

56520

AN OBJECT ORIENTED APPROACH TO DESIGN OF A MODULAR SHOP
FLOOR CONTROLLER

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

HAKKI ÖZGÜR ÜNVER

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

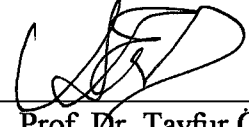
IN

THE DEPARTMENT OF MECHANICAL ENGINEERING


YÜKSEKÖĞRETİM KURULU
DENEYİM MERKEZİ

SEPTEMBER 1996


Approval of the Graduate School of Natural and Applied Sciences.


Prof. Dr. Tayfur ÖZTÜRK
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.


Prof. Dr. Ediz PAVKOÇ
Head of the Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.


Prof. Dr. Ömer ANLAĞAN
Supervisor

Examining Committee Members

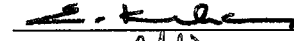
Prof. Dr. Metin AKKÖK (Chairman)



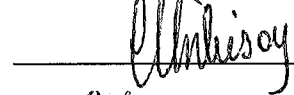
Prof. Dr. Ömer ANLAĞAN (Supervisor)



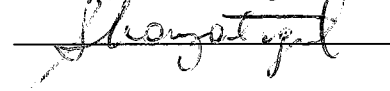
Prof. Dr. Engin KILIÇ



Prof. Dr. Samim ÜNLÜSOY



Doç. Dr. Sinan KAYALIGİL



ABSTRACT

*AN OBJECT ORIENTED APPROACH TO DESIGN OF A MODULAR SHOP
FLOOR CONTROLLER*

Ünver, Hakkı Özgür

M.S., Department of Mechanical Engineering

Supervisor : Prof. Dr. Ömer Anlağan

September 1996, 104 pages

The present study aims to develop a group of softwares to control a shop floor system using an object oriented programming approach. The shop floor system mentioned is a pilot Flexible Manufacturing System built in Mechanical Engineering Department of Middle East Technical University not only to increase the quality of education in this area but also to provide Research and Development facilities to Small and Medium Sized Enterprises (SMEs) in Industry.

A hierarchical approach is implemented for the control strategy of the system. The system contains two flexible manufacturing cells, each containing a

CNC machine tool, a robot for material handling, a part buffer and a cell control computer which controls the device functions. Above one layer host control computer controls the cell activities and transportation between the cells which is performed by a closed loop conveyor.

User can prepare a batch connecting to the common database of the shop floor which is in the host computer and schedule it for the system with the built-in scheduler of the host controller software. After downloading the necessary G-Code and order files to cell computers via LAN, the automatic production cycle can be started.

During the operation of the cycle user can monitor the status and location of each part in the batch and status of the devices on the computers. After the cycle is completed statistical information can be obtained like total operation time, idle time, machining time and cell utilization for each cell.

Keywords: Shop Floor Control, Object Oriented Approach, Flexible Manufacturing Systems.

ÖZ

***MODULER BİR İMALAT SİSTEMİ DENETLEYİCİSİ TASARIMI İÇİN
NESNEYE YÖNELİK YAKLAŞIM***

Ünver, Hakkı Özgür

Yüksek Lisans, Makina Mühendisliği Bölümü

Tez Yöneticisi : Prof. Dr. Ömer Anlağan

Eylül 1996, 104 sayfa

Bu çalışmada, nesneye yönelik programlama yaklaşımı kullanılarak, bir imalat sisteminin denetlenmesi için bir grup yazılım geliştirilmesi amaçlanmıştır. Söz konusu imalat sistemi, Orta Doğu Teknik Üniversitesi, Makina Mühendisliği Bölümünde, sadece eğitim amaçlı değil, aynı zamanda sanayideki Küçük ve Orta Ölçekli İşletmelere araştırma ve geliştirme olanakları sağlamak için kurulan pilot bir Esnek İmalat Sistemidir.

Sistemin denetleme stratejisi için hiyerarşik bir yaklaşım uygulanmıştır. Sistem herbiri bir CNC takım tezgahı, bir taşıma robotu, bir parça deposu ve

denetleme için bir bilgisayar içeren iki esnek imalat hücresinden oluşmaktadır. Bir seviye yukarıda, ana denetleme bilgisayarı hücre olaylarını ve bir kapalı çevrim konveyör tarafından yapılan hücreler arası parça taşıma işlemini denetler.

Kullanıcı ana bilgisayarda bulunan, imalat sisteminin ortak veri tabanına bağlanarak bir parça kümesi hazırlayabilir ve ana denetleyici yazılımı içinde bulunan işlem zamanlayıcısı algoritmasını kullanarak sistem için işlem zamanlaması yapabilir. Gerekli G-Kodu ve emir dosyalarını hücre bilgisayarlarına yerel ağ ile aktardıktan sonra otomatik üretim çevrimi başlatılabilir..

İşlem sırasında kullanıcı her parçanın durum ve konumunu ve de makinaların durumunu bilgisayarlardan izleyebilir. Çevrim bittikten sonra her hücre için toplam operasyon zamanı, boş zaman, işleme zamanı ve hücre verimi gibi istatistiki bilgiler alınabilir.

Anahtar Kelimeler: İmalat Sistemi Denetlenmesi, Nesneye Yönelik yaklaşım ,Esnek İmalat Sistemleri.

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my supervisor, Prof. Dr. Ömer Anlağan, for his guidance, supervision, and support throughout the study.

I am also indebted to Prof. Dr. S. Engin Kılıç for his valuable comments.

Very special thanks go to resque squad, Mr. Can Saygın and Mr. Ümit Coşkun, for their first-aid with all their means, whenever I am in trouble with the thesis work.

I would like to thank to my friends, Okan Bilkay, S. Emrah Halıcı, Gülşah Halıcı, Yavuz İlik, Derya Karakan, S. Savaş Kırımlioğlu, Çağlar Kıral, Metin Koyuncu, Cüneyt Özbal, Oytun Öztürk, Kerem Pekkan, Murat Şavkılıoğlu, Taner Yöney, Can Ali Yücesoy for lighting a candle when it's dark.

Also, I am grateful to Mr. Mehmet Çakmak and machine shop team for their efforts and co-operation in shop work.

Finally, words are inadequate sometimes but, greatest thanks go to my parents who shaped me with never-ending patience, love, and sacrifice.



to my parents

TABLE OF CONTENTS

ABSTRACT	iii
ÖZ	v
ACKNOWLEDGEMENTS	vii
TABLE OF CONTENTS	viii
LIST OF FIGURES	xii
LIST OF TABLES	xvi
CHAPTER	
1. INTRODUCTION	1
1.1 What is CIM?	1
1.2 NBS and ISO Models For Manufacturing	3
1.3 A Pilot CIM Implementation: METUCIM	4
1.4. Scope of the Study	7
1.5 Outline of the Thesis	8
2. LITERATURE SURVEY	10
2.1 Flexible Manufacturing.....	10
2.1.1 Flexible Manufacturing Cell.....	10
2.1.2 Flexible Manufacturing Systems	12
2.2 Shop Floor Control Architectures.....	16
2.2.1 Hierarchical Architecture	17

2.2.2 Heterarchical Architecture	17
2.2.3 Modularity of a Control Architecture ...	17
2.3 Object Oriented Programming	18
2.4 Networking Technologies in Manufacturing	19
2.4.1 ISO/OSI Reference Model	19
2.4.2 TCP/IP	21
2.4.3 MAP/TOP	21
2.4.4 EIA RS-232 C.....	22
2.5 Data Management Systems	23
2.6 Previous Work	24
3. THE SHOP FLOOR EQUIPMENT OF METUCIM..	29
3.1 Original Equipment of The System	29
3.2 Modifications Done on The System	35
4. THE SOFTWARE PACKAGES DEVELOPED.....	37
4.1 Introduction	37
4.2 C++ and Windows Environment	37
4.3 Host Controller	38
4.3.1 Ethernet Driver Object (Server).....	41
4.3.2 Scheduler Object	43
4.3.3 Dispatcher Object	46
4.3.4 Monitor Object	46
4.3.5 Cell Manager Object	47
4.3.6 Conveyor Manager Object	47

4.3.7 Common Database	48
4.4 Cell Controller	49
4.4.1 Ethernet Driver Object (Client)	52
4.4.2 Cell Squencer Object	52
4.4.3 Dispatcher Object	53
4.4.4 Monitor Object	54
4.4.5 CNC Driver Object	54
4.4.6 ROC Driver Object	55
5. TEST RUNS.....	56
5.1. First Test Run	56
5.2 Second Test Run	61
6. CONCLUSION AND FUTURE WORK.....	63
REFERENCES	67
APPENDICES	
A. USER'S MANUAL	71
A.1 Installation	71
A.2 FMS Boot up Sequence	72
A.3 Host Controller	73
A.3.1 Host Controller Main Window	73
A.3.2 Menu Bar	75
A.3.2.1 File/Exit	75
A.3.2.2 Scheduler/File Database	75
A.3.2.3 Scheduler/MS-Access Database	77

A.3.2.4 Dispatcher/Open schedule	78
A.3.2.5 Dispatcher/Close schedule	80
A.3.2.6 Dispatcher/Download	80
A.3.2.7 Dispatcher/Execute	80
A.3.2.8 Connections/Open	80
A.3.2.9 Connections/Close	81
A.3.2.10 Options/Directories	81
A.3.2.11 Statistics/Op-times	82
A.3.2.12 About	83
A.4 Cell Controller	83
A.4.1 Cell Controllers Main Window	83
A.4.2 Menu Bar	84
A.4.2.1 System/Enable Cell Mode	84
A.4.2.2 System/Disable Cell Mode	84
A.4.2.3 System/Enable FMS Mode	85
A.4.2.4 System/Disable FMS Mode	85
A.4.2.5 System/Exit	85
A.4.2.6 DNC Functions/Manual Control	85
A.4.2.7 About	86
B. G-CODE FILES OF TEST RUNS	
B.1 First Test Run	87
B.2 Second Test Run	92
C. SAMPLE CLASSES	97

D. TEST RUN OF BUILT-IN SCHEDULER 101



LIST OF FIGURES

FIGURES

1.1 CIM Technology.....	2
1.2 Hierarchy Levels of METUCIM	5
2.1 Structural Forms of FMS	15
2.2 Layers of OSI Architecture	20
2.3 The Structure of the NBS/NIST Cell Controller	24
2.4 Two Level Cell Controller	26
2.5 The Structure of X-Cell System	27
3.1 General View of the System	30
3.2 View of the First Cell Controller	32
3.3 View of the Closed Loop Conveyor	33
3.4 Layout of FMS	35
4.1 Predictive Schedule in Form of Gantt Chart	39
4.2 Reactive Schedule Generated After a Completed Cycle...	40
4.3 Host Controller Structure	42
4.4 Routes of the System	45
4.5 Main View of the Cell Controller	50
4.6 Cell Controller Structure	51

5.1 Drawing of Part-1	56
5.2 Drawing of Part-2	57
5.3 Drawing of Part-3	57
5.4 Schedule File Generated by the Built-in Scheduler	58
5.5 Order file of Part-1	58
5.6 Order file of Part-2	59
5.7 Order file of Part-3	59
5.8 Statistics Data of First Test Run	60
5.9 Statistics Data of Second Test Run	62
A.1 Host Controller Main Window	74
A.2 Order Entry Dialog Box	76
A.3 File Save Dialog Box	77
A.4 Select Database Dialog Box	78
A.5 File Open Dialog Box	79
A.6 A Schedule Displayed in Form of a Gantt Chart.....	79
A.7 Directories Dialog Box	81
A.8 Statistics Dialog Box	82
A.9 Cell Controller Main Window	84
A.10 Manual Control Dialog Box	86
B.1 G-Code file for Lathe Process of Part-1	87
B.2 G-Code file for Lathe Process of Part-2	88
B.3 G-Code file for Lathe Process of Part-3	89
B.4 G-Code file for Milling Process of Part-1	90

B.5 G-Code file for Milling Process of Part-2	91
B.6 Modified G-Code File for Lathe Process of Part-1	92
B.7 Modified G-Code File for Lathe Process of Part-2	93
B.8 Modified G-Code File for Lathe Process of Part-3	94
B.9 Modified G-Code File for Mill Process of Part-1	95
B.10 Modified G-Code File for Mill Process of Part-2	96
C.1 Header File of Dispatcher Class of Host Controller	97
C.2 Header File of Conveyor Manager Class of Host Controller	98
C.3 Header File of CNC Driver Class of Cell Controller	99
C.4 Header File of Dispatcher Class of Cell Controller	100
D.1 Generated Schedule Order	104

LIST OF TABLES

TABLES

4.1 Fields of Common Database	49
4.2 List of Commands and Verification Codes for Cell Devices	54
5.1 Machining Times of First Test Run	59
5.2 Machining Times of Second Test Run	61
D.1 Test Batch for Built-in Scheduler	101
D.2 Results of Step-2 for Type M12	102
D.3 Results of Step-2 for Type M21	102
D.4 Results of Step-3 for Type M12	102
D.5 Results of Step-4 for Type M21	103

CHAPTER 1

INTRODUCTION

1.1 What is CIM ?

Industry is a generic description covering several activities which add value to a product or provide a service. Industry can be categorized as service (e.g. transport), process (e.g. chemical), and manufacturing (e.g. cars). Manufacturing industry produces goods in finite amount (discrete production). The process industries have long been automated and computerized. The manufacturing industries have been automated to a certain degree but mainly in the mass production environments. The small batch production environments have been automated since the late 1970s, with the introduction of the CNC, but have not had these islands of automation integrated into the total factory environment [1].

Until very recently integration of information and data, transfer were always manual, in other words using paper as the transfer medium. With the advent of computers and electronic data communications, it has been possible to integrate manufacturing activities. The integration of the islands of activities in an enterprise is the main goal of CIM. Although there is not one definition of CIM that is universally accepted, the emerging consensus is that Computer-Integrated Manufacturing is a management philosophy in which the functions of design and manufacturing are

rationalized and coordinated using computer, communication, and information technologies. According to Mize and Palmer, "rationalized" in this context means:

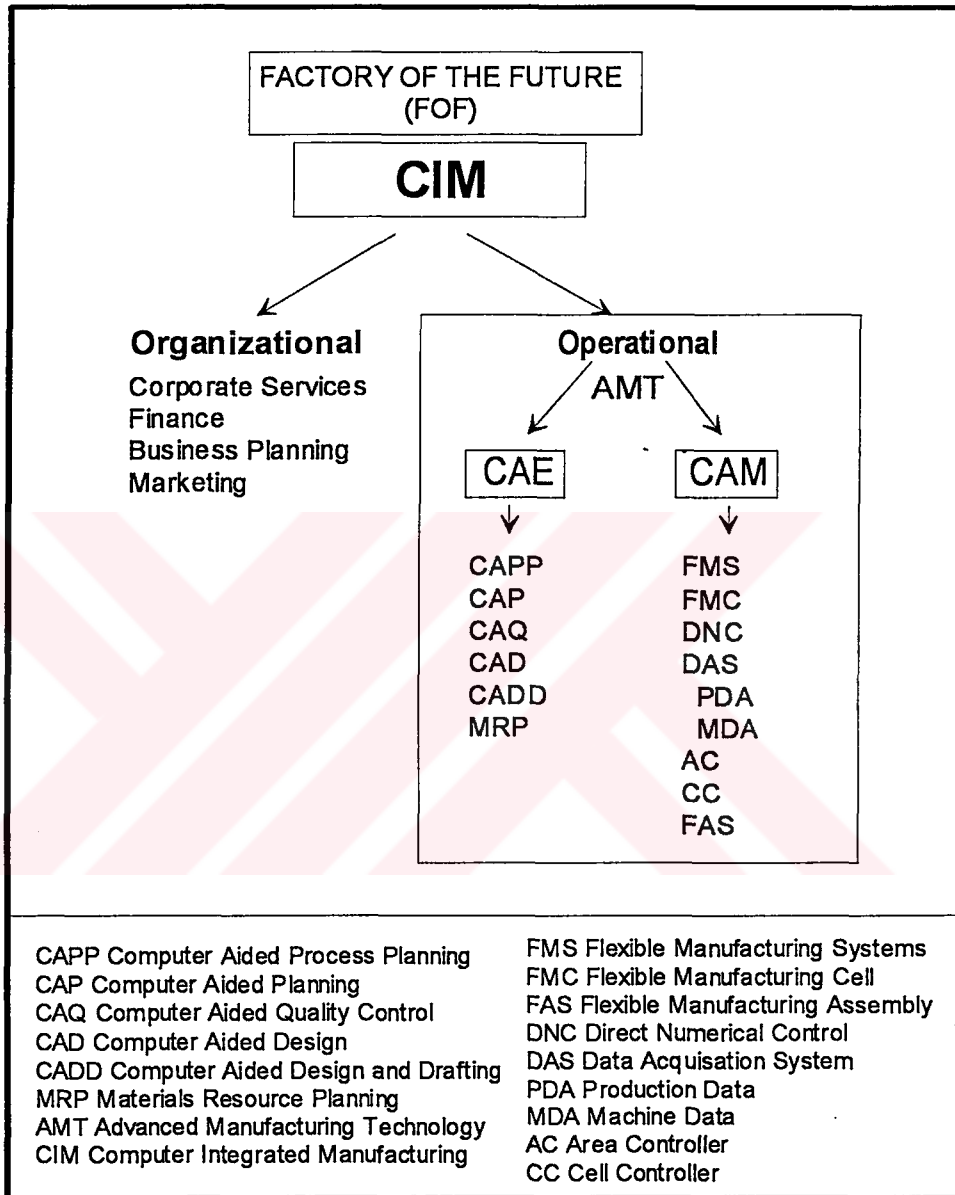


Figure 1.1 CIM Technology

The entire manufacturing system, from product definition and raw material acquisition to the disposition of the final product, is carefully analyzed such that every operation and element can be designed to contribute in the most efficient and effective way to the achievement of clearly enunciated goals of the enterprise [2]. Figure 1.1 shows the manufacturing activities in a CIM system.

1.2 NBS and ISO Models For Manufacturing [3]

The National Bureau of Standards (NBS) in the United States of America has been established a hierarchical control model for automated systems. It recognizes five hierarchical models namely; facility, shop, cell, workstation and equipment. The facility is the highest level having three sub-systems; manufacturing engineering, information management, and production management. The shop is responsible for the real time management of jobs and resources on the shop floor. The cell level manages the sequencing of batches and materials handling facilities. The workstation directs and coordinates a set of equipment on the shop floor. A workstation consists of a set of equipment set up to realize a particular task. The equipment controllers are linked directly to individual pieces of equipment within a workstation. Within this model, tasks at the highest level are decomposed into sequences of sub-tasks which are passed down to next lower level of the hierarchy.

The International Standards Organization (ISO) has developed a Factory Automation Model composed of a six-level hierarchical structure. The levels are: enterprise, facility/plant, section/area, cell, station, and equipment. The enterprise level is responsible for the achievement of the mission of the enterprise and its planning horizon is measured in years and months. The facility/plant level is

responsible for the implementation of the enterprise functions and the reporting of the status information to the enterprise level. It includes manufacturing engineering, information management, production management, scheduling and production engineering. The section/area level is responsible for the provision and allocation of resources and the coordination of production on shop floor. Typically this level operates within a horizon of several days or weeks. The cell level is responsible for the sequencing of jobs through the various stations. Its functions include resource analysis and assignment, making decisions on job routings, dispatching jobs to individual stations and the monitoring of task and station status. The station level is responsible for the direction and coordination of relatively small integrated workstations. It operates in virtual real time with planning horizons of milliseconds to hours. In a sense a station is a virtual machine. The equipment level realizes the physical execution of tasks on machines and it has a planning horizon ranging from several milliseconds to several minutes.

1.3 A Pilot CIM Implementation: METUCIM

An attempt to establish a pilot CIM system in Mechanical Engineering Department of Middle East Technical University (METU), named as METUCIM, has been initiated not only to increase the quality of education in this area but also to provide Research and Development facilities for Small and Medium-size Enterprises (SMEs) in Turkish Industry.

The shop floor equipment in the pilot CIM system of Middle East Technical University, is a Flexible Manufacturing System basically having two manufacturing cells and a closed loop conveyor as the main material handling

device. One of the cells includes a CNC lathe, the other has a CNC milling machine and a Coordinate Measuring Machine. Both cells include a robot for inner-cell material handling and an I/O buffer. At the cell level of the control hierarchy, cell devices are controlled by the cell computer. Tasks of the cell controllers and the closed loop conveyor are controlled by the host computer at the shop floor management level.

