ?)
                    +---------UPPER() (function  in ?)
                    +---------.AND.SYSTEM() (function  in ?)
                    +---------SELECT() (function  in ?)
                    `---------CHEKSTOP     (procedure in PLA4.PRG)
                         |    SYSTEM.DBF (database)
                         +---------SYSTEM() (function  in PLA4.PRG)
                         |    +---------STR() (function  in ?)
                         |    `---------SELECT() (function  in ?)
                         +---------.AND..NOT.SYSTEM() (function  in ?)
                         +---------PLANREPO     (procedure in PLA4.PRG)
                         |    `---------PLAREPO     (procedure in PLA4.PRG)
                         |         +---------STR() (function  in ?)
                         |         +---------VAL() (function  in ?)
                         |         +---------EOF() (function  in ?)
                         |         +---------INT() (function  in ?)
                         |         +---------SUBSTR() (function  in ?)
                         |         +---------DTOC() (function  in ?)
                         |         +---------PLAAVE     (procedure in PLA4.PRG)
                         |         +---------STKAVE     (procedure in PLA4.PRG)
                         |         +---------MACHAVE     (procedure in PLA4.PRG)
                         |         +---------PLAPAST     (procedure in PLA4.PRG)
                         |         |    `---------INKEY() (function  in ?)
                         |         +---------STKPAST     (procedure in PLA4.PRG)
                         |         |    `---------INKEY() (function  in ?)
                         |         `---------MACHPAST     (procedure in PLA4.PRG)
                         |              `---------INKEY() (function  in ?)
                         +---------PRT     (procedure in PLA4.PRG)
                         |    +---------SYS() (function  in ?)
                         |    `---------PLANREPO     (procedure in PLA4.PRG)
                         |         `---------PLAREPO     (procedure in PLA4.PRG)
                         |              +---------STR() (function  in ?)
                         |              +---------VAL() (function  in ?)
                         |              +---------EOF() (function  in ?)
                         |              +---------INT() (function  in ?)
                         |              +---------SUBSTR() (function  in ?)
                         |              +---------DTOC() (function  in ?)
```

```
 |                |                |
 |                |                ├──────────PLAAVE      (procedure in PLA4.PRG)
 |                |                ├──────────STKAVE      (procedure in PLA4.PRG)
 |                |                ├──────────MACHAVE     (procedure in PLA4.PRG)
 |                |                ├──────────PLAPAST     (procedure in PLA4.PRG)
 |                |                |    └──────────INKEY() (function  in ?)
 |                |                ├──────────STKPAST     (procedure in PLA4.PRG)
 |                |                |    └──────────INKEY() (function  in ?)
 |                |                └──────────MACHPAST    (procedure in PLA4.PRG)
 |                |                    └──────────INKEY() (function  in ?)
 |                ├──────────.AND..NOT.SHUT.AND..NOT.SYSTEM() (function  in ?)
 |                ├──────────.NOT.SYSTEM() (function  in ?)
 |                ├──────────.AND.SYSTEM() (function  in ?)
 |                └──────────STR() (function  in ?)
 ├──────────EOF() (function  in ?)
 └──────────RECNO() (function  in ?)
```

System: PLA4.PRG
Author: JUAN IGNACIO IGARTUA
08/05/91 18:27:17
Database Structure Summary

---

10 databases in the system
SYSTEM.DBF
STATUS.DBF
BILL.DBF
PLANN.DBF
STOCK.DBF
MACHINE.DBF
PLAPAST.DBF
STKPAST.DBF
MACHPAST.DBF
SETUP.DBF

Structure for database : SYSTEM.DBF
Number of data records : 1
Last updated : 07/01/91 at 9:17

Field	Field name	Type	Width	Dec	Start	End
1	STARTDATE	Date	8		1	8
2	FACTDATE	Date	8		9	16
3	FACTTIME	Character	4		17	20
4	SYS_STOP	Logical	1		21	21
5	GAM_STOP	Logical	1		22	22
6	HOUR_RATE	Numeric	7	2	23	29
7	TOT_OVERH	Numeric	10	2	30	39
8	NEWENQ	Logical	1		40	40
9	NEWENQREC	Character	9		41	49
10	FREQUENCY	Numeric	3		50	52
11	GOODSERV	Logical	1		53	53
12	REINDEX	Logical	1		54	54
13	STATRESO	Character	4		55	58
14	STATCODE	Character	4		59	62
15	WAITTIME	Numeric	4		63	66
16	RESTART	Logical	1		67	67
17	RESTATUS	Logical	1		68	68
18	THROUGHPUT	Numeric	10	2	69	78
19	ASSETS	Numeric	10	2	79	88
20	SYSTOCK	Numeric	10	2	89	98
21	SALREPORT	Logical	1		99	99
22	PLAREPORT	Logical	1		100	100
23	ENQUPERIOD	Numeric	1		101	101
** Total **			102			

FoxDoc did not find any associated index files

Used by: PLA4.PRG
: CHEKSTOP (procedure in PLA4.PRG)

Structure for database : STATUS.DBF
Number of data records : 7
Last updated : 07/01/91 at 9:13

Field	Field name	Type	Width	Dec	Start	End
1	CODE	Character	4		1	4
2	QUANTIT	Numeric	4		5	8
3	STOCK	Numeric	8		9	16
4	DATE	Date	8		17	24
5	RESOURCE	Character	4		25	28
6	CAPACITY	Numeric	4		29	32
7	TIME	Numeric	5	2	33	37
8	SETUP	Numeric	6	2	38	43
9	TOTTIME	Character	5		44	48
10	CDATE	Character	12		49	60
** Total **			61			

FoxDoc did not find any associated index files

Used by: OPEN (procedure in PLA4.PRG)
: GETKEY (procedure in PLA4.PRG)

Structure for database : BILL.DBF
Number of data records : 5

Last updated : 07/01/91 at 9:09

Field	Field name	Type	Width	Dec	Start	End
1	CODE	Character	4		1	4
2	DESCRIPT	Character	20		5	24
3	MINDELIVDD	Numeric	2		25	26
4	MAXDELIVDD	Numeric	2		27	28
5	MAXQUANTIT	Numeric	6	2	29	34
6	MINQUANTIT	Numeric	6	2	35	40
7	STANDPRICE	Numeric	6	2	41	46
8	MATCOST	Numeric	6	2	47	52
9	TOT_ENQU	Numeric	4		53	56
10	PRICE_LIMT	Numeric	6	2	57	62
11	DISCOUNT	Numeric	4	2	63	66
12	QUANT_DISC	Numeric	6	2	67	72
13	DELIV_LIMT	Numeric	4		73	76
14	MRKT_PRICE	Numeric	6	2	77	82
15	MRKT_LIMT	Numeric	6	2	83	88
16	ONTIMEDELV	Numeric	6	2	89	94
17	ONTIMLIMT	Numeric	6	2	95	100
18	STANDHOURS	Numeric	6	2	101	106
19	OVERHEAD	Numeric	6	2	107	112
20	STAND_COST	Numeric	6	2	113	118
21	NUM_ENQU	Numeric	4		119	122
22	RESOUR_COD	Character	4		123	126
23	PROCESTIME	Numeric	5	2	127	131
24	SET_UP	Character	7		132	138
** Total **			139			

This database appears to be associated with index file(s):
: BILLNA.IDX (CODE)

Used by: OPEN (procedure in PLA4.PRG)

Structure for database : PLANN.DBF
Number of data records : 5
Last updated : 07/01/91 at 9:17

Field	Field name	Type	Width	Dec	Start	End
1	CODE	Character	4		1	4
2	DESCRIPT	Character	20		5	24
3	QUANTIT	Numeric	6	2	25	30
4	START_DATE	Date	8		31	38
5	START_TIME	Character	4		39	42
6	ORDER_NUM	Numeric	4		43	46
7	SEQUENCE	Numeric	4		47	50
8	RESOUR_COD	Character	4		51	54
9	PRODUCT_1	Numeric	10	2	55	64
10	PRODUCT_0	Numeric	10	2	65	74
11	ORD_STAT_1	Character	8		75	82
12	ORD_STAT_0	Character	8		83	90
13	FINIS_DATE	Date	8		91	98
14	FINIS_TIME	Character	4		99	102
15	NEED_TIME	Numeric	6	2	103	108
** Total **			109			

This database appears to be associated with index file(s):
: ORPLANN.IDX (RESOUR_COD+CHR(SEQUENCE))

Used by: OPEN (procedure in PLA4.PRG)

Structure for database : STOCK.DBF
Number of data records : 5
Last updated : 07/01/91 at 9:13

Field	Field name	Type	Width	Dec	Start	End
1	CODE	Character	4		1	4
2	DESCRIPT	Character	20		5	24
3	STOCK	Numeric	8		25	32
** Total **			33			

This database appears to be associated with index file(s):
: CODESTK.IDX (CODE)

Used by: OPEN (procedure in PLA4.PRG)

```
Structure for database : MACHINE.DBF
Number of data records :       3
         Last updated : 07/01/91 at   9:13
Field  Field name  Type         Width     Dec      Start        End
    1  RESOUR_COD  Character        4                  1          4
    2  REDESCRIPT  Character       15                  5         19
    3  AVAIL       Character        3                 20         22
    4  T_AVAIL     Numeric          5                 23         27
    5  CAPACITY    Numeric          4                 28         31
    6  LASTSETUP   Character        7                 32         38
    7  SET_UP1     Numeric          6       2         39         44
    8  SET_UP2     Numeric          6       2         45         50
    9  SET_UP3     Numeric          6       2         51         56
   10  SET_UP4     Numeric          6       2         57         62
   11  SET_UP5     Numeric          6       2         63         68
** Total **                        69

This database appears to be associated with index file(s):
       : RESOUR.IDX  (RESOUR_COD)

Used by: OPEN           (procedure in PLA4.PRG)
```

```
Structure for database : PLAPAST.DBF
Number of data records :       0
         Last updated : 07/01/91 at   9:09
Field  Field name  Type         Width     Dec      Start        End
    1  CODE        Character        4                  1          4
    2  DESCRIPT    Character       20                  5         24
    3  INCOMPLETE  Numeric          3                 25         27
    4  STARTED     Numeric          3                 28         30
    5  FINISHED    Numeric          3                 31         33
    6  PRODUCTION  Numeric         10       2         34         43
    7  WIP         Numeric         10       2         44         53
    8  AVE_BATCH   Numeric          6       2         54         59
    9  SETUPNUM    Numeric          3                 60         62
   10  LEADTIME    Numeric          6       2         63         68
   11  STAND_COST  Numeric          6       2         69         74
   12  DATE        Date             8                 75         82
** Total **                        83

This database appears to be associated with index file(s):
       : CDPLA.IDX   (CODE+DTOC(DATE))

Used by: OPEN           (procedure in PLA4.PRG)
```

```
Structure for database : STKPAST.DBF
Number of data records :       5
         Last updated : 07/01/91 at   9:09
Field  Field name  Type         Width     Dec      Start        End
    1  CODE        Character        4                  1          4
    2  DESCRIPT    Character       20                  5         24
    3  STOCK       Numeric          8                 25         32
    4  DATE        Date             8                 33         40
** Total **                        41

This database appears to be associated with index file(s):
       : CDSTK.IDX   (CODE+DTOC(DATE))

Used by: OPEN           (procedure in PLA4.PRG)
```

```
Structure for database : MACHPAST.DBF
Number of data records :       3
         Last updated : 07/01/91 at   9:09
Field  Field name  Type         Width     Dec      Start        End
    1  RESOUR_COD  Character        4                  1          4
    2  DESCRIPT    Character       15                  5         19
    3  CAPACITY    Numeric          4                 20         23
    4  PRODUCTION  Numeric          7       2         24         30
    5  PRODUCTIV   Numeric          6       2         31         36
    6  SETUPTIME   Numeric          7       2         37         43
    7  SETUPNUM    Numeric          2                 44         45
    8  DATE        Date             8                 46         53
** Total **                        54
```

```
This database appears to be associated with index file(s):
       : CDMACH.IDX  (RESOUR_COD+DTOC(DATE))

Used by: OPEN           (procedure in PLA4.PRG)

----------------------------------------------------------------------


Structure for database : SETUP.DBF
Number of data records :       5
          Last updated : 07/01/91 at  9:09
Field  Field name  Type         Width    Dec    Start      End
    1  CODE        Character        4               1        4
    2  TYPE        Character        7               5       11
    3  SETUP       Numeric          6      2       12       17
** Total **                        18

This database appears to be associated with index file(s):
       : SETUP.IDX  (CODE)

Used by: OPEN           (procedure in PLA4.PRG)

