FACTORY WIDE AUTOMATION

by

E. Allan Mertin

B.Sc.(M.E.), The University of Manitoba, 1986

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

IN

THE FACULTY OF GRADUATE STUDIES

Department of Mechanical Engineering

We accept this thesis as conforming
to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

August 1996

Abstract

The objective of this thesis is to develop and test the ability of a simple, distributed control architecture to synchronize multiple machine processes. The work reported in this thesis is based on the extension of the UBC control architecture to a distributed system.

The resulting network has been both simulated and tested using an NC lathe and process control element. The results are encouraging, it is believed that the system developed has significant practical promise.

Table of Contents

List of Tables

List of Figures

Nomenclature

A:	Filter Lead Constant in the Z-Domain
A:	Area of Cut $[mm^2]$
a:	Depth of Cut [mm]
B:	Filter Lag Constant in the Z-Domain
Fa:	Axial Thrust Force [N]
Fr:	Radial Thrust Force [N]
Ft:	Total Thrust Force [N]
f:	Frequency
G1: G2: G3:	Digital Filter Parameters
He:	Equivalent Chip Thickness
Jeq:	Equivalent Motor Inertia [N.m.s^2]
Ka:	Servo Amp Current Gain [A/V]
Ke:	Encoder Gain [pulses/rad]
Kda:	D/A Converter Gain [V/Level]
Kp:	Filter Gain
Kt:	Motor Torque Constant [Nm/A]
Le:	Engaged Cutting Edge [mm]
s:	Feed Rate [mm/rev]
T:	Slave Sampling Time
tm:	Motor Mechanical Time Constant [sec]

te:	Motor Electrical Time Constant [sec]
V:	Voltage [Volts]
Ψ:	Tool Approach Angle [rad]
Ψe:	Effective Angle [rad]
ε:	Angle of Obliquity [rad]

Acknowledgments

The author wishes to express sincere gratitude to Dr. Ian Yellowley. His guidance and experience was crucial during the research; the encouragement and his idea for this thesis have been inspirational. The author would also like to recognize and thank Mr. Rudi Seethaler for his assistance with laboratory equipment and Mr. Gord Wright for his assistance with the development and fabrication of circuits. The assistance and friendship of the authors colleagues is also appreciated. The author as a member of the Canadian Forces acknowledges the support provided by the Canadian Forces Post Graduate Training Plan.

Finally, the author would like to thank his family for their unyielding support and wishes to dedicate this work to the memory of AB and VM.

Chapter 1

Distributed Process Control

1.1 Introduction

Distributed process control is the technique of using a master computer to direct overall control of a large process through subordinate computers. The advantages of distributed control are the division of processing resources and improved reliability and system failure tolerance. The application of distributed process control to a manufacturing environment has resulted in improved performance, but has primarily been used by larger production facilities. This is due to the economies of scale which are required for the significant investment in equipment and personnel. Thus, there appears to be a necessity to provide the ability for economical distributed control in the manufacturing environment.

Computer aided manufacturing (CAM) has resulted in significant progress in the automation of machine tools, which have progressed from manual control to digital computer control, commonly known as Computer Numerical Control (CNC). This advance from manual to digital control has resulted in improved quality, speed, and reliability, and the ability to control more complex operations.

The next logical step in automating manufacturing is to provide the CNC machine tools or other intelligent devices in a process the opportunity to further improve their performance. This can be achieved by increasing individual computing capability and improving planning through the use of a distributed system. Past efforts to provide distributed control in the manufacturing environment have been based on integrating all of the functions in an enterprise. The cost of implementing and supporting such a system has proven prohibitive.

In this chapter distributed control is introduced. Then, the application of computers to manufacturing is discussed and the integration of distributed control to an automated manufacturing environment is presented. Finally, the application of a distributed control system using a UBC designed controller is proposed.

1.2 Historical Development of Distributed Control

Control system development has been the result of industrial demand and technological advances. The industrial demand stems from an ever increasing complexity in the processes which are to be controlled and the increasing costs, such as material, labour and energy. This has driven the desire to have a totally integrated plant management system.

The implementation of device control really began with governors and mechanical controllers which were installed on the equipment to be controlled. This level of control provided local information such as set points, readouts and output to the operator. Coordination of the process was the responsibility of the operators who relied on trip switches and calling over to their colleagues for a verbal confirmation of the system's performance. This level of control worked until the complexity and size of the process to be managed grew beyond the ability of the operators to optimize it [21].

The next improvement was the development of centralized control based on pneumatics. A pneumatic system can be broadly defined as a grouping of pressure regulators which react to temperature, relative humidity or another pressure. A simple example of such a system is temperature control using a pneumatic thermostat. As the temperature changes, a regulator in the thermostat varies the signal which is piped to a control device. The changing pneumatic signal is measured by the control device which reports, and if necessary, causes corrective action; for example turning on or off a heater or an air conditioner. For larger applications involving numerous measurements, the pneumatic signals would be transmitted to a control room. At the control room, as in the example above, all of the information would be processed and if required an instruction returned to the source, or elsewhere. This improvement of a centralized system allowed all of the operators to be co-located. Thus, it allowed improved coordination and control which lead to higher productivity, reduced losses and improved safety [11,15,21].

With the advancement of electronics and computers the architecture of centralized control systems shifted from pneumatics to electronics. The desire to shift from pneumatic to digital control was driven by the increasing complexity of systems, which computers could better control, and the reduced distributed line cost, since wire is cheaper than pneumatic tubes. Use of electronic systems also eliminated the lag inherent with pneumatic systems.

This review of the application of computer based control systems will begin with a closer look at the devices which are to be controlled. The majority of naturally occurring phenomena which require control are continuous and analog. Their inputs and outputs vary over a continuous range. This is exemplified by a sink faucet which can be either off or on to a varying degree, from a slight trickle to fully open. The other type of device to be controlled is binary. Binary devices have two distinct states, on or off, such as a motor governed to operate at a set speed.

The application of computer control to analog and binary devices required an interface to allow communication between a computer and the devices to be controlled. Communication with analog devices became possible with the advent of Digital - to - Analog (D/A) and Analog - to - Digital (A/D) converters. These devices allow conversion between a computer's binary digital word and an analog device's unique Direct Current (dc) voltage. By calibrating this system, it is possible to bi-directionaly communicate a digital instruction or a dc voltage through a converter, which can produce an interpretable instruction, such as a motor speed setting. Similarly, communication with binary devices became possible with the advent of the Programmable Logic Controller (PLC). This device is a computer based programmable switch which allows on and off instructions to be actioned [16,21].

As computers, PLC's, A/D and D/A converters became economically available, they were employed to acquire data, optimize processes and display status for supervisory purposes. There was little or no communication between the controllers. Later these computers were connected together, providing distributed control. The architecture of the distributed computer control systems evolved in two different types, commonly known as the hybrid control system and the central control computer system [15,21].

The hybrid control system used discrete hardware and a central supervisory computer, which provided the control function. The challenge in this system was to interface the numerous different types of hardware which made up the system. The architecture of a hybrid control system is shown in Figure 1.1.

In contrast, the central computer control system provided all functions from a high performance central computer. This system, due to the high cost of its central computer, was only accepted by the electrical utility industry; the hybrid system became the norm elsewhere. The architecture of the central computer control system, which normally has a primary and backup computer providing central computing, is shown in Figure 1.2.

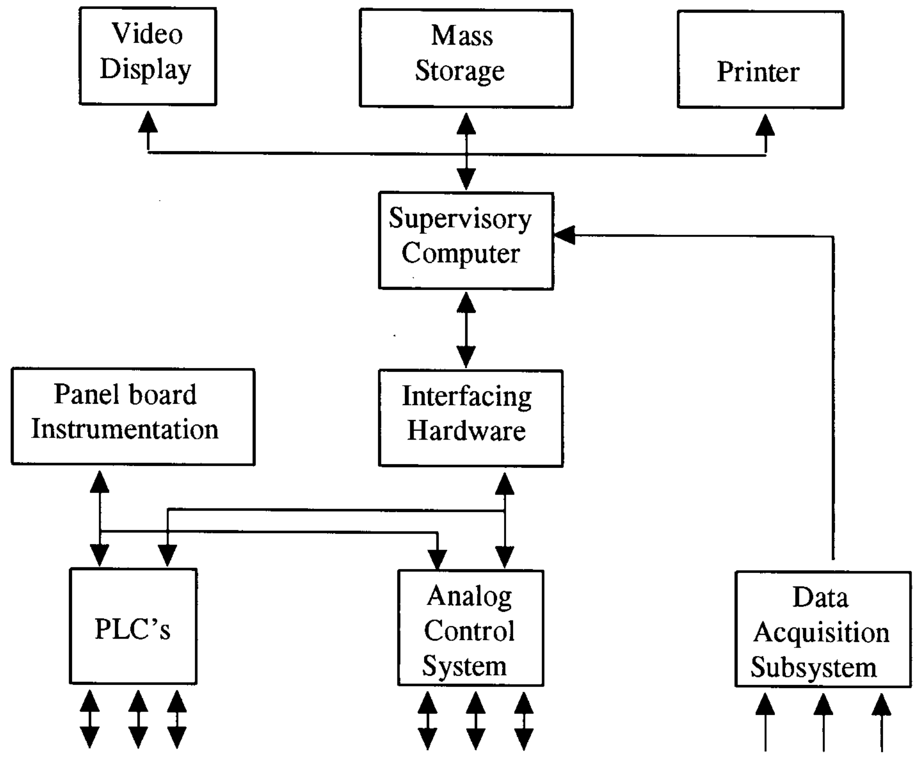

Figure 1.1. Hybrid Control System Architecture

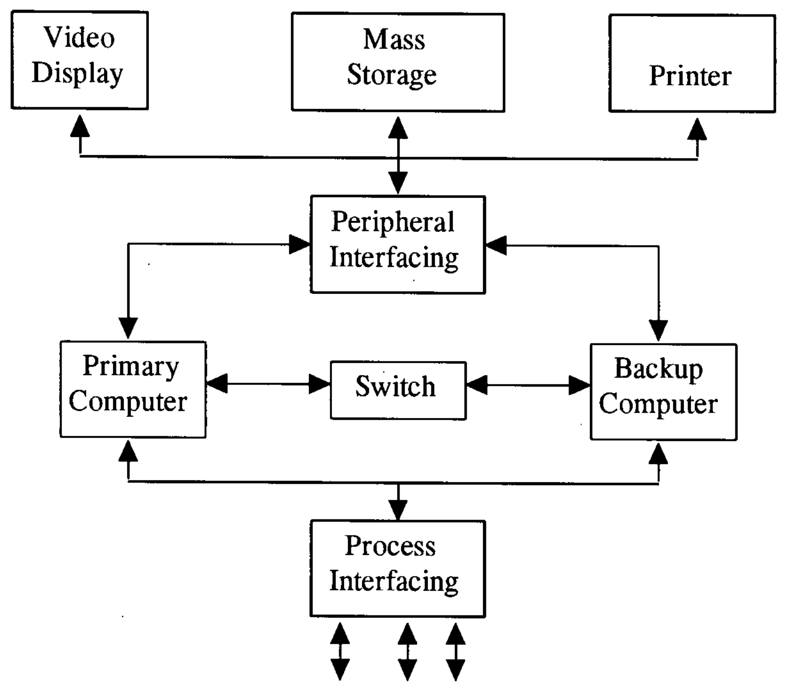

Figure 1.2. Central Computer Control System Architecture

By the 1970's, due to the limitations of both systems, a new architecture known as the generalized distributed control system became the norm. This architecture divided the functions of the control system into an interface to the controllers source, a human interface, and a computing function. This architecture is shown in Figure 1.3 [15,21,30].

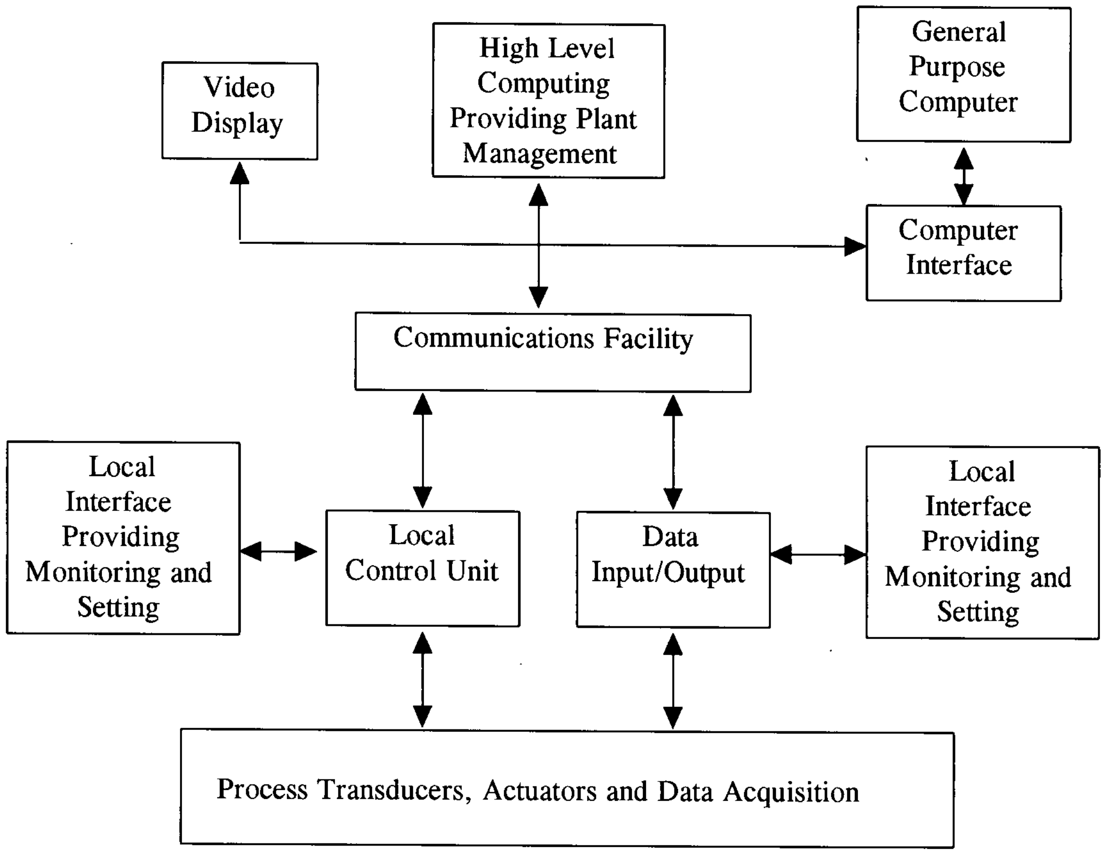
Video Display
High Level Computing Providing Plant Management
General Purpose Computer
Computer Interface
Communications Facility
Local Interface Providing Monitoring and Setting
Local Control Unit
Data Input/Output
Local Interface Providing Monitoring and Setting
Process Transducers, Actuators and Data Acquisition

Figure 1.3. Generalized Distributed Control System Architecture

1.3 CNC Machines

The availability of economical electronic components and computers as discussed above lead to their application in controlling machine tools and in the management of manufacturing. Concurrent with the distributed control advances, machine tools became automated in the mid 1950's when MIT developed the first

Numerical Control (NC) machine in 1952. The official Electronic Industrial Association definition for an NC machine is "a system in which actions are controlled by direct insertion of numerical data at some point. The system must automatically interpret at least some of portion of the data" [1].

An NC machine consists of three distinct parts: data input, control unit and a machine tool. The data input is a set of instructions which tell the machine what to do. This includes axis movement, spindle speed, tool change, feed rate, coolant start, start and stop. The data input is in the form of a tape, floppy disk or other similar means which offers portability. The option also exists for manual input by the operator known as Manual Data Input (MDI). The control unit is a processor on the machine which converts the instructions of the data input to a set of interactions for the machine. The machine tool is the controlled equipment, whose servo motors action the instructions received from the control unit.

A major improvement in machine tool control was achieved in the early 1970's with the addition of a microcomputer to the control system. The microcomputer provided enhanced programming, memory and built-in diagnostics. This lead to the name change to Computer Numerical Control (CNC) machine. A further improvement upon a stand-alone CNC machine was to include a main computer which can control and support several machines simultaneously. The support provided by the main computer includes an available data base of data input for various tasks, queuing, and monitoring of status. The machine tool, while busy on a job, can provide the task's status to the servicing computer, or service requests for additional data from the supporting computer. This system of supervisor computer data support to CNC machines is known as Direct Numerical Control (DNC). The addition of DNC allows for the improved efficiency of electronic support and the elimination of tapes or other portable data input. The expansion of DNC to include further distributed interaction leads to the topic of Computer Integrated Manufacturing [1,29].

1.4 Computer Integrated Manufacturing (CIM)

CIM is the integration of manufacturing and business activities in a distributed system. This is accomplished by allowing full communication between all manufacturing functions: marketing, engineering design, research and development (R & D), manufacturing, financial planning and personnel. It must be emphasized that a CIM system should allow Computer Aided design (CAD) and Computer Aided Manufacturing (CAM) information to be communicated in order to fully capitalize on the intent of CIM. CIM will then allow decision makers access to all data available to allow the best decisions to be made and provide the ability to effectively communicate those decisions. The CIM communication model is shown in Figure 1.4. Note that the arrows in the figure represent the ability to communicate and share data; they do not represent the lines of responsibility, which can be included as an additional feature in the system [1,15,25,29].

CIM uses controllers, sensors, computers and computer networks to completely control the manufacturing process. Its architecture is based on a communication system which provides for the interfacing of all computer systems. Management functions, which tend to be based on standardized systems, must be interfaced with proprietary systems prevalent in a manufacturing environment. This is normally accomplished by having a Local Area Network (LAN) service a given function. Each of these functional LANs are networked to the central system to allow sharing of information.

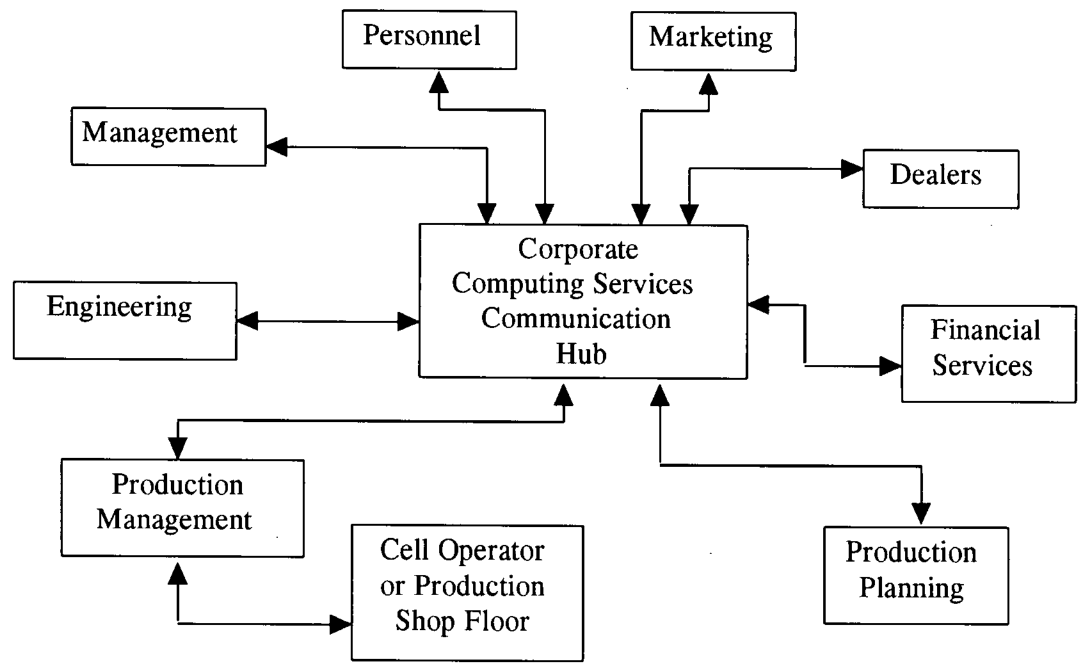

Figure 1.4. CIM Communication Model

Commercial CIM systems are generally based on the Open System Interconnection (OSI) communication model. The OSI model is not a standard but rather an architectural model, upon which various standards are based. The OSI model has seven layers; layers one through four are responsible for how the data is transmitted and layers five through seven are responsible for how the data is interpreted. The difficulty in implementing CIM is that the low level communication standards used by automation hardware are varied and often proprietary [15,25,30].

In 1980, in an effort to standardize communication requirements of equipment which operate on lower level communication standards, General Motors developed the Manufacturing Automation Protocol (MAP). MAP is based on the ISO seven layer communication model with the added functionality of the Manufacturing Message Formatted Standard (MMFS). MMFS defines the interaction between programmable controllers and CNC machines or robots. Specifically it provides a protocol for production information such as device status, program load, and job queuing. It also enhances reliability by detecting missing or duplicate data and initiating re-transmission to provide guaranteed message delivery. By the mid 1980's MAP had become a standard which was accepted both in North America and Europe as a method of extending CIM toward the production functions of an enterprise. In 1985 in an effort to build upon MAP, Boeing Aerospace developed a subset of MAP to enable office computers to communicate with the MAP. This protocol was called Technical Office Protocol (TOP). The combination of MAP and TOP provided for the integration of a CIM system as suggested by Figure 1.4 [1,21,25, 30].

MAP and TOP are actually separate protocols, and are based on different standards. MAP is based on IEEE 802.5 which provides the token passing standard, a deterministic protocol providing known maximum and minimum message passing times. Known message passing times are required for control of a manufacturing process. In contrast, TOP is based on the IEEE 802.3 standard, or Ethernet, which provides for equal opportunity non-deterministic access to the network by all users. Conflict caused by concurrent network use in TOP is managed by a Carrier Sense Multiple Access Collision Detection (CSMA/CD) algorithm. This algorithm manages the network by not allowing transmission unless the network is idle. The communication between systems using MAP and those using TOP is normally based on applications that use the services defined by MAP and TOP protocols [15,21,30].

1.5 Conclusions and Discussion

The manufacturing industry is progressing towards the use of CIM. It offers the efficiency of information availability throughout the manufacturing process. There has been significant effort to development of both MAP and TOP; however, they have not yet been fully developed. Thus, an industrial communication standard has not yet emerged. The access to information provided by TOP, save for access to lower level communication, has now become the norm. In contrast the distributed communication of manufacturing process data as provided by MAP, especially in smaller companies, is not common. CIM based on the combined use of MAP and TOP is presently in use, but does require significant investment in both equipment and technical personnel.

In the next section the Open Architecture Controller (OAC) is proposed as a model upon which distributed manufacturing process control, and information similar to that provided by MAP and TOP, can be based. Specifically, the UBC Controller, an OAC, is introduced and proposed as a control system which can provide a CIM capability similar to MAP and TOP, but with reduced investment requirements.

Chapter 2

The UBC Controller

2.1 Introduction

An open architecture controller is a well documented system which allows users an ease of extension and reconfiguration which is not possible with current CNC systems. In general terms such systems comprise distinct modules for the operator interface, the motion control tasks and the PLC tasks.

The UBC controller is an OAC CNC controller which has recently progressed from research to being licensed for commercial use. It is a multi-axis controller which has been specifically designed to coordinate a large number of axes, and to allow integration of process control with position and velocity control. The UBC controller's initial success is based on OAC's acceptance as a desirable alternative to embedded purpose-built controllers which may have proprietary restrictions.

2.2 Control Architecture

The UBC controller utilizes an STD 32 Bus computer (see Appendix 1). The controller consists of three component types: a master computer, a PLC, and a slave controller for each axis. This high level architecture of the UBC controller is shown in Figure 2.1 [24]. The master computer is a 486 IBM PC compatible processor which is capable of controlling up to 15 slave Input / Output (I/O) axis controllers. The slave controllers are I/O control processors. The master updates the position of each slave processor every 32 ms. The slave controllers are digital lead-lag controllers with a sampling time of 1.0 ms. They interpolate 32 times between every master position instruction. Thus, the architecture transfers the majority of the computational work load from the master to the slave controllers where both loop closing and second stage interpolation is performed. The PLC provides an interface for connection of the UBC controller to operator push buttons such as stop, start, etc. [32,35].

Each slave controller is connected to a state line. The state line is an open collector line which allows equal opportunity for all slave controllers to influence its state. Any slave controller can pull the state line low in response to an error or other criteria. When the state is high each slave controller proceeds as an ordinary lead-lag controller. However, when the state line is low, the slave controllers will not increment their position, but will continue with their closed loop position. Manipulation of the sequence of the state line's condition can also vary velocity, as per a binary signal such as 010101 (50%). This allows each slave controller to slow down or stop the system when it can not follow the programmed path at the set velocity [32,35].

In addition to the state line, each slave processor is also connected to a Synchronization Line, known as a sync line. The signal on the sync line is provided by one of the slave controllers, normally the x-axis. All of the slave controllers use the sync line for timing purposes.