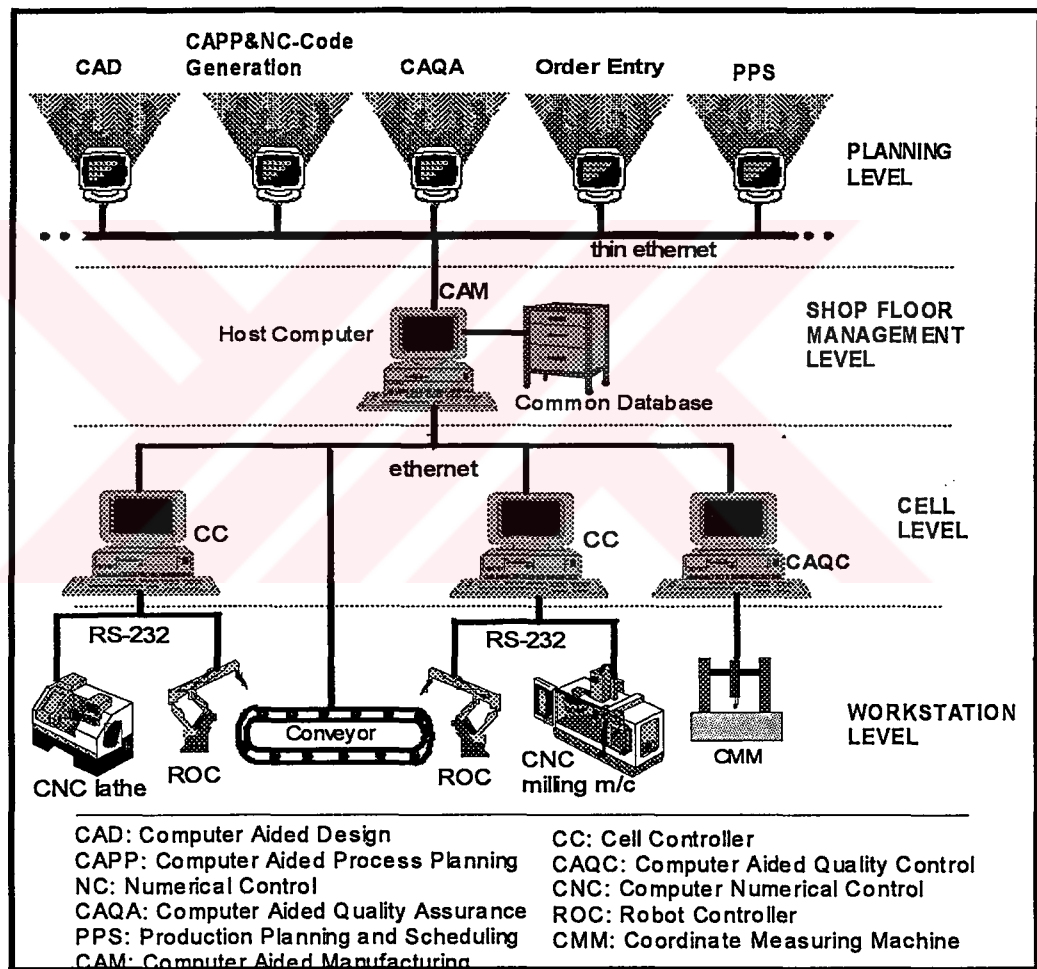


Figure 1.2 Hierarchy Levels of METUCIM

Saygin *et al.* proposed the design stage of METUCIM based on the following principles [16]:

1. **Standardization Principle:** Whenever possible, use standard (or standardized) module in both software and hardware (like network, interfaces, transfer formats, etc.).
2. **Capability Principle:** Define and understand the limitations and capabilities of the existing manufacturing system on which the CIM concept will be built.
3. **Flexibility Principle:** Whenever possible, postpone the decision making to the lowest control level (late commitment) and never specify unnecessary constraints on the production resources unless needed. This principles leads to "alternative" production resources (machine tool flexibility, routing flexibility, cutting tool flexibility, etc.)

The capabilities of the manufacturing system include turning, milling, automated material handling and inspection. The functional areas covered by the CIM system are Computer Aided Design (CAD), Computer Aided Process Planning (CAPP) and NC-Code Generation, Order Entry, Production Planning and Scheduling (PPS), Computer Aided Manufacturing (CAM) and Computer Aided Quality (CAQ) Assurance and Control.

The management of the computer aided "islands" of activities is achieved by a hierarchical control structure, characterized by four levels as shown in Figure 2.

Top level corresponds to planning level where CAD, CAPP, NC-Code Generation, CAQA, Order Entry and PPS are realized. At the Shop Floor Management Level, CAM activities are carried out by host computer. At the cell level, controlling of cell devices take place by the cell computers under the supervision of the host computer. The bottom level is the actual manufacturing equipment at the workstation level, including CNC machine tools, robots and conveyor.

1.4 Scope of the Study

The scope of the study is to develop a computer software family to achieve the real time control of the Shop Floor Manufacturing Equipment. Developed programs are the cell controllers running on cell computers at cell level of METUCIM and shop controller running on host computer at shop floor management level.

Cell controllers receive the orders from the host controller and carry out the relevant discrete functions of the manufacturing resources within the cell. The connections between the cell controllers and cell devices are established using RS-232 C interface and 16 bit in, 16 bit out Industrial I/O boards.

The host controller receives the predictive schedule from the planning level and carries out the operational functions of the cell controllers and the conveyor for the execution of the schedule. It can also access to the common database of the shop floor level and generate a predictive schedule using built-in schedule algorithm for a two-machine problem. The data communication link between the cell computers and the host computer is established using TCP/IP protocol on ethernet hardware.

The developed methodology is based on an object-oriented modeling approach where particular tasks of the controllers are assigned to classes. C++ programming language is used and the developed software family runs under Windows environment. The effectivity of the real-time management is achieved by message system between objects offered by the Windows environment. These are used as links for triggering necessary procedures of classes to respond events occurring in the physical system.

1.5 Outline of the Thesis

In chapter II, related literature survey will be presented. Flexible Manufacturing Systems, Shop Floor Control Approaches, Networking Technologies, database management in manufacturing, and Object-Oriented Approach to Shop Floor Control will be reviewed.

In chapter III, developed softwares will be explained in detail. The structure of the host controller and the cell controller will be figured out. The class hierarchy of the softwares and object-oriented methodology used for the real-time control of the system will be emphasized.

In chapter IV, results of the test runs executed on the system will be presented.

In chapter V, results of this study will be summarized and recommendations for further works on this subject will be given.

In appendix A, user's manual will be given for program operators.

In appendix B, NC-Codes of the parts that are used in the test runs are given.

In appendix C, some example classes in the program source codes will be given.

In appendix D, a test run and its results for the built-in schedule algorithm will be presented.



CHAPTER 2

LITERATURE SURVEY

2.1 Flexible Manufacturing

Flexibility in manufacturing can be defined as the ability to adjust the primary process according to new requirements [4]. To be truly flexible, the flexibility must exist during the entire life cycle of a product, from design to manufacturing to distribution. The degree and level of flexibility of a system is limited by technological boundaries. Therefore, the flexible manufacturing systems are not the solution to all manufacturing problems and the user's expectation from an FMS must be well established around the system's capability [5].

2.1.1 Flexible Manufacturing Cell

The manufacturing cell concept was introduced by the Production Engineering Laboratory at the University of Trondheim in Norway [6]. A manufacturing cell has been defined in many ways. Martin presented several definitions for manufacturing cells [7].

1. A manufacturing cell is a group of people and processes located in a specific area dedicated to the production of a family of parts or products.

2. Grouping of dissimilar types of machinery needed to manufacture a complete family of parts.
3. One or more machine tools linked by common material handling devices under the control of a centralized cell controller for the purpose of producing the given requirements of a family of parts.

A more comprehensive definition of a manufacturing cell is given by Franks et.al. as follows [8]:

4. A group of manufacturing resources, consisting of machines and workstations which are organized and scheduled as an entity to accept discrete parts, sub-assemblies and materials, the cell then adds value through processing to create a new identifiable product as its output. Cells may be automated, semi-automated, manually operated or a combination of these types.

A FMC contains a large number of equipment. O' Grady suggests that a cell is usually made up of [9]:

1. Computer processor containing a cell control system.
2. NC machines or other processing equipment
3. Robots or other material handling devices
4. Storage elements

Highlighting these definitions, a lean definition for a FMC can be given as:

A flexible manufacturing cell is a group of manufacturing resources, linked by a material handling system. This group is controlled by a central computer to produce a family or several families of parts in random order.

2.1.2 Flexible Manufacturing Systems

Flexible Manufacturing Systems have a number of potential definitions. The literal meaning is a logical arrangement (system) of transformation processes (manufacturing) that is adjustable to change (flexible) [10]. A number of definitions can be given as follows:

1. A group of CNC machine tools linked by an automated materials handling system, whose operation is integrated by supervisory computer control. Integral to an FMS is the capability to handle any number of similar families of part in random order [11].
2. Flexible Manufacturing Systems are highly automated production systems, able to produce a great variety of different parts by using the same equipment and the same control systems [12].
3. A Flexible Manufacturing System is a manufacturing system in which groups of numerically controlled machines (machine centers) and a material handling system work together under computer control [13].

An FMS may take several structural forms. B. L. MacCarthy and J. Liu proposed 5 different structural forms of an FMS as the structures shown in Figure 2.1 [14].

1. A flexible manufacturing system (FMS) is a production system capable of producing a variety of part types, which consists of CNC or NC machine tools connected by an automated material handling system. The operation of the whole system is under computer control.
2. A single flexible machine (SFM) is a computer controlled production unit which consists of a single CNC or NC machine with tool changing capability, a material handling device and a part storage buffer.
3. A flexible manufacturing cell (FMC) is a type of FMS consisting of a group of SFMs sharing one common material handling device.
4. A multi-machine flexible manufacturing system (MMFMS) is a type of FMS which consists of a number of SFMs connected by an automated material handling system which includes two or more material handling devices or is otherwise capable of visiting and serving two or more machines at a time.
5. A multi-cell flexible manufacturing system (MCFMS) is a type of FMS which consists of a number of FMCs and possibly a number of SFMs if necessary, all connected by an automated material handling system.

Though it is very hard to distinguish a FMC from a FMS using the

definitions in the literature, the primary differences can be given as follows [15]:

1. Cells lack central computer control with real time routing, load-balancing and scheduling logic. Cell controller receives the orders from the host computer located at a higher level.
2. Cells are typically tool capacity constrained. This limits the part spectrum that could be run through a cell at a given time without stopping the equipment and manually exchanging the tools to accommodate different workpieces. On the other hand, an FMS with automated tool delivery and tool management system can automatically adjust the tool requirements within the system.
3. Cells have generally have less flexibility than an FMS as they are restricted to a relatively tight family of parts.

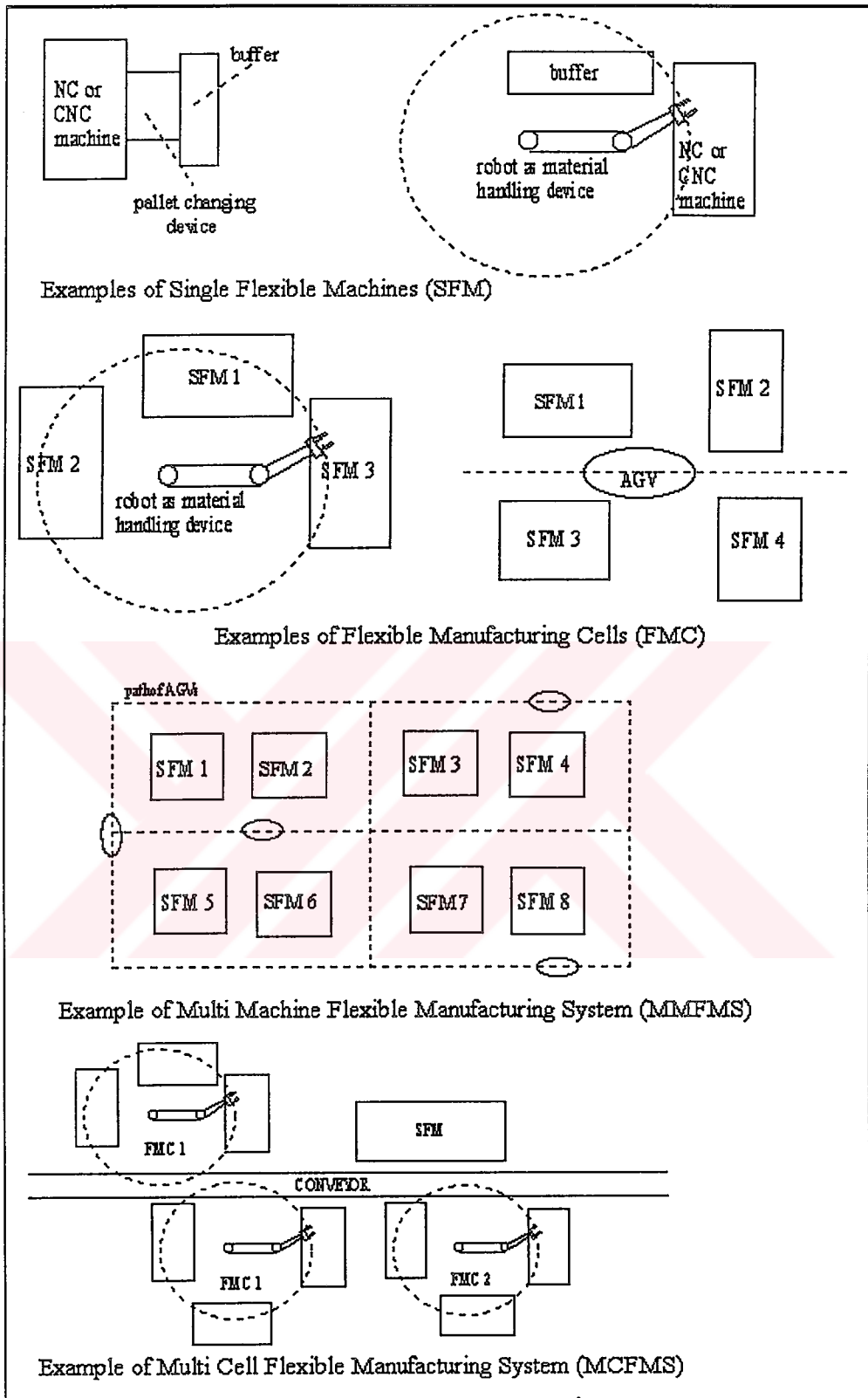


Figure 2.1 Structural Forms of FMS

Regarding these primary differences, it can be concluded that the distinction between an FMS and an FMC occurs due to the degree of control employed. In the broadest sense, a Flexible Manufacturing System might be composed of several manufacturing cells. Host controller views the cells as a black box for shop floor control purposes. The work orders are scheduled according to the capabilities of the cells and distributed among them. On the other hand, each cell controller considers a chunk of more detailed low-level and cell-specific data to further sequence the sub-tasks among the entities (such as machine tool, robot, buffer, etc.) within the cell [16].

2.2 Shop Floor Control Architectures

During the manufacturing process, many random events may occur. Most of these random events are expected but the timings are unknown so production control relies on a good shop-floor control system to link higher level planning, order release, detailed scheduling and shop floor data collection.

An architecture allows one to present a complex system in a simple structure. An architecture is an abstract description of a system; it specifies subsystems of a certain system. It supports the analysis of relations as an aid to understand complexities in a design environment. In a shop-floor control environment, the control architecture determines how control functions are distributed and coordinated [4].

2.2.1 Hierarchical Architecture

A hierarchical architecture is characterized by the usage of control levels and contains several control modules arranged in a pyramidal structure. The distinct levels have all their own tasks to do and own functions to use. They have no way but to obey one level up and rule one level down. Thus, the control decisions are operated top-down with status reporting bottom up as a master/slave relation. There is a complete data abstraction between the levels so that only necessary information flow occurs between subsequent levels [4].

2.2.2 Heterarchical Architecture

Heterarchical control structures have distributed autonomous entities that communicate with other entities without the master/slave relation. The main concept in this architecture is the goal of full local autonomy and a co-operative approach to global decision making. Supervisory decision making is located locally at the point of information gathering rather than in a central location. Full local autonomy requires that controllers are intelligent and that global information is minimized or eliminated. This implies that only local databases will be maintained within the system [4].

2.2.3 Modularity of a Control Architecture

Modular design applies the concept of module as the basic thread for building the system architecture. The steps for the system design are the following [4].

1. Define the functions of the system

2. Assign each function to a module
3. Specify in detail the interfaces between the modules
4. Design the non-interface part for each module separately.

The modules in a system architecture should have the following properties [4]:

1. Modules have a clear and recognizable function
2. Modules have explicitly defined interfaces, with clear distinction between input and output.
3. Modules can be designed independently
4. A module can be tested and validated as a stand alone unit
5. A module can be integrated with other modules without further testing, as long as the interfaces match.
6. A module can be operated independently: without knowledge of specification and state of environment except for input-interface
7. A module can be exchanged when interfaces remain unchanged.

Properties 1,2 express that modules are basic building blocks in an architecture: function and interface. Properties 3-6 are referred as "module independence"; they express that the interfaces should effectively decouple the modules. Property 7, is the golden rule to the flexibility of a control system.

2.3 Object Oriented Programming

Object oriented programming is the organization of software as a collection of discrete objects that incorporate both data structure and behavior. Programs written in object-oriented languages involve the definition of objects which

store both the data and procedures required for the processing to be performed, computation proceeding via passing of messages between objects [17]. A message is simply a call request to a procedure. Such a request need not be concerned with the details of how the data is represented within the object providing the service nor with the details of how the manipulation is carried out.

The essence of O-OP is provided by the concepts of encapsulation and inheritance, encouraging respectively, software modularity and the reuse of existing code modules. Encapsulation, the grouping and hiding of data from any other objects derived from either the same base class or other classes, significantly reduces the software complexity and increases modularity. Further it allows more rapid system development, comprehension, maintenance and modification. Inheritance promotes reuse of class modules by defining new classes inheriting any features needed in the base class or modifying the ones that are useless for the derived class's task.

2.4 Networking Technologies in Manufacturing

2.4.1 ISO/OSI Reference Model

International Standards Organization (ISO) introduced the Open System Interconnection (OSI) Reference Model, a layered network architecture, with the goal of international standardization of computer network protocols. The OSI Model is said to be open systems architecture because it connects computer systems that are open for communications without any manufacturer and operating system similarity.

Communicating between dissimilar systems is not a simple process.

Specifications must address this complexity and accommodate changes as technology evolves. This is one reason why the architects decided to segment the specifications into groups called layers. The 7 layers proposed by ISO are shown in Figure 2.2.

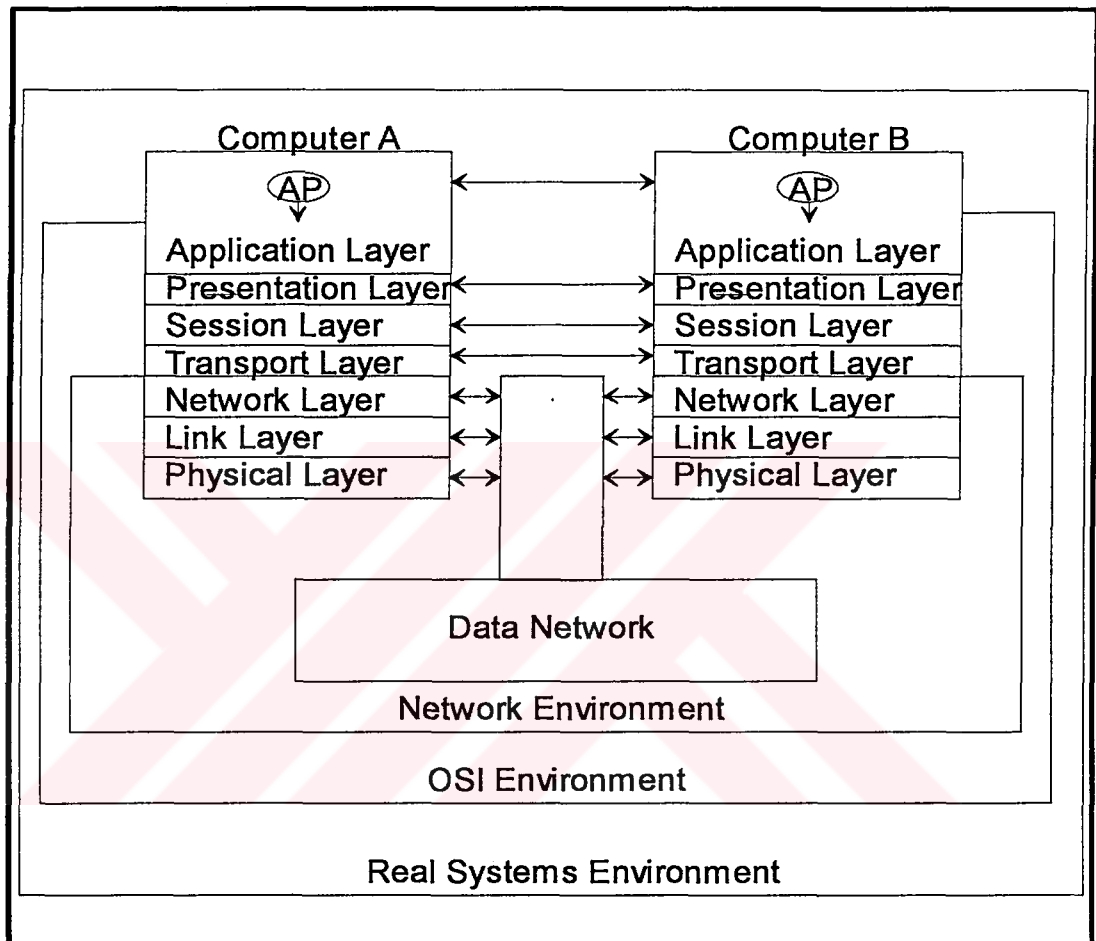


Figure 2.2 Layers of OSI Architecture

Layer 1 specifies the physical hardware, layer 2 is implemented by combination of hardware and software and the rest 5 layers are implemented entirely by software [23].

