DISTRIBUTED CONTROL SYSTEM FOR CNC MACHINE TOOLS

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

BY

FURKAN A. KANBUROĞLU

IN PARTIAL FULFILMENTS OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

JUNE 2009

Approval of the thesis:

DISTRIBUTED CONTROL SYSTEM FOR CNC MACHINE TOOLS

submitted by **FURKAN A. KANBUROĞLU** in partial fulfillment of the requirements for the degree of **Master of Science in Mechanical Engineering, Middle East Technical University** by,

Prof Dr. Canan ÖZGEN _____

Dean, **Graduate School of Natural and Applied Sciences**

Prof Dr. Suha ORAL _____

Head of Department, **Mechanical Engineering**

Asst. Prof. Dr. Melik DÖLEN _____

Supervisor, **Mechanical Engineering Dept., METU**

Asst. Prof. Dr. A. Buğra KOKU _____

Co-Supervisor, **Mechanical Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. S. Engin KILIÇ _____

Mechanical Engineering Dept., METU

Asst. Prof. Dr. Melik DÖLEN _____

Mechanical Engineering Dept., METU

Asst. Prof. Dr. Buğra KOKU _____

Mechanical Engineering Dept., METU

Asst. Prof. Dr. İlhan KONUKSEVEN _____

Mechanical Engineering Dept., METU

Assoc. Prof. Dr. Veysel GAZİ _____
Ectrical and Electronics Engineering Dept., TOBB ETU

Date: 11.05.2009

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: Furkan A. KANBUROĞLU

Signature:

ABSTRACT

DISTRIBUTED CONTROL SYSTEMS FOR CNC MACHINE TOOLS

Kanburoğlu, Furkan A.

M.S. Department of Mechanical Engineering

Supervisor: Asst. Prof. Dr. Melik DÖLEN

Co-supervisor: Asst. Prof. Dr. Buğra KOKU

June 2009, 136 Pages

"Numerically Controlled" (NC) machine tools, which are automatically operated by encoded (digital) commands, are capable of machining components with quality and quantity. Manufacturing industry heavily depends on these machines. Many different control architectures have been adapted in today's CNC technology. Centralized control system is quite popular in industry due to its ease of implementation. If the number of controlled axes on a CNC machine tool (>3), increases so does the computational burden on the central processors. Hence, more powerful processors are needed. An alternative architecture, which is not commonly used in CNC technology, is the decentralized (distributed) control. In this topology, the tasks handled by the distributed controllers that are interconnected to each other by a communication network. As the need arises, a new controller can be added easily to the network without augmenting the physical configuration. Despite its

attractive features, this architecture has not been fully embraced by the CNC industry. Synchronization among the axes in the coordinated motion is proven to be quite challenging.

In this thesis, alternative distributed controller architecture was proposed for CNC machine tools. It was implemented on a 3-axis CNC milling machine. Open-loop control performance was investigated under various conditions. Different communication protocols along with different physical communication interfaces and a number of controller hardware were devised. An industry-standard network (RS-485) was set up by interconnecting these distributed controllers. Different data transmission protocols were devised in order to establish appropriate communication methods. Also, computer software (a.k.a. graphical user interface), which can coordinate the interconnected controllers, interpret NC part programs and generate reference position data for each axis, was designed within the scope of this thesis.

Keywords: Distributed Motion Control, Centralized Control, Networked Control System, Communication Protocols, CNC Machine Tools.

ÖZ

SAYISAL DENETİMLİ TAKIM TEZGAHLARI İÇİN DAĞITIK HAREKET KONTROL SİSTEMİ

Kanburoğlu, Furkan A.

Yüksek Lisans, Makina Mühendisliği Bölümü

Tez Yöneticisi: Yard. Doç. Dr. Melik DÖLEN

Yardımcı Tez Yöneticisi: Yard. Doç. Dr. Buğra KOKU

Haziran 2009, 136 Sayfa

Bilgisayar denetimli (CNC) takım tezgahları sayısal olarak kodlanmış komut dizinlerini otomatik olarak çalıştırıp, yüksek miktar ve kalitede parça işleyebilen makina sistemleridir. Bu nedenle, bu tip makinalar üretim endüstrisi için vazgeçilmez niteliktedir. Gününüz CNC sistemlerinde çeşitli kontrol mimarileri kullanmaktadır. Bunlardan biri de merkezi kontrol sistemleridir ve endüstriyel uygulamalarda kolay tatbik edilebildiklerinden çokca kullanılmaktadır. Hiç şüphe yok ki CNC takım tezgahlarında denetlenecek eksen sayısı arttıkça (>3) merkezi kontrol sisteminin üzerindeki yük de artmaktadır. Bunun yerine, dağıtık kontrol sistemleri, CNC işlem merkezlerinde bir alternatif olarak tercih edilebilir. Bu kontrol sistemi yapısında işlemler dağıtık denetleyicilerle yapılmaktadır. Bu dağıtık denetleyiciler bir iletişim ağı ile birbirlerine bağlanarak, tanımlanan bir iletişim

protokolüyle haberleşmektedirler. Eğer gerekli görülürse, yeni bir denetleyici bu iletişim ağına kolayca dahil edilebilir ve bu ekleme sistemin bütününe (başka bir deyişle diğer denetleyicilere) ek bir yük getirmez. Tüm bu olumlu özelliklerine karşın, bu yapı CNC üreticileri tarafından pek tercih edilmemektedir. Bu sistemde, koordinasyon içinde hareket ettirilmesi gereken eksenlerin eşgüdümü çok da kolay olmayan bir konudur.

Bu tez çalışmasında, bilgisayar destekli takım tezgahları için alternatif bir dağıtık kontrol sistemi önerilmiştir. Önerilen bu sistem, 3 eksenli bir freze tezgahına uygulanmış, çeşitli koşullar altında bu sistemin açık döngü kontrol performansı incelenmiştir. Çalışma esnasında değişik iletişim protokolleri ile kontrol donanımları geliştirilmiştir. Ortaya konan kontrol donanımları birbirlerine elektronik ileitşim yöntemleriyle bağlanarak, kontrol ağları oluşturulmuştur. Tez çalışması kapsamında, farklı protokoller denenerek, en uygun iletişim yöntemleri araştırılmıştır. Ayrıca, denetleyiciler arasında kordinasyonun sağlaması, kullanıcı tarafından girilen parça programlarının yorumlanması ve referans konum komutlarının üretilmesi için bir bilgisayar yazılımı (bir başka adıyla grafik kullanıcı arayüzü) geliştirilmiştir. Ayrıca, bu bilgisayar programını kullanılarak, çeşitli interpolasyon algoritmaları da sınannıştır.

Anahtar Kelimeler: Dağıtık Hareket Kontrolü, Merkezi Kontrol, Şebekelenmiş Kontrol Sistemleri, İletişim Protokolleri, CNC Takım Tezgahları.

Dedicated to a brand new life.

ACKNOWLEDGEMENTS

I express sincere appreciation to Asst. Prof. Dr. Melik Dölen and Asst. Prof. Dr. Buğra Koku for their perfect guidance, endless encouragement, trust and support throughout this thesis.

I wish to thank to my friends Onur YARKINOĞLU, Ergin KILIÇ, my colleagues Bilal BAYRAM, Çağrı İLÇE, Tacettin ÖZTÜRK, Ufuk ÖZGEN for their support, friendship and moral assistance.

I am most grateful to my beloved life partner Tuğba MENZİR for her extraordinary, never ending support, tolerance and understanding. Also, I offer sincere thanks to my family for their endless support and encouragement.

Finally, I want to thank SPARC GROUP for financial and technical support for this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

τ_k	Time Delay (µsec)
τ_k^{sc}	Communication delay between sensor and actuator (µsec)
τ_k^{c}	Computational delay in the controller (µsec)
τ_k^{ca}	Communication delay between controller and actuator (µsec)
V_f	Feed speed (µsec)
Z_n	Number of teeth (-)
f	Feed rate (m/min)
P_c	Power demand in the cutting operation (kW)
a_e	Depth of cut (mm)
a_p	Width of cut (mm)
k_c	Cutting force per mm^2 (N/mm^2)
η	Machine efficiency (-)
h_m	Average chip thickness (mm)

CHAPTER 1

INTRODUCTION

1.1 Introduction

"Numerically Controlled" (NC) machine tools, which are automatically operated by encoded (digital) commands, are capable of machining components with quality and quantity. Early machine tools were programmed through a program stored on a (punched) tape while their operating parameters could not be easily changed via a user friendly interface. Later, part programs were transferred to the machine via a serial communication protocol from a mainframe computer. These machine tools with hardwired control systems were eventually evolved in time through the advancements in digital electronics. The introduction of microcomputers can be considered as a breakthrough for NC machine tools and has transformed them into "Computer Numerically Controlled" (CNC) machine tools where a number of onboard computers can be incorporated to the system. As a consequence, the manufacturing industry, which heavily depends on this technology for high-quality machining, was radically changed.

Besides superior (tool-position) controls, advancing computer technology has also offered many versatile utilities such as program editing, file/data management, and on-board diagnostics. Furthermore, computer-aided design tools enabling sophisticated machining simulation are also integrated into the CNC units of

contemporary machine tools. Facilitating NC part program language, these tools have eventually evolved to become an indispensible part of manufacturing industry.

Many different control architectures have been adapted in today's CNC technology. Among these architectures, centralized control system, which is illustrated in Fig. 1.1, are quite popular in industry due to its ease of implementation.

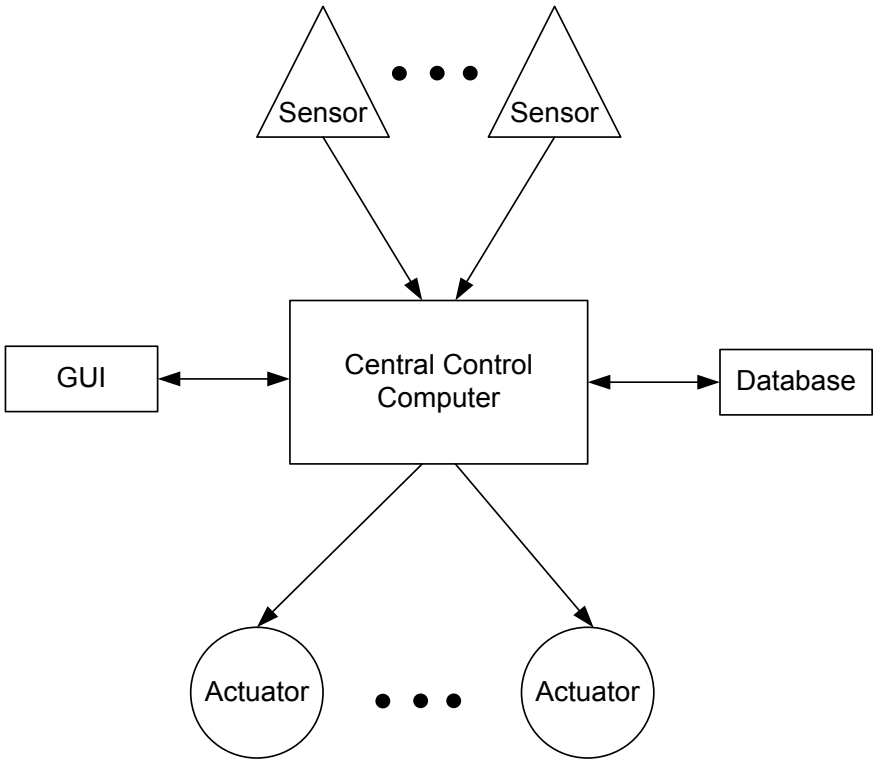


Figure 1.1 Centralized controller architecture

In such systems, a central processor (usually a DSP) not only generates proper position commands for each axis but also performs relevant control computations in real-time. It is obvious that as the number of controlled axis on a CNC machine tool increases (>3), the computational burden on the central processor swells considerably. Hence, more powerful processors are then needed to handle the

resulting computational load. Despite the fact that fast processors (with multiple cores) are becoming widely available in the market (at reasonable prices), such an approach is not economically sound owing to the fact that massive development efforts (i.e. PCB and firmware design) are usually involved in the design of the overall system as the needs (and performance expectations) of manufacturing industry grow on a continuous basis.

An alternative architecture used in CNC technology is a decentralized (distributed) one as shown in Fig. 1.2. In this topology, the tasks handled by the system are distributed among various nodes of the network. Each node deals with a particular operation using a dedicated processor with relatively modest resources. The nodes are connected to each other by a high-speed serial bus such as RS-485, CAN, Profibus, Ethernet, etc. Usually, coordination as well as communication of these nodes are performed by a master processor.

As the need arises, a new node can be added easily to the network without augmenting the physical configuration. Despite its attractive features, this architecture has not been fully embraced by the CNC industry owing to the fact that the synchronization among the axes in the coordinated motion is proven to be quite challenging. Furthermore, for electronic gearing applications where one or more axes is to follow the motion of a master axis accurately, the bus system cannot effectively transfer huge amount of (redundant) data needed to be exchanged among "electronically" coupled axes.

It is critical to note that the brushless DC motors and their drives are the bread-and-butter of motion control applications. Modern motor drive systems do posses the ability to control the position of its servomotor accurately provided that appropriate

sensors are connected to the drive system. If a dedicated controller for each motor drive system is deployed to handle not only the data trafficking but also the synchronization of the axes (via appropriate abstraction), the design of a decentralized control system, which meets the expectations of the industry, might become economically feasible.

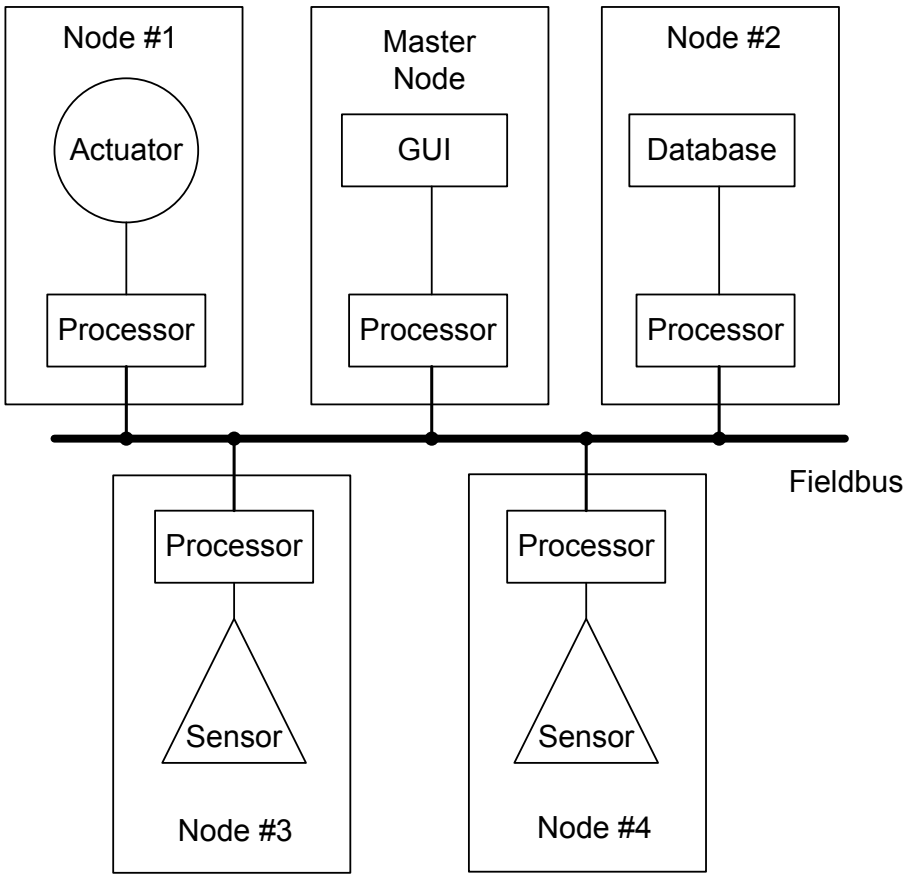


Figure 1.2 Distributed control architecture

1.2 Objective of Thesis

The motivation of this work is to propose a distributed CNC machine controller architecture, where each actuator of the CNC machining center is controlled by

relatively small, but identical distributed controllers. As mentioned earlier, many of the commercial CNC units employ powerful microcomputers (but expensive) or embedded PCs which are equipped with industry-standard serial communication interfaces. Furthermore, such systems employ a custom-tailored graphical user interface incorporating a number of utilities. Unfortunately, despite their performance, their architectures do not provide flexibility for small-scale applications where economical solutions are desperately sought. On the other hand, employing modest microcontrollers and a standard PC, one can develop an alternative distributed CNC machine controller with considerable graphical user interface. Consequently, the objective of this goal is to apply the developed systems on a three-axis machine tool and is to evaluate the performance of the resulting system in a number of machining operations.

1.3 Outline of the Thesis

This outline of this thesis is as follows: In Chapter 1, a brief introduction on CNC machine control systems and their expanded use in manufacturing industry are presented. In Chapter 2, a survey of research efforts on distributed control systems are given and the properties of networked control systems concepts are elaborated. In Chapter 3, a brief summary of the serial communication systems, which lie at the heart of a distributed motion control system, is given. Then, detailed information about (prospective) serial-communication interfaces is presented. Devised communication protocols are also discussed and evaluated in that chapter. In Chapter 4, a distributed-control architecture is developed and corresponding firmware is discussed in detail. Chapter 5 deals with the implementation and performance evaluation of developed distributed controllers to an old CNC milling

machine, which formerly utilizes traditional centralized controller architecture, are represented. In Chapter 6, Graphical User Interface and Machine DLL, a graphical user interface, which is subjected to operate distributed control system and DLL file that can be considered as device driver of distributed controllers, are presented in detail. In Chapter 7, conclusions on results and further research directions are presented.

CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

A major trend of industrial control systems is to integrate computing, communication and control into different levels of machine [1, 5, 9, 12, 13 and 32]. Traditional communication architecture, which has been implemented in industry for decades, utilizes a centralized controller unit that has the ability to integrate sensors and actuators. Unfortunately, expanding physical setups and expecting enhanced functionality oftentimes pushes such systems to their limits [3, 17, 20, 21 and 23].

Today, industry requires modularity, decentralization of control, integrated diagnostics, quick and easy maintenance and low cost to compel their expanding manufacturing capability [22, 29 and 30]. Introduction of network communication systems can reduce complex hardware architectures, improve flexibility and scalability and raise the idea re-configurability and functionality [6, 10, 17 and 25]. Changing communication architecture introduces the time delays which are directly related to time required for signal conditioning and information processing [2]. Such characteristics of time delays can be constant, bounded or random [35]. Basically, it depends on physical attributes of the communication interface, number of nodes connected to the network and developed communication protocol [26]. However, analyzing of networking delays is a diverse and progressive research area.

Most of the Networked Control Systems research has been focused on two areas: communication technology and controller design. Simply put, a message transfer protocol guarantees the network quality where a controller responsible for quality of performance.

2.2 Distributed Control Systems

Distributed control systems (DCS) are essentially spatially networked embedded nodes that are interconnected by means of wired and/or wireless communication infrastructure and protocols interacting with the environment and each other. Examples of distributed control systems include industrial automation, building automation, office and home automation, intelligent vehicle systems and advanced aircraft and spacecraft systems [14, 33 and 34]. Common feature of these applications is that large number of devices interconnected to each other over a relatively large area to perform a certain task. It is obvious that the processing load on a centralized control unit is large if all computations are to be handled by this unit. Also, a large amount of wiring has to set up to connect all the units to centralized processor.

Distributed control systems offer an alternative solution. In literature, these systems are also referred as Networked Control Systems (NCS) where different modules are able to communicate each other [54]. The development of appropriate control architectures and associated controller design algorithms for such systems are crucial for success since the sensors and actuators are frequently distributed over a large area [19, 41].

Distributed control systems are commonly utilized in complex engineering systems [11, 18 and 24]. Communication networks and sensor networks are considered as the essential elements of the main infrastructure.

G. Wyeth at al. developed a distributed digital controller for PUMA 560 six DOF industrial robot arm [17]. In their study, the centralized control system developed consists of a controller for each joint of which are networked together via a CAN Bus. A PC provides a user interface for proposed control system. They utilize a DSP for each controller in order to perform desired joint control operations of the robot arm. In fact, the CAN bus eliminates complex wiring and increase functionality of the joint controllers. Note that in modern control systems, tracking performance is governed by (state) feedback controller gains as well as sampling frequency. Usually, higher gains combined with high sampling frequency is desirable for best command tracking performance. Hence, distributed controllers can be utilized at very high sampling frequencies (20 kHz and up). Furthermore, diagnostics and debugging of controller system become easier. As far their research is concerned, the proposed controller, which is simple, cost effective, and easier to implement, exceeds the performance of the original system.

Similarly, H. A. Thompson et al. demonstrate a distributed aero engine control system [33]. System is identified as safety critical. Hence, they utilize the dual line architecture to incorporate more computational power for control and sensing operations. They also obtain such benefits can be summarized as; retrofitting capability, improved reliability, and improved diagnostics.

The main goal of the progressive research and development works are united under the name of the intelligent manufacturing systems [23]. A distributed control system communicates over a network between its distributed processors. Actions of the

distributed controllers are coordinated properly to achieve desired task(s) with maximum speed and accuracy. Therefore, they interact not only with the manufacturing environment but also the control system and its elements (sensors).

2.3 Networked Control Systems and Time Delays

The solution of modern control problems is to distribute the processing functions over networked controllers [4, 17]. Such controllers (called as nodes) of the distributed/ networked control system share a common communication channel that generally has a bus topology. The resulting architectures require less wiring than traditional ones. Therefore, hardware cost decreases dramatically despite increasing flexibility.

Control systems can be divided into modular subsystems that connect to main system directly. This modularity results in improved diagnostics and maintenance [15, 33]. As presented earlier, distributed systems has become attractive alternative to traditional centralized solutions. However, these systems also bring number vital problems that have to be dealt properly. Otherwise, control system performance will dramatically degrade.

Two main problems can be stated as addressing and timing [39, 49 and 50]. In a shared bus network, each data packet has to be augmented by identifiers, headers or encoded operands that are indicating destination and/or source of the transferred message. Furthermore, distributed units may have to wait for some amount of time before they send out a message. Actually, main limitation here is both physical communication interface and communication protocol being utilized. Based on the communication protocol, the number of nodes connected to control network may also degrade the communication performance. That is, time delays inherently

present in the network could directly influence the overall performance of the system. While designing the controller, elements of the communication system (both protocol and physical communication interface) need to be modeled and have to be added into the overall system model [6, 31 and 54,].

For several decades, modeling and control of networked systems have been studied. In general, delays occur during the transmission of information. Large scale systems (such as telecommunication-, manufacturing-, transportation-, and power generation systems) can be given as typical examples of time-delay systems [36].

Frequency domain (classical) and time domain (modern) approaches are used to analyze these systems [53, 55]. The classical approach employs analytical or graphical methods to identify of the eigenvalues of the characteristic system. Standard graphical methods such as root locus, Bode, and Nyquist plot are applicable to find out the corresponding transfer functions. For a discrete-time with time delays, the system stability can also be identified by root locus analysis in the z-domain.

Unfortunately, classical approach can be used for mostly simple cases. For systems exhibiting multi-delay, functional case must be utilized. That is, using the functional approach, one can derive the delayed differential equations and characterize the systems. In fact, multiple delays and time varying delays can also be modeled using this approach and their stability can be analyzed using Lyapunov's second method [36, 37 and 53]. Note that most of the applications with time delays are considered as non deterministic. Therefore, robust controllers can be implemented on the known systems structure to overcome the delay uncertainty [47].

The computation time delay of microprocessor is regarded as another source of delay. It takes time to execute the instructions in the control firmware. Execution time has direct effect on the control performance [33, 48].

A networked control system employs a number of sensors and actuators in a network [38]. Sending commands and receiving data takes time while implementing the corresponding procedures. The transfer time can either be nearly constant or varying in a random fashion. Also, another issue for such systems is whether the nodes are event driven or clock driven [43, 44, 46 and 52]. Event driven nodes start their activity when an event occurs while the clock driven nodes initiate their activity at specified time frames. As illustrated in Fig. 2.1, there are essentially three kinds of time delays in a networked control system:

1. Communication delay between the sensor and controller,
2. τ_k^{sc} computational delay in the controller,
3. τ_k^c communication delay between controller and actuator, τ_k^{ca}.

Time delay, τ_k , for the control system includes the sampling of the sensor data, calculation of the control signal and sending control signal to the actuator node:

$$\tau_k = \tau_k^{sc} + \tau_k^c + \tau_k^{ca} \tag{2.1}$$

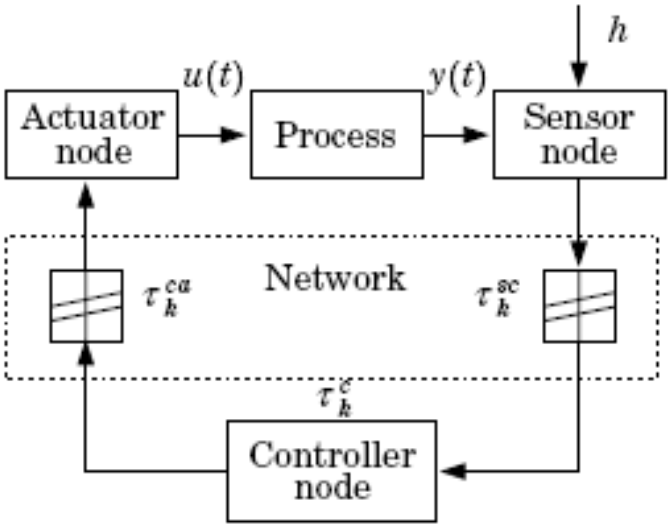


Figure 2.1 Distributed control system with induced delays [16]

Time delays have different characteristics depending on both hardware and software which need to be modeled. The network delays vary due to networking load, network protocol based operations (like invoking master or slave nodes) and physical network failures. Time delay characteristics of control networks installed in industrial systems can be classified as stochastic, bounded and constant. Simplest network delay model assumes that delays are constant for all transfers in the communication network. This can be a good model even if the network has varying delays. Assigning the worst-case delay or a mean value delay is oftentimes practiced in this analysis [16]. Designation of such constant delay element is important for system stability and performance.

Note that a closed-loop system can be transformed into a time invariant one by introduction of buffers as illustrated in Fig. 2.2 [16]. All nodes are considered as clocked and synchronized. If buffers are longer than the maximum time delay in the control system, the system equations turns into

$$x_{k+1} = Ax_k + Bu_{k-\Delta 1} \tag{2.2}$$

$$y_k = Cx_k \tag{2.3}$$

where Δ_1 is the length in samples of the buffer at the actuator node. If the buffer at the controller node is assumed to have the length Δ_2 samples, the process output available for the controller at a time k becomes

$$w_k = y_k - \Delta_2 \tag{2.4}$$

In that case, control design problem can be reformulated as a standard sampled data problem. The information set is available for the calculation of u_k is

$$W_k = \{w_k, w_{k-1,\dots}\} \tag{2.5}$$

An LQG optimal controller can be designed for this control problem [16]:

$$u_k = \xi W_k \tag{2.6}$$

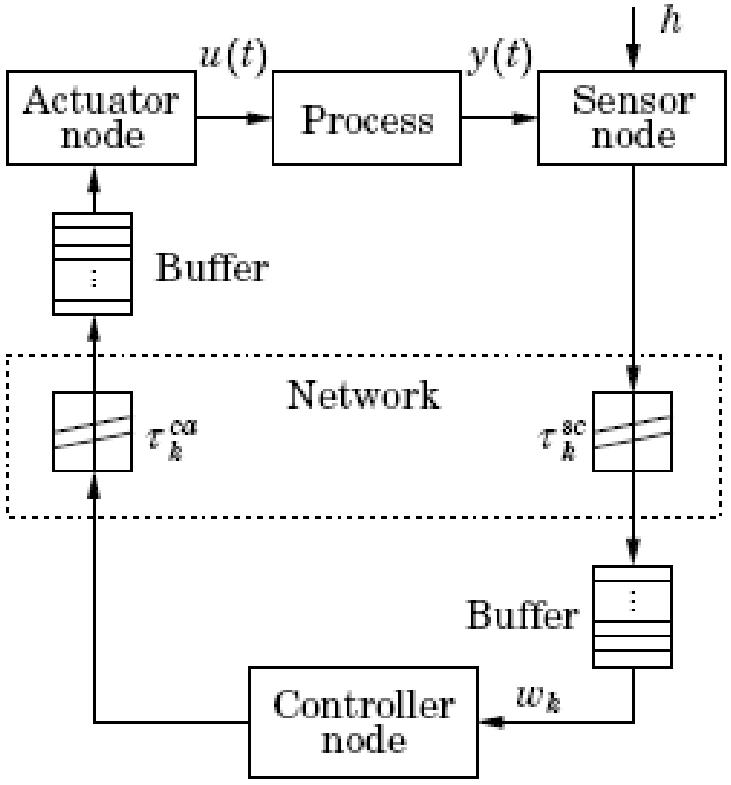


Figure 2.2 Distributed control system with buffers [16]

This method handles delays, which are longer than the sampling period. But evaluated control delays by this method become larger than necessary. It follows that it is a suitable method for event-driven systems. For time driven systems, the controller performance dramatically degrades.

2.4 Synchronization

Most distributed systems encountered in practice are asynchronous. Traditional deterministic, fault tolerant clock synchronization algorithms assume bounded communication delays [7, 28, 40 and 42]. Thus, they cannot be directly used to synchronize clocks in asynchronous systems.

Synchronization of distributed nodes is a research area itself. The only purpose of the clock synchronization is to set the clocks of system nodes to coherent values. In any distributed system, there exists a coordination and communication requirement which is particularly important in real-time control applications [8, 16]. Global coordination and synchronization must be realized with respect to a common basis of time. However, the only clock that is usually available at each node is local clock of micro computers utilized. This clock oscillates at a nominal frequency and generates pulse steams which are subjected to use as increments of the clock counter. Synchronization of local clock of the distributed controllers is maintained by timed increments of the controllers' clock counters so they act together like a centralized controller. Note that the synchronization may be implemented by either software or hardware [27].

Sending synchronization signals over communication network can be considered as software synchronization. On the other hand, hardware synchronization is also an

appropriate technique to synchronize system nodes. This method, which requires an additional wiring just to distribute a global clock signal, can be regarded as very strict method of synchronization. Theoretically, it is only way to synchronize system nodes.

2.5 Research Opportunities

Distributed control systems are spatially interconnected systems that have sensors, actuators and controllers interconnected by communication networks. The introduction of such systems can address the demands of modern industrial and commercial systems. The change of the communication architecture from centralized to the networked/distributed one is proven to be challenging as it introduces different forms of time delay uncertainty in closed loop system dynamics. These time delays directly related to computational time required for physical signal coding and communication processing.

Distributed devices on the network perform control of dedicated operations such as controlling actuator(s) or conditioning sensor data. Therefore, they interact between not only the manufacturing environment but also the control system (and its corresponding elements like sensors). Besides the communication delays, there exists computational delays that needs to be identified and added up the control system model. Presented distributed control architectures facilitate communication networks that are composed of expensive equipments. Besides, they utilize costly and powerful embedded controllers.

The main goal of this research is to look into the (relatively) unexplored aspects of the above-mentioned systems and develop a decentralized control architecture that is suitable for “not-so demanding” applications.