----------------------------------------------------------------------
```

System: PLA4.PRG
Author: JUAN IGNACIO IGARTUA
08/05/91 18:27:19
Data Dictionary

Field Name	Type	Len	Dec	Database
ASSETS	N	10	2	SYSTEM.DBF
AVAIL	C	3	0	MACHINE.DBF
AVE_BATCH	N	6	2	PLAPAST.DBF
CAPACITY	N	4	0	MACHPAST.DBF
				STATUS.DBF
				MACHINE.DBF
CDATE	C	12	0	STATUS.DBF
CODE	C	4	0	PLANN.DBF
				BILL.DBF
				SETUP.DBF
				PLAPAST.DBF
				STKPAST.DBF
				STATUS.DBF
				STOCK.DBF
DATE	D	8	0	PLAPAST.DBF
				STKPAST.DBF
				STATUS.DBF
				MACHPAST.DBF
DELIV_LIMT	N	4	0	BILL.DBF
DESCRIPT	C	20	0	STKPAST.DBF
				PLAPAST.DBF
				STOCK.DBF
				PLANN.DBF
				BILL.DBF
DESCRIPT	C	15	0	MACHPAST.DBF
DISCOUNT	N	4	2	BILL.DBF
ENQUPERIOD	N	1	0	SYSTEM.DBF
FACTDATE	D	8	0	SYSTEM.DBF
FACTTIME	C	4	0	SYSTEM.DBF
FINISHED	N	3	0	PLAPAST.DBF
FINIS_DATE	D	8	0	PLANN.DBF
FINIS_TIME	C	4	0	PLANN.DBF
FREQUENCY	N	3	0	SYSTEM.DBF
GAM_STOP	L	1	0	SYSTEM.DBF
GOODSERV	L	1	0	SYSTEM.DBF
HOUR_RATE	N	7	2	SYSTEM.DBF
INCOMPLETE	N	3	0	PLAPAST.DBF
LASTSETUP	C	7	0	MACHINE.DBF
LEADTIME	N	6	2	PLAPAST.DBF
MATCOST	N	6	2	BILL.DBF
MAXDELIVDD	N	2	0	BILL.DBF
MAXQUANTIT	N	6	2	BILL.DBF
MINDELIVDD	N	2	0	BILL.DBF
MINQUANTIT	N	6	2	BILL.DBF
MRKT_LIMT	N	6	2	BILL.DBF
MRKT_PRICE	N	6	2	BILL.DBF
NEED_TIME	N	6	2	PLANN.DBF
NEWENQ	L	1	0	SYSTEM.DBF
NEWENQREC	C	9	0	SYSTEM.DBF
NUM_ENQU	N	4	0	BILL.DBF
ONTIMEDELV	N	6	2	BILL.DBF
ONTIMLIMT	N	6	2	BILL.DBF
ORDER_NUM	N	4	0	PLANN.DBF
ORD_STAT_0	C	8	0	PLANN.DBF
ORD_STAT_1	C	8	0	PLANN.DBF
OVERHEAD	N	6	2	BILL.DBF
PLAREPORT	L	1	0	SYSTEM.DBF
PRICE_LIMT	N	6	2	BILL.DBF
PROCESTIME	N	5	2	BILL.DBF
PRODUCTION	N	7	2	MACHPAST.DBF
PRODUCTION	N	10	2	PLAPAST.DBF
PRODUCTIV	N	6	2	MACHPAST.DBF

PRODUCT_0	N	10	2	PLANN.DBF
PRODUCT_1	N	10	2	PLANN.DBF
QUANTIT	N	6	2	PLANN.DBF
QUANTIT	N	4	0	STATUS.DBF
QUANT_DISC	N	6	2	BILL.DBF
REDESCRIPT	C	15	0	MACHINE.DBF
REINDEX	L	1	0	SYSTEM.DBF
RESOURCE	C	4	0	STATUS.DBF
RESOUR_COD	C	4	0	BILL.DBF
				MACHINE.DBF
				MACHPAST.DBF
				PLANN.DBF
RESTART	L	1	0	SYSTEM.DBF
RESTATUS	L	1	0	SYSTEM.DBF
SALREPORT	L	1	0	SYSTEM.DBF
SEQUENCE	N	4	0	PLANN.DBF
SETUP	N	6	2	SETUP.DBF
				STATUS.DBF
SETUPNUM	N	2	0	MACHPAST.DBF
SETUPNUM	N	3	0	PLAPAST.DBF
SETUPTIME	N	7	2	MACHPAST.DBF
SET_UP	C	7	0	BILL.DBF
SET_UP1	N	6	2	MACHINE.DBF
SET_UP2	N	6	2	MACHINE.DBF
SET_UP3	N	6	2	MACHINE.DBF
SET_UP4	N	6	2	MACHINE.DBF
SET_UP5	N	6	2	MACHINE.DBF
STANDHOURS	N	6	2	BILL.DBF
STANDPRICE	N	6	2	BILL.DBF
STAND_COST	N	6	2	BILL.DBF
				PLAPAST.DBF
STARTDATE	D	8	0	SYSTEM.DBF
STARTED	N	3	0	PLAPAST.DBF
START_DATE	D	8	0	PLANN.DBF
START_TIME	C	4	0	PLANN.DBF
STATCODE	C	4	0	SYSTEM.DBF
STATRESO	C	4	0	SYSTEM.DBF
STOCK	N	8	0	STKPAST.DBF
				STATUS.DBF
				STOCK.DBF
SYSTOCK	N	10	2	SYSTEM.DBF
SYS_STOP	L	1	0	SYSTEM.DBF
THROUGHPUT	N	10	2	SYSTEM.DBF
TIME	N	5	2	STATUS.DBF
TOTTIME	C	5	0	STATUS.DBF
TOT_ENQU	N	4	0	BILL.DBF
TOT_OVERH	N	10	2	SYSTEM.DBF
TYPE	C	7	0	SETUP.DBF
T_AVAIL	N	5	0	MACHINE.DBF
WAITTIME	N	4	0	SYSTEM.DBF
WIP	N	10	2	PLAPAST.DBF

MARKETING/SALES SUB-PROGRAM

SAL6.PRG

```
*:*************************************************************************
*:                  This program is the one that simulates all the activities
*:       Program: SAL6.PRG  of the sales department.Together with the Planning and System
*:                  ones, conforms the Nottingham Polytechnic's gaming-simulation.
*:        System: SAL6.PRG
*:        Author: JUAN IGNACIO IGARTUA
*:     Copyright (c) 1991, NOTTINGHAM POLYTECHNIC
*: Last modified: 08/05/91     17:48
*:
*:  Procs & Fncts: ERR_FIX
*:               : SYSTEM()
*:               : OPEN
*:               : CHEKSTOP
*:               : MAINMENU
*:               : INPTQUO
*:               : SAL_FORM
*:               : SAL_SAYS
*:               : SAL_GETS
*:               : SAL_STOR
*:               : MARKING
*:               : SAL_REPL
*:               : INPTPRI
*:               : SAL_EDIT
*:               : SAL_DISP
*:               : GETKEY
*:               : SAYLINE
*:               : SAL_AREA0
*:               : SAL_AREA1
*:               : SAL_CODE
*:               : SAL_STAT
*:               : FILT_CODE
*:               : FILT_STAT
*:               : SALREPO
*:               : SALAVE
*:               : SALFMT
*:               : PRT
*:               : SAL_BROW
*:               : POS
*:               : ACCEPT
*:
*:          Calls: ERROR()        (FOXBASE+ function)
*:               : MESSAGE()      (FOXBASE+ function)
*:               : ERR_FIX        (procedure in SAL6.PRG)
*:               : CTOD()         (FOXBASE+ function)
*:               : REPLICATE()    (FOXBASE+ function)
*:               : CHR()          (FOXBASE+ function)
*:               : SYSTEM()       (function  in SAL6.PRG)
*:               : OPEN           (procedure in SAL6.PRG)
*:               : SELECT()       (FOXBASE+ function)
*:               : CHEKSTOP       (procedure in SAL6.PRG)
*:               : MAINMENU       (procedure in SAL6.PRG)
*:
*:           Uses: SYSTEM.DBF
*:
*:      Documented 08/05/91 at 18:31               FoxDoc  version 1.0
*:*************************************************************************
*
set date british
set procedure to sal6.prg
on error do err_fix with error(),message()
set default to v:
set status off
set talk off
set bell off
set scoreboard off
set heading off
*
public signal,mstandprice,mququantit,mquprice,mqudate,;
mqutime,mqudeliver,mcode,pgdn,returnkey,pgup
public delrecord,recnum,choice,promptrow,promptbar,;
oldrecnum,screenatr,statusatr,windowatr,signal1
public promptatr,hiliteatr,filtstat,filtcode,ordchoice,;
codchoice,rowbottom,rowprompt,listlast,reccord,sys_start
public firstpos,lastpos,newenq,pause,start,waittime,lastdisp,;
browrec,move,bkey,xx,filtstat0,inedit,mmfactdate,chgorder
public tempdd,tempdd1
*
screenatr="R+/N,N/W"
statusatr="BU/N,N/W"
windowatr="R+/N,N/W"
promptatr="GR+/N,N/W"
hiliteatr="N/W"
*
filtstat="ENQU"+"QUOT"
```

```
filtstat0="ENQU"+"QUOT"
filtcode="ALL"
*
rowbottom = 20
rowprompt = rowbottom + 3
*
store 0 to recnum,oldrecnum,browrec,lastdisp
store ctod("01/01/99") to firstpos,lastpos
store " " to choice,xx
store "2" to ordchoice
store "0" to codchoice
promptrow=22
promptbar=replicate(chr(196),80)
pgdn=chr(3)
pgup=chr(18)
returnkey=chr(13)
delrecord=chr(7)
bkey=.f.
inedit=.f.
start=.f.
signal=.f.
signal1=.f.
sys_start=.t.
newenq=.f.
pause=.f.
move=.t.
chgorder=.f.
*
set exclusive off
select 10
use system
**
scr=.f.
do while system("reindex")  && wait until the reindex is finished
   if .not.scr              && in the system program.
      set color to bu+/n
      @ 7,24 to 12,54 double
      set color to gr+/n
      @ 9,25 say "REINDEXING THE MASTER SYSTEM"
      @ 10,34 say "PLEASE WAIT"
      scr=.t.
   endif
enddo
**
waittime=system("waittime")
**
set color to
clear
set color to bu+/n
@ 7,24 to 12,54 double
set color to gr+/n
@ 9,27 say "SETTING THE SALES SYSTEM"
@ 10,34 say "PLEASE WAIT"
**
do open
set view to saldep
**
do chekstop with select()
store j->factdate to tempdd,tempdd1
set color to
clear
do mainmenu
*
*!*****************************************************************
*!                        This procedure allows the user to call to two different
*!    Procedure: MAINMENU    activities depending on the menu selection done.
*!
*!    Called by: SAL6.PRG
*!
*!       Calls: SELECT()       (FOXBASE+ function)
*!            : CHEKSTOP       (procedure in SAL6.PRG)
*!            : INPTQUO        (procedure in SAL6.PRG)
*!            : INPTPRI        (procedure in SAL6.PRG)
*!
*!*****************************************************************
procedure mainmenu
******************
do while .t.
   set color to
   clear
   m_menu=0
   set color to g/n
   @ 3,16,17,62 box " ╒═╕│╛═╘│ "
   set color to gr/n
   @ 5,26 to 7,53 double
```

```
     set color to bu/n
     @  6,32 say "SALES WORKAREA"
     set color to br/n
     @ 11,24 to 14,55
     set color to bg/n
     @ 12,26 prompt "1.DISPLAY/INPUT OF QUOTATION"
     @ 13,26 prompt "2.INPUT OF STANDARD PRICE   "
     menu to m_menu
     do chekstop with select()
     do case
     case m_menu=1
       do inptquo
     case m_menu=2
       do inptpri
     endcase
   enddo
   *
*!*****************************************************************
*!                    This procedure calls the display/input procedures used
*!   Procedure: INPTQUO   to quote the customers' enquires received.
*!
*!   Called by: MAINMENU       (procedure in SAL6.PRG)
*!
*!       Calls: SAL_BROW       (procedure in SAL6.PRG)
*!
*!*****************************************************************
   procedure inptquo
   *****************
   clear
   select 1
   goto top
   do sal_brow
   return
   *
*!*****************************************************************
*!                    This procedure creates the editing form used to quote
*!   Procedure: SAL_FORM  the enquires.
*!
*!   Called by: SAL_EDIT       (procedure in SAL6.PRG)
*!
*!       Calls: SYSTEM()       (function  in SAL6.PRG)
*!
*!*****************************************************************
   procedure sal_form
   ******************
   system1=system("factdate")
   @  0,72 say system1
   set color to &screenatr
   clear
   set color to &promptatr
   @ promptrow-1,0 say promptbar
   *
   set color to bu/n
   @ 5,24 to 7,54 double
   set color to r/n
   @  6,31 say "QUOTATIONS INPUT"
   set color to br/n
   @ 10, 8 say "Article Code      "
   @ 10,37 say "Description  "
   @ 12, 8 say "Quantity         "
   @ 13, 8 say "Normal Price     "
   set color to bg/n
   @ 13,37 say "Quoted Price              "
   set color to br/n
   @ 14, 8 say "Delivery Date   "
   set color to bg/n
   @ 14,37 say "Quoted Delivery Date      "
   set color to g/n
   @ 17, 8 say "Enquire Date    "
   @ 17,37 say "Quotation Date          "
   @ 18, 8 say "Enquire Time        "
   @ 18,37 say "Quotation Time              "
   return
   *
*!*****************************************************************
*!                    This procedure shows the information attached to
*!   Procedure: SAL_SAYS  each one of the enquires.
*!
*!   Called by: SAL_EDIT       (procedure in SAL6.PRG)
*!
*!*****************************************************************
   procedure sal_says
   ******************
   set color to n/w
   @ 10,27 say code
```

```
@ 10,50 say descript
@ 12,26 say enquantit  picture "999.99"
@ 13,26 say enprice    picture "999.99"
set color to ,n/w
@ 13,64 get mquprice    picture "999.99"
@ 14,24 say endeliver
@ 14,62 get mqudeliver  picture "@E"
@ 17,24 say endate
@ 17,62 say qudate
@ 18,27 say entime
@ 18,65 say qutime
clear gets
return
*
*!*********************************************************************
*!                      This procedure allows the input of the quotation
*!    Procedure: SAL_GETS  information attached to each enquire.
*!
*!    Called by: SAL_EDIT       (procedure in SAL6.PRG)
*!
*!*********************************************************************
procedure sal_gets
*****************
set color to ,n/w
@ 13,64 get mquprice   picture "999.99"
@ 14,62 get mqudeliver picture "@E"
read
return
*
*!*********************************************************************
*!                      This procedure stores the default values in the
*!    Procedure: SAL_STOR  quotation variables to be edit.
*!
*!    Called by: SAL_EDIT       (procedure in SAL6.PRG)
*!
*!*********************************************************************
procedure sal_stor
*****************
* ---Initialize memvars with field contents and date().
store quprice   to mquprice
store qudeliver to mqudeliver
return
*
*!*********************************************************************
*!                      This procedure marks the enquires after being
*!    Procedure: MARKING     quoted.
*!
*!    Called by: SAL_EDIT       (procedure in SAL6.PRG)
*!             : SAL_BROW       (procedure in SAL6.PRG)
*!
*!        Calls: QUPRICE<>0()   (FOXBASE+ function)
*!
*!*********************************************************************
procedure marking
*****************
┌──if quprice<>0.and.qudeliver<>ctod('00/00/00')
│    replace status with 'QUOT'
│    unlock
└──endif
return
*
*!*********************************************************************
*!                      This procedure replaces the quotation information
*!    Procedure: SAL_REPL   entered in the <SALES> datafile.
*!
*!    Called by: SAL_EDIT       (procedure in SAL6.PRG)
*!
*!        Calls: EOF()          (FOXBASE+ function)
*!
*!*********************************************************************
procedure sal_repl
*****************
mqudate=j->factdate
mqutime=j->facttime
┌──if tempdd1>tempdd
│    mqudate=(j->factdate-1)
└──endif
┌──if .not. eof()
│    * ---Replace only if there is an available record
│    replace;
│    qudate      with mqudate,;
│    qutime      with mqutime,;
│    quprice     with mquprice,;
│    qudeliver   with mqudeliver
│    unlock
```