The IEEE standards that specify the two lowest layers are [19]:

1. IEEE 802.2, data or logical link control
2. IEEE 802.3, carrier-sense multiple access with collision detection (CSMA/CD) bus LANs, such as Ethernet
3. IEEE 802.4, token passing bus LANs, such as MAP
4. IEEE 802.5, token passing ring LANs, such as IBM's token-ring network

2.4.2 TCP/IP

TCP/IP does not directly follow the OSI model. TCP/IP provides a means for addressing intermediate protocol levels and in fact is often combined with ethernet in a communications approach that defines both the lower and middle aspects of the system. TCP/IP functions by dividing any message provided to these middle layers into packets, and then sending packets of 64K bytes at a time through the communication network. It must also reassemble the packets into correct order at the receiving user [20,21].

2.4.3 MAP/TOP

MAP was developed specifically for use in a factory environment. General Motors (GM) Corporation has been the leading advocate of this particular protocol. The objective was to establish one set of LAN protocols for communications between intelligent devices, such as computer-controlled machine tools, engineering workstations, process controllers, factory-floor terminals and control rooms. MAP is a broadband, token-bus network protocol based on the OSI

model that will accommodate a broad range of manufacturing environments. Layers 1 and 2 obey IEEE 802.4 specification. The MAP User' s Committee has defined the top five layers.

Boeing proposed the TOP (Technical and Office Protocol) as a supplement to MAP. TOP and MAP networks are compatible. A core set of protocols, layers 3,4,5,6, and parts of 7, are common to both. At layer 7 they diverge; TOP specifies application protocols that address requirements of the office instead of factory-floor requirements. They also diverge at the lowest two levels, where TOP specifies the less expensive, less deterministic 802.3 baseband-media and media access technique in place of the 802.4 broadband that MAP uses [22].

2.4.4 EIA RS-232 C

RS-232 C standard was published in 1969 by the Electronic Industries Association. Earlier versions RS-232 A and B were first originated in the late 1950s. It has been developed for interface between data terminal equipment (DTE) and data communications equipment (DCE) employing serial binary data interchange. The standard describes the interface between a terminal (a DTE) to a modem (a DCE) for the transfer of serial data [23,24].

In manufacturing, it is a widely accepted standard for serial connection of Computer Numerically Controlled machines by most of the manufacturers. The drawback of RS-232 standard is the limited length of cables at high baud rates. The generally accepted maximum length is approximately 50 feet at 9600 baud. Moreover though limited network applications can be implemented using RS-232, it is not designed to be an open system, so it is best suitable for DNC control of CNC

machines in manufacturing applications.

2.5 Data Management Systems

The objective of a database management system is to give access to information in a timely and concise manner and to effectively manage information as a corporate asset.

There are three major database management systems:

1. Hierarchical
2. Network
3. Relational

The relational database management systems are the most suitable technology for implementing the data management function of shop floor control systems. This is mainly due to inherent flexibility and their potential for distributiness.

Data within relational databases may be accessed through Structured Query Language (SQL) which is a data definition and data manipulation language for relational databases. Using SQL a database may be created, populated with data, and its data modified. The SQL language can be used interactively by the user to query the database or alternatively it can be used directly from most common programming languages [3].

2.6 Previous Work

One of the first cell controllers was developed by National Institute of Standards (NIST) and it is a well-defined modular cell controller. It consists of four parts, namely, operator command system, queue configuration manager, schedulers and dispatchers. The structure of the cell controller is as shown in Figure 2.3.

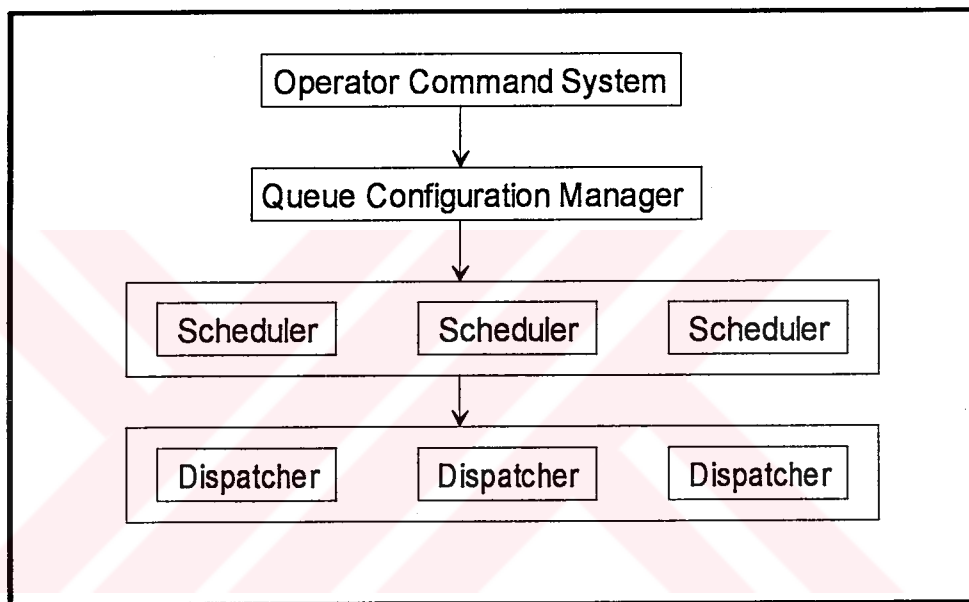


Figure 2.3 The Structure of The NIST Cell Controller

In this system, the queue configuration manager module interfaces to the scheduler modules below and the operator command system above. Whenever a new order must be processed, the queue configuration manager creates an entry in the cell job queue, determines the tasks necessary to meet that request and assigns those tasks to appropriate schedulers [25].

Another early stage cell controller is the MCCS designed by Politecnico di Milano of Italy. This system is proposed for a small (4-6 machine, with single material handling device) flexible manufacturing cell. Software modules in this system include: job scheduling modules, job control and accounting modules, and monitoring modules. The job scheduling modules are to receive job orders which are a medium-term production plan (3 to 6 months) from the factory computer, plan cell capacity, produce a short-term plan (1-2 weeks) and create shift by shift the scheduled job list for a cell work shift. The job control and accounting modules are to deal with the real-time control of cell operations. Monitoring modules are to monitor the operation of the cell.

A cell controller with two control levels is developed by the South Carolina Research Authority (SCRA) of USA. This cell controller architecture is shown in Figure 2.4. At the high level, there are scheduler , dispatcher, reporter, WIP tracker and a database which holds the manufacturing requirements file, build schedule, inventory file, routing file and production history file. At the low level, coordination control is mainly carried out. This level is managed by a programmable cell manager which has connection to an operator interface, display CAD, a real-time SPC, a distributed database and device interfaces [26].

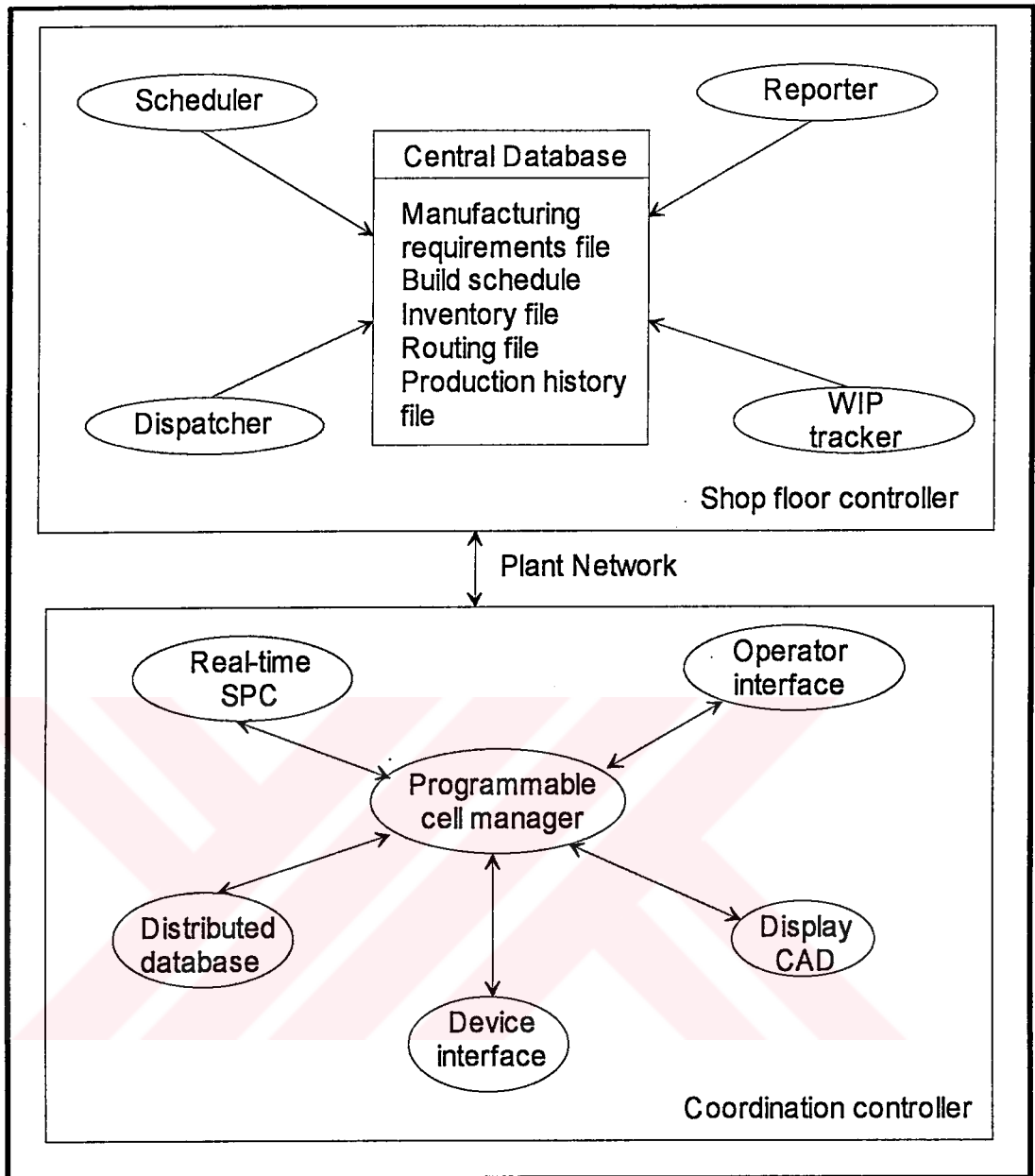


Figure 2.4 Two Level Cell Controller

An object-oriented programming approach is also being used in the design of a cell controller by O'Grady and his research group. This cell control system, called X-Cell, uses object oriented approach to break the cell control functions into objects. They consider that their approach is potentially powerful in

that it allows great flexibility in system architectures. The architecture of the cell control system is shown in Figure 2.5 [27].

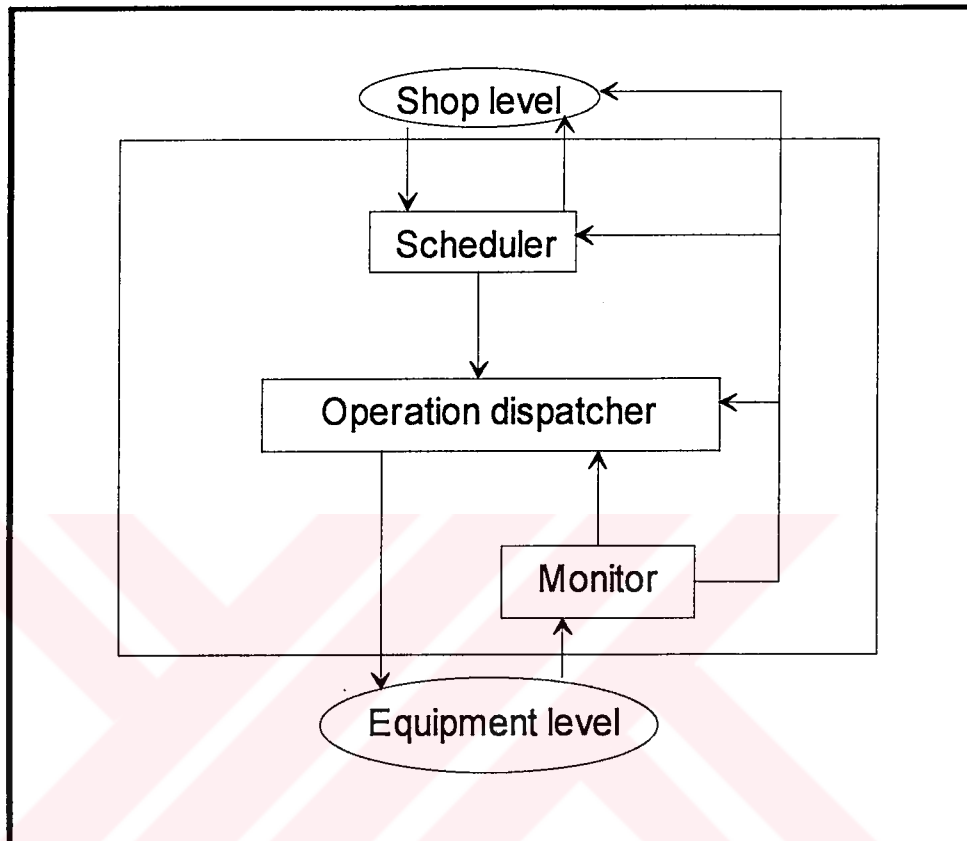


Figure 2.5 The Structure of X-Cell System

Another cell controller developed using the object-oriented programming approach has been developed by the Gintic Institute of Manufacturing Technology, Nanyang Technical University of Singapore. This cell controller is to plan and coordinate the production activities within a manufacturing shop floor. It governs the very-short term detailed planning, execution, and monitoring activities needed to control the flow of an order from the moment the order is released by the planning system for execution until the order is completed. They suggest that their

object-oriented approach can enhance the modularity and reusability of the cell controller [28].

In the CIM laboratory of the Cooperative Engineering Center (CEC) in Amsterdam, three different designs for shop floor control systems are implemented in a scale model factory for PCB assembly and test. Two different control architectures are specified and implemented: a hierarchical control architecture and a decentralized control architecture. The hierarchical architecture is characterized by a two-level scheduling and dispatching. At the highest level a work order is received and planned in a static schedule. The second level provides dispatching and detailed dynamic scheduling. Feedback is obtained by monitoring the status of the model factory. The decentralized model is characterized by autonomous controllers for various parts of the model factory. This results in a pull oriented control of the model factory. It has been implemented in two different implementation models. In the first implementation, information system modules have been defined for each autonomous controller. These modules are consequently implemented independent of each other. The second implementation is based on object oriented design. Generic objects are defined, with an abstract client object and an abstract server object at the top. Consequently, the design and implementation of the individual modules has been derived from these generic objects [29].

CHAPTER 3

THE SHOP FLOOR EQUIPMENT OF METUCIM

3.1 Original Equipment of The System

The shop floor manufacturing equipment of METUCIM is a pilot Flexible Manufacturing System which was delivered to Mechanical Engineering Department of Middle East Technical University in connection with the commercial contract between Tecquipment Limited and the Council of Higher Education for the supply of equipment to the Turkish Universities. The manufacturing equipment is a Flexible Manufacturing System which can be classified as a multi-cell flexible manufacturing system (MCFMS) according to structural forms that an FMS may take, proposed by MacCarthy and Liu [14].

The Flexible Manufacturing System at the Mechanical Engineering Department basically includes two manufacturing cells and a closed-loop conveyor as the main material handling system, where all are controlled by the supervisory (host) computer. A general view of the system is given in Figure 3.1.



Figure 3.1 General View of the System

Both cells are equipped with a CNC machine (Cell-1 has a CNC lathe and Cell-2 includes a CNC milling machine), robots for inner-cell transportation and one of the cells has a transportable input/output buffer which can be located in the cell depending on the product mix. A view of the first cell containing one lathe, one robot and a buffer is given in Figure 3.2. and a view of the closed loop conveyor is given in Figure 3.3

List of the original items in the system are as follows (Except the host controller as it is supplied by the Mechanical Engineering Department):

1. CNC Lathe (Denford/UK): PC based, medium-duty lathe having two simultaneously controlled axis. It has an automatic turret having 8 tool stations. Door and chuck of the lathe are controlled by the software and pneumatically powered.
2. CNC Milling Machine (Denford/UK): PC based, medium-duty milling machine having three simultaneously controlled axis. It has an automatic tool magazine having 6 tool stations. Door and chuck of the lathe can be controlled by the software and pneumatically powered.
3. One Closed Loop Conveyor (SKF/UK): Uni-directional closed loop conveyor having 14 cups for rotational parts. It is driven by an electric motor and a gear unit.
4. One Co-ordinate Measurement Machine (Kemco/UK): 3-axis CMM having a feature based control software. It is out of the scope of this study as there is technical problems with the hardware of the machine.
5. Two Robots (Mitsubishi/JAPAN): 6-axis controlled material handling robot. It can store programs in its RAM or EPROM and they can be executed by external triggering or program commands

can be send by RS-232 connection to the controller

6. Two Cell Computers : 386-SX 25 CPU , 4 MB RAM, 40 MB Disk storage, Flytech I/O board and RS-485 communication card installed.
7. One part buffer having a hopper and a chute.



Figure 3.2 View of the First Cell

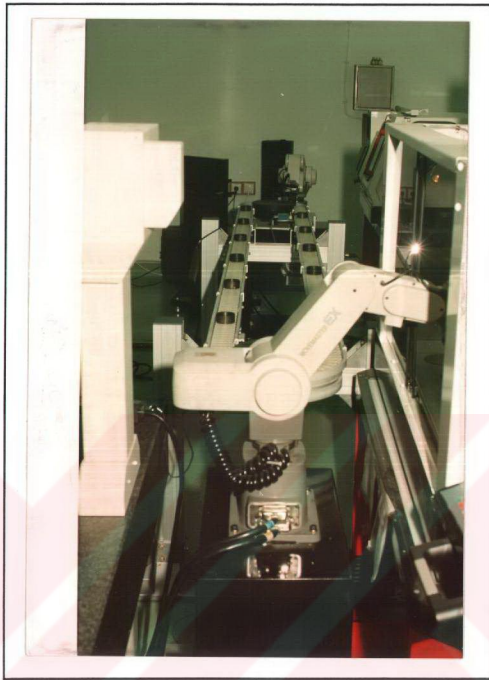


Figure 3.3 View of the Closed Loop Conveyor

Communication between the cell devices is done by Flytech Industrial I/O cards that are designed for industrial process control, fully compatible with the IBM personal computer. It offers 16 sets independent relay out to control external devices and 16 sets photo couple inputs to detect the input signal. 16 bits in, 16 bits out channels from the card are connected to a interface module that has ports to devices. The layout of the Pilot FMS is shown in Figure 3.4.

The original system was so called a pilot FMS system but it was lacking of many essential functions of an FMS:

1. There is no Direct Numerical Control (DNC) link between the cell computers and the CNC machine tools. So as there is no way of downloading NC-Codes to machine tool controllers which means that only one type of part can be produced in one production cycle.
2. The host computer software can not make any load balancing or scheduling between the two cells. Raw material can enter the system from cell-1 and leave the system from the same cell. So the only possible routing for a part is processing in cell-1 and then in cell-2.
3. The network hardware between the cell computers and the host computer is based on RS-485 standard which is a low-cost device connection. This is not an open architecture for manufacturing communications which can support long-term flexibility of the system.