CHAPTER 3

COMMUNICATION SYSTEMS AND PROTOCOL DEVELOPMENT

3.1 Introduction

As was presented in previous chapter, communication is one of the main challenges in distributed control systems. Proper selection and implementation of a physical communication interface with a convenient communication protocol is essential. In this chapter, serial communication interfaces along with the corresponding protocols are presented. Then, the developed communication protocols are introduced and their evaluation is given.

3.2 Data Transmission

Data transmission takes place between data terminal equipment (DTE) and data circuit terminating equipment (DCE) thru various communication interfaces whose multi-layer communication protocol specifications are defined with corresponding standards. In data transmission, there exist two different electrical interface circuits: single-ended and differential. The definition of these systems follows.

3.2.1 Single Ended Data Transmission

The main characteristic of the single-ended data transmission circuits is that data transmission is performed on a single line and that the logical state of the signal is interpreted with respect to the ground. Fig. 3.1 illustrates such a circuit.

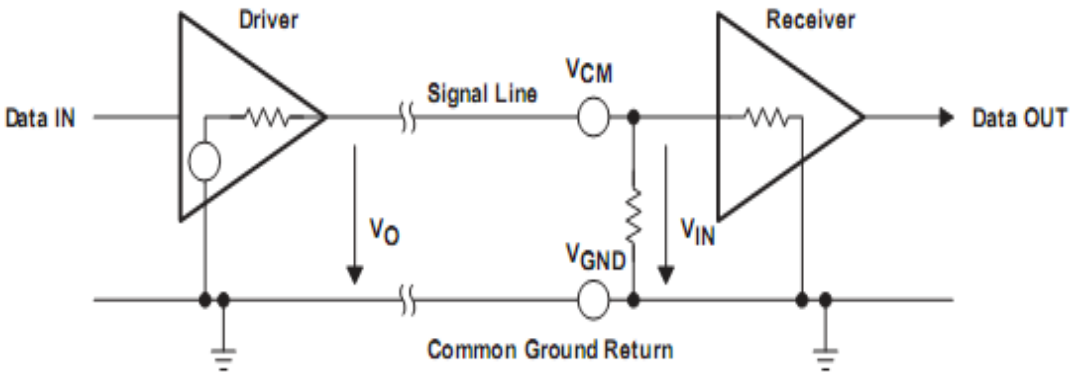


Figure 3.1 Single ended data transmission [51]

A single-ended transmission circuitry requires only a single line hence this configuration is very simple, easy to implement and cost effective. Long distances or noisy environment sometimes forces to add up a better cable shielding and additional ground lines to system. This configuration is known to have poor noise immunity. Main problem here is ground wires. They are part of the circuitry and transient voltages may lead to signal degradation. This results in the misinterpretation of the signal or false receiver triggering. Also crosstalk may be observed at higher frequencies.

3.2.2 Differential Data Transmission

In this configuration, both receiver and transmitter signal lines are necessary for differential data transmission. On one line, the actual signal is transmitted where as the second line is used to carry the inverted form of this signal. The receiver detects the voltage difference between two transfer lines and switches the output depending on which magnitude of this potential difference. Fig. 3.2 shows a typical differential data transmission circuitry.

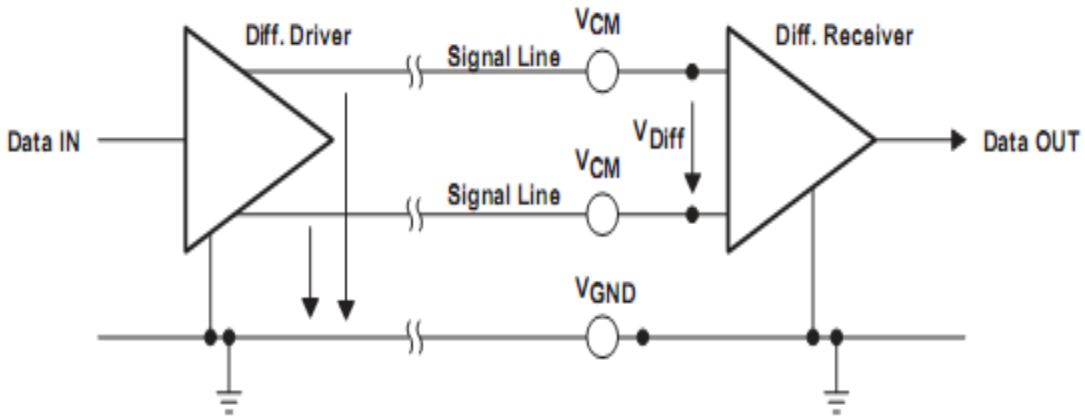


Figure 3.2 Differential data transfer [51]

As illustrated in Fig. 3.3, a twisted cable pair is exclusively used in these interfaces because such cable configurations offer noise-immunity benefits over flat- or ribbon cables. If twisted pair is used, both of the wires are affected by noise sources equally. The noise produces a common mode signal, which is called each other, when the difference signal is taken by the receiver [57]. Also, the correct line termination avoids any unwanted fluctuations and allows the transmission of data at

higher rates. With different interface voltages, signal transmission rates can reach up to 10Gbps.

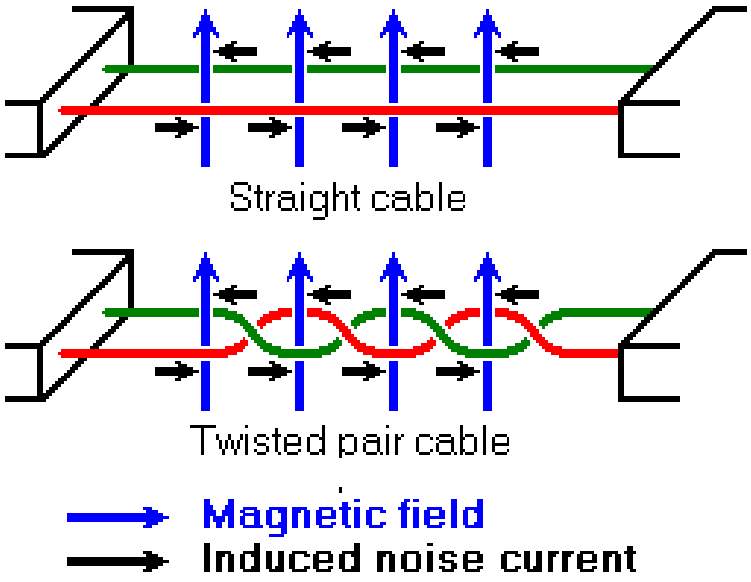


Figure 3.3 Cable configurations [57]

Although balanced interface circuitry consists of a generator with differential outputs, balanced transfer lines increases cost of such interfaces. However, developing this complex interface circuitry with CMOS fabrication process overcomes this difficulty. Hence, data transceiver chips are widely available in the market at affordable prices.

3.3 Data Transmission Topologies

In telecommunication technology, there exists data transfer topologies that can be used when one tries to connect such interfaces among each other. The detailed descriptions of the topologies follow.

3.3.1 Point-to-Point (Simplex)

As shown in Fig. 3.4, point-to-point data transmission topology can be implemented with one transmitter and one receiver per line. In this topology (which is commonly referred to as unidirectional data transmission), data can be transmitted in only one direction.

Figure 3.4 Point-to-point connection [81]

3.3.2 Multi-drop (Distributed, Simplex)

As shown in Fig. 3.5, the multi-drop data transfer topology is obtained when one transmitter and more than one receiver are used. Since this topology inherits the characteristics of point-to-point data transmission, data can be transferred along one direction.

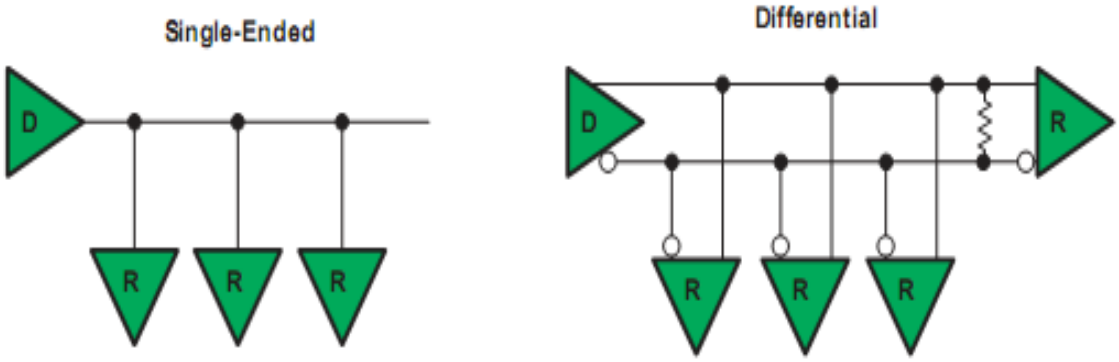


Figure 3.5 Multi-drop connection [81]

3.3.3 Multipoint (Multiplex)

As illustrated in Fig. 3.6, this augmented topology enables the bidirectional data transmission. Here, there exists a transmitter-receiver pair per line. This pair is called as transceiver.

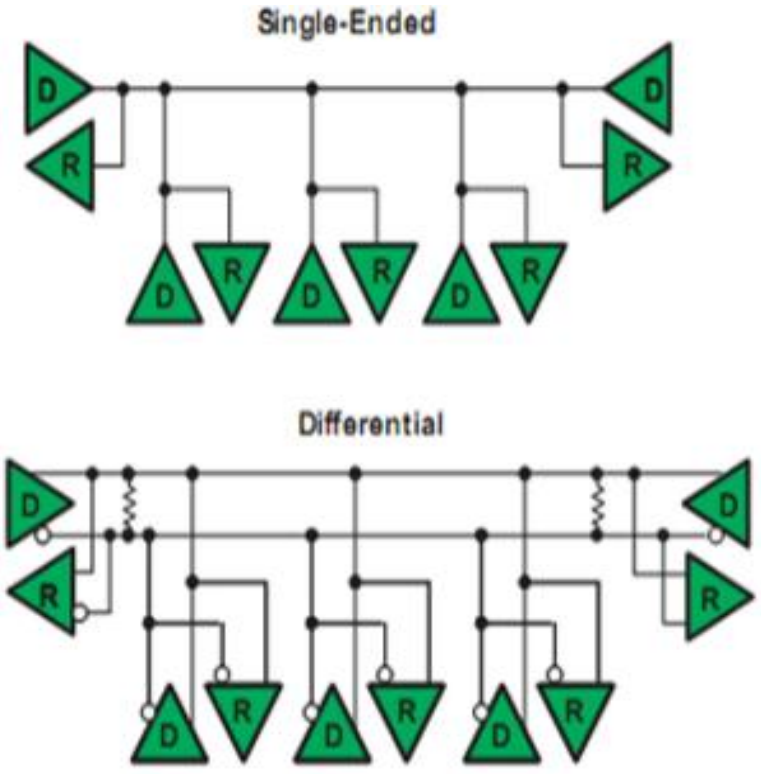


Figure 3.6 Multi-point connection [81]

3.4 Physical Communication Interfaces

In this section, various communication standards used over the afore-mentioned topologies will be evaluated.

3.4.1 EIA – 232

EIA-232 (Recommended Standard 232) is the most elementary serial communication protocol that is adapted by the Electronic Industries Association. The latest version RS-232 C released at the end of 1960s. The signal of the EIA-232 has two logic levels: high logic (+5V to +15V) and low logic (-5V to -15V). High logic level is identified by negative voltage levels where low level is defined by positive voltage levels. According to EIA-232 standards, maximum cable length must have a maximum capacitance of 2500 pF**.** In order to limit any reflections that occurred in rise and fall instances of the signal, maximum slope (gradient) of the signal is limited to 30V/μs [51]**.** Note that this allowable gradient also depends on data transfer rate. High frequency signals do deteriorate significantly if transferred long distances. Table 3.1 illustrates the cable length baud rate correlation.

Table 3.1 EIA-232 baud rate versus cable length [51]

Baud rate	**Max cable length [m]**
19200	15
9600	150
4800	30
2400	900

EIA-232 standard is essentially asynchronous serial communication method where data bit stream is not sent over a strict time frame. Fig. 3.7 illustrates a stream of serial data being transferred via EIA-232. Note that the original data stream has to be augmented with start-, stop-, and error detection bits to guarantee the transmission of data properly. Each data character starts with an attention bit which is also known as start bit. Data bits directly follow this start bit. A bit value of 1 (TRUE) causes negative voltage levels whereas a bit value of 0 (FALSE) represented as a space. For error detection purposes, one may include a (even or odd) parity bit. Finally, the transferred data stream is terminated by a stop bit.

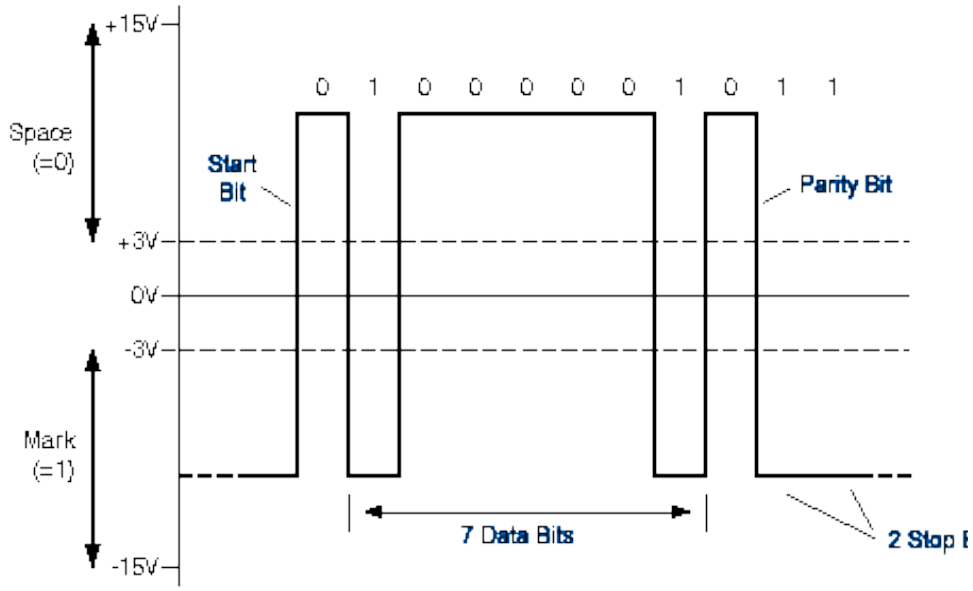


Figure 3.7 An EIA-232 stream [56]

3.4.2 EIA – 422

Unlike EIA-232, EIA-422 (formerly known as RS 422) allows multi-drop connection of a driver to (a maximum of) ten receivers. Its data transfer rates can be expanded up to 35Mbits/s with a (maximum) cable length of 1200m. As stated previously, higher data transfer rates generally require the reduction of the cable length.

3.4.3 EIA – 485

EIA-485 (RS 485) is the most widely used communication interface for many industrial data acquisition- and communication systems. TAI/EIA-485 standard defines the communication half-duplex and differential. 32 nodes are theoretically possible. According to this standard, the driver must deliver a minimum output voltage of 1.5V. Common voltage levels may vary between -7V and 12V. Note that this communication method has high noise tolerance.

3.4.4 CAN

Controlled Area Network (CAN) was introduced by Bosch Co. in 1980 for automotive applications. It was to replace the knotty signal cable wiring and reduce them to two – wire bus. In fact, CAN is applicable for appliances that require large number of small messages in a short period. Unlike other address-based communication protocols, CAN is message-based system and is especially useful when system-wide data consistency is required.

In CAN, every node on the bus reads the identifier then decides whether or not to read the rest of the message. When a node wants to transmit a message, it has to compare its relative priority to that of the network message. If it is less than or equal to the importance, it has to wait until the bus is clear. ISO 11898 (the CAN standard) describes the differential and half-duplex data transmission. In this standard, the maximum cable length is 40 m and a maximum of 30 nodes is allowed on the bus. Notice that signal transmission rates can reach up to 1Mbps. A single shielded or unshielded twisted pair cable with 120Ω is also utilized in CAN [33].

3.5 Communication Protocols

A communication protocol is said to be the set of standard rules for data representation, signaling, authentication, and error detection required to send information over a communications channel. Communication protocols for digital computer network communication have features intended to ensure reliable interchange of data over an imperfect communication channel. Communication protocol basically follows certain rules so that the system works properly. Next, popular communication protocols are elaborated.

3.5.1 TCP/IP

Transmission Control Protocol/Internet Protocol (TCP/IP) is also known as Internet Protocol Suit even though its development began in the 1960s. TCP/ IP was originally considered for UNIX operating system. It was designed to make network robust and automatically recover from the failure of any device on the network.

Moreover, it allows the construction of very large networks that require little central management.

TCP operates at the transport layer, the middle layer in the seven layer OSI (open systems interconnection) reference model. This layer is responsible for maintaining reliable end-to-end communications across the network. IP, in contrast, is a network layer protocol, which is the layer just below the transport layer.

On the other hand, the IP protocol deals only with packets (i.e., the most fundamental unit of TCP/IP data transmission), TCP enables two hosts to establish a connection and exchange streams of data. TCP guarantees delivery of data and also guarantees that packets will be delivered in the same order in which they were sent [82].

3.5.2 MODBUS

Modbus is a master–slave serial communications protocol proposed by Modicon. It was first released in 1979. Most Modbus devices utilize EIA-485 physical communication interface. But the format of the Modbus messages is independent from the type of physical interface used. Each Modbus message has four basic elements which presented in the Table 3.2

Table 3.2 Basic elements of Modbus messages [45]

Field	Description
Device Address	Address of receiver
Function Code	Code defining message type
Data	Data block
Error Check	Test for communication errors

Since Modbus is a master – slave protocol, there is no way for a field device (a slave node) to transmit any kind of data if it is not requested by master node. Fig. 3.8 represents a general Modbus message frame.

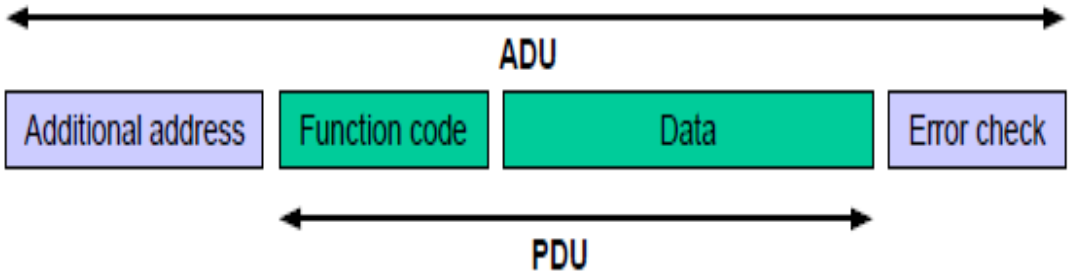


Figure 3.8 General Modbus frame [45]

3.5.3 MACRO

Macro is a distributed machine control communication standard. It stands for motion and control ring optical and is primarily designed for multiple masters and slaves. Communication is always initiated by master by sending out data packet with appropriate address (Fig. 3.9). Next node receives the data stream and checks the

address whether the address data is the same as its own local address. Then, the node takes data from the packet and releases the packet through the ring. If the local address is different, the node just passes the data without reading. Finally data is returned to the master. [83]

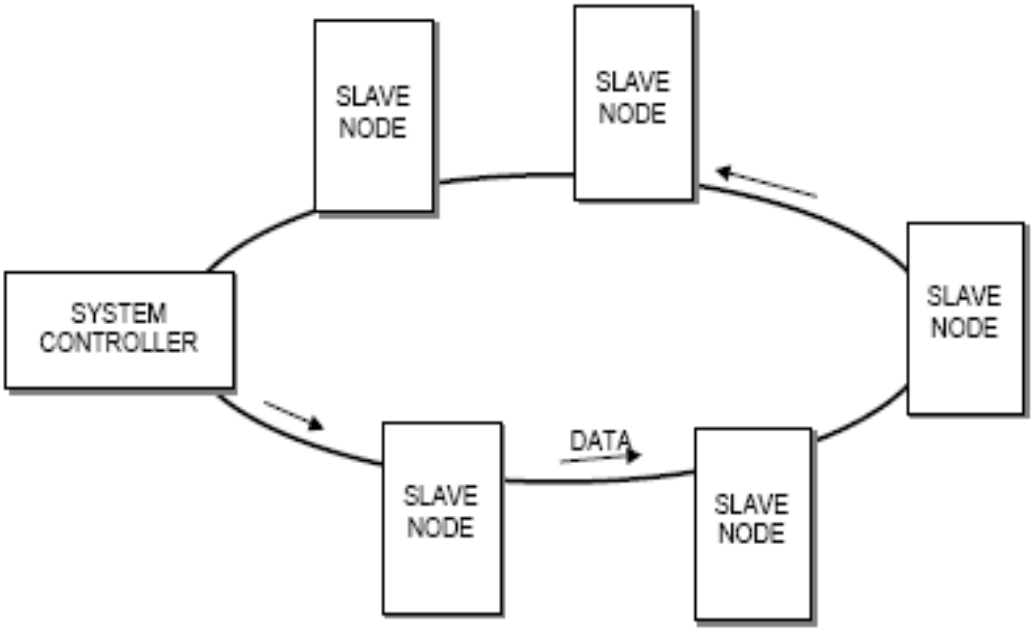


Figure 3.9 MACRO ring network [83]

3.5.4 SERCOS

The SERCOS standing for serial real-time communication system which is a digital motion control bus. It is an open controller-to-intelligent digital device interface that is designed for high-speed serial communication of standardized closed-loop (real-time) data over a (noise-immune) fiber optic ring (SERCOS I & II) or Industrial Ethernet cable (SERCOS III). A generic SERCOS interface is illustrated in Fig. 3.10.

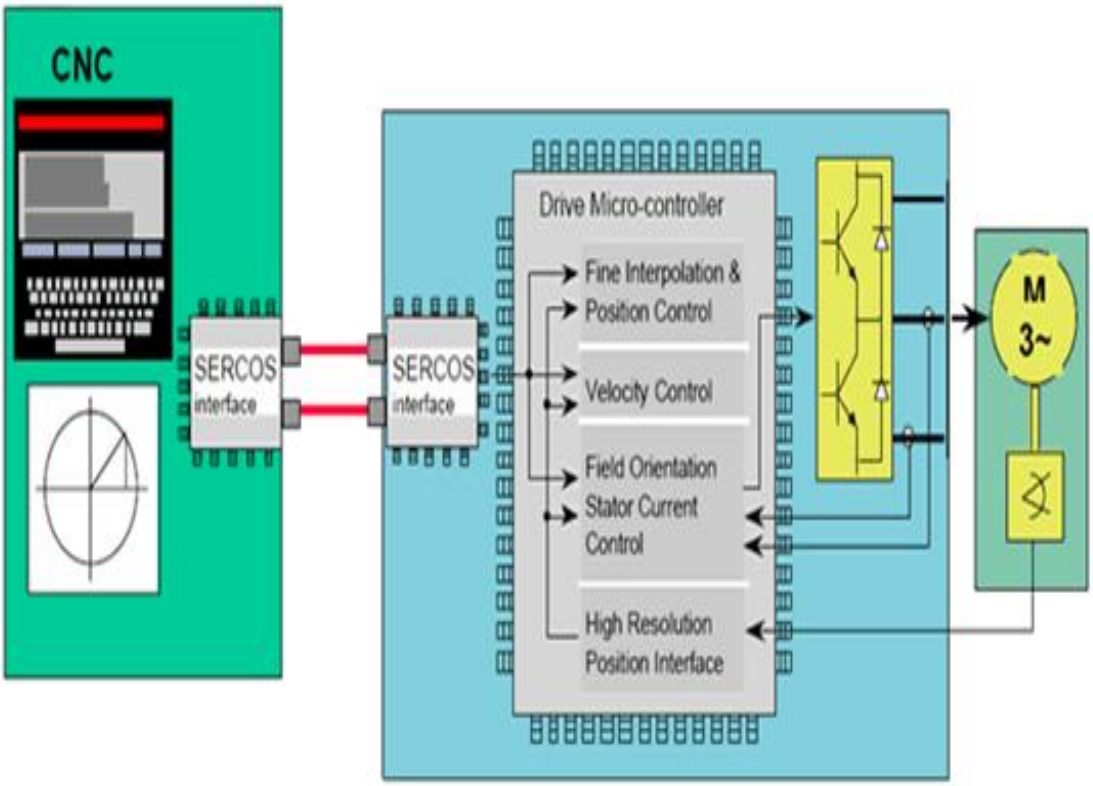


Figure 3.10 SERCOS interface [84]

In fact, the SERCOS interface reduces connectivity problems in control systems. It exchanges data between controller and (motor) drives and synchronizes actuators for precise coordinated moves. Using this interface enables coordinated operation of controllers built by different manufacturers. Hence, extensive reliability of the SECROS interface enables flexible, modular and fully distributed controller design.

3.6 Developed Communication Protocols

Utilizing EIA-232, full-duplex EIA-422 and EIA-485 communication interfaces several communication protocols can be developed. First bidirectional approaches are to be applied on the control network. Then unidirectional protocols will follow.

3.6.1 Bidirectional Protocol

Communication layer of the software handles the communication network. It produces the interpolated commands for each axis into two byte data and starts the communication stream. Note that this stream can be regarded as both message- and address based since the transaction is triggered by data transmission while each controller (node) in the network fetches the data in the proper position (address) of the transferred array.

Data array consists of 13 bytes as shown in Fig. 3.11. It can handle a very limited number of distributed units that are interconnected to each other via EIA-232 communication interface.

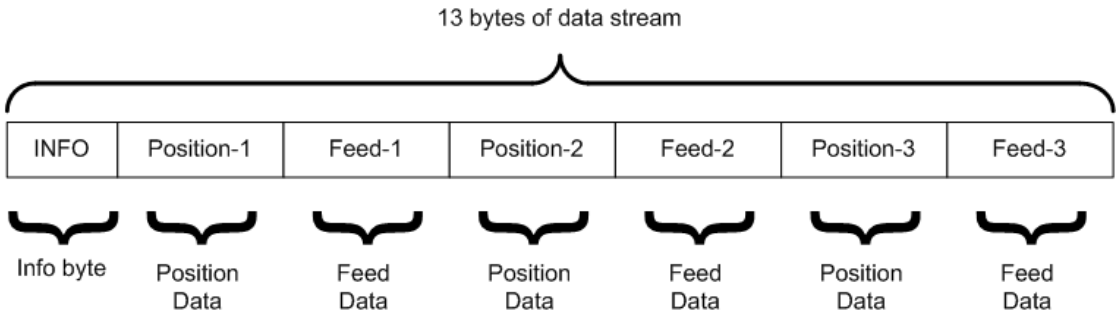


Figure 3.11 Bidirectional protocol data array

When the stream is received, all controllers receive 13 bytes. According to the info byte (the first byte of the stream), controller decides on the operating mode. Relevant (position) command parameters are addressed in the byte array. Note that the entire tool path of a machine tool is defined by an NC part program. The tool trajectory is processed and is divided into small line segments that are defined by positional increments assuming that the velocity along a particular segment ("feedrate") is constant. Physical configuration of the protocol is illustrated in the Fig.3.12.

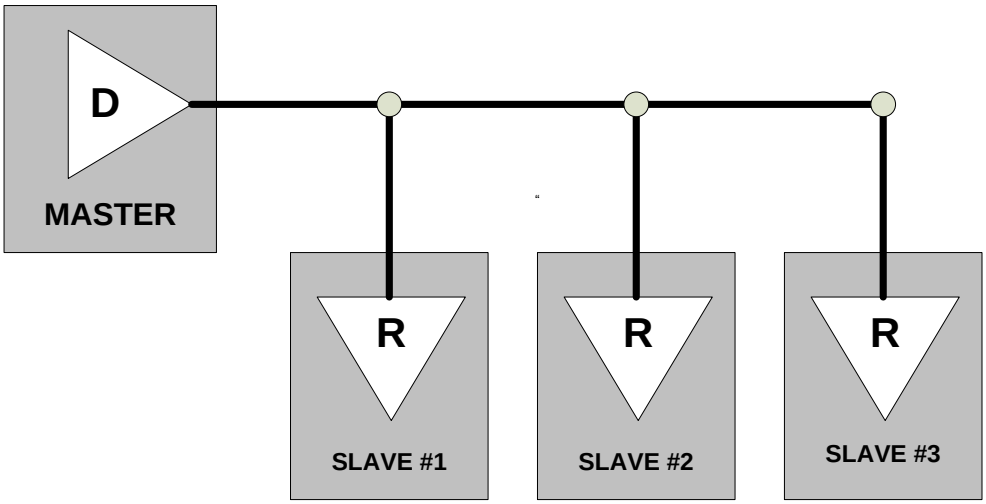


Figure 3.12 Physical configuration of bidirectional protocol

As stated earlier, this communication is bidirectional. Hence, the slave nodes cannot reply to any kind of message. This is a serious drawback of motion control applications as the user interface needs the status information on slave nodes (axis controllers). Furthermore, increasing the number of units will also swell the network's load. In other words, the quality of system (QoS) will decrease as number of slave nodes is increased. Experimental results and limitations of the

communication protocol is demonstrated in the Fig. 3.13. Notice that the desired bandwidth of the position control system is 100 Hz within the scope of this study. In that case, the control algorithm has to be executed in 10ms. According to this assessment, increased network load dramatically degrades quality of performance (QoP).

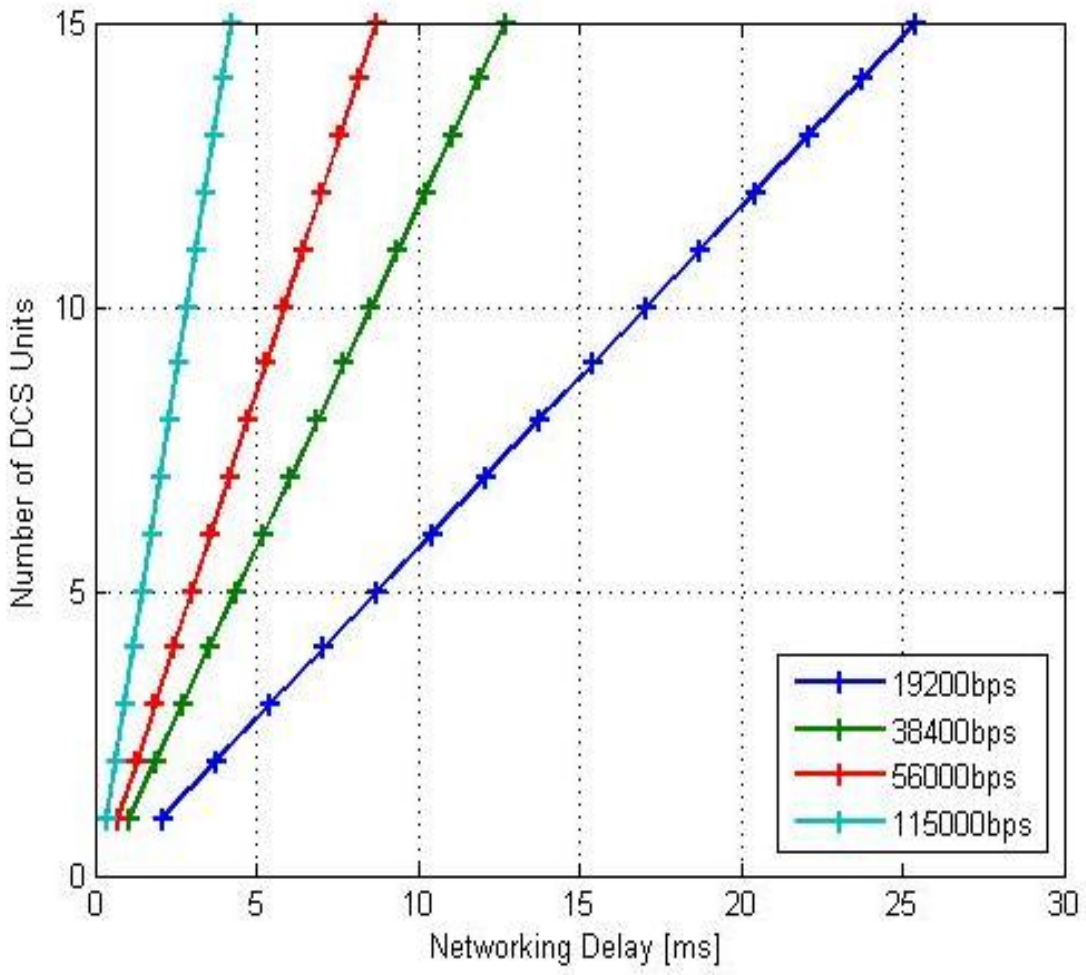


Figure 3.13 Evaluation of the bidirectional protocol

The main advantage of this protocol is that it is extremely simple to implement. Only one data stream is sufficient for execution. Address definitions are fixed so that there is no need for additional indicators to address such controllers. On the other

hand, increased number of distributed controllers will also increase the number of bytes in the data stream. Hence, quality of performance (QoP) will degrade with increased number of controllers. Furthermore, it is not flexible as each controller has different firmware to extract relevant part from the transmitted data.