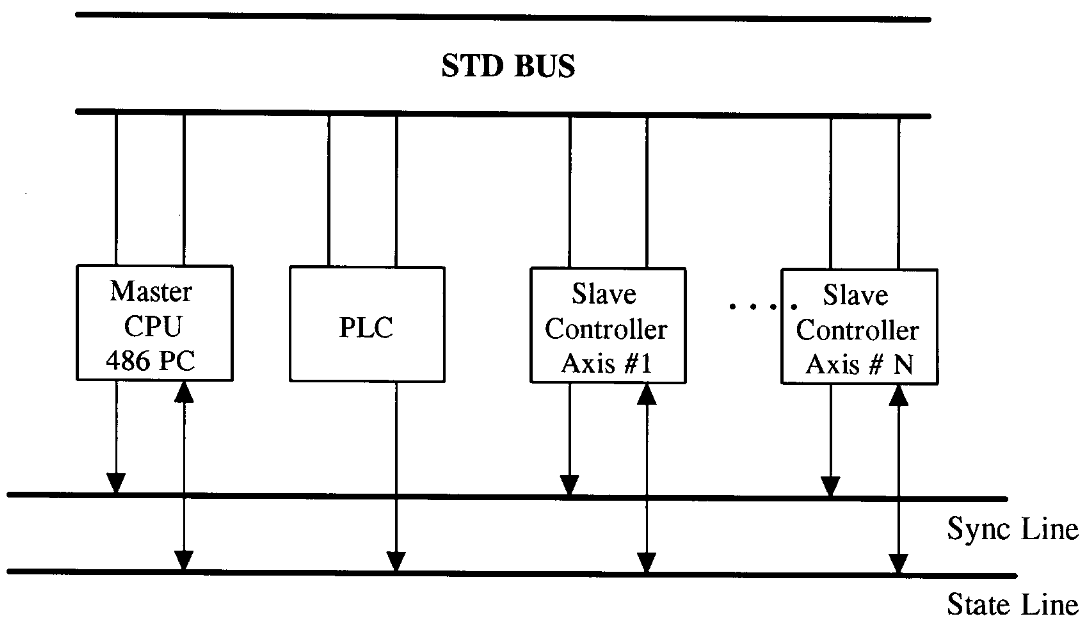

Figure 2.1. UBC Controller High Level Architecture

The slave controllers receive position increments at every sampling of the master computer. These position increments come from the master computer in the form of a first stage interpolation. The slave controllers perform a second stage interpolation. This procedure reduces the sampling time to 0.5 ms and allows velocity shaping in order to reduce acceleration levels between position increments. A further operation is performed to modulate velocity with the state line, to reduce acceleration levels below the saturation levels of the amplifiers. This modulated signal is fed into the lead-lag servo controller. A block diagram of a slave controller is shown in Figure 2.2 and of the PLC board in Figure 2.3 [32,35].

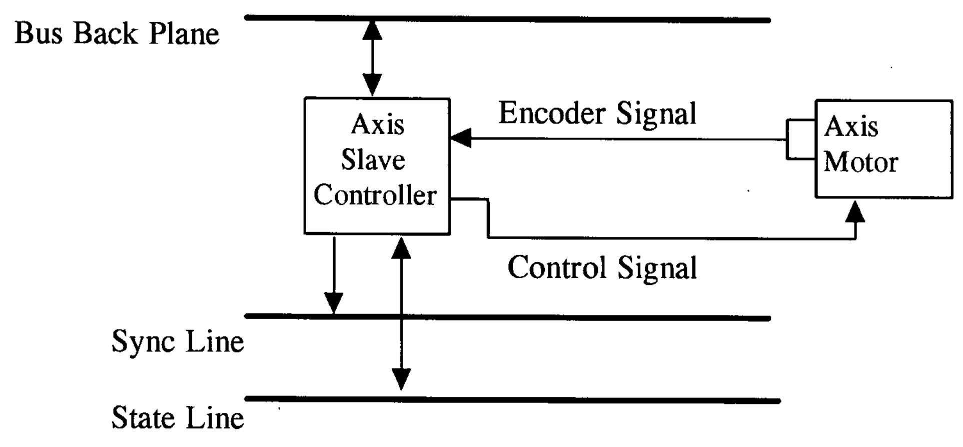

Figure 2.2. Block Diagram of a Slave Controller

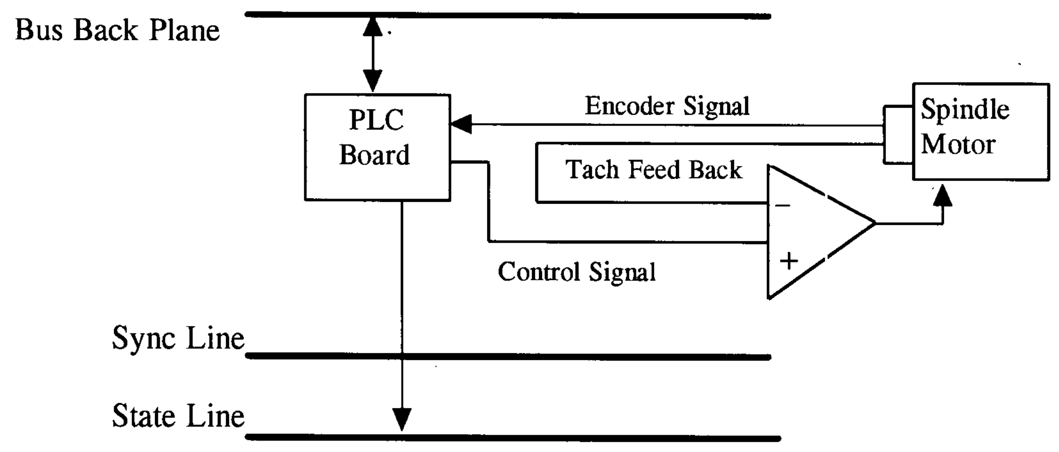

Figure 2.3. Block Diagram of PLC Board

2.3 Previous Research

Development of the UBC controller began at the University of British Columbia in 1988 by Dr. I. Yellowley at the Manufacturing Engineering Laboratory in the Mechanical Engineering Department. The development of this controller has progressed from initial research to license for commercial use by CimProvisor High Tech Industries in 1994. Previous contributions to the development of the UBC controller are as follows:

i) Late 1980's P.R. Pottier, researched the development of slave I/O controllers.

ii) In 1992 R. Adekani implemented force control and investigated the integration of CAM using Autocad and NC Polaris.

iii) In 1993 R. Seethaler developed a high speed contouring algorithm. This algorithm also addressed real time error control by looking at position error in relation to a reference velocity.

iv) Currently Dr. L. Yang is researching multiple state lines to expand upon the communication available with a single state line.

This thesis will contribute to the development of the UBC controller by researching the application of its control technique over a distributed system. This concept is introduced in the next section.

2.4 Application to Distributed Network

The application of the UBC controller to a distributed system is based upon the extended use of the state line. The expansion of the state line influence to an external source which could simultaneously influence a number of independent UBC controllers is considered a distributed system. Such a system could function with a lower level control network connected to the Local State Line (LSL) and is shown in Figure 2.4.

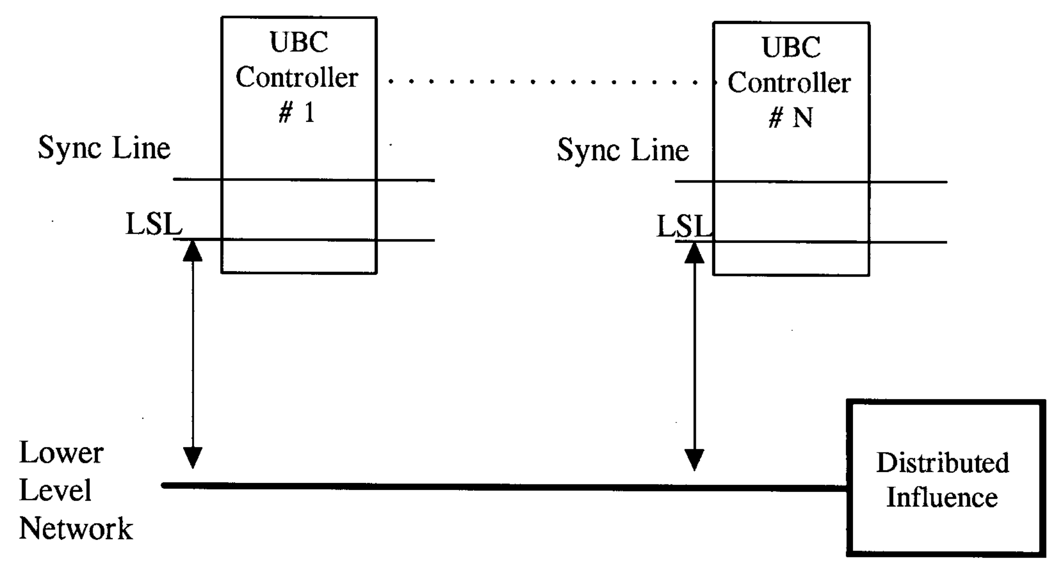

Figure 2.4. Distributed System Schematic

For the purposes of this thesis the distributed system to be tested is a CNC lathe equipped with a UBC controller. The lathe's controller is connected to a lower level control network and a higher level communication network. The distributed system is to be controlled on the basis of forces read from a dynamometer. The dynamometer readings are processed by an independent computer which is capable of triggering the lower level control network when forces read exceed set limits. There is also an additional master computer which can monitor the lower level control network and provide higher level support via the higher level communication network, such as an NC program data base. This higher level network is interfaced to the UBC controller by a network adapter which can be installed into the STD computer. In this way the lathe can demonstrate CIM consisting of control on the lower level network and CIM support on the higher level communication network. The larger system encompassing several individual machine tools to be controlled as referred to in Figure 2.4 has, due to limited experimental resources, been reduced to one machine tool with one external influence. This experimental system although not as large, will demonstrate the concept of distributed control. The distributed experimental system is shown in Figure 2.5.

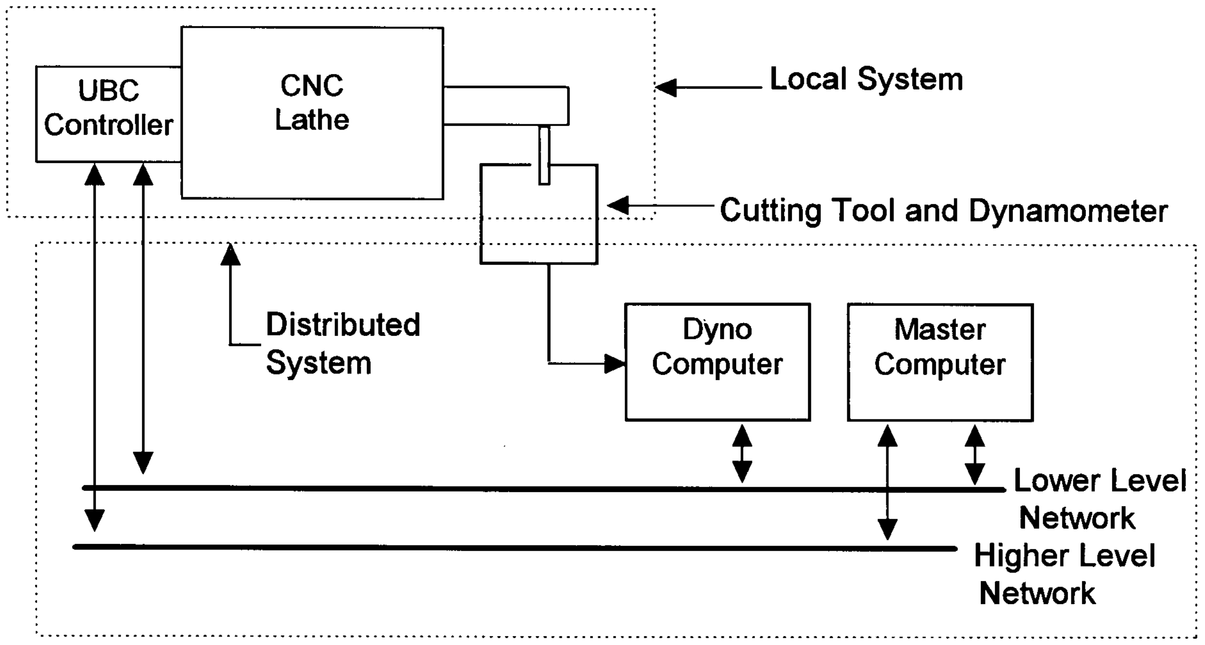

Figure 2.5. Schematic of Distributed Experimental System

The next section introduces the method of process control used by the UBC controller, the basic idea of the distributed state line at the lower level network derived directly from this idea.

2.5 Process Control

This section presents the control process upon which the lower level distributed network is based. Adaptive control applications to machining are also discussed in the light of the rather different approach is used here.

The manipulation of the LSL by the lower level network is used to demonstrate network control. In this case a machine tool is being manipulated. This manipulation of the lower level network on the machine tool is based on a simple robust approach. In contrast adaptive control is a method of control which continually monitors parameters of the system being controlled, and adjusts the control of the parameters to maintain performance.

Literature on classical control defines adaptive control as a system which adapts the controller's parameters as well as the controlled parameters to changes in the process conditions. Literature on the adaptive control of machining defines adaptive machining as a system which adjusts machining conditions based on feedback [34].

Adaptive machining is further classified as Adaptive Control Constraint (ACC) and Adaptive Control Optimization (ACO). ACC attempts to maximize metal removal rate while observing a practical parameter such as spindle torque or cutting force. ACO attempts to optimize performance based on an economical model relating the rate of metal cutting to the cost of machining.

ACC systems typically maximize the metal removal rate by increasing the feed until a parameter limit is reached. The development of these ACC systems is challenged by the need to provide stability over a wide range of cutting conditions. The problem is that controlled parameters vary with changes in the process, such as depth of cut. This can result in the controller's performance degrading to the point of instability. To maintain stability, parameter adaptive systems were developed. These systems vary the controller's parameters, normally gain, based on feedback information. This, maintains the closed loop control performance during process variations [7,23,20].

ACO systems attempt to optimize an economic performance index, such as equivalent chip thickness, while maintaining system constrains (force). This method models information about a number of parameters. Thus, it has the potential to provide greater optimization than ACC, which is based on one constraint such as force or spindle torque. However, due to the difficulty in including tool wear in the optimization model, ACC systems have remained the preferred method of adaptive controlling machining [22].

Adaptive control is presented as background to how control of machining is normally done. The simulation and experimental evaluation presented in chapter 5 do not use adaptive control; rather a simple robust approach is used.

The next section introduces flexible manufacturing systems and how the UBC controller fits into flexible manufacturing.

2.6 Flexible Manufacturing System (FMS)

Manufacturing is the process of making goods. The methods of manufacturing evolved from manufacturing a single item by hand to automated manufacturing. This section introduces factory automation, FMS and its control.

2.6.1 Factory Automation

Fixed automation is a manufacturing system in which the sequence of operations is fixed by the equipment configuration. An example of this type system is a conveyor based production line.

Programmable automation is a manufacturing system designed with the capability to change the sequence of operations to accommodate different product configurations. An example of this type of manufacturing system is a process which utilizes CNC machines whose actions are programmable.

Flexible automation is a manufacturing system which is capable of producing a variety of parts with no time lost to change over. It is known as FMS. FMS involves the automation of a manufacturing process, material handling, and warehousing. It is further reviewed in the next sub-section.

2.6.2 FMS

In an effort to build upon the manufacturing savings realized by mass production, FMS was pursued to further automate the manufacturing process. FMS can be defined as a system of machines in which parts are automatically transported under computer control from one machine to another. The system should also be able to produce a wide variety of parts. FMS is best suited to low volume and high variety systems. It fills the gap between high production transfer-lines and low production NC machines [17, 19].

FMS emerged from machining and the traditional machine tool industry. It has since expanded to include other manufacturing activities such as assembly, testing, welding, etc. A manufacturing cell is the smallest building block of an FMS. A manufacturing cell is a grouping of equipment which works together to produce a finished product or a product which is finished to a level where it can progress to another cell within the system [29].

In 1975, the first manufacturing cell was developed. It was a machining center capable of unmanned operation. It consisted of a CNC machine, an automatic tool changer and an automatic pallet changing system, which automated both stock and completed part movement [29].

2.6.3 FMS Architecture

FMS is based on automating the manufacturing process which has been broken down into manufacturing cells. To achieve FMS criteria of flexible and preferably unmanned automation, the system architecture should be capable of achieving the following [29]:

i) Be highly automated and programmable.

ii) Provide direct accessing (robot) or random material handling (automatic guided vehicles), rather than serial access (conveyor).

iii) Provide automated part, tool and storage facilities.

iv) Provide high level computer control based on distributed processing, data base and links to other CIM activities such as CAD, CAM etc.

v) Provide the ability to re-route manufacturing activities in the event of a manufacturing cell's planned or unplanned stoppage.

Table 2.1 shows the hierarchy of the different levels in an FMS architecture [29]. The control of manufacturing cells and FMS is presented in the next sub-section.

Table 2.1 FMS Functional Architecture

Level	Function
Company (main frame)	- order processing - scheduling
Plant (super mini-computer)	- decision support - CAD/CAM - production/process planning
Department (mini-computer)	- work order scheduling - inventory/material handling - tool control - performance reporting
Cell Controller	- FMS control - work piece routing - pallet/tool loading - diagnostics

2.6.4 Cell / FMS Control

Cell or FMS control has normally been based on having a cell, or module of an FMS, centrally controlled by a computer. This central computer maintains a task queue which is executed on a first-in-first-out and / or priority basis. An example is a CNC machining cell, equipped with a robot. There will be a set order of tasks the cell controller will initiate in order to move stock material through the machining process, until the finished product is sent to an inventory system. This type of cell controller is only concerned with an order of instructions which can be prioritized and executed from a queue. This type of cell is not usually concerned with synchronizing tasks.

Synchronization of control has in the past been of academic interest only. However, interest in synchronizing real-time control is growing, especially in the areas of data transmission and production management.

Synchronization of data transmission within a communication network is concerned with the operation of data bases in a high demand environment where availability of the network or the data base is a concern. In such networks when the requirement of getting the information from the data base is of time critical importance, access to the network and the information becomes important. Access to the network can be controlled by clocked timing. This will ensure guaranteed network access at known intervals. To achieve synchronization of time critical data the information within the data base can be provided with a time stamp. This will allow the data to be related to the temporal characteristics of the process under control, and allow it to be synchronized to a related process [18].

Another application where synchronization of data transfer is of importance is within a multi-processor computer network, where there are multiple masters. In such a system, data transmission is normally controlled by a self-timed bus arbitrator. A master processor on the bus transmits to a slave processor based on timing provided by the arbitrator. The slave must synchronize with the bus timing by either receiving a clock signal or deriving it from the master. Once the transmission is complete the data bus is released by the master for use by the next master [4].

Production management systems are concerned with monitoring the performance and providing instruction to the managed process. With respect to manufacturing cells or FMS the instructions from the management system are to allow for flexibility in the process. An example is an automobile production line where each vehicle on the line is identified by a bar code. The management system would allow each vehicle on the line to be individually controlled to unique specifications. The flexibility of the differing specification instructions is required to allow continuous line production [17].

Depending on the type of manufacturing tasks in a given production line, they may require synchronization. If synchronization is required, control would have to be distributed along the concerned section of the production line. The UBC controller could be used to distribute these synchronized instructions. The next section presents how control is distributed and how the UBC controller could be used within a distributed system.

2.6.5 Significance of Distributed Network Control

Distributed control has traditionally been hierarchical. For example when controlling machine tools, they are normally organized into manufacturing cells.

Cellular manufacturing is an arrangement of one or more machine tools or processes which collectively produce one or more parts. The cell is often organized in a "U" shape or loop which allows efficient movement in and out of the cell. Hierarchical control of these manufacturing cells is based on having functional levels controlled by a supervisory or individual computer. Such a system is presented in Table 2.1, and is normally composed of three primary levels, often with a higher corporate level as shown in Figure 2.6 [28].

In Figure 2.6 at the corporate management level, objectives are input and production of the lower levels may be monitored. At the central coordination level all cells are coordinated, support is provided, timing and progress are monitored and controlled. Each cell controller coordinates a grouping of processes which make up a cell. Within the cell are CNC machine tools or other processes which are individually controlled. These processes and manufacturing cells are linked to their higher levels of control, to allow specified control of each level within the hierarchical system [12].

An alternative to the hierarchical system is a network system based on the UBC controller. As previously discussed this network would be capable of monitoring and controlling a number of UBC controllers connected to a lower level network. The value of such a network system is the saving resulting from the elimination of a level of supervision. Instead monitoring could be done by sensors, similar to the hierarchical system; but, the input from these sensors could directly influence the lower level control network. This type of system could be tailored to provide synchronizing within an FMS as discussed in the previous section.

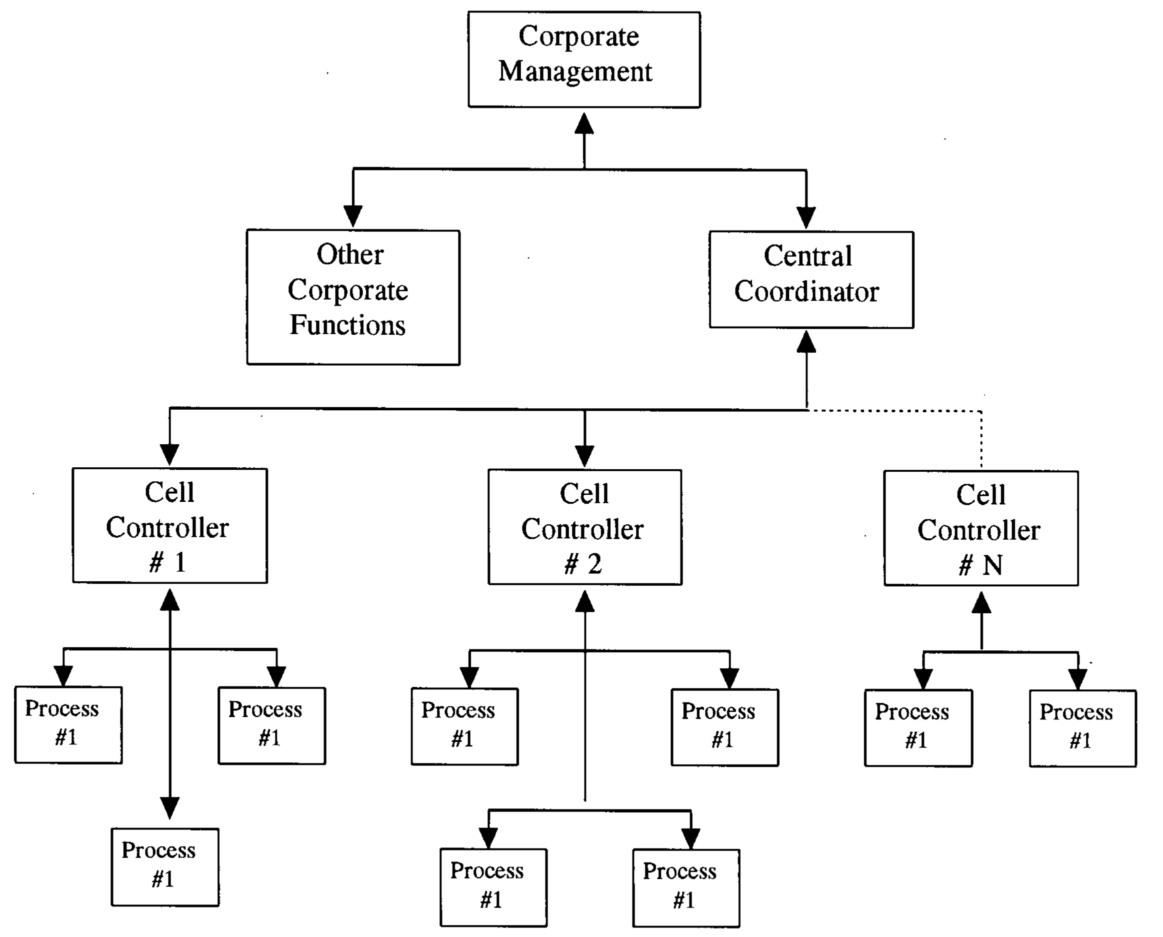

Figure 2.6. Hierarchical Control System

2.7 Computer Aided Process Planning (CAPP)

Process planning can be defined as a set of instructions which create the detailed work interactions required to produce a component. This planning process began as a manual exercise based on experience. The advent of computers and their development lead to Computer Aided Design (CAD) and CAPP to assist in design and manufacturing.

CAD systems were originally developed to assist draftsmen and designers in producing drawings and in maintaining a library of drawings. Later CAD was expanded to assist in the design by providing modeling, such as solid modeling and finite element modeling. Thus, modern CAD systems are able to provide information such as geometric data, surface finish, tolerances, etc. to assist in design and manufacturing.

CAPP systems use the information available from a CAD system to produce the manufactured goods. CAPP is the link between the design and manufacturing functions within a manufacturing enterprise. It translates the information from CAD drawing into technical manufacturing information such as material, equipment, tooling, sequencing of operations, etc. The role of CAPP is to provide effective inventory control, production scheduling, performance control, etc.

The development of CAPP systems is based on two different approaches: variant and generative. The variant approach uses coded information on a part to identify similar parts which have an existing plan. The existing plan is modified to the requirements of the new part, to produce its plan. The generative approach uses logic to integrate data available at higher levels, to generate an acceptable lower level plan. This approach uses a computer to find the optimal plan which considers all options, including those which would have normally been discarded by human planners. Thus, a plan based on all available information and options is developed [39].

In reviewing CAPP approaches it appears that the greatest opportunity to automate and optimize the planning process exists with the generative approach. The variant approach does offer a viable alternative; however, it is limited by the effort required to establish a data base and by part compatibility, and it offers minimal opportunity for theoretical optimization. Limitations acknowledged, the variant approach is still the most popular type of commercial CAPP system available, due to its ease of development. However, due to the potential to automate the generative approach, which could allow mathematical modeling and higher level input to determine the best process plan, it will likely be direction of future CAPP development [39,13].

2.7.1 CAPP Application to a UBC Controller Based Distributed Network

The unique architecture of the UBC controller should allow the interfacing of a lower level network which provides the control, and a higher level communication network. The logic of the lower level network should have the ability to control all equipment connected to its network. This logic line could for CAPP purposes be monitored. Monitoring this lower level network could provide a connected process or individual machine with the ability to confirm that network's status before proceeding with an instruction. Higher level monitoring could also determine status, such a speed of operation, to reroute tasks within the process plan. Information could also be requested from an operator using the higher level communication network.