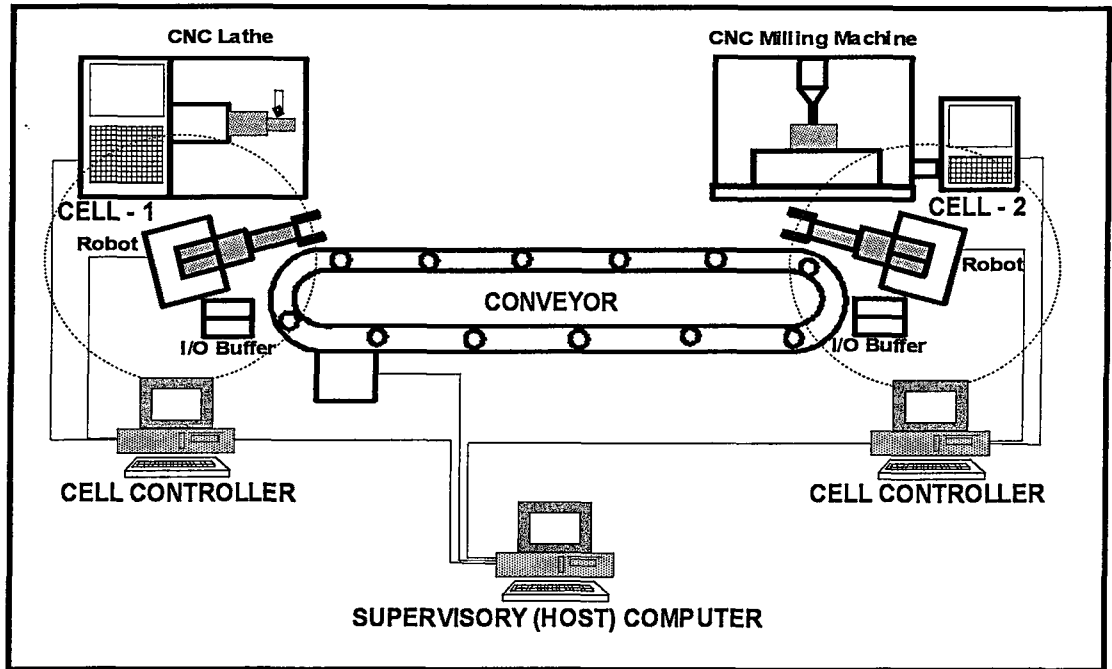


Figure 3.4 Layout of FMS

3.2 Modifications Done on The System

As the original system has been inspected, it was obvious that a lot of hardware modifications should be done before developing of new controller softwares. The modifications done on the system can be given as follows:

1. The serial I/O cards of the machine tool controllers are replaced with new ones and RS-232 connection is established between the cell controllers and machine tool controllers. So by downloading NC-codes of parts to machine controllers it can be possible to

produce more than one type part in a cycle.

2. A new part buffer is manufactured in the Machine Shop of Mechanical Engineering Department of METU for the second cell so that parts can enter and leave the system either from cell-1 or from cell-2. So a proper scheduling algorithm can be implemented in control softwares allowing a number of routings.
3. The RS-485 communication cards between cell controllers and host computer is replaced with ethernet cards and the system is connected to Local Area Network (LAN) of the Mechanical Engineering Department. So the network structure is converted to an open architecture that can support long term hardware flexibility.

CHAPTER 4

THE SOFTWARE PACKAGES DEVELOPED

4.1 Introduction

In the hierarchical architecture of METUCIM, two software packages for the management of cell computers and host computer tasks have been developed. The developed softwares run under Microsoft Windows environment and C++ compiler is used to develop them.

4.2 C++ and Windows Environment

Windows operating environment offers a unique graphics user interface, multi-tasking capability and an event-driven execution environment. C++ is an extension of commonly used C programming language providing object oriented programming features. C++ was developed at AT&T Bell Laboratories in the early 1980s and is still evolving. Software package used for the development of software is BC++ 4.5 produced by Borland International for Windows programming.

Windows applications were originally programmed by ANSI C but today

Object-Oriented Programming is greatly preferred for Windows Programming. C++ provides the opportunity to manipulate Windows functions in an object-oriented manner. Object Windows Library introduced by Borland International with the coming of Turbo C++ for Windows 3.0 is an object oriented class library that encapsulates the behaviors that Windows applications usually exhibit. It can be considered as a simple interface to Windows environment that offers a framework to get rid of hard and time-consuming programming of Windows in structural style [30].

4.3 Host Controller

The host controller receives the predictive schedule from the planning level and carries out the operational functions of the cell controllers and the conveyor for the execution of the schedule. It can also generate a predictive schedule for the part types in the shop database using its built-in schedule algorithm. A generated predictive schedule by the built-in scheduler is presented in the form of a Gantt chart by the application's graphics interface and a sample is shown in Figure 4.1.

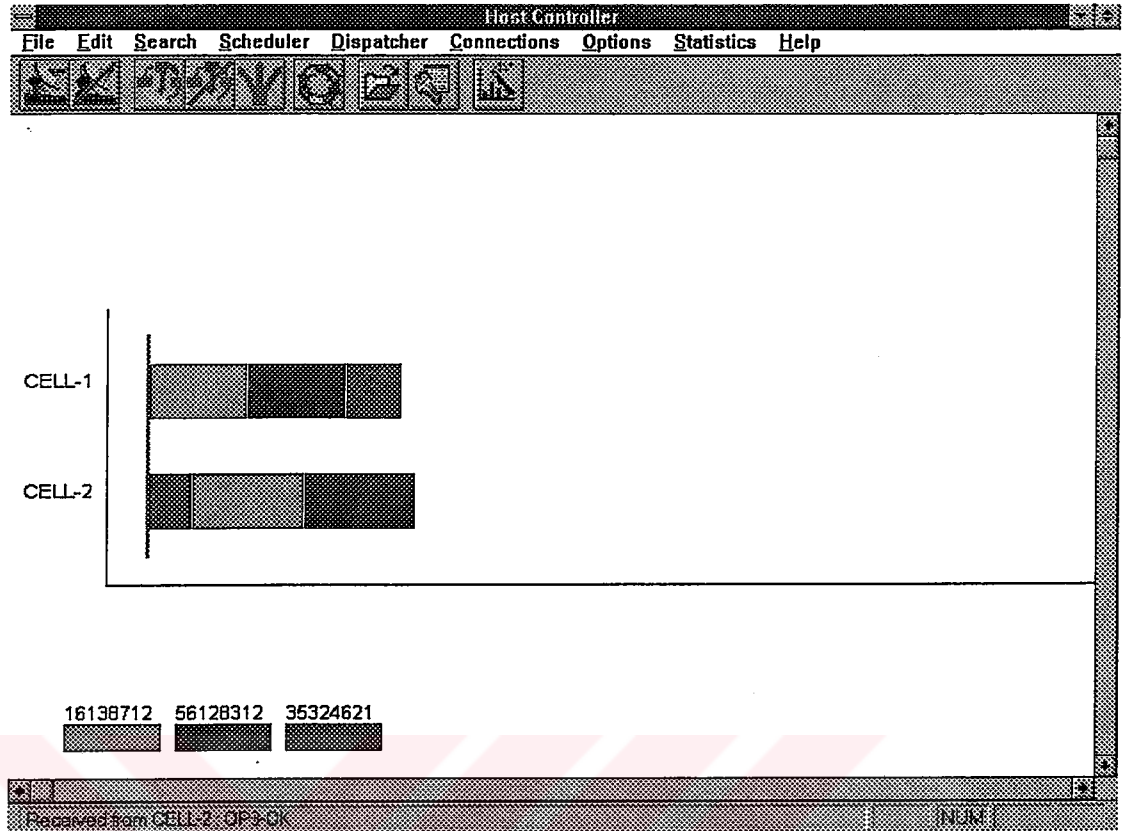


Figure 4.1 Predictive Schedule in Form of Gantt Chart.

At this level, the predictive schedule is considered as a reference schedule to be realized, but due to the dynamic nature of the FMS, the host controller has to take reactive actions against the unpredictable events like change in the machining times of parts or the material handling times of parts. Eventually, the reactive schedule is generated due to these changes as shown in Figure 4.2.

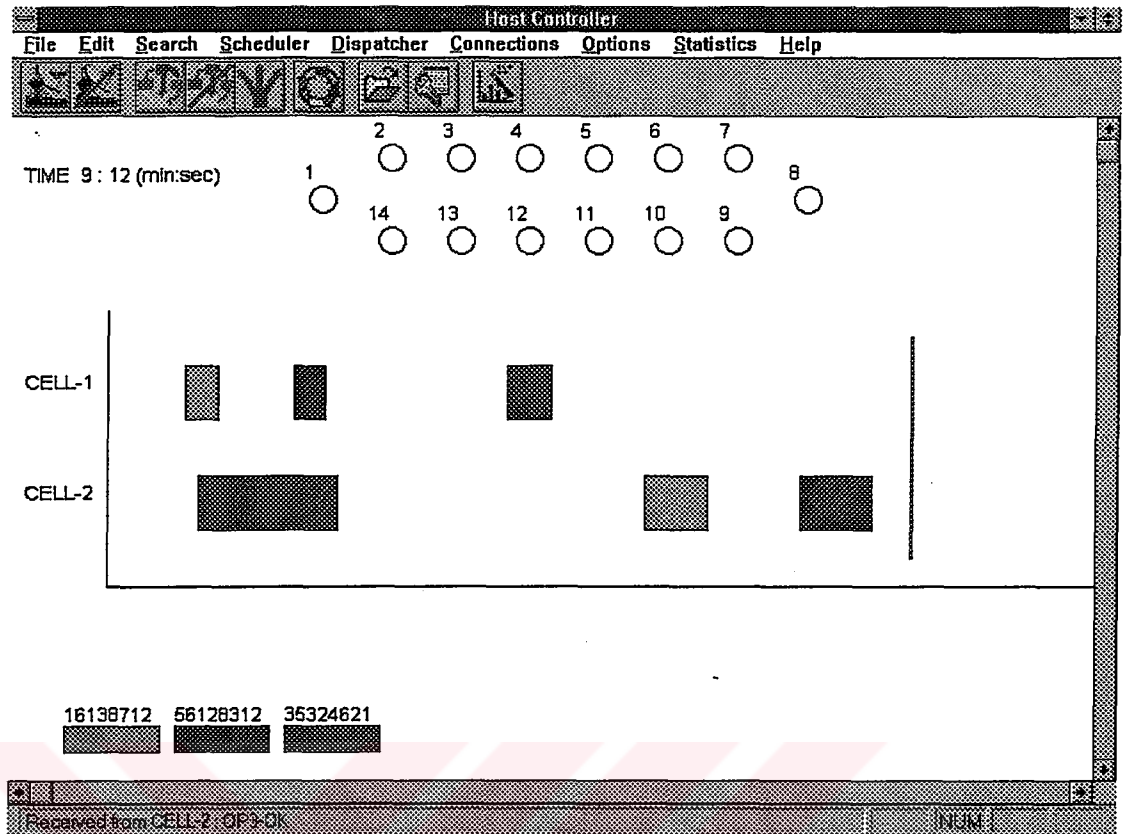


Figure 4.2 Reactive Schedule Generated After a Completed Cycle.

For the management of the host computer tasks, several objects are designed and implemented as in Figure 4.3. Each object is assigned responsibilities for certain tasks of the host computer. The objects of the host computer control structure and their assigned tasks are as follows:

4.3.1 Ethernet Driver Object (Server)

This object gives services to other objects to establish connection between cell, shop floor management and planning levels using TCP/IP protocol. User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) are two most important protocols of the transport layer of TCP/IP suite. UDP provides a procedure for application programs to transfer data over the network with a minimum overhead as there is no need to establish a connection. There is no method in the protocol to verify that the data reached its destination exactly. The data may be lost, duplicated or arrive out of order. So it is an unreliable protocol to use. Instead, TCP provides a reliable way of data transmission establishing a client/server connection between the communicating devices. It is called as a connection-oriented protocol because two ends exchange a handshaking dialogue before data transmission begins. TCP takes a stream of data, breaks it into datagrams and sends each one using IP. If any datagrams are lost or damaged during transmission, it detects these facts and resends the missing datagrams. Considering these facts among the two protocols, Transmission Control Protocol is used at Transport Layer.

The Windows Sockets Application Programming Interface (Winsock API) is a library of functions that implements the socket interface to TCP/IP protocol suite. After linking the application with the WINSOCK.LIB import library, the functions in the library for a TCP or UDP connection can be used by the application program for data transmission.

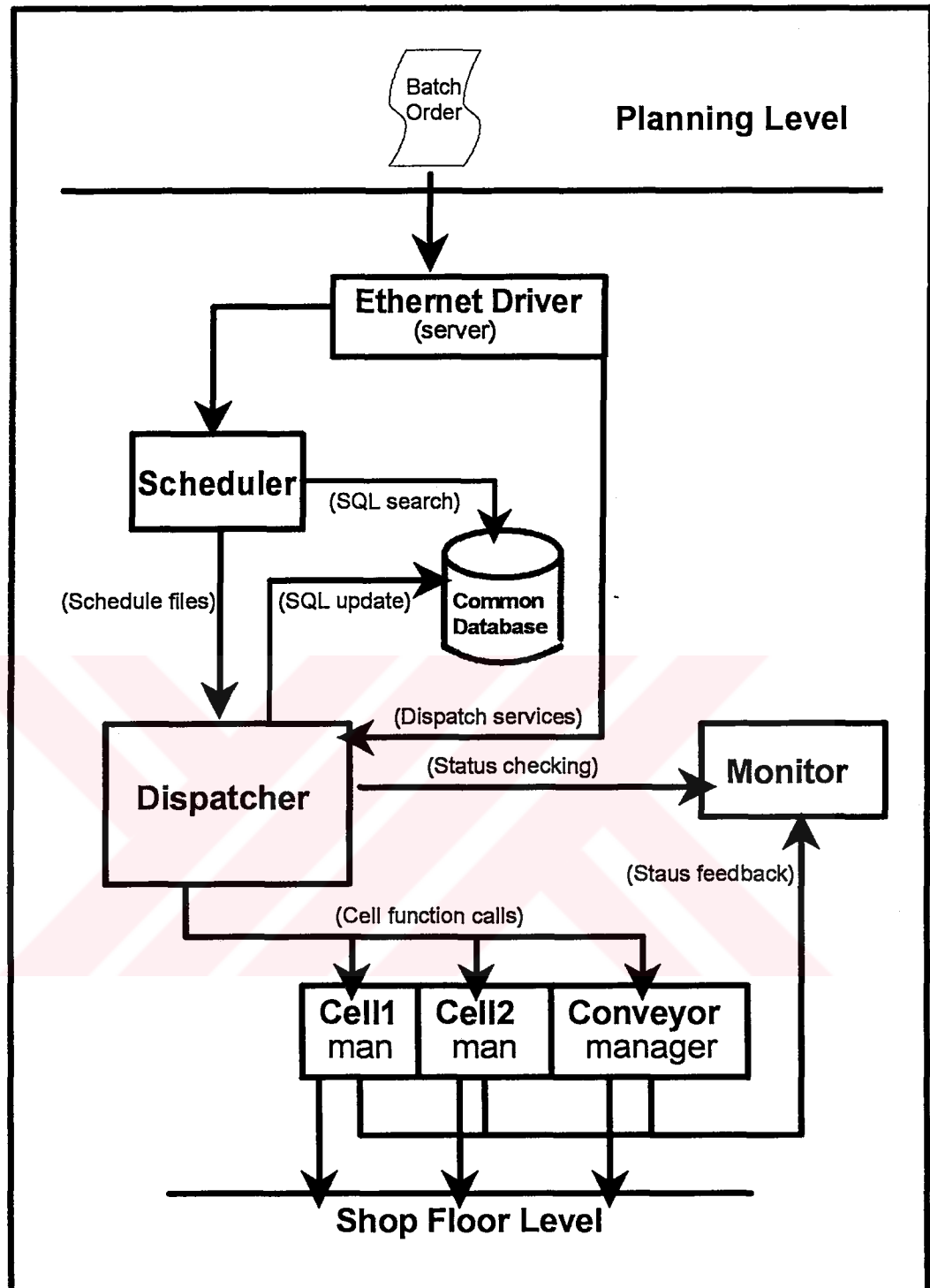


Figure 4.3 Host Controller Structure

4.3.2 Scheduler Object

The *Scheduler* object generates a scheduled batch for the selected materials in the shop database. By the nature of the Pilot FMS, the scheduling task can be regarded as a two-machine problem. Among the well known two-machine, flow-shop scheduling methodologies, Johnson's Algorithm which is modified by Kusiak to be applicable for job-shop manufacturing, is used for predictive scheduling. On the other hand, the material handling time is not considered in this algorithm. Due to this fact, the predictive schedule does not contain any information about the transportation time of parts between the cells. This fact is reflected to the reactive schedule by considering the operation of the conveyor. In other words, the predictive schedule generated by Kusiak's Algorithm is regarded as a reference (initial) schedule and further modified by the host computer on real-time basis while generating the reactive schedule.

Johnson's algorithm for n operations on two machines, each operation in the order of machine M1 and machine M2 so that maximum flow time is minimized can be given as [31]:

Step 1: Set $k=1$, $l=n$

Step 2: For each operation, store the shortest processing time and corresponding machine number.

Step 3: Sort the resulting list, including the triplets " operation number/

processing time/ machine number” in increasing value of processing time.

Step 4: For each entry in the sorted list :

IF machine number is 1, then

(I) set the corresponding operation number in position k,

(II) set $k = k+1$.

ELSE

(I) set the corresponding operation number in position l,

(II) set $l = l-1$.

END IF

Step5: Stop the entire list of operations has been exhausted.

This algorithm was further improved by Kusiak for two-machine job shop problem. For a two machine job shop manufacturing of a set of n operations there will be four possible route types as in Figure 4.4.

1. Route A Operations to be processed only on machine M1
2. Route B Operations to be processed only on machine M2
3. Route C Operations to be processed on both machines in the order of M1 and then M2.
4. Route D Operations to be processed on both machines in the order of M2 and then M1.

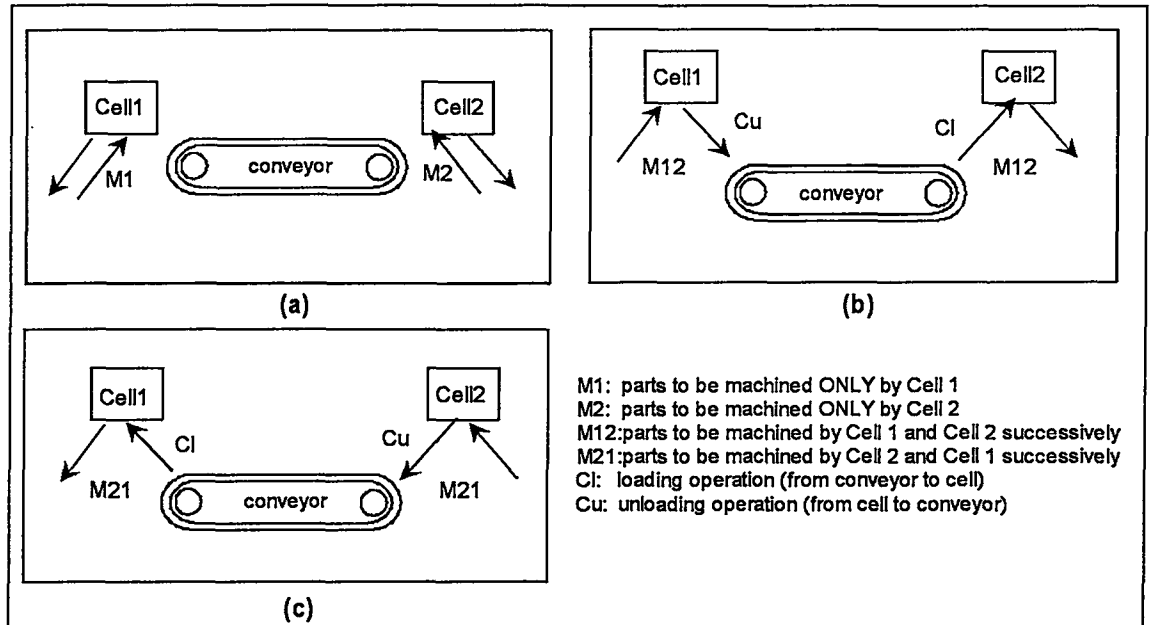


Figure 4.4 Routes of the System

The algorithm to construct an optimal schedule is [31]:

Step 1: Schedule the operations of route A in any order to obtain the sequence Sa

Step 2: Schedule the operations of route B in any order to obtain the sequence Sb

Step 3: Schedule the operations of route C according to Johnson's flow shop algorithm producing the sequence Sc

Step 4: Schedule the operations of route D according to Johnson's flow shop algorithm producing the sequence Sd

Step 5: Construct an optimal schedule as follows:

Machine M1: Sc, Sa, Sd

Machine M2: Sd, Sb, Sc

4.3.3 Dispatcher Object

Workbase of the *Dispatcher* is the schedule received from the planning level or the schedule generated by the built-in scheduler. This object initializes driver objects for managing the cells and closed-loop conveyor and gives timer services to these objects for synchronization. It reads the required schedule file from the disk and loads the part sequences into the buffers of *Cell Managers*.

Another duty of the *Dispatcher* object is the co-ordination of the downloading of order and G-Code files to cell computers before starting a production cycle. *Dispatcher* uses the services provided by the *Ethernet Driver* to complete this task.

4.3.4 Monitor Object

The structure of the *Monitor* is similar to a look up table. It contains the operation status of the cells and the conveyor which is checked by *Dispatcher* and *Cell Managers*.

4.3.5 Cell Manager Object

This object is responsible for operation file synchronization for an individual cell. A partpiece can be processed in a cell with the execution of three different operation files. These operation files are for loading of a part into a CNC, manufacturing of the part in the CNC and unloading of the part from the CNC. The operation files will be explained in detail in Section 4.4.

Request for conveyor allocation is another duty of the cell manager. Whenever an unloading/loading operation for transportation of a material using the conveyor is required, a request message to the message queue of the *Conveyor Manager* is posted using the user-message features of the Windows system. After the right positioning of the conveyor, receiving an acknowledgment message from the *Conveyor Manager*, required operation with the conveyor is performed.