3.6.2 Unidirectional Protocol

Unidirectional communication protocol enables the communication between master and slave devices where their physical communication interface is described in Fig. 3.14.

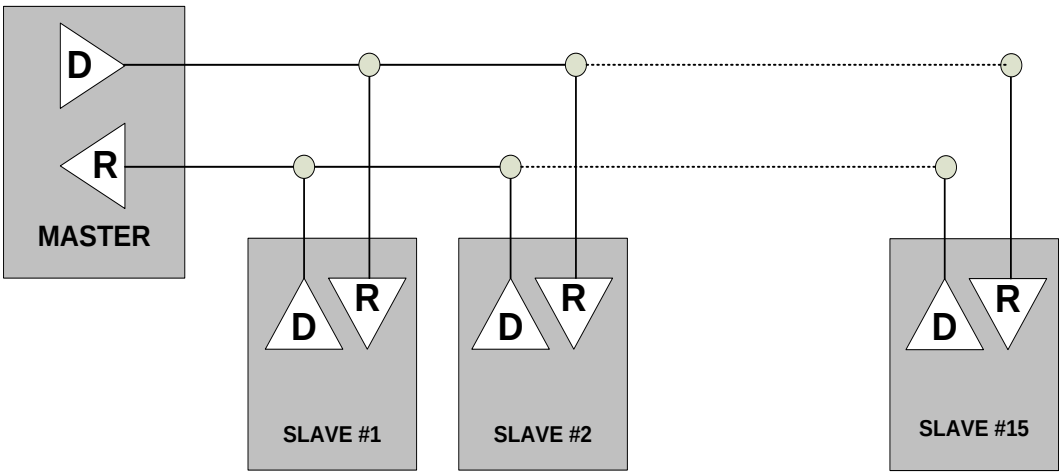


Figure 3.14 Physical configuration for unidirectional protocol

Utilizing the full-duplex EIA-485 communication, distributed devices can reply to some requests that are transmitted by master node. By definition, this protocol is limited to 15 distributed controller units. Each unit facilitates identical firmware with user selectable unit address. This is a strictly-defined (address-based) communication protocol which utilizes its own syntax as tabulated in Table 3.3. Usage of such communication language makes distributed units more flexible.

Table 3.3 Communication syntax

Command	Explanation
UID + “0”	Set unit for data transfer
UID + UID	Set unit as master & prepare for data transfer
“j” + UID	Start jogging
“s” + UID	Unit stop
“h” + UID	Unit go home position
“r” + “r”	Reset System

Protocol simply uses two bytes of data that is briefly explained in Table 3.3. Utilization of the buffer, which needs to be filled with reference position commands periodically, degrades system’s performance. Accordingly, various communication speeds were tested to determine the required time for filling buffer up. Fig. 3.15 illustrates the results. While the system is executing predefined commands in the buffer, the number of nodes in the system does not affect the performance. Host only transfers a signal when a synchronization request is pending. This constitutes only 3 bytes of data. Assuming that the controller’s sampling period is 100 Hz, the effect of software synchronization is described in Fig. 3.16. Notice that it is a robust protocol as the slave nodes can only speak with master node. Therefore, the protocol can be classified as master-slave protocol.

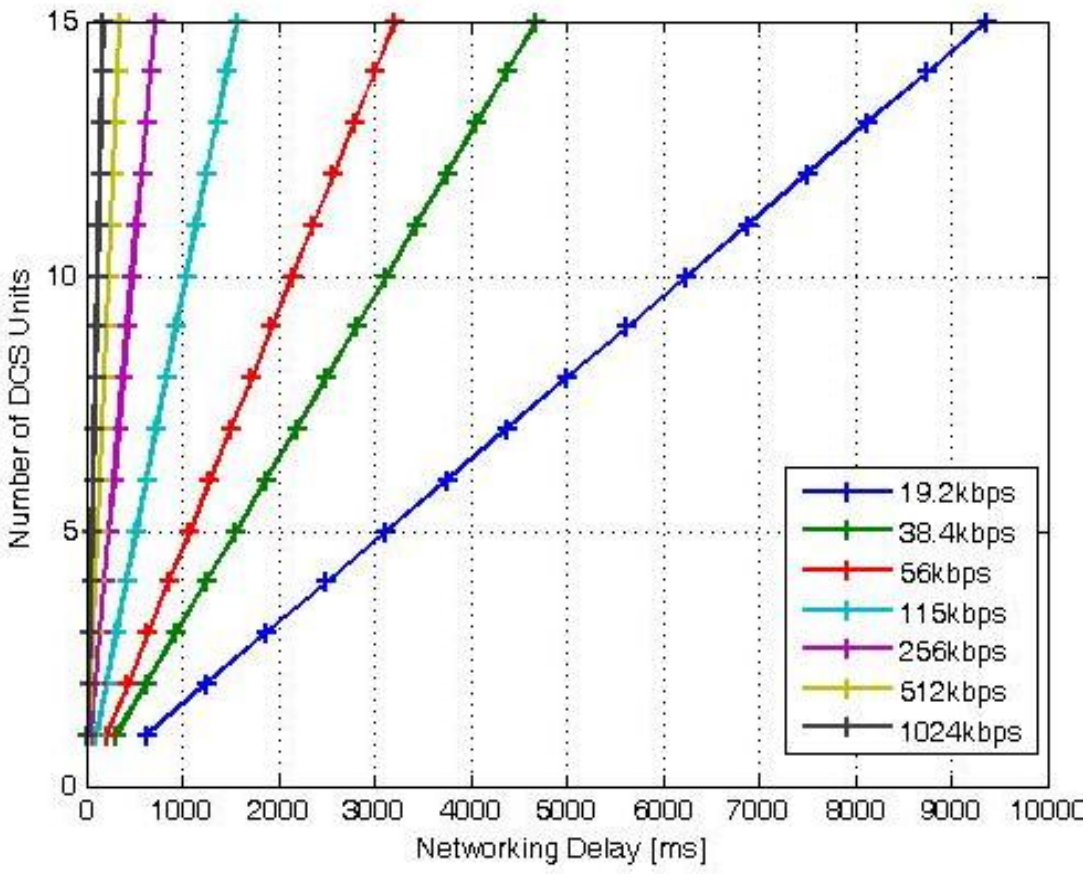


Figure 3.15 Evaluation of the unidirectional protocol

With respect to the advantages, it is a simple (yet flexible) communication protocol that utilizes master-slave communication. Slave devices can reply transmitted inquiries by master node. Increased number of distributed nodes does not affect either QoS or QoP. Furthermore, each controller (slave node) has identical firmware.

On the other hand, a limited number of devices (up to fifteen) can be utilized in this protocol. Slave nodes cannot transmit any data unless it is requested by master node. Additionally, slave devices cannot communicate each other.

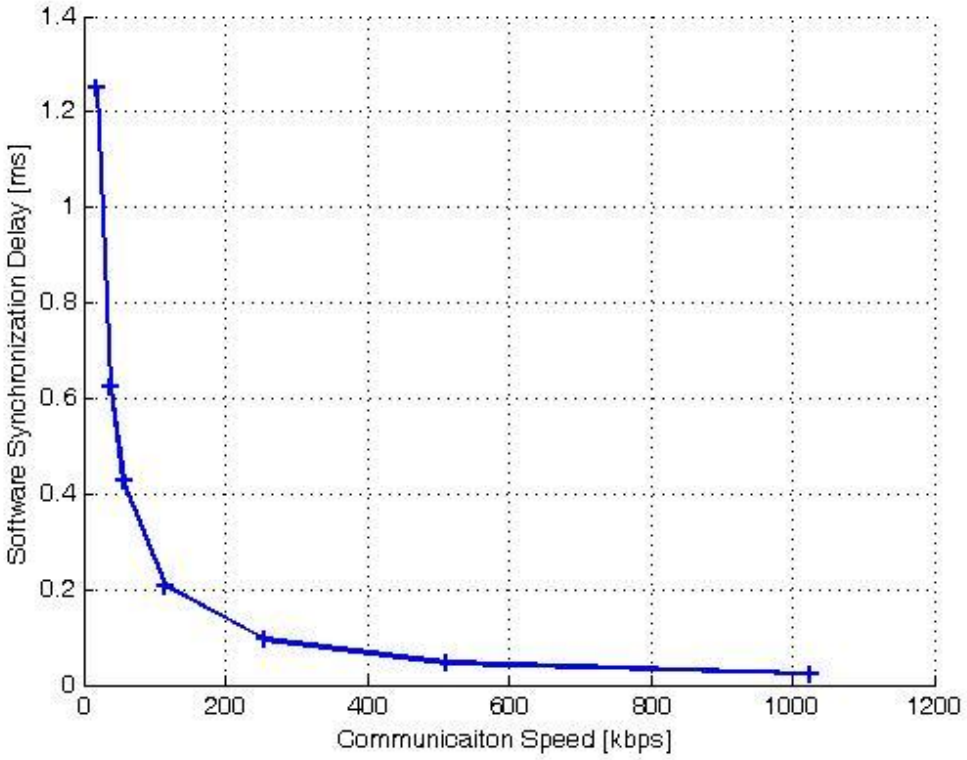


Figure 3.16 Evaluation of software synchronization delay

3.6.3 Unidirectional Protocol with Communication Between Slaves

In this thesis, the devised controller has a reconfigurable communication interface. Using the jumpers on controllers, EIA-485 communication interface can be configured as half- or full duplex. Setting communication interface to half-duplex enables communication between slaves. This enhancement brings the idea of multi-master communication. This diagram of this arrangement is given in the Fig. 3.17. As stated earlier, this protocol has its own problems like data collision during transmission since data transfer line is same for both receive and transfer.

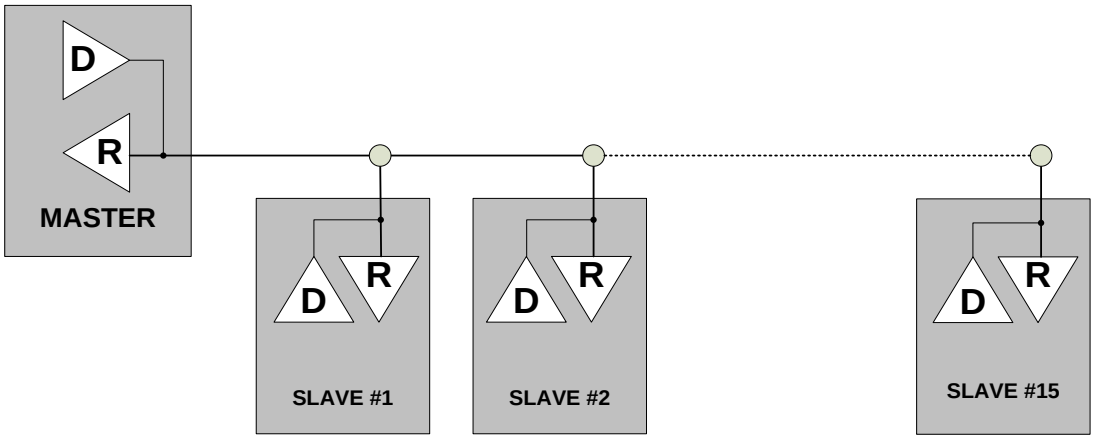


Figure 3.17 Physical configuration of unidirectional protocol with communication between slaves

3.7 Distributed Controller Scheme

During thesis work, various controller implementations have been considered. In order to determine the best distributed controller design, three controller models (communication interfaces, protocols) have been developed and implemented. This section summarizes the development efforts.

3.7.1 Method A

Method A can be considered as brute method for networked control as every single instruction cycle of reference commands are sent to each distributed controllers by utilizing a common fieldbus (EIA-232). It is obvious that the proposed method requires only one directional communication. Hence, the slave nodes are to listen to only the message being transmitted by the host. Reference position of the each controller is interpolated and embedded into the transferred message. Unfortunately,

the developed communication protocol rejects the message transaction initiated by slave nodes. Furthermore, this method can be applicable when only a limited number of devices in the distributed control network is utilized. There is no need for additional synchronization between slave nodes. Reference position commands, which are generated by the host PC, can serve as software synchronization signal too. A real-time operating system should be facilitated on the PC because of the timing purposes. A pseudo code implementing the code is given as follows:

Table 3.4 Pseudo code for Method A

```
Void main () {
   // initialization and get unit ID
   ID = InitializeController();
   While(TRUE) {
      If(serial_is_data_in()){
         // read transferred message and encode it
         Data = Read_serial_data(ID);
         // execute transferred message
         Execute(Data);
      }
   }
}
```

It can be perceived that in Method A, position data is first created as small line segments. Hence, smaller segments increase precision of the incremental motion while increasing the network load. Note that EIA-232 supports data transmission rates up to 115kbps while utilizing 15 nodes. As it is illustrated in Fig. 3.18, increasing sampling frequency of the control system dramatically shortens the remaining execution time for the controller algorithm. Hence, this situation pushes the limits and finally system cannot reproduce any kind of reaction.

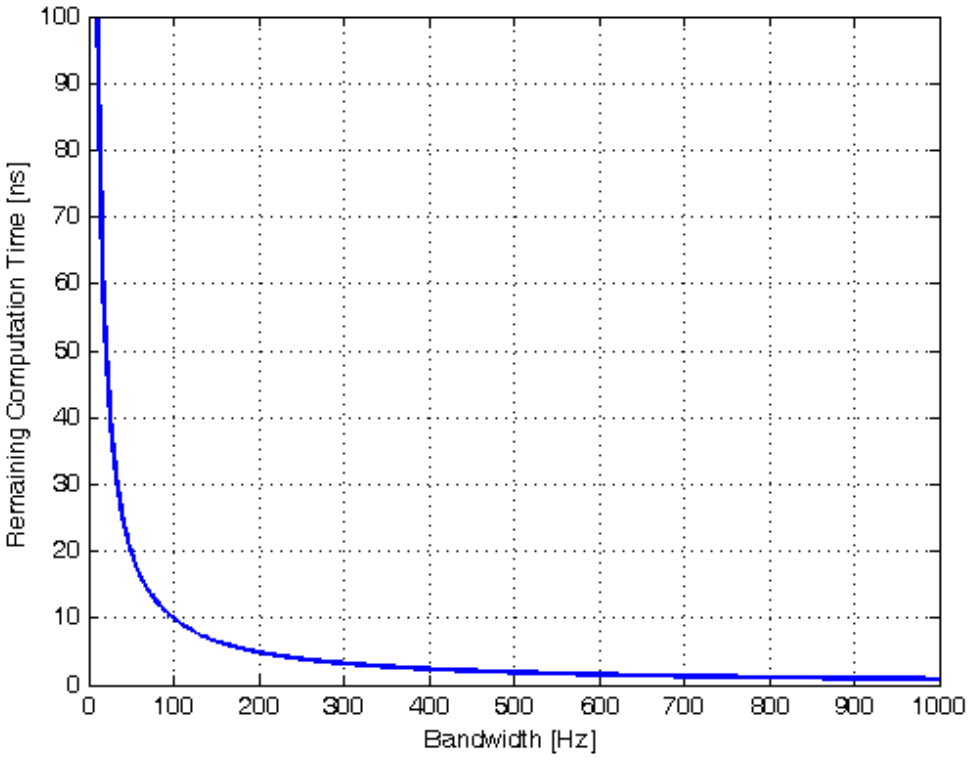


Figure 3.18 Remaining computational time

3.7.2 Method B

Method B is developed as alternative controller architecture for distributed position control. It has communication layer, interpolation layer, and position control layer. Unlike other method, it incorporates several microcomputers, which are dedicated to handle each layer. A data bus connects all the microcomputers to each other via SPI communication interface. PC software hosts distributed communication network. Communication network is based on the EIA-232 physical interface.

Main goal of the Method B, whose pseudo code is given in Table 3.5, is to reduce the communication needs and does an optimization between computation time and communication performance. Notice that all displacements, which are performed by

actuators, can be modeled as small line and arc segments. Unlike Method A, which uses only small line segments, the Method B incorporates constant coefficient difference equations (CCDE) of prescribed motions. Machine tool movements, which are generally described by Eqn. (3.1) and Eqn. (3.2) (for circular patterns) and by Eqn. (3.3) and Eqn. (3.4) (for the linear patterns), do incorporate constant parameters (A, B, …, G) of which are then transferred by PC host as command signals.

$$x(k) = A.x(k-1) - B.y(k-1) \tag{3.1}$$

$$y(k) = C.x(k-1) + D.y(k-1) \tag{3.2}$$

$$x(k) = E.x(k-1) + F \tag{3.3}$$

$$y(k) = G.y(k-1) + H \tag{3.4}$$

In conclusion, the PC software (master node) does the interpretation of the position commands (i.e. computing the coefficients of the CCDEs) while the controllers (slave nodes) directly handle the interpolation. Notice that the slave nodes can listen to the message of the master but they are able to reply inquiries of the master as well.

Table 3.5 Pseudo code for Method B

```
#INT_SERIAL
Void Inquiry() {
   If(host_inquiries_position()) {
      Send_serial_data(Position0);
   }
}
Void main () {
   // initialization and get unit ID
   ID = InitializeController();
   While(TRUE) {
      If(serial_is_data_in()){
         // read transferred message and encode it
         Data = Read_serial_data(ID);
         // use previous position data and interpolate
         Position1 = Interpolate(Position0,Data);
         // execute transferred message
         Execute(Position1);
         // now data history is replaced
         Position0 = Position1;
      }
   }
}
```

3.7.3 Method C

Method C inherits the best features of the prior techniques. That is, just like Method A, it employs small line segments. A common EIA-485 field bus is facilitated as network which is controlled by a host PC. Normally, PCs do not include EIA-485 serial communication interface. Therefore, a PCI based serial communication card can be incorporated.

In this technique, the host fills the buffers of the controllers at the beginning and selects controller as master. Controllers are coordinated by software synchronization. When start signal is received, each controller begins to empty their buffer. Each data, which takes up particular space in the buffer, has a time index.

This time index is monitored by both the host and slave nodes. Hence, the host does not require the generation of additional inquiries. The pseudo code for Method C is given in Table 3.6.

Table 3.6 Pseudo code for Method C

```
Void main () {
  // initialization and get unit ID
   ID = InitializeController();
   k = 0;   // reset time index
   While(TRUE) {
      If(serial_is_data_in()){
         // read transferred message and encode it
         Data = Read_serial_data(ID);
         [Select, Master] = Implement_Protocol(Data);
         Switch(Select){
            Case 1: Fill_Buffer(Select);
            Case 2: Jog(Select);
            Case 3: Go_Home_Position();
            Case 4: Execute(Select);
                    //increase time index
                    ++k;
                    If(MASTER) Send_serial_data(OK());
         }
      }
   }
}
```

Main contribution of the Method C is that number of units can only be limited by capacity of communication network. Note that filling buffer again and again disrupts the continuing machining process. Thus, employing cyclic buffer may improve the performance of the system. Related benefits are presented in the Figs. 3.15 and 3.16.

3.8 Summary

Digital communication interfaces and popular communication protocols were presented in this chapter. Developed hardware architectures were discussed and developed communication protocols were presented. As far as industrial applications are concerned, communication interfaces must have high noise immunity. Single-ended data transmission interface should not be used. According to application, there may lots of data to need to be transferred. Also there may exist n-many nodes to perform operation. A proper communication protocol which uses transmission interface efficient is essential. Besides proper synchronization of distributed controllers is also a necessity.

Utilization of bidirectional protocol with EIA-485 interface seems to be an adequate solution of those problems. Also, EIA-485 today can offer up to 10Mbps communication speed. Hence, software synchronization and disruption of the buffer initialization do not affect control system performance. Dumping relevant data elements into buffer offers simple communication structure.

CHAPTER 4

CONTROL SYSTEM HARDWARE ARCHITECTURE

4.1 Introduction

As described in Chapter 2, centralized control systems generally utilizes one (or more) powerful microcomputer(s) performing all desired operations such as command generation, book keeping, multi-axis (real-time) motion control, etc. Functional use of this architecture is strictly defined and the corresponding hardware does not tolerate to any timing errors.

Main aim of this study is to develop (practical) distributed controller which takes advantage of high-speed serial communication network. During this study, a communication network is created and identical controller units are distributed through that communication network.

In this thesis, several distributed controller architectures are devised. A distributed controller employing EIA-232 interface is presented in the Appendix A. Since the EIA-232, which has low noise immunity, has low data transmission speed, the resulting controller topology is not suitable for demanding applications. Hence, a novel controller, which utilizes (half / full duplex configurable) EIA-485 interface, is developed. The attributes of this controller follow.

4.2 Hardware Architecture

First of all, the developed (control system) hardware has to be working in conjunction with a host PC or an IPC (Industrial PC). The PC running a high-level software controls the communication network and performs the relevant interpolation functions. Note that in today's technology, the CAM (Computer Aided Manufacturing) modules can easily generate NC part programs. Instead of converting CAD data to NC part programs, the CAM programs may directly generate proper position commands for distributed controllers. Hence, the sole purpose of the distributed controller is to implement advanced motion control algorithms in a synchronized fashion.

The developed controller is equipped with double CPUs, which are Microchip's PIC16F88 and dsPIC30F4011. The dsPIC30F4011, which contains a RISC CPU, has a wide range of peripheral interfaces. This 16-bit digital signal controller is specifically designed for motor control and industrial applications. Some of the relevant features of this chip are as follows:

- 30 MIPS operation speed,
- 2048 bytes of SRAM,
- 9 channels of 10-bit (1 Msps) Analog-to-Digital (A/D) converter
- 2 UART modules with FIFO buffers,
- 1 CAN module (CAN 2.0B compliant),
- 3-wire Serial Peripheral Interface (SPI) modules (supports four frame modes),

- 2-wire Inter-Integrated Circuit (I²C) bus (supports multi-master/slave mode and 7/10 bit addressing),
- PWM output channels,
- quadrature encoder interface with 16-bit up/down position counter,
- 16-bit compare output functions.

The PIC16F88 is a powerful CMOS flash based 8-bit micro controller. It features the followings:

- 8 MHz internal oscillator,
- 256 bytes of EEPROM data memory,
- a capture/compare/PWM, an Addressable USART,
- a synchronous serial port that can be configured as either 3-wire Serial Peripheral Interface (SPI) or the 2-wire Inter-Integrated Circuit (I²C) bus,
- 7 channels of 10-bit Analog-to-Digital (A/D) converter
- 2 Comparators that make it ideal for advantage analog / integrated level applications in automotive, industrial, appliances and consumer applications.

Functional block diagram of the designed hardware is given in the Fig. 4.1. In this system, dsPIC30F4011 has an important role in the developed distributed control system. First, it functions as a communication interface. Its 2KB of SRAM are used as a reference command buffer. As presented in previous chapter, using prescribed protocol, the reference commands sent by the host PC are stored in the SRAM of the chip for later execution.

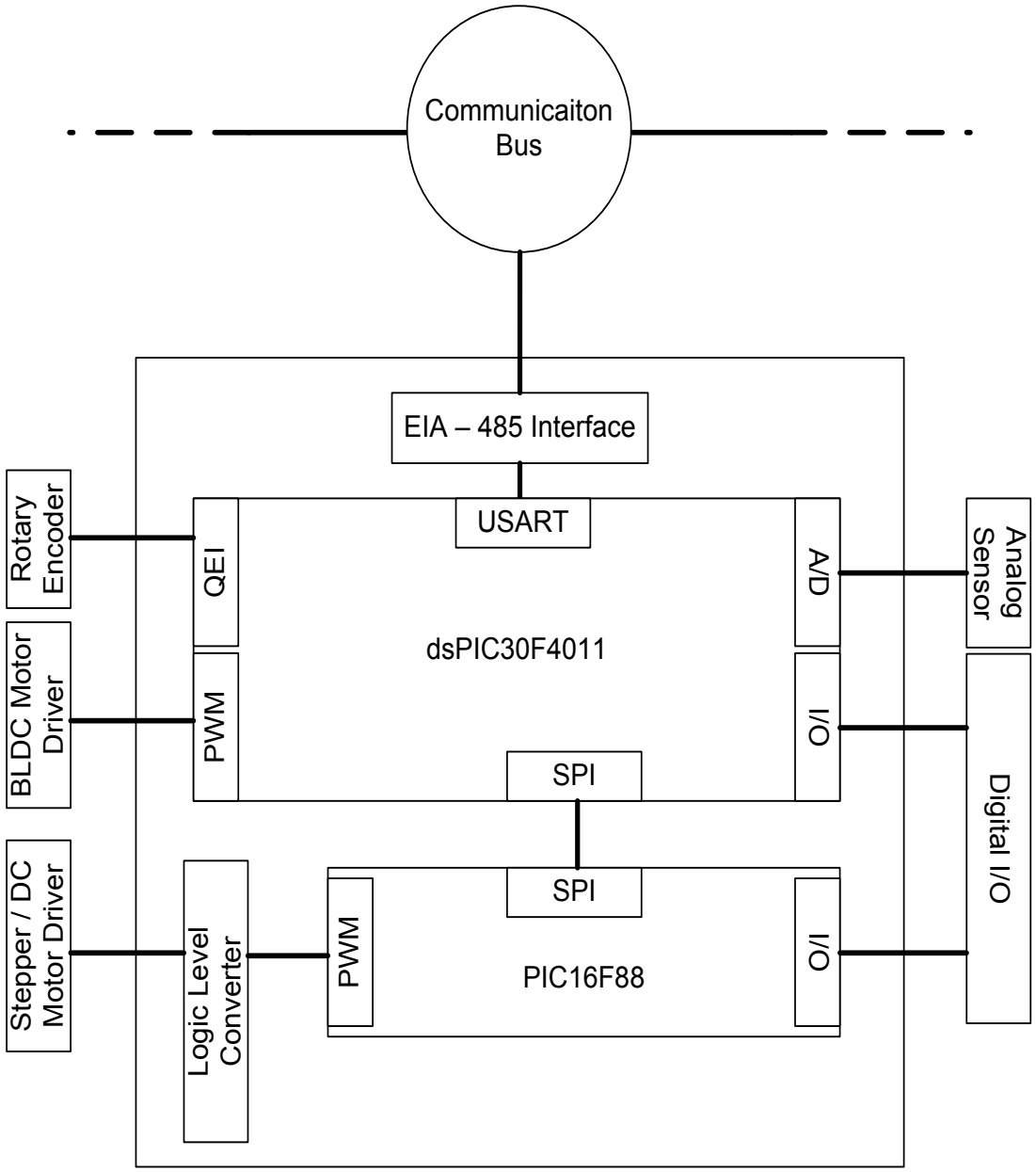


Figure 4.1 Hardware block diagram

It is critical to note that using dsPIC30F4011's quadrature encoder interface, one can perform closed-loop control of electrical motors (AC, DC servomotors). Designed controller has brush-type DC, brushless DC and stepper motor interface. Utilizing analog to digital (A/D) converters, analog sensor interfacing can be implemented.

One of the UARTs of dsPIC30F4011 is connected to EIA-485 transceivers. Two of MAX485 EIA-485 transceivers utilized in order to interact with controllers and full-duplex communication network. This enhancement makes the controller a part of the multi-point communication network. The PCB of the developed controller is given in Fig. 4.2 while Figs. 4.3, 4.4 and 4.5 illustrate its circuit schematics.

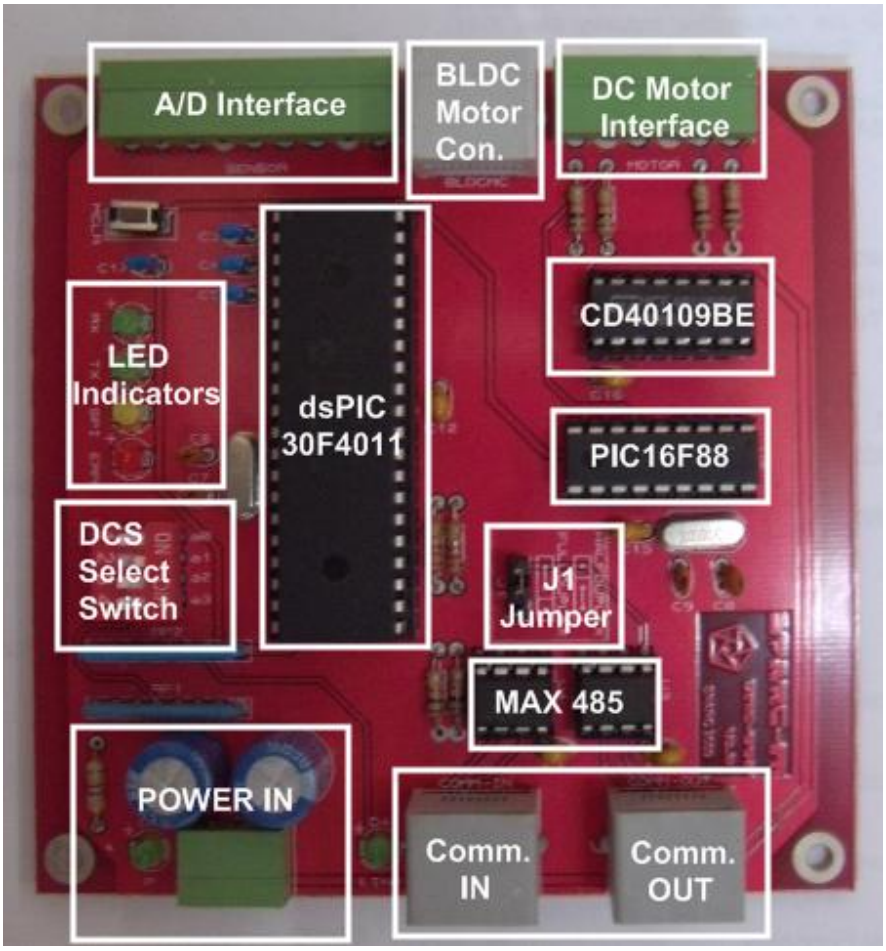


Figure 4.2 Controller PCB

The controller can work with either half or full duplex communication networks. Using the J1 jumper on the PCB, one can switch between the communication modes.

dsPIC30F4011 and PIC16F88 use SPI communication interface to communicate each other where dsPIC30F4011 masters the SPI communication.