Thus, the UBC controller based network could be used to acquire real-time information for CAPP purposes. The information available by monitoring the lower level network could be used for go / no go confirmation by an individual machine or process, and for initial planning and amending an existing process plan in operation.

2.8 Conclusions and Discussion

Presently, the method of providing distributed control in a manufacturing environment has been hierarchical. Its implementation has usually been based on MAP and TOP. However due to the continuous changes to MAP and TOP, cost, and the limitations of their interopertability, an industrial communication standard has not emerged.

Given the unique architecture of the UBC controller, it should be possible to distribute its state line control feature over a network. The control provided by a distributed state line could be used to provide a synchronous one bit logic instruction. The major advantage of such a system would be the possibility to synchronize technological monitoring tasks, generated by the CAPP system, with motion generated by the interpolator. The distributed nature of the system would allow the tasks to be carried out on separate computer platforms

Chapter 3

Networking Solutions

3.1 Introduction

In this section real-time control and the issues surrounding networking hardware and software are presented.

The UBC controller is a basis for providing a CNC machining cell with CIM. As previously discussed, CIM is the complete integration of all information and automation within a networked computer system. The integration of CIM with a UBC controller based system can be accomplished by providing the automated control with a lower level network, and providing CIM support, such as DNC, file transfer, etc., with a higher level network.

For the lower level network, which is concerned with real-time automation control, there are two options, either direct port interface manipulation or standard protocol software. A review of both of these options and the requirement for deterministic real-time control is presented, along with a review of the interconnection concerns which result from the use of different computers, e.g. PC, Macintosh (Mac) and UNIX systems.

The higher level network is concerned with providing support such as DNC (file transfer), a data base and basic communication by an email system. The physical networks and their software options are reviewed.

3.2 Real Time Control

Whenever a computer system is required to acquire data, emit data or interact with its environment at a precise time, the system is known as a real-time computer system. For control purposes a real-time system is a system whose performance is of critical importance to the industrial system or process to which it is connected. The sequence of the instructions provided by the system must be predictable to provide positive control. Real-time systems must be deterministic. An example of a system requiring real-time control is a safety monitor on a pressure vessel. Pressure sampling must occur at known intervals to ensure that the vessel remains within safety limits. The alternative is a non-deterministic system such as a time-shared computer system. For example, this type of system could be used in a library to provide access to a data base by multiple users. Use of the system would be random, with possible priority given to key tasks. Obviously a shared system would be inappropriate for controlling industrial processes [19].

3.3 Networking Physical Interfaces

Networking computers and other devices is done by utilizing interface ports. The network either provides desired information or is influenced by the computer, which reads and writes to the network via the interface port. This can be done by either using software that implements standard protocols or by proprietary lower level direct manipulation of the lines on the port interface. This latter method of control can be easily accomplished using common programming techniques and has the advantage of reducing computing overhead. However, the communication facilities provided by a port are limited to number of pins or lines which are available at the port interface being used. The software protocols are described later in this section.

The port interfacing options available on computers are normally limited to parallel, serial and Small Computer System Interface (SCSI, pronounced "scuzzy"). Serial and parallel ports are usually available on PC computers. UNIX workstations normally have serial ports, and parallel ports that can be requested as an option. Mac's have a SCSI port and an Apple Computer printer port similar to the parallel port. These are the normal port configurations for the above computers; however, any of the these computers can be customized to include the desired port interface. Computers can usually also be equipped with other network adapters such as Ethernet; these are discussed later in this section.

Computers transfer data in two ways: parallel and serial. Parallel data transfer is in eight bit bytes. The parallel interface uses a thick 25 line cable, of which eight lines are used for data transfer, and the remaining lines are used for status signaling and grounding. Although the parallel port is normally used for interfacing a computer to a printer, it can be used to interface with other parallel devices. The limitation of parallel communication is its maximum cable length of a few meters, which greatly limits its applications [19,24].

In serial communication, data is transmitted one bit at a time. Thus, it may be slower than parallel communication, but it has the advantage of not being limited by cable length and can be transmitted over telephone lines. The maximum speed is determined by both the physical interface and the protocol used. As internal communication on a computer bus is not by bit, but rather by byte (8 bits) or another data transfer standard, the data must be converted to and from serial data, for example by using a shift register at the adapter interface [19,24].

SCSI is a disk drive to disk controller interface standard, used by Mac's, but available for use by other computers. SCSI is a high performance interface which can be connected to any SCSI type device, not just hard drives. SCSI devices can be daisy chained up to a maximum of seven devices and have data transfer rates up to 80 Mbits/sec. Given the limit of seven devices in a SCSI network and the wide acceptance of PC based systems, use of SCSI interfacing is not the norm for control systems [24].

There are numerous physical networks available; however there are only two types of communication, deterministic and non-deterministic. Deterministic networks normally use the token passing method of communication. Non-deterministic networks normally use the CSMA/CD method of communication, which detects if the network is available. Two of the most common communication networks, one of each type , Token Ring and Ethernet will be reviewed. In addition the Field Bus, an automation network will also be reviewed.

3.3.1 Token Ring Networks

IBM's PC Local Area Network (LAN) is Token Ring. Token Ring is a closed loop system, as shown in Figure 3.1, which is capable of transmitting at 4 or 16 Mbits/sec. The system of communication is based on circulating a "token". The token circulates the ring, sequentially, making itself available to each of the stations. If a station wishes to transmit, it must wait until it receives the token, and then send its data (known as a "frame") within the token. Once the frame arrives at its destination, it is copied and processed. The destination station then returns the frame to the ring, with an acknowledgment that it has been received and processed. The frame continues around the ring until it arrives its originating station, which on reading the acknowledgment in the token, removes it from the ring. Usually, only one frame can circulate the ring at a time. The circulation of the token in the ring network is controlled by a token holding timer located with the system manager. System management is provided by a computer on the network, and backup management can be provided by any other computer on the network. When the frame or token is circulating on the ring, each station acts as a repeater, transmitting the data to the next station on the ring. In this way the token ring network is not limited by distances or speed as are other bus based systems. Although token use is sequential there is also the opportunity to designate differing priorities to stations, allowing the transmission order to be customized [10].

Since the token ring network provides sequential timed transmission, it could be considered as a network capable of providing real-time communication. Interfacing the token ring network to other computers or digital devices is dependent on use of a Token Ring adapter.

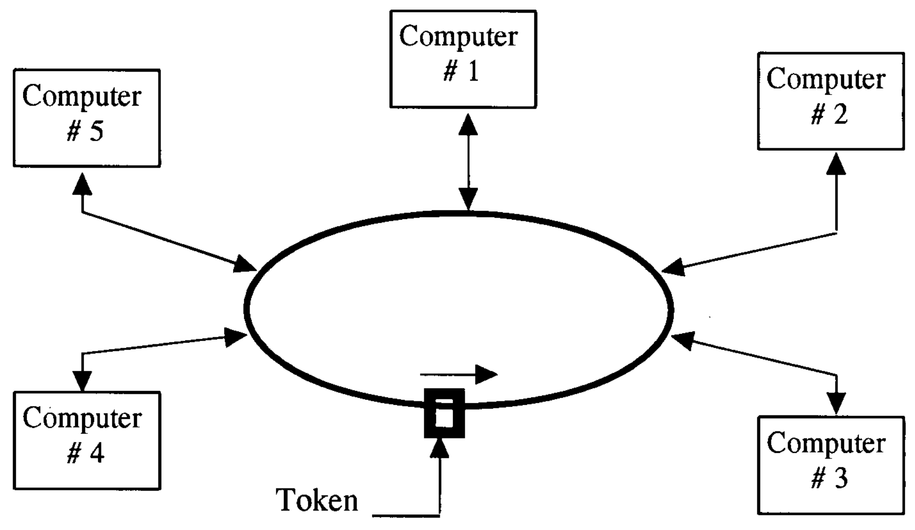

Figure 3.1. Token Ring Network

3.3.2 Field Bus

Field Buses are a type of LAN used for data acquisition and control of sensors and actuators on machines or in a factory. There are numerous emerging standards such as Bitbus (Intel), FIP (France), CAN bus (Germany) and Profibus (Germany) which are all registering their standard in an attempt to become the "Field Bus" standard. Although different, they are all based on a Field Bus adapter, I/O controllers and high speed serial communication, as shown in Figure 3.2. Cabling is by low cost twisted pair and transmission is based on the RS-485 standard. The field bus standards differ in hardware and transmission protocol. The hardware differences are in the field bus adapter and the I/O cards. Interconnection between the Field Bus I/O card and a networked computer is based on standard connections. The differing transmission protocol consists of a message which includes an address, data and administrative features. For example, the Bitbus message protocol consists of a 20 byte message, which includes a 7 byte header and 0 to 13 information bytes [5,7].

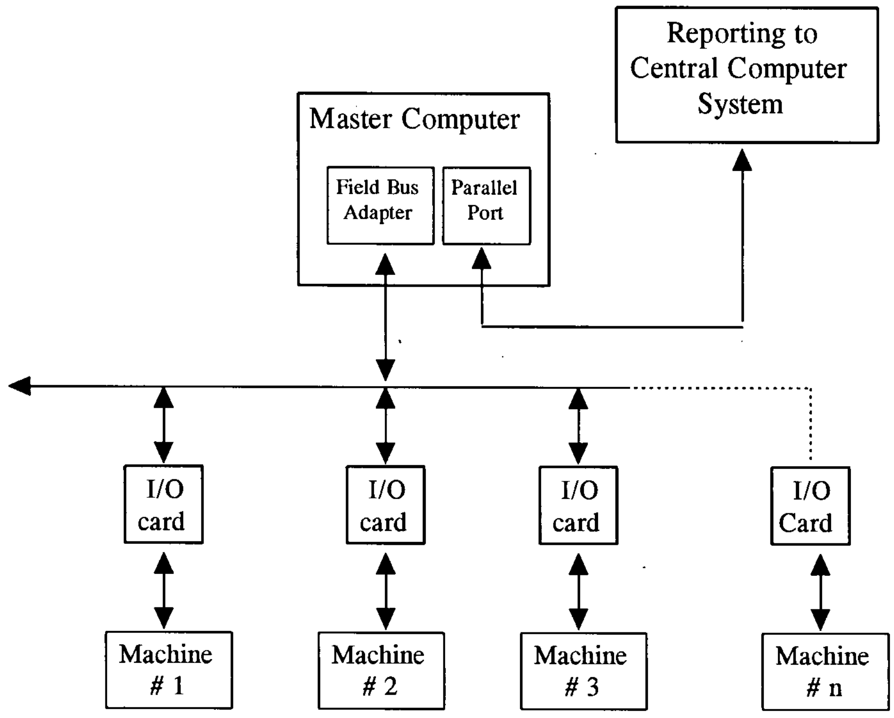

Figure 3.2 Field Bus Network

As can be seen from Figure 3.2, Field Bus has a hierarchical architecture, operating in a master / slave relationship. A Field Bus can operate in either synchronous mode, where each I/O card is polled sequentially, or self clocked mode, which uses an internal clock in the Field Bus adapter, to guarantee fixed interval computing for real-time application. The number of stations which can be operated is dependent on the field bus standard used. Repeaters are normally used to extend transmission length, but they also reduce transmission speed. The various Field Bus standards are capable of transmitting 31.5 Kbits/sec to 2.5 Mbits/sec, dependent on the mode of operation and number of stations. Figure 3.2 also shows a parallel link to a central computer system to show that process information could be relayed to a higher level for management purposes [5,9].

Field bus interfaces are supported by firmware utilities, which eases programming in high level languages such as "C". This allows qualified users to customize a field bus system to their requirements. Field bus is an evolving standard which provides an efficient and economical real-time automation network.

3.3.3 Ethernet

Ethernet is a network interface which provides economical communication between computers and other digital equipment. Its communication is based on synchronous bit sequences called frames. A frame is a series of bits which represents various fields of information including addresses (source and destination of the frame), data and other administrative instructions. All stations on an Ethernet can transmit and receive simultaneously; a frame is processed by the station whose address matches the frame's destination address. Transmission conflict caused by concurrent network use is managed by the CSMA/CD algorithm, which checks the network, allowing transmission only when the network is idle. Ethernet was intended to be used for automation, data processing and access to other terminals. However, due to its non-deterministic nature, Ethernet was not intended to accommodate real-time computing. Use of Ethernet is dependent on all stations within the network using an Ethernet adapter. Table 3.1 provides the characteristics of Ethernet [14,31].

Table 3.1. Primary Characteristics of Ethernet

Data Transmission Rate	10 Mbits / sec
Maximum Station Separation	2.5 Km
Maximum Number of Stations	1024
Medium	Shielded Coaxial Cable, Twisted Pair, Fiber Optics and Others

3.4 Networking Software

The networking software options are numerous but are limited by system compatibility and the required computational ability of a computer to efficiently use the networking software. Software compatibility, protocols, operating systems and the programs required to manage and use the networking software are presented in this section.

3.4.1 Software Compatibility

Software compatibility is normally addressed by the vendor who provides software options to operate their equipment in various computer environments. Usually software for PC computers is provided. Depending on the software and its intended application, UNIX systems (workstations) may be included; however, it is less likely that Macs would be accommodated. Workstations are not usually used within the manufacturing environment of CIM due to their cost; rather, they are utilized for engineering and design work, where their computation power is needed. Macs are normally used for their word-processing features, and have not become accepted as a computer to be used for industrial applications. Thus, PC computers have emerged as a standard for industrial applications.

Networking software, like most other software, normally has specific computing environment requirements. Usually these parameters include processor type, minimum memory and minimum hard disk space. Although documentation on these minimum requirements are readily available, information on the precise type of computer that a vendor recommends operating their software on should be sought to avoid disappointing performance.

3.4.2 Transmission Control Protocol / Internet Protocol (TCP/IP)

The defacto networking software standard has become the TCP/IP protocol suite. It is non-proprietary and provides high speed communication. TCP is a software protocol which allows communication between end station computers or other digital equipment. Data travels between stations in a TCP segment composed of seven, eight bit bytes. The TCP segments travel across the network in IP datagrams of variable length; IP defines how the datagrams are routed. TCP/IP protocols operate independent of most network data link level protocols, enabling the TCP/IP protocol to operate on virtually any network. Thus, TCP/IP may be used within a LAN or a wider network such as the global Internet. Due to this wide scope of use and its use as the Internet network software standard, TCP/IP has emerged as the software networking standard for all applications [14,31].

3.4.3 Network Basic Input / Output System (NetBios)

Although there are many other existing protocols available for communication, the only other one considered here due to its wide acceptance is the NetBios protocol. NetBios is a low overhead software interface developed by IBM; it allows PC computers to communicate. NetBios can be used within the PC, but is intended to be used on a network. It is the network software on which IBM has based their Token Ring network. NetBios consists of functions and return codes. The functions are called to provide various utilities such as "send message", "receive message" etc. Return codes are provided to advise of a called function's status, such as "interface busy", "check adapter", etc. All IBM PC computers have NetBios installed; PC compatibles may or may not be provide with NetBios. Regretfully NetBios has been developed to varying standards; thus, in general it is not possible to mix NetBios protocols from different vendors. As NetBios is IBM's PC networking software standard, it has been widely accepted and is used as the software communication interface in many programs. As a result, TCP/IP includes guidelines that describe how to map NetBios operations into equivalent TCP/IP operations. NetBios has been introduced here as it is widely accepted and because it could be considered as a communication interface which may provide low overhead communication [8,33].

3.4.4 Operating Systems (O/S)

O/S's which primarily include DOS, MAC, OS/2, and UNIX do influence the network protocol selection. However, this thesis will only be concerned with the DOS operating system. Thus, computer operating systems will not be discussed further.

3.4.5 Network Management Systems

The LAN network management system is the software which provides the central management of the LAN. The functions of this system include monitoring and tracking network performance and activity, security, fault analysis and configuration. There are numerous systems available. Table 3.2 provides a listing of the major LAN network management systems available for the physical networks previously presented [10].

Table 3.2 LAN Network Management Systems

Network manager	**Core O/S**	**Client O/S**	**Protocols Supported**	**Physical Networks Supported**
IBM LAN Network Manager *	OS/2	OS/2, DOS	NetBios	Token Ring, Ethernet and others
Novel NetWare**	NetWare	OS/2, DOS	NetBios, TCP/IP IPX	Token Ring, Ethernet and others

* Requires NetView, the primary manager located in the main frame. NetView is compatible with TCP/IP.

** Includes NetView entry point module. Internet Packet Exchange (IPX) is Novel's network layer protocol.

3.4.6 Application Programs

Application programs provide access to the software programs which are available on a computer. These programs normally have a graphical interface to ease in the use of different programs on a computer. Normally utilities which provide file transfer, email, security and other network support are or can be included. The application programs fall into two categories: independent and X/windows application programs. Independent application programs allow singular use by a computer of its programs and processor, which may be in a single task or multi-tasking environment. The X/windows application programs allow independent operation as well as terminal operation. Terminal operation allows a computer (host) to gain access to another computer (client) as a terminal, running its programs or the client's on either the client computer or the host computer.

Two standards for independent applications programs are Microsoft's Windows for Working Groups and IBM's OS/2 WARP Connect. For X/windows network application programs there are numerous products available, all of which claim compatibility with most programs and interoperability with other X/windows network Application Programs.

3.5 Conclusions and Discussion

Real-time control, network physical interfaces and network software have been presented. The over-riding requirement for the CIM application of this thesis is that the network system provide both real-time machine tool control and higher level communication.

An important consideration in selecting a network system is interoperability. Machine tools and other manufacturing control systems often have unique protocols to provide for specific requirements such as job queuing and progress reporting etc., previously discussed. These features are usually proprietary and specific to the control system of the process being controlled. This posses no difficulty so long equipment from the same vendor is used and no attempt to network information beyond the controlled process is attempted. If networking of control information is desired then a network which has an expanded capability must be used.

The UBC controller with its unique method of control based on its LSL and sync line can be controlled by a distributed real-time control system and receive higher level CIM support without use of the expensive MAP /TOP protocol suite. The selection and design of the network system which will be used to implement CIM in a UBC controller based environment is presented in the next chapter.

Chapter 4

System Selection and Design

4.1 Introduction

This chapter describes the process of the selection and design of a networked arrangement based upon the UBC controller. The network consists of two separate levels. A lower level automatic control network provides real time control to a machine tool equipped with a UBC controller. This lower level control of the UBC controllers is based on the manipulation of the LSL by the lower level automatic control network.

The higher level network provides CIM support such as file transfer which a machine tool or operator in a machining cell can access as a resource. This network is not subject to the constraints of real-time computing, but rather is based on information access and privileges.

The chapter concludes with a discussion on how the two systems are integrated to provide a complete CIM capability to a UBC controller based machining cell.

4.2 Networking the Lower Level Control Network

The lower level control of the UBC controller system is provided by the connection of the lower level automatic control network from the master computer and autonomous within the UBC controller. The lower level automatic control network is triggered high or low as dictated by the distributed influence, which for this application is the dynamometer force readings.

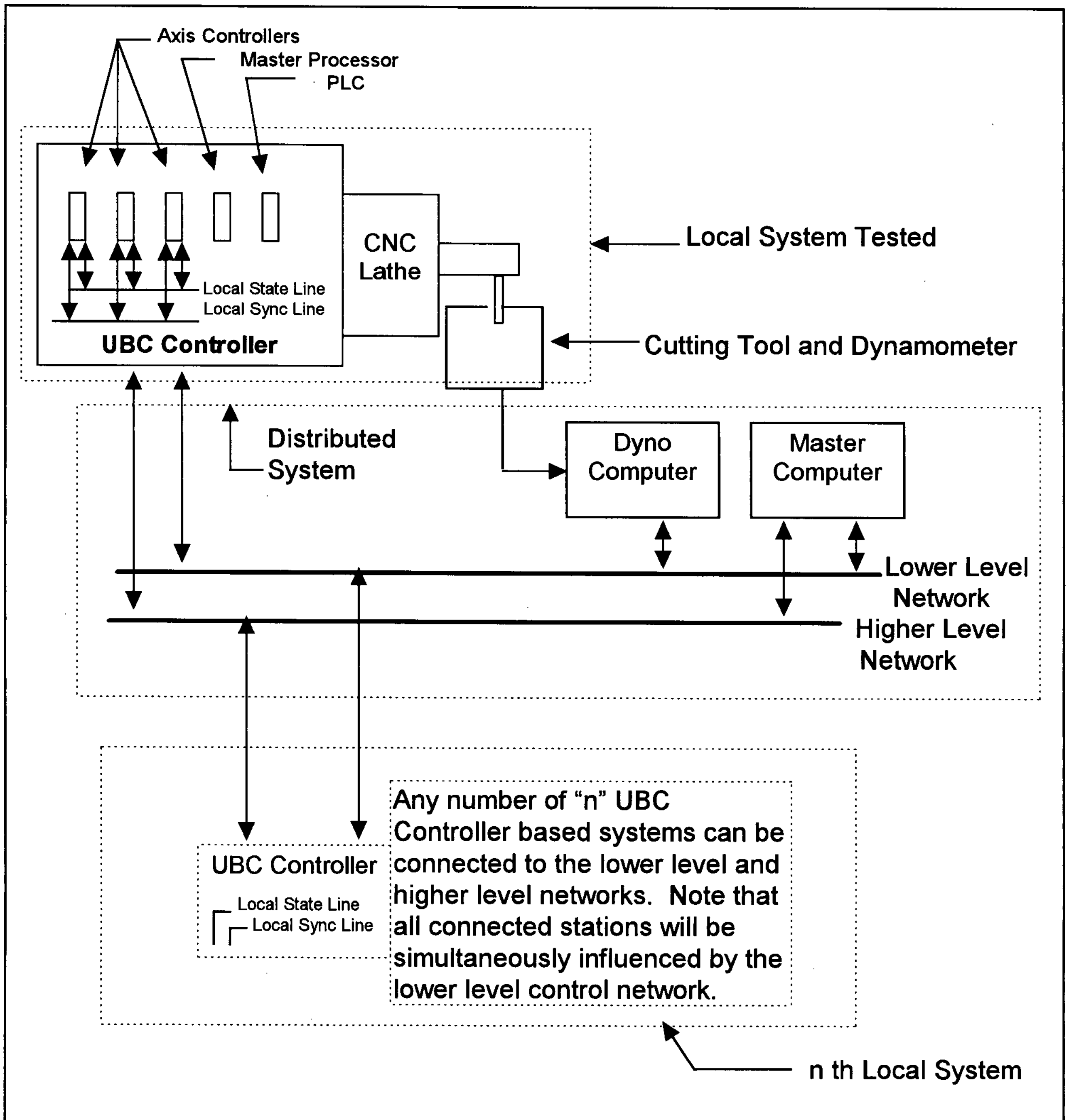

Figure 4.1. Schematic of Distributed Experimental System

Since the LSL is an open collector type circuit, the lower level automatic control network hardware must also be an open collector type circuit, in order to allow the lower level network to influence the UBC controller's LSL. As discussed in section 3.3, control of the lower level automatic control network can be either by using software that implements standard protocols or by proprietary lower level direct manipulation of the lines on the computer interface ports. The advantages and disadvantages of both options are shown in Table 4.1 [31].

Table 4.1. Lower Level Network Implementation Tradeoffs

	Advantages	Disadvantages
Direct Port Interface Manipulation	- Very fast and reliable - Frees computer from routine operations	- Difficult to change - Longer development time
Standard Protocol Software	- Easy to modify; flexible - Reliable	- Due to software overhead, can be slow - Ties the computer down to routine tasks - Expensive to purchase and develop

Both options are capable of providing the control link between the lower level automatic control network and LSL. The key design considerations are that the system be as quick as possible and be deterministic.

Using a standard software protocol introduces additional processing overhead, resulting in delayed network response. Even the smaller overhead introduced by NetBios is considered too much for this application. The standard protocol software packages are typically available only for standard network interfaces such as Ethernet or Token Ring. These interfaces have their own processing requirements, further adding to the overhead on the network. For these reasons, the standard protocol software solution is not considered ideal.

An alternative implementation of the state line system control using a Field Bus is possible, and would be as shown in Figure 4.2. Commercially available Field Bus systems described in section 3.3.2 include all necessary hardware and software and can be made to meet the criteria of being real-time. However, interfacing the Field Bus with the UBC controller, the selected controller for this system, would require a hardware interface compatible with the UBC controller's open collector state line. As shown in Figure 4.2 a Field Bus system would require a Field Bus network as well as a lower level network. The lower level network would provide the distributed control from the Field Bus network to the LSL of the UBC controller on each of the individual machines. This would allow simultaneous control of all networked equipment.

Unfortunately, this system would also be subject to an overhead delay resulting from the processing of the Field Bus by the master computer. Thus, it is also not considered ideal.