4.3.6 Conveyor Manager Object

All the material handling activities for transportation between cells are carried out by this object. Status of the conveyor, part identities and their location on the conveyor are all logged by this object. It has a message queue to store the conveyor operation requests that are coming from *Cell Managers* and responds to these messages considering the availability of conveyor. The queue in the *Conveyor Manager* has a FIFO (first in first out) structure. If for any reason a request can not

be responded -a part may not be on the conveyor for a loading request by a cell-, it can be sent to auxiliary queues and saved there for further processing.

4.3.7 Common Database

Common database of the shop floor is maintained using MS-Access 2.0 database program. In order to connect to the database and send commands written in Sequential Query Language (SQL), Open DataBase Connectivity (ODBC) standard of Microsoft is used.

ODBC defines a method of connecting to data sources that is open to as many applications and data sources as possible. To accomplish this openness, the application and the database must agree on a common method of accessing the database. This is implemented as a published standard defining a complete set of Application Program Interface (API) function calls and a similarly complete SQL syntax set.

The database drivers are contained in Dynamic Link Libraries (DLLs). These drivers will transform the ODBC API functions into function calls supported by the particular data source being used. Similarly, the drivers transform the ODBC SQL syntax into syntax accepted by the data source. Different data sources can be accessed by just loading a different DLL [29].

The fields and example records of the common database are given in

Table 4.1.

Table 4.1 Fields of Common Database.

PART_NAME	REF_NUMBER	RAW_MATERIAL	TYPE
PART-1	16138712	70	M12
PART-2	35324621	70	M21
PART-3	56128312	70	M12
LATHE_G_CODE_FILE	LATHE_TIME	MILL_G_CODE_FILE	MILL_TIME
16138712.MIR	25	16138712.FNC	31
35324621.MIR	23	35324621.FNC	81
56128312.MIR	51	56128312.FNC	32

First field is the name of the part. Reference number field is a unique eight digit number given to each part. Raw material field gives the length of the raw material rod which are stored in cell buffers. Type of the part indicates the route of the part which is a route A part for M1, route B part for M2, route C part for M12 and route D part for M21. The G-code filename and time fields for turning and milling processes give the filenames and cutting times of parts in the database.

4.4 Cell Controller

Cell controller receives the orders from the host computer and carries out the relevant discrete functions of the manufacturing resources within the cell. In achieving the realization of the orders, a cell controller has to control the pre-defined tasks to carry out the transportation and machining of parts within the cell. The cell controller establishes connection with the cost controller as a client and obeys the

commands coming from host controller sending back necessary feedback about the cell status. The tasks that it should handle within the cell are material flow within cell, establish DNC with CNC machines and synchronization of device functions. The cell controller also displays the cell devices status to the operator as in Figure 4.5.

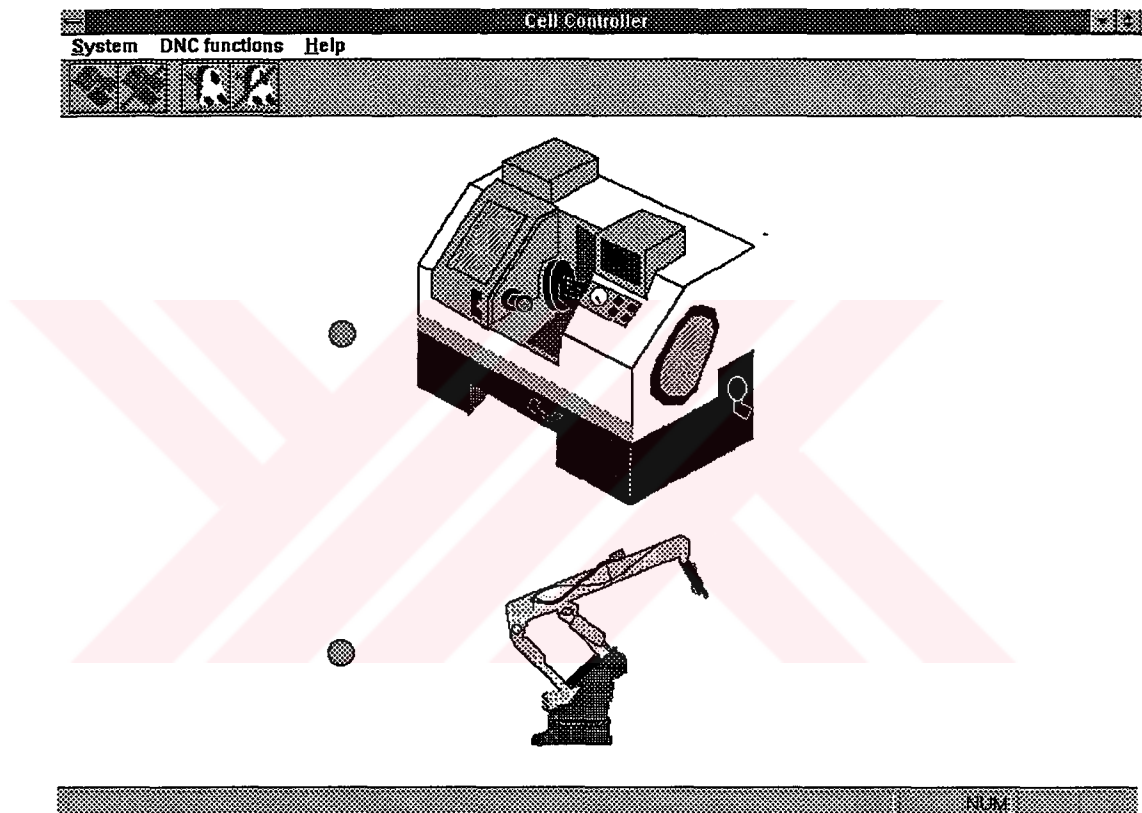


Figure 4.5 Main View of the Cell Controller

The objects of the cell controller structure can be seen in Figure 4.6 and their assigned tasks are as follows:

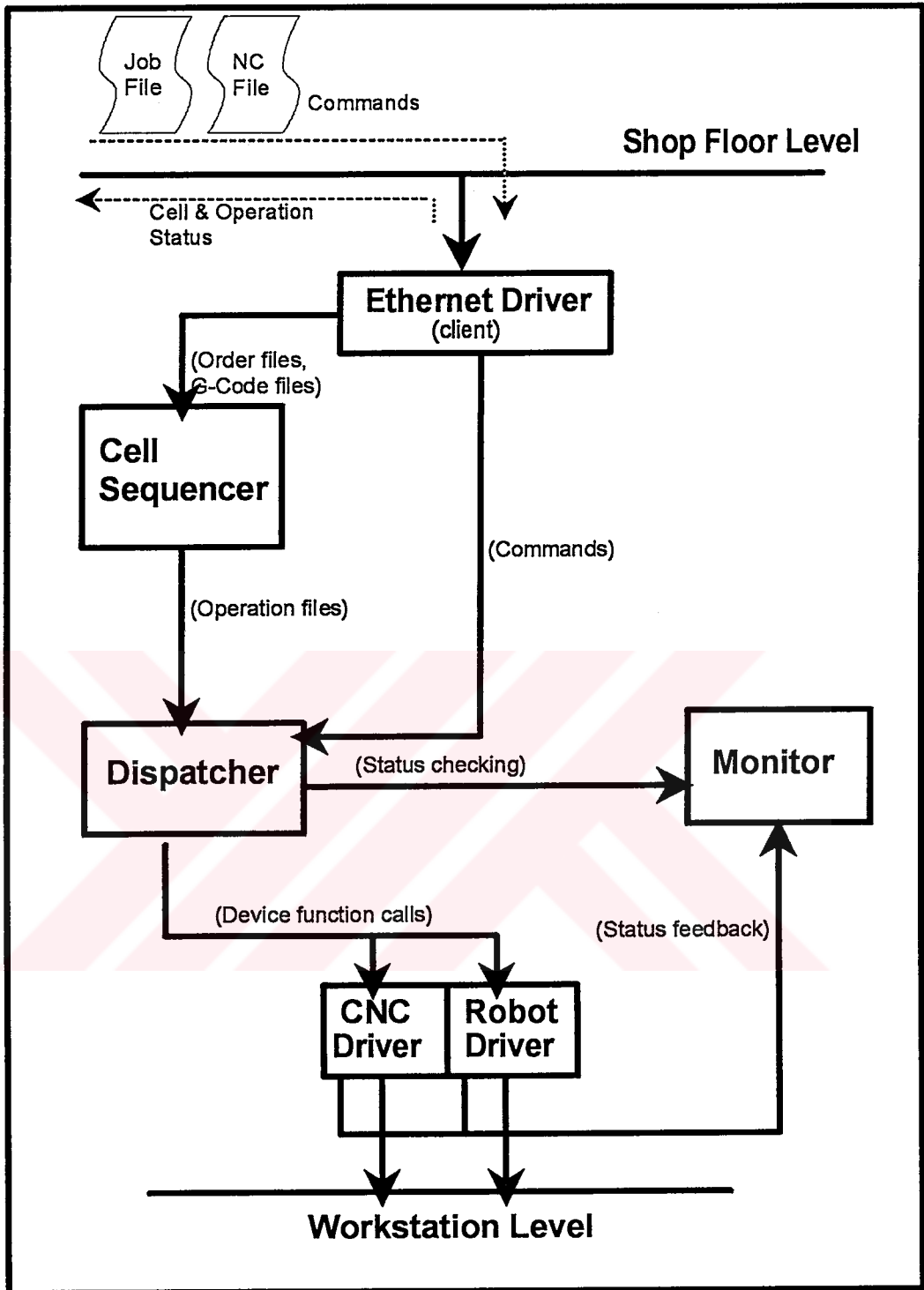


Figure 4.6 Cell Controller Structure

4.4.1 Ethernet Driver Object (Client)

This object establishes connection with the *Ethernet Driver (Server)* object of the shop controller. It receives the commands and G-Code files send by host controller posting them to the destination objects and provides services to *Dispatcher* object for operation status feedback to the shop controller.

4.4.2 Cell Sequencer Object

Main duty of this object is to generate operation files to sequence the operation of the cell devices. In order to process a part in a cell there are three different operations to be accomplished. These are:

1. Loading of the part from the buffer or the conveyor into the CNC chuck
2. Downloading and execution of a G-Code file to the CNC controller to machine the part.
3. Loading of the part from the CNC chuck into the buffer or the conveyor.

An example of a part load operation file with its check commands for operation safety are as follows:

- 01 CMRB-01 VRNOCYC VRCKOPN VRDROPN
- 02 CMCK-CL VRNOCYC
- 03 CMRB-02 VRNOCYC VRCKCLD VRDROPN
- 04 CMDR-CL VRNOCYC

The first line is a command for the robot to accomplish the task no.1 which refers to loading part from buffer to the machine tool. The trailing verification commands are to check that whether the machine tool is not in cycle, the chuck is open and the door is open. Similarly, the second line is the command to close the chuck, the third line refers to move the robot back to park position and the last line is to close the door. A complete list of commands and verification codes are given in Table 4.2.

4.4.3 Dispatcher Object

Workbase of the cell *Dispatcher* is the sequence files generated by the cell sequencer. It has the real time control over cell devices.(Machine Tool Controller (CNC) and Robot Controller (ROC)). By the initiation of the commands coming from host controller it can open and read a certain operation file and it performs synchronization of the commands in these files whilst checking the machine status in the *Monitor* object.

Table 4.2 List of Commands and Verification Codes for Cell Devices

<i>No</i>	<i>Command/ Verification code</i>	<i>Description</i>
1	CMRB-XX	Execution of the robot move number referred as XX.
2	CMDR-OP	Open the CNC door
3	CMDR-CL	Close the CNC door
4	CMCK-OP	Open the CNC chuck
5	CMCK-CL	Close the CNC chuck
6	CMDW- <filename>	Download the G-Code file to the CNC controller and execute
7	VRNOCYC	Verify that the CNC is not in cycle
8	VRDROPN	Verify that door of the CNC is open
9	VRDRCLD	Verify that door of the CNC is closed
10	VRCKOPN	Verify that chuck of the CNC is open
11	VRCKOCLD	Verify that chuck of the CNC is closed

4.4.4 Monitor Object

The *Monitor* object contains the status of the cell devices (Machine Tool Controller (CNC) and Robot Controller (ROC)) which is checked by *Dispatcher*.

4.4.5 CNC Driver Object

Connection for Machine Tool Controller using RS-232 C standard is

established by this object. All DNC functions of the controller are inherited in this object and activated by *Dispatcher* object during execution of operation files. It updates the status of the CNC in *Monitor* object.

4.4.6 ROC Driver Object

Connection with Robot Controller using an industrial I/O board is established by this object. Task numbers are sent to Robot Controller for jumping to the subroutines of Robot programs that are in the RAM of the Robot Controller. It also updates status of Robot in *Monitor* object.



CHAPTER 5

TEST RUNS

5.1 First Test Run

Three different parts are recorded in the database file that is used for the test runs. The drawings of these parts are in Figures 5.1, 5.2, 5.3. The database named “db1.dbf” is accessed by selecting Scheduler/MS Access 2.0 Database in menu and using Select Database dialog box as shown in Figure A.4. The batch is constructed by selecting 3 times for each part using the dialog box in Figure A.2 in User’s Manual.

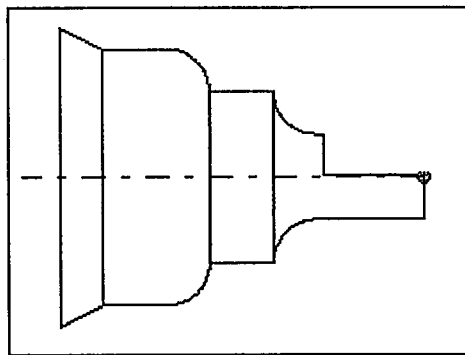


Figure 5.1 Drawing of Part-1

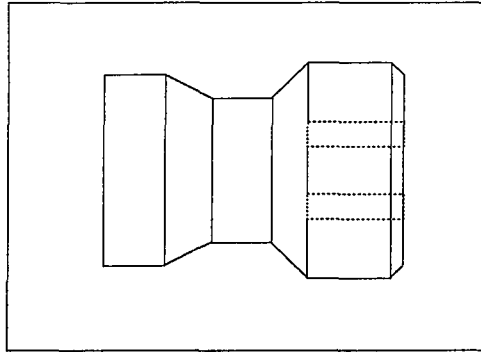


Figure 5.2 Drawing of Part-2

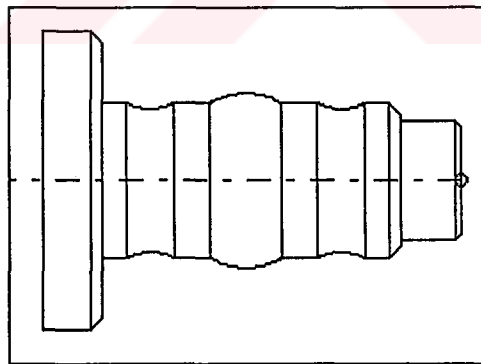


Figure 5.3 Drawing of Part-3

After the parts are selected and OK button is pressed a name should be given to the schedule file generated by the FileSave dialog box as shown in Figure A.3. Then, the scheduler object generates the schedule file and the order files to send to the cell controller shown in Figures 5.4, 5.5, 5.6, 5.7 successively.

```
cell1 9
OP11 16138712.drd
OP12 16138712.drd
OP13 16138712.drd
OP14 56128312.drd
OP15 56128312.drd
OP16 56128312.drd
OP17 35324621.drd
OP18 35324621.drd
OP19 35324621.drd
cell2 6
OP21 35324621.drd
OP22 35324621.drd
OP23 35324621.drd
OP24 16138712.drd
OP25 16138712.drd
OP26 16138712.drd
```

Figure 5.4 Schedule File Generated by the Built-in Scheduler

```
PART_NAME: PART-1
REF_NUMBER: 16138712
RAW_MATERIAL: 70
TYPE: M12
LATHE_G_CODE_FILE_NAME: 16138712.MIR
LATHE_PROCESS_TIME: 205
MILL_G_CODE_FILE_NAME: 16138712.FNC
MILL_PROCESS_TIME: 125
```

Figure 5.5 Order File of Part-1

PART_NAME: PART-2
 REF_NUMBER: 35324621
 RAW_MATERIAL: 70
 TYPE: M21
 LATHE_G_CODE_FILE_NAME: 35324621.MIR
 LATHE_PROCESS_TIME: 130
 MILL_G_CODE_FILE_NAME: 35324621.FNC
 MILL_PROCESS_TIME: 139

Figure 5.6 Order File of Part-2

PART_NAME: PART-3
 REF_NUMBER: 56128312
 RAW_MATERIAL: 70
 TYPE: M1
 LATHE_G_CODE_FILE_NAME: 56128312.MIR
 LATHE_PROCESS_TIME: 224
 MILL_G_CODE_FILE_NAME: 56128312.FNC
 MILL_PROCESS_TIME: 0

Figure 5.7 Order File of Part-3

After the cycle is completed process time of each part is measured and recorded to the database again. In Table 5.1 this data is presented.

Table 5.1 Machining Times of First Test Run

Part Name	Ref-Number	Lathe Time	Mill Time
Part-1	16138712	205	100
Part-2	35324621	130	139
Part-3	56128312	224	0

The statistical data for each cell including Total Operation Time, Idle Time, Machining Time and Cell Utilization can be obtain by selecting Statistics/OP-times menu as in Figure 5.8.

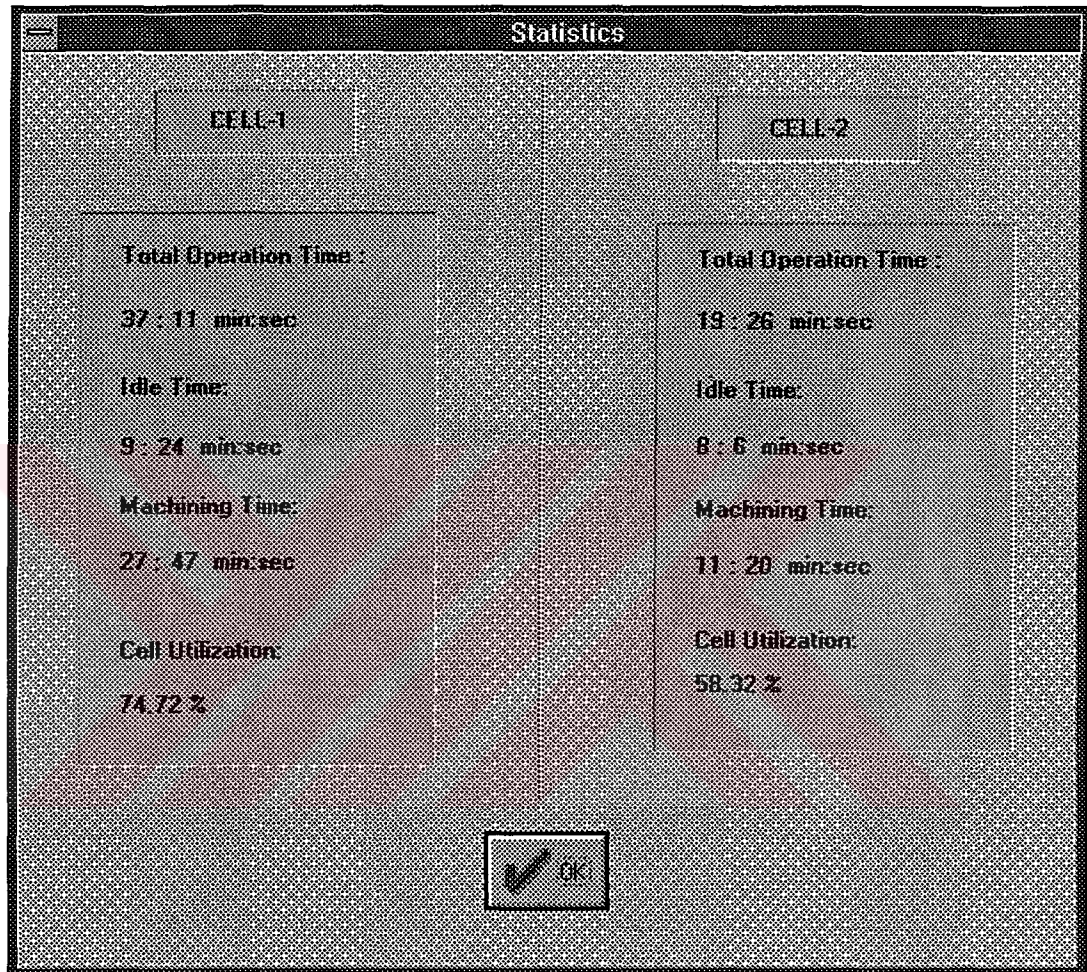


Figure 5.8 Statistics Data of First Test Run

5.2 Second Test Run

In the second test run the same batch with the first test run are used with modified G-Code files. The feedrate parameters in the G-Code files decreased in order to increase the actual cutting times of the parts. The modified G-Code files of the parts are in Appendix B.