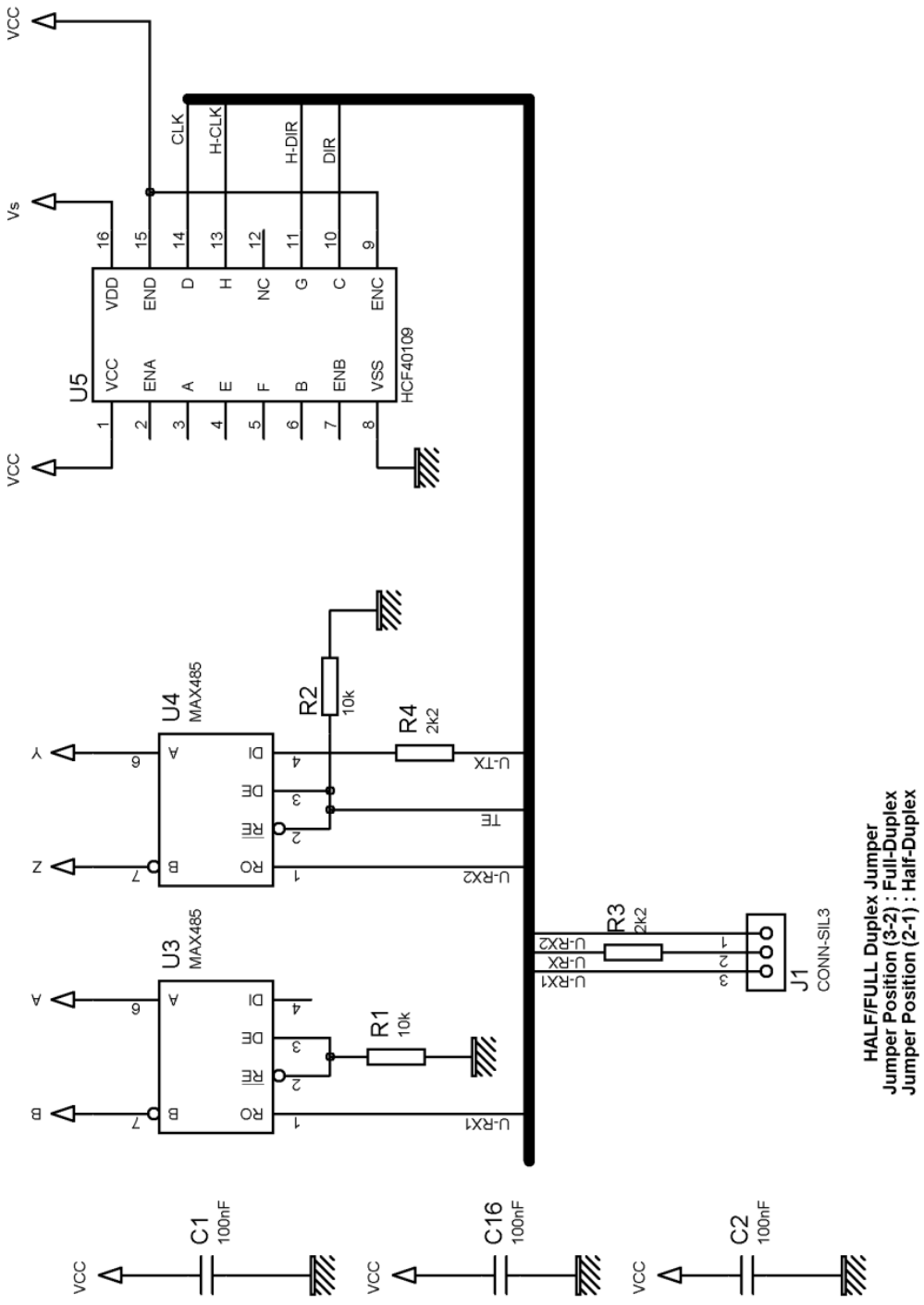


Figure 4.3 Communication interface

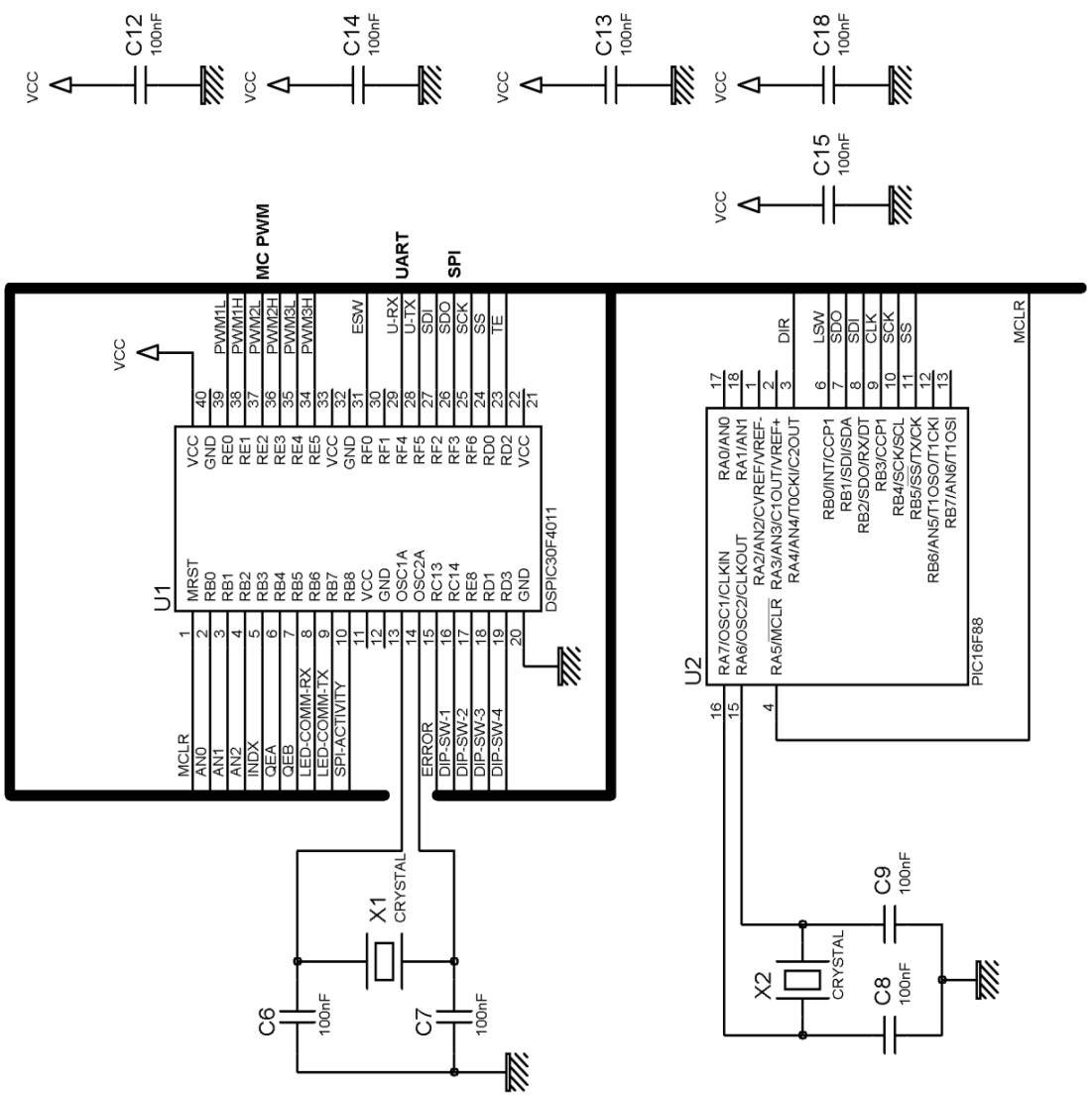


Figure 4.4 Main controller

.

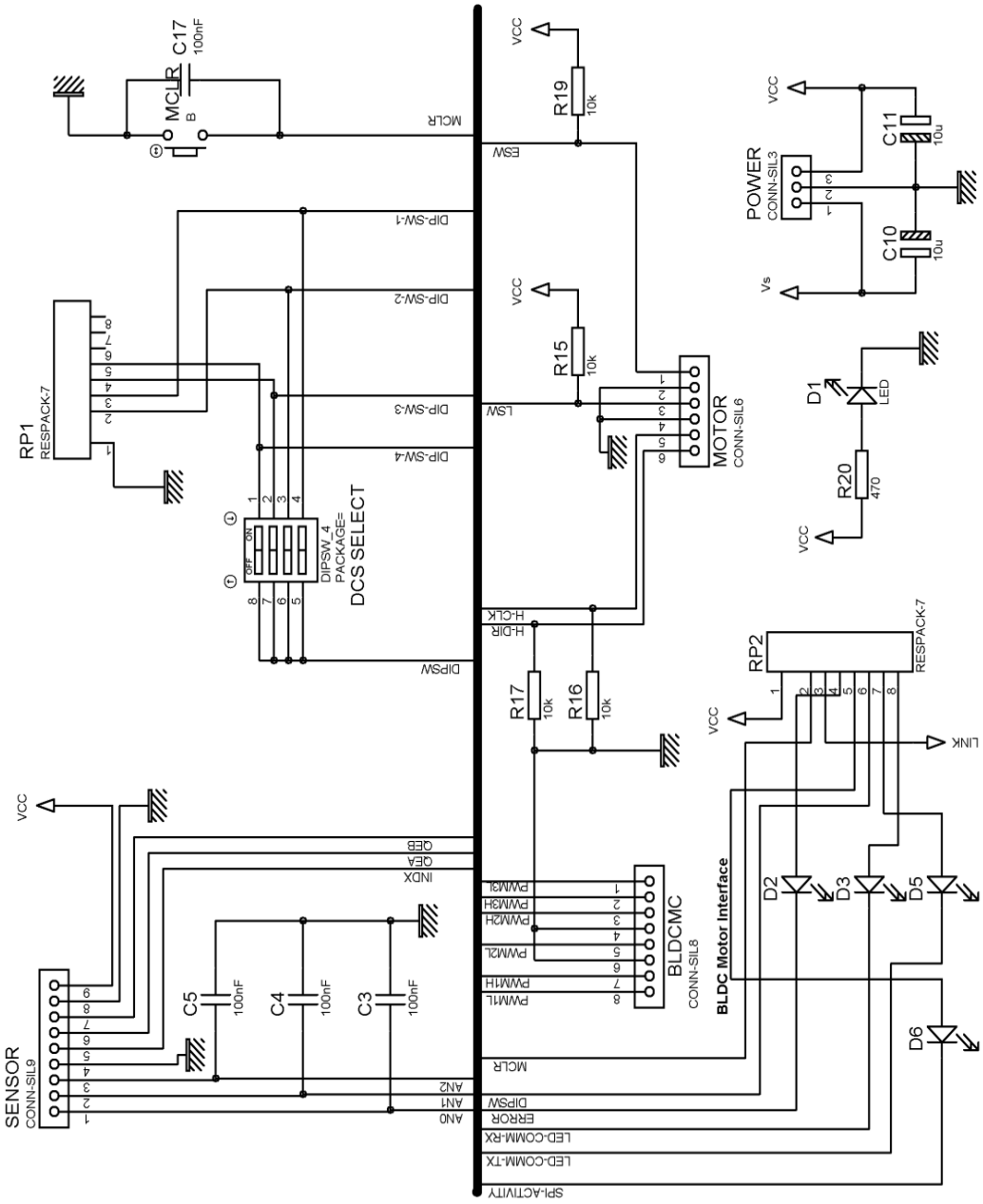


Figure 4.5 Peripheral connections

Please note that PIC16F88 is dedicated to perform command signal generation for stepper motor drivers. This 8-bit microcontroller relieves considerable processing load on the dsPIC30F4011. Here, CD40109BE (buffer) is interfaced with the

PIC16F88. For proper connection between controller and motor driver that operate on different logic signals (CMOS, TTL). It contains low-to-high voltage level shifting circuits featuring individual three state output capability. Signal level conversion is widely used in interfacing circuits.

Several experiments were conducted on CD40109BE (CMOS quad low-to-high voltage level shifter IC) during the design phase. Duration of the conversion process has a direct effect on the control system performance. In order to identify performance of the CD40109BE, different frequencies of clock signals are applied and response of the CD40109BE is observed. Fig. 4.6 summarizes the results.

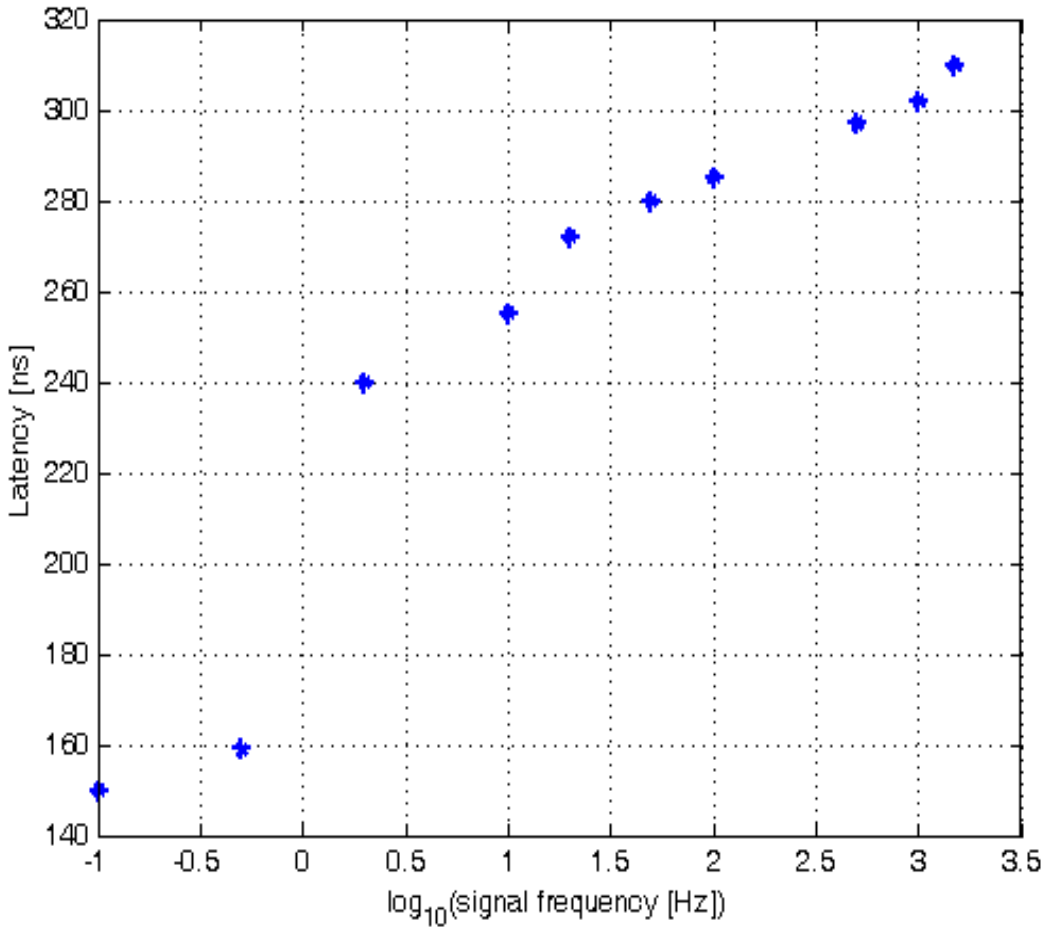


Figure 4.6 Signal frequency versus latency

As presented in Fig. 4.6, the controller performance diminishes with the increasing frequency. Actually, during the testing of the CD4019BE's limits are pushed and it is expected to handle pulse trains at 2 MHz - a signal which will never be generated by controller itself. As can be seen in Figs 4.7 and 4.8, a command TTL signal (which is presented as light trace on the scope screen) is applied to CD40109 and it is expected to shift this signal to the CMOS (10V) level. Shifted signal is represented as a dark trace on the figures. As can be clearly seen in Fig. 4.7, 2MHz of clock signal is well beyond the operating zone of the CD40109BE's limits. In order to address dynamic response of the chip, the clock frequency reduced to 100 KHz. As it is presented in Fig. 4.8, CD40109BE can properly shift those signals to the 10V level. Thus, 100 kHz is also considered as command signal limit for this hardware configuration.

Figure 4.7 Command signal (2MHz)

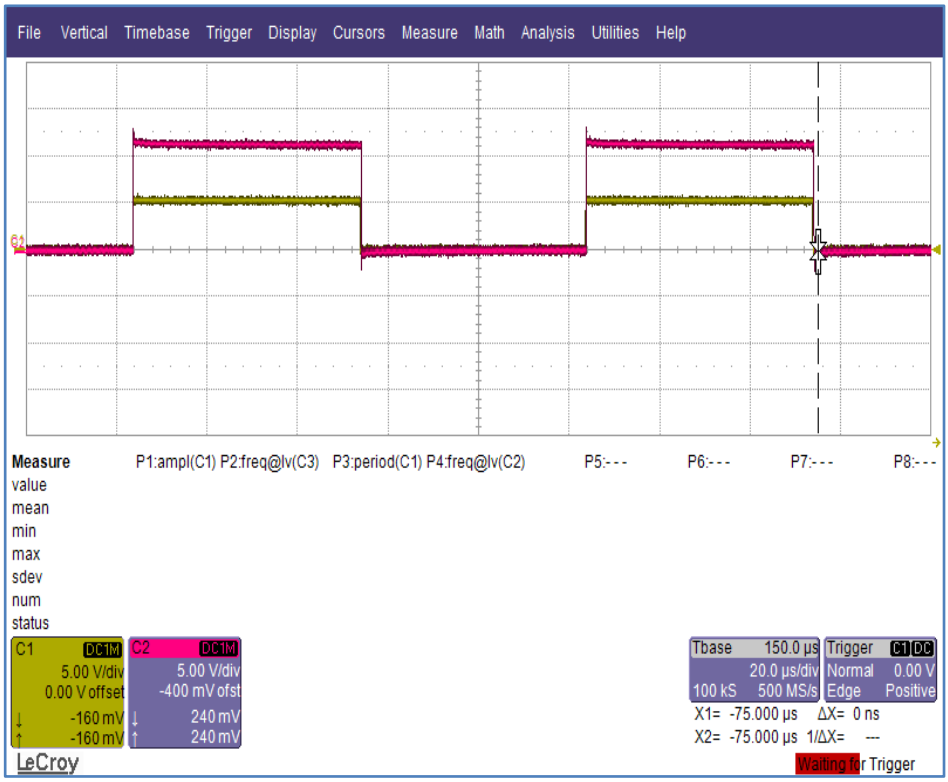


Figure 4.8 Command signal (100kHz)

4.3 Firmware

Firmware for both dsPIC30F4011 and PIC16F88 MCU's developed on the CCS C IDE [80] which is presented in the APPENDIX B section. Also functional block diagrams are given in the Fig. 4.9. As discussed in Chapter 3, the program starts out with the reception of one byte whose element strictly defined in communication protocol. According to this byte pair, a device sets itself to perform requested operations.

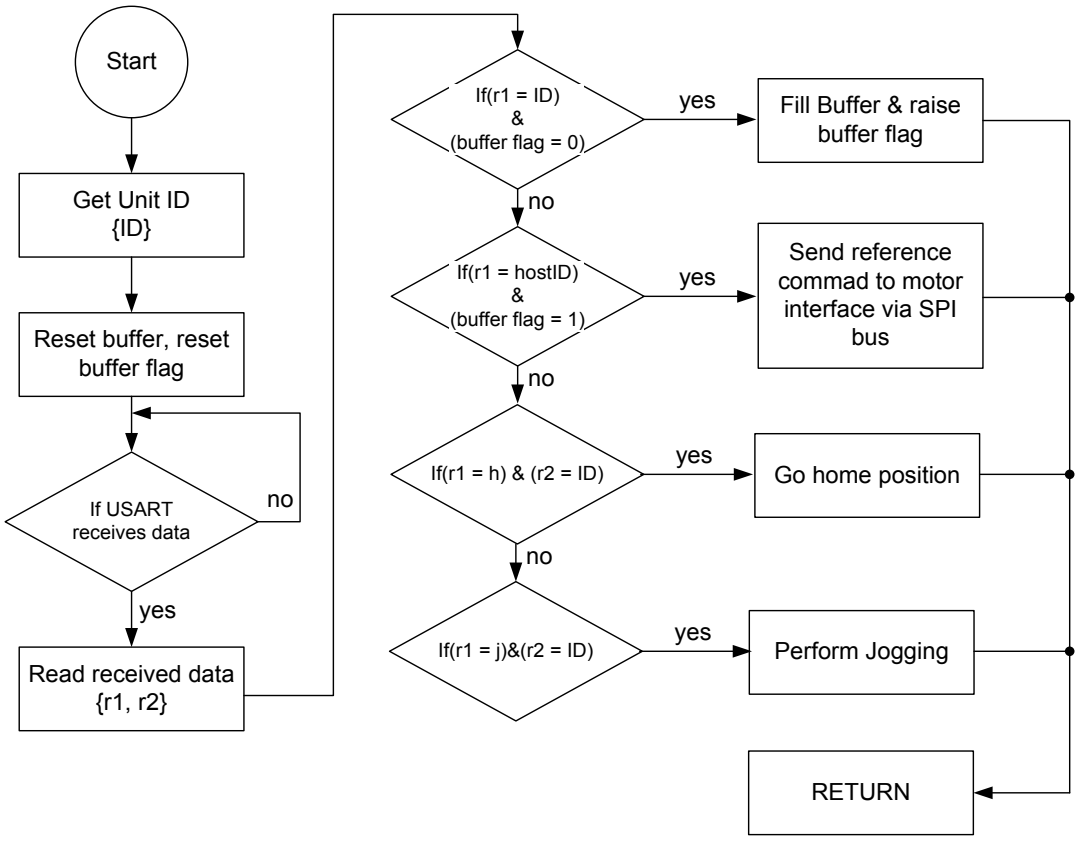


Figure 4.9 General description of firmware

4.4 Summary

Motion controller hardware, which has hierarchical structure utilizing double microprocessors, is proposed. The number of devices that can be connected to the same bus is limited only by maximum bus capacitance. Developed hardware and communication network support up to fifteen controllers. Firmware can be modified to accommodate all the peripheral sources of the microcontroller units depending on the control task at hand.

CHAPTER 5

CNC MACHINE TOOL AND APPLICATION

5.1 Introduction

The distributed control system, which is developed within the scope of this thesis, is implemented on a desktop CNC milling machine. This machine is one of the DENFORD's old milling machines (Model: STARMILL-ATC). As shown in Fig. 5.1, the STARMILL-ATC is a three–axis milling machine utilizing an automatic tool changer system. Since some of its components (including, control cards and automatic tool changer) was out-of-order, the original electronic control unit of this machine had to be completely changed within the scope of this thesis. Unfortunately, its technical documentation is not available electronically. The details about this machine follow.

5.2 General Structure of the CNC Milling Machine

As mentioned earlier, the STARMILL is a desktop CNC machine (Fig. 5.1) and is basically used for educational purposes. Three axes of the machine are actuated by stepper motors, while the automatic tool changer is driven by a pneumatic piston and the tool magazine includes another stepper motor to index the tools. Likewise, the spindle facilitates a DC motor with field windings. Table 5.1 tabulates the technical specifications of the machine.

Table 5.1 STARMILL ATC specifications

Machine Size	600 (L) × 550 (W) × 900 (H) [mm]
Machine Weight	100 kg
Table Size	350 × 130 [mm]
Travel X-Axis	150 [mm]
Travel Y-Axis	100 [mm]
Travel Z-Axis	100 [mm]
Max. Spindle Speed	2000 [rpm]
Max. Feed Rate	1.2 [m/min]
Mains Supply Requirements	Single Phase
Spindle Motor Power	0.16 [kW]
Axis Motor Type	Stepper
Input Voltage	240 VAC
Output Current	7A
Frequency	50Hz

With respect to the mechanical drive system, all axes consist of same motors as well as motor drivers. A timing belt based transmission system, which is illustrated in Fig. 5.2, reduces the angular speed of the motor and transmits the generated torque to the ball screw shaft whose pitch length (h) is 5 mm.

First timing belt pulley has 30 teeth ($z_1 = 30$) while the second timing pulley has 12 teeth ($z_2 = 12$). In full-step mode, the stepper motor generates 200 steps per revolution ($\Delta\theta = 1/200$ [turns]). Using this information, the basic length unit (BLU) of the machine can be calculated as

$$BLU = \frac{z_2}{z_1} \cdot \Delta\theta \cdot h = \frac{12}{30} \cdot \frac{1}{200} \cdot 5 = 0.01\ [mm] \quad (5.1)$$

Hence, the motion resolution for all axes is 10 microns. Next section focuses on the machine's control system.

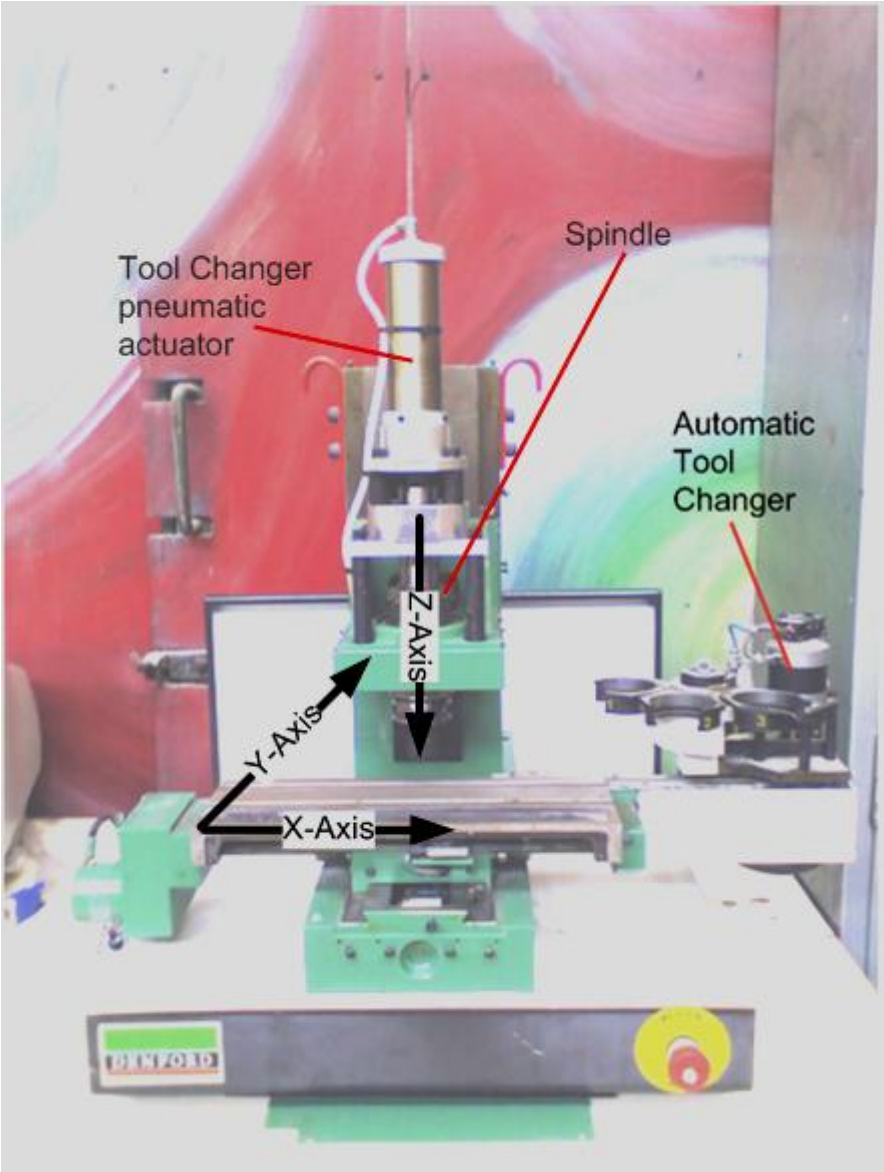


Figure 5.1 Denford' s STARMILL ATC

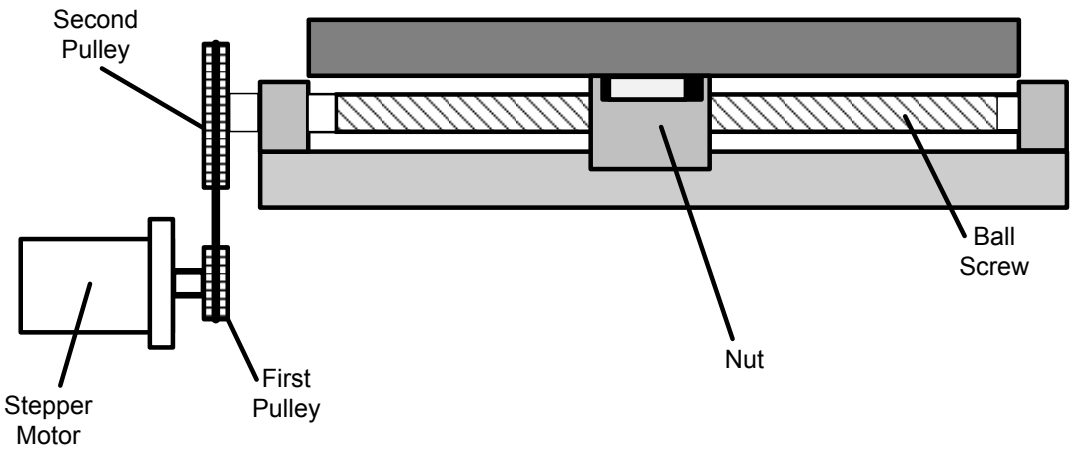


Figure 5.2 Feed drive mechanism of STARMILL-ATC

5.3 STARMILL-ATC Control System

As mentioned earlier, STARMILL-ATC employs stepper motors to drive the fundamental axes. Stepper motors' operating principles are inherently different from the DC motors which can rotate continuously as long as a constant potential difference is applied to its terminals. On the other hand, stepper motors divide a full turn into a number of small steps. Like switched reluctance motors; when the power control circuitry (i.e. the stepper motor driver) energizes motor windings in a proper order, the rotor rotates by a prescribed amount. The STARMILL utilizes the Parker-Hannifin (Model: SD2) stepper motor drivers as illustrated in Fig. 5.3. This motor driver, which makes good use of L298/L297 stepper motor driver ICs, can deliver a continuous (cumulative) current of 2A.

Figure 5.3 STARMILL-ATC motor drive system

Note that in the original machine, Siemens PC 612 F B1300-F405, which is a simple motion control card, is used to control not only the motor drivers to create coordinated axis motion but also all peripheral units of the machine tool. This card along with its parallel port interface is shown in Fig. 5.4. Since it is an obsolete product, its user manual is not available in any form (printed or electronical).