```
    endif
return
**
*!*************************************************************************
*!                      This procedure manages the possible conflicts that could
*!    Procedure: ERR_FIX   arise from the fact that we are working in a multi-user
*!                      environment. Its functioning is thoroughly explained in
*!    Called by: SAL6.PRG  one of the thesis's sections.
*!
*!       Calls: RLOCK()        (FOXBASE+ function)
*!
*!*************************************************************************
procedure err_fix
*****************
parameters errnum,mess
**
** Error: File in use by another.
if errnum=108
   save screen to screen
   clear
   set color to gr+/bu
   @ 7,17 clear to 15,61
   @ 7,17 to 15,61 double
   set color to r*+/bu
   @ 11,24 say "Please wait to append a record."
   restore screen from screen
   retry
endif
**
** Error: Record in use by another.
if errnum=109
   save screen to screen
   clear
   time=0
   set color to gr+/bu
   @ 7,17 clear to 15,61
   @ 7,17 to 15,61 double
   set color to r*+/bu
   @ 11,26 clear to 11,54
   @ 11,26 say mess
   @ 12,26 say "Press any key to continue."
   read
   do while .not. rlock().and.time<1000
      time=time+1
   enddo
   if time<1000
      restore screen from screen
      retry
   else
      clear
      set color to gr+/bu
      @ 7,17 clear to 15,61
      @ 7,17 to 15,61 double
      set color to r*+/bu
      @ 11,19 say "Record cannot be locked.Try again later."
      @ 12,26 say "Press any key to continue."
      read
      restore screen from screen
   endif
endif
**
** Error: Record is not locked.
if errnum=130
   time=0
   do while .not. rlock().and.time<1000
      time=time+1
   enddo
   if time<1000
      retry
   endif
endif
*
*!*************************************************************************
*!                      This procedure checks for the status of the game forcing the
*!    Procedure: CHEKSTOP  Sales program to do different things in each case. Its functioning
*!                      is thoroughly explainde in one of the thesis's sections.
*!    Called by: SAL6.PRG
*!             : MAINMENU      (procedure in SAL6.PRG)
*!             : GETKEY        (procedure in SAL6.PRG)
*!
*!       Calls: SYSTEM()       (function  in SAL6.PRG)
*!             : .AND..NOT.SYSTE(FOXBASE+ function)
*!             : SALREPO       (procedure in SAL6.PRG)
*!             : PRT           (procedure in SAL6.PRG)
*!             : .AND..NOT.SHUT.(FOXBASE+ function)
```

```
*!              : .NOT.SYSTEM()  (FOXBASE+ function)
*!              : .AND.SYSTEM()  (FOXBASE+ function)
*!              : STR()          (FOXBASE+ function)
*!
*!          Uses: SYSTEM.DBF
*!
*!****************************************************************************
procedure chekstop
******************
parameters aselect
private shut
shut=.f.
signal=.f.
do while .t.
  do case
  case system("sys_stop").and..not.system("gam_stop")
    if .not.pause
      if system("salreport")
        do salrepo
        do prt
        select 10
        replace salreport with .f.
        unlock
        start=.t.
      endif
      if .not.signal
        save screen to screen1
        set color to
        clear
        set color to gr+/bu
        @ 5,14 clear to 14,64
        @ 5,14 to 14,64 double
        set color to bg+/bu
        @ 9,17 say "THE GAME HAS BEEN FROZEN OR A DAY HAS ENDED."
        @ 11,30 say "WAIT UNTIL RESTART."
        sys_start=.f.
        signal=.t.
      endif
      if system("restart").and..not.shut.and..not.system("gam_stop")
        close databases
        select 10
        use system
        shut=.t.
      else
        if .not.system("restart").and.shut
          set view to saldep
          shut=.f.
        endif
      endif
    else
      filtstat="FIRMLOSTENQU"
      set color to gr+*/bu
      @ 24,70 say "STOPPED"
      set color to
      exit
    endif
  case .not.system("gam_stop").and..not.system("sys_stop")
    if signal.and..not.start.and..not.pause
      restore screen from screen1
    endif
    start=.f.
    sys_start=.f.
    set color to
    @ 24,70 clear to 24,76
    exit
  case system("gam_stop").and.system("sys_stop")
    if sys_start
      if .not.signal1
        set color to
        clear
        set color to bu+/n
        @ 7,24 to 12,54 double
        set color to gr+/n
        @ 9,26 say "SETTING THE MASTER SYSTEM"
        @ 10,34 say "PLEASE WAIT"
        signal1=.t.
        start=.t.
      endif
    else
      close all
      clear all
      set color to
      clear
      set color to bu+/n
      @ 8,24 to 12,54 double
```

```
            set color to gr+/n
            @ 10,29 say "THE GAME HAS FINISHED"
            set color to
            set console off
<        cancel
         endif
      endcase
   enddo
   sys_start=.f.
   aselect=str(aselect)
   select &aselect
   return
   *
*!***********************************************************************
*!                     This procedure allows the input of the standard price
*!    Procedure: INPTPRI  for the selected article.
*!
*!    Called by: MAINMENU       (procedure in SAL6.PRG)
*!
*!       Calls: GETKEY          (procedure in SAL6.PRG)
*!            : FOUND()         (FOXBASE+ function)
*!
*!***********************************************************************
   procedure inptpri
   *****************
   mcode="    "
   set color to
   clear
   private inpchoice
   inpchoice="*"
   set color to r/n
   @ 6,24 to 8,54 double
   set color to bu/n
   @  7,28 say "INPUT OF STANDARD PRICE"
   set color to g/n
   @ 10,13 to 17,65
   set color to br/n
   @ 12,18 say "Code  "
   @ 12,31 say "Descript  "
   @ 15,18 say "Standcost "
   set color to bg/n
   @ 15,40 say "Standprice  "
   *
   set color to gr+/n,n/w
   @ 22,0 say " EDIT: {E}xit <Return> "
   *
   do while .t.
      do getkey with inpchoice,"EC"+returnkey
      do case
      case inpchoice="E"
v        exit
      case inpchoice=returnkey
         set color to ,n/w
         @ 12,25 get mcode picture "@!NNNN" valid mcode<>"    "
         read
         select 2
         find &mcode
         if found()
            store standprice to mstandprice
            @ 12,42 say descript
            @ 15,30 say stand_cost
            @ 15,42 get mstandprice picture "999.99"
            read
            replace standprice with mstandprice
            unlock
         else
            set color to gr+/bu
            @ 9,14 clear to 13,64
            @ 9,14 to 13,64 double
            set color to bg+/bu
            @ 11,21 say "THE CODE INPUT DOES NOT EXIST."
            @ 15,39 say ""
            wait "TRY ANOTHER ONE"
            retry
         endif
      endcase
   enddo
   return
   *
*!***********************************************************************
*!                       This procedure calls the procedures used to edit/quote
*!    Procedure: SAL_EDIT   a selected enquire.
*!
*!    Called by: SAL_BROW       (procedure in SAL6.PRG)
*!
```

```
*!        Calls: RLOCK()        (FOXBASE+ function)
*!             : SAL_FORM       (procedure in SAL6.PRG)
*!             : SAL_STOR       (procedure in SAL6.PRG)
*!             : SAL_SAYS       (procedure in SAL6.PRG)
*!             : GETKEY         (procedure in SAL6.PRG)
*!             : .AND..NOT.SYSTE(FOXBASE+ function)
*!             : SAYLINE        (procedure in SAL6.PRG)
*!             : SAL_GETS       (procedure in SAL6.PRG)
*!             : SAL_REPL       (procedure in SAL6.PRG)
*!             : MARKING        (procedure in SAL6.PRG)
*!
*!*********************************************************************
procedure sal_edit
******************
do while .not. rlock()
enddo
private row,lastpage,editchoice,ndxchoice
private isblank,isunique,isedited
row = promptrow
expr = ""
store .f. to isedited,isblank,isunique
do sal_form
do sal_stor
do sal_says
editchoice = "*"
ndxchoice = "*"
* ---Loop until {Return} is pressed.
* ---The following loop is really a "REPEAT/UNTIL <cond>".
do while .t.
   set color to &promptatr
   @ row,0 say "EXIT/VIEW:  {B}ack  <Return> "
   do getkey with editchoice,"B"+returnkey
   do case
   case editchoice = "B"
      unlock
      exit
   case editchoice = returnkey
      if status="ENQU".and..not.system("SYS_STOP")
         * ---Edit the record.
         isedited = .t.
         do sayline with row,"Press {Ctrl-W} to Exit"
         do sal_gets
         do sal_repl
      endif
   endcase
enddo
if isedited
   do marking
   pagepaint=.t.
endif
return
*
*!*********************************************************************
*!                         This procedure displays all the existing enquires regardless
*!     Procedure: SAL_DISP   of their status (quoted,firm or lost).
*!
*!     Called by: SAL_BROW       (procedure in SAL6.PRG)
*!
*!        Calls: ROW()          (FOXBASE+ function)
*!             : EOF()          (FOXBASE+ function)
*!             : SPACE()        (FOXBASE+ function)
*!
*!*********************************************************************
procedure sal_disp
******************
parameter row,listrecs
if listrecs > 1
   * ---Display heading when listing the entire page.
   set color to &statusatr
   @ rowtop-1,0
   @ rowtop-1,0 say ""
   do case
   case pancol = 1
      ?? "ARTICLE_CODE-DESCRIPTION-----------PRICE-----QUANTITY----DELIVERY_DATE--STATUS"
   case pancol = 2
   endcase
   * ---Clear the window area.
   set color to &windowatr
   @ rowtop,0 clear to rowbottom,79
endif
* ---Display the records.
set heading off
set color to &windowatr
@ row-1,0 say ""
do case
```

```
case pancol = 1
    list next listrecs code,"      ",descript,"",quprice,"   ",enquantit,"        ",qudeliver,"   ",status off
    if listrecs>1
        lastdisp=row()-2
    endif
case pancol = 2
endcase
set heading on
if eof()
    goto bottom
endif
if listrecs>1
    lastpos=qudeliver
endif
if listrecs > 1.and.browrec=1
    select 10
    replace newenqrec with space(9)
    unlock
    select 1
endif
return
*
*!*****************************************************************************
*!                      This procedure gets a keyboard input done by
*!    Procedure: GETKEY     the user.
*!
*!    Called by: INPTPRI        (procedure in SAL6.PRG)
*!             : SAL_EDIT       (procedure in SAL6.PRG)
*!             : SAL_CODE       (procedure in SAL6.PRG)
*!             : SAL_STAT       (procedure in SAL6.PRG)
*!             : SAL_BROW       (procedure in SAL6.PRG)
*!
*!       Calls: SYSTEM()        (function  in SAL6.PRG)
*!             : SUBSTR()       (FOXBASE+ function)
*!             : RECNO()        (FOXBASE+ function)
*!             : POS            (procedure in SAL6.PRG)
*!             : IIF()          (FOXBASE+ function)
*!             : INKEY()        (FOXBASE+ function)
*!             : UPPER()        (FOXBASE+ function)
*!             : CHR()          (FOXBASE+ function)
*!             : SELECT()       (FOXBASE+ function)
*!             : CHEKSTOP       (procedure in SAL6.PRG)
*!
*!*****************************************************************************
procedure getkey
****************
parameter choice,keychars
private keycode
choice = "*"
do while .not. (choice $ keychars)
    tempdd=tempdd1
    tempdd1=j->factdate
    if system("sys_stop")
        store .t. to goodfin
        store .t. to scr
        do while system("goodserv")
            if scr
                set typeahead to 0
                set color to bu+/n
                @ 11,25 to 13,55 double
                set color to gr+*/n
                @ 12,27 say "calculating sales response"
                scr=.f.
            endif
            goodfin=.f.
        enddo
        if .not.goodfin
            set typeahead to 20
            goodfin=.t.
            store (system("FACTDATE")-1) to mmfactdate
            select 1
            set filter to qudate=mmfactdate.or.status$ filtstat0
            goto top
            select 10
            replace goodserv with .f.
            unlock
            select 1
            pagepaint=.t.
            newenq=.t.
exit
        endif
    endif
    waittime=system("waittime")
    xx=system("newenqrec")
    if substr(xx,1,3)<>"   "
```

```
        recnum=recno()
        do pos with xx,lastpos,firstpos
      endif
      move=iif(newenq,.f.,.t.)
      if .not.move.and..not.inedit
        set typeahead to 0
        keycode=0
        choice=" "
        bkey=.t.
      else
        if bkey
          set typeahead to 20
          bkey=.f.
        endif
      endif
      keycode = inkey()
      if keycode > 0
        choice = upper(chr(keycode))
        if choice $ "E"
          pause=.f.
        else
          if system("FACTTIME")<>"8:00"
            pause=.t.
          endif
        endif
        do chekstop with select()
      endif
      if .not.move
        exit
      endif
    enddo
    return
    *
*!*************************************************************************
*!                        This procedure displays a line in a certain
*!    Procedure: SAYLINE    screen position.
*!
*!    Called by: SAL_EDIT       (procedure in SAL6.PRG)
*!
*!*************************************************************************
    procedure sayline
    *****************
    parameter row,strg
    set color to &promptatr
    @ row,0 clear
    @ row,0 say strg
    return
    *
*!*************************************************************************
*!                        This procedure calculates the possible article codes
*!    Procedure: SAL_AREA0  to filter the <SALES> datafile by.
*!
*!    Called by: FILT_CODE      (procedure in SAL6.PRG)
*!
*!       Calls: STR()          (FOXBASE+ function)
*!            : EOF()          (FOXBASE+ function)
*!            : SAL_CODE       (procedure in SAL6.PRG)
*!
*!*************************************************************************
    procedure sal_area0
    *******************
    private i,ii
    select 2
    goto top
    i=0
    ii=str(0,1)
    code&ii="ALL"
    i=i+1
    ii=str(i,1)
    do while .not. eof()
      if code<>"    "
        public code&ii
        code&ii=code
      endif
      if .not. eof()
        i=i+1
        ii=str(i,1)
        skip
      endif
    enddo
    select 1
    do sal_code with i
    return
    *
*!*************************************************************************
```