After the cycle is completed machining times are recorded as shown in Table 5.2. As the cutting feedrate parameters are reduced in the G-Code files of the second run, it can be observed that new values are increased.

Table 5.2 Machining Times of Second Test Run

Part Name	Ref-Number	Lathe Time	Mill Time
Part-1	16138712	223	126
Part-2	35324621	137	146
Part-3	56128312	256	0

The statistical data of the second run is shown in Figure 5.9

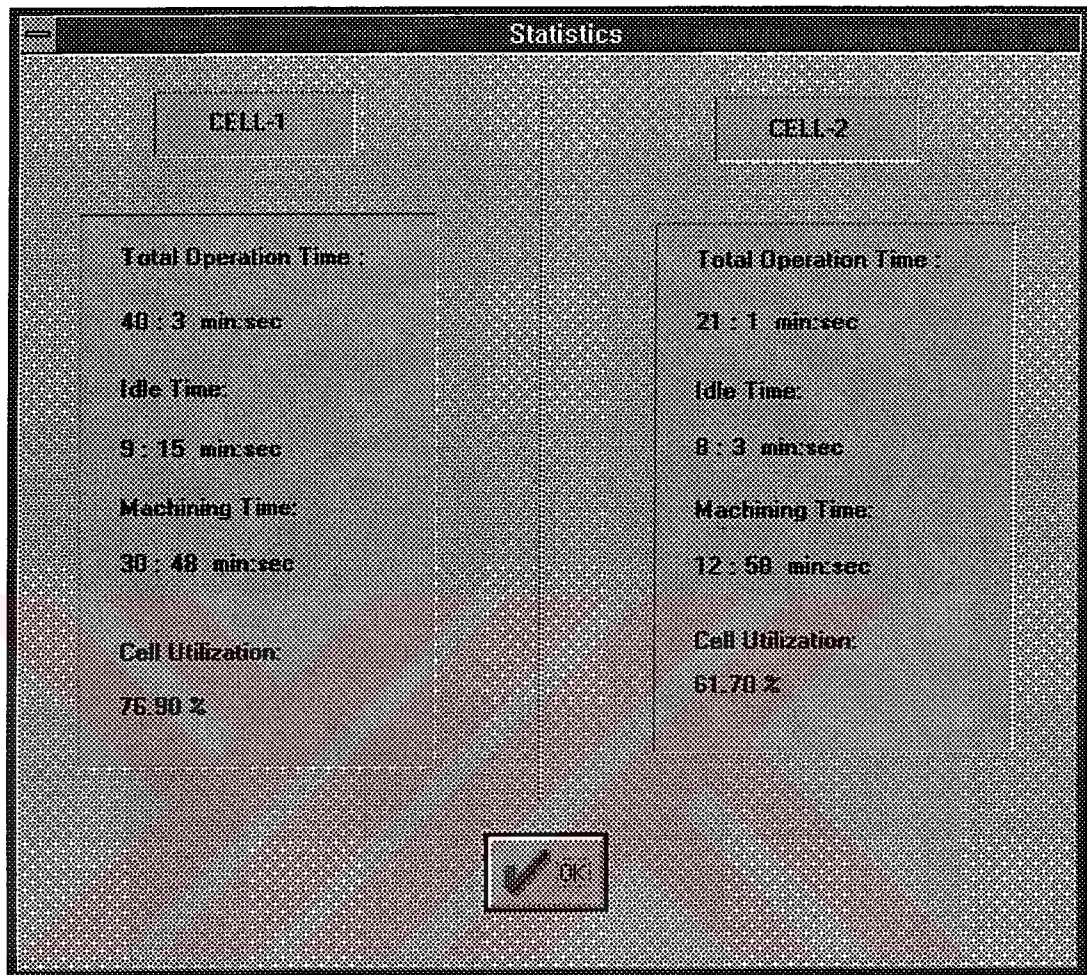


Figure 5.9 Statistics Data of Second Test Run

CHAPTER 6

CONCLUSION AND FUTURE WORK

The goal of this study is to design a hierarchical control architecture and develop control softwares for a Pilot Flexible Manufacturing System in the Mechanical Engineering Department of the Middle East Technical University. C++ programming language is used and the developed software family runs under Windows environment.

Before starting the software development, necessary hardware modifications on the system are completed. Between the cell controllers and CNC controllers, RS-232 connection is established to make Direct Numerical Control (DNC). RS-485 connection between the cell controllers and the host computer is converted to Ethernet hardware and communication is done using Windows Sockets which is based on TCP/IP protocol. A new part buffer is manufactured in the Machine Shop of Mechanical Engineering Department of METU for the second cell so that parts can enter and exit to the system both from cell-1 and cell-2. Thus, it became possible to implement a proper scheduling algorithm in control softwares allowing four different routings.

An Object-Oriented Design and Programming methodology is used to achieve a more flexible and modular software family. During the design stage, it has been noticed that the information hiding and the encapsulation mechanism provided by object-oriented approach significantly reduce the software complexity and increases modularity. Further, it allows more rapid system development, comprehension, maintenance and modification. Due to these facts, it can be concluded that the object-oriented approach leads to a generic controller structure. By defining specific tasks, the host and the cell controllers are differentiated and linked in a hierarchical manner to form the internal structure of the FMS controller.

In the structure of the hierarchical control the host controller has the main duty of controlling the part scheduling and dispatching of orders to cells in a production cycle. Johnson's two machine flow-shop algorithm which is modified by Kusiak for job-shop problem, is used as the built-in scheduling algorithm of the host controller. Generated schedules are displayed in form of a Gantt chart on the screen. Management of the closed loop conveyor, which is used to transport material between cells, is also duty of the host controller. A message driven technique is used to respond to the requests for conveyor operations coming from cells.

A common database for the shop floor is constructed in MS-Access database program. Host controller can access this database using Open Database Connectivity (ODBC) standard of Microsoft and give Sequential Query Language (SQL) statements to get search results for the parts in the database.

The cell controllers has the main duty of sequencing the inner-cell transportation by the robots and CNC processing. According to the route information of each part operation files for loading, unloading and manufacturing operations are generated by the cell sequencer and with the commands coming from the cell controller these files are loaded and executed by the cell dispatcher. G-Code and order files are received from the host controller by the cell controllers and G-Code files are downloaded to the CNC controllers using RS-232 connections.

In order to improve the softwares and approach to a more generic and modular shop floor control following future works are recommended:

1. Object of the controllers can be separated and each can be made an executable application. Using Dynamic Data Exchange (DDE) features of Windows environment these small applications can communicate between each other and modularity can be greatly increased with this approach.
2. A standard class library can be developed and for the needs of each different FMS a new software group can be developed in a short period time by picking up necessary objects from the library. So with the use of a generic class library softwares can run on any kind of FMS regardless of the hardware and configuration differences.

3. Common Database can be enhanced more to handle a massy database of a real shop environment with better performance. A remote PC can be devoted as a database server to the system. Access to this database can be achieved by using ODBC server functions for ethernet hardware. Drawing information of each part can be kept in the database server and reached by the host controller or any other application working in the shop environment whenever needed.
4. In case of two or more machines dispatching rules can be used such as Shortest Processing Time (SPT), Earliest Due Date (EDD), First Come First Served (FCFS), First in System First Served (FISFS).
5. Management of the transportation between cells can be made more effective by coupling the message system by a rule-base or algorithms which can optimize the conveyor responses to requests from cells.
6. New classes for quality control of the parts can be developed and Co-Ordinate Measurement Machine (CMM) of the laboratory can be integrated to the system. Also Statistical Process Control (SPC) modules can be developed.
7. New classes for error reporting and error handling can developed.

REFERENCES

- [1] Parrish, D.J., 1990, *Flexible Manufacturing*, Butterworth-Heinemann, Oxford.
- [2] Mize, Joe H., and Glenn Palmer, 1989, "Some Fundamentals of Integrated Manufacturing," *International Industrial Engineering Conference Proceedings*, Toronto, Canada, pp. 575-580.
- [3] Bauer, A., Bowden, R., Browne, J., Duggan, J., Lyons, G., *Shop Floor Control Systems: From design to implementation*, Chapman & Hall.
- [4] Pels, H.J., Wortmann, J.C., Zwegers, A.J.R., 1995 "Flexibility in manufacturing: an architectural point of view", *Preprints of CIM at Work Conference*, Kaatsheuvel, The Netherlands.
- [5] Maleki R.A., 1991, *Flexible Manufacturing Systems; The Technology and Management*, Prentice-Hall, New Jersey.
- [6] Merchant, M.E., 1987, "The Factory of the Future - Technological Aspects", *Towards the Factory of the Future*, ASME, PED-1, pp. 25-36.
- [7] Martin, J., 1989, "Cells drive manufacturing strategy", *Manufacturing Engineering*, January, pp.49-57.
- [8] Franks, I., Loftus, M., and Wood, N. T. A., 1990, "Discrete cell control", *International Journal of Production Research*, Vol. 28(9), pp.1623-1633.

- [9] O'Grady, P., Bao, H. and Lee, K.H., 1988, "Issues in Intelligent Cell Control for Flexible Manufacturing Systems", *Computers in Industry*, Vol. 9, pp. 845-861.
- [10] O' Grady P., 1989, "Flexible Manufacturing Systems: Present Development and Trends", *Computers in Industry*, Vol 12, pp. 241-251.
- [11] Young, C. and Greene, A., 1986, "Flexible Manufacturing Systems", *American Management Association*, New York.
- [12] Kaltwasser, J., Hercht, A., and Lang, R., 1986, "Hierarchical control of manufacturing systems", *ICAF Information Control Problems in Manufacturing Technology*, Suzdal, USSR, pp.37-44.
- [13] Byrkkett, D.L., Ozden, M. H., Patton, J. M., 1988, "Integrating flexible Manufacturing systems with traditional manufacturing, planning and control." *Production and Inventory Management Journal*, Vol. 29, pp.15-21.
- [14] Maccarthy, B. L., LIU, J., 1993, "A new classification scheme for flexible manufacturing systems", *Int. J. Prod. Res.*, Vol. 31, No. 2, pp. 299-309.
- [15] Luggen, W. W., 1991, *Flexible Manufacturing Cells and Systems*, Prentice-Hall Int. Inc., pp.54-57, America.
- [16] Saygin. C., Ünver. Ö., Kılıç, E., Anlağan, Ö., 1996, "A Reference Architecture for a Generic Manufacturing Cell Controller", *7th International Machine Design and Production Conference*, to be presented.
- [17] Rogers, P., "Representation of the cell control task", *Manufacturing cells: control, programming and integration.*, Butterworth-Heinemann, pp.173-179.

- [18] Bedworth D., Henderson, M., Wolfe, P., 1991, *Computer Integrated Manufacturing*, Mc-Graw Hill.
- [19] Krumney, A., Kolman, J., 1987, "LAN Hardware Standards", *PC Tech Journal*, pp. 55-68.
- [20] Dumas, A., 1995, *Programming Winsock*, SAMS, pp.30-32, IndianaPolis.
- [21] Mitchell, F.H., 1991, *CIM systems, An introduction to computer integrated manufacturing*, Prentice-Hall, pp.370-371, New Jersey.
- [22] Rauch-Hindin, Wendy, 1986, "Revamped MAP and TOP Mean Business", *Mini-Micro Systems*, pp. 95-109.
- [23] Halsall, F., 1996, *Data Communications, Computer Networks and Open Systems*, Addison-Wesley.
- [24] Campbell, J., 1984, *The RS-232 Solution*, SYBEX, Berkeley.
- [25] Jones, A.T., McLean, C.R., 1984, "A cell control system for AMRF", *Proceedings of the 1984 International Computers in Engineering Conference and Exhibition*, pp. 353-359, Las Vegas, California.
- [26] Pelusi, J., 1989, "Cell control-the high and low of it", *Instrumentation and Control Systems*, pp 37-38.
- [27] O'Grady, P. and Seshadri, R., 1991, "X-cell-intelligent cell control using object-oriented programming (Part I)", *Computer-Integrated Manufacturing Systems*, 4, (3), pp. 157-163.

[28] Yep, C. K., Boey, S. H., Goh, J., 1993, "A flexible cell controller for Gintic flexible manufacturing systems". *Proceedings of the 2nd International Conference on Computer Integrated Manufacturing*, pp. 354-363.

[29] Timmermans, P., Szakal, L., Riessen, R., 1993, "On the modular design of shop floor control systems", *Proceedings of 9th CIM-Europe Annual Conference*, Amsterdam.

[30] Ünver, Ö., Saygın, C., Anlağan, Ö., Kılıç, E., 1996, "Design of an FMS Controller: An Object-Oriented Approach", *Second International Conference on Application of Fuzzy Systems and Soft Computing*, Siegen, Germany.

[31] Andrew, K, 1990, *Intelligent Manufacturing Systems*, Prentice Hall Int. Editions.

APPENDIX A

USER'S MANUAL

A.1 Installation

Host and cell controllers run under Windows 3.1 and higher versions. In order to install first insert the diskette in drive A: and copy the files in cell directory to the hard disk of the cell controller, and the files in the host directory to the host controller.

Files for the host controller are the followings:

- shop.exe
- odbcinst.dll
- odbcjt16.dll
- winsock.dll
- owl250.dll
- bc450rtl.dll
- bids45.dll
- bwcc.dll

Files for the cell controller are the followings:

- cell.exe
- winsock.dll
- owl250.dll
- bc450rtl.dll
- bids45.dll
- bwcc.dll

A.2 FMS Boot up Sequence

As FMS system of Middle East Technical University has a quite complicated structure, boot up of the devices should be in a specific order.

1. Turn the CNC controllers on using the red main power keys behind the machines.
2. Turn on the cell computers
3. Turn on the emergency stop box power key. After this you should see the Power on leds lighting on the interface modules under the monitors of the cell computers.
4. Turn on the host computer.
5. Enter windows environment both in cell computers and in host computer.
6. Run the cell controllers by clicking the mouse on the cell controller

icons. After this you should see the system enable led of interface boxes on.

7. Run the host controller by clicking the mouse on the host controller icon.
8. Turn on the robot controllers.
9. Push the reset and start buttons on the controller boxes so that to load and run the programs in the eprom of the robot controllers.
10. Reset the axis of the CNC machines using the membrane keypads.
Load the required offset file and leave at automode menu.

A.3 Host Controller

A.3.1 Host Controller Main Window

The host controller's main window is shown in Figure A.1

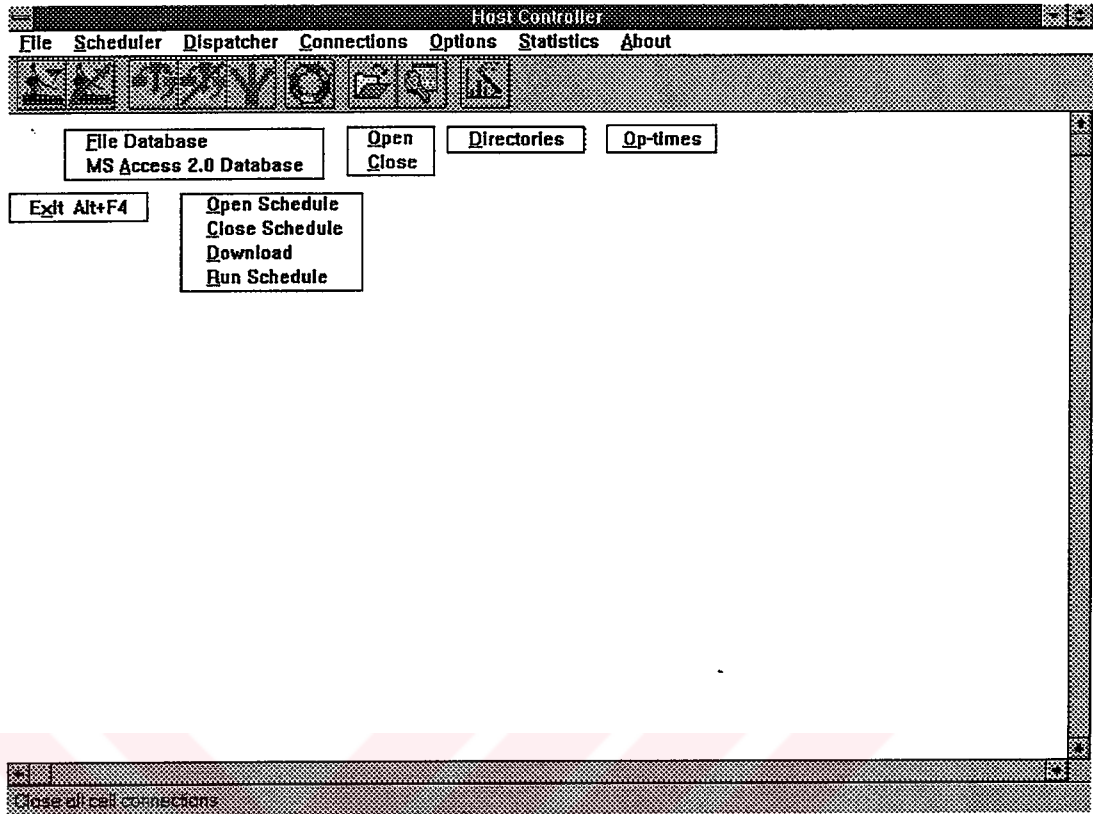


Figure A.1 Host Controller Main Window

- The minimize and maximize buttons appear in the upper right hand corner of the window,
- The Title Bar appears at the top of the window,
- The Menu Bar appears just below the Title Bar, and
- The Button Bar appears just below the Menu Bar.
- The Status Bar appears at the bottom of the window.

A.3.2 Menu Bar

As windows applications are menu driven softwares, they are very easy to use. User can select each function on the menu to activate or use the buttons that are under the menu bar for fast operation.

A.3.2.1 File/Exit

Closes the application and returns to program manager.

A.3.2.2 Scheduler/File Database

You can use this menu to prepare a batch using the order files in the current database directory.

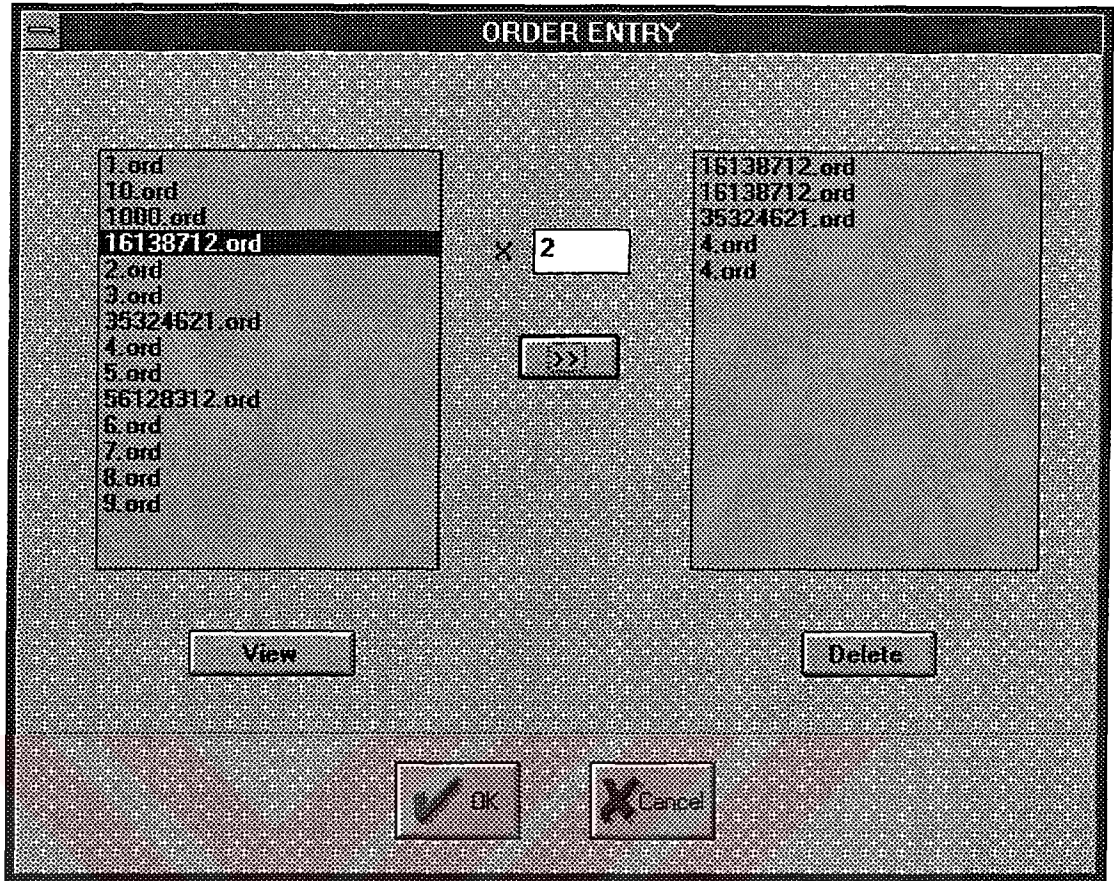


Figure A.2 Order Entry Dialog Box.