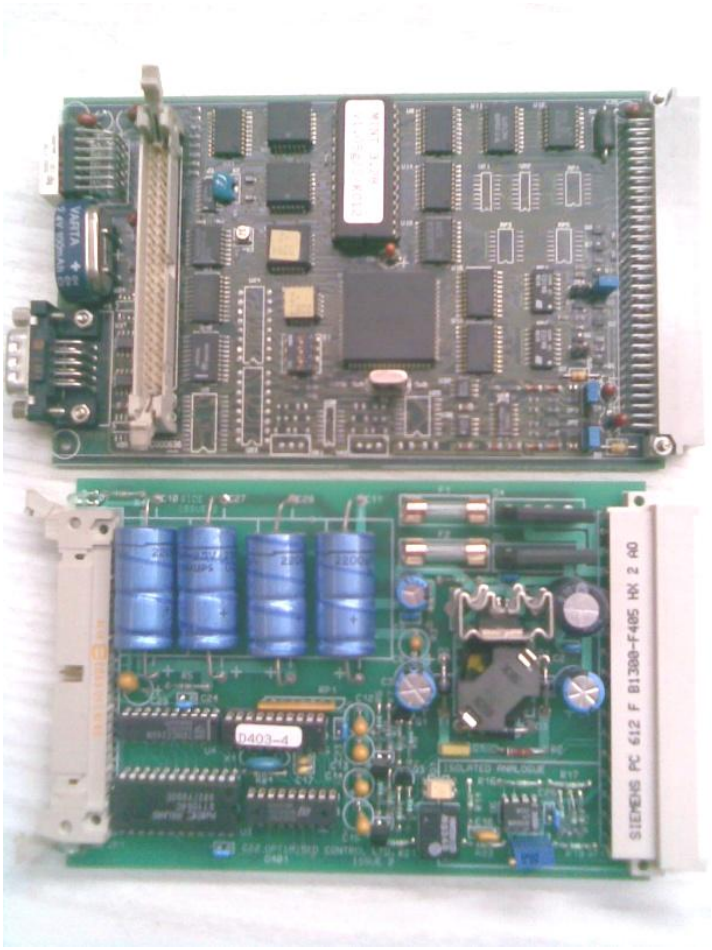


Figure 5.4 Control cards of STARMILL.

It is critical to note that the system is coupled to a PC which employs an MS-DOS based user interface/utility program. As illustrated in Fig. 5.5, it has a text (editor) window where NC part programs are loaded, saved, and edited. The software, which is mostly used for NC code verification, works in conjunction with the control cards. Unfortunately, the utility software is not supported by the DENFORD Company and a current (Windows operating system) version does not exist.

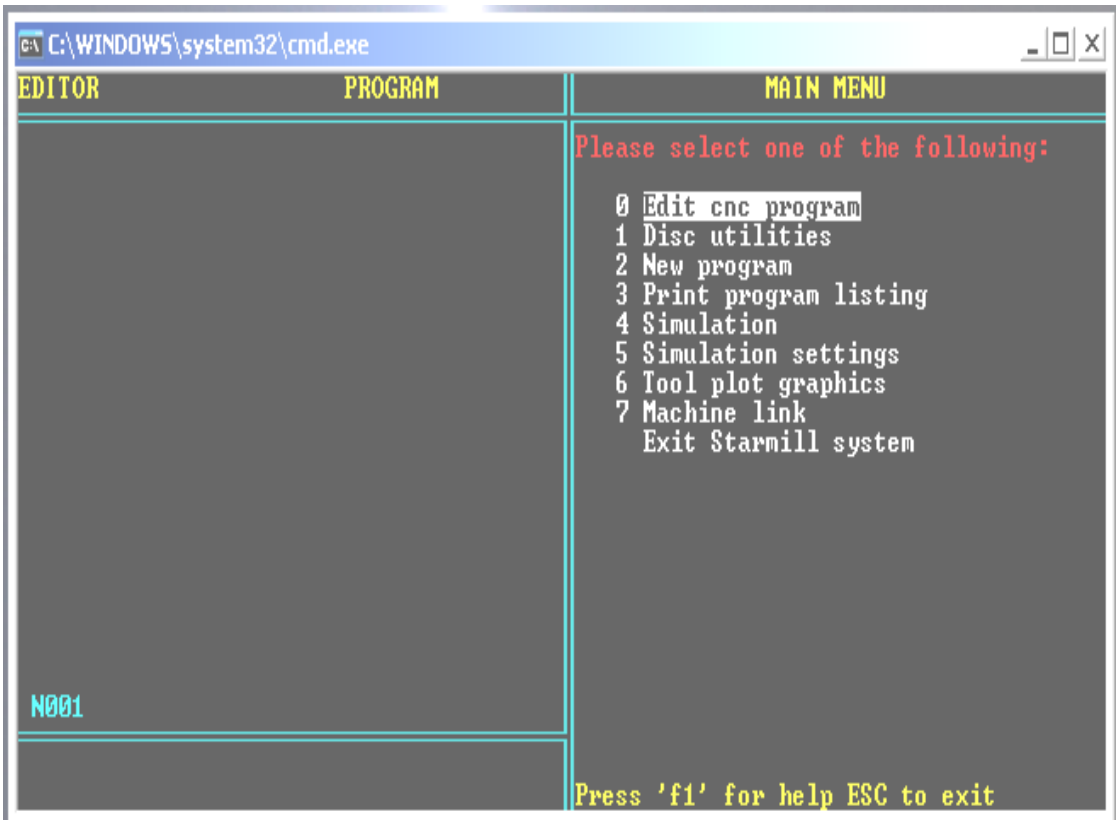


Figure 5.5 A Snapshot from the STARMILL user utility software

Notice that most of the information on the control system is collected from the DENFORD's support forums and technical support department. Hard-copy documents like motor driver manuals, cabling diagrams etc. could not be included to this thesis due to space restrictions. The following section concentrates on the application of the distributed motion control system to this "decommissioned" CNC machine tool.

5.4 Application of Developed Control System

Designed distributed controller, which is described in Chapter 4, is applied to the machine tool. The graphical user interface, which is defined in the next chapter, is closely coupled to this control system to operate the machine seamlessly. Unlike the original control system, the host PC and the controllers devised (Fig.5.6) in this thesis are connected to each other over EIA-485 fieldbus as illustrated in Fig. 5.7.

Figure 5.6 Developed control system

It is critical to notice that a standard PC does not have any EIA-485 communication interface. Hence, a PCI based communication card that supports two EIA-485 devices (two or four wired), is installed to the host PC.

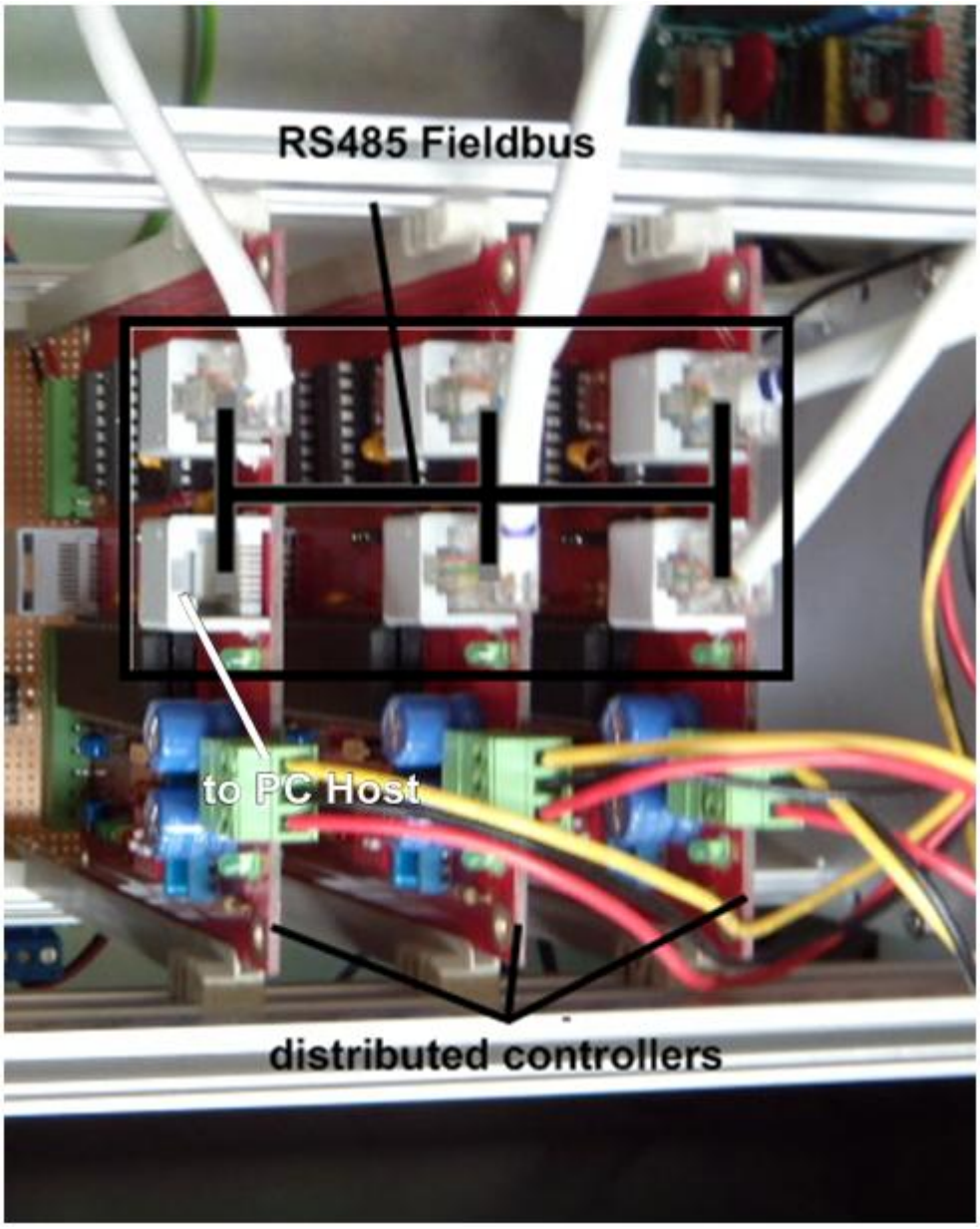


Figure 5.7 Developed control network

5.5 Performance Study

Developed controller performance needs to be studied. Since the STARMILL-ATC is not equipped with any position sensors, only the open-loop performance of the machine can be evaluated.

Note that the CNC machine tool with a BLU of 0.01 [mm] does not meet the machining requirements of the industrial applications. However, the implementation is to highlight some of the key attributes for the developed control system. Hence, large-scale application of this new system can be indirectly determined.

First, the rapid positioning capability (Experiment I) is observed. Note that in rapid positioning mode, it is not expected to sculpture surfaces but quick and accurate positioning of the cutting tool is required. Depending on the acceleration and deceleration rates of the axes in motion, high inertial forces may occur. As a consequence, the stepper motors may skip a number of steps. Therefore, a proper acceleration and deceleration of axial movement are vital to maintain the proper operation of machine tool in this mode.

In this experiment, a comparator (dial gage) is employed to test rapid positioning error of the table as illustrated in Fig. 5.8. An arbitrary trajectory is defined as shown in Fig. 5.9. Likewise, Table 5.2 lists the NC program generating this trajectory. After the execution of this program, the table is expected to return to the starting point. In all conducted experiments, no significant deviation on the dial gage is observed when the table goes back to its original location.

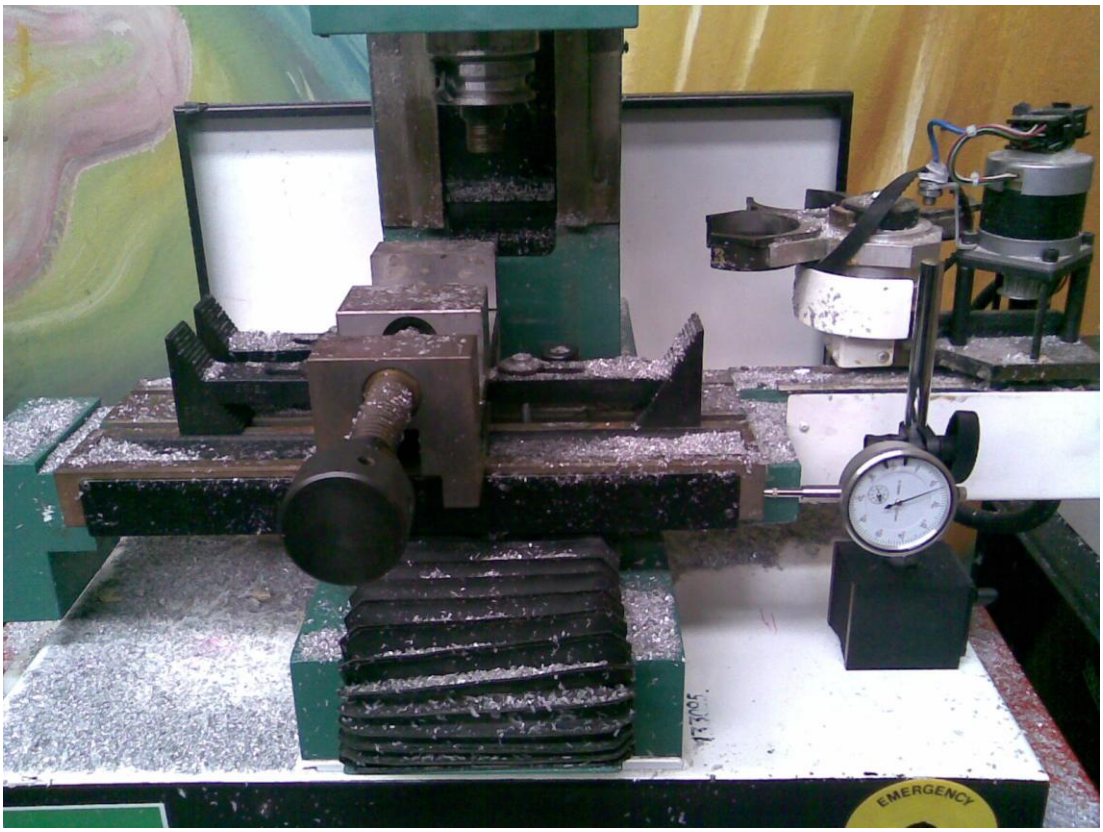


Figure 5.8 Experimental setup for the first and second experiments

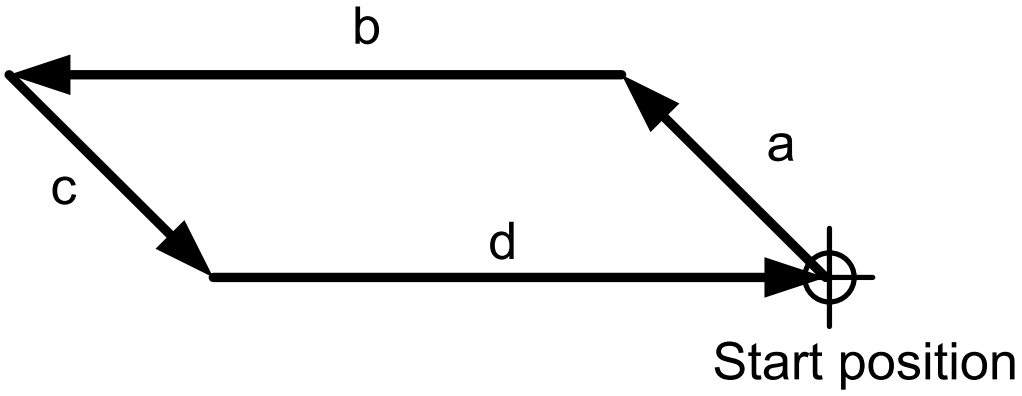


Figure 5.9 Trajectory used in the first experiment.

Table 5.2 Program listing of the trajectory in experiment I.

```
O001
N010 G21 G40 G91
N020 G0 X70 Y20
N030 X-70 Y-20
N040 M5
```

Second, sculpturing performance on an Aluminum (6061- T6) workpiece is studied. There exist several parameters that influence the forces acting on the cutter as well as the workpiece. The cutting forces (a.k.a. disturbance), are known to have adverse effects on the control system's performance. During these experiments (Experiment II), a constant depth of cut (0.5 mm) is maintained along X axis for a given feed-rate. The dial gage deviations are then recorded for various travel spans. The results are presented in Fig. 5.10.

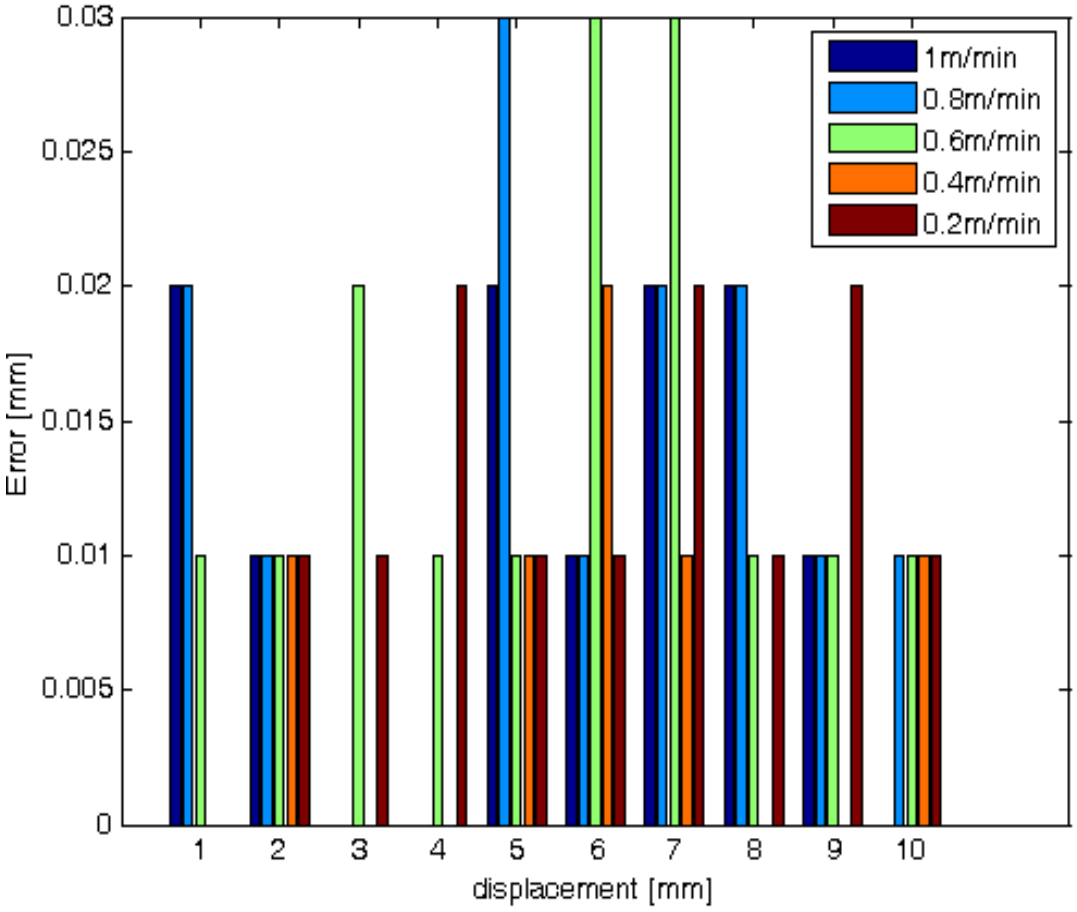


Figure 5.10 Positioning errors observed in experiment II.

Note that the cutting forces developed on a CNC machine tool causes deviation from the desired course. To compensate for the resulting positioning error, measurement devices are exclusively utilized in CNC technology to feed position information to the (closed-loop) control system. However, for an open-loop control system used in this study, any deviation from the desired trajectory accumulates in time and is left uncompensated. As consequence, the errors shown in Fig. 5.10 (1…3×BLU) are observed. It is critical to note that the vibration of the comparator stand along with the (workpiece) clamping errors do contribute to these errors as well.

As a third experiment, the machining performance of the system (in the XY plane) is investigated. An aluminum block (6061-T6) is to be sculptured. Note that this ductile material is of low strength and can be easily machined on a low-power machine tool. Fig. 5.11 illustrates the resulting workpiece.

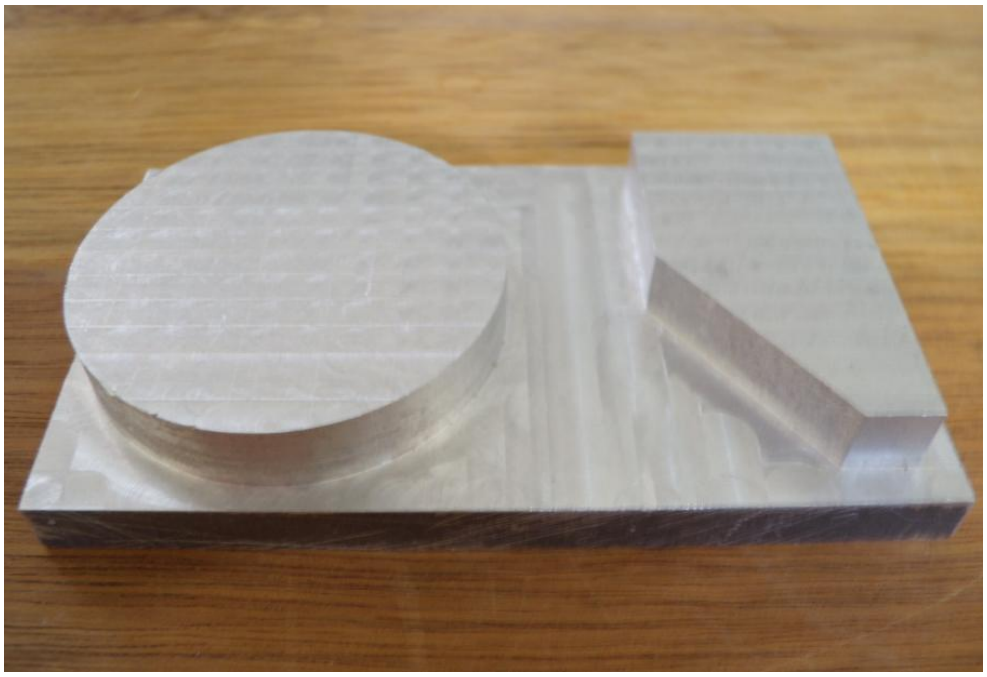


Figure 5.11 Sculptured workpiece, experiment III

A solid carbide tool, whose parameters are summarized in the Table 5.3, is utilized in this experiment. Using the available spindle power (160 W), the maximum feed-rate in full (or partial) immersion cut for the selected depth of cut (a_e) and width of cut (a_p) can be calculated. That is, the feed speed can be computed as

$$v_f = n.Z_n.f_z \left[\frac{mm}{\min}\right] \tag{5.3}$$

Power demand in the cutting operation is

$$P_c = \frac{a_p . a_e . v_f}{6.10^7 . \eta} k_c [kW]$$

(5.4)

Cutting force per mm^2 for operation becomes

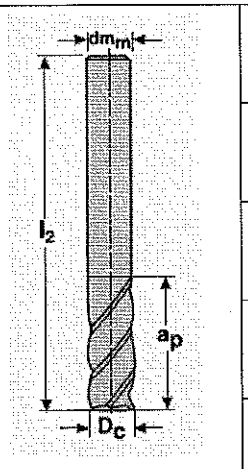
$$k_c = \frac{1 - 0.01.\gamma_0}{h_m^{m_C}} k_{c1.1} \left[\frac{N}{mm^2} \right] \tag{5.5}$$

where average chip thickness is

$$h_m = \frac{360 . f_z . a_e}{\pi . D_c . \omega_e} \sin \kappa [mm] \tag{5.6}$$

$k_{c1.1}$ = 700 [MPa] , m_c = 0.25; machining operation parameters in the Table 5.4 can be obtained [79].

Table 5.3 Cutting tool parameters



D_C	6	mm
dm_m	6	mm
l_2	65	mm
a_P	20	mm
z_n	2	-

Table 5.4 Machining parameters

a_e[mm]	a_p [mm]	ω_e [°]	h_m[mm]	γ_0 [°]	k_c[N/mm^2]	f_z [feed/ tooth]
1	6	180	0.1	14	1796.6	0.59

Employing a comparator with a resolution of 1 micron, the deviation from the desired geometry can be examined. First, a grid is drawn on the surface of the workpiece as shown in Fig. 5.12. Then, a reference point is selected where the comparator is reset. The measurements are taken at each point on the grid.

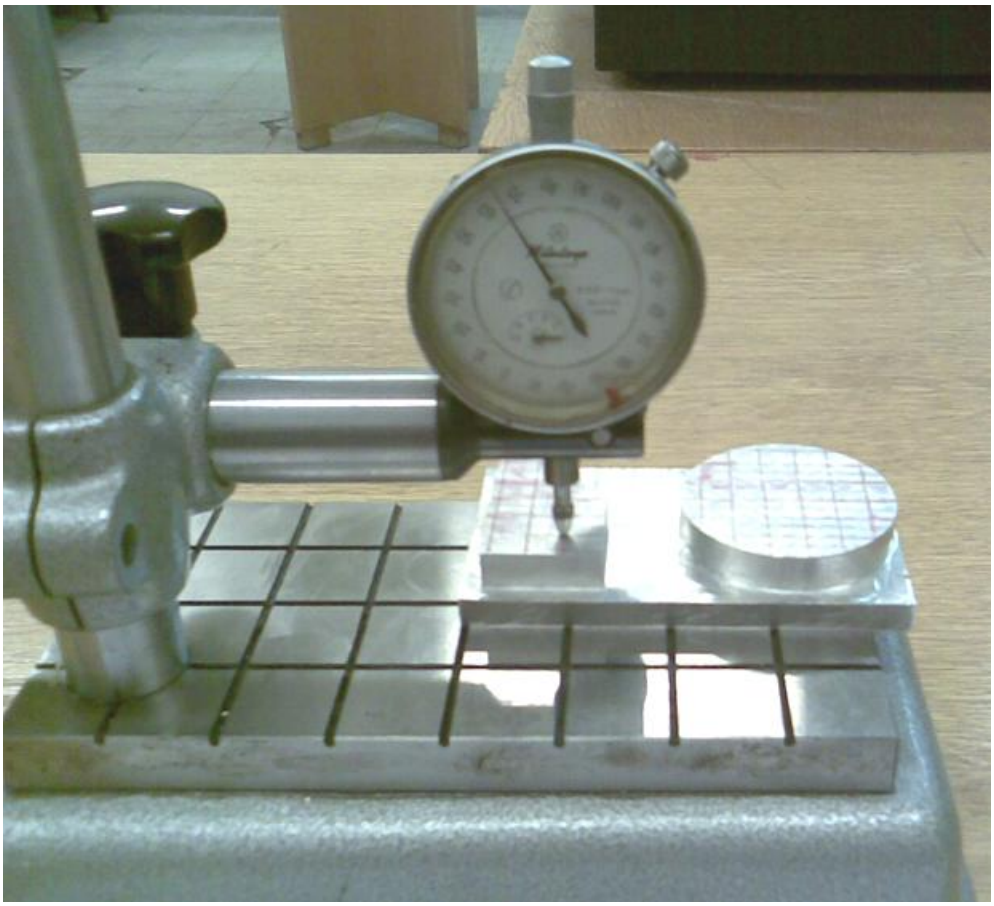


Figure 5.12 Dial gage used in experiment III

Deviation from reference point is presented in the Fig. 5.13. The biggest relative error is 0.126 mm while the standard deviation of the measurements are calculated as 0.0520 [mm].

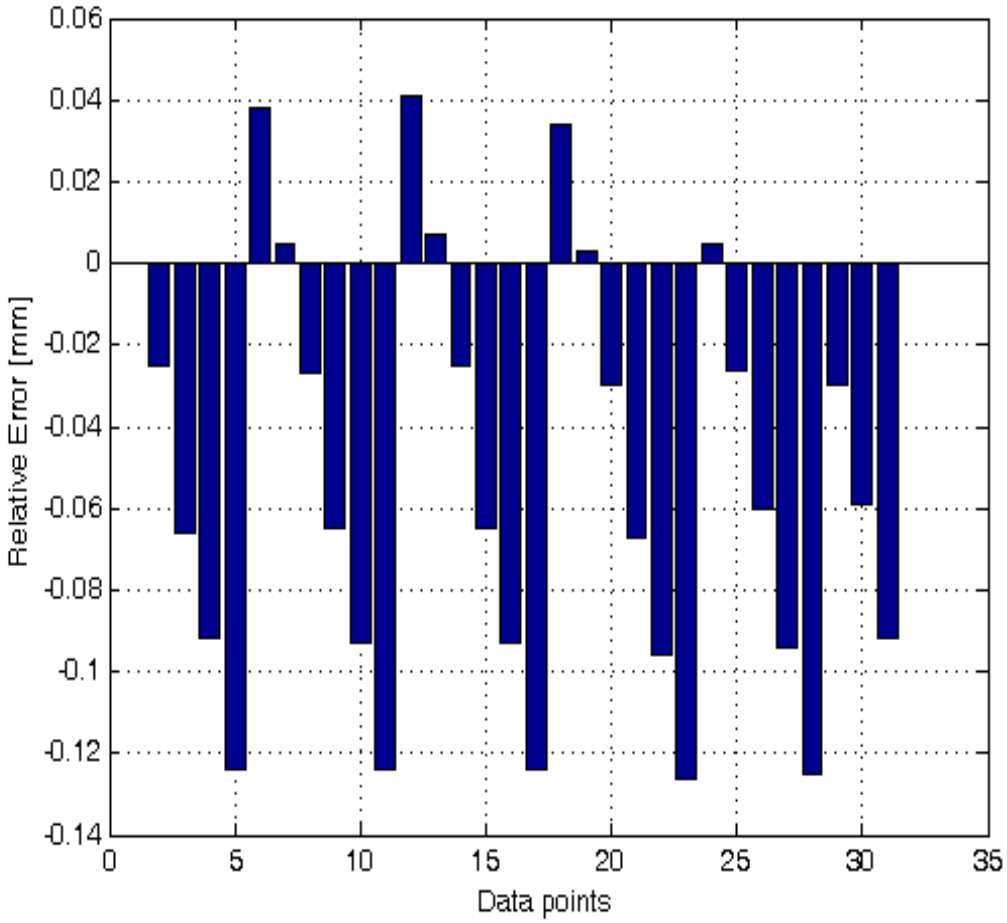


Figure 5.13 Relative error from reference point

Notice that absolute measurements on the surface can be attained by resetting the comparator with the utilization of Johnson gage blocks as demonstrated in Fig. 5.14. Once the reference height is set, the measurements can be taken on the grid. Using collected data, surface plots for circular- and rectangular sections on the workpiece are obtained as shown in Figs. 5.15 and 5.16.