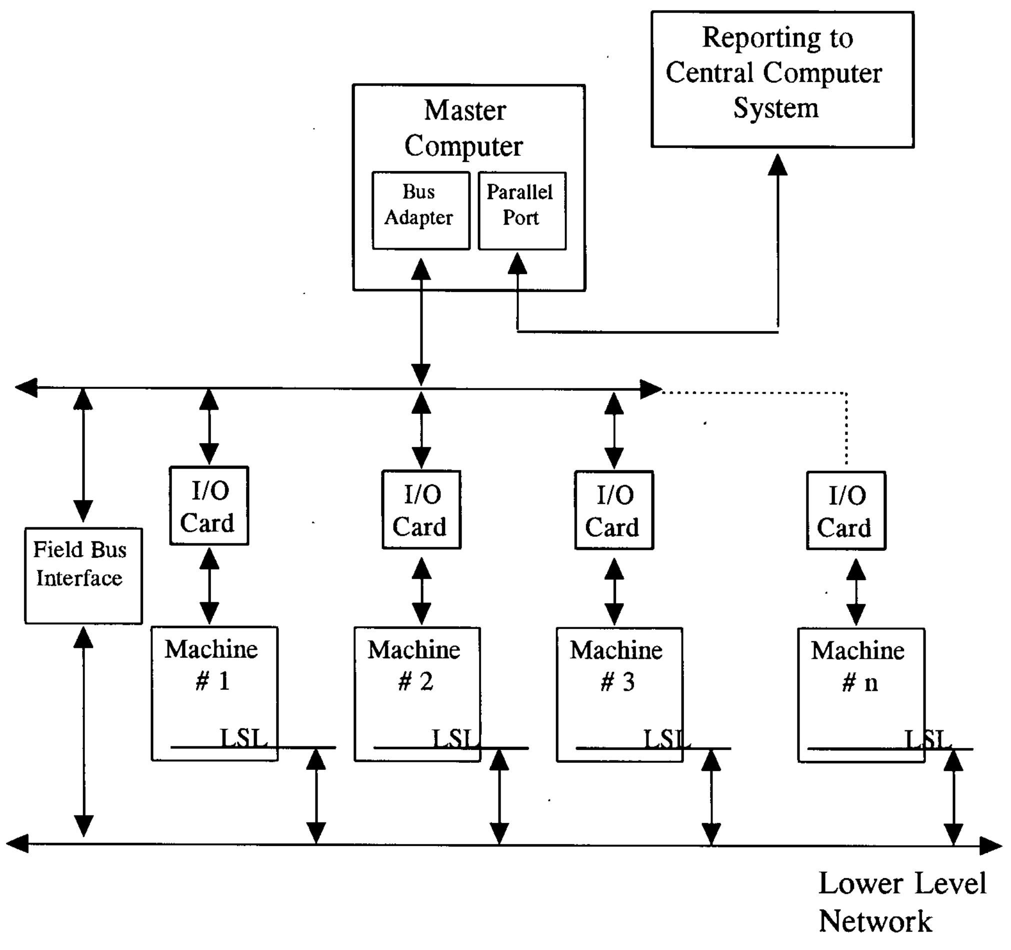

Figure 4.2. Field Bus Based Lower Level Network System

Another alternative is to simply interface the external influence (in this case the dynamometer) and the individual machine controllers directly to the distributed lower level network. This interface method will be referred to as direct port manipulation. This solution requires that this interface to the lower level network be intelligent, for example a computer reading the dynamometer output via an A/D converter to determine if the lower level network should be triggered. This concept is similar to the

Field Bus, which receives its input via a Field Bus I/O card connected to a computer. However, central processing of the input by the Field Bus computer is omitted. Instead a device directly affects the lower level network, resulting in a quicker response at the networked machines from the influencing device. The only concern regarding this system is the software in the influencing devices. Although it is similar to the Field Bus software interface, it would not be centralized; therefore, setup and debugging the network would be more complex.

The direct port manipulation solution offers fast and deterministic communication between the open collector lines of the LSL and lower level network. A limitation of this solution is in its inflexibility compared to standard software protocols. However, given that the system is not complex, the advantage of flexibility provided by software protocols in this experimental application is not considered significant. The direct port manipulation solution meets the criteria of being quick and can be deterministic; thus, this solution was pursued and is presented in the next section.

4.3 Distributed State Line and Synchronization Line Design

The requirement for a faster response, dictated by the functions of a machining cell, and the opportunity to control the lower level network without the additional investment in the field bus, lead to the pursuit of a new network based on an interface of the external influence network with the LSL.

As with the UBC controller this distributed network would require both a state line and its own synchronization line. As previously acknowledged the lower level network will connect to the LSL to provide the distributed control of the state line system. The distributed synchronization line is required to provide the lower level network 's control with a timing pulse, which will make it deterministic. Thus, the lower level network will consist of a distributed synchronization line and a Distributed State Line (DSL).

For experimental purposes the DSL selected for the system is a 12 volt Direct Current (dc) line, which is readily available from the power supply of a computer. It is acknowledged that for the DSL to function over a larger area, as would be expected for a commercial application, a higher voltage may be required to avoid degradation of the DSL signal.

The distributed synchronization line requires a clock whose timing pulse is distributed to all equipment connected to the system. Instead of adding an separate clock and line, nominal alternating current (ac) (115 volt, 60 Hz) that is readily available is utilized. Timing is obtained from the nominal ac signal by using a zero crossing detector which converts the signal's sine wave to a pulse which can be read. The graphical representation of this conversion is shown in Figure 4.3.

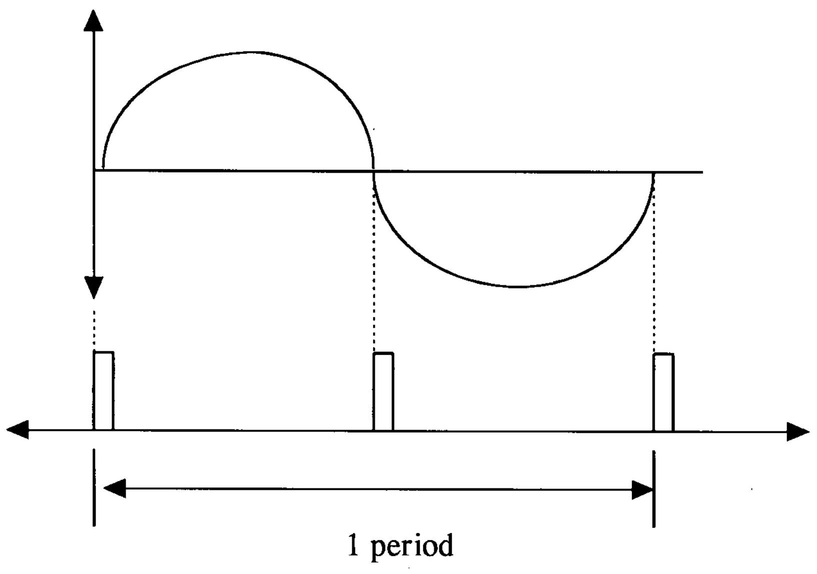

Figure 4.3. Zero Crossing Capture of the Timing Pulse

As one period of the nominal ac signal is 1/60 th of a second, the time between zero crossing detection can be calculated as shown:

$$T = 1 / f \tag{4.1}$$

$$T = (1/2) \times (1/60) = 0.00833 \text{ sec}$$

$$T = 8 \text{ ms}$$

The interfacing of the DSL and distributed synchronization line to the computers is accomplished by a circuit board which was designed to operate from the parallel port of a PC computer. This parallel port Distributed Interface Board (DIB) has been designed to allow the computer to read the distributed synchronization line and trigger the DSL high or low. Complete details of the DIB is provided in Appendix 2.

The interface of the DSL and distributed synchronization line to the UBC controller is provided by the State Line Interface Board (SIB) shown in Appendix 3. The purpose of the SIB is to interface the local UBC controller to the lower level network (the DSL and distributed synchronization line), and to isolate the two systems from each other.

The SIB has been designed to include a delayed response. This delay has been included to smooth the response of the servo motor and to demonstrate the ability of the SIB to control the influence of the lower level network on a UBC controller. Smoothing of the servo motor response in this situation is required due to the similarity in the response time of the servo motor and distributed synchronization line. The distributed synchronization line has a clock timing of 8 ms. The high performance servo motors have a mechanical time constant (velocity loop) of 12 ms. Due to the motor time constants and distributed synchronization line timing being of the same magnitude it is possible that if the DSL is rapidly triggered, the response of the servo motor would result in large velocity variations of the feed rate. This would be caused by the response at the motor being almost equal to the marginally quicker distributed synchronization line. By providing the circuit delay on the DSL's influence of the LSL the resulting response at the servo is an averaged reflection, rather than a choppy

The delayed response is caused by cycling the state line system high and low is graphically shown in Figure 4.4. The delay is provided by a feature in the SIB's circuit which produces the high / low cycling for a short period based on a delayed "one shot". The SIB was designed to provide a 50 percent feed rate for an 8.7 ms interval on triggering the DSL from low to high, and for a 5 ms interval on triggering the DSL from high to low. Further details of this feature are provided in Appendix 2.

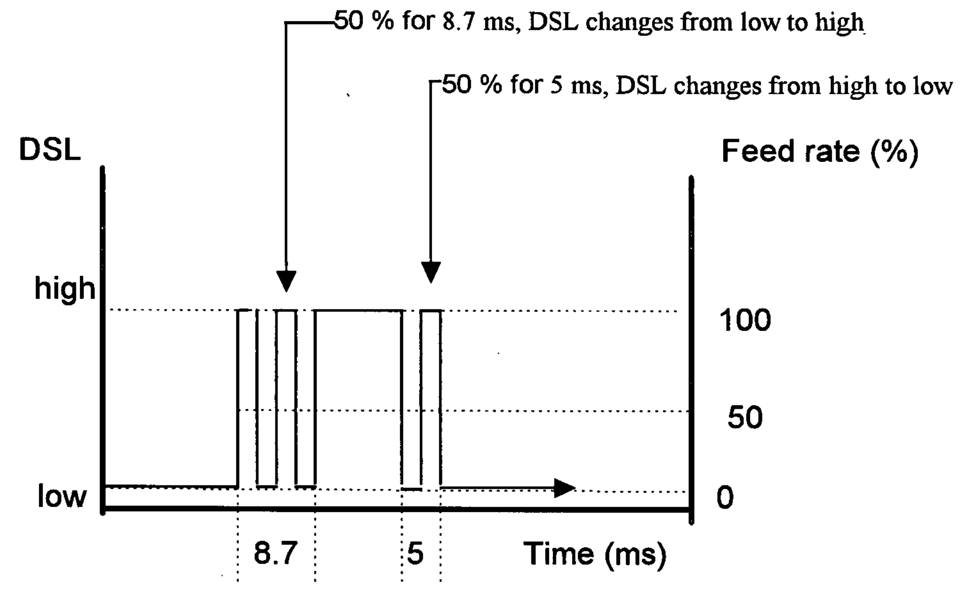
50 % for 8.7 ms, DSL changes from low to high
50 % for 5 ms, DSL changes from high to low
DSL
Feed rate (%)
high
low
100
50
0
8.7
5
Time (ms)

Figure 4.4. SIB Triggering of the DSL

Figure 4.5 graphically shows the smoothing effect of the SID. Rapid DSL state changes are displayed along the x-axis. The servo motor response is super-imposed on the DSL state changes which does not show the triggering delays as shown in Figure 4.4.

Figure 4.5 graphically shows the smoothing effect of the SID. Rapid DSL state changes are displayed along the x-axis. The servo motor response is super-imposed on the DSL state changes which does not show the triggering delays as shown in Figure 4.4.

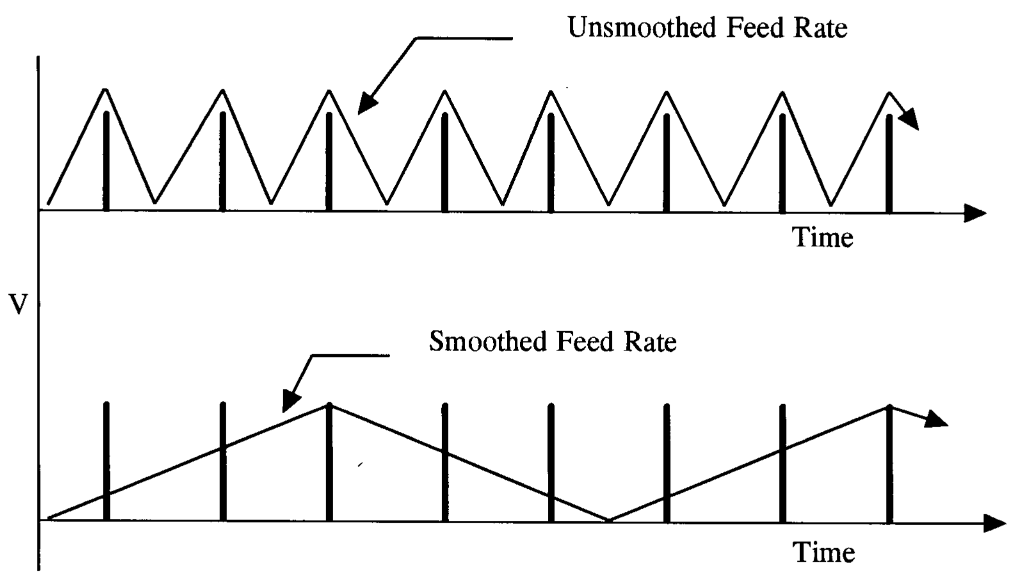

Figure 4.5. Servo Motor Response

For this application a program, listed in Appendix 4, processes the dynamometer force readings which have been read through the A/D converter. If the force read exceeds a limit then the DSL is triggered low; conversely, if the force read is within limits, the DSL is trigged high. All program actions are based on the 8 ms timing provided by the distributed synchronization line, which the program obtains from the DIB.

Details of how this distributed network performs in simulated and experimental conditions are provided in the next chapter.

4.4 Higher Level CIM Support Network

The high level network CIM support is intended to provide DNC, file transfer, data base and email capability. This network is a communication link and is not required to adhere to the real-time constraints of the lower level network. The implementation of this network and the application program which supports these capabilities are presented in this section.

The physical networks previously presented were Field Bus, IBM's Token Ring, and Ethernet. Field Bus does not support higher level communication such as file transfer; thus, it was not considered. Given that there is no intention to use this network for any real-time application, the deterministic token passing feature of Token Ring is not an advantage. Ethernet is an economical network which has wider acceptance in industry. TCP/IP is a non-proprietary open system protocol and Internet standard which has wider acceptance than NetBios. Given the flexibility that Ethernet and TCP/IP offer, this network and protocol are selected as the network system for the higher level CIM support.

The application program selection is based on considering an independent program such as Microsoft's "Windows" or an X/windows program. The independent application program does meet all of the criteria of the higher level network; however, it does not offer the greater flexibility provided by X/windows. For example X/windows provides the ability for a computer to act as a terminal allowing access to other computers on the network. This allows the user of a slower computer to run programs on a faster computer which may be idle or have the ability to accept the additional load, and it will allow access, to workstations which also run X/windows. This latter feature will allows on-line access to workstation data such as a blue print data base, very helpful information which may not normally be available on a PC. Thus X/windows application programs are considered the preferable option.

There are numerous X/windows applications programs available, offering similar features. For this experimental system Quarterdeck's DESQview/X was selected. It is an X/windows application program which provides multitasking, DOS and Microsoft Windows compatibility, and data transfer, remote computing and compatibility with other X/windows systems. DESQview/X has its own LAN network manager which is compatible with those discussed in Section 3.4.5 as well as others. DESQview/X also comes with Novel's TCP/IP kernel. The file transfer and remote computing facilities are dependent on access list security, which each computer must individually set. The remote computing facility allows complete host / client use of computers. It also includes a remote shell command, which allows a command to be sent to a remote computer; the remote computer will execute the command as if it were entered locally. For example, although slow due to software overhead, this facility, would allow operation of a cell by remote commands. Given the lower level control network, a supervisor with access could monitor the cell's DSL and also send an instruction to the cell master computer to affect the DSL control of the cell. This is not a real-time link, but does allow higher level influence on a manufacturing cell. Source code for a remote shell command from the cell master computer to influence the DSL is provided in Appendix 4.

Email on the system would allow both local mail as well as wider access given that the TCP/IP protocol is used. Numerous email programs are available; however, the email facility will not be pursued at this time and will not be included in the experiments and system evaluation in the next chapter, due to the limited added value it would provide to the experiment.

Figure 4.6 shows the configuration of the higher level CIM support network and its integration with the lower level control network in the experimental system architecture.

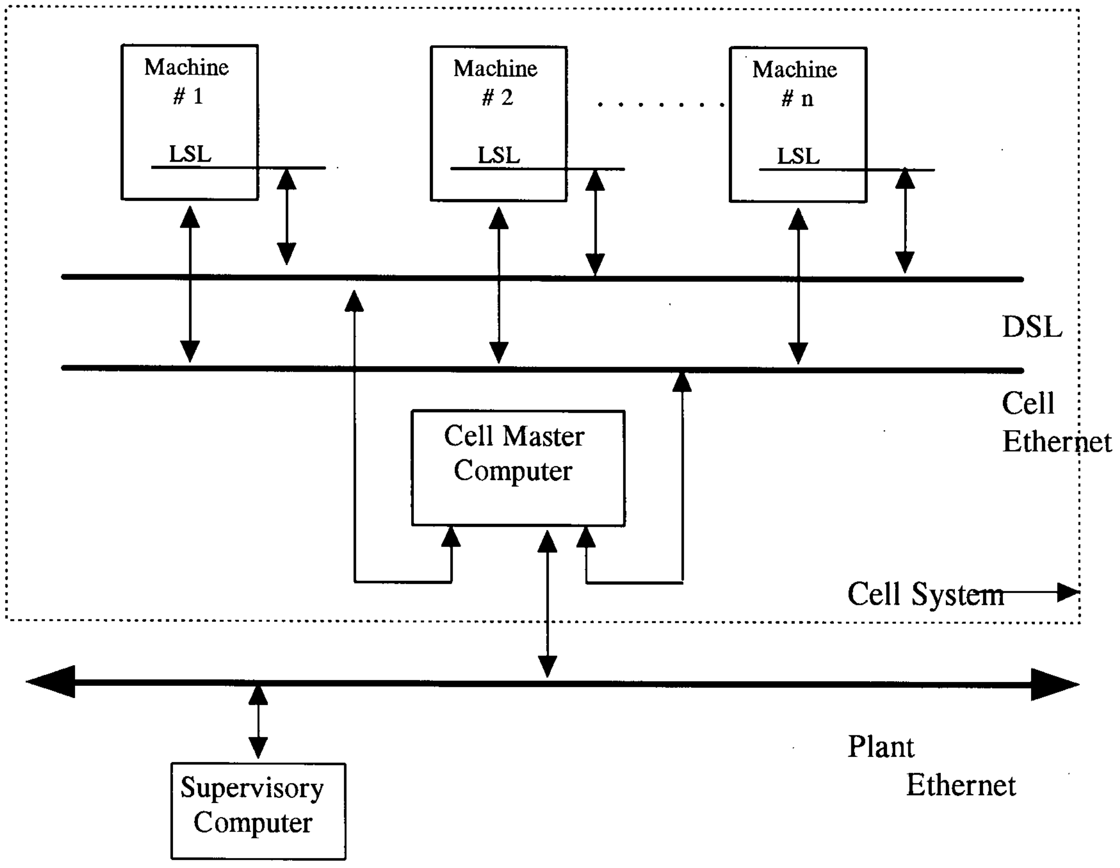

Figure 4.6. Integrated Higher and Lower Level Networks

Figure 4.7 shows the configuration of the experimental system.

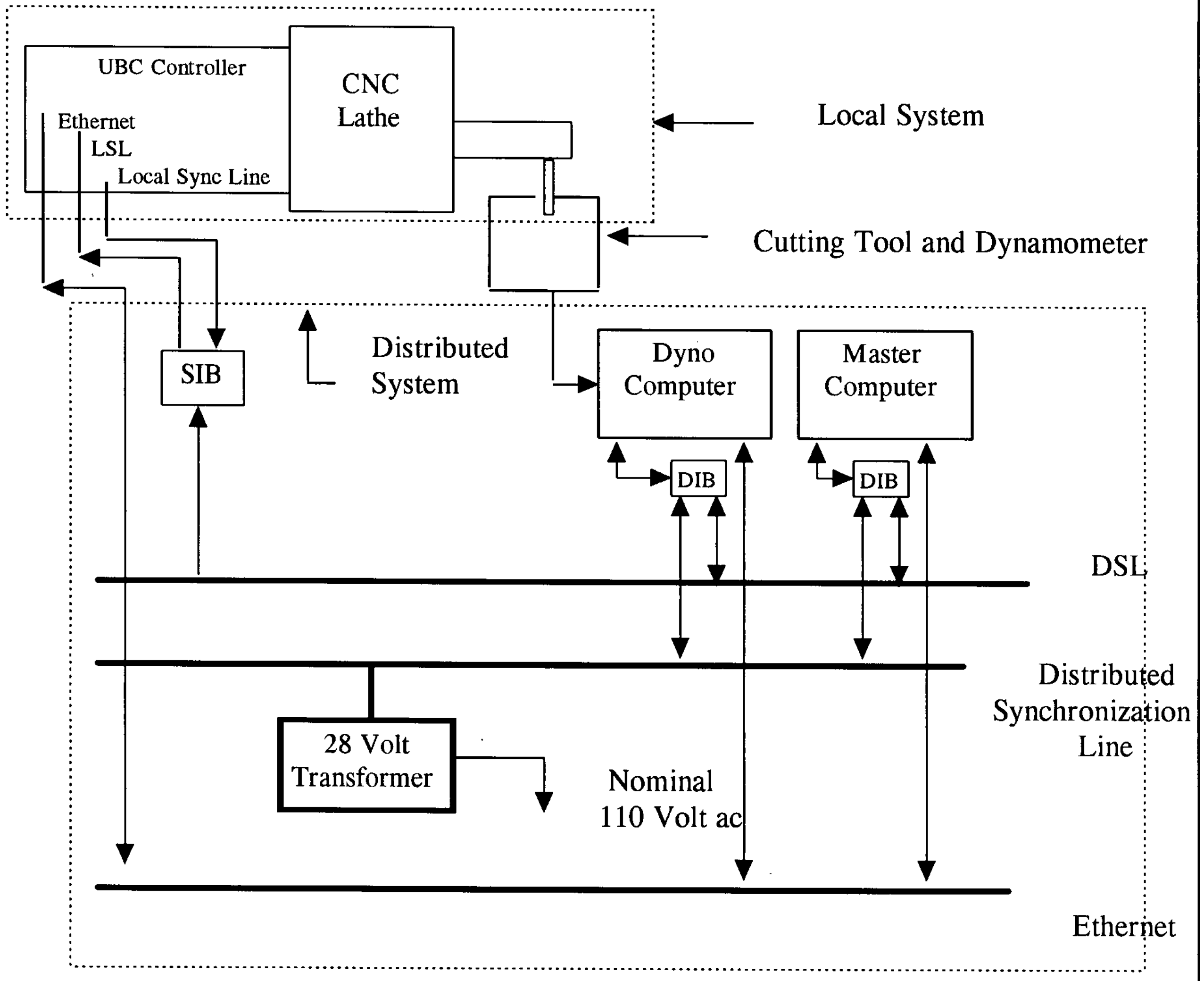

Figure 4.7 Detailed Schematic of Experimental System

4.5 Conclusions and discussion

The CIM system proposed in this chapter consists of a lower level control network and a higher level CIM support network.

The lower level control network selected consists of a hardware interface (DIB) from an external influence (dynamometer) to the DSL. The DIB is interfaced to the parallel port of the computer. The computer samples the dynamometer with an A/D converter. A program monitors both the forces read on the A/D converter and the distributed synchronization line, via the DIB interface. If the forces read are beyond set limits then the program writes to the parallel port, instructing the DIB to influence the DSL. This network was extensively tested by simulating its performance and by experimentation; the results are presented in the next chapter.

The higher level network consists of commercial products; the physical network is Ethernet, the protocol is TCP/IP and the network manager and application program are Quarterdeck's DESQview/X. These products were installed in the UBC controller (STD computer) and the master computer shown in Figure 4.7. The ability for the master computer to monitor the DSL and communicate with the UBC controller via the Ethernet was confirmed. However, further analyses of this higher level network was not conducted as it does not easily lend itself to simulation nor experimental testing. The higher level network's performance will not be examined. Specifications and further information on performance is available from the vendor.

Chapter 5

Simulation and Experimental Results

5.1 Introduction

In this chapter the evaluation of the lower level network begins with a review and definition of the tests apparatus, as shown in Figure 4.6. Then the model used for the simulation of the lower level network is presented. The chapter concludes with the presentation and comparison of the simulated and experimental results.

5.2 Experimental Apparatus

As shown in Figure 4.6 the apparatus is a CNC Hitachi Seiki lathe controlled by a UBC controller. The UBC controller is interfaced with both the lower level network and the higher level network. This discussion is concerned exclusively with the lower level network. The UBC controller interface to the lower level network is via the SIB. The SIB is driven by the local synchronization line and provides the interconnection of the DSL and LSL. The DSL can only be influenced by the distributed influence which in this case is a dynamometer reading of the cutting forces from the lathe. This external influence is interpreted by a computer, which on analyzing the forces from the dynamometer decides if the DSL should be pulled low or not. The analysis of the force reading by the computer is based on the distributed synchronization line timing of 8 ms. The interfacing of the force analyzing computer with the lower level network is via the

DIB. The DIB is polled to obtain the distributed synchronization timing, and provides an interface to the DSL.

The Z axis position loop of this system is presented in block diagram form in Figure 5.1. This is a second order system.