```
*!                         This procedure calculates the possible statuses
*!     Procedure: SAL_AREA1   to filter the <SALES> datafile by.
*!
*!     Called by: FILT_STAT      (procedure in SAL6.PRG)
*!
*!         Calls: SAL_STAT       (procedure in SAL6.PRG)
*!
*!**********************************************************************
   procedure sal_area1
   *******************
   public status0,status1,status2,status3,status4
   status0="ENQU"
   status1="ENQU"+"QUOT"+"FIRM"
   status2="LOST"
   status3="FIRM"
   select 1
   do sal_stat
   return
   *
*!**********************************************************************
*!                         This procedure calculates the filter of the <SALES>
*!     Procedure: SAL_CODE   datafile based on the selected article code.
*!
*!     Called by: SAL_AREA0      (procedure in SAL6.PRG)
*!
*!         Calls: STR()          (FOXBASE+ function)
*!              : GETKEY         (procedure in SAL6.PRG)
*!
*!**********************************************************************
   procedure sal_code
   ******************
   parameter numbcode
   private n,nn,codekey,phrase
   n=0
   phrase=""
   codekey=""
   nn=str(n,1)
   set color to &promptatr
   do while n<numbcode
      phrase=phrase+"  "+nn+"-"+code&nn
      codekey=codekey+nn
      n=n+1
      nn=str(n,1)
   enddo
   @ rowprompt-1,0 clear
   @ rowprompt-1,0 say promptbar
   @ rowprompt,0 clear
   @ rowprompt,0 say "SET CODE:"+phrase
   do getkey with codchoice,codekey+returnkey
   do case
   case codchoice = returnkey
                 return
   case codchoice $ codekey
      filtcode=code&codchoice
   endcase
   return
   *
*!**********************************************************************
*!                         This procedure calculates the filter of the <SALES>
*!     Procedure: SAL_STAT   based on the selected status.
*!
*!     Called by: SAL_AREA1      (procedure in SAL6.PRG)
*!
*!         Calls: GETKEY         (procedure in SAL6.PRG)
*!
*!**********************************************************************
   procedure sal_stat
   ******************
   set color to &promptatr
   @ rowprompt-1,0 clear
   @ rowprompt-1,0 say promptbar
   @ rowprompt,0 clear
   @ rowprompt,0 say "SET STATUS:  0-ENQU+QUOT  1-ENQU+QUOT+FIRM  2-LOST  3-FIRM  "
   do getkey with ordchoice,"0123"+returnkey
   do case
   case ordchoice = returnkey
                 return
   case ordchoice = "0"
      filtstat=status&ordchoice
   case ordchoice = "1"
      filtstat=status&ordchoice
   case ordchoice = "2"
      filtstat=status&ordchoice
   case ordchoice = "3"
      filtstat=status&ordchoice
```

```
endcase
return
*
*!*************************************************************************
*!                          This procedure filters the <SALES> datafile based
*!      Procedure: FILT_CODE  on the selected article code.
*!
*!      Called by: SAL_BROW        (procedure in SAL6.PRG)
*!
*!         Calls: SAL_AREA0       (procedure in SAL6.PRG)
*!              : EOF()          (FOXBASE+ function)
*!              : RECNO()         (FOXBASE+ function)
*!
*!*************************************************************************
procedure filt_code
*******************
do sal_area0
 if codchoice="0"
    set filter to wait<=waittime.and.code<>filtcode.and.status$ filtstat
 else
    set filter to wait<=waittime.and.code=filtcode.and.status$ filtstat
 endif
goto top
 if eof()
    set color to &promptatr
    @ rowprompt,0 clear
    @ rowprompt,0 say "No matching records."
    wait
    @ rowprompt,0 clear
    set filter to wait<=waittime.and.status $ filtstat0
    goto top
 endif
recnumtop=recno()
firstpos=qudeliver
pagepaint=.t.
set color to &screenatr
clear
return
*
*!*************************************************************************
*!                          This procedure filters the <SALES> datafile based
*!      Procedure: FILT_STAT   on the selected status.
*!
*!      Called by: SAL_BROW        (procedure in SAL6.PRG)
*!
*!         Calls: SAL_AREA1       (procedure in SAL6.PRG)
*!              : EOF()          (FOXBASE+ function)
*!              : RECNO()         (FOXBASE+ function)
*!
*!*************************************************************************
procedure filt_stat
*******************
do sal_area1
 if codchoice="0"
    set filter to wait<=waittime.and.code<>filtcode.and.status$ filtstat
 else
    set filter to wait<=waittime.and.code=filtcode.and.status$ filtstat
 endif
goto top
 if eof()
    set color to &promptatr
    @ rowprompt,0 clear
    @ rowprompt,0 say "No matching records."
    wait
    @ rowprompt,0 clear
    set filter to wait<=waittime.and.status $ filtstat0
    goto top
 endif
recnumtop=recno()
firstpos=qudeliver
pagepaint=.t.
set color to &screenatr
clear
return
*
*!*************************************************************************
*!                          This procedure manages all the report generations
*!      Procedure: SALREPO    for the Sales department.
*!
*!      Called by: CHEKSTOP        (procedure in SAL6.PRG)
*!              : PRT             (procedure in SAL6.PRG)
*!
*!         Calls: EOF()           (FOXBASE+ function)
*!              : INT()           (FOXBASE+ function)
*!              : VAL()           (FOXBASE+ function)
```

```
*!              : SUBSTR()      (FOXBASE+ function)
*!              : DTOC()        (FOXBASE+ function)
*!              : SALAVE        (procedure in SAL6.PRG)
*!              : SALFMT        (procedure in SAL6.PRG)
*!
*!**********************************************************************
    procedure salrepo
    *****************
    set color to
    store 0 to n,mave_enqu,mave_quot,mper_quot,mave_firm,;
    mper_firm,movr_delti,mper_ontim,mave_qpric,mper_qpric,;
    mper_qdeli,mper_fpric,mper_fdeli,mdeliverie,mtodeliv,mvolume
    select 3
    goto top
    ╔══do while .not. eof()
    ║  actcode=code
    ║  mdescript=descript
    ║  ╔══do while code=actcode
    ║  ║  actweek=int(val(substr(dtoc(date),1,2))/5)
    ║  ║  ╔══do while int(val(substr(dtoc(date),1,2))/5)=actweek.and.code=actcode
    ║  ║  ║  n=n+1
    ║  ║  ║  do salave
    ║  ║  ║  ┌──if .not. eof()
    ║  ║  ║  │  skip
    ║  ║  ║  ├──else
    ║  ║  v══════════════exit
    ║  ║  ║  └──endif
    ║  ║  ╚══enddo
    ║  ║  skip -1
    ║  ║  do salfmt
    ║  ║  skip
    ║  ║  store 0 to n,mave_enqu,mave_quot,mper_quot,mave_firm,;
    ║  ║  mper_firm,movr_delti,mper_ontim,mave_qpric,mper_qpric,;
    ║  ║  mper_qdeli,mper_fpric,mper_fdeli,mdeliverie,mtodeliv,mvolume
    ║  ╚══enddo
    ╚══enddo
    signal=.f.
    return
    *
*!**********************************************************************
*!                      This procedure calculates the average of the <SALES>
*!    Procedure: SALAVE   datafile's key fields.
*!
*!    Called by: SALREPO        (procedure in SAL6.PRG)
*!
*!**********************************************************************
    procedure salave
    ****************
    mave_enqu=(mave_enqu+ave_enqu)/n
    mave_quot=(mave_quot+ave_quot)/n
    mper_quot=(mper_quot+per_quot)/n
    mave_firm=(mave_firm+ave_firm)/n
    mper_firm=(mper_firm+per_firm)/n
    movr_delti=(movr_delti+ovr_deltim)/n
    mper_ontim=(mper_ontim+per_ontime)/n
    mave_qpric=(mave_qpric+ave_qprice)/n
    mper_qpric=(mper_qpric+per_qprice)/n
    mper_qdeli=(mper_qdeli+per_qdeliv)/n
    mper_fpric=(mper_fpric+per_fprice)/n
    mper_fdeli=(mper_fdeli+per_fdeliv)/n
    mdeliverie=(mdeliverie+deliveries)/n
    mtodeliv=(mtodeliv+todeliv)/n
    mvolume=(mvolume+volume)/n
    return
    *
*!**********************************************************************
*!                      This procedure displays the reports concerning the
*!    Procedure: SALFMT   Sales department's performance.
*!
*!    Called by: SALREPO        (procedure in SAL6.PRG)
*!
*!       Calls: INKEY()         (FOXBASE+ function)
*!
*!**********************************************************************
    procedure salfmt
    ****************
    set color to
    clear
    set color to r/n
    @ 1,24 say "╔══════════════════════════════╗"
    @ 2,24 say "║  SALES DEPARTMENT'S PAST     ║"
    @ 3,24 say "║   PERFORMANCE (Page 1)       ║"
    @ 4,24 say "╚══════════════════════════════╝"
    @ 7, 0 say "Week"
    @ 7,15 say actweek picture "99"
```

```
@ 7,20 say "Date"
@ 7,30 say date
@ 9, 0 say "Code"
@ 9,13 say actcode
@ 9,20 say "Descript"
@ 9,30 say mdescript
@ 12,27 say "DAY     OVERALL"
@ 13, 0 say "Number of enquires"
@ 13,25 say ave_enqu picture "999.99"
@ 13,34 say mave_enqu picture "999.99"
@ 13,65 say "DAY     OVERALL"
@ 14, 0 say "Number of quotations"
@ 14,25 say ave_quot picture "999.99"
@ 14,34 say mave_quot picture "999.99"
@ 14,43 say "% over enquires"
@ 14,63 say per_quot picture "999.99"
@ 14,72 say mper_quot picture "999.99"
@ 15, 0 say "Number of sales orders"
@ 15,25 say ave_firm picture "999.99"
@ 15,34 say mave_firm picture "999.99"
@ 15,43 say "% over quotations"
@ 15,63 say per_firm picture "999.99"
@ 15,72 say mper_firm picture "999.99"
@ 16, 0 say "To be delivered"
@ 16,25 say todeliv picture "999999"
@ 16,34 say mtodeliv picture "999999"
@ 17, 0 say "Shippings"
@ 17,25 say deliveries picture "999999"
@ 17,34 say mdeliverie picture "999999"
@ 17,43 say "% shipped on time"
@ 17,63 say per_ontime picture "999.99"
@ 17,72 say mper_ontim picture "999.99"
@ 19,45 say "DAY           OVERALL"
@ 20,13 say "Volume of Sales orders (£)"
@ 20,42 say volume picture "9999999.99"
@ 20,56 say mvolume picture "9999999.99"
@ 21,13 say "Average Delivery Delay"
@ 21,44 say ovr_deltim picture "99.99"
@ 21,59 say movr_delti picture "99.99"
*
? inkey(3)
*
set color to
clear
set color to r/n
@ 4,23 say "╔══════════════════════════════════════╗"
@ 5,23 say "║    SALES DEPARTMENT'S PAST           ║"
@ 6,23 say "║     PERFORMANCE (Page 2)             ║"
@ 7,23 say "╚══════════════════════════════════════╝"
@ 10, 2 say "Week"
@ 10,17 say actweek picture "99"
@ 10,22 say "Date"
@ 10,32 say date
@ 12, 2 say "Code"
@ 12,15 say actcode
@ 12,22 say "Descript"
@ 12,32 say mdescript
@ 14, 2 say "Quoted Enquires' Success % :"
@ 14,45 say "Achieved orders success % :"
@ 15,22 say "DAY"
@ 15,32 say "OVERALL"
@ 15,63 say "DAY"
@ 15,70 say "OVERALL"
@ 16, 3 say "Price"
@ 16,21 say per_qprice picture "999.99"
@ 16,33 say mper_qpric picture "999.99"
@ 16,44 say "Price"
@ 16,62 say per_fprice picture "999.99"
@ 16,71 say mper_fpric picture "999.99"
@ 17, 3 say "Delivery Date"
@ 17,21 say per_qdeliv picture "999.99"
@ 17,33 say mper_qdeli picture "999.99"
@ 17,44 say "Delivery Date"
@ 17,62 say per_fdeliv picture "999.99"
@ 17,71 say mper_fdeli picture "999.99"
*
? inkey(3)
*
set color to
clear
return
*
*!*********************************************************************
*!                         This procedure allows the user to select the reports'
*!      Procedure: PRT        output device (screen or printer).
```

```
*!
*!     Called by: CHEKSTOP       (procedure in SAL6.PRG)
*!
*!          Calls: SYS()         (FOXBASE+ function)
*!               : SALREPO       (procedure in SAL6.PRG)
*!
*!**************************************************************************
    procedure prt
    *************
      do while .t.
         set color to
         clear
         m_menu=0
         set color to bu+/n
         @ 4,18 to 17,61
         @ 6,24 to 8,54 double
         set color to br+/n
         @  7,29 say "PRINT OUT OF REPORTS"
         set color to bu+/n
         @ 10,24,14,50 box " ╓─╖ ║╜─╙║ "
         set color to bg+/n
         @ 11,29 prompt "1.TO PRINTER.     "
         @ 12,29 prompt "2.PAST BEHAVIOUR."
         @ 13,29 prompt "3.EXIT.           "
         set color to bu+/n
         @ 16,29 say "DEVICE :"
         ?? sys(101)
         menu to m_menu
         do case
         case m_menu=1
            set device to print
         case m_menu=2
            do salrepo
            set device to screen
            set color to
                       exit
         case m_menu=3
            set device to screen
            set color to
                       exit
         endcase
      enddo
    return
    *
*!**************************************************************************
*!                           This procedure displays the enquires allowing also
*!     Procedure: SAL_BROW    their edition.It's based on the "browse" algorithm which is
*!                           thoroughly explained in one of the thesis's sections.
*!     Called by: INPTQUO        (procedure in SAL6.PRG)
*!
*!          Calls: CHR()         (FOXBASE+ function)
*!               : SYSTEM()      (function in SAL6.PRG)
*!               : EOF()         (FOXBASE+ function)
*!               : RECNO()       (FOXBASE+ function)
*!               : XX<>SPACE()   (FOXBASE+ function)
*!               : POS           (procedure in SAL6.PRG)
*!               : SAL_DISP      (procedure in SAL6.PRG)
*!               : FI()          (FOXBASE+ function)
*!               : GETKEY        (procedure in SAL6.PRG)
*!               : BOF()         (FOXBASE+ function)
*!               : SAL_EDIT      (procedure in SAL6.PRG)
*!               : FILT_STAT     (procedure in SAL6.PRG)
*!               : FILT_CODE     (procedure in SAL6.PRG)
*!               : .NOT.SYSTEM() (FOXBASE+ function)
*!               : ACCEPT        (procedure in SAL6.PRG)
*!               : MARKING       (procedure in SAL6.PRG)
*!
*!**************************************************************************
    procedure sal_brow
    ******************
    private pancol,panmax,panlast,recnumtop,recnumlast,skiprecs
    private home,endkey,uparrow,downarrow,leftarrow,rightarrow
    private row,rowtop,keystrokes,rowlast,pagepaint
    * ---Initialize constants.
    home = chr(1)
    endkey = chr(6)
    uparrow = chr(5)
    downarrow = chr(24)
    leftarrow = chr(19)
    rightarrow = chr(4)
    keystrokes = "EACL"+uparrow+downarrow+home+leftarrow+;
    rightarrow+endkey+pgdn+pgup+returnkey
    rowtop = 1
    skiprecs = rowbottom - rowtop + 1
    scrr=.f.
```