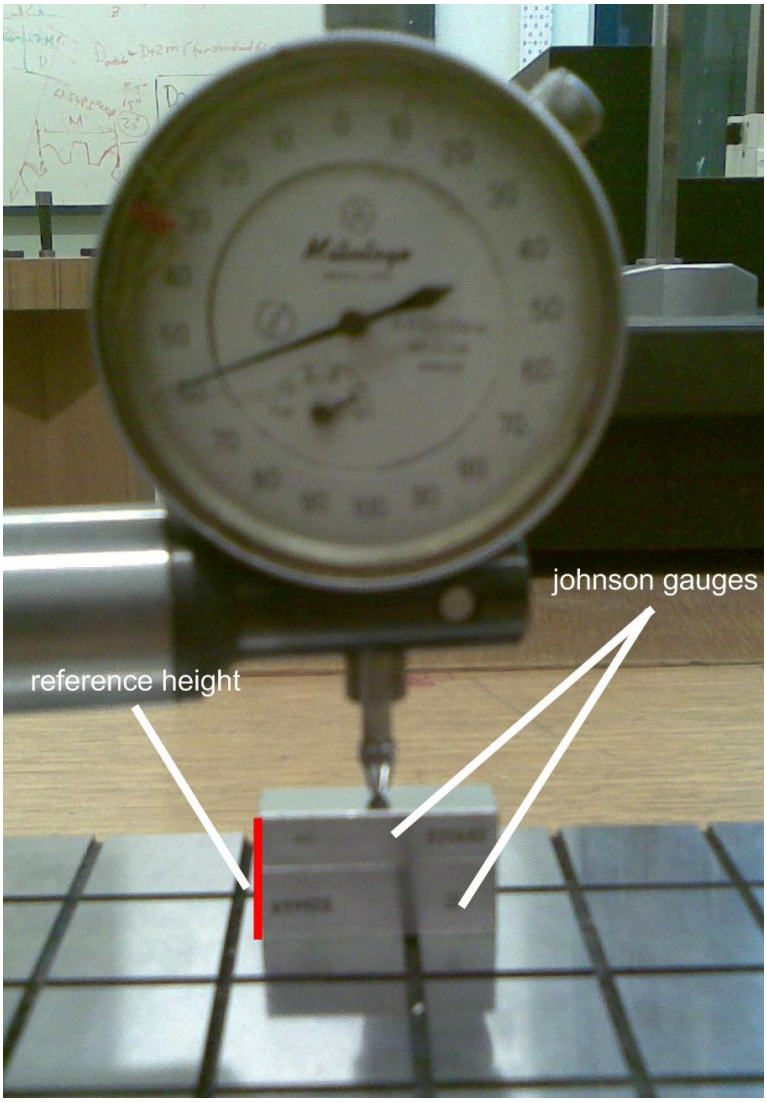


Figure 5.14 Johnson gauge sets used in the experiment

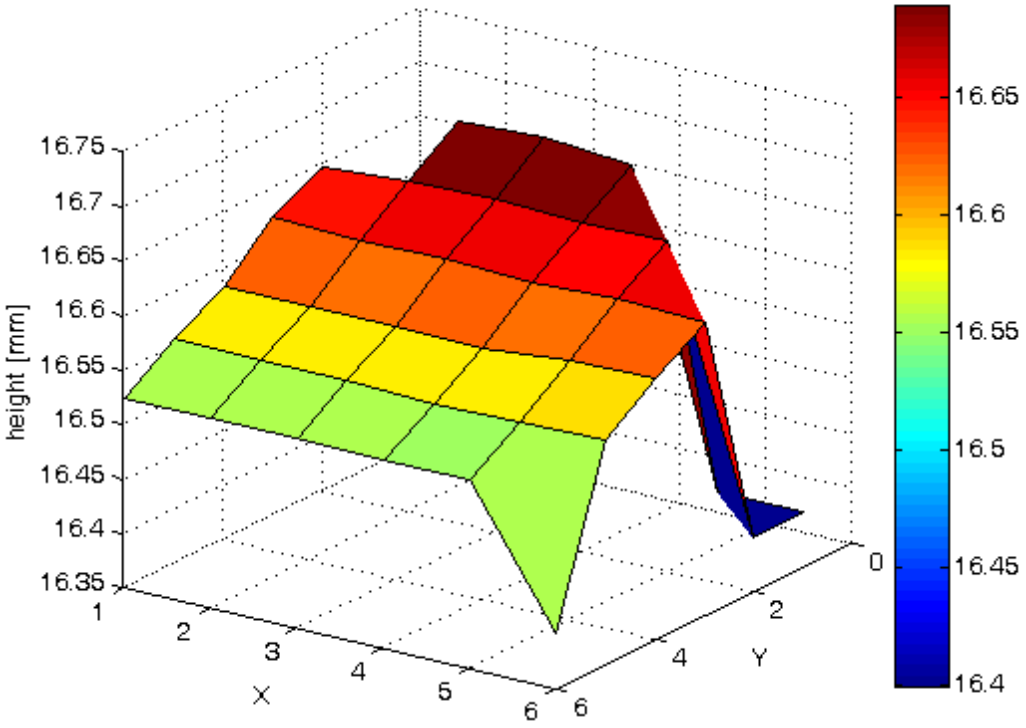


Figure 5.15 Surface plot, circular section

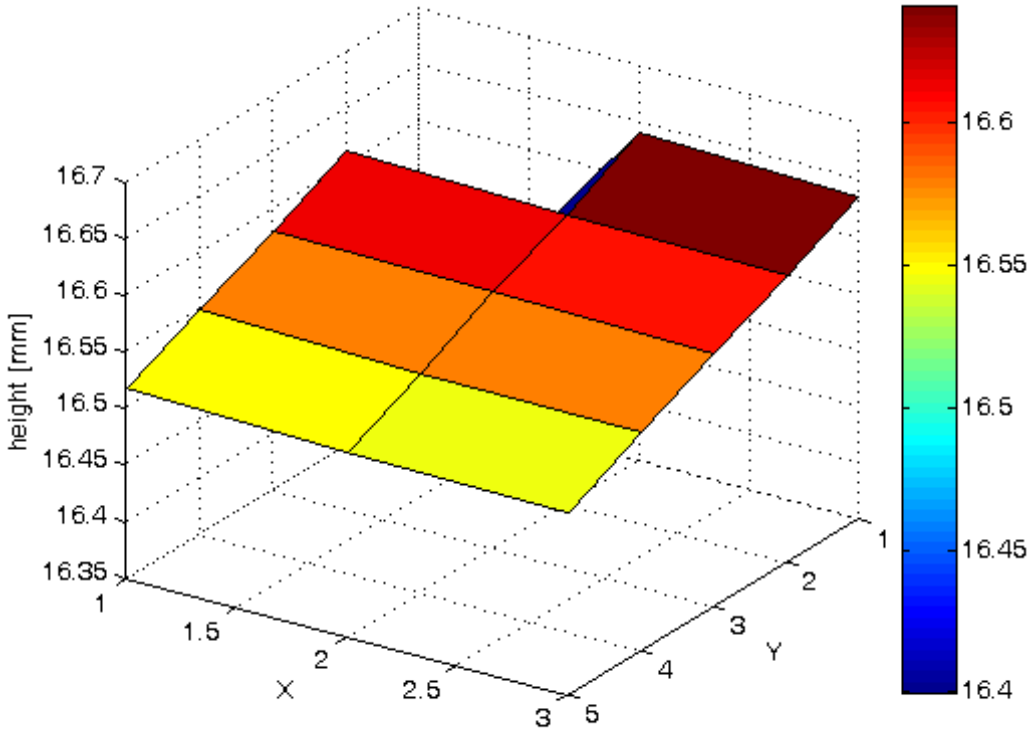


Figure 5.16 Surface plot, rectangular section

According to given plots, the average height of the circular path is 16.5965 [mm] while the average height for the rectangular section is 15.5719 [mm]. As an interpretation, some waviness along the X direction is observed owing to the fact that during the machining operation, the Z axis motor is at rest. Therefore, height differences for each pass might not be attributed to the Z axis motor. It is probably related to the geometrical errors associated with the clamping device (vise). Fig.5.8 illustrates the vise, which is used for machining operations.

Using a digital compass (with a resolution of 10 microns), certain dimensions of the part is measured as illustrated in Fig. 5.17. As can be seen from Fig. 5.18, some deviations from the nominal dimensions are observed. Maximum error occurred in 3rd data point which is 0.12 [mm] average is 0.035 [mm]. Note that at the beginning positioning commands for that circular path may include errors because of converting floating point numbers to integer ones. Fig. 5.19 contains a graphical representation of evaluation of the circular trajectory

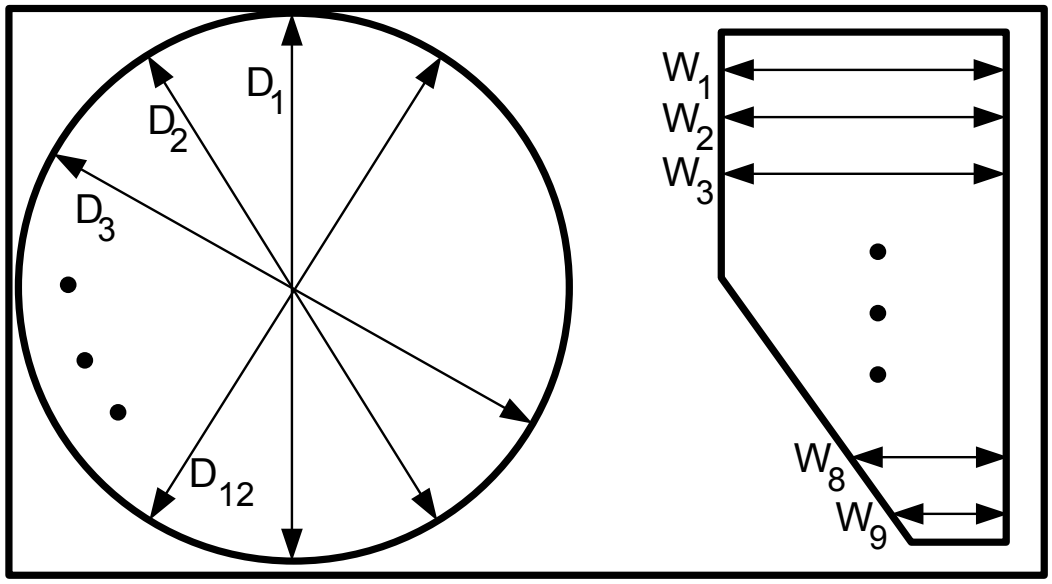


Figure 5.17 Measurement taken on the workpiece

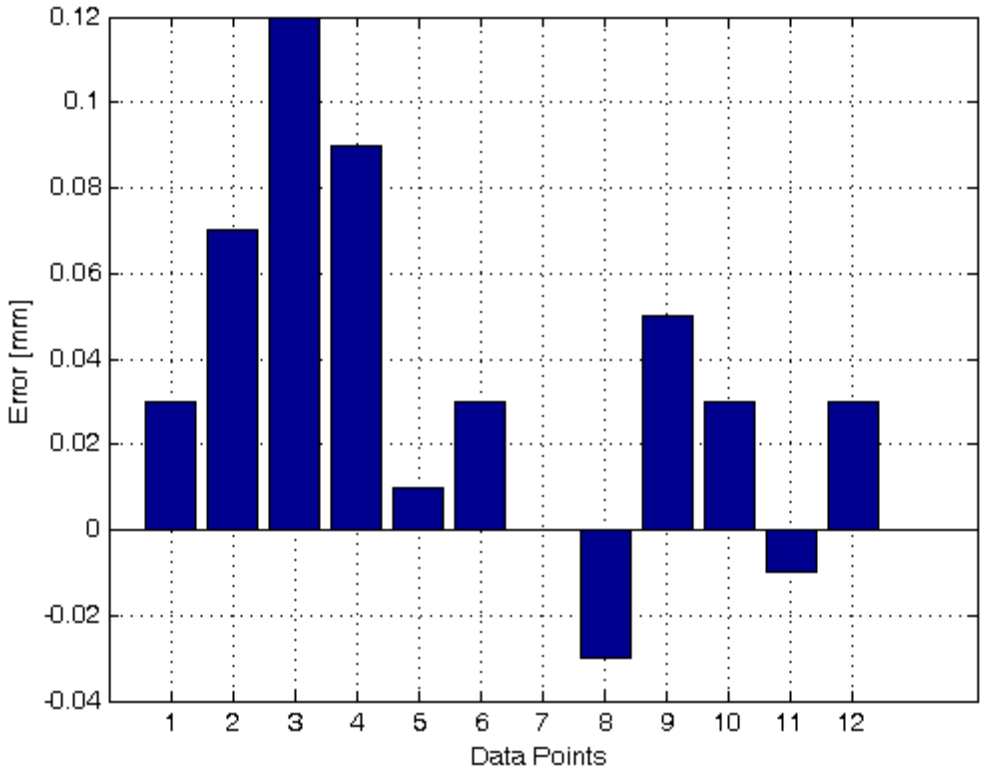


Figure 5.18 Tracking Performance of the Circular Path

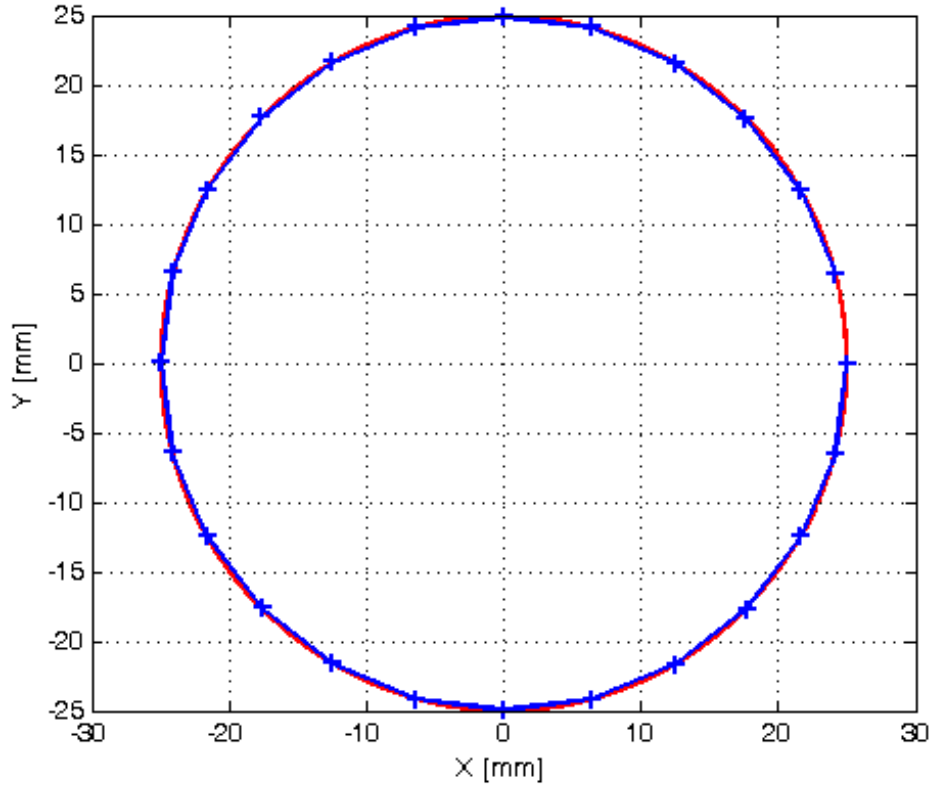


Figure 5.19 Evaluation of the circular path

Rectangular part of the workpiece is measured by the digital compass. Results are presented in the Fig. 5.20. Measurement error increases at the inclined part of the rectangular section. It is obvious that a digital compass is not proper tool to measure inclined surface.

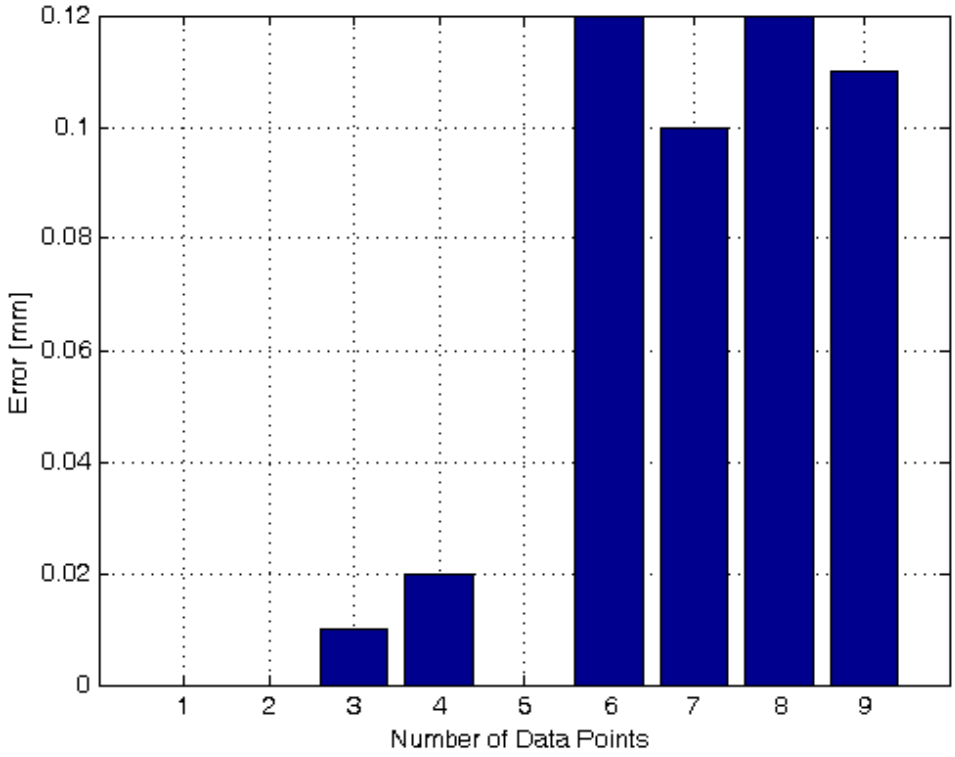


Figure 5.20 Tracking performance of the rectangular path

The object of the last experiment is to investigate 3D interpolation capability of the system. Actually, parameters of the performance investigation are related to not only the controllers but also the other components (motors, motor drivers, machine components, etc.). During the experiments, calculated cutting parameters are used. Workpiece is given in the Fig. 5.21. Note that the selected workpiece has diverse attributes that allow the investigation of coordinated motion for all motors. It has a truncated pyramid in the middle and circular patterns at four sides of the workpiece.

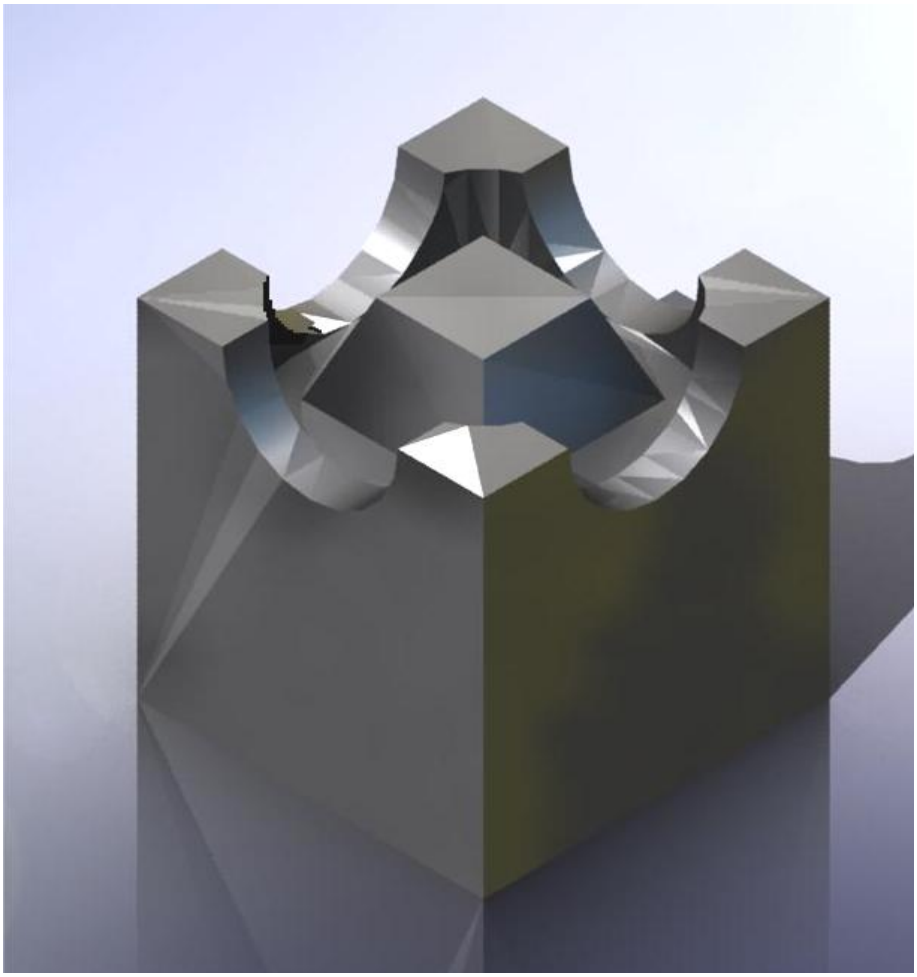


Figure 5.21 Workpiece definition in the CAD environment

Notice that STARMILL – ATC itself was designed for 2½ axes operations that only two of their axes (X and Y) move simultaneously at any one time. Motor and driver selection are made according to this criteria. On the other hand, the developed control system is suitable for n-axes coordinated motion. Therefore, the controller pushes to the limit of machine tool. Sculpturing performance is investigated with the previously calculated machining parameters. Also, higher feed rates and cutting depths are avoided to stay within the set operating parameters.

While sculpturing the truncated pyramid, axial speeds of X, Y and Z are constant. During the performance evaluation, rapid change in the speed of the Z - axis introduces large amount of the positioning error. Hence, in the absence of the position sensors, the corresponding errors accumulate. As a consequence, the workpiece is clearly deformed as presented in Fig. 5.22.

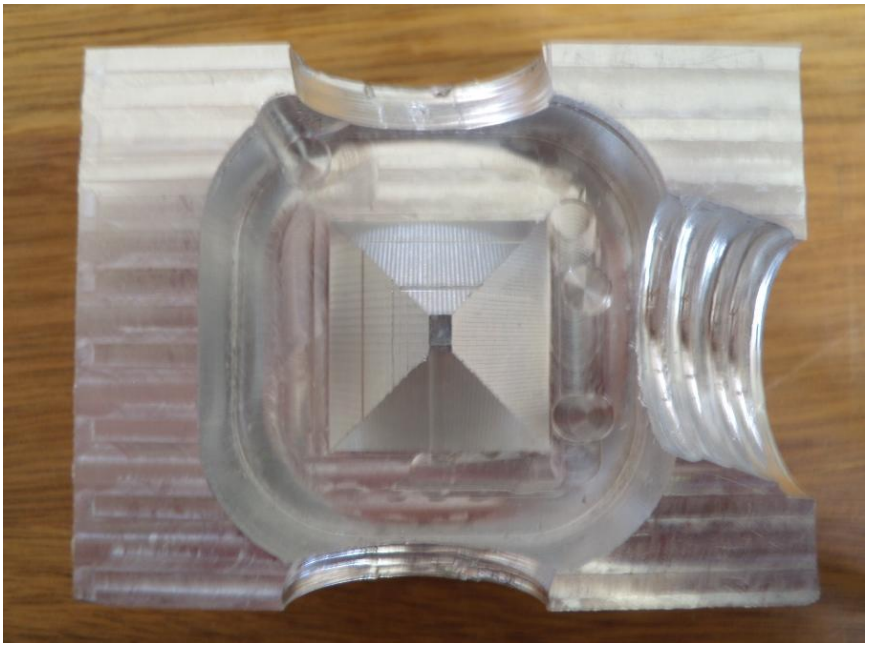


Figure 5.22 Badly deformed workpiece

In Chapter 6, the verification of the command generation is presented. Illustrated deformation, which is mainly observed in Z direction, is caused by rapid change in the axial speed of that motor. Hence, the motor driver cannot deliver the required current to drive the motor properly.

The workpiece is finalized by clamping the corresponding planes to XY plane. Performing 2D circular interpolation in the XY plane yields the desired geometry as shown in Fig. 5.23.

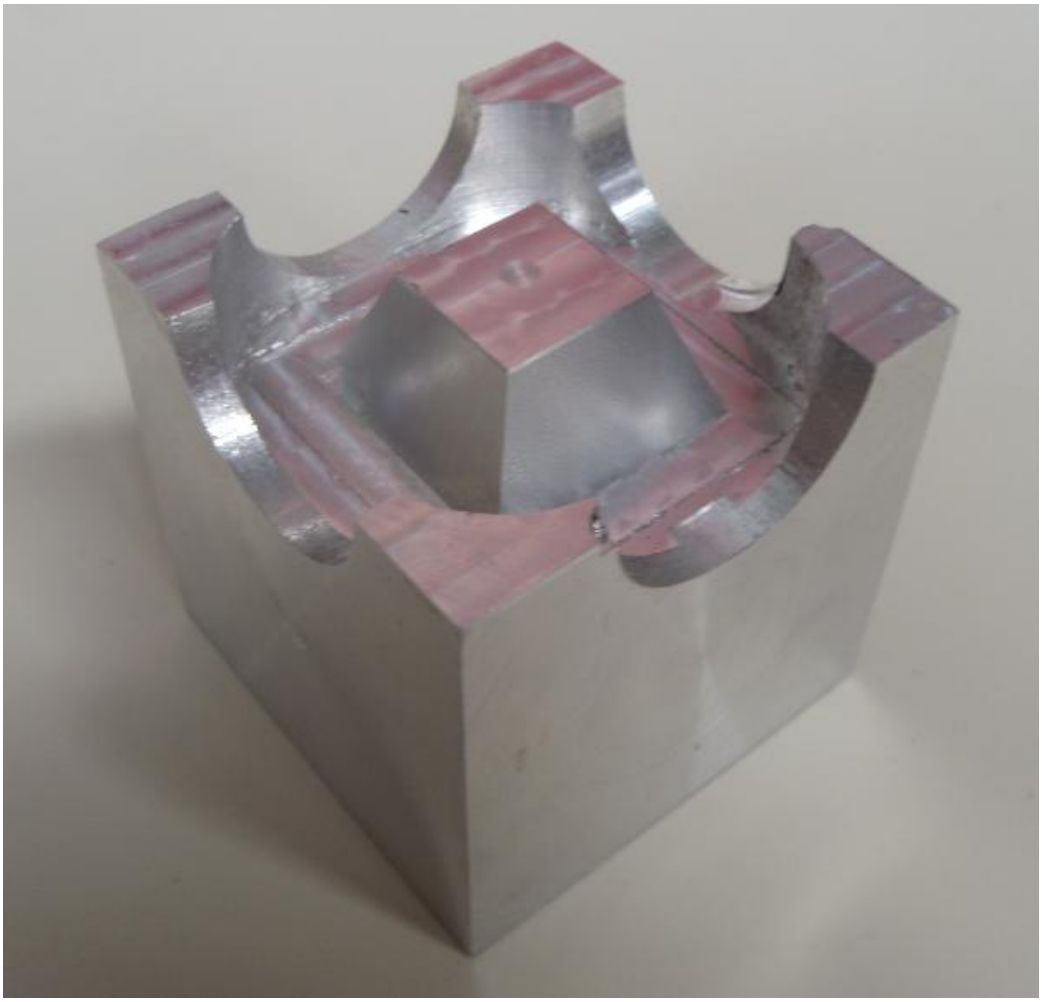


Figure 5.23 Final workpiece

5.6 Summary

Utilizing developed hardware, several experiments on the machine tool were conducted. Open-loop performance is observed during those experiments. A low strength (ductile) material for the workpiece is selected to reduce cutting forces / power. The machining performance of the resulting CNC machine was found promising for machining operations in the XY plane. According to experimental results, CNC milling machine, which was controlled by the devised hardware, had a machining accuracy of ±0.05 [mm]. As presented, the vibrations of the machine tool had an adverse effect on the cutting performance when the cutting parameters (depth of cut, feed-rate, spindle speed, etc.) were not selected properly. As it was presented, Z-Axis has lack of positioning performance. It seems to be properly selected by machine designers. During the experimental study it was concluded that rapid changing direction and speed cannot be tracked by motor driver. Hence large amount of positioning error accumulates. More powerful motor driver selection is essential for adequate operation of motors.

CHAPTER 6

GRAPHICAL USER INTERFACE AND ITS FEATURES

6.1 Introduction

Graphical user interface (GUI), enables users to control and operate the CNC Machine Tool fully. As opposed to text based interfaces, GUIs offer graphical icons, visual indicators to fully represent the information and actions available for the user. All of the program controls performed through direct manipulation of the graphical elements.

Designing visual elements, combination and representation of them are the most important points of the GUI design. Popular CNC controllers have their distinctive GUIs. One of the popular CNC machine controller manufacturer, Fanuc, employs a diverse GUI for its products which is presented in the Fig. 6.1

For short, a GUI encapsulates visual gadgets and controller objects. A good user interface design is directly driven by the users not by the system components. Most of the CNC machine controllers utilize visualization of tool path in operation, a text editor, numerical position of the cutting tool and simulation of the NC part program.

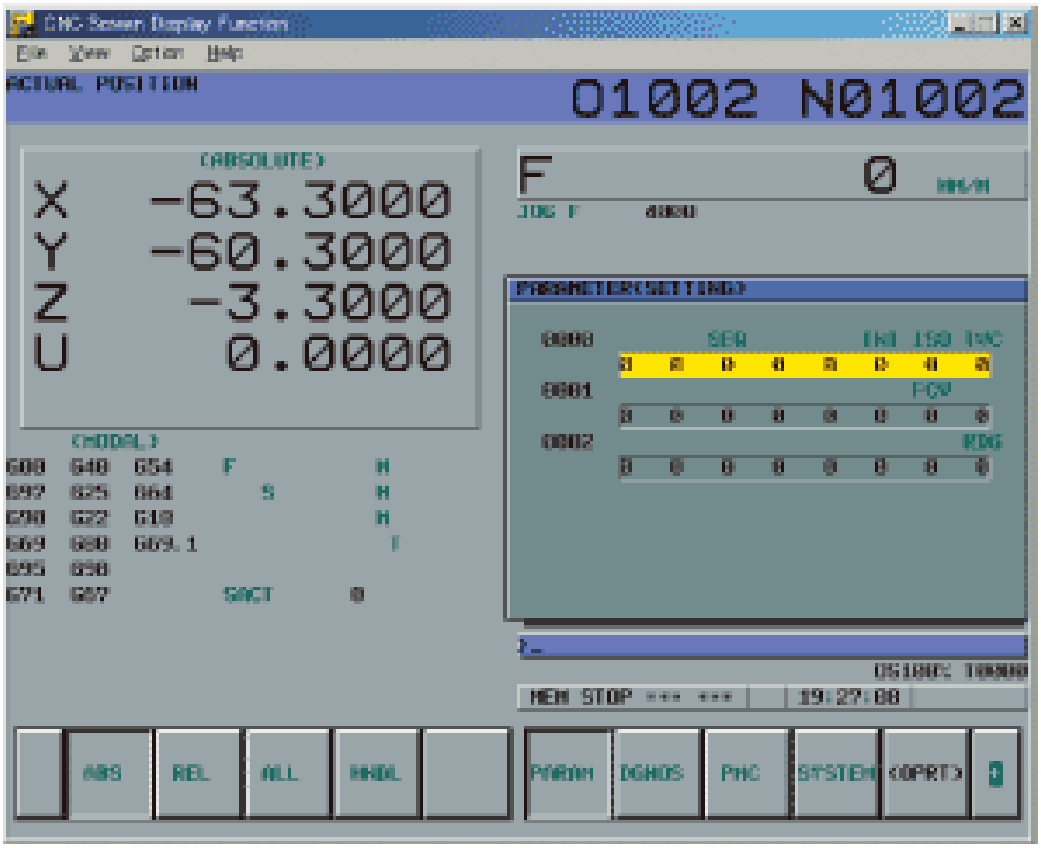


Figure 6.1 Graphical user interface of the FANUC CNC control unit [85]

Behind the GUI, there exist many control classes which are dedicated to perform machine control and operation, communication, interpretation of the part programs interaction with CAM modules of the CAD programs etc. Basically development of an advanced GUI is out of this thesis's scope. Developed simple GUI represents development of a simple GUI process as machine control software and includes usage of the machine dll.