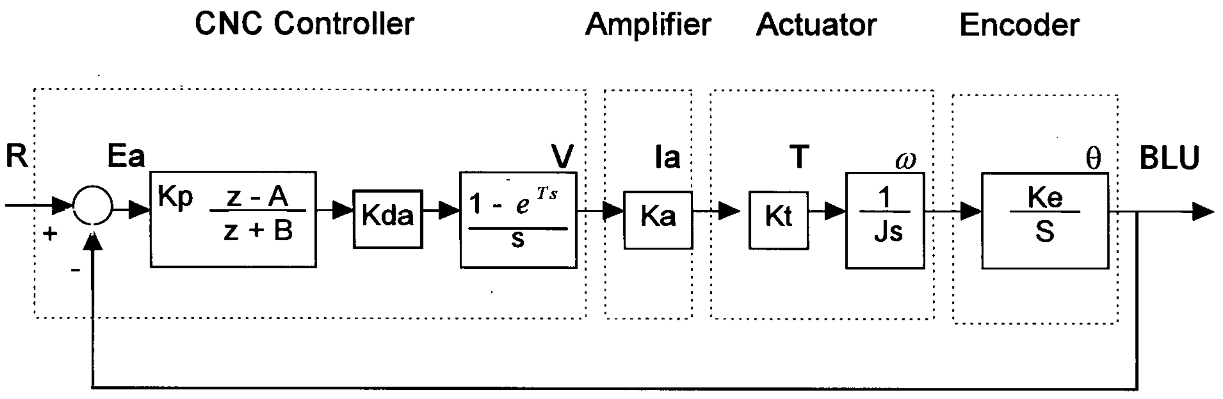

Figure 5.1. Z-Axis Block Diagram

The constant values for the lathe's Z axis servo motor are listed in Table 5.1.

Table 5.1 Z-axis Position Control Loop Constants

Z-axis Control Loop Constants	Symbol	Value
Digital Filter Parameters	Kp A B	90.00 0.985 0.25
Encoder Gain	Ke	1273 [pulses/rad]
D/A Converter Gain	Kda	0.0049 [V/Level]
Servo Amp Current Gain	Ka	13 [A/V]
Motor Torque Constant	Kt	0.1 [Nm/A]
Motor Mechanical Time Constant	tm	0.012 [sec]
Motor Electrical Time Constant	te	0.010 [sec]
Equivalent Motor Inertia	Jeq	0.0013 [N.m.s^2]

The motor constants were obtained from the motor's vendor. The 12 bit D/A converter has a range of $\pm$ 10 volts. The servo amplifier gain was obtained by measuring the current tap of the amplifier, in response to a known input voltage. The z-axis motor has encoder feedback; a tachometer generator was not included.

The simulation uses the information in Table 5.1 to model the feed drive mechanics and tool motion. The cutting force Fv is based on the force model and cutting edge forces as shown in Figures 5.2 and 5.3.

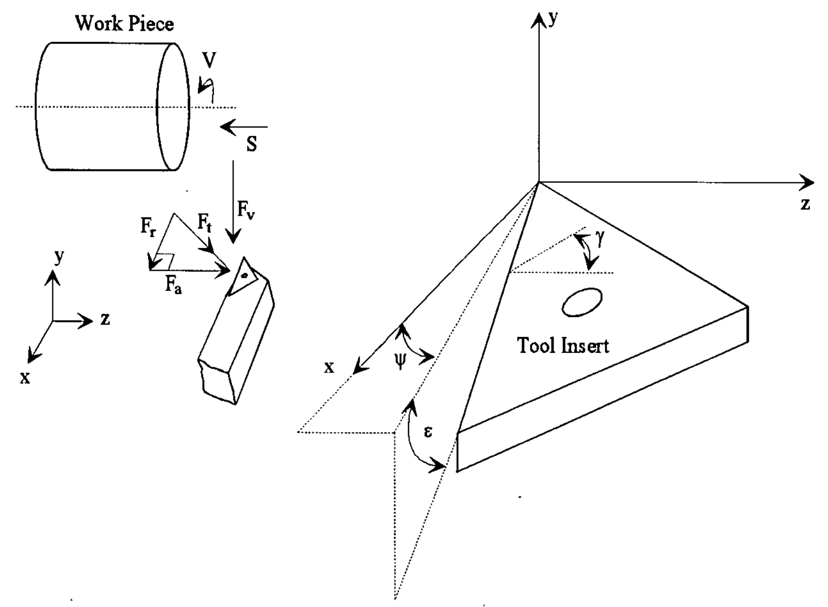

Figure 5.2. Bar Turning Tool Force Geometry

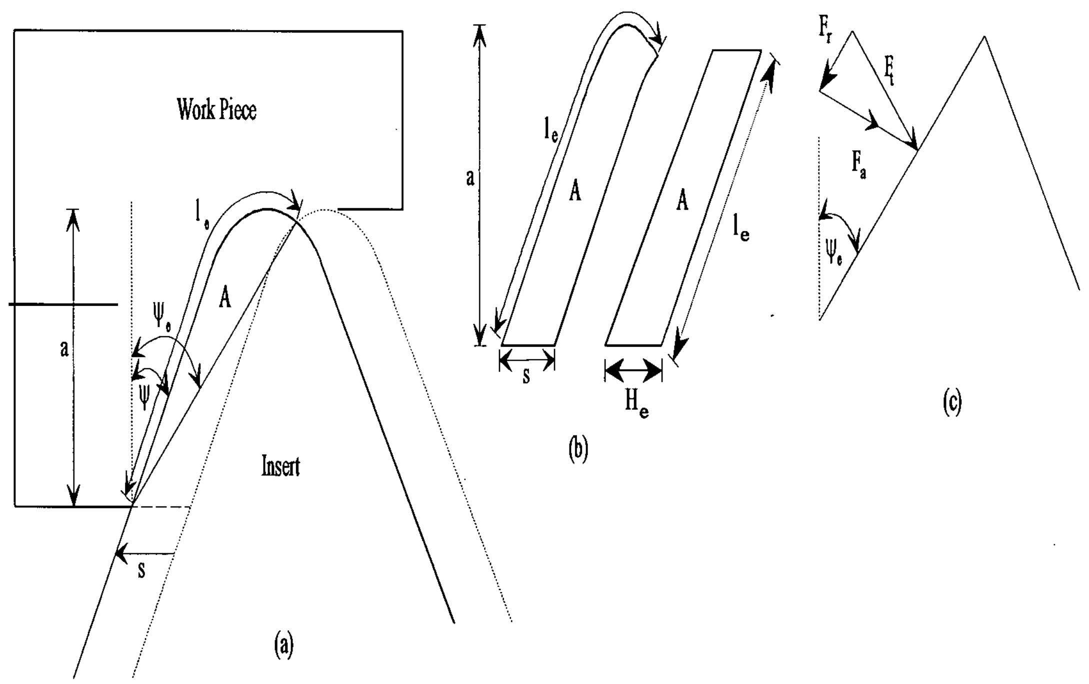

Figure 5.3. Tool Workpiece Interface Geometry

Figure 5.2 shows the total thrust force (Ft) is composed of the axial (Fa) and radial (Fr) forces. Chip sections from the turning may be represented by the equivalent chip thickness model. This model is a mean value of the chip thickness averaged over the engaged cutting length [6,26,37]. The following equations are used to model the equivalent chip thickness (He):

$$\text{He} = \frac{\text{Area}}{\text{engaged cutting edge length}} = \frac{\text{A}}{\text{Le}} \tag{5.3}$$

$$\text{Area} = \text{depth of cut x feed per revolution} = \text{a x s} \tag{5.4}$$

$$\text{Le} = \frac{(a - R(1-\sin\psi))}{\cos\psi} + R\left(\frac{\Pi}{2} - \psi\right) + \frac{s}{2} \tag{5.5}$$

where: R: tool nose radius
ψ: tool approach angle [rad]

The equivalent chip thickness combines the effects of the approach angle and tool nose radius on the cutting forces and temperature [6]. Nakayama et. al. [26] showed that at constant cutting velocity the cutting forces and temperature are a linear function of the equivalent chip thickness. The orthogonal cutting equations is:

$$\frac{\text{Fv}}{\text{Le}} = K_1 \text{ x He} + K_2 \tag{5.6}$$

Equation 5.6 is used to calculate the main cutting force Fv. The cutting pressure constants K1 and K2 were previously determined by experimentation [2]. They were obtained by taking three straight passes with the same tool at a constant depth at different feed rates; this allowed simultaneous solving for the constants. The pressure constants and tool characteristics are listed in Table 5.2.

Table 5.2 Cutting Tool Characteristics and Pressure Constants

	Symbol	Value
Tool Nose Radius	R	0.8 [mm]
Tool Approach Angle	Ψ	-3 [deg]
Rake Angle (effective angle)	Ψe	-6 [deg]
Cutting Pressures	K_1 K_2	836 55

The evaluation, by both simulation and experimentation, of the performance of this lower level control network is done for the conditions, listed in Table 5.3.

Table 5.3 Network Tests

Test #	Depth of Cut (mm)	Derivative	Max Force (N)
0001	2	0.00	800
0002	2	0.00	350
0003	2	0.10	350
0004	2	0.20	350
0005	2	0.40	350
0006	taper 4 - 0 mm depth over 20 mm length	0.40	350

The taper and normal cutting passes are shown in Figure 5.4

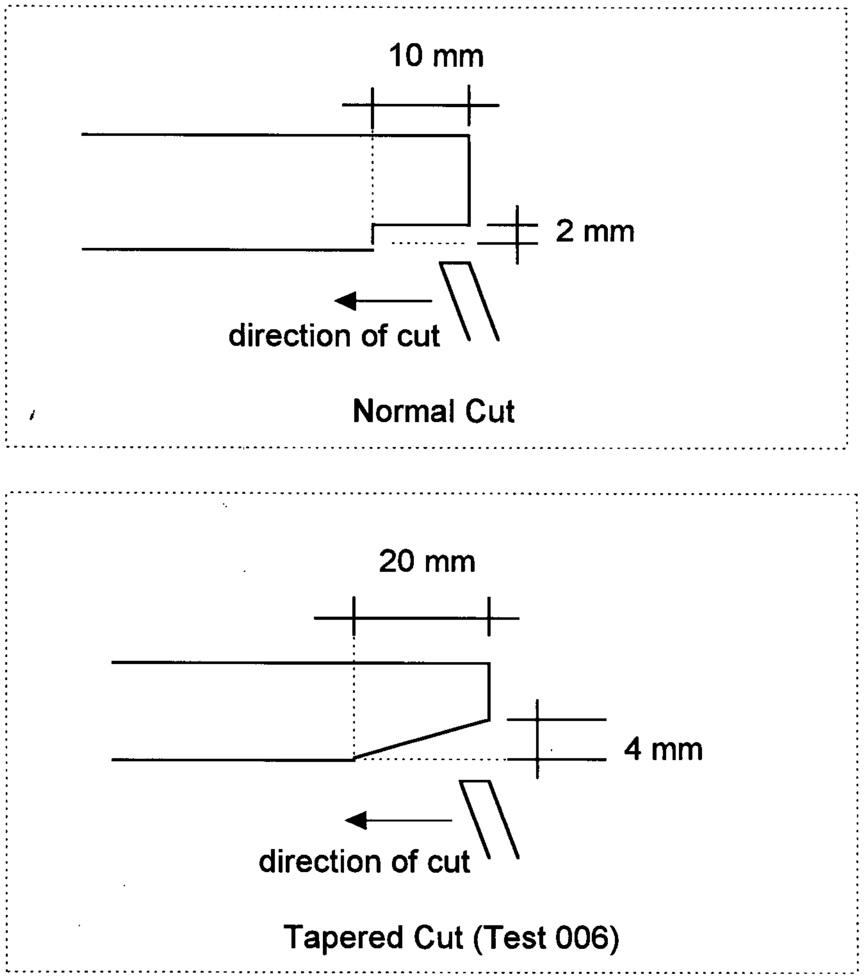

Figure 5.4. Material Cutting

All testing is done with the lathe operating at 600 Rotations Per Minute (RPM) and a z-axis feed rate of 0.25 mm per revolutions. The cutting length is approximately 10 mm and the depth of cut is 2 mm. Test 0001, which has a high force constraint, shows the resulting force with no constraining effect by the system. The remaining tests have a force constraint of 350 N which allows the lower level network to control the z-axis feed rate influenced by the processed force readings. In Test 001 the controller does limit the force without a derivative; however, as the test results show, force levels oscillate about the set force constraint. Thus, a derivative action is added to reduce the oscillating force. The derivative action progressively increases from 0.0 to 0.40 in Tests 002 through 005, clearly demonstrating its effect on reducing oscillations.

The distributed timing is 8 ms which results in a maximum of 9 ms delay in response by the system. This delay is a result of sampling at 8 ms by the distributed synchronization line on the DSL, and a response time of 1 ms by the UBC controller from the LSL. The UBC controller's synchronization line provides a 1 ms sampling time which results in 1 ms response time from pulling the DSL low to the low being read by the UBC controller through the SIB and actioning on the LSL. This response time is graphically shown in Figure 5.5.

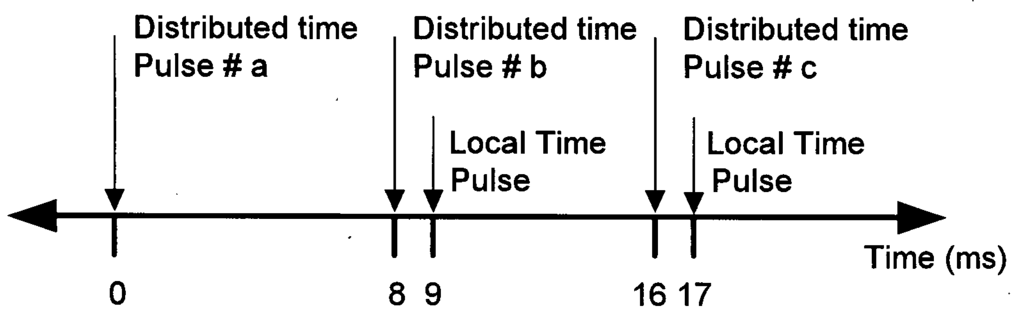
Distributed time Pulse # a
Distributed time Pulse # b
Distributed time Pulse # c
Local Time Pulse
Local Time Pulse
0
8
9
16
17
Time (ms)

Figure 5.5. Response Time

The simulated and experimental evaluations are presented in the next two sections.

5.3 Simulated Results

The simulated evaluation is conducted using the simulation source code listed in Appendix 4. The force simulation is based on equation 5.6. To calculate the instantaneous force required by the simulation, the force was held below a set maximum force limit. This is achieved by reducing the feed rate when the main cutting force and predicted change in the cutting force exceed the maximum force limit. This calculation is based in equation 5.7.

$$F_v + K_d \frac{dF_v}{dt} \leq F_{v\max} \qquad (5.7)$$

The simulation program also assumes that the entire active cutting edge remains in full contact. In reality, tool deflection and possible inconsistency of material result in contact which may be less than simulated and difficult to model. The simulated results for Tests 001 through 005 listed in Table 5.3 are presented in the following figures.

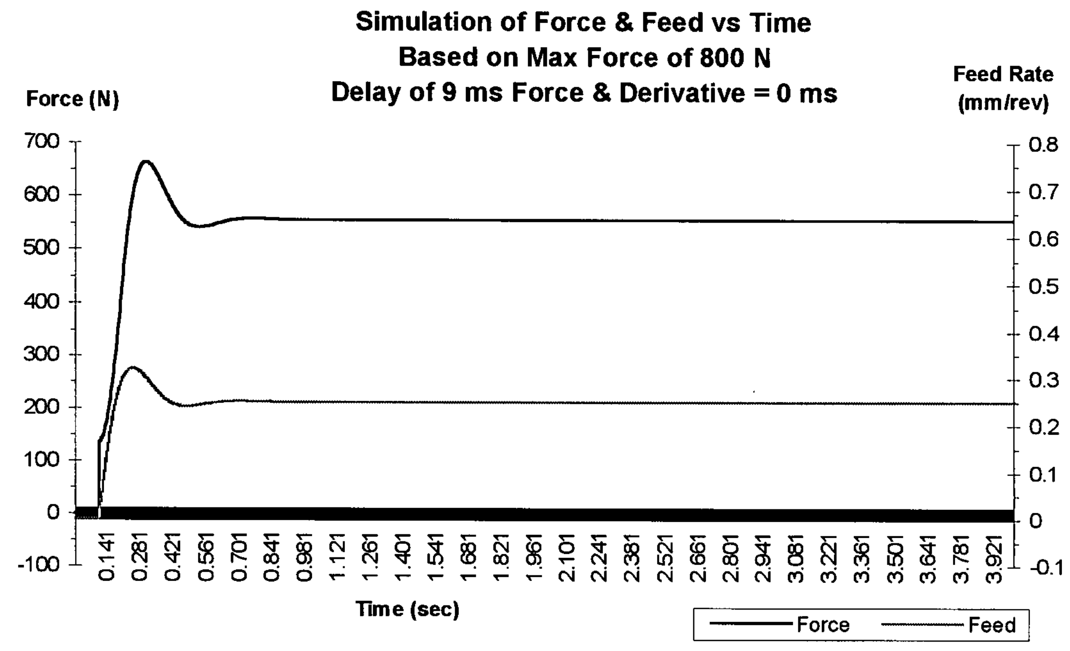

Figure 5.6 Test 001 Simulation

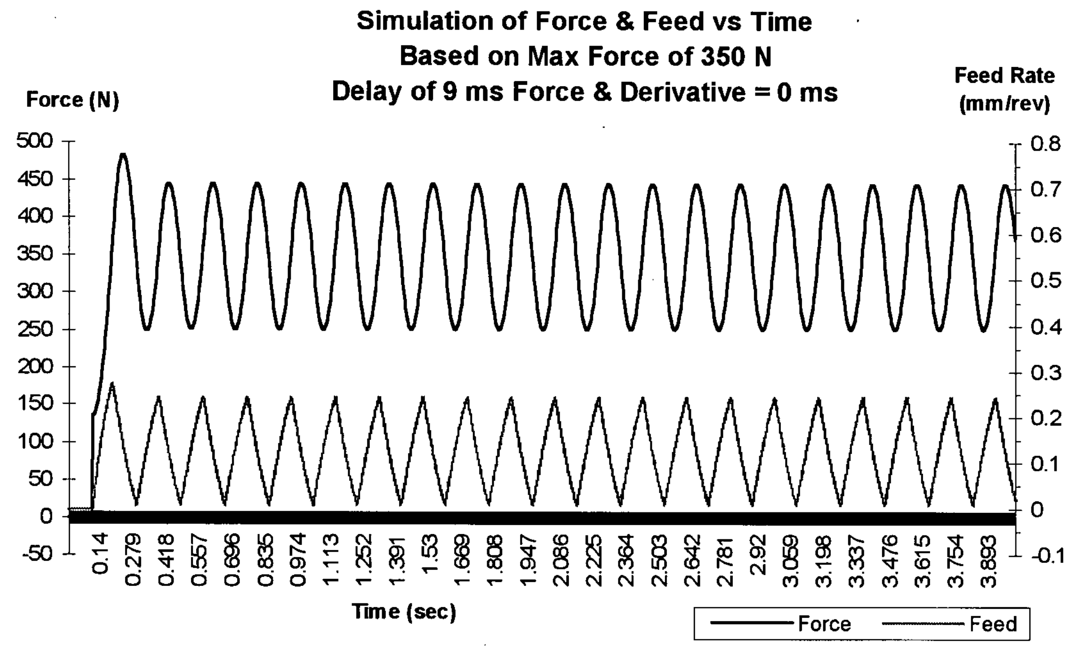

Figure 5.7 Test 002 Simulation

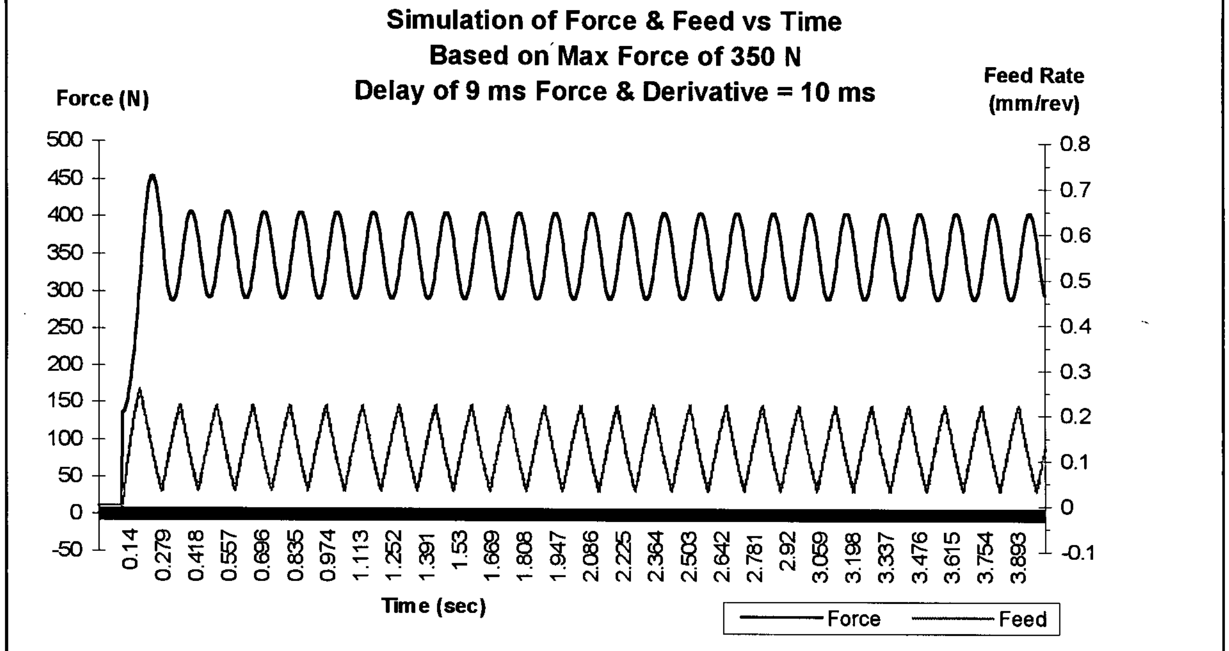

Figure 5.8 Test 003 Simulation

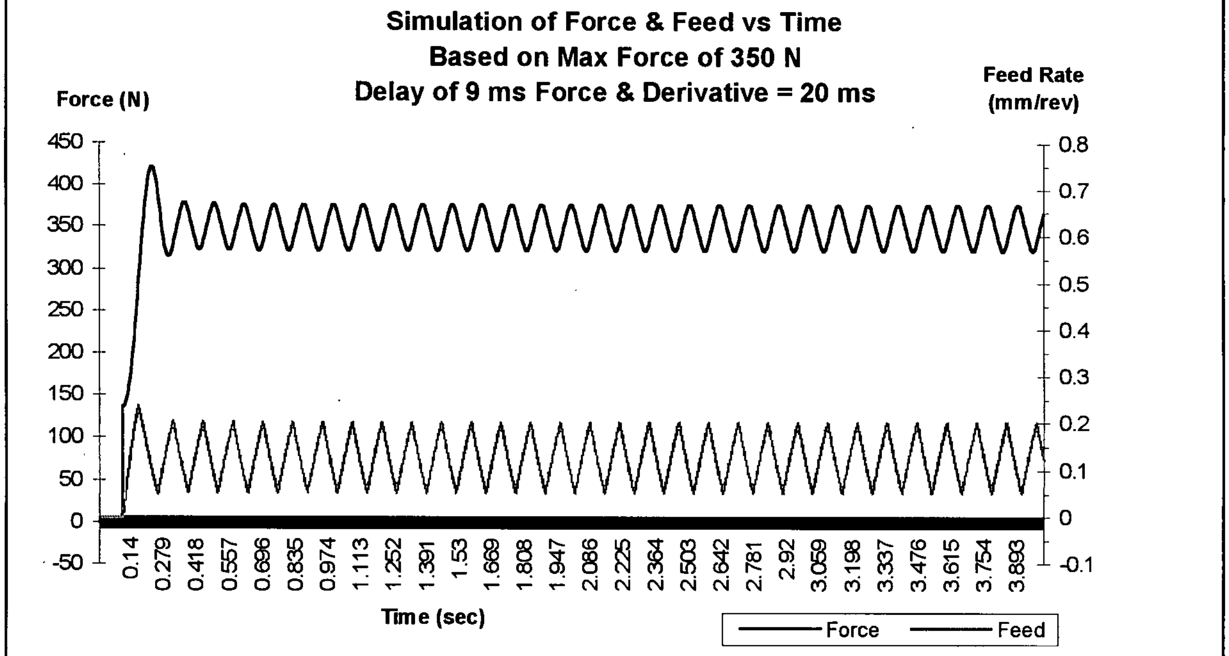

Figure 5.9 Test 004 Simulation

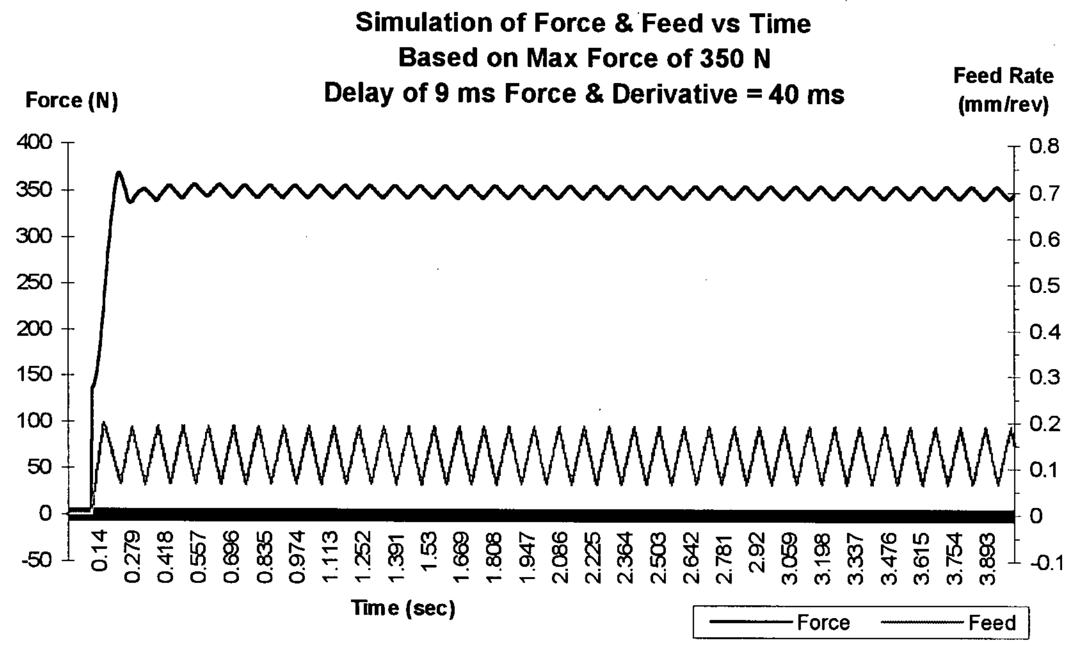

Figure 5.10 Test 005 Simulation

Figure 5.5 shows Test 001, the system operating without the constraining influence of distributed force monitoring. A constant force of approximately 550 N at a constant feed rate of 0.25 N is produced. In Test 002 a constraining maximum force of 350 N was added. This resulted in the system attempting to hold the force below 350 N. The resulted as shown in Figure 5.7 is that the force varying from 450 to 250 N and the feed rate varied from 0.25 to 0 mm/rev. In Tests 003 through 005 derivative action was added. The derivative provides a prediction of expected forces, which is used to control the feed rate to keep the force below the maximum force limit. As the derivative increase, its prediction also increased, resulting in progressively less overshoot from the maximum force limit of 350 N. The sampling frequency for tests 002 through 005 varied from 5 to 9 Hz, progressively increasing as the derivative increased.