As seen in Figure A.2, You can select from the parts in the database using the right arrow button and add it to your batch. By increasing the multiplication factor, you select more than one from a part with one click to the right arrow.

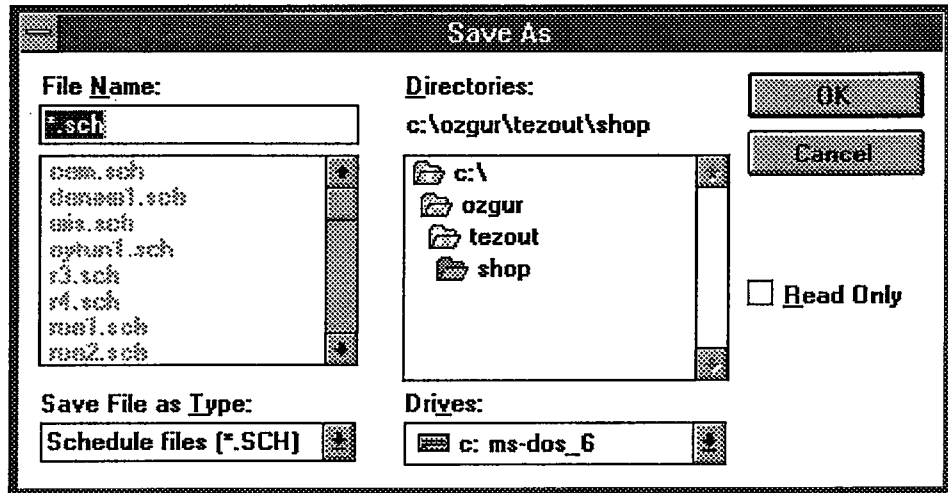


Figure A.3 File Save Dialog Box

Using the delete button, it is possible to remove a part from your batch. After your press the OK button FileSave dialog box will want from you a filename for the schedule file and saved the scheduled part list in this file as in Figure A.3.

A.3.2.3 Scheduler/MS-Access Database

Using this menu you can access to a MS-Access database and prepare a batch from the part that are recorded in the database. You can select the database file using the Select Database Dialog box as in Figure A.4.

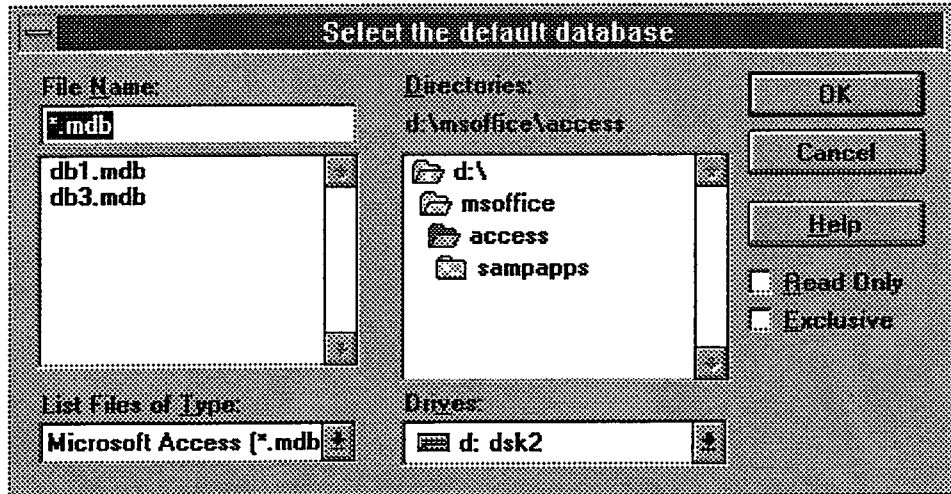


Figure A.4 Select Database Dialog Box

After selecting the database with the same dialog box of the File Database menu it is possible to prepare a batch and save it to the disk similar to section A.3.2.2

A.3.2.4 Dispatcher/Open schedule.

You can open a schedule file with File Open dialog box as in Figure A.5. Opened schedule will be displayed in form of a Gantt chart in the client area of the application's window frame as in Figure A.6.

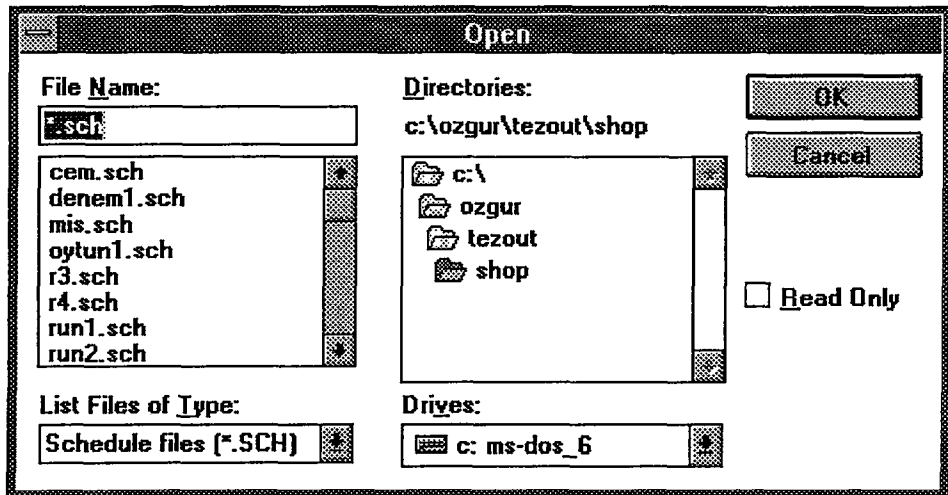


Figure A.5 File Open Dialog Box.

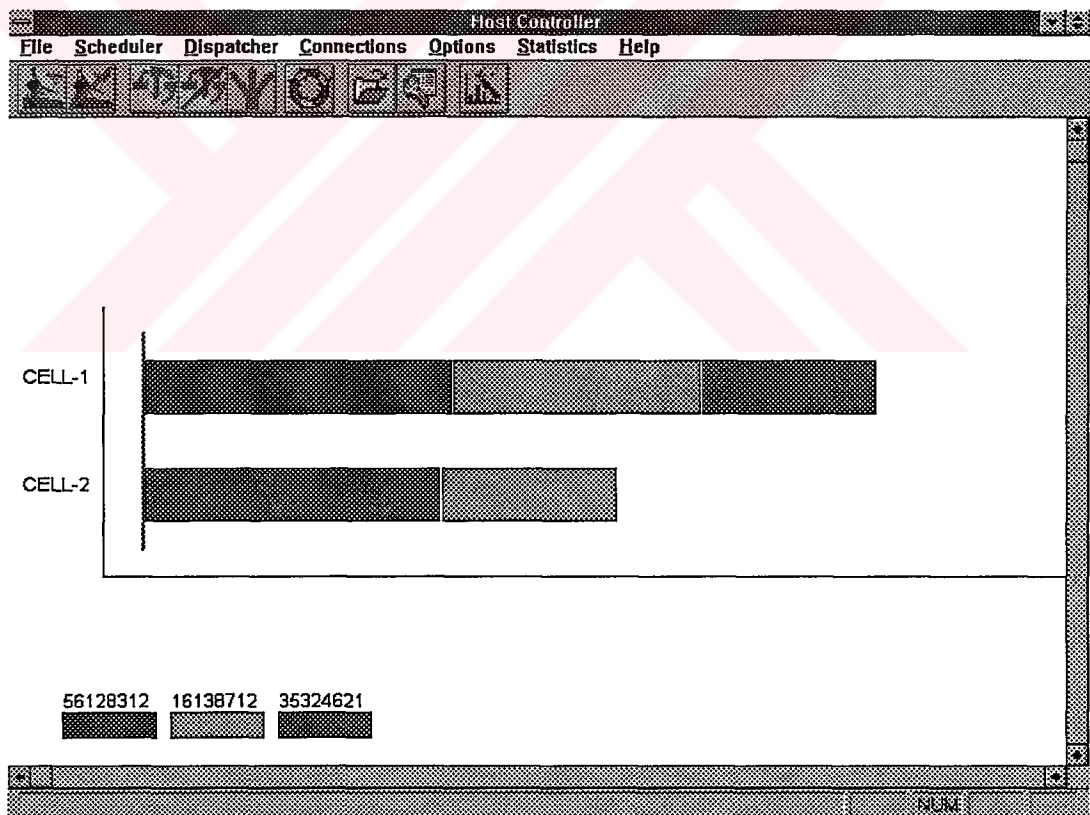


Figure A.6 A Schedule Displayed in Form of a Gantt Chart.

A.3.2.5 Dispatcher/Close schedule.

Closes an active schedule file and cleans the window.

A.3.2.6 Dispatcher/Download.

Prior to execution of a cycle it is necessary to download the G-Code and order files of the parts in the active schedule to the cell controllers. So just selecting this item all the necessary files will be transferred. Until the transfer will be completed mouse arrow will turn to clock. It can take about a few seconds to complete the operation.

A.3.2.7 Dispatcher/Execute

This will start the physical execution of a production cycle which belongs to the active schedule. During the cycle the operator can watch the current status of the system from the window and observe the delays caused by transportation.

A.3.2.8 Connections/Open

You must open the connections of the host controller to listening for any request coming from a cell controller. Before this Trumtel.exe of version 1.1 or higher should be run. When a connection request from a cell controller is received

and accepted by the host controller a message box will appear.

A.3.2.9 Connections/Close

You can close the connections to listening using this item. If any connection has been established with the cell controllers they will break and relevant message boxes will be displayed.

A.3.2.10 Options/Directories

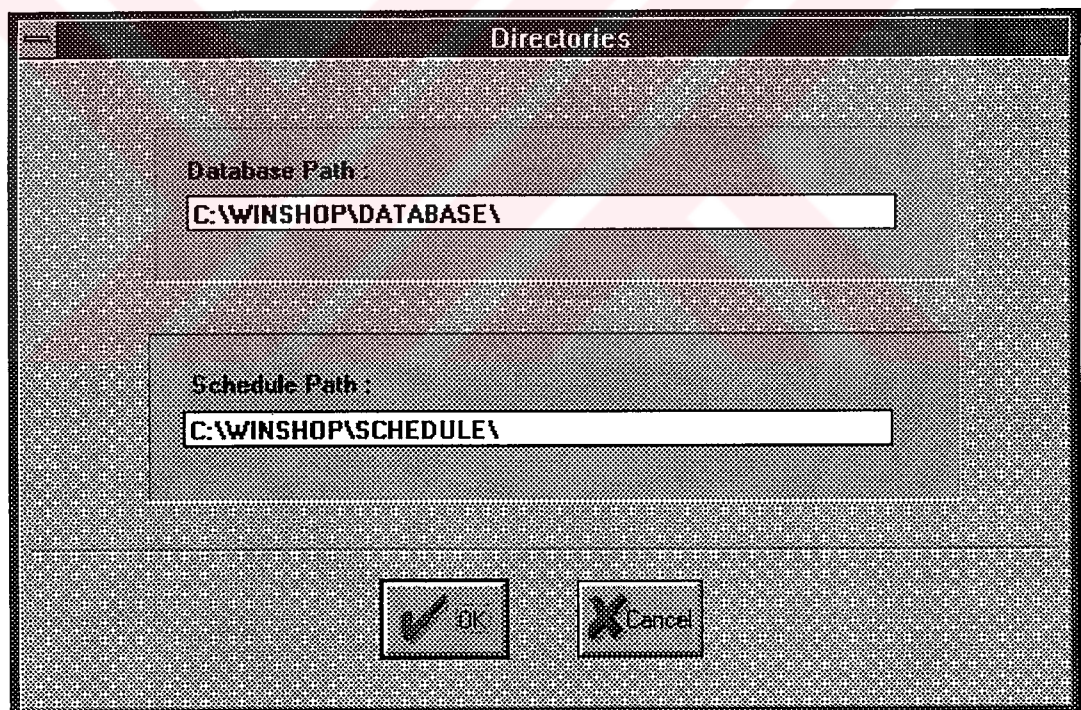


Figure A.7 Directories Dialog Box.

Using this dialog box, the directories that the program will search for the database files like G-Code or order files can be modified as shown in Figure A.7.

A.3.2.11 Statistics/Op-times

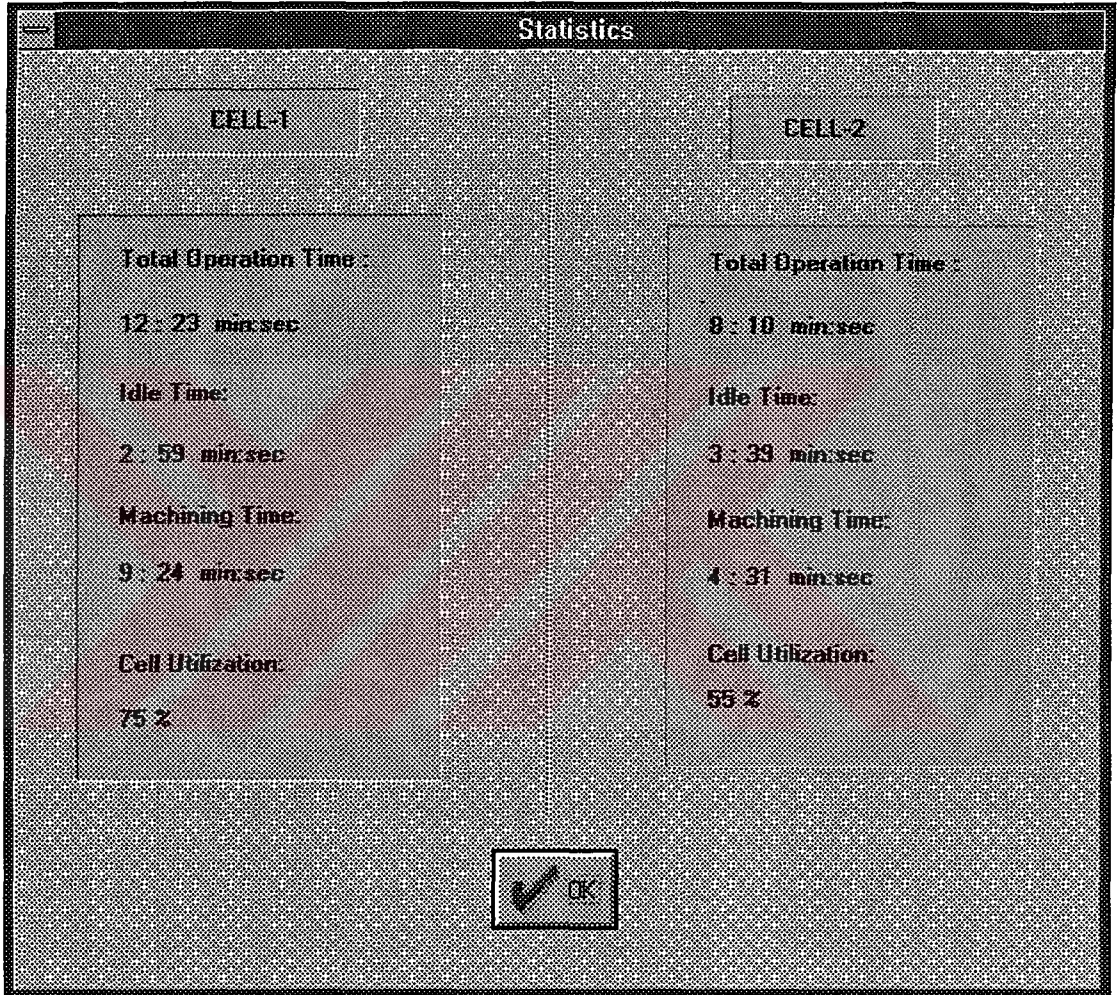


Figure A.8 Statistics Dialog Box

After a cycle is completed, it is possible to get statistical information about the completed cycle. Total operation time, idle time, machining time and cell

utilization can be get for each cell. The view is in Figure A.8.

A.3.2.12 About

You can get information about the programmer and the supervisor.

A.4 Cell Controller

A.4.1 Cell Controller Main Window

The Main Window of the cell controller is for monitoring the device status of the cell. When the circle near the device is green this indicates that device is idle when the circle is red device is busy.

The cell controller's main window is shown in Figure A.9.

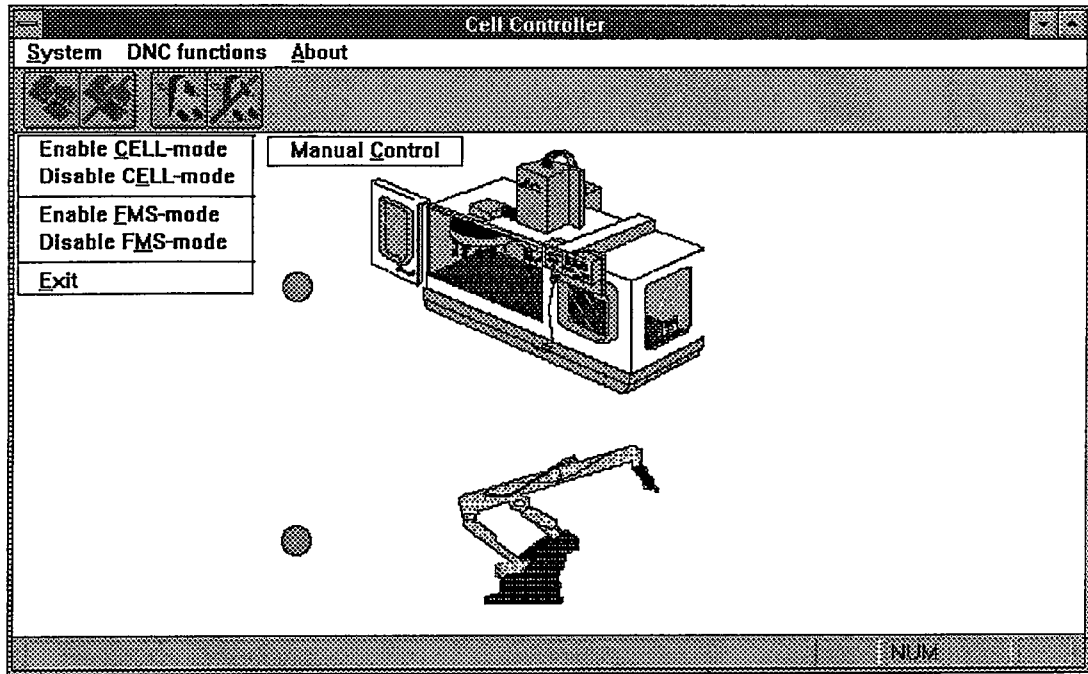


Figure A.9 Cell Controller Main Window

A.4.2 Menu Bar

A.4.2.1 System/Enable Cell Mode

You can select this when you want to run the cell devices using manual control only. When you this the cell will not try to connect to host controller.

A.4.2.2 System/Disable Cell Mode

This will disable the system from cell mode.

A.4.2.3 System/Enable FMS Mode

This will connect the cell to the host controller as a client and wait for the commands coming from the host controller. It is also possible to use manual control at this mode.

A.4.2.4 System/Disable FMS Mode

This will disable the system from FMS mode and close the connection with the host controller.

A.4.2.5 System/Exit

Closes the application and returns to program manager.

A.4.2.6 DNC Functions/Manual Control

This menu will activate the dialog box that you can control the cell device functions manually. As shown in Figure A.10 it is possible to control functions of CNC, robot and conveyor.

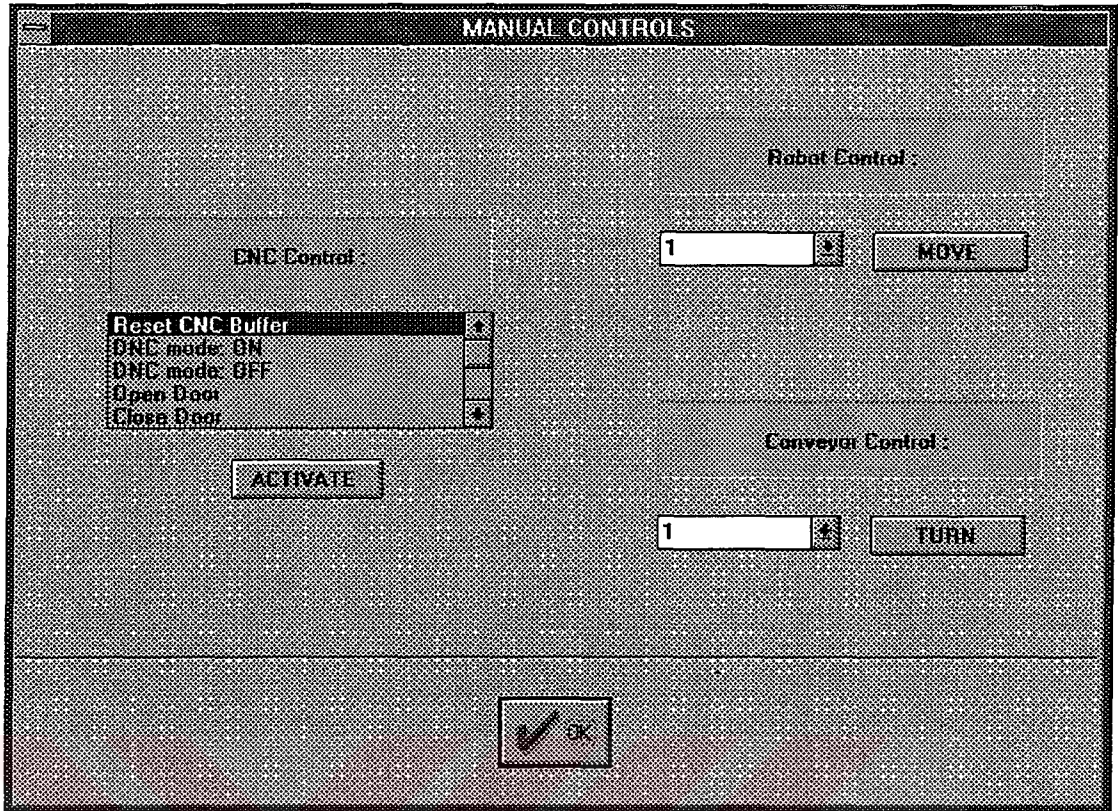


Figure A.10 Manual Control Dialog Box

In order to open the DNC mode with the CNC controller CNC buffer should be reset and Set DNC mode on should be selected.