```
waittime=system("waittime")
goto top
if eof()
    do while .t.
        if .not.scrr
            set color to bu+/n
            @ 8,24 to 13,54 double
            set color to gr+/n
            @ 10,29 say "NO RECORDS AVAILABLE"
            @ 11,34 say "PLEASE WAIT"
            scrr=.t.
        endif
        waittime=system("waittime")
        goto top
        recnum=recno()
        xx=system("newenqrec")
        if xx<>space(9)
            recnum=recno()
            do pos with xx,lastpos,firstpos
        endif
        if newenq
            goto top
            exit
        endif
    enddo
endif
* ---Initialize local variables.
goto top
row = rowtop
recnum = recno()
recnumtop = recnum
firstpos=qudeliver
pagepaint = .t.
pancol = 1
panlast = 1
panmax = 1
* ---Perform BROWSE.
set color to &screenatr
clear
* ---The following loop is really a "REPEAT/UNTIL <cond>".
do while .t.
    if pagepaint
        goto top
        recnumtop=recno()
        if pancol = panlast
            row = rowtop
        endif
        panlast = pancol
        if newenq
            waittime=system("waittime")
            newenq=.f.
            goto top
            recnumtop=recno()
            if browrec<>0
                bot=.f.
                do while recno()<>browrec.and..not.bot
                    skip
                    if eof()
                        goto bottom
                        bot=.t.
                    else
                        row=row+1
                    endif
                enddo
            endif
        endif
        recnum = recno()
        goto recnumtop
        do sal_disp with (rowtop),skiprecs
        goto recnum
        pagepaint = .f.
        browrec=0
    endif
    set color to &promptatr
    @ rowprompt-1,0 say promptbar
    @ rowprompt,0 say ;
    "BROWSE:  (E)xit  (A)ccept  (C)ode  fi(L)ter  <Arrows>  <PgDn>  <PgUp>  <Return> "
    set color to bu+,n
    @ rowprompt+1,18 say "CURRENT SELECTION: CODE="
    ?? filtcode
    @ rowprompt+1,47 say " STATUS="
    ?? filtstat
    set color to &promptatr
    @ row,0 say chr(16)
    @ row,77 say chr(17)
```

```
do getkey with choice,keystrokes
do while choice $ uparrow+downarrow.and..not.newenq
    @ row,0 say " "
    @ row,77 say " "
    if move
        if choice=uparrow
            skip -1
            do case
            case bof()
                goto top
            case row > rowtop
                row = row - 1
            otherwise
                recnumtop = recno()
                firstpos=qudeliver
                * ---Scroll window down.
                scroll rowtop,0,rowbottom,79,-1
                do sal_disp with row,1
            endcase
        else
            skip
            do case
            case eof()
                goto bottom
            case row < rowbottom
                row = row + 1
            otherwise
                * ---Adjust top-of-page record pointer.
                recnum = recno()
                goto recnumtop
                skip
                recnumtop = recno()
                firstpos=qudeliver
                goto recnum
                * ---Scroll window up.
                scroll rowtop,0,rowbottom,79,1
                do sal_disp with row,1
            endcase
        endif
    endif
    set color to &promptatr
    @ row,0 say chr(16)
    @ row,77 say chr(17)
    do getkey with choice,keystrokes
enddo
do case
case choice = "E"
    exit
case choice = returnkey
    inedit=.t.
    if system("SYS_STOP").or.(.not.system("SYS_STOP").and.status="ENQU")
        save screen to salist
        recnum=recno()
        do sal_edit
        set color to
        clear
        if .not.pagepaint
            set color to &screenatr
            restore screen from salist
            if status="QUOT"
                do sal_disp with row,1
                set color to &promptatr
                @ row,0 say chr(16)
                @ row,77 say chr(17)
                goto recnum
            endif
        endif
    endif
    inedit=.f.
case choice = pgdn
    if .not. eof()
        goto recnumtop
        skip skiprecs
        if eof()
            goto bottom
        endif
        recnumtop = recno()
        firstpos=qudeliver
        pagepaint = .t.
    endif
case choice = pgup
    if .not. bof()
        goto recnumtop
        skip -skiprecs
        if bof()
```

```
               goto top
            endif
            recnumtop = recno()
            firstpos=qudeliver
            pagepaint = .t.
         endif
      case choice = "L"
         do filt_stat
      case choice = "C"
         do filt_code
      case choice = home
         pagepaint = (pancol < > 1)
         pancol = 1
      case choice = leftarrow
         if pancol > 1
            pancol = pancol - 1
            pagepaint = .t.
         endif
      case choice = rightarrow
         if pancol < panmax
            pancol = pancol + 1
            pagepaint = .t.
         endif
      case choice = endkey
         pagepaint = (pancol < > panmax)
         pancol = panmax
      case choice = "A"
         if .not.system("SYS_STOP").and.move
            if status="ENQU"
               do accept
               do marking
               recnum=recno()
               do sal_disp with row,1
               set color to &promptatr
               @ row,0 say chr(16)
               @ row,77 say chr(17)
               goto recnum
            endif
         endif
      endcase
   enddo
   set filter to wait<=waittime.and.status $ filtstat0
   goto top
   return
   *
*!*****************************************************************************
*!                       This procedure obtains the most updated values of
*!     Function: SYSTEM()    the fields of the <SYSTEM> datafile.
*!
*!     Called by: SAL6.PRG
*!              : CHEKSTOP      (procedure in SAL6.PRG)
*!              : SAL_FORM      (procedure in SAL6.PRG)
*!              : GETKEY        (procedure in SAL6.PRG)
*!              : SAL_BROW      (procedure in SAL6.PRG)
*!
*!         Calls: STR()         (FOXBASE+ function)
*!              : SELECT()      (FOXBASE+ function)
*!
*!*****************************************************************************
   procedure system
   ****************
   parameters fieldd
   store str(select(),2) to base
   select 10
   goto top
   system=&fieldd
   select &base
   return system
   *
*!*****************************************************************************
*!                       This procedure creates the "view" used during the
*!     Procedure: OPEN        run of the Sales program.
*!
*!     Called by: SAL6.PRG
*!
*!          Uses: MARKET.DBF
*!              : BILL.DBF
*!              : SALPAST.DBF
*!
*!       Indexes: DEMARK.IDX
*!              : BILLNA.IDX
*!              : CDSAL.IDX
*!
*!*****************************************************************************
   procedure open
```

```
**************
set exclusive off
select 1
use market index demark
set filter to wait<=waittime.and.status $ filtstat0
select 2
use bill index billna
select 3
use salpast index cdsal
set safety off
create view saldep from environment all
set safety on
close databases
**
*!*********************************************************************
*!                        This procedure checks the screen position of the
*!    Procedure: POS       ewly arrived enquire.
*!
*!    Called by: GETKEY         (procedure in SAL6.PRG)
*!             : SAL_BROW       (procedure in SAL6.PRG)
*!
*!        Calls: VAL()          (FOXBASE+ function)
*!             : SUBSTR()       (FOXBASE+ function)
*!             : CTOD()         (FOXBASE+ function)
*!             : .OR.()         (FOXBASE+ function)
*!             : MIN()          (FOXBASE+ function)
*!             : SPACE()        (FOXBASE+ function)
*!
*!*********************************************************************
procedure pos
*************
parameter yy,lpos,fpos
rpos1=iif(substr(yy,1,3)="   ",0,val(substr(yy,1,3)))
rpos2=iif(substr(yy,4,3)="   ",0,val(substr(yy,4,3)))
rpos3=iif(substr(yy,7,3)="   ",0,val(substr(yy,7,3)))
if rpos1<>0
  goto rpos1
  pos1=qudeliver
else
  pos1=ctod("01/01/99")
endif
if rpos2<>0
  goto rpos2
  pos2=qudeliver
else
  pos2=ctod("01/01/99")
endif
if rpos3<>0
  goto rpos3
  pos3=qudeliver
else
  pos3=ctod("01/01/99")
endif
if pos1<>ctod("01/01/99").and.((pos1>=fpos.and.pos1<=lpos).or.(lastdisp<20))
  pagepaint=.t.
endif
if pos2<>ctod("01/01/99").and.((pos2>=fpos.and.pos2<=lpos).or.(lastdisp<20))
  pagepaint=.t.
endif
if pos3<>ctod("01/01/99").and.((pos3>=fpos.and.pos3<=lpos).or.(lastdisp<20))
  pagepaint=.t.
endif
minpos=min(pos1,pos2)
minpos=min(minpos,pos3)
if pos1=minpos
  browrec=rpos1
else
  if pos2=minpos
    browrec=rpos2
  else
    browrec=rpos3
  endif
endif
select 10
replace newenqrec with space(9)
unlock
select 1
if pagepaint
  newenq=.t.
endif
if .not.newenq
  goto recnum
endif
return
**
```

```
*!**********************************************************************
*!                          This procedure checks the acceptance or not
*!     Procedure: ACCEPT     of the input done.
*!
*!     Called by: SAL_BROW        (procedure in SAL6.PRG)
*!
*!         Calls: EOF()          (FOXBASE+ function)
*!
*!**********************************************************************
   procedure accept
   ****************
   store j->factdate to mqudate
   store j->facttime to mqutime
     if tempdd1>tempdd
       store (j->factdate-1) to mqudate
     endif
     if .not. eof()
       * ---Replace only if there is an available record
       replace qudate      with mqudate
       replace qutime      with mqutime
       unlock
     endif
   return
   *: EOF: SAL6.ACT
```

System: SAL6.PRG
Author: JUAN IGNACIO IGARTUA
08/05/91 18:32:05
Tree Diagram

```
SAL6.PRG
 |   SYSTEM.DBF (database)
 |---------ERROR() (function  in ?)
 |---------MESSAGE() (function  in ?)
 |---------ERR_FIX    (procedure in SAL6.PRG)
 |   └---------RLOCK() (function  in ?)
 |---------CTOD() (function  in ?)
 |---------REPLICATE() (function  in ?)
 |---------CHR() (function  in ?)
 |---------SYSTEM() (function  in SAL6.PRG)
 |   |---------STR() (function  in ?)
 |   └---------SELECT() (function  in ?)
 |---------OPEN    (procedure in SAL6.PRG)
 |       MARKET.DBF (database)
 |       BILL.DBF (database)
 |       SALPAST.DBF (database)
 |---------SELECT() (function  in ?)
 |---------CHEKSTOP    (procedure in SAL6.PRG)
 |   |   SYSTEM.DBF (database)
 |   |---------SYSTEM() (function  in SAL6.PRG)
 |   |   |---------STR() (function  in ?)
 |   |   └---------SELECT() (function  in ?)
 |   |---------.AND..NOT.SYSTEM() (function  in ?)
 |   |---------SALREPO    (procedure in SAL6.PRG)
 |   |   |---------EOF() (function  in ?)
 |   |   |---------INT() (function  in ?)
 |   |   |---------VAL() (function  in ?)
 |   |   |---------SUBSTR() (function  in ?)
 |   |   |---------DTOC() (function  in ?)
 |   |   |---------SALAVE    (procedure in SAL6.PRG)
 |   |   └---------SALFMT    (procedure in SAL6.PRG)
 |   |       └---------INKEY() (function  in ?)
 |   |---------PRT    (procedure in SAL6.PRG)
 |   |   |---------SYS() (function  in ?)
 |   |   └---------SALREPO    (procedure in SAL6.PRG)
 |   |       |---------EOF() (function  in ?)
 |   |       |---------INT() (function  in ?)
 |   |       |---------VAL() (function  in ?)
 |   |       |---------SUBSTR() (function  in ?)
 |   |       |---------DTOC() (function  in ?)
 |   |       |---------SALAVE    (procedure in SAL6.PRG)
 |   |       └---------SALFMT    (procedure in SAL6.PRG)
 |   |           └---------INKEY() (function  in ?)
 |   |---------.AND..NOT.SHUT.AND..NOT.SYSTEM() (function  in ?)
 |   |---------.NOT.SYSTEM() (function  in ?)
 |   |---------.AND.SYSTEM() (function  in ?)
 |   └---------STR() (function  in ?)
 └---------MAINMENU    (procedure in SAL6.PRG)
     |---------SELECT() (function  in ?)
     |---------CHEKSTOP    (procedure in SAL6.PRG)
     |   |   SYSTEM.DBF (database)
     |   |---------SYSTEM() (function  in SAL6.PRG)
     |   |   |---------STR() (function  in ?)
     |   |   └---------SELECT() (function  in ?)
     |   |---------.AND..NOT.SYSTEM() (function  in ?)
     |   |---------SALREPO    (procedure in SAL6.PRG)
     |   |   |---------EOF() (function  in ?)
     |   |   |---------INT() (function  in ?)
     |   |   |---------VAL() (function  in ?)
     |   |   |---------SUBSTR() (function  in ?)
     |   |   |---------DTOC() (function  in ?)
     |   |   |---------SALAVE    (procedure in SAL6.PRG)
     |   |   └---------SALFMT    (procedure in SAL6.PRG)
     |   |       └---------INKEY() (function  in ?)
     |   |---------PRT    (procedure in SAL6.PRG)
     |   |   |---------SYS() (function  in ?)
     |   |   └---------SALREPO    (procedure in SAL6.PRG)
     |   |       |---------EOF() (function  in ?)
     |   |       |---------INT() (function  in ?)
     |   |       |---------VAL() (function  in ?)
     |   |       |---------SUBSTR() (function  in ?)
     |   |       |---------DTOC() (function  in ?)
     |   |       |---------SALAVE    (procedure in SAL6.PRG)
     |   |       └---------SALFMT    (procedure in SAL6.PRG)
     |   |           └---------INKEY() (function  in ?)
     |   |---------.AND..NOT.SHUT.AND..NOT.SYSTEM() (function  in ?)
     |   |---------.NOT.SYSTEM() (function  in ?)
```