5.4 Experimental Results

The experiments listed in Table 5.3 were conducted on the previously described test apparatus which consists of a Hitachi Seiki lathe equipped with a UBC controller and the distributed system. The experimental results are presented in Figures 5.11 through 5.22. For each test (001 through 006) a plot of the Force and Feed Rate versus Time and a graphical representation from an oscilloscope monitoring the experimental system are shown.

Figure 4.6 shows the experimental apparatus. The Dyno Computer which monitors the dynamometer readings, has an A/D board. The A/D board results are used by the program listed in Appendix 4 to trigger the DSL. The A/D dynamometer results are also retained in an array, which has been plotted as shown in the following Figures. The oscilloscope is connected to the experimental system to provide a graphical representation of the status of the system. The DSL and LSL readings are obtained by connecting to those lines. The feed rate is obtained from a tach generator which is connected to the z-axis motor. The force readings are obtained from connecting to the dynamometer signal.

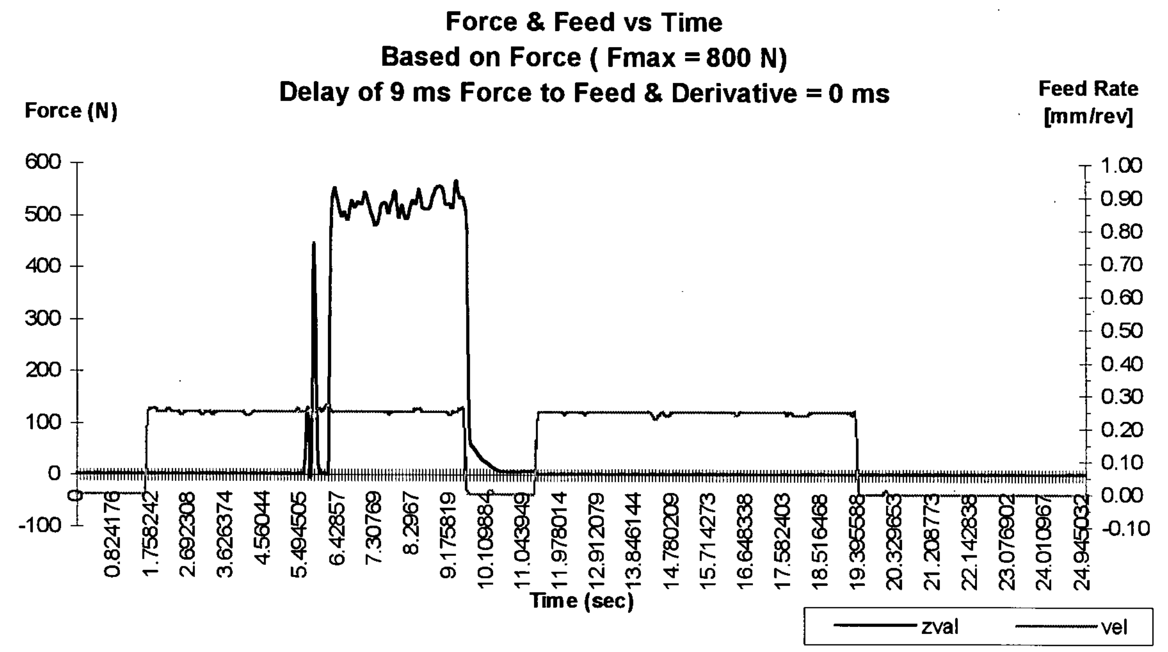

Figure 5.11. Test 001 Experimental Results

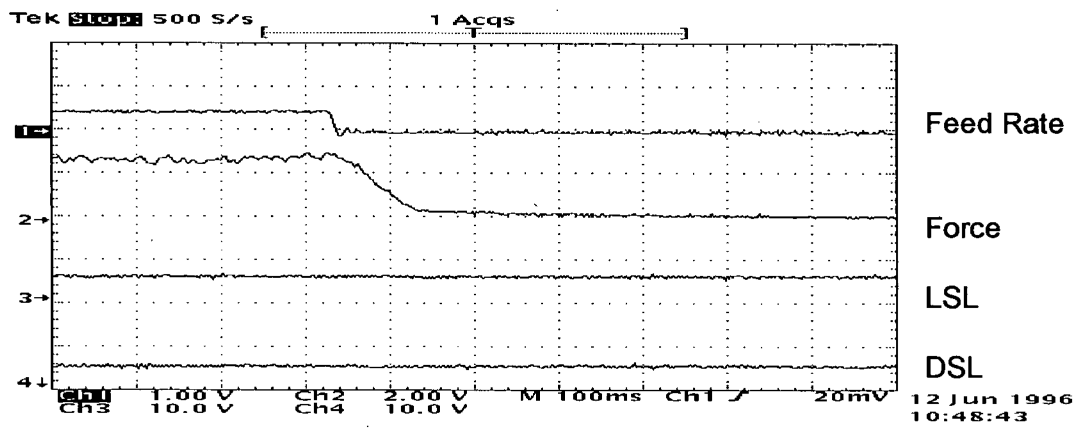

Figure 5.12. Test 001 Experimental Oscilloscope Plot

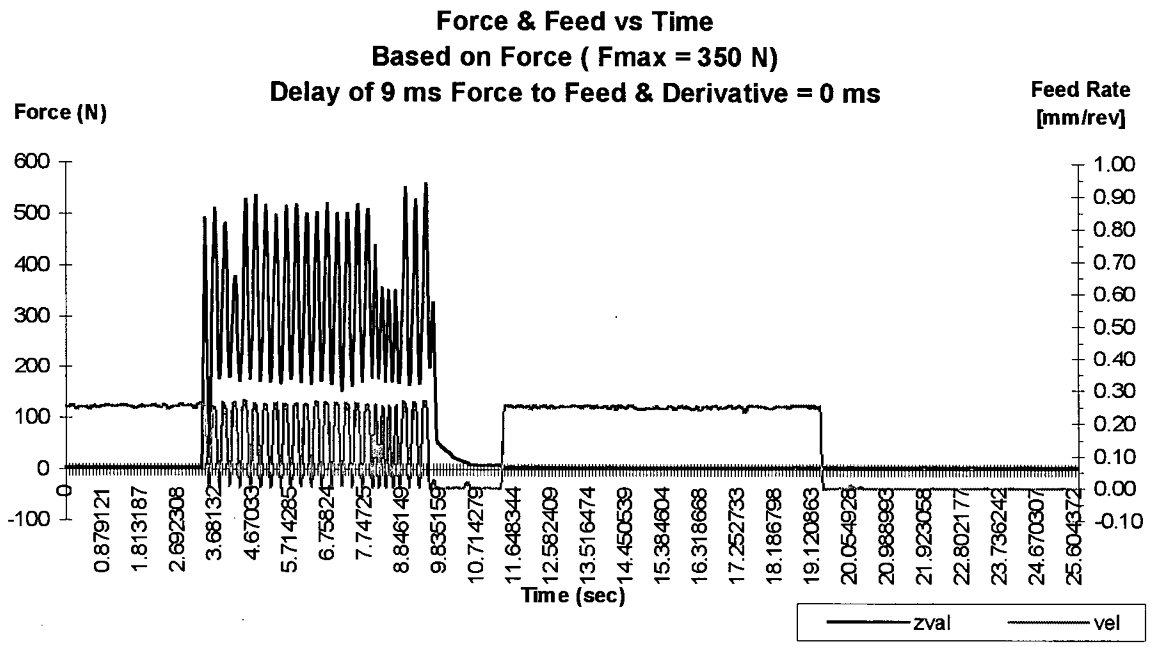

Figure 5.13. Test 002 Experimental Results

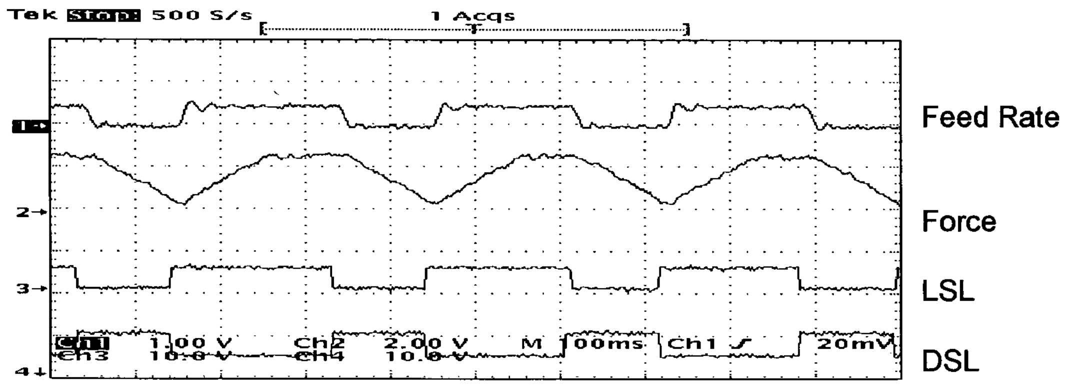

Figure 5.14. Test 002 Experimental Oscilloscope Plot

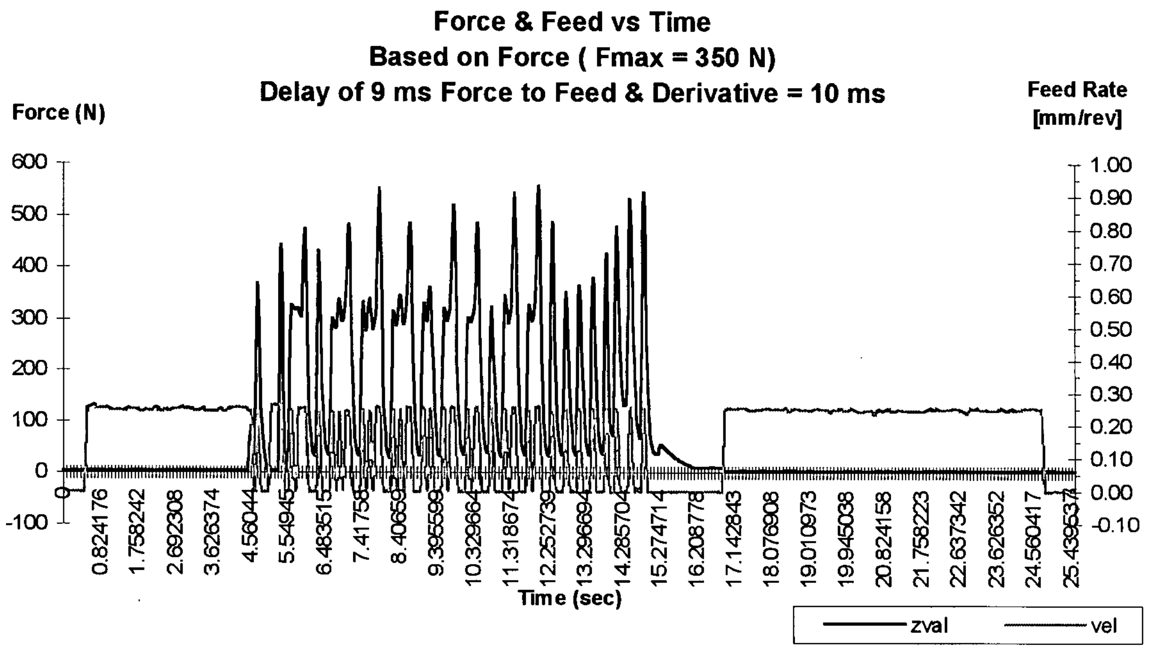

Figure 5.15. Test 003 Experimental Results

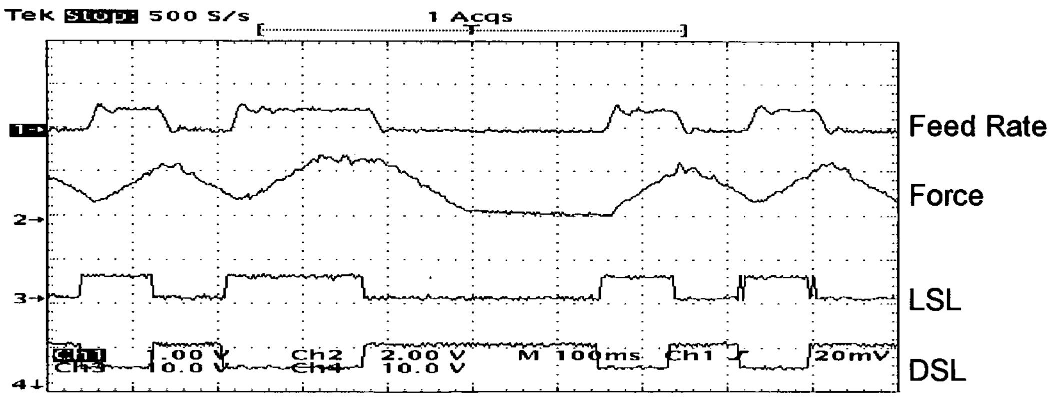

Figure 5.16. Test 003 Experimental Oscilloscope Plot

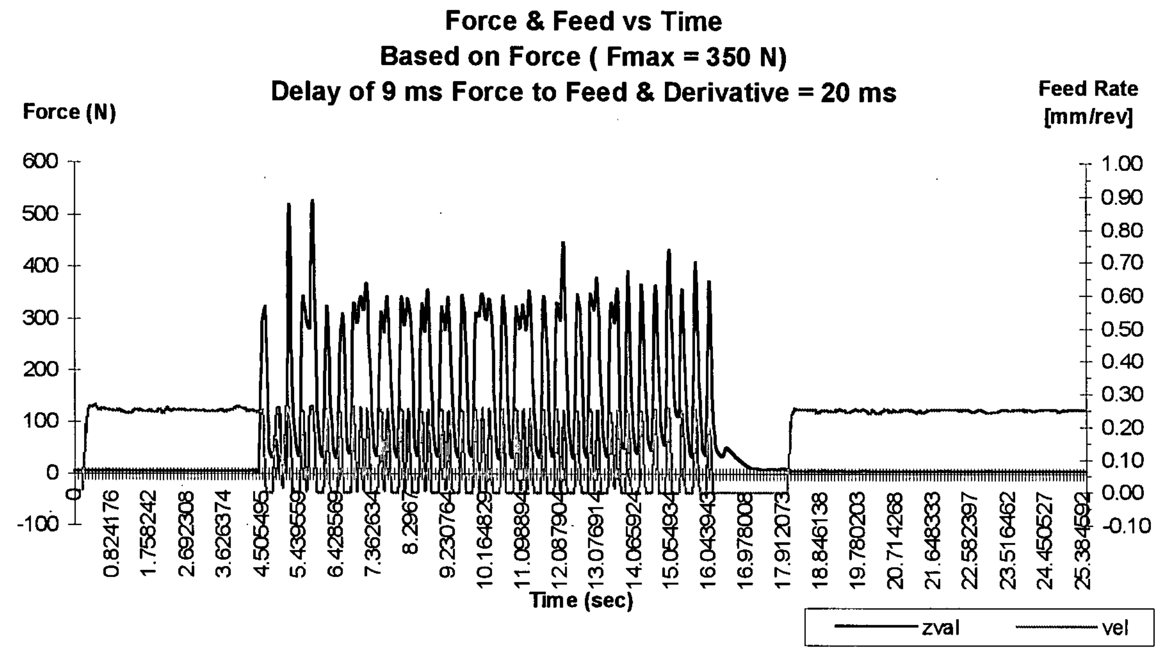

Figure 5.17. Test 004 Experimental Results

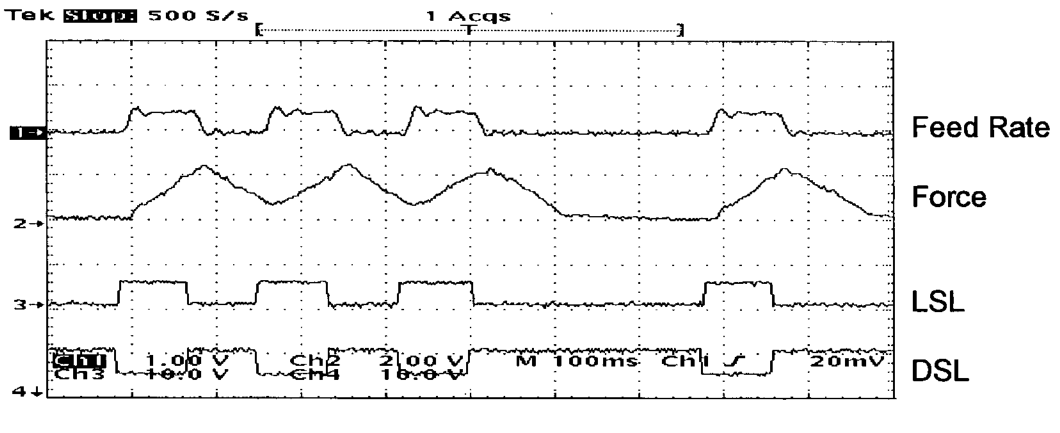

Figure 5.18. Test 004 Experimental Oscilloscope Plot

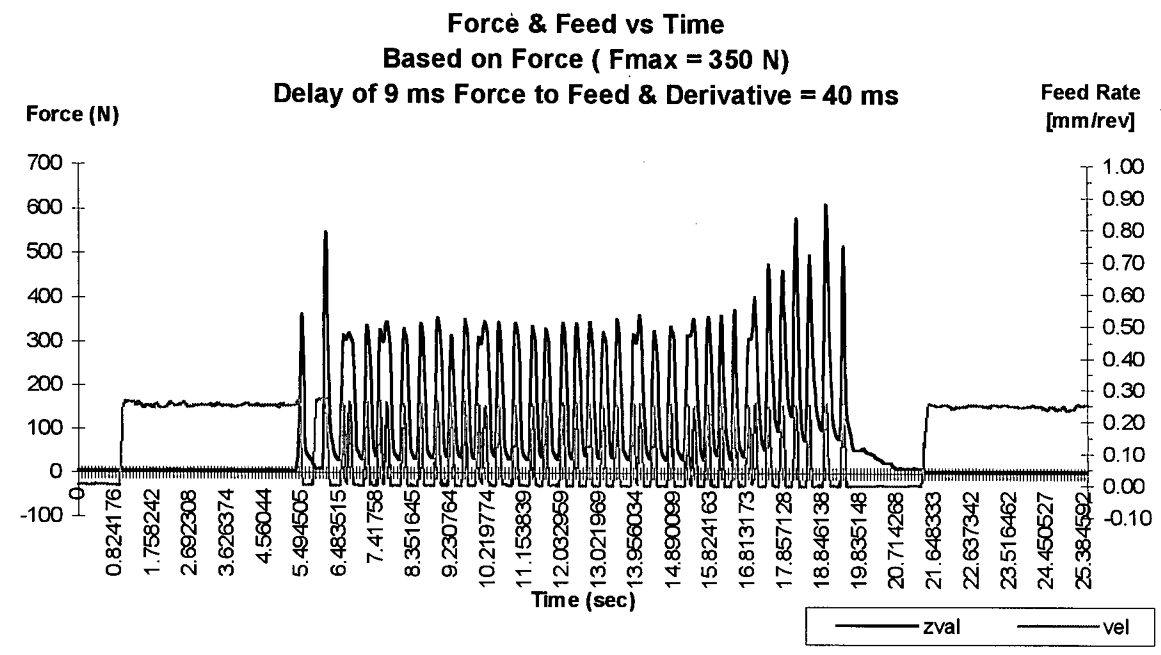

Figure 5.19. Test 005 Experimental Results

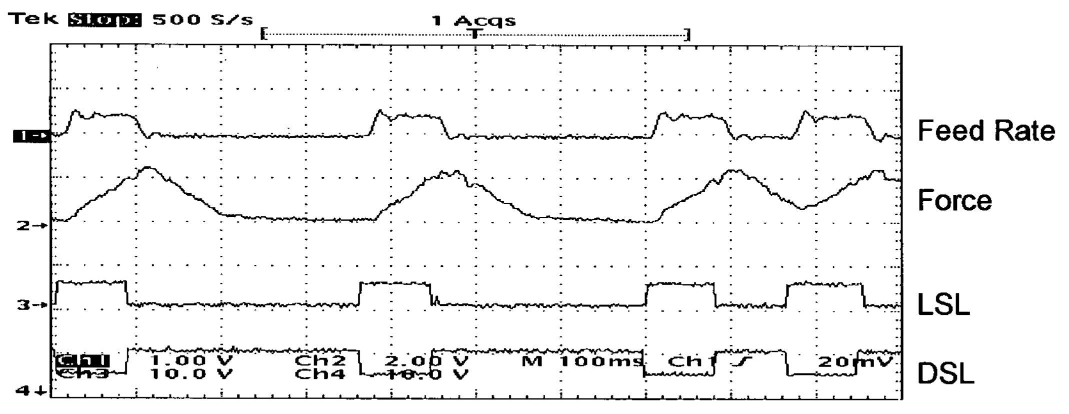

Figure 5.20. Test 005 Experimental Oscilloscope Plot

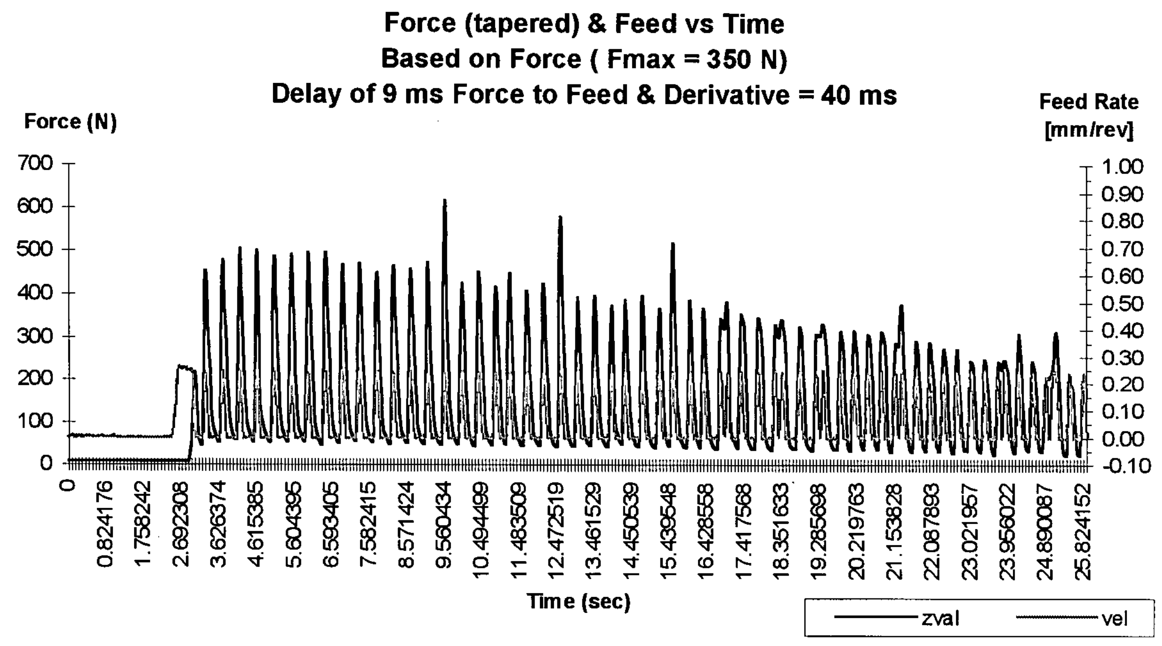

Figure 5.21. Test 006 Experimental Results

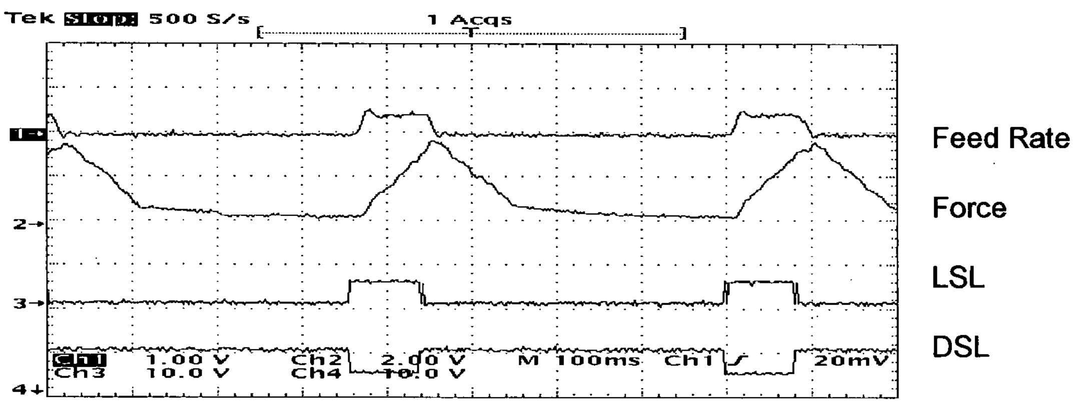

Figure 5.22. Test 006 Experimental Oscilloscope Plot

Figure 5.11 shows Test 001, the system in this case is operating without the constraining influence of distributed force monitoring. A constant force of approximately 520 N at a constant feed rate of 0.25 mm/rev is produced. In Test 002 a maximum force constraint of 350 N was added. This resulted in the system attempting to hold the force below 350 N; however there is significant overshoot to approximately 520 N, as seen in Figure 5.13. In Tests 003 through 005 derivative action was added. The derivative provides a prediction of expected forces, this is expected to improve the response of the system.