A.4.2.7 About

You can get information about the programmer and supervisor.

APPENDIX B

G-CODE FILES OF TEST RUNS

B.1. First Test Run

```
[BILLET X30 Z45  
M62  
N2 G98 G21 G97 S2000 M03  
N3 G28 U0 W0  
N4 M06 T01  
N5 G00 X32 Z2  
N6 G01 Z0 F40  
N7 G01 X-1 F50  
N8 G00 X32 Z1  
N9 G00 Z-0.000  
N10 G00 X31.000  
N11 G01 X26.000  
N12 G01 Z-31.500 F150  
N13 G00 X31.000 Z-30.500  
N14 G00 Z-0.000  
N15 G00 X27.000  
N16 G01 X22.000  
N17 G01 Z-20.854 F150  
N18 G00 X27.000 Z-19.854  
N19 G00 Z-0.000  
N20 G00 X23.000  
N21 G01 X18.000  
N22 G01 Z-19.627 F150  
N23 G00 X23.000 Z-18.627  
N24 G00 Z-0.000  
N25 G00 X19.000  
N26 G01 X14.000  
N27 G01 Z-13.373 F150  
N28 G00 X19.000 Z-12.373  
N29 G00 Z-0.000  
N30 G00 X15.000  
N31 G01 X10.000  
N32 G01 Z-12.146 F150  
N33 G00 X15.000 Z-11.146
```

Figure B.1 G-Code File for Lathe Process of Part-1


```
N34 G00 X32.000
N35 G00 X32.000
N36 G00 X32.000 Z5
N37 G00 X8.000 Z1
N38 G01 Z0 F30
N39 G01 Z-10
N40 G02 X16 Z-14 I4.000
N41 G01 Z-20
N42 G01 X16.000 F50
N43 G03 X24.000 Z-24.000 K-4.000
N44 G01 Z-30
N45 G01 X24.000 F50
N46 G01 X28.000 Z-34.000 F50
M05
G28 U0 W0
M64
```

Figure B.1 (Continued)

```
BILLET X30 Z50
M62
N2 G98 G21 G97 S2000 M03
N3 G28 U0 W0
N4 M06 T01
N5 G00 X32 Z2
N6 G01 Z0 F40
N7 G01 X-1 F50
N8 G00 X32 Z1
N9 G00 Z-0.000
N10 G00 X32.000
N11 G01 X27.000
N12 G01 Z-30.000 F150
N13 G00 X32.000 Z-29.000
N14 G00 X32.000
N15 G00 Z-13.000
N16 G00 X28.000
N17 G01 X23.000
N18 G01 Z-22.500 F150
N19 G00 X28.000 Z-21.500
N20 G00 X32.000
N21 G00 X32.000
N22 G00 X32.000 Z5
N23 G00 X23.000 Z1
N24 G01 Z0 F50
N25 G01 X25 Z-2.000
N26 G01 Z-10
N27 G01 X25.000 F50
N28 G01 X20.000 Z-15.000 F50
N29 G01 Z-20
N30 G01 X20.000 F50
N31 G01 X25.000 Z-25.000 F50
```

Figure B.2 G-Code File for Lathe Process of Part-2

```
N32 G01 X25.000 F50
N33 G01 Z-30
N34 G28 U0 W0
N42 M05
M64
```

Figure B.2 (Continued)

```
[BILLET X30 Z50
M62
N2 G98 G21 G97 S2000 M03
N3 G28 U0 W0
N4 M06 T01
N5 G00 X32 Z2
N6 G01 Z0 F40
N7 G01 X-1 F50
N8 G00 X32 Z1
N9 G00 Z-0.000
N10 G00 X32.000
N11 G01 X27.000
N12 G01 Z-35.000 F150
N13 G00 X32.000 Z-34.000
N14 G00 Z-0.000
N15 G00 X28.000
N16 G01 X23.000
N17 G01 Z-29.500 F150
N18 G00 X28.000 Z-28.500
N19 G00 Z-0.000
N20 G00 X24.000
N21 G01 X19.000
N22 G01 Z-29.500 F150
N23 G00 X24.000 Z-28.500
N24 G00 Z-0.000
N25 G00 X21.608
N26 G01 X16.608
N27 G01 Z-29.500 F150
N28 G00 X21.608 Z-28.500
N29 G00 Z-0.000
N30 G00 X20.000
N31 G01 X15.000
N32 G01 Z-29.500 F150
N33 G00 X20.000 Z-28.500
N34 G00 Z-0.000
N35 G00 X17.000
N36 G01 X12.000
N37 G01 Z-5.000 F150
N38 G00 X17.000 Z-4.000
N39 G00 X32.000
N40 G00 X32.000
```

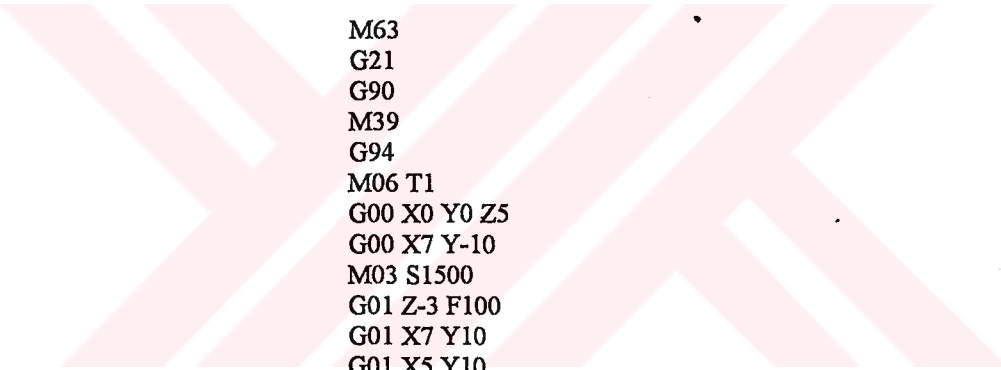
Figure B.3 G-Code File for Lathe Process of Part-3

```

N41 G00 X32.000 Z5
N42 G00 X9.500 Z1
N43 G01 Z0 F30
N44 G01 X10 Z-0.500
N45 G01 Z-5
N46 G01 X12.000 F50
N47 G01 X13 Z-6
N48 G01 Z-8
N49 G02 X13.000 Z-12.000 I3.464 K-2.000 F50
N50 G01 Z-15
N51 G03 X13.000 Z-21.000 I-5.196 K-3.000 F50
N52 G01 Z-24
N53 G02 X13.000 Z-28.000 I3.464 K-2.000 F50
N54 G01 Z-30
N55 G01 X24.000 F50
N56 G01 X25 Z-31
N57 G01 Z-35
N58 G28 U0 W0
N66 M05
M64

```

Figure B.3 (Continued)



```

M63
G21
G90
M39
G94
M06 T1
G00 X0 Y0 Z5
G00 X7 Y-10
M03 S1500
G01 Z-3 F100
G01 X7 Y10
G01 X5 Y10
G01 X5 Y-10
G00 X7 Y-10
G01 Z-6
G01 X7 Y10
G01 X5 Y10
G01 X5 Y-10
G01 X7 Y-10
G01 Z-9
G01 X7 Y10
G01 X5 Y10
G01 X5 Y-10
G01 X7 Y-10
G00 Z20
M05
G28 X0 Y0
M65

```

Figure B.4 G-Code File for Milling Process of Part-1

```
M63  
G21  
G90  
G94  
M06 T3  
M03 S1500  
G00 X-5 Y0 Z3  
G83 G99 X-5 Y0 Z-2 Q1 R2 F100  
X5  
G80  
M05  
M06 T4  
M03 S1500  
G00 X-5 Y0 Z3  
G83 G99 X-5 Y0 Z-10 Q3 R2 F100  
X5  
G80  
G98  
M05  
G28 X0 Y0  
M65
```

Figure B.5 G-Code File for Milling Process of Part-2

B.2 Second Test Run

```
[BILLET X30 Z45  
M62  
N2 G98 G21 G97 S2000 M03  
N3 G28 U0 W0  
N4 M06 T01  
N5 G00 X32 Z2  
N6 G01 Z0 F40  
N7 G01 X-1 F50  
N8 G00 X32 Z1  
N9 G00 Z-0.000  
N10 G00 X31.000  
N11 G01 X26.000  
N12 G01 Z-31.500 F100  
N13 G00 X31.000 Z-30.500  
N14 G00 Z-0.000  
N15 G00 X27.000  
N16 G01 X22.000  
N17 G01 Z-20.854 F100  
N18 G00 X27.000 Z-19.854  
N19 G00 Z-0.000  
N20 G00 X23.000  
N21 G01 X18.000  
N22 G01 Z-19.627 F100  
N23 G00 X23.000 Z-18.627  
N24 G00 Z-0.000  
N25 G00 X19.000  
N26 G01 X14.000  
N27 G01 Z-13.373 F100  
N28 G00 X19.000 Z-12.373  
N29 G00 Z-0.000  
N30 G00 X15.000  
N31 G01 X10.000  
N32 G01 Z-12.146 F100  
N33 G00 X15.000 Z-11.146  
N34 G00 X32.000  
N35 G00 X32.000  
N36 G00 X32.000 Z5  
N37 G00 X8.000 Z1  
N38 G01 Z0 F30  
N39 G01 Z-10  
N40 G02 X16 Z-14 I4.000  
N41 G01 Z-20
```

Figure B.6 Modified G-Code File for Lathe Process of Part-1

```
N42 G01 X16.000 F50
N43 G03 X24.000 Z-24.000 K-4.000
N44 G01 Z-30
N45 G01 X24.000 F50
N46 G01 X28.000 Z-34.000 F50
M05
G28 U0 W0
M64
```

Figure B.6 (Continued)

```
[BILLET X30 Z50
M62
N2 G98 G21 G97 S2000 M03
N3 G28 U0 W0
N4 M06 T01
N5 G00 X32 Z2
N6 G01 Z0 F40
N7 G01 X-1 F50
N8 G00 X32 Z1
N9 G00 Z-0.000
N10 G00 X32.000
N11 G01 X27.000
N12 G01 Z-30.000 F100
N13 G00 X32.000 Z-29.000
N14 G00 X32.000
N15 G00 Z-13.000
N16 G00 X28.000
N17 G01 X23.000
N18 G01 Z-22.500 F100
N19 G00 X28.000 Z-21.500
N20 G00 X32.000
N21 G00 X32.000
N22 G00 X32.000 Z5
N23 G00 X23.000 Z1
N24 G01 Z0 F50
N25 G01 X25 Z-2.000
N26 G01 Z-10
N27 G01 X25.000 F50
N28 G01 X20.000 Z-15.000 F50
N29 G01 Z-20
N30 G01 X20.000 F50
N31 G01 X25.000 Z-25.000 F50
N32 G01 X25.000 F50
N33 G01 Z-30
N34 G28 U0 W0
N42 M05
M64
```

Figure B.7 Modified G-Code File for Lathe Process of Part-2

```
[BILLET X30 Z50
M62
N2 G98 G21 G97 S2000 M03
N3 G28 U0 W0
N4 M06 T01
N5 G00 X32 Z2
N6 G01 Z0 F40
N7 G01 X-1 F50
N8 G00 X32 Z1
N9 G00 Z-0.000
N10 G00 X32.000
N11 G01 X27.000
N12 G01 Z-35.000 F100
N13 G00 X32.000 Z-34.000
N14 G00 Z-0.000
N15 G00 X28.000
N16 G01 X23.000
N17 G01 Z-29.500 F100
N18 G00 X28.000 Z-28.500
N19 G00 Z-0.000
N20 G00 X24.000
N21 G01 X19.000
N22 G01 Z-29.500 F100
N23 G00 X24.000 Z-28.500
N24 G00 Z-0.000
N25 G00 X21.608
N26 G01 X16.608
N27 G01 Z-29.500 F100
N28 G00 X21.608 Z-28.500
N29 G00 Z-0.000
N30 G00 X20.000
N31 G01 X15.000
N32 G01 Z-29.500 F100
N33 G00 X20.000 Z-28.500
N34 G00 Z-0.000
N35 G00 X17.000
N36 G01 X12.000
N37 G01 Z-5.000 F100
N38 G00 X17.000 Z-4.000
N39 G00 X32.000
N40 G00 X32.000
N41 G00 X32.000 Z5
N42 G00 X9.500 Z1
N43 G01 Z0 F30
N44 G01 X10 Z-0.500
N45 G01 Z-5
N46 G01 X12.000 F50
N47 G01 X13 Z-6
N48 G01 Z-8
N49 G02 X13.000 Z-12.000 I3.464 K-2.000 F50
N50 G01 Z-15
N51 G03 X13.000 Z-21.000 I-5.196 K-3.000 F50
```

Figure B.8 Modified G-Code File for Lathe Process of Part-3

```
N52 G01 Z-24
N53 G02 X13.000 Z-28.000 I3.464 K-2.000 F50
N54 G01 Z-30
N55 G01 X24.000 F50
N56 G01 X25 Z-31
N57 G01 Z-35
N58 G28 U0 W0
N66 M05
M64
```

Figure B.8 (Continued)

```
M63
G21
G90
M39
G94
M06 T1
G00 X0 Y0 Z5
G00 X7 Y-10
M03 S1500
G01 Z-3 F70
G01 X7 Y10
G01 X5 Y10
G01 X5 Y-10
G00 X7 Y-10
G01 Z-6
G01 X7 Y10
G01 X5 Y10
G01 X5 Y-10
G01 X7 Y-10
G01 Z-9
G01 X7 Y10
G01 X5 Y10
G01 X5 Y-10
G01 X7 Y-10
G00 Z20
M05
G28 X0 Y0
M65
```

Figure B.9 Modified G-Code file for Mill Process of Part-1


```
M63
G21
G90
G94
M06 T3
M03 S1500
G00 X-5 Y0 Z3
G83 G99 X-5 Y0 Z-2 Q1 R2 F70
X5
G80
M05
M06 T4
M03 S1500
G00 X-5 Y0 Z3
G83 G99 X-5 Y0 Z-10 Q3 R2 F70
X5
G80
G98
M05
G28 X0 Y0
M65
```

Figure B.10 Modified G-Code File for Mill Process of Part-2

APPENDIX C

SAMPLE CLASSES

```
class dispatch {
public:
dispatch (TWindow* wnd,HWND Hwin,cletdrv* ethdrv);
~dispatch();
protected:
char buf[100];
int i,cell1num,cell2num;
TWindow* wind;
int iipro,jjpro;
public:
void startcycle(void);
void cycle(void);
void cycle2(void);
void concycle(void);
void readschedulefile(char *file);
void prodowncell1(void);
void prodowncell2(void);
void startdownload(void);
void schcell1(void);
void updatedb(void);
cellman *cell1;
cellman *cell2;
monitor *mon;
cletdrv* etherdrv;
cnvman* conman;
unsigned int mytime;
unsigned char ord1,ord2;
int ipro,jpro;
HWND HMain;
shopApp* myapp;
};
```

Figure C.1 Header File of Dispatcher Class of Host Controller

```

class cnvman {
public:
cnvman (monitor* mo,HWND HWin,cletdrv* etdrv);
~cnvman();
void findemptycell1(void);
void findemptycell2(void);
void finditemcell1(void);
void finditemcell2(void);
LRESULT queueevent(WPARAM w, LPARAM l);
void updatecnv(void);
void cycle(void);
void indexok(void);
void sendcmessage(void);
void clearbusy(void);
void click(void);
monitor *mont;
cletdrv* eth;
HWND hwin;
unsigned char req,cell,resreq,rescell,busy;
int turnno;
long int item,resitem;
TQueue<long int> *partque, *c1waitpartque,
*c2waitpartque;
TQueue<unsigned char> *cellque, *reqque, *c1waitcellque,
*c2waitcellque, *c1waitreqque, *c2waitreqque;
};

```

Figure C.2 Header File of Conveyor Manager Class of Host Controller

```

class cncdrv{
public:
cncdrv(TWindow* parent, HWND HWind,monitor*
Bmoni);
~cncdrv(void);
int slc;
void loadprg(char *kr);
void resetCNCbuffer(void);
void opendoor(void);
void closedoor(void);
void openchuck(void);
void closechuck(void);
void setDNC(void);
void unsetDNC(void);
void downprg(char *file);
void send(int port,unsigned char cr);
void notify();
void down(void);
void close(void);
void onanevent(void);
fstream *prog;
HWND hwn;
monitor* Bmon;
TWindow* prnt;
notdialog *ndialog;
int CPort,linecount;
unsigned char buf[30];
DCB CommData;
cellbApp *app;
};

```

Figure C.3 Header File of CNC Driver Class of Cell Controller

```

class dispatcher{
public:
dispatcher (TWindow* parent, HWND HWin,SOCKET
sc,client *clie);
dispatcher(TWindow* parent,HWND HWin);
~ dispatcher ();
void readopfile(char *filename);
void opcomplete(void);
void executeopfile(void);
void prnocyc(void);
void prdropn(void);
void prdrclد(void);
void prckopn(void);
void prckclد(void);
void prescheck(void);
robotdrv *rdrv;
cncdrv *cdrv;
convdrv *cnvdrv;
monitor *Bmonitor;
optable *oplist;
HWND HWind;
SOCKET sck;
client *cli;
protected:
int j, totop,slc;
int errcode;
char message1[100];
};

```

Figure C.4 Header File of Dispatcher Class of Cell Controller

**T.C. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ**

APPENDIX D

TEST RUN OF BUILT-IN SCHEDULER

In order to test the built-in schedule a batch with 13 parts is given. Relevant machining times and part-types are in Table D.1

Table D.1 Test Batch for Built-in Scheduler

Operation no	Machine 1 Time	Machine 2 Time	Type
1	6	3	M12
2	2	9	M12
3	4	3	M12
4	1	8	M12
5	7	1	M12
6	4	5	M12
7	7	6	M12
8	8	2	M21
9	5	8	M21
10	2	7	M21
11	8	3	M21
12	0	3	M2
13	5	0	M1

The result of Step-2 for Types M12 and M21 are in Table D.2 and D.3 respectively

Table D.2 Results of Step-2 for Type M12

Operation no	Time	Machine no
1	3	2
2	2	1
3	3	2
4	1	1
5	1	2
6	4	1
7	6	2

Table D.3 Results of Step-2 for Type M21

Operation no	Time	Machine no
8	2	2
9	5	1
10	2	1
11	3	2

Sorting in step-3 in increasing value of processing time results in Table D.4 and Table D.5.

Table D.4 Results of Step-3 for Type M12

Operation no	Time	Machine no
4	1	1
5	1	2
2	2	1
3	3	2
1	3	2
6	4	1
7	6	2

Table D.5 Results of Step-4 for Type M21

Operation no	Time	Machine no
8	2	2
10	2	1
11	3	2
9	5	1

Step 4 produces the following optimal schedule for flow shops of types M12 and M21:

M12: (4,2,6,7,1,3,5)

M21: (10,9,11,8)

Applying Kusiak's modification for job-shop case and inserting the single machine types into right places the final schedule is:

Machine M1: (4,2,6,7,1,3,5,13, 10,9,11,8)

Machine M2: (10,9,11,8,12, 4,2,6,7,1,3,5)

Generated schedule file for the batch is in Figure D.1

cell1 12
OP11 42456712.ord
OP12 25324612.ord
OP13 65842512.ord
OP14 75125112.ord
OP15 36128312.ord
OP16 16138712.ord
OP17 56746312.ord
OP18 13465701.ord
OP19 10927421.ord
OP110 91635321.ord
OP111 11856721.ord
OP112 83426421.ord
cell2 12 -
OP21 10927421.ord
OP22 91635321.ord
OP23 11856721.ord
OP24 83426421.ord
OP25 12343502.ord
OP26 42456712.ord
OP27 25324612.ord
OP28 65842512.ord
OP29 75125112.ord
OP210 36128312.ord
OP211 16138712.ord
OP212 56746312.ord

Figure D.1 Generated Schedule Order