GUI can be identified as:

- **GRAPHIC WINDOW** for 2D representation of the tool on XY, YZ and ZX planes

- **ACTUAL POSITION WINDOW** for numerically display of feed rate, spindle speed, and actual position of the cutting tool
- **STATUS WINDOW** for presentation of the NC part program operands
- **EDITOR WINDOW** for loading, saving and editing NC part programs
- **PREWIEW MODE** for simulation of NC part program in order to prevent improper NC part program execution.

6.2 FMILL Software Program of SPARCMILL

FMILL program is developed for proper operation of developed distributed controllers. Unfortunately it cannot be adapted any CAM program but, utilization NC part program editor, one can compose and execute an NC part program. GUI is presented in the Fig. 6.2.

It is a simple CNC machine interface that can provide basic operator demands. Additional indicators are also added to the program. Actual position window gives relevant absolute position according to the previously set home position. Scrolls which are allocated in the actual position window give the cutting tool position in machine coordinate systems and user can interpret the imminence to the limit switches. Spindle speed and feed rate indicator windows are placed to indicate the present spindle speed and feed rate.

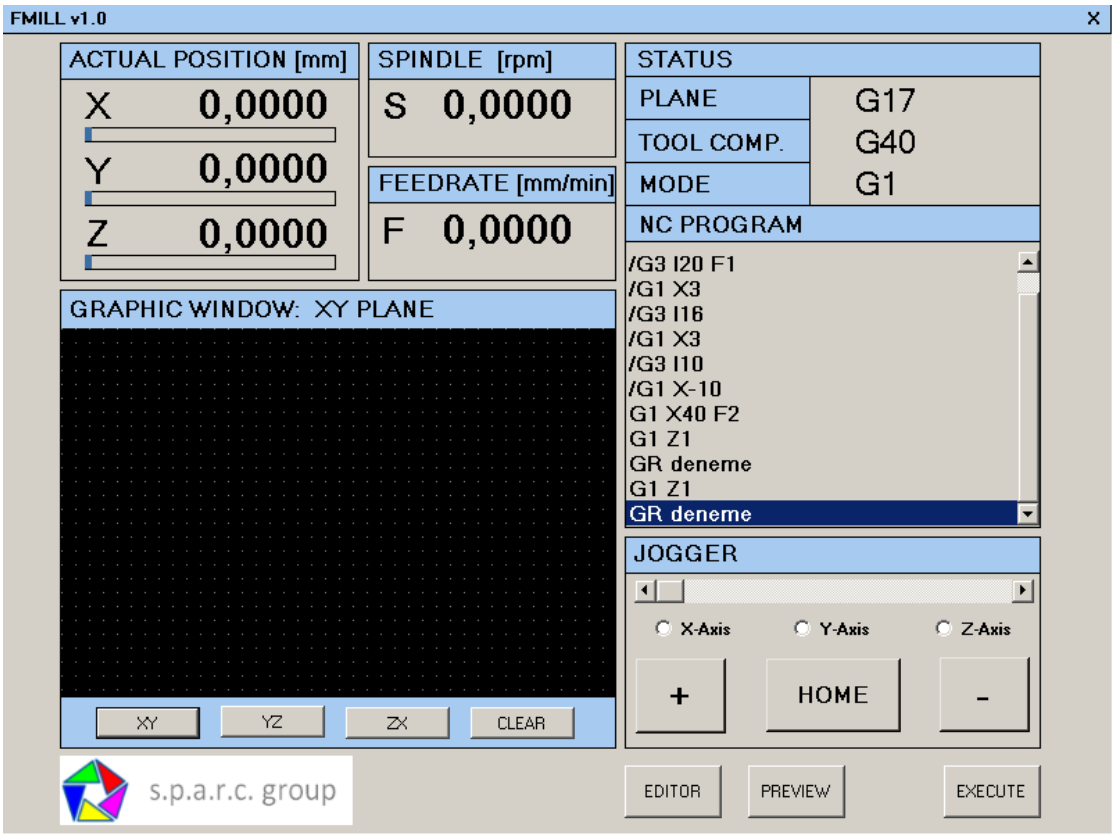


Figure 6.2 FMILL graphical user interface

In status window current working plane, tool compensation mode and motion mode are displayed. When a tool path simulation is needed pressing preview button will provide tool path trajectory on the graphic window. Executed can be started by pressing the execute button. A block diagram representation of the GUI is illustrated in the Fig. 6.3.

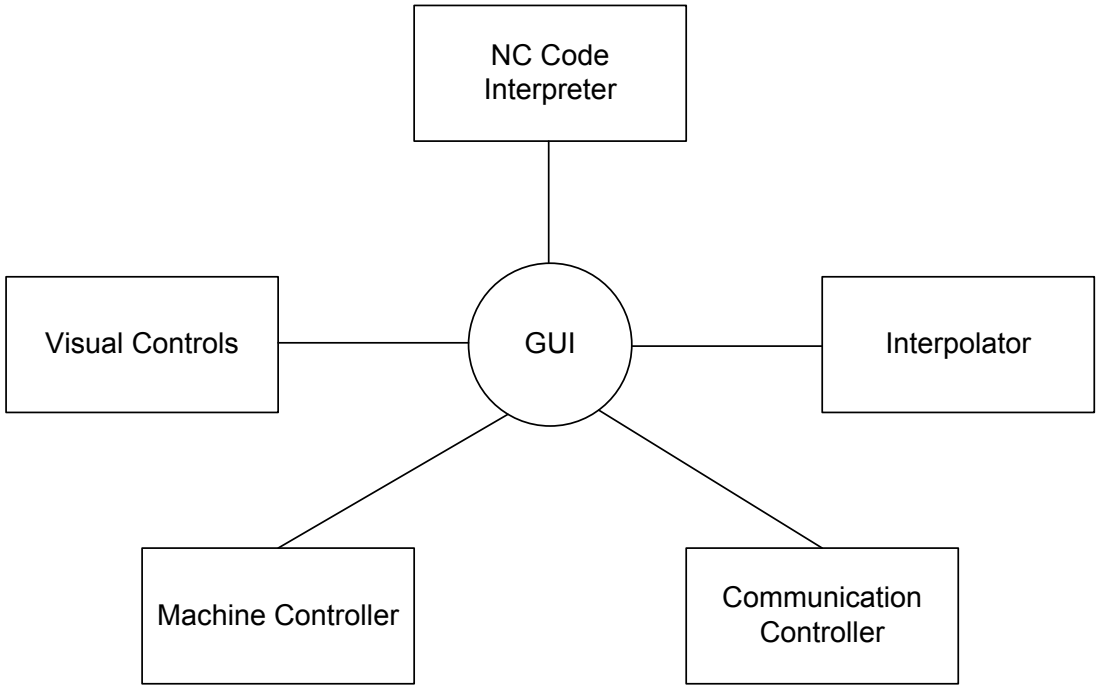


Figure 6.3 Block diagram representation of the GUI

6.3 Machine .dll

DLL stands for dynamic-link library. Main idea is that loading the subroutines into an application program at runtime rather than linking in the compile time with placing them separate locations in the hard drive. Therefore only a minimum amount of work is done by the compiler.

A machine DLL is developed for software utilizations of the distributed controllers. Mainly it includes some directives and handles communication protocol.

- **Constructor:**

  `SPARCMILL_DCS` can be included any file with the serial port name and initialized communication speed.

- **PrepareData:**

  `PrepareData` returns a byte array that includes encoded reference position commands. Input arguments are delta displacement by means of millimeters that data type is double and feed rate, desired velocity of the given displacement in the previous argument by means of meter per minute that data type is double.

- **AddtoQueue:**

  `addtoqueue` is utilized for constructing a byte array for each axis according to given incremental displacements. Input arguments are incremental displacements by means of millimeters for each axis, dx, dy, and dz. Also tangential desired speed by means of meters per minute is an argument too. All the arguments for these functions are double.

- **SendData***:*

  `SendData` is subjected to send 1500 of bytes of constructed data array. Its arguments are ID of the destination distributed control unit by means of bytes, ID of the master distributed control unit by means of bytes and byte array which includes encoded reference position commands data.

- **Execute:**

  This function uses global variables that are constructed by **`addtoqueue`** function. Also it controls the communication protocol as well as the software synchronization.

- **Jog:**

  `Jog` function employs unit ID and feed rate by means of bytes as arguments. Basically, this function invokes the jog operation in the controller firmware which described in the previous chapters.

- **JogStop:**

  This function stops the jog. It raises the **`jogstop`** interrupt in the controller firmware. It should be used after jog function is executed. It returns nothing.

This control class does not include additional machine control classes. Hence one who tries to develop software for those distributed controller, should develop an interpolation class.

6.4 NC Command Interpreter

Namely, command line interpreters is a kind of computer program that reads lines of text entered by user and interprets them in the predefined context. NC command interpreter, which can interpret standard NC commands, is developed during this thesis work. By using the editor window, the user can load or write an NC part program. Basic motion commands, which can be utilized by the interpreter, are summarized in the Table 6.1. In the FMILL software program, a user control is dedicated to interpret NC part program. NC command that can be interpreted by the FMILL is presented in the Table 6.1.

6.5 Reference-Pulse Interpolator

The interpolation for CNC machine control systems is defined as, construction of data points between given path formation and given two point definitions in two dimensional space. As far as parameters of interpolation are defined in the NC part program, interpreted command parameters are interpolated by interpolation subroutines.

A reference-pulse interpolator [58, 59] is developed for interpolating high level motion commands which are interpreted by the NC command interpreter. As it is cited in third chapter synchronization is a vital point for contouring the given trajectory in NC part program and distributed controllers are not internally connected each other to synchronize themselves [61]. Therefore, reference pulse interpolator not only generates position commands but also guaranties the reference commands for a specified time interval and arranges them for all axes. In other words interpolator determines number of BLUs for each axis at predefined time interval.

Table 6.1 Interpreted NC commands

G0	Rapid Motion
G1	Linear Motion
G2	CW Circular Motion
G3	CCW Circular Motion
G17	X-Y Plane selection
G18	Y-Z Plane Selection
G19	Z-X Plane Selection
G20	Programming in inches
G21	Programming in mm
G40	Tool radius compensation off
G41	Tool radius compensation left
G42	Tool radius compensation right
G90	Absolute programming
G91	Incremental programming
GR	Execute subroutine

6.5.1 Rapid Motion

This motion mode, generally, uses the maximum axial speed and employs for positioning of the cutting tool. It is not recommended for cutting operations. When the command is invoked, controlled actuator moves its maximum axial speed to the specified displacement as it is presented in the Fig. 6.4.

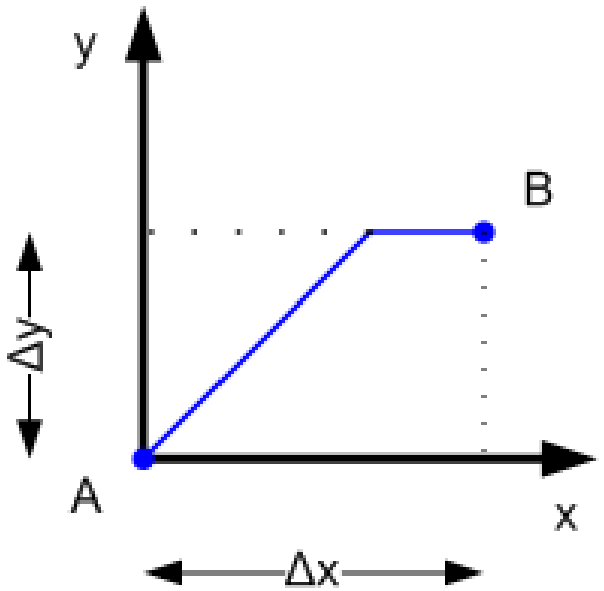


Figure 6.4 Rapid move

Hence there is no need to precise synchronization. Therefore each axis is commanded to complete the given linear displacement in the NC part program with its maximum available speed.

6.5.2 Linear Motion

This motion mode is facilitated to command rectilinear line movements which are illustrated in the Fig. 6.5 in general. Linear interpolation basically used for cutting operations. It can keep the linear path as close as possible. Unlike the rapid motion, resultant speed of axes is given as a linear motion parameter and called as linear motion feed rate. Hence, precise synchronization of each axis is obligatory. In order to synchronize axes, interpolator generates relevant numbers by means of BLU.

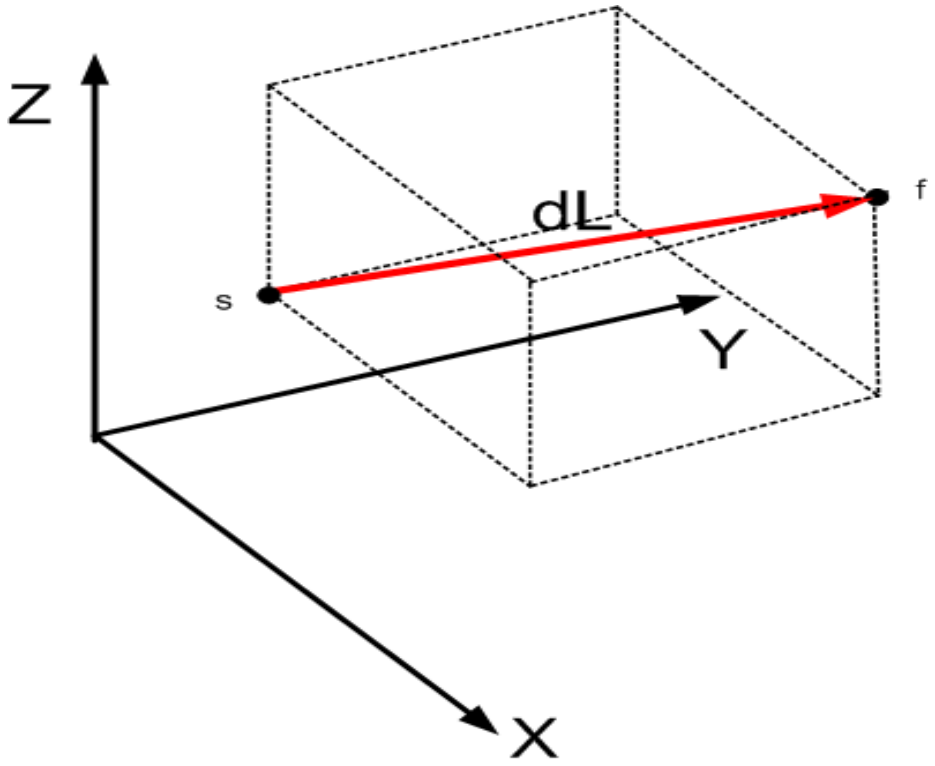


Figure 6.5 Linear move

The linear path which is given in Fig. 6.5, needs to be passed trough in T_t minutes where feed is defined in the NC part program by means of [m/ min]:

$$T_t = \frac{\sqrt{\Delta x^2 + \Delta y^2}}{feed} \tag{6.1}$$

If predefined time interval defined as T, number of reference commands can be calculated as

$$N = int\left\{\frac{T_t}{T}\right\} \tag{6.2}$$

This number will be executed in the predefined time interval and after each execution is completed a new set of numbers are created with considering the given resultant feed rate.

Finally linear interpolation equation turns into

$$x(k) = x(k-1) + \Delta x \tag{6.3}$$

$$y(k) = y(k-1) + \Delta y \tag{6.4}$$

Following section discusses circular interpolation and its features.

6.5.3 Circular Motion

In this motion mode, controller generates a circular path which is defined in NC part program as illustrated in the Fig.6.6. First, given trajectory is realized and then trajectory is divided into linear segments with start and ending points.

Additionally, circular motion is more complicated than a rectilinear one. Using brute computational techniques, it takes a relatively long time to calculate since calculation involves trig functions (like sine and cosine). Hence, lookup tables or approximation functions can be employed instead of brute computation algorithms. Circular path definitions are not the only non-linear path definitions. In order to sculpture such complex surfaces, interpolation algorithms have been evolved. Parametric curves [70, 72, 77], NURBS [63, 75] are the most common techniques.

Relation between the length of a circular path in Fig. 6.6 and central (sweep) angle has a following relation [70]:

$$|AB| = \Delta\theta_t \cdot R \tag{6.5}$$

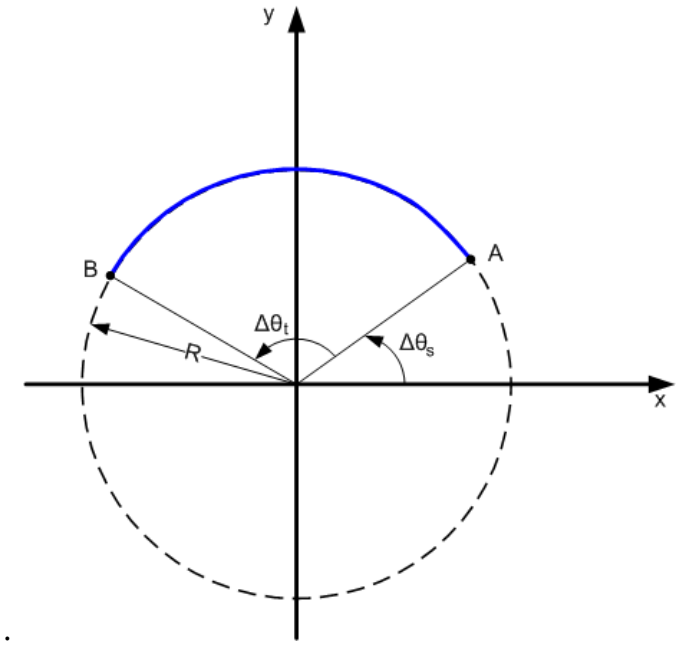


Figure 6.6 Circular move

where $\Delta\theta_t$ is defined as central angle and R is the radius. Assuming the center of the circular arc is located in the origin, the circular path can be described as

$$X = R \cdot cos\theta \tag{6.6}$$

$$Y = R \,.\, sin\theta \tag{6.7}$$

Using the parametric equation of the circle, next point to be interpolated is represented as

$$\begin{bmatrix} X(i+1) \\ Y(i+1) \end{bmatrix} = \begin{bmatrix} cos\Delta\theta_t & -sin\Delta\theta_t \\ sin\Delta\theta_t & cos\Delta\theta_t \end{bmatrix} \begin{bmatrix} X(i) \\ Y(i) \end{bmatrix} \tag{6.8}$$

If $\Delta\theta_t$ angle is divided into n many $\Delta\theta$'s

$$sin\Delta\theta \cong \Delta\theta \tag{6.9}$$

$$cos\Delta\theta \cong 1 \tag{6.10}$$

Then Eqn. (6.6) yields into:

$$X(i+1) = X(i) - Y(i) \cdot \Delta\theta_i \tag{6.11}$$

$$Y(i+1) = X(i) \cdot \Delta\theta_i + Y(i) \tag{6.12}$$

As a result of Eqn. (6.7), instead of higher order polynomials, circular interpolation algorithm boils down to first order polynomials which only include addition, subtraction and multiplication. Using previous command history, computational time can be decreased.

6.6 Verification of the Interpolator

As mentioned in the previous chapter, the interpolation is carried out by the FMILL software. Controller performance is discussed in the previous chapters. For controller software, it is expected to generate relevant points by employing interpolation algorithms. Proper generation of reference motion commands is not the only challenge. Reference commands are first generated as floating point numbers. Note that the path has to be expressed in term of machine BLUs which in turn leads to integer representation of the tool path. Reference commands, which are smaller than BLU, will not be executed and interpreted as zero. But, their summation, which may be larger than a BLU, will lead to a positioning error. In order to identify such systematic errors, a complementary function must be written. It can produce a ***csv*** file that includes encoded commands. Then, these commands are to be interpreted by distributed controllers. Decoding of such commands, gives positional increments. Integrating incremental command eventually yields the tool path. Note that the generated data array is to be compared with theoretical data. Fig. 6.7 shows

the difference between generated reference commands and the theoretical one for this tool path illustrated in Fig 6.8.

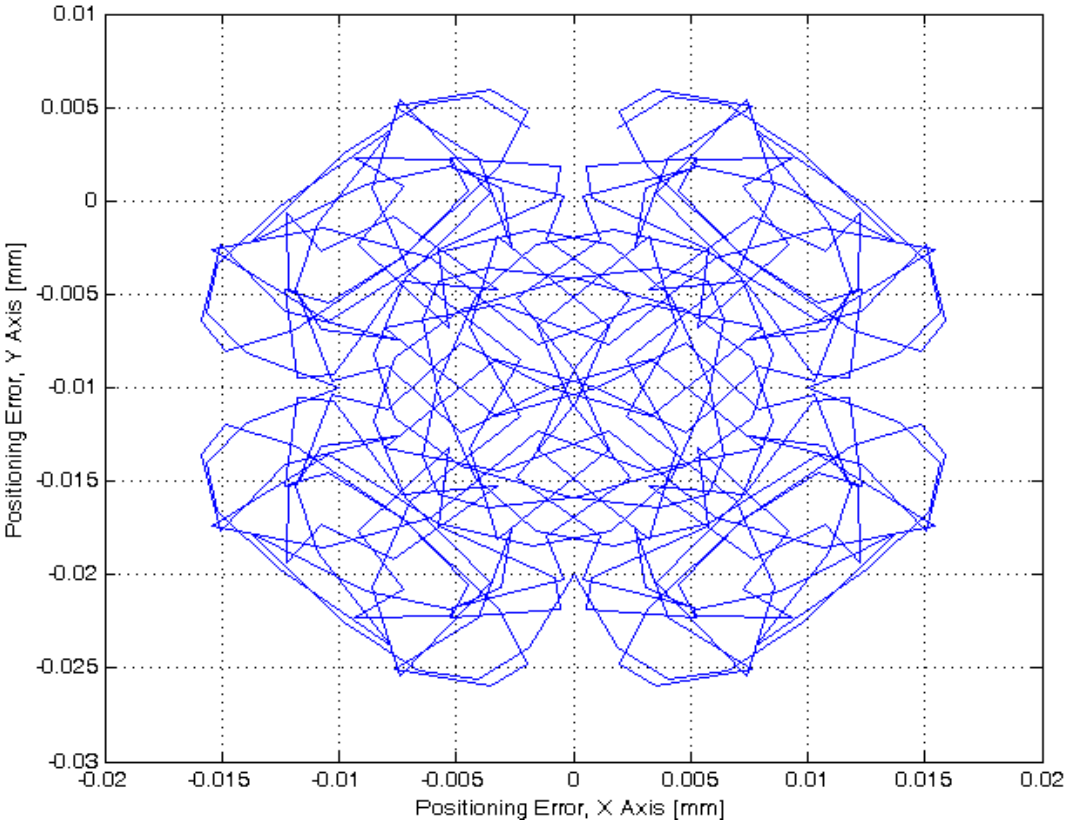


Figure 6.7 Command generation error

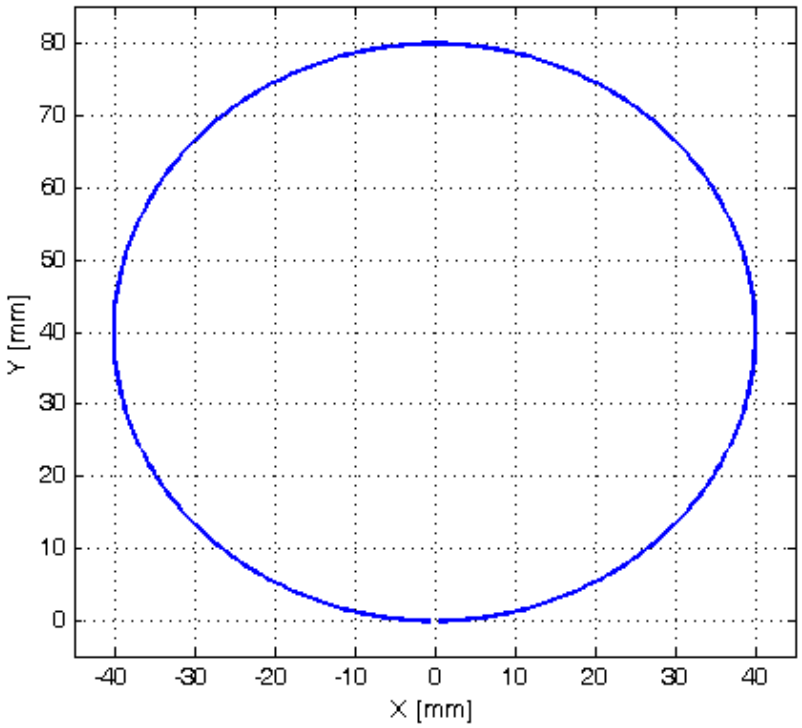


Figure 6.8 Generated tool path

6.7 Summary

As presented in the previous sections, computer software is developed for distributed control system designed in this work. Also applicable machine dynamic link library (DLL) is developed and its source code is listed in the Appendix D. CNC machine can be used more efficiently with better user interfaces. Many of the advanced operation attributes, which are beyond the scope of the thesis, can be added.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

Purpose of this thesis was the development and implementation of distributed control system for CNC machine tools. Different structures have been implemented that are elaborated in Chapter 4 and Appendix A.

Due to their networked architecture, the implementation of distributed control systems exhibits both computational- and communication delays and these delays can decrease the performance of the control system. Modeling of such networking and computational delays is sometimes necessary. Basically, computational delays can easily be modeled and tested. Networking speed of control system is selected by taking into account such delays.

Distributed control systems for motion control applications have also a synchronization problem. Depending on the application, the synchronization is essential. Especially, multi-axis movements do require synchronized implementation. If networking delays are modeled, one can synchronize the control system by properly manipulating the command generated. In that case, control system turns into a real-time control system. This problem can be identified by sending out repeated data patterns in the communication network. Command generation strategies for real-time systems are also studied in this work.

Control system network topology presented in Chapter 4. Master node of this control system can be considered as PC and the corresponding control software. Unfortunately, a real-time control system cannot be driven by a PC directly. Interrupts must be invoked by the slave nodes. Requesting or giving feedback information is needed. Therefore, amount of data, which need to be transferred along the communication network, will be increased. This situation might be solved by using faster communication interfaces.

Developed distributed control system which includes hardware, firmware and software was implemented on an open-loop controlled (3-axis) CNC machining center. Machining performance under various conditions was investigated. Different physical communication architectures and communication protocols were developed and presented in the previous sections. Different workpiece definitions were sculptured. Overall performance of the control system was found promising.

Furthermore a developed graphical user interface forced the limits of the distributed controllers. First 2D interpolation ability was tested. Then distributed controllers were forced to exhibit their 3D interpolation skills.

To sum up; flexible, modular, economic, retrofitable, distributed CNC control unit is developed and implemented. Results are obtained for an open-loop control system.

7.2 Future Work

A promising distributed hardware allowing the use of many peripheral units (brushless DC, analog sensors, optical position encoders, etc.) were developed within the scope of this thesis. Furthermore, a communication protocol, which is an essential part of this hardware was devised. Only open loop performance of the system was studied utilizing stepper motors without any position sensors. Since the

developed controller card do possess a powerful digital signal controller (16-bit DSP engine, integrated quadrature encoder interface, 10 MSample/s A/D converter, motor controller interface, etc.) closed-loop performance of the controllers could be investigated by simply connecting advanced motor drivers (servo-motor drivers, closed-loop stepper motors drivers) to this system without any augmentation on the system. A new communication protocol, which encapsulates large number of CNC machine features, can be developed as well. The devised hardware stores all the commands (issued by the host PC) in a buffer inside the digital signal controller (2kWords ~ 1800 position commands per axis). When the information in the buffer is exhausted, the host PC needs to fill this buffer again with new reference positions. Hence, a slight interruption in the machining process is inevitable in this scheme due to the transfer of this data from the host PC. To overcome this problem, one can incorporate a cyclic buffer to perform machining operation without any interruption. This is especially important for high-speed machining where slight interruption in the process might lead to undesirable effects on the surface of the workpiece.

The control system was implemented on a (3-axis) CNC machining center which was designed for 2½D interpolation[1]. Despite the fact that the Z-axis of this machine is properly selected to carry all the loads associated with this axis, the Z-axis motor driver, which can deliver only 2 A of continuous current, would not be able to drive this motor adequately. As a consequence, the Z – axis cannot follow the issued commands. Therefore, a powerful motor driver (like Parker-Hannifin SD5) can be utilized to rectify this situation. Note that in this thesis, the acceleration and

[1] In such an interpolation scheme, the interpolation is performed in the XY plane while Z axis is stationary

deceleration profile of all axes are controlled by the motor-driver itself in a hardwired fashion[2]. Thus, it is advisable to regulate these profiles inside the position control loop.

Since the spindle motor is of low power (160 W), the spindle motor can be swapped with a more powerful so that one can machine high-strength metals that definitely draw more power (> 500W). Otherwise, relevant cutting parameters (e.g. feed-rate, radial- and axial depth of cut) have to be restricted which in turn seriously hinders the cutting potential of the CNC machine tool.

Finally, the ATC of the machine was not fixed within the scope of this work due to time restrictions imposed. As a future work, it is recommended that the pneumatic tool holder actuator be replaced with a linear electric motor (i.e. an electric motor coupled to a lead-screw shaft). Hence, the resulting machine will not require any external compressed air sources.

Even though, the designed graphical user interface does have adequate control feature to operate the machine, it can be definitely enhanced by including new features including 3D machining simulation, advanced NC-code parser (with "Intellisense" capability), improved manual data input (MDI) features, tool radius/length offset compensation, tool database management, etc. Furthermore, since the technological trend in CNC technology is towards the use of touch-screen LCD, one can develop a GUI which can support such devices.

[2] A capacitor (C25) on the PCB of the driver (PH-SD2) must be replaced with an adequate one. Usually, smaller capacitances leads to increased acceleration/deceleration profiles.

APPENDIX – A

Developed Hardware Architectures

During this thesis work several distributed control architectures had been tried so far. Each of the control system was added into an RS232 based point-to-point multi drop communication network.

A.1. First Architecture

First architecture can be regarded as a preliminary work to understand distributed control systems and is the first attempt to realize the concept in a small scale. It was a complete control system with own motor driver which was influenced by [17]. It is not functionally distributed by means of control components [1]. But a hierarchical architecture [18, 25] was used in order to reduce the complexity (Fig. A.1). It is a networked control system [48] that constitutes multi drop RS232 communication interface [51] (Fig. A.2).