Some of the plotted results show high force reading as the cutting tool end its cut, this is due to the tool running into a higher depth of cut during the last few revolutions of the spindle. The cutting passes were conducted as shown in Figure 5.4.

Since the results of the tests were not completely convincing. A new test was carried out which was expected to lead to much larger nominal force values. The test cut (tapered) was as shown below Table 5.3, the maximum depth was 4 mm. At this depth based on the results of the unrestricted cut shown in Figure 5.11, a force of approximately 1000 N would have been expected. However, Figure 5.21 shows that the system responds well. It is not able to hold the 350 N force limit, but does succeed in decreasing the force to approximately 500 N, half the expected force. Clearly in practice a safety factor would be included to prevent damage.

5.4 Comparison of Simulated and Experimental Results

There are differences in the simulation and experimental results as described in the above discussion. These differences can be attributed primarily to backlash, friction and the simulation force model used. The lathe tested, being a mechanical drive system, experiences backlash. The backlash of the system is estimated to be 0.001 to 0.002 inches. The amplitude of displacement about the mean is calculated to be 0.0012 inches, in a typical situation (see Figure 5.13 and Appendix 6). Thus, the backlash and simulation are of the same magnitude. The requirement for a larger amplitude of motor oscillation will, likely, reduce the bandwidth of the system.

In the same manner, the friction forces will add to the torque measured at the motor. Since the required frequency is close to the bandwidth then such an opposing force will have the influence of reducing the response frequency.

The cutting force model is a good approximation of the system, however, it does not produce the exact forces as are measured experimentally. The major approximation is the assumption of a constant edge force component. This causes some errors during the transients.

The response frequency measured from the simulations (5 - 9 Hz) are higher than the experimental (3 - 4 Hz) results. This is a result of the above influences, as well as the remaining error sources which include the feed rate modulation (Figure 4.3), and the electrical time constant for the motor.

It should perhaps be pointed out that, in order to suppress cutting force variation, then the cutting response frequency must be a multiple of the spindle frequency (10 Hz).

When this frequency is maintained, depth of cut is maintained as the path of the cut repeated. The proof is provided in Appendix 6.

5.5 Conclusions and Discussion

The simulated and experimental results confirm the ability of the lower level network to use the distributed force influence (force limit) to control the system connected to it. The difference between the simulated and experimental results are due primarily to the effects of backlash and friction.

The next chapter presents the conclusions of this thesis and recommends future work.

Chapter 6

Conclusions and Recommendations For Future Research

6.1 Conclusions

This thesis has shown that the UBC controller can be effectively used as a basis for a distributed control system. The experimental system consisted of a lower level control network and a higher level communication network. Various options for the software and hardware of both networks were examined. The lower level control network extends the UBC controller's state line to a distributed network, to provide synchronized real-time control. This network was evaluated by testing and simulation to prove its effectiveness. The higher level communication network consisted of an Ethernet network using TCP/IP protocol. This network was installed and confirmed effective.

The advantage of this system is that it offers the ability to provide synchronized real-time control.

6.2 Recommendations For Future Research

The lower level network's control is based on the logic of a single state line. In order to fully utilize the potential of this network, a multiple state line network should be investigated. This would allow better control, as a central computer could communicate with individual processes connected to their own state line.

The higher level network offers the opportunity to monitor the lower level system, to obtain real-time synchronous information. The use of this information for process planning should be investigated. This would allow real-time process planning, which could amend existing plans based on current process information.

Bibliography

[1] Amirouche, F. M., “Computer-Aided Design and Manufacturing”, Prentice-Hall Inc., New Jersey,1993, pp. 257 - 274, 469 - 489.

[2] Ardekani, R., “Integration of Manufacturing Process Control and Optimization with a General Purpose Multi-Processor Controller”, Master of Applied Science in Mechanical Engineering Thesis, University of British Columbia, Vancouver, British Columbia, Canada,1992.

[3] Askin, R. G., Stradridge, C. R., “Modeling and Analysis of Manufacturing Systems”, John-Wiley Inc., USA, 1993, pp. 125 - 148.

[4] Best, W. David, “Method and Apparatus for Self-Timed Digital Transfer and Bus Arbitration”, United States Patent 5,140,680, August 18, 1992.

[5] Blome, W.,"Why Employ an Open Field Bus System”, Industrial Computing, September / October, 1989.

[6] Bus, C., et al, On the Significance of Equivalent Chip Thickness”, Annals of the CIRP, Vol XVIV, 1971, pp. 121-124.

[7] Chen, B. and Chang, Y.,"Robust PI Controller Design of a Constant Turning Force System”, International Journal of Machine Tools and Manufacture, Vol. 31, No. 3, 1991, pp. 257 - 272.

[8] Comer, Douglas, E.,"Internetworking With TCP/IP”, Prentice - Hall, USA, 1995, pp. 577.

[9] Costello, S. and Finster, K.,"Distributed System uses BITBUS Industrial Network for Real Time Control of Battery Chargers”, Micro / Sys Inc. Sales Publication, Glendale, California, USA.

[10] Davidson, R., P. and Muller, N., J.,"Internetworking LANs: Operation, Design, and Management”, Artech House Inc., USA, 1992, pp. 52 - 56, 333 - 358.

[11] Edwards, Harry, J.,"Automatic Controls for Heating and Air Conditioning: Pneumatic - Electric Control Systems”, McGraw - Hill Inc., USA, 1980, pp. 25 - 52.

[12] Eversheim, Walter, et al.,"Production Engineering The Competitive Edge”, Redwood Press Ltd, UK, 1991, pp. 343 - 375.

[13] Fisher, Andrew, D.,"The Development of a Generative Computer-Aided Process Planning Package for the Peripheral Milling Process", Master of Engineering (Mechanical)Thesis, McMaster University, Hamilton, Ontario, Canada,1988.

[14] Fortier, Paul. J.,"Handbook of LAN Technology", McGraw - Hill Inc., USA, 1992, pp. 663 - 669, 287 - 288.

[15] Hordeski, M. F., "CAD/CAM Techniques", Reston Publishing Company Inc., Virginia, 1986, pp. 522 - 531.

[16] Hunter, Ronald, P.,"Automated Process Control Systems: Concepts and Hardware", Prentice - Hall, New Jersey, 1978, pp. 226 - 249.

[17] Imai, S., et al,"Production Management System and Method of Transmitting Date", United States Patent 5,150,288, September 22,1992.

[18] Izikowitz, I. and Rodd, M. G.,"Networking for the Year 2001 - An Irrelevant Issue", Proceedings of the Factory 2001 - Integrating Information and Material Flow Conference, Cambridge, UK, July, 1990, pp. 76 - 80.

[19] Lawrence, Peter, D. and Mauch, Konrad. "Real-Time Microcomputer System Design: An Introduction", McGraw - Hill Inc., USA, 1987, pp. 3 - 25.

[20] Lin, B., S. and Masory, O.,"Optimal, Variable Gain Adaptive Control System for Turning", Transactions of SME: Proceedings of the 15th NAMRC, Vol. 2, 1987, p. 578.

[21] Lukas, M. P., "Distributed Control Systems", Van Norstrand Reinhold Company Inc., New York, 1986, pp. 1 - 16.

[22] Koren, Y.,"Computer Control of Manufacturing Systems", McGraw Hill, New York, 1983.

[23] Koren, Y. and Masory, O.,"Stability Analysis of a Constant Force Adaptive Control System for Turning", Journal of Engineering for Industry, Vol. 107, 1985, pp. 295 - 300.

[24] Mazidi, J. G. and Mazidi, M. A.,"The 80x86 IBM PC & Compatible Computers", Prentice - Hall, New Jersey, 1995, pp. 291, 252, 320.

[25] Meyer, W., "Expert Systems in Factory Management Knowledge-Based CIM", Ellis Horwood, England, 1990, pp. 105 - 113.

[26] Nakayama, K., Arai, M., Takei, K.,"Semi-Emperical Equation for Three Components of Resultant Cutting Force", Annals of the CIRP, Vol 32, No. 1, 1983, pp. 33-35.

[27] Nand, K., Jha,"Handbook of Flexible Manufacturing Systems", Academic Press Inc., USA, 1991, pp. 49 - 59.

[28] Pressman, Rodger, s. and Williams, John, E.,"Numerical Control and Computeraided Manufacturing", John Wiley and Sons, USA, 1977, pp. 281 - 302.

[29] Ranky, P. G., "Computer Integrated Manufacturing", Prentice-Hall International Ltd., UK, 1986, pp. 167 - 185, 305 - 376.

[30] Reardon, Ray. "Future Networks", Blenheim Online Publications, London, UK, 1989, pp. 205 - 222.

[31] Rembold, U., et al,"Interface Technology for Computer-Controlled Manufacturing Processes", Maecel Dekker Inc., USA, 1983, pp. 235 - 277.

[32] Seethaler, R., "A New Contouring Algorithm for the UBC Controller", Master of Applied Science in Mechanical Engineering Thesis, University of British Columbia, Vancouver, British Columbia, Canada,1993.

[33] Schwader, David, W.,"C Programmer's Guide to NetBios, IPX, and SPX", Sams Publishing, Indiana, USA, 1992, pp. 3 - 38.

[34] Ulsoy, A. G., et al,"Principal Developments in the Adaptive Control of Machine Tools", Journal of Dynamic Systems, Measurement, and Control, Vol.105, 1983, pp. 107 - 112.

[35] Yellowely, I., et al,"The UBC Open Architecture Controller", Internal Document - Department of Mechanical Engineering, University of British Columbia, Vancouver, British Columbia, 1993.

[36] Yellowley, I., et al,"Multiple Slave Control", United States Patent 5,519,602, May 21,1996.

[37] Yellowley, I and Ardekani, R.,"The control of Multiple constraints Within an Open Architecture Machine Tool Controller", ASME: Journal of Manufacturing Science and Engineering, Vol 118, May 1996.

[38] Yellowley, I and Pottier, P., R.,"The Integration of Process and Geometry Within an Open Architecture Machine Tool Controller", International Journal of Machine Tools and Manufacture, Vol.34, No. 2, pp. 277-293.

[39] Yellowley, I and Kusiak, A.,"Observations on the use of computers in the process planning of Machine components", Transactions of the CSME, Vol. 9, No. 2, 1986, pp. 70-74.

Appendix 1

STD Computer and Lathe

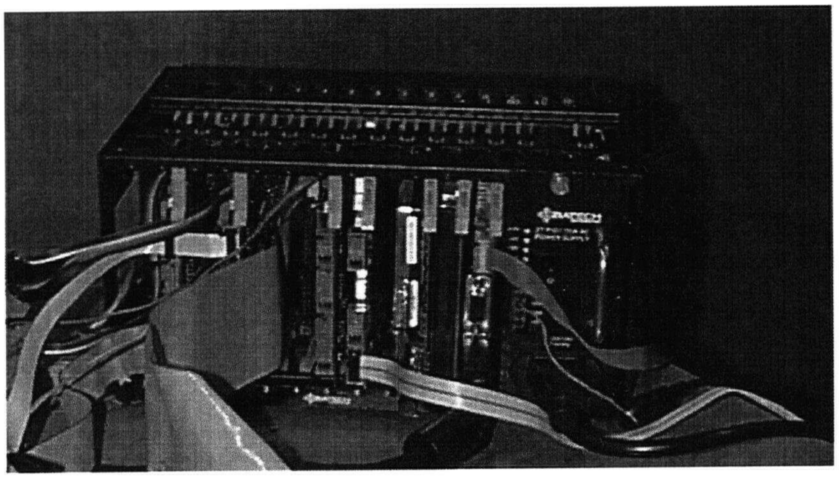

Figure A1.1. Picture of the STD Computer

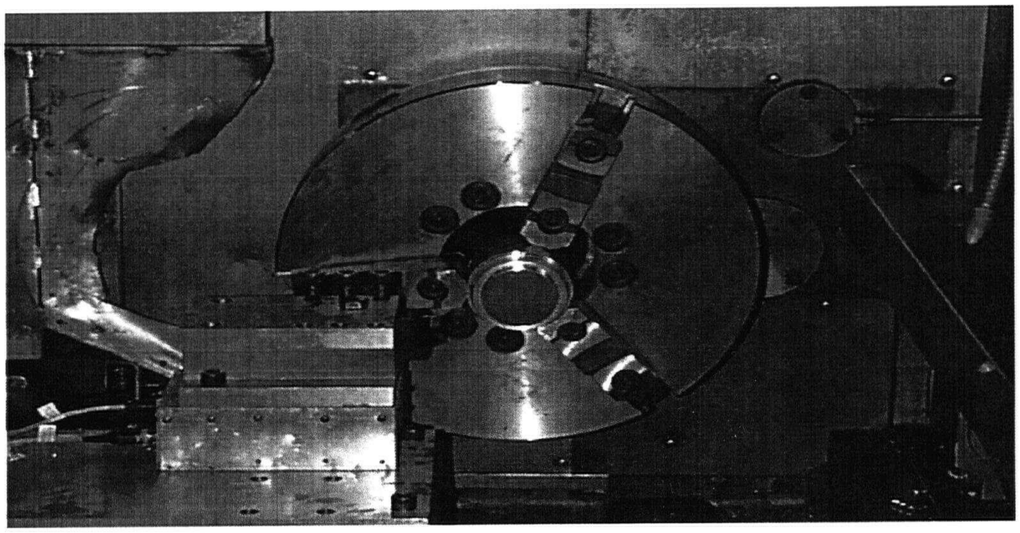

Figure A1.2. Picture of the Lathe

Figure A1.1 is a picture of the STD computer used for the UBC controller. It is a commercially available STD type computer. The computer shown above is a ZT 210 STD32 (9 slot) STD computer, manufactured by Ziatech Corporation. The computer shown above holds the UBC controller which is used to provide the CNC control to the lathe shown in Figure A1.2.

Figure A1.2 is a picture of the spindle of the lathe, showing the work piece and the dynamometer which holds the cutting tool. The lathe used is a Hitachi Seiki Hiturn 1000.

Appendix 2

Parallel Port Distributed Interface Board (DIB)

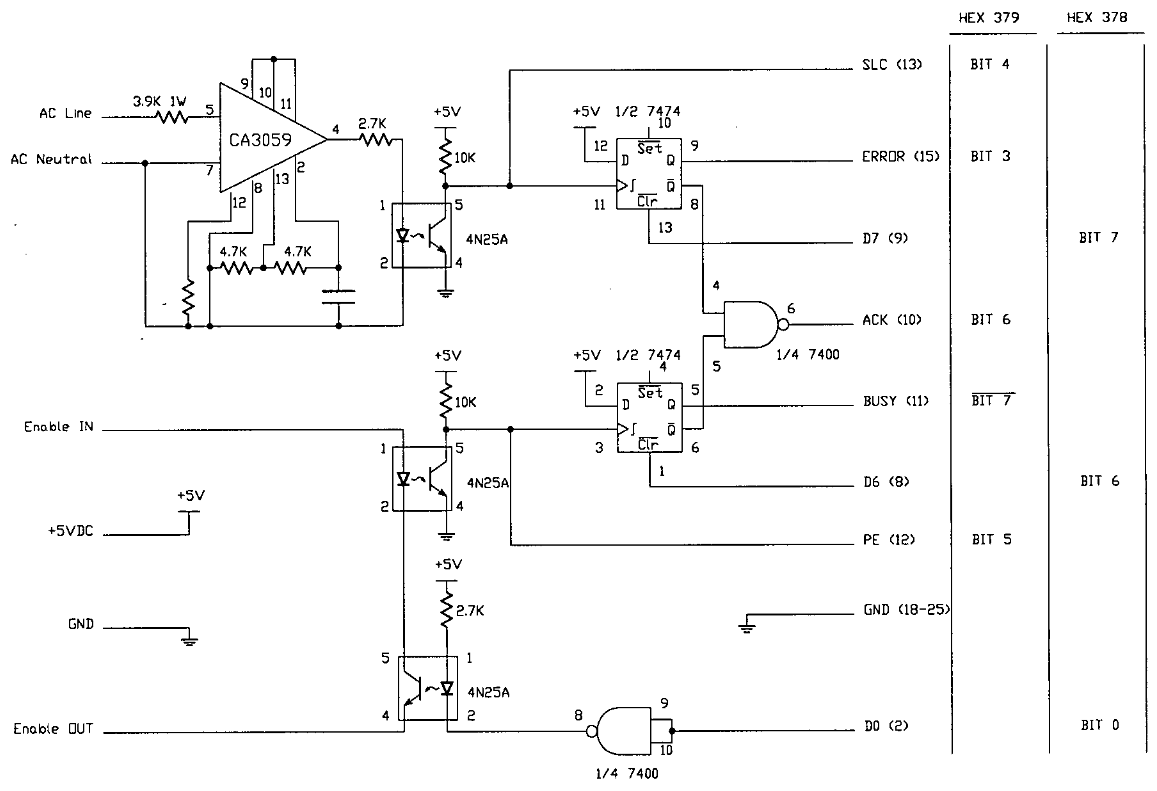

Figure A2.1. Circuit Drawing of DIB

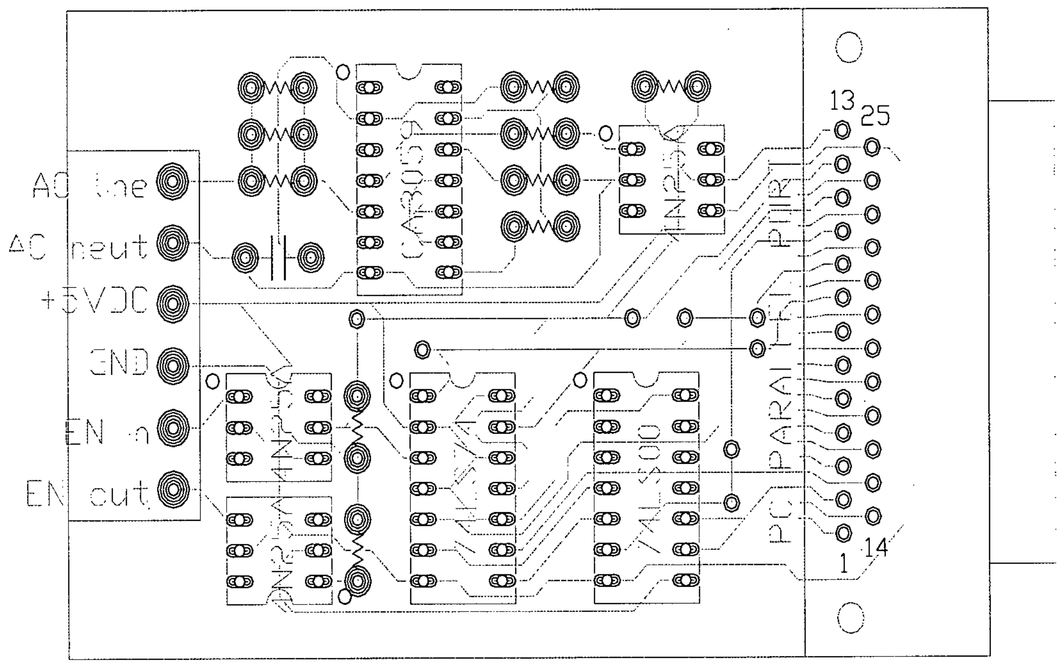

Figure A2.2. Schematic of DIB

Figures A2.1 and A2.2 show the circuit drawing and schematic of the DIB. The function of the DIB is to obtain a timing pulse from the nominal ac signal and set the DSL active (high) or inactive (low).

The nominal ac signal is input to a zero crossing detector which determines when the signal crosses the zero axis. From the zero crossing detector the signal is optically isolated. The isolated signal is then latched with an interrupt request latch which allows the zero crossing to be read by the parallel port. Once the parallel port has read the latched signal, the latch is cleared to enable the next zero crossing to be read.

The DSL is also connected to the DIB which allows the parallel port to both read the status of the DSL and write it active or inactive.

The DSL is powered by 5 volts obtained from the power supply of the computer it is connected to. As a safety measure the nominal ac signal (distributed synchronization line) has been stepped down from 110 volts to 28 volts. For the experimental apparatus this 28 volt ac signal has distributed centrally, which allows shared use of a single transformer.

Figure A2.3 shows a picture of the DIB.

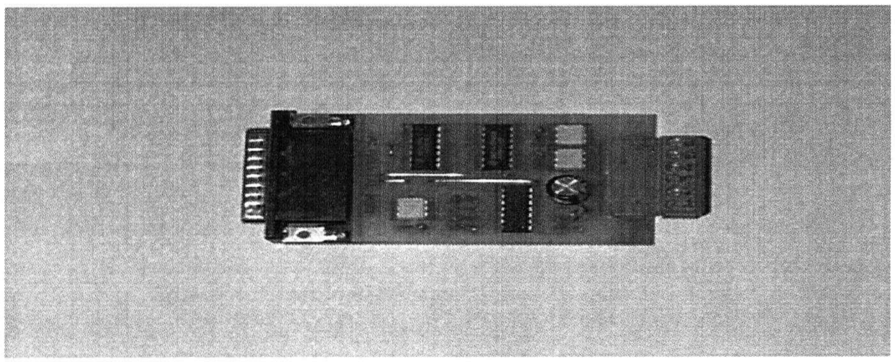

Figure A2.3. Picture of DIB

Appendix 3

State Line Interface Board (SIB)

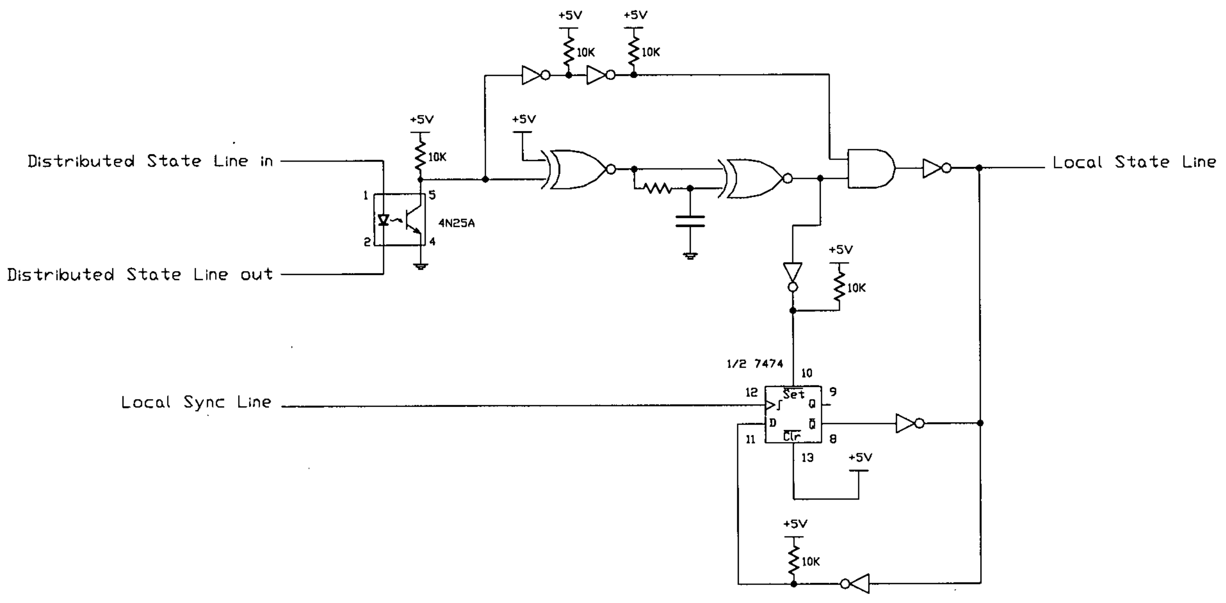

Figure A3.1. Circuit Drawing of SIB

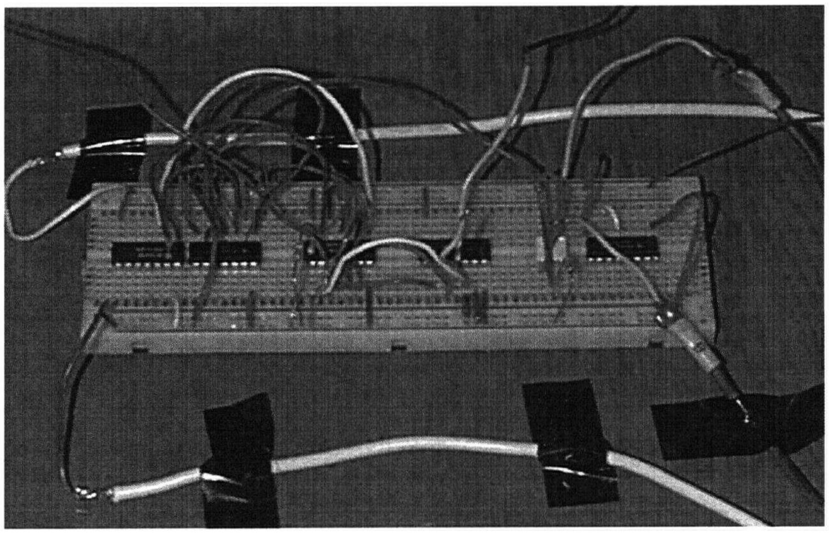

Figure A3.2. Picture of the SIB

Figure A3.1 shows a circuit drawing of the SIB and Figure A3.2 is a picture of the SIB. The function of the SIB is to interface the LSL and DSL. It allows the synchronization line system to maintain its open line characteristics, the DSL being able to pull the LSL inactive. Only one SIB circuit board was required for the experimental apparatus; thus, as shown in Figure A3.2 it was assembled on a bread board.