```
    .AND.SYSTEM() (function  in ?)
    STR() (function  in ?)
INPTQUO     (procedure in SAL6.PRG)
  SAL_BROW      (procedure in SAL6.PRG)
    CHR() (function  in ?)
    SYSTEM() (function  in SAL6.PRG)
      STR() (function  in ?)
      SELECT() (function  in ?)
    EOF() (function  in ?)
    RECNO() (function  in ?)
    XX< >SPACE() (function  in ?)
    POS     (procedure in SAL6.PRG)
      VAL() (function  in ?)
      SUBSTR() (function  in ?)
      CTOD() (function  in ?)
      .OR.() (function  in ?)
      MIN() (function  in ?)
      SPACE() (function  in ?)
    SAL_DISP     (procedure in SAL6.PRG)
      ROW() (function  in ?)
      EOF() (function  in ?)
      SPACE() (function  in ?)
    FI() (function  in ?)
    GETKEY      (procedure in SAL6.PRG)
      SYSTEM() (function  in SAL6.PRG)
        STR() (function  in ?)
        SELECT() (function  in ?)
      SUBSTR() (function  in ?)
      RECNO() (function  in ?)
      POS      (procedure in SAL6.PRG)
        VAL() (function  in ?)
        SUBSTR() (function  in ?)
        CTOD() (function  in ?)
        .OR.() (function  in ?)
        MIN() (function  in ?)
        SPACE() (function  in ?)
      IIF() (function  in ?)
      INKEY() (function  in ?)
      UPPER() (function  in ?)
      CHR() (function  in ?)
      SELECT() (function  in ?)
      CHEKSTOP      (procedure in SAL6.PRG)
        SYSTEM.DBF (database)
        SYSTEM() (function  in SAL6.PRG)
          STR() (function  in ?)
          SELECT() (function  in ?)
        .AND..NOT.SYSTEM() (function  in ?)
        SALREPO      (procedure in SAL6.PRG)
          EOF() (function  in ?)
          INT() (function  in ?)
          VAL() (function  in ?)
          SUBSTR() (function  in ?)
          DTOC() (function  in ?)
          SALAVE      (procedure in SAL6.PRG)
          SALFMT      (procedure in SAL6.PRG)
            INKEY() (function  in ?)
        PRT      (procedure in SAL6.PRG)
          SYS() (function  in ?)
          SALREPO      (procedure in SAL6.PRG)
            EOF() (function  in ?)
            INT() (function  in ?)
            VAL() (function  in ?)
            SUBSTR() (function  in ?)
            DTOC() (function  in ?)
            SALAVE      (procedure in SAL6.PRG)
            SALFMT      (procedure in SAL6.PRG)
              INKEY() (function  in ?)
        .AND..NOT.SHUT.AND..NOT.SYSTEM() (function  in ?)
        .NOT.SYSTEM() (function  in ?)
        .AND.SYSTEM() (function  in ?)
        STR() (function  in ?)
    BOF() (function  in ?)
    SAL_EDIT      (procedure in SAL6.PRG)
      RLOCK() (function  in ?)
      SAL_FORM      (procedure in SAL6.PRG)
        SYSTEM() (function  in SAL6.PRG)
          STR() (function  in ?)
          SELECT() (function  in ?)
      SAL_STOR      (procedure in SAL6.PRG)
      SAL_SAYS      (procedure in SAL6.PRG)
      GETKEY      (procedure in SAL6.PRG)
        SYSTEM() (function  in SAL6.PRG)
          STR() (function  in ?)
          SELECT() (function  in ?)
        SUBSTR() (function  in ?)
```

```
|        |    |    |----------RECNO() (function  in ?)
|        |    |    |----------POS      (procedure in SAL6.PRG)
|        |    |    |     |----------VAL() (function  in ?)
|        |    |    |     |----------SUBSTR() (function  in ?)
|        |    |    |     |----------CTOD() (function  in ?)
|        |    |    |     |----------.OR.() (function  in ?)
|        |    |    |     |----------MIN() (function  in ?)
|        |    |    |     └----------SPACE() (function  in ?)
|        |    |    |----------IIF() (function  in ?)
|        |    |    |----------INKEY() (function  in ?)
|        |    |    |----------UPPER() (function  in ?)
|        |    |    |----------CHR() (function  in ?)
|        |    |    |----------SELECT() (function  in ?)
|        |    |    └----------CHEKSTOP    (procedure in SAL6.PRG)
|        |    |        |  SYSTEM.DBF (database)
|        |    |        |----------SYSTEM() (function  in SAL6.PRG)
|        |    |        |     |----------STR() (function  in ?)
|        |    |        |     └----------SELECT() (function  in ?)
|        |    |        |----------.AND..NOT.SYSTEM() (function  in ?)
|        |    |        |----------SALREPO     (procedure in SAL6.PRG)
|        |    |        |     |----------EOF() (function  in ?)
|        |    |        |     |----------INT() (function  in ?)
|        |    |        |     |----------VAL() (function  in ?)
|        |    |        |     |----------SUBSTR() (function  in ?)
|        |    |        |     |----------DTOC() (function  in ?)
|        |    |        |     |----------SALAVE     (procedure in SAL6.PRG)
|        |    |        |     |----------SALFMT     (procedure in SAL6.PRG)
|        |    |        |         └----------INKEY() (function  in ?)
|        |    |        |----------PRT     (procedure in SAL6.PRG)
|        |    |        |     |----------SYS() (function  in ?)
|        |    |        |     └----------SALREPO     (procedure in SAL6.PRG)
|        |    |        |         |----------EOF() (function  in ?)
|        |    |        |         |----------INT() (function  in ?)
|        |    |        |         |----------VAL() (function  in ?)
|        |    |        |         |----------SUBSTR() (function  in ?)
|        |    |        |         |----------DTOC() (function  in ?)
|        |    |        |         |----------SALAVE     (procedure in SAL6.PRG)
|        |    |        |         └----------SALFMT     (procedure in SAL6.PRG)
|        |    |        |             └----------INKEY() (function  in ?)
|        |    |        |----------.AND..NOT.SHUT.AND..NOT.SYSTEM() (function  in ?)
|        |    |        |----------.NOT.SYSTEM() (function  in ?)
|        |    |        |----------.AND.SYSTEM() (function  in ?)
|        |    |        └----------STR() (function  in ?)
|        |    |----------.AND..NOT.SYSTEM() (function  in ?)
|        |    |----------SAYLINE     (procedure in SAL6.PRG)
|        |    |----------SAL_GETS     (procedure in SAL6.PRG)
|        |    |----------SAL_REPL     (procedure in SAL6.PRG)
|        |    |    └----------EOF() (function  in ?)
|        |    └----------MARKING     (procedure in SAL6.PRG)
|        |        └----------QUPRICE<>0.AND.QUDELIVER<>CTOD() (function  in ?)
|        |----------FILT_STAT     (procedure in SAL6.PRG)
|        |    |----------SAL_AREA1     (procedure in SAL6.PRG)
|        |    |    └----------SAL_STAT     (procedure in SAL6.PRG)
|        |    |        └----------GETKEY     (procedure in SAL6.PRG)
|        |    |            |----------SYSTEM() (function  in SAL6.PRG)
|        |    |            |     |----------STR() (function  in ?)
|        |    |            |     └----------SELECT() (function  in ?)
|        |    |            |----------SUBSTR() (function  in ?)
|        |    |            |----------RECNO() (function  in ?)
|        |    |            |----------POS     (procedure in SAL6.PRG)
|        |    |            |     |----------VAL() (function  in ?)
|        |    |            |     |----------SUBSTR() (function  in ?)
|        |    |            |     |----------CTOD() (function  in ?)
|        |    |            |     |----------.OR.() (function  in ?)
|        |    |            |     |----------MIN() (function  in ?)
|        |    |            |     └----------SPACE() (function  in ?)
|        |    |            |----------IIF() (function  in ?)
|        |    |            |----------INKEY() (function  in ?)
|        |    |            |----------UPPER() (function  in ?)
|        |    |            |----------CHR() (function  in ?)
|        |    |            |----------SELECT() (function  in ?)
|        |    |            └----------CHEKSTOP     (procedure in SAL6.PRG)
|        |    |                |  SYSTEM.DBF (database)
|        |    |                |----------SYSTEM() (function  in SAL6.PRG)
|        |    |                |     |----------STR() (function  in ?)
|        |    |                |     └----------SELECT() (function  in ?)
|        |    |                |----------.AND..NOT.SYSTEM() (function  in ?)
|        |    |                |----------SALREPO     (procedure in SAL6.PRG)
|        |    |                |     |----------EOF() (function  in ?)
|        |    |                |     |----------INT() (function  in ?)
|        |    |                |     |----------VAL() (function  in ?)
|        |    |                |     |----------SUBSTR() (function  in ?)
|        |    |                |     |----------DTOC() (function  in ?)
|        |    |                |     |----------SALAVE     (procedure in SAL6.PRG)
|        |    |                |     |----------SALFMT     (procedure in SAL6.PRG)
```

```
                                       INKEY() (function  in ?)
                              PRT      (procedure in SAL6.PRG)
                                    SYS() (function  in ?)
                                    SALREPO      (procedure in SAL6.PRG)
                                        EOF() (function  in ?)
                                        INT() (function  in ?)
                                        VAL() (function  in ?)
                                        SUBSTR() (function  in ?)
                                        DTOC() (function  in ?)
                                        SALAVE      (procedure in SAL6.PRG)
                                        SALFMT      (procedure in SAL6.PRG)
                                            INKEY() (function  in ?)
                               .AND..NOT.SHUT.AND..NOT.SYSTEM() (function  in ?)
                               .NOT.SYSTEM() (function  in ?)
                               .AND.SYSTEM() (function  in ?)
                               STR() (function  in ?)
              EOF() (function  in ?)
              RECNO() (function  in ?)
         FILT_CODE      (procedure in SAL6.PRG)
              SAL_AREA0      (procedure in SAL6.PRG)
                   STR() (function  in ?)
                   EOF() (function  in ?)
                   SAL_CODE      (procedure in SAL6.PRG)
                       STR() (function  in ?)
                       GETKEY      (procedure in SAL6.PRG)
                           SYSTEM() (function  in SAL6.PRG)
                                STR() (function  in ?)
                                SELECT() (function  in ?)
                           SUBSTR() (function  in ?)
                           RECNO() (function  in ?)
                           POS      (procedure in SAL6.PRG)
                                VAL() (function  in ?)
                                SUBSTR() (function  in ?)
                                CTOD() (function  in ?)
                                .OR.() (function  in ?)
                                MIN() (function  in ?)
                                SPACE() (function  in ?)
                           IIF() (function  in ?)
                           INKEY() (function  in ?)
                           UPPER() (function  in ?)
                           CHR() (function  in ?)
                           SELECT() (function  in ?)
                           CHEKSTOP      (procedure in SAL6.PRG)
                         SYSTEM.DBF (database)
                               SYSTEM() (function  in SAL6.PRG)
                                    STR() (function  in ?)
                                    SELECT() (function  in ?)
                               .AND..NOT.SYSTEM() (function  in ?)
                               SALREPO      (procedure in SAL6.PRG)
                                    EOF() (function  in ?)
                                    INT() (function  in ?)
                                    VAL() (function  in ?)
                                    SUBSTR() (function  in ?)
                                    DTOC() (function  in ?)
                                    SALAVE      (procedure in SAL6.PRG)
                                    SALFMT      (procedure in SAL6.PRG)
                                        INKEY() (function  in ?)
                               PRT      (procedure in SAL6.PRG)
                                    SYS() (function  in ?)
                                    SALREPO      (procedure in SAL6.PRG)
                                        EOF() (function  in ?)
                                        INT() (function  in ?)
                                        VAL() (function  in ?)
                                        SUBSTR() (function  in ?)
                                        DTOC() (function  in ?)
                                        SALAVE      (procedure in SAL6.PRG)
                                        SALFMT      (procedure in SAL6.PRG)
                                            INKEY() (function  in ?)
                               .AND..NOT.SHUT.AND..NOT.SYSTEM() (function  in ?)
                               .NOT.SYSTEM() (function  in ?)
                               .AND.SYSTEM() (function  in ?)
                               STR() (function  in ?)
              EOF() (function  in ?)
              RECNO() (function  in ?)
         .NOT.SYSTEM() (function  in ?)
         ACCEPT      (procedure in SAL6.PRG)
              EOF() (function  in ?)
         MARKING      (procedure in SAL6.PRG)
             QUPRICE<>0.AND.QUDELIVER<>CTOD() (function  in ?)
INPTPRI      (procedure in SAL6.PRG)
    GETKEY      (procedure in SAL6.PRG)
         SYSTEM() (function  in SAL6.PRG)
              STR() (function  in ?)
              SELECT() (function  in ?)
         SUBSTR() (function  in ?)
```