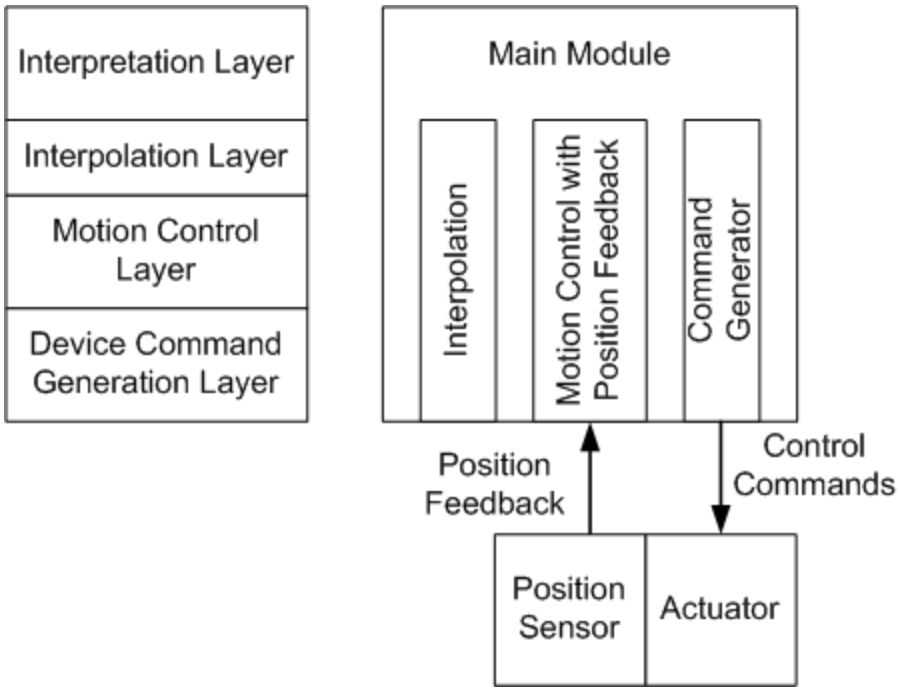


Figure A.1 First architecture

A program on a PC, which can be considered as the host of the network, interprets and pre-processes NC part programs. This software has an ability to generate data streams which include reference position commands and can be accepted as another node on the control system. When data is sent over the communication network, each controller reads the whole data. It is a message based protocol which was influenced by CAN [35, 41, and 52]. When the stream was read, each of the identical controllers do the followings:

- interprets pre-processed NC commands,
- interpolates the reference commands,
- generates clock signals for its stepper motor driver.

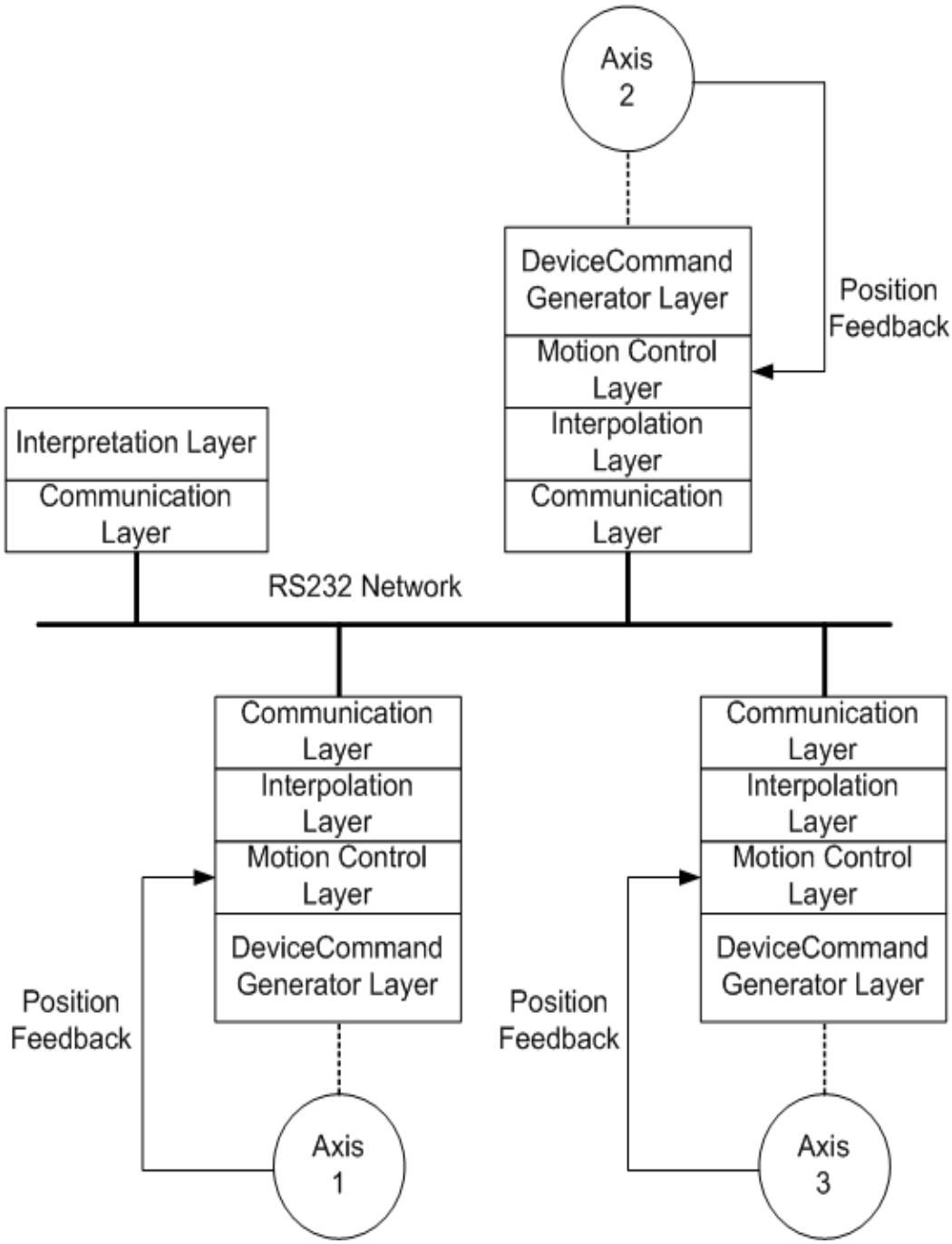


Figure A.2 DCS over RS232 network, first architecture

Command generator, interpolator, can be classified as reference-pulse interpolator which was influenced by [64, 69]. Each of the points was generated by means of BLU of the controlled system and sent through the actuator interfaces.

Main advantage of this architecture is its compact structure which includes controller and motor driver (Figs. A.3 and A.4).

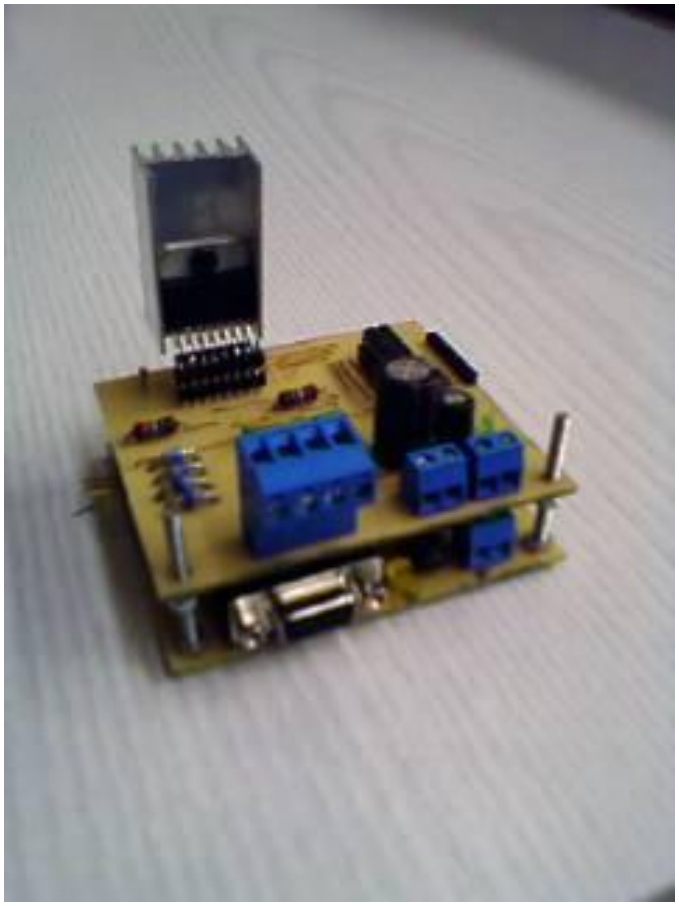


Figure A. 3 Photo of the first architecture

Hardware of the control system, which is illustrated in the Fig. A.4, can be divided into two parts: controller and actuator interface. It has one microprocessor and one RS232 interface on the controller side. Also L297-L298 pair that accepts clock signals up to 2 kHz to drive stepper motors.

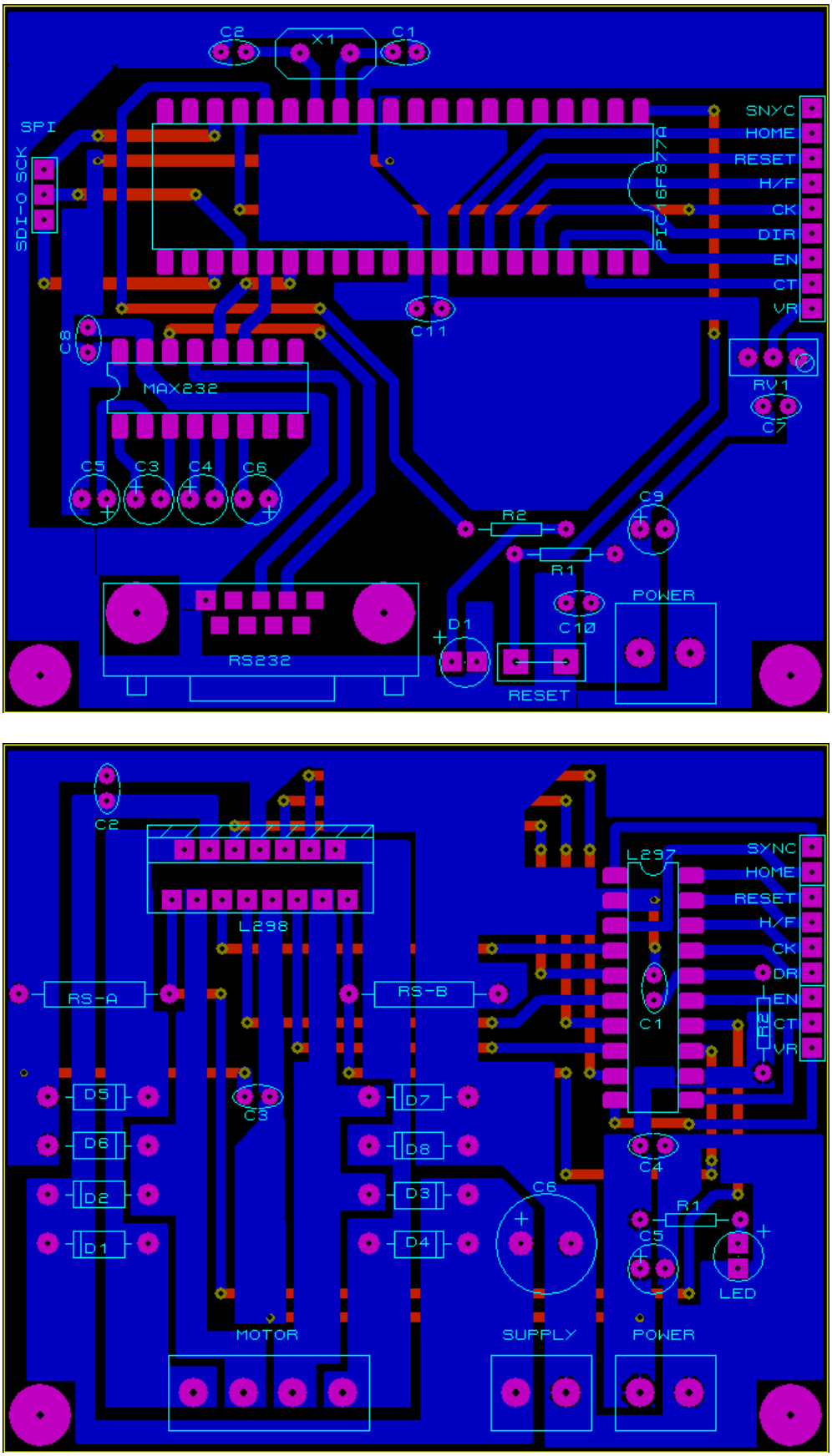


Figure A.4 First architecture, hardware PCB

This networked control system has a breakpoint (unsafe). All the controllers are needed to be synchronized by a synchronization engine [42]. In order to synchronize distributed controllers which are identical, they run same algorithms with the same data structure. An experiment was conducted during this thesis which was related to synchronization without any interconnection or interruption. Using the same function block with the internal timers of PIC microcontrollers, execution time was calculated and a tiny deviation, which could be eliminated, was observed.

After all, using only hierarchy in the firmware introduces huge calculation delays [2]. These delays can be compensated by faster hardware solutions or by parallel programming of such micro-controllers. Instead of using complex software, functionally distributed control architecture (with fast and capable micro-controllers) can reduce the computational time.

A.2. Second Architecture

Second architecture can be considered as a milestone of this work. It is based on the functionally distributed control architecture [1]. Based on the idea, two additional micro controllers added up on the controller hardware. Three of the micro controllers are connected to each other internally via serial peripheral interface (SPI) communication interface. One of the micro processors is responsible for internal communication which is also the communication interface. 16-bit micro controllers are used instead of 8-bit micro controllers in this architecture. One of the important advantages of new micro controllers is that they constitute different peripheral

interfaces like three UART interfaces, encoder interface. A hierarchical architecture is used as a tradeoff for complexity (Fig. A.5).

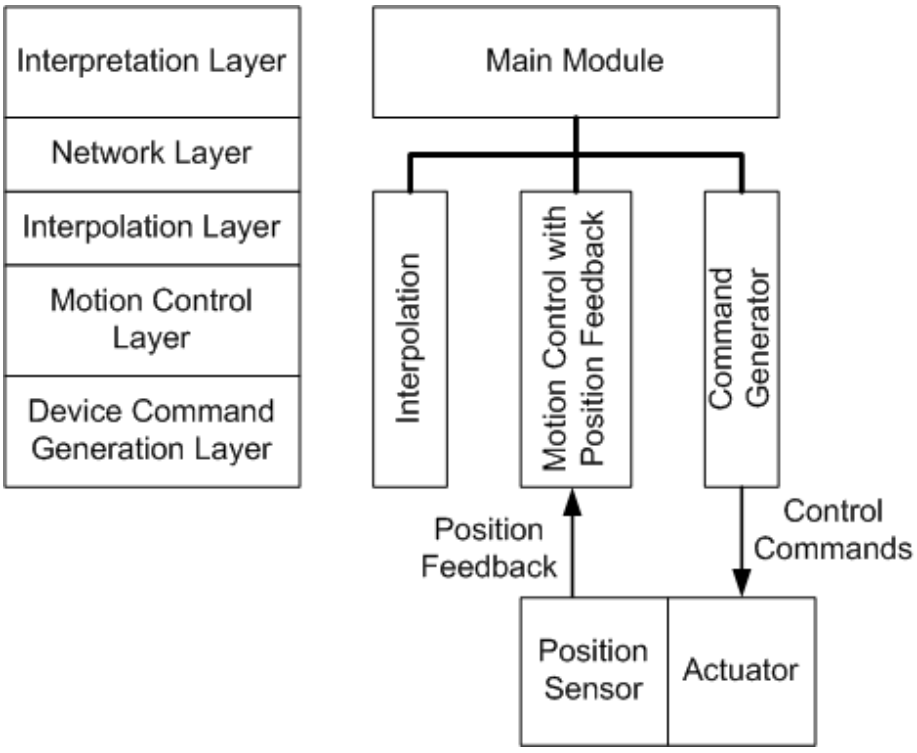


Figure A.5 Second architecture

Like the first architecture, a computer program with an NC code interpreter masters the communication network and starts the communication stream on the RS232 network (Fig. A.6). When the stream is received from the communication layer, using the internal communication network, reference commands are passed between communication- and the interpolation layer. Then, the interpolated reference commands are passed to motion control layer. This layer is also responsible for measurement of axis position. Interpolation and motion control are connected to each other over a hierarchy. Motion controller and interpolator are handled by one micro computer. Device command generator layer takes the relevant instruction in

order to generate driver signals which activate the motor within the motor driver's capability.

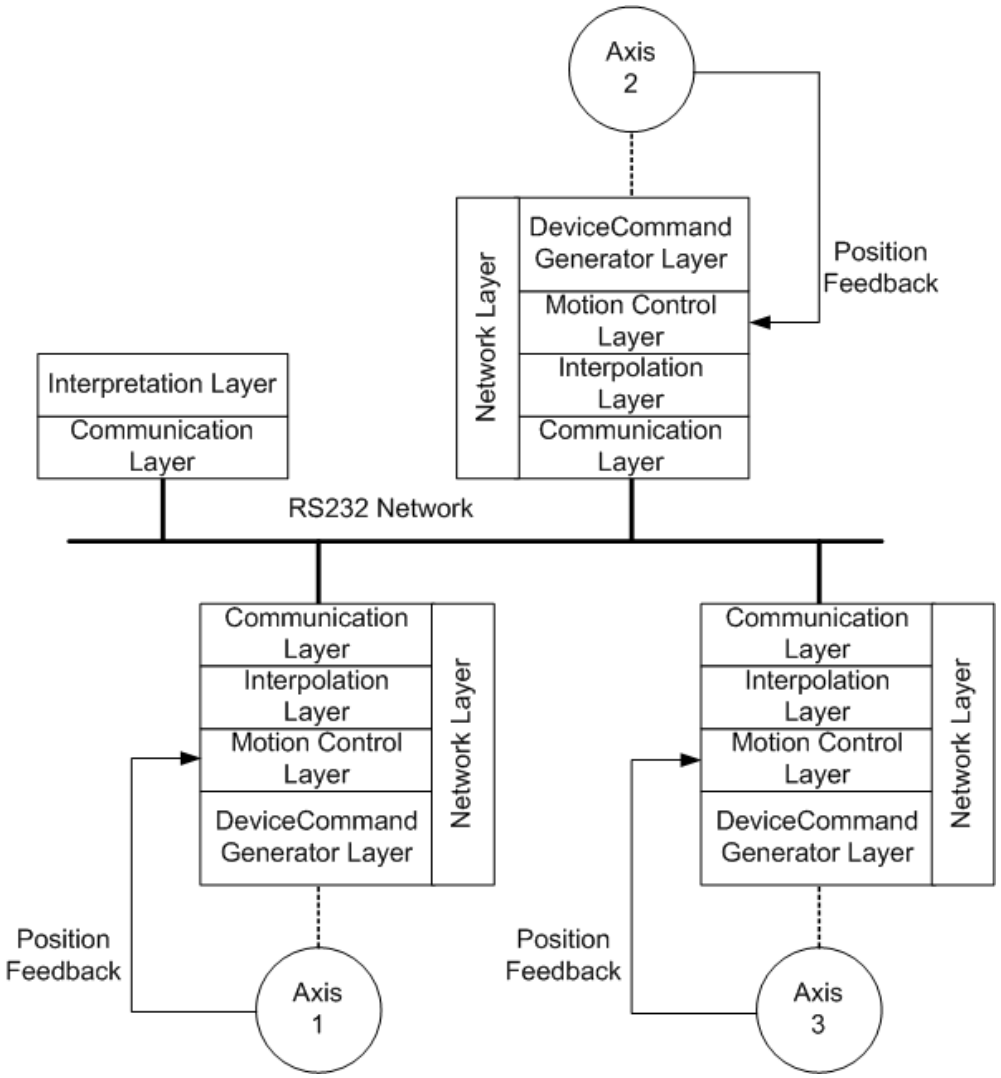


Figure A.6 DCS over RS232 network, second architecture

Interpolator can be categorized as a reference-pulse interpolator. It uses line segments, no matter how complex trajectory is. Such an algorithm is easy to implement. But it takes more of the instruction time of the motion controllers. Also, this computational load arises a vital point about this type of controller which disrupts the synchronization of the whole system. Synchronization is necessary to

tracing a two or three dimensional path with the given feed rate as a parameter. Here the problem is directly related to axial position displacements within a distinct time period. Main contribution of the second controller is its functionally distributed architecture (Fig. A.7).

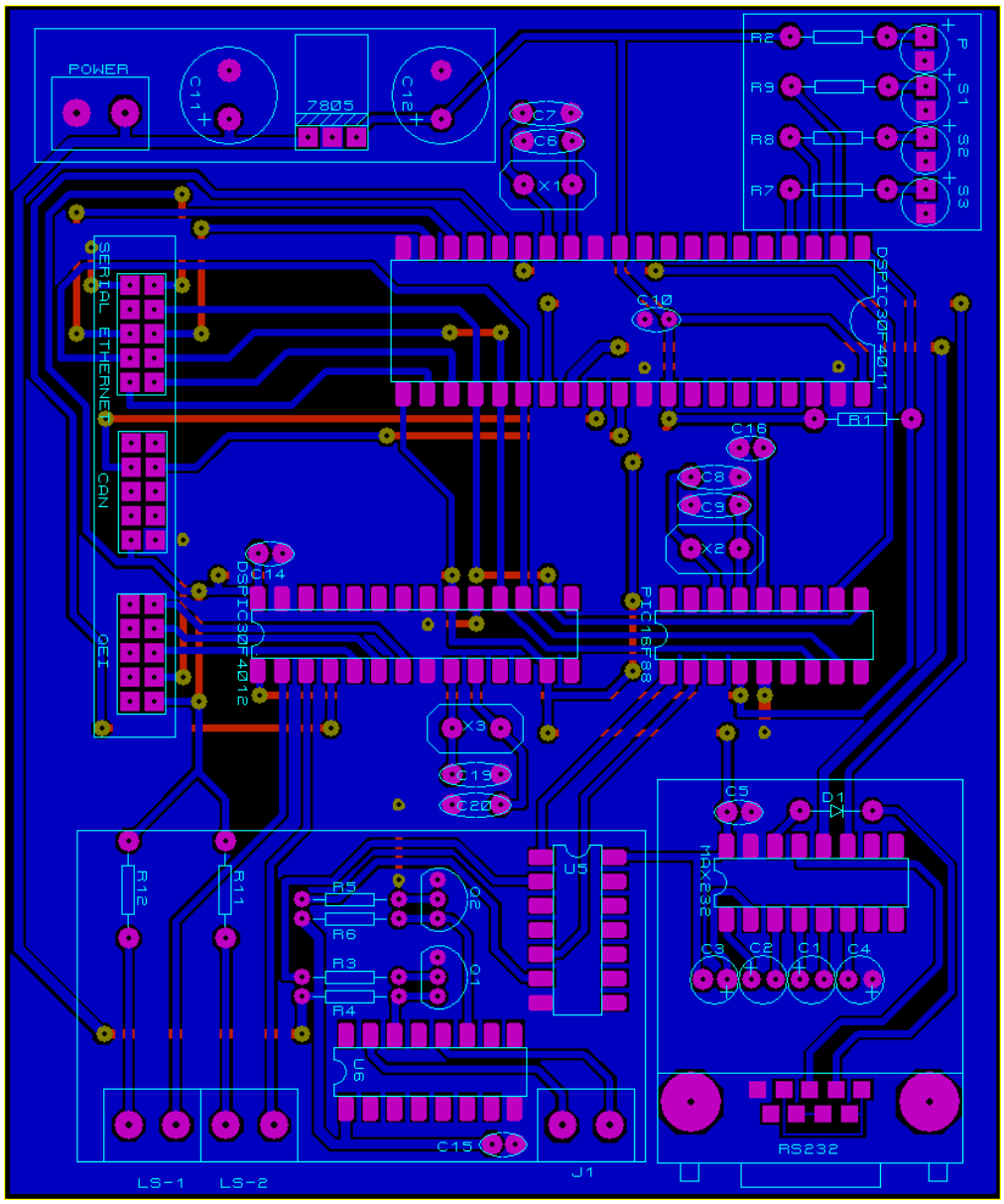


Figure A.7 Second architecture, hardware PCB

Unfortunately, this architecture does not have any synchronization interface. Duration of any interpolation instruction may not be identical. Therefore, there

exists a tradeoff between adding up an instruction synchronization engine and sending out interpolated reference commands (driven communication host).

The first option introduces an additional complexity to inter-communication protocol. Therefore, interpolating the reference commands using NC command interpreter with computer software needs only additional interpolator sub program (in the computer software). Also, rearranging the byte stream makes a new architecture possible.

A.3. Third Architecture

Third architecture, whose block diagram is represented in the Fig. A.8, can be classified as functionally distributed hierarchical distributed control system. This control system behaves like a PC driven real-time control network. It has two micro processors: one for communication and measurement, the other for command generation for the motor driver. It has very strict hierarchical networking between functional modules. They are connected to each other with parallel connection of five general purpose I/O pins. Additionally, computer software hosts the communication. It can interpret NC commands using its NC command interpreter algorithm. Then it can interpolate the interpreted NC commands using reference-pulse interpolation algorithm and controls the communication network between the distributed controllers. First, each line segment is created on the PC then the relevant informant is put onto the network.

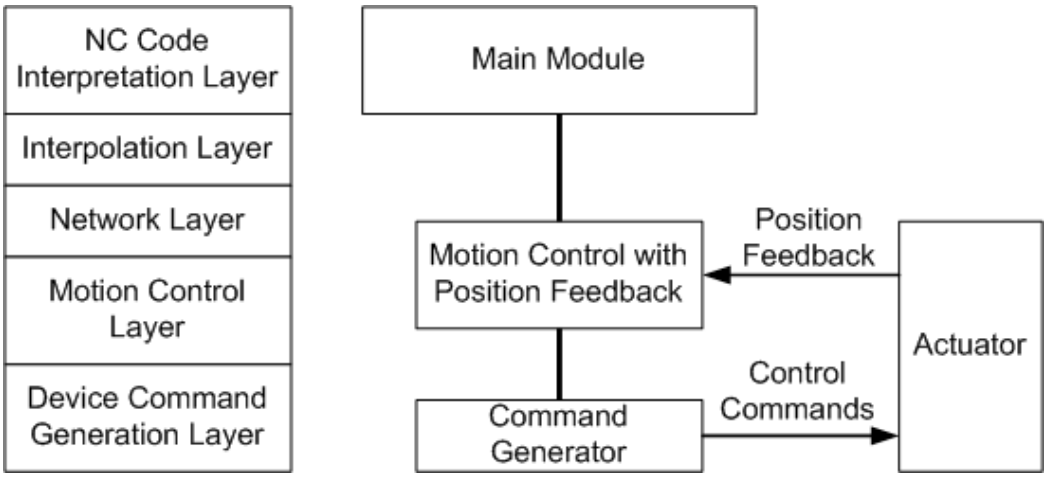


Figure A.8 Third architecture

A software program on the PC can also be considered as a host of the communication network. It has NC part program interpretation, reference command interpolation which is a reference-pulse interpolator and communication module. At this point, the software has object oriented structure. When data stream is released through the communication network, it inherits the properties of the message based and address based communication protocols. Line segments transferred as two byte data. Addition information is visualized in the Figs. A.9, A.10 and A.11. For further information on communication protocol, reader is encouraged to refer to Chapter 3.

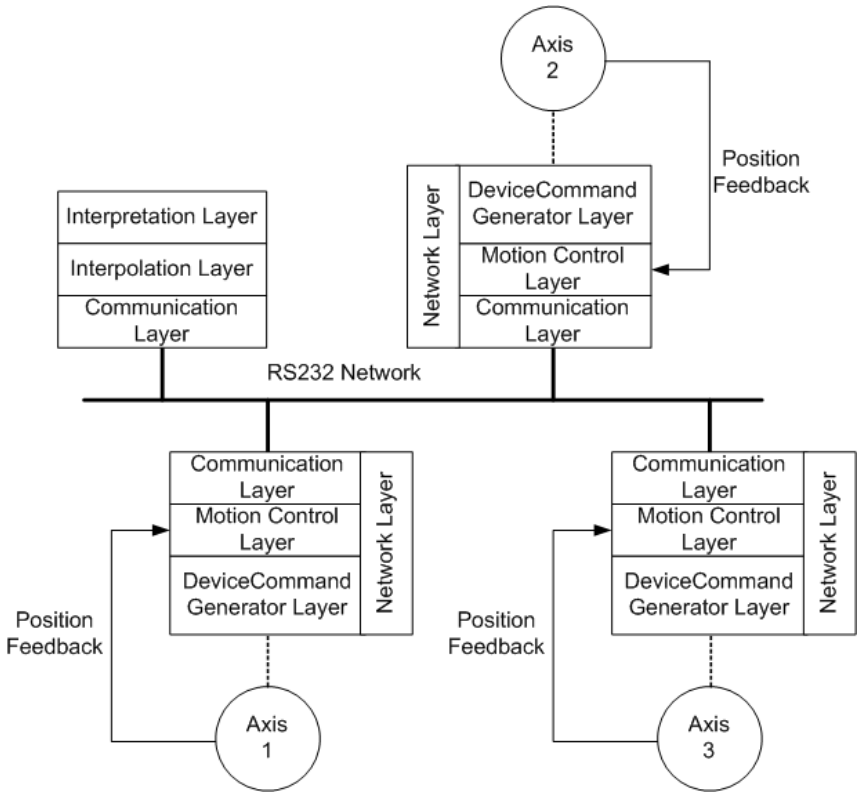


Figure A.9 DCS over RS232 network, third architecture

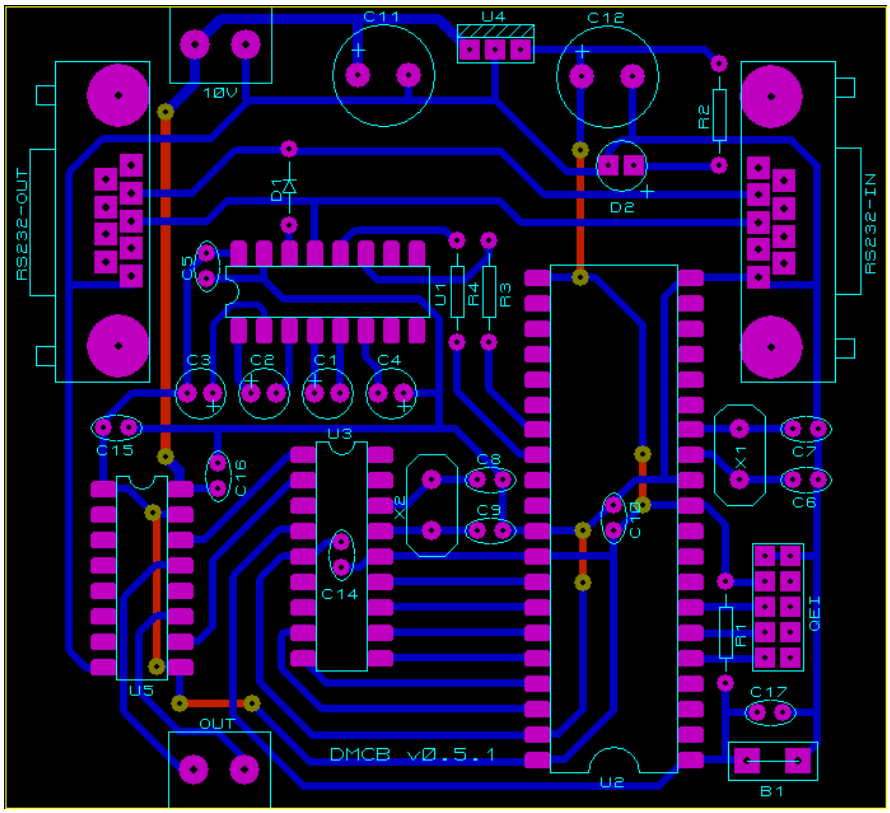


Figure A.10 Third architecture, hardware PCB