The DSL is input to the SIB though an optical isolator to isolate it from the LSL. An edge trigger one shot pulse conversion sets an RC time constant. An RC time constant is a circuit which configures a resistor and capacitor to produce a given time in seconds. The SIB's RC time constant is 8.7 ms on triggering the state line system inactive to active and 5 ms on triggering the state line system active to inactive. The difference in the RC timing is due to the input threshold voltage of the exclusive OR gate. The timing provided by the RC time constant is used by an inverter on the LSL. This inverter, inverts the LSL every sync pulse on the local UBC controller, producing a 010101010..... signal. This alternating 01010101..... signal provides the delayed 50 percent feed rate when the LSL is triggered. Beyond the RC time constant the circuit is latched to the state line of the DSL. As previously discussed this inverter feature was included in the SIB to smooth the servo motor response and show it's ability to control and influence the state line system and see its experimental effect.

Appendix 4

Dynamometer A/D Source Code

```
/* PROGRAM TITLE: Dyno A/D Program

Author: E. Allan Mertin          Dated: May 3, 1996

This is a C Program for use with Keithley Metrabytethe A/D board, type DAS_20.
The "C" libraries used by this program are available from the vendor with the
board.  Compiler used is Borland C++ v3.1.

The program processes Dyno force data which if above limits will effect
the state line.  The data can be plotted using Excell.

The program must be used in a project with a DAS20xx.LIB.  The xx in DAS20xx.LIB
is for the memory model size selcected, the large memory model was used when
running this code. The following software switches are to be set in options: set the
memory model as per the "lib" selected[in options/compiler/code gen], no case sensitive
link[in options/linker/settings] and add lib directoryc:\borlandc\tvision\lib[in
options/directories]

NOTE THIS PROGRAM TRIGGERS THE DISTRIBUTED STATE LINE AS
FOLLOWS:

   # 1 - IF THE FORCE IS TOO HIGH THEN THE STATE LINE IS SET INACTIVE
   # 2 - IF THE FORCE DROPS THEN THE STATE LINE IS SET ACTIVE */


#include <stdio.h>
#include <stdlib.h>
#include <dos.h>
#include <conio.h>
#include <stdarg.h>
#include <float.h>
#include <signal.h>
#include <string.h>
```

```
/* global Variables */

#define kd  0.050              /* derivative gain */
#define f_limit   650.0         /* force (N) limit for the machine tool */
#define c_limit   1000.0          /* force above results in corrections */

FILE *outfile;                 /* output file pointer */

/* Function Prototypes */

int initialize(void);
void time_p(void);
int scan( float xval[], float yval[], float zval[], float  vel[], int j);
void prn(float xval[], float yval[], float zval[], float vel[], int j);
void print(float xval[], float yval[], float zval[], float vel[], int j);

das20(int,int *);              /* prototype for the das20 library */

/* START OF MAIN() */

void main(void)
{
  int j, quit, flag_s, count, out;
  char name[20];
  char xx[5];
  float xval[10000], yval[10000], zval[10000], vel[10000];

  int end = 100;          /* TBD data array size based on time */

  j=1;            /* main() loop */
  quit = 0;       /* main() quit Flag */
  out = 1;        /* print control every out-th value printed */
  count = 1;      /* print looper */

  outportb (0x378,1);  /* set state line active */

  clrscr();
  printf("\n\n\n\n\n\n\n\n\n\n\n\n\nEnsure that all equipment is ready then key\n");
  printf("\n\n'return' to start\n");
  gets(xx);
  sscanf(xx,"%s",&xx);

  quit = initialize();
  if (quit == 1)
    {
     printf("\nPROGRAM STOPPED DUE TO A STATE LINE ERROR IN
               INITIALIZE()\n");
```

```
      gets(xx);
      sscanf(xx,"%s",&xx);
     }

  if ((outfile=fopen("fout","w")) == NULL)              /* output file opened */
   {
     printf ("cannot open output file\n");
     exit(1);
   }

  fprintf(outfile,"%s\n"," j vel zval");

  strcpy("fout",outfile);

  while (j < end )
  {
    flag_s = scan( xval, yval, zval, vel, j);

    if (flag_s == 11)
         {
          printf("\nFEED RATE ZERO DUE TO CORRECTED FORCE
                    REDUCTION\nOVERRIDE STOP REQUITED\n");
          gets(xx);
          sscanf(xx,"%s",&xx);
         }

    if (count == out)                      /* control loop for output */
          {
            print(xval, yval, zval, vel, j);    /* put output into output file */
            count = 0;
          }
          count += 1;

    j = j + 1;
  }
    fclose(outfile);                     /* closes output file */
}

/* END OF MAIN() */


/* LISTING OF FUNCTIONS */


/* Print(): prints the output gathered from scan() into an output file */

void print(float xval[], float yval[], float zval[], float vel[], int j)
```

```
 {
  fprintf(outfile," %d %e %e\n",j, vel[j], zval[j]);
  return;
 }



/* INITIALIZE(): initializes the DAS 20 board and the I/O ports */

int initialize(void)
 {

  int mode, data[5], error, state;
  int base = 0x300;          /* base address (hex) */

  mode = 0;                  /* initialization mode */
  data[0] = base;            /* base address */
  data[1] = 7;               /* input gain */
  data[2] = 1;               /* DMA channel */

  if((error = das20(mode,data)) != 0)
   {
    printf("\nMODE 0 ERROR = %d\n",error);
    return 1;
   }

   else
    {
     printf("\nDAS 20 INITIALIZED AT PORT: %3x (hex)\n",base);
     outport(base+0,0);          /* clears the A/D port */
     outport (0x379,0);          /* clears the read parallel port */
     clrscr();                   /* clears the screen */
     outport(0X378,( inport(0x378) I 1)); /* activates the state line */

     printf("\ A/D AND READ PORTS FOR INPUT AND ACTIVATED THE
               STATE LINE \n");
    }

  state = (inport(0x379) & 32);          /* bit 5 (decimal 32) mask */

  if( state == 32 )
   {
    printf("\nSTATELINE IS HIGH - NOT ACTIVE\n");
    return 1;
   }

  if( state == 0 ) printf("\nSTATELINE IS LOW - ACTIVE\n");
  return 0;
 }



/* time_p(): gets time from the A/C pulse using a zero crossing detector, */
/* which reads if the neutral axis has been crossed                        */

void time_p(void)
{
  int quit, readt, j;
  j = 0;
  outportb(0x378,( inport(0x378) & 1));     /* D7(9) clears E(15) in prep for E(15) read
                                              with a bit 0  (decimal 1) mask */
  outportb(0x378,( inport(0x378) I 128));  /* D7(9) activates E(15) in prep for E(15)
read
                                              with a bit 7 & bit 0  (decimal 128)

  /* note: bit 0 is masked high to enable the state line bit 0 high */

  while(j == 0)
   {
    readt = inport(0x379) & 8;
    if (readt == 8)
     {
      outportb(0x378,( inport(0x378) & 1));      /* D7(9) clears E(15) in prep for E(15)
                                                   read with a bit 0  (decimal 1) mask */
      outportb(0x378,( inport(0x378) I 128));   /* D7(9) activates E(15) in prep for E(15)
                                              read with a bit 7 & bit 0  (decimal 128) mask */

                /* note: bit 0 is masked high to enable the state line bit 0 high */

      return;
     }
   }
}



/* SCAN(): gets the looped digital readings from "i" which is         */
/* incrementing for each channel.                                      */

int scan (float xval[], float yval[], float zval[], float vel[],int j)
 {
  int x, i, data[5], mode, flag, error;
  float r_data, mval, mval_o, cval, volt;

  i=1;                    /* loop through x(1), y(2) z(3) & vel(4) */
```

```
while (i < 5)
  {
   mode = 3;
   if(i==4) data[0]= 5;        /* VEL gain range bipolar +/- 0.5v */
       else  data[0] = 1;      /* FORCE gain range bipolar +/- 10v */

   data[1] = i;                /* channel # */

   if((error = das20(mode,data)) != 0)
      {
       printf("\nMODE 3 ERROR = %d  on channel # %d\n",error,i);
       return 1;
      }

   r_data = data[0];          /* reads the data */

   /* printf("\nTHE INPUTED A/D R_DATA = %.3f\nREAD FROM CHANNEL:
                 %d\nTHE VOLTAGE = %.3f\n",r_data,i,volt);  */

  /* A/D board set for 200 mech units per volt */

   if( i == 1 ) xval[j] = (r_data - 1.000021) * 0.977;

   else if( i == 2 ) yval[j] = (r_data - 2.8) * 0.977;

   else if( i == 3 ) zval[j] = (r_data - 1) * 0.977;

   else if( i == 4 ) vel[j] = abs(r_data - 62) * 0.01062 * 1.2;

   else  printf("\n LOOP ERROR SCAN() DATA CORRUPTED, i = %d\n",i);

   i = i + 1;
  }

 if(j == 1) mval_o = 0.0;
   else
      {
       mval_o = max(xval[j-1],yval[j-1]);
       mval_o = max(mval_o,zval[j-1]);
      }
 mval = max(xval[j],yval[j]);           /* determines the max force for */
 mval = max(mval,zval[j]);               /* check against limit */

 cval = mval + kd * (mval - mval_o)/(0.008); /* derivative: 0.008 is the dt based on AC
                                                 period */

 if(abs(mval) < f_limit)
```

```
  {
   if(abs(cval) > c_limit)
    {
     printf("\nMAX FORCE READ IS ABOVE THE LIMIT PROGRAM STOPPED\nTHE
               MAX FORCE READ = %.3f\nTHE MAX LIMIT = %.3f\nTHE
               CORRECTABLE LIMIT = %.3f\n",mval,f_limit,c_limit);
     outport(0X378,0);     /* setting the state line low */
    }
   else  outport(0X378,1); /* activates the stateline */
  }
  else
  {
    outport(0X378,0);  /* de-activates the state line */
    printf("\nFORCE BEYOND CORRECTABLE LIMIT\n");
    return 11;
  }
}


/* END OF FUNCTION LIST                                                   */




/* THE FOLLOWING DOCUMENTATION PROVIDES THE ERROR CODES FOR
   KEITHLEY METRABYTE'S DAS 20 A/D BOARD.


mode 0 . . . . .   0 : no error
           . . . . .   1 : base address out of range <512 or >1008
           . . . . .   2 : interrupt level <2 or >7
           . . . . .  -2 : mode number is not equal to zero (0)
           . . . . .   3 : MDA level not 1 or 3
           . . . . .  -3 : board not present or I/O address wrong


mode 1 . . . . .   0 : no error
           . . . . .   1 : mode out of the range, <0 or >29
           . . . . .  -2 : driver not initialized
           . . . . .  11 : illegal channel
           . . . . .  12 : illegal gain
           . . . . .  13 : illegal queue command


mode 3 . . . . . .   0 : no error
           . . . . .  -3 : hardware error
```

```
..... -1 : mode out of the range, <0 or >29
..... 31 : gain /input range out of range
..... .32 : channel # out of range, for diff 0 - 7
```

Appendix 5

Simulation Source Code

```
/* This code provides for sim of force triggering with a 9 ms delay */

/*
  to run in dos execute a command line:

  sim 1 0.25 600 fmax 20 0.001 0/0.04 0/.040 3 1

  Borland 3.01 c/c++ used to run this program

  to get output look in c:\borlandc\bin\out1
  to plot output use excel                          */

/* Note: This code is modification of Mr. Ramin Ardekani's code, his thesis is
   referenced in the bibliography                                             */


#include "stdio.h"
#include "string.h"
#include "math.h"

/*************************************************************/
/* Force Monitor Simulation program.                         */
/*************************************************************/


#define g1   .7031            /*    Digital filter constants:
#define g2   .6925                  O(k)=g1*e(k)-g2*e(k-1)-g3*O(k-1)
#define g3   .25
                                  g1 = Kp/2**g4        g2 = Kp(1 -dtA)/2**g4
                                  g3 = (1- dtB)        g4 = ceil(ln(Kp)/ln(2))    */


#define Kv   N/A              /* velocity loop gain [V/V]  */
#define Ke   1273             /* encoder gain [pulse/rad]   636.6 2506.377  */
#define Kda  0.0049           /* D/A gain [V/level] */
#define Ka   13               /* Servo Amp Current gain [A/V] */
#define Jeq  0.0013           /* equivalent inertia of system [N.m.s^2] */
#define Kt   1                /* Motor Torque constant [N.m/A] */
#define taum .012             /* Motor mechanical time constant [sec] */
#define B    Jeq/taum         /* Motor friction torque */
#define Hg   N/A              /* Tachometer gain [V/rad/s] */
#define Tmax Imax*Kt          /* Current limit [A] (Imax*Kt/Jeq) */
```

```
#define Vcmax 10                /* D/A voltage output limit */
#define l   0.01                /* ball screw pitch [m] */
#define Mub  0.01               /* ball screw guide coeff of friction */
#define dm   0.033              /* mean diameter of the ball screw [m] */
#define lambda -6.0             /* rake angle deg */
#define psi   -3.0              /* approach angle deg */
#define Mu     0.6+0.005*(lambda)  /* effective coeff of friction */
#define R      0.8              /* tool nose radius [mm] */
#define r      (Mu-tan(lambda))/(1+Mu*tan(lambda))  /* force ration Ft/Fv */
#define K1     836              /* Cutting Pressures [N/mm^2] */
#define K2     55               /* [N/mm] */

#define Pul    1049.8685        /* 1574.803 [Pulses/mm] */
#define i_1rev floor(60./(RPM*0.001)) /* i/rev   dt = 0.001 */
#define i_prev floor(60000./RPM) /* i - for previous spindle revolution */
#define Size  4000              /* Array size required */
#define PI    3.141592654

float Tf,T;                     /* Friction and total torque [N.m] */
float Fmax;                     /* Force constraint [N] */
float Imax;                     /* Current limit [A] */
float dPv;
float Step = 0.0;               /* step change in DOC [mm] */
float DOC = 2.0;                /* depth of cut [mm] */
float I;                        /* Current output */
float Vc[2];                    /* Command velocity [V] */
float Vo[2], dVo[2];            /* Output velocity arrays  [rad/s] */
float Pc[3], Po[Size];          /* Command, output and Position error */
                                                                    [BLU]
*/
float F[2];                     /* Force array [N] */
float dF;                       /* rate of change of force */
float dF8;                      /* 8th rate of change of force */
float Pstep;                    /* location of step change in DOC */
int Tstp;                       /* Time of step change in DOC [S] */
float Kd_pos, Kd_neg;           /* gain of derivative control [S] */
float RPM;                      /* spindle speed [RPM] */
float s;                        /* feed [mm/rev] */
float dt;                       /* sampling time [S] */
float Fa;                       /* axial force [N] */
int output;                     /* data output interval pts/sec */
int Out;                        /* data output interval ms/pt  */
FILE *outfile;
```

```
Print (i)
int i;
{
        fprintf(outfile,"\n %.4f %e %e",(i*dt),Pc[2],Po[i]);
        fprintf(outfile," %e %e %e %e
%d",F[1],Vc[1]*(Ke/Pul)/(RPM/60),Vo[1]*(Ke/Pul)/(RPM/60),I,i);
}

/**************************************************************
 Initialize arrays with initial conditions
     - no motion for the first revolution of the spindle
       eg 75 ms for 800 RPM
**************************************************************/
Init ()
{
        int i;
        int count=0;
        for (i=0; i<2; i++) {
                F[i]=Vc[i]=Vo[i]=dVo[i]=Pc[i]=I=0.0;
        }
        Pc[3] = 0;
        for (i=0; i<i_1rev; i++) {
                Po[i]= 0.0;
                if (count == Out) {
                                Print(i);
                                count = 0;
                }
                count += 1;
        }
}

/****************************************************************
 Finds the torque
****************************************************************/

float Torque ()
{
        float T;
        T =  Ka*Kt*(Vc[1]);  /* - Vo[1]*Kv*Hg);*/
        if (abs(T) > Tmax) {         /* current limit */
                if (T < 0)  T = -Tmax;
                else        T = Tmax;
        }
        return T;
}
```

```
/*****************************************************************
 Routine to implement current limit
 *****************************************************************/
Chk_Curr ()
{
        float I;
        I = Ka*(Vc[1]);  /* - Vo[1]*Kv*Hg);*/
        /*if (abs(I)>Imax) {
                if (I<0) Vo[1] = ((-Imax/Ka)-Vc[1])/(-Kv*Hg);
                else    Vo[1] = ((Imax/Ka)-Vc[1])/(-Kv*Hg);*/

          if (abs(I)>Imax) {
                if (I<0) Vo[1] = Vo[0]+Imax*Kt/Jeq*dt;
                else    Vo[1] = Vo[0]-Imax*Kt/Jeq*dt;
        }
}

/*****************************************************************
 Routine to calculate the main cutting force at current
  depth of cut and feed.

 *****************************************************************/
float Force (depth,feed)
float depth, feed;
{
        float Fv;  /* main cutting force [N] */
        float Le;  /* active cutting edge length [mm] */

        if(feed <=0) Le = 0;
          else  Le = depth-R+R*PI/2+feed/2;

        Fv = K1*depth*feed+K2*Le;
        return Fv;
}

/*****************************************************************
 Routine to simulate i-th interval in a turning process
 with force constraint
 *****************************************************************/
Simulate (i,j)
int i,j;
{
        char xx[5];

        float Fvar1, Fvar2;   /* forces encountered during a transient */
        float Perri, Perri_1; /* position error in BLU */
        float Fss;            /* force during steady cutting */
        float Kd;             /* derivative control gain */
if (i == Tstp) Pstep = Po[i-1]; /* start of the step change in DOC */

if ((i >= Tstp) && (i <= (Tstp+i_1rev))) {
        Fss = Force (DOC,Po[i-1]-Po[i-i_1rev]);
        Fvar1= Force (DOC+Step,Po[i-1]-Pstep);
        Fvar2 = Force(DOC+Step, Po[i-1]-Pstep);
        F[1] = Fss+Fvar2-Fvar1;
        if (i == Tstp+i_1rev) DOC += Step;      /* adjust DOC after 1 rev. */
}
else {
        F[1] = Force(DOC,Po[i-1]-Po[i-i_1rev]);
}
Fa = r*F[1]*cos(psi);                    /* axial force [N] */
Tf = Fa*(dm/2)*((l+PI*Mub*dm)/(PI*dm-Mub*l)); /* Friction torque [N.m] */

if (j == 7)  dF8 = (F[1] - F[0])/dt;             /* check sign of slope */

if(j == 0){ dF = dF8;
          j = -16;}

if (dF < 0) Kd = Kd_neg;                  /* adjust gain based on sign */
  else    Kd = Kd_pos;

F[0] = F[1];

if ((Kd*dF) <= (Fmax-F[1]))
        Pc[2] = Pc[1]+dPv;            /* Command Position [mm] */
else
        Pc[2] = Pc[1];        /* Pc[i] = Pc[i-1] */
j=j+1;

Perri = (Pc[1] - Po[i-1])*Pul;    /* Position error in pulses (BLU's) */
Perri_1 = (Pc[0] - Po[i-2])*Pul;  /* Position error in pulses (BLU's) */
Vc[1] = (g1*Perri-g2*Perri_1-g3*Vc[0])*Kda;    /*  [V]  */

  /* extrapolated acceleration */
if (abs(Vc[1]) > Vcmax) {              /* D/A output limit */
        if (Vc[1] < 0)  Vc[1] = -Vcmax;
        else          Vc[1] = Vcmax;
}

dVo[0] = Torque()/Jeq;
Vo[1]   = Vo[0] + dVo[0]*dt;         /* predicted velocity */
Chk_Curr();                          /* check current limit */

dVo[1]   = (Torque()/Jeq)*(Vc[1]); /* - Vo[1]*Kv*Hg); /* actual accel. */
dVo[0] = (dVo[0] + dVo[1])/2;        /* mean value */
```

```
        Vo[1]   = Vo[0] + dVo[0]*dt;        /* actual value */
        Chk_Curr();                        /* check current limit */
        Vo[0]  = (Vo[0] + Vo[1])/2;        /* mean value */
        Po[i]   = Po[i-1] + Vo[0]*dt*Ke/Pul;  /* actual value */
        I = Ka*(Vc[1]);                   /* - Vo[1]*Kv*Hg);*/

        Vc[0] = Vc[1]/Kda;             /* check simulations with this change  */
        Vo[0] = Vo[1];                 /* otherwise Vc[0] = Vc[1] ! */
        dVo[0] = dVo[1];
        Pc[0] = Pc[1];
        Pc[1] = Pc[2];
}

main(argc,argv) /* outfile, RPM, Tstp, Gain */
int argc;
char *argv[];
{
        int i, j, count;
        char name[20];

        count = 0;
        j = 0;

        if (argc != 11) {
                printf ("input: outfile, RPM, Tstp, Gaind.\n");
                exit(0);
 }

 RPM = atof(argv[3]);                /* Spindle speed [RPM] */
 s = atof(argv[2]);                  /* feed [mm/rev] */
 dt = atof(argv[6]);                 /* sampling time [S] */
 Kd_pos = atof(argv[7]);             /* derivative gains [S] */
 Kd_neg = atof(argv[8]);             /* derivative gains [S] */
 Tstp = floor(atof(argv[9])/dt);     /* i corresponding to start time of step */
 Out  = atoi(argv[10]);              /* data loging interval */
 Fmax = atof(argv[4]);               /* force limit */
 Imax = atof(argv[5]);               /* current limit [A] */
 dPv = (s*RPM*dt)/60.;               /* Position ramp [mm/i] */

 strcpy (name, "out");
 strcat (name,argv[1]);

 if ((outfile=fopen(name,"w")) == NULL) {
  printf ("cannot open output file\n");
  exit(1);
 }

        fprintf(outfile,"%s"," Time Pc Po");
        fprintf(outfile,"%s\n"," Force Vc Feed I i");

        output = (RPM/60)*3;            /* 3 times highest observed frequency */
        output = floor(output/(dt*1000)); /* adjust for sampling rate */
        Init ();

        for (i=i_1rev; i<Size; i++) {  /* start simulation at */
                Simulate(i,j);
                if (count == Out) {
                        Print(i);
                        count = 0;
                }
                count += 1;
        }

        fclose(outfile);
        exit(0);
}
```

Appendix 6

Calculations

Calculation of Amplitude of Displacement

$V = Vo + Vn \times \sin(wt)$

where:

V:	velocity
Vo:	initial velocity (assumed zero)
Vn:	current Velocity (0.1 mm/rev)
w:	frequency

$$V = 0 + \frac{0.1 \times 600}{60} \times \sin(5 \times 2\pi \times t)$$

$$V = 1 \times \sin(31.4 \times t)$$

$$X = \int \sin (31.4t)\, dt$$

$$X = \frac{-1}{31.4} \cos(31.4t) \Big|_{0}^{0.001}$$

the amplitude of displacement $= \frac{1}{31.4} = 0.0318$ mm

$= 0.0012$ inches

Calculation of Spindle Frequency Influence on Feed Velocity

$V = Vo + Vn \times \sin(wt)$ where:

V:	velocity
Vo:	initial velocity
Vn:	current Velocity
w:	frequency

The objective is to find a likely value of frequency (w) so that force variation is zero.

Assume:

F = k (X / current chip thickness) = spindle feed (rad/sec)

$$Xn = \int_{(n-1)t}^{nt} (Vo + Vn \times \sin(wt))dt \qquad \text{where:} \qquad t = \underline{2\pi}$$

$$= Vo \times t - \frac{Vn \times}{w} \cos(wr) \Big|_{(n-1)t}^{nt}$$

$$= Vo \times t - \frac{Vn}{w} (\cos(nwt) - \cos w(n-1)t)$$

$$= tVo + \frac{2Vn}{w} (\sin \frac{(2n-1)}{2} tw \sin \frac{(wt)}{2})$$

the requirement is:

$$\frac{(2(n-1))}{2} tw = 0, \pi, 2\pi \,\ldots\ldots\ldots\, n\pi \qquad \text{or} \qquad \frac{wt}{2} = 0, \pi, 2\pi \,\ldots\ldots\ldots\, n\pi$$

$w = [F/\pi]\,[1/2n-1]\,[0, \pi, 2\pi \,..........\, n\pi]$

$$n >= \frac{1}{(2n-1)}\,[\,0, F, 2F \,.........\, nF] \qquad (A6.1)$$

or

$$w = F(\,0, 1, 2.............n) \qquad (A6.2)$$

(A6.1) is a special case which will occur for specific values of w,

(A6.2) is the general solution.