```
|   |-----------RECNO() (function  in ?)
|   |-----------POS      (procedure in SAL6.PRG)
|   |     |-----------VAL() (function  in ?)
|   |     |-----------SUBSTR() (function  in ?)
|   |     |-----------CTOD() (function  in ?)
|   |     |-----------.OR.() (function  in ?)
|   |     |-----------MIN() (function  in ?)
|   |     └-----------SPACE() (function  in ?)
|   |-----------IIF() (function  in ?)
|   |-----------INKEY() (function  in ?)
|   |-----------UPPER() (function  in ?)
|   |-----------CHR() (function  in ?)
|   |-----------SELECT() (function  in ?)
|   └-----------CHEKSTOP      (procedure in SAL6.PRG)
|       |   SYSTEM.DBF (database)
|       |-----------SYSTEM() (function  in SAL6.PRG)
|       |     |-----------STR() (function  in ?)
|       |     └-----------SELECT() (function  in ?)
|       |-----------.AND..NOT.SYSTEM() (function  in ?)
|       |-----------SALREPO      (procedure in SAL6.PRG)
|       |     |-----------EOF() (function  in ?)
|       |     |-----------INT() (function  in ?)
|       |     |-----------VAL() (function  in ?)
|       |     |-----------SUBSTR() (function  in ?)
|       |     |-----------DTOC() (function  in ?)
|       |     |-----------SALAVE      (procedure in SAL6.PRG)
|       |     └-----------SALFMT      (procedure in SAL6.PRG)
|       |          └-----------INKEY() (function  in ?)
|       |-----------PRT      (procedure in SAL6.PRG)
|       |     |-----------SYS() (function  in ?)
|       |     └-----------SALREPO      (procedure in SAL6.PRG)
|       |          |-----------EOF() (function  in ?)
|       |          |-----------INT() (function  in ?)
|       |          |-----------VAL() (function  in ?)
|       |          |-----------SUBSTR() (function  in ?)
|       |          |-----------DTOC() (function  in ?)
|       |          |-----------SALAVE      (procedure in SAL6.PRG)
|       |          └-----------SALFMT      (procedure in SAL6.PRG)
|       |               └-----------INKEY() (function  in ?)
|       |-----------.AND..NOT.SHUT.AND..NOT.SYSTEM() (function  in ?)
|       |-----------.NOT.SYSTEM() (function  in ?)
|       |-----------.AND.SYSTEM() (function  in ?)
|       └-----------STR() (function  in ?)
└-----------FOUND() (function  in ?)
```

System: SAL6.PRG
Author: JUAN IGNACIO IGARTUA
08/05/91 18:31:58
Database Structure Summary

---

4 databases in the system
SYSTEM.DBF
MARKET.DBF
BILL.DBF
SALPAST.DBF

Structure for database : SYSTEM.DBF
Number of data records : 1
Last updated : 07/01/91 at 9:17

Field	Field name	Type	Width	Dec	Start	End
1	STARTDATE	Date	8		1	8
2	FACTDATE	Date	8		9	16
3	FACTTIME	Character	4		17	20
4	SYS_STOP	Logical	1		21	21
5	GAM_STOP	Logical	1		22	22
6	HOUR_RATE	Numeric	7	2	23	29
7	TOT_OVERH	Numeric	10	2	30	39
8	NEWENQ	Logical	1		40	40
9	NEWENQREC	Character	9		41	49
10	FREQUENCY	Numeric	3		50	52
11	GOODSERV	Logical	1		53	53
12	REINDEX	Logical	1		54	54
13	STATRESO	Character	4		55	58
14	STATCODE	Character	4		59	62
15	WAITTIME	Numeric	4		63	66
16	RESTART	Logical	1		67	67
17	RESTATUS	Logical	1		68	68
18	THROUGHPUT	Numeric	10	2	69	78
19	ASSETS	Numeric	10	2	79	88
20	SYSTOCK	Numeric	10	2	89	98
21	SALREPORT	Logical	1		99	99
22	PLAREPORT	Logical	1		100	100
23	ENQUPERIOD	Numeric	1		101	101
** Total **			102			

FoxDoc did not find any associated index files

Used by: SAL6.PRG
: CHEKSTOP (procedure in SAL6.PRG)

Structure for database : MARKET.DBF
Number of data records : 10
Last updated : 07/01/91 at 9:17

Field	Field name	Type	Width	Dec	Start	End
1	CODE	Character	4		1	4
2	DESCRIPT	Character	20		5	24
3	ENQUANTIT	Numeric	4		25	28
4	ENPRICE	Numeric	6	2	29	34
5	ENDATE	Date	8		35	42
6	ENTIME	Character	4		43	46
7	ENDELIVER	Date	8		47	54
8	QUPRICE	Numeric	6	2	55	60
9	QUDATE	Date	8		61	68
10	QUTIME	Character	4		69	72
11	QUDELIVER	Date	8		73	80
12	STATUS	Character	4		81	84
13	REALDELIV	Date	8		85	92
14	DELIVERED	Logical	1		93	93
15	SUCCSPRICE	Numeric	6	2	94	99
16	SUCCSDELIV	Numeric	6	2	100	105
17	WAIT	Numeric	4		106	109
** Total **			110			

This database appears to be associated with index file(s):
: DEMARK.IDX (QUDELIVER)

Used by: OPEN (procedure in SAL6.PRG)

Structure for database : BILL.DBF
Number of data records : 5
Last updated : 07/01/91 at 9:09

Field	Field name	Type	Width	Dec	Start	End

1	CODE	Character	4		1	4
2	DESCRIPT	Character	20		5	24
3	MINDELIVDD	Numeric	2		25	26
4	MAXDELIVDD	Numeric	2		27	28
5	MAXQUANTIT	Numeric	6	2	29	34
6	MINQUANTIT	Numeric	6	2	35	40
7	STANDPRICE	Numeric	6	2	41	46
8	MATCOST	Numeric	6	2	47	52
9	TOT_ENQU	Numeric	4		53	56
10	PRICE_LIMT	Numeric	6	2	57	62
11	DISCOUNT	Numeric	4	2	63	66
12	QUANT_DISC	Numeric	6	2	67	72
13	DELIV_LIMT	Numeric	4		73	76
14	MRKT_PRICE	Numeric	6	2	77	82
15	MRKT_LIMT	Numeric	6	2	83	88
16	ONTIMEDELV	Numeric	6	2	89	94
17	ONTIMLIMT	Numeric	6	2	95	100
18	STANDHOURS	Numeric	6	2	101	106
19	OVERHEAD	Numeric	6	2	107	112
20	STAND_COST	Numeric	6	2	113	118
21	NUM_ENQU	Numeric	4		119	122
22	RESOUR_COD	Character	4		123	126
23	PROCESTIME	Numeric	5	2	127	131
24	SET_UP	Character	7		132	138
** Total **			139			

This database appears to be associated with index file(s):
: BILLNA.IDX (CODE)

Used by: OPEN (procedure in SAL6.PRG)

Structure for database : SALPAST.DBF
Number of data records : 1
Last updated : 07/01/91 at 9:17

Field	Field name	Type	Width	Dec	Start	End
1	CODE	Character	4		1	4
2	DESCRIPT	Character	20		5	24
3	DATE	Date	8		25	32
4	AVE_ENQU	Numeric	6	2	33	38
5	AVE_QUOT	Numeric	6	2	39	44
6	PER_QUOT	Numeric	6	2	45	50
7	AVE_FIRM	Numeric	6	2	51	56
8	PER_FIRM	Numeric	6	2	57	62
9	OVR_DELTIM	Numeric	4	2	63	66
10	PER_ONTIME	Numeric	6	2	67	72
11	AVE_QPRICE	Numeric	6	2	73	78
12	PER_QPRICE	Numeric	6	2	79	84
13	PER_QDELIV	Numeric	6	2	85	90
14	PER_FPRICE	Numeric	6	2	91	96
15	PER_FDELIV	Numeric	6	2	97	102
16	DELIVERIES	Numeric	6		103	108
17	VOLUME	Numeric	10	2	109	118
** Total **			119			

This database appears to be associated with index file(s):
: CDSAL.IDX (CODE+DTOC(DATE))

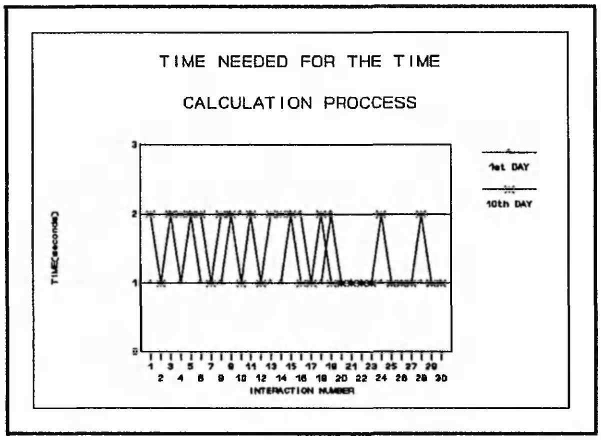
Used by: OPEN (procedure in SAL6.PRG)

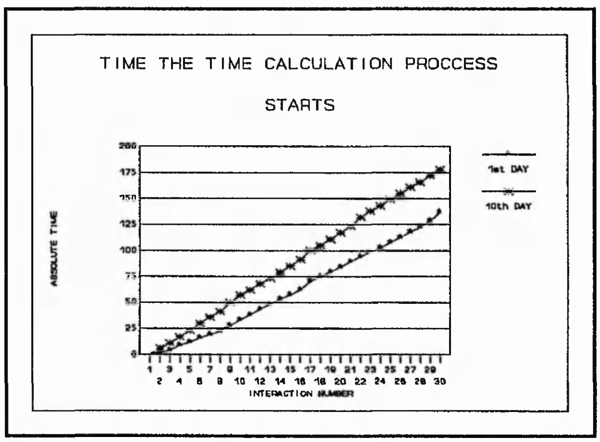
System: SAL6.PRG
Author: JUAN IGNACIO IGARTUA
08/05/91 18:31:59
Data Dictionary

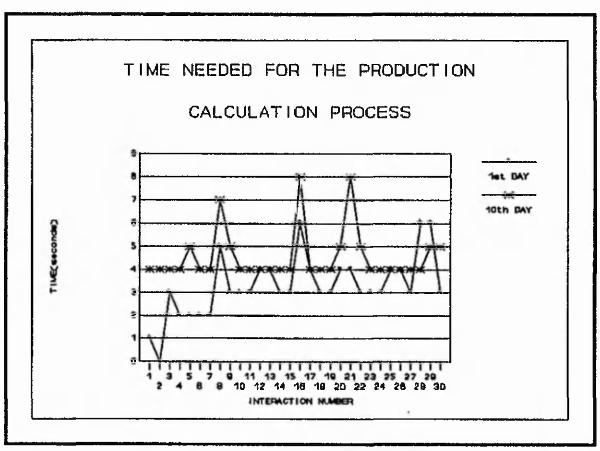
Field Name	Type	Len	Dec	Database
ASSETS	N	10	2	SYSTEM.DBF
AVE_ENQU	N	6	2	SALPAST.DBF
AVE_FIRM	N	6	2	SALPAST.DBF
AVE_QPRICE	N	6	2	SALPAST.DBF
AVE_QUOT	N	6	2	SALPAST.DBF
CODE	C	4	0	BILL.DBF
				SALPAST.DBF
				MARKET.DBF
DATE	D	8	0	SALPAST.DBF
DELIVERED	L	1	0	MARKET.DBF
DELIVERIES	N	6	0	SALPAST.DBF
DELIV_LIMT	N	4	0	BILL.DBF
DESCRIPT	C	20	0	SALPAST.DBF
				BILL.DBF
				MARKET.DBF
DISCOUNT	N	4	2	BILL.DBF
ENDATE	D	8	0	MARKET.DBF
ENDELIVER	D	8	0	MARKET.DBF
ENPRICE	N	6	2	MARKET.DBF
ENQUANTIT	N	4	0	MARKET.DBF
ENQUPERIOD	N	1	0	SYSTEM.DBF
ENTIME	C	4	0	MARKET.DBF
FACTDATE	D	8	0	SYSTEM.DBF
FACTTIME	C	4	0	SYSTEM.DBF
FREQUENCY	N	3	0	SYSTEM.DBF
GAM_STOP	L	1	0	SYSTEM.DBF
GOODSERV	L	1	0	SYSTEM.DBF
HOUR_RATE	N	7	2	SYSTEM.DBF
MATCOST	N	6	2	BILL.DBF
MAXDELIVDD	N	2	0	BILL.DBF
MAXQUANTIT	N	6	2	BILL.DBF
MINDELIVDD	N	2	0	BILL.DBF
MINQUANTIT	N	6	2	BILL.DBF
MRKT_LIMT	N	6	2	BILL.DBF
MRKT_PRICE	N	6	2	BILL.DBF
NEWENQ	L	1	0	SYSTEM.DBF
NEWENQREC	C	9	0	SYSTEM.DBF
NUM_ENQU	N	4	0	BILL.DBF
ONTIMEDELV	N	6	2	BILL.DBF
ONTIMLIMT	N	6	2	BILL.DBF
OVERHEAD	N	6	2	BILL.DBF
OVR_DELTIM	N	4	2	SALPAST.DBF
PER_FDELIV	N	6	2	SALPAST.DBF
PER_FIRM	N	6	2	SALPAST.DBF
PER_FPRICE	N	6	2	SALPAST.DBF
PER_ONTIME	N	6	2	SALPAST.DBF
PER_QDELIV	N	6	2	SALPAST.DBF
PER_QPRICE	N	6	2	SALPAST.DBF
PER_QUOT	N	6	2	SALPAST.DBF
PLAREPORT	L	1	0	SYSTEM.DBF
PRICE_LIMT	N	6	2	BILL.DBF
PROCESTIME	N	5	2	BILL.DBF
QUANT_DISC	N	6	2	BILL.DBF
QUDATE	D	8	0	MARKET.DBF
QUDELIVER	D	8	0	MARKET.DBF
QUPRICE	N	6	2	MARKET.DBF
QUTIME	C	4	0	MARKET.DBF
REALDELIV	D	8	0	MARKET.DBF
REINDEX	L	1	0	SYSTEM.DBF
RESOUR_COD	C	4	0	BILL.DBF
RESTART	L	1	0	SYSTEM.DBF

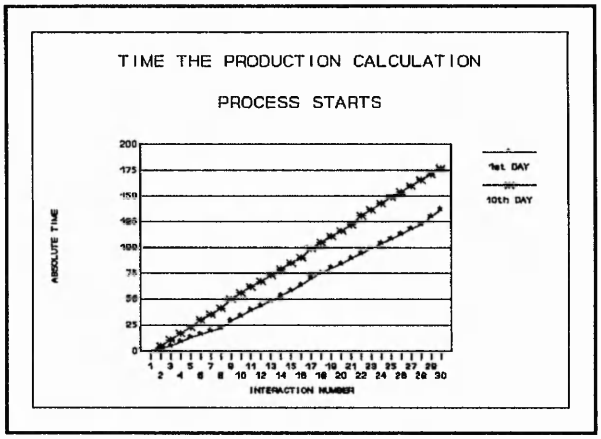
RESTATUS	L	1	0	SYSTEM.DBF
SALREPORT	L	1	0	SYSTEM.DBF
SET_UP	C	7	0	BILL.DBF
STANDHOURS	N	6	2	BILL.DBF
STANDPRICE	N	6	2	BILL.DBF
STAND_COST	N	6	2	BILL.DBF
STARTDATE	D	8	0	SYSTEM.DBF
STATCODE	C	4	0	SYSTEM.DBF
STATRESO	C	4	0	SYSTEM.DBF
STATUS	C	4	0	MARKET.DBF
SUCCSDELIV	N	6	2	MARKET.DBF
SUCCSPRICE	N	6	2	MARKET.DBF
SYSTOCK	N	10	2	SYSTEM.DBF
SYS_STOP	L	1	0	SYSTEM.DBF
THROUGHPUT	N	10	2	SYSTEM.DBF
TOT_ENQU	N	4	0	BILL.DBF
TOT_OVERH	N	10	2	SYSTEM.DBF
VOLUME	N	10	2	SALPAST.DBF
WAIT	N	4	0	MARKET.DBF
WAITTIME	N	4	0	SYSTEM.DBF

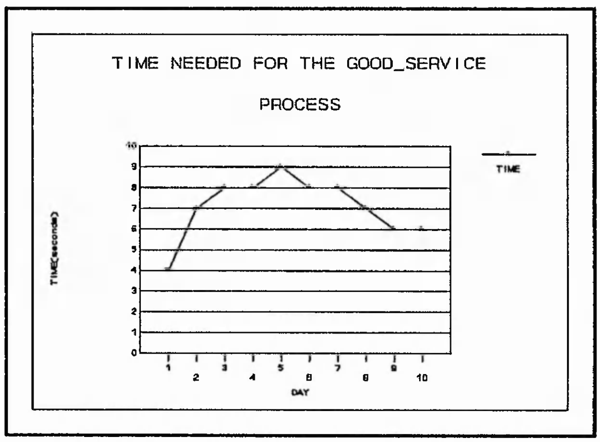
APPENDIX D
Evaluation of the game

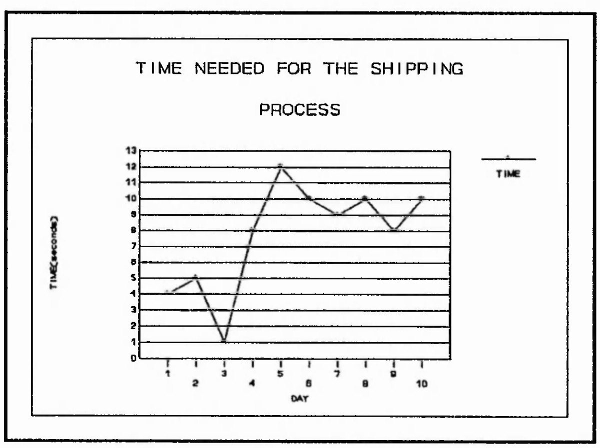

TIME NEEDED FOR THE TIME
CALCULATION PROCCESS
TIME(seconds)
INTERACTION NUMBER
1st DAY
10th DAY

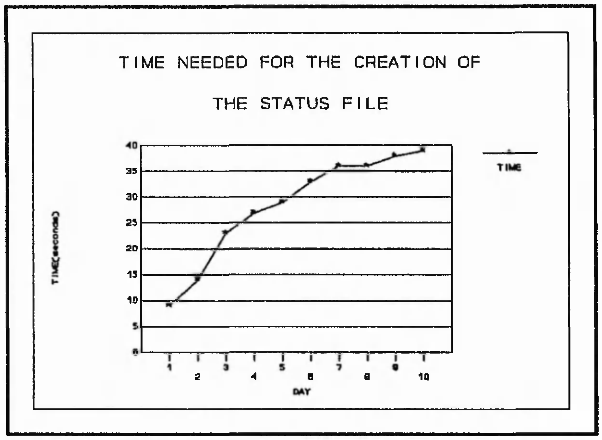

TIME THE TIME CALCULATION PROCCESS
STARTS
ABSOLUTE TIME
INTERACTION NUMBER
1st DAY
10th DAY

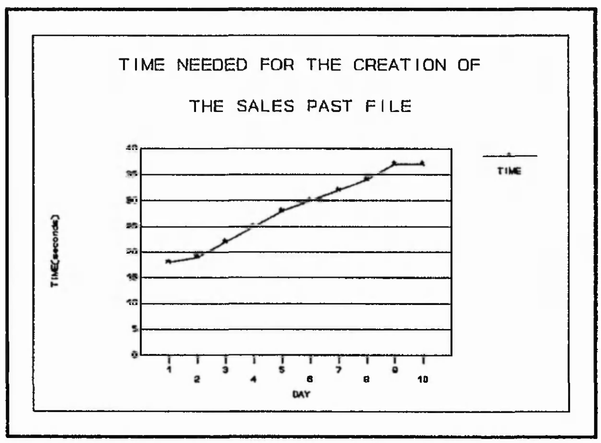

TIME NEEDED FOR THE PRODUCTION
CALCULATION PROCESS
TIME(seconds)
INTERACTION NUMBER
1st DAY
10th DAY

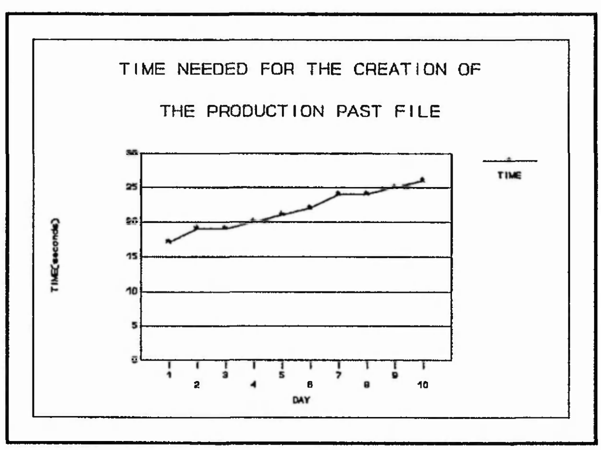

TIME THE PRODUCTION CALCULATION
PROCESS STARTS
ABSOLUTE TIME
INTERACTION NUMBER
1st DAY
10th DAY


TIME NEEDED FOR THE GOOD_SERVICE
PROCESS
TIME(seconds)
DAY
TIME


TIME NEEDED FOR THE SHIPPING
PROCESS
TIME(seconds)
DAY
TIME


TIME NEEDED FOR THE CREATION OF
THE STATUS FILE
TIME(seconds)
DAY
TIME


TIME NEEDED FOR THE CREATION OF
THE SALES PAST FILE
TIME(seconds)
DAY
TIME


TIME NEEDED FOR THE CREATION OF
THE PRODUCTION PAST FILE
TIME(seconds)
DAY
TIME

APPENDIX E
Seventh National Conference on Production Research

AN INTERACTIVE GAMING-SIMULATION OF MANUFACTURING ORGANISATIONS.

Authors:
G.TRANFIELD
J. I. IGARTUA

ABSTRACT.

Manufacturing organisations are inevitably divided into individual departments at some level. These departments will have their own objectives and the success of the organisation depends on the extent to which these departmental objectives are in line with the overall objectives of the organisations.Sadly this is not the case in many western manufacturing companies.

Many companies would therefore like to realign their departmental objectives and an important step in this process is to educate their staff. They must understand how their decisions affect other parts of the company and through this the overall company performance. This is inevitably a dynamic process as decisions are being made throughout an organisation in response to a continually changing environment.

Work is therefore taking place at Nottingham Polytechnic to develop a dynamic, multi-user, computer-based gaming-simulation which is designed to highlight the interactions between departments. This paper reports the progress of that work.

1.-INTERNAL CO-ORDINATION OF MANUFACTURING ORGANISATIONS.

Many authors have argued that the main problems facing manufacturing industry, particularly in the West, are organisational rather than technological. Schonberger [1] gives a classic example comparing two factories manufacturing similar products. The Japanese factory uses older technology but is more successful by almost any basis of measurement which he attributes to their superior organisation.

More recently a study conducted by the management consultants A.T.Kearny [2] concluded that in Britain of the 9 billion pounds spend on factory automation each year, some 600 million are wasted.

To some extent organisational problems arise from the sheer scale of Manufacturing companies and for many years Burbidge [3,4,5] has been pointing out the benefits of breaking down shop floor activities into product based rather than functional elements. In fact some companies have taken this philosophy to extremes and created product based business units within a factory containing the production and supporting business functions (accounting, marketing, purchasing, etc) in physical distinct areas [6].

However it is unlikely that many companies will be able to pursue this option. Most will continue to have an organisation based on functional activities like marketing, production planning, purchasing and manufacturing. The challenge faced by these companies is then to co-ordinate the activities in the individual areas.

This is not an easy task. A main theme of Goldratt's work is that:-

"The total of local optima is not equal to the optimum of the total." [7]

and that the targets set for most managers encourages them move towards local optima [8] at the expense of the global optimum. Additionally many of the reasons given in the literature (i.e. [9]) for the problem with Computer Aided Production Management Systems can be viewed as problems arising from the lack of co-ordination between departments. An overstated Master Production Schedule is the lack of co-ordination of Production and Sales. Inaccurate data often arises because the department supplying the data does not realise the importance of the data to the functioning of other departments. Lack of education again means that one department does not realise what another department is doing with the system.

This lack of co-ordination is probably most serious when an organisation is trying to undergo major change [10]. A survey of UK industry last year [11] concluded:-

"Compartmentalisation in an organisation creates diversiveness and breeds conflict which is a significant obstacle for change. Some managers' clear awareness of this does not seem to have a major impact on the type of initiatives planned for the future."

2.-ROLE OF EDUCATION.

There is clearly no simple solution to this problem. It must be solved on many fronts by providing sufficient and appropriate resources, generating enthusiasm and commitment, and providing the right kind of leadership from the top of the organisation. However, the first basic step in solving the problem must be for people to understand the problem, and the importance of education in this respect has been the subject of at least one book [12].

Traditionally education in this area has involved lectures and discussions and more recently there has been a considerable number of videos produced on the subject by such companies as the Oliver Wight organisation. These are useful ways to provide information but are rather limited in the way they can help people understand a dynamic situation of the sort that exists within a manufacturing organisation. Additionally any activity that requires the students to just listen and not participate tends to get very boring.

More recently a number of computer aided learning programs have been developed and the use of 14 packages in the field of P.O.M. (Production and Operations Management) was reviewed by Wieters & Williams [13].Unfortunately the majority of these packages are simply programs that can check whether the user is able to perform the standard (and often unrealistic) calculations that have come to form a significant part of courses in this area, and do not contribute to an understanding of the dynamics of the interactions in a manufacturing organisation.

3.-GAMING-SIMULATION.

The idea of gaming-simulation is defined by Greenblat [14]. Like a game it provides the players with goals. At the same time it in some way reflects real life and is consequently a simulation.

The majority of gaming-simulations applied to industry are business games which concentrate on different companies competing for a single market i.e. they concentrate on the external interactions of the company rather than the internal ones.

Other games (eg TEAMSKILL [14]) are based on the internal decisions made in the company but a team is given the problem. They have to provide one "answer" for each simulated month which is a form defining the next month's plan including scheduling of production, maintenance and purchasing .

This kind of gaming-simulation, however, ignores two important aspects of the real manufacturing situation. Firstly in the game there are pressures to bring the participants together. They are jointly presented with the problem and they are required to present their answer jointly by filling in one form. This contrasts with the real situation where people work in separate offices and are judged independently.

Secondly there is a very artificial treatment of time. In the game one month's results are presented and decisions are made for the next month. In reality, events (eg the receipt of an order) occur throughout the working period.

4.-NOTTINGHAM POLYTECHNIC GAME.

The aim of the current work is to produce a gaming-simulation that overcomes these problems and will help people understand the nature of the interdepartmental conflict in manufacturing and how it affects the overall factory performance. In order to do this we require a game where the players are performing the different functional tasks in an organisation and where the decisions of one player affect the opportunities offered to other players.

In order to achieve this we are developing a game based on a network of personal computers with the main data stored on a central file server using a multi-user relational database management system (FOXBASE+). The program being run in one of the computers is used to control the game as well as to generate enquiries. The player who is performing the sales role is then required to respond to the enquiries by quoting a price and delivery date. The success of his quotation is dependent on these two factors based on a simple linear relationship as shown in Fig.1.

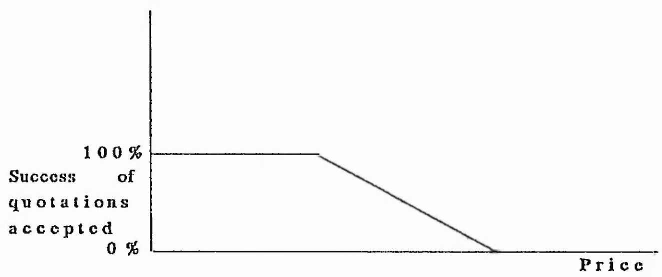


Figure 1

The resultant sales are then passed on to the production department which has to schedule the set-ups and production of the products. The current model is restricted to 5 products, 3 machines and a single operation to produce the existing products, although the basic structure of the system is designed to allow for more complex models to be built in the future.

Feedback from Production to Sales then exists within the game because the rate of enquiries is dependent on the success of the factory to meet the quoted delivery dates.

The game is played interactively with the screens being refreshed as enquiries are received and orders are accepted. At daily intervals the company performance is displayed on screen with the option of printed reports being created.

5.-CONCLUSION.

At present the game is very much at a prototype stage. This partly reflects the fact that although the FOXBASE+ system provided a very convenient means of creating certain features of the system (screens, menus, etc) it required quite a lot of effort to create an interactive multi-user system with a common view of time.

We are hoping to use the game on participants from manufacturing companies and to do this two main factors have to be decided. Firstly, should the players be told the principles on which the simulation is operating or should it be viewed as a "black-box" whose characteristics need to be discovered by trail and error? Secondly, to what extent should the organisers of the game try to manipulate the interactions between the players?

What we originally intend to do is to be purposely divisive, warning each player that the other is unlikely to do very well and encourage them to seek their own independent goals. Later we hope to discuss the idea of co-operation and let then discover the effect of operating in this way.

The game also needs development into other departments but feedback from the use of a simple 2 departmental model is required before moving in this direction.

REFERENCES

[1] "World Class Manufacturing: The lessons of Simplicity applied.", R.Schonberger, 1986.

[2] "Computer Integrated Manufacture: Competitive Advantage or Technological Dead End?", A.T.Kearkey, 1989.

[3] "The Introduction of Group Technology.", J.L.Burbidge, London: Heinemann, 1975.

[4] "Group Technology in the Engineering Industry.", J.L.Burbidge, Mechanical Engineering Publication, 1979.

[5] "Production Control: The future choice.", 5th National Conference of Production Research, J.L.Burbidge, 1989.

[6] Private Communication, M.Mitchell, Siemens AG.

[7] "The Theory of Constraints Journal, volumes 1-6", .Goldratt and R.E.Fox, 1988-1989.

[8] "The Goal: A process of ongoing improvement.",E.M.Goldratt R.E.Fox, 1986.

[9] "MRP: A review of failure and a proposal of recovery using CBS.", BPICS Control, G.Archer, December 1990/January 1991.

[10] "Barriers to employee involvement in technological change. More than a case of the good guys and the bad guys.", Adv. Manuf. Eng., B.Burnes, April 1990.

[11] "Managing change in the 90's - People in Business.", May 1990.

[12] "Implementing CIM: Computer Integrated Manufacturing.", A.Kochan and D.Cowan, IFS 1986.

[13] "Software Supporting Introductory Courses in Production Operations Management.", Jnl. of Prod. and Inv. Man. 30(1), 1989.

[14] "Designing Games and Simulations - An Illustrated Handbook.", C.S.Greenblat, Sage, 1988.

[15] "Try your Production Ability - go in for TEAMSKILL!", Metalworking Production, July 1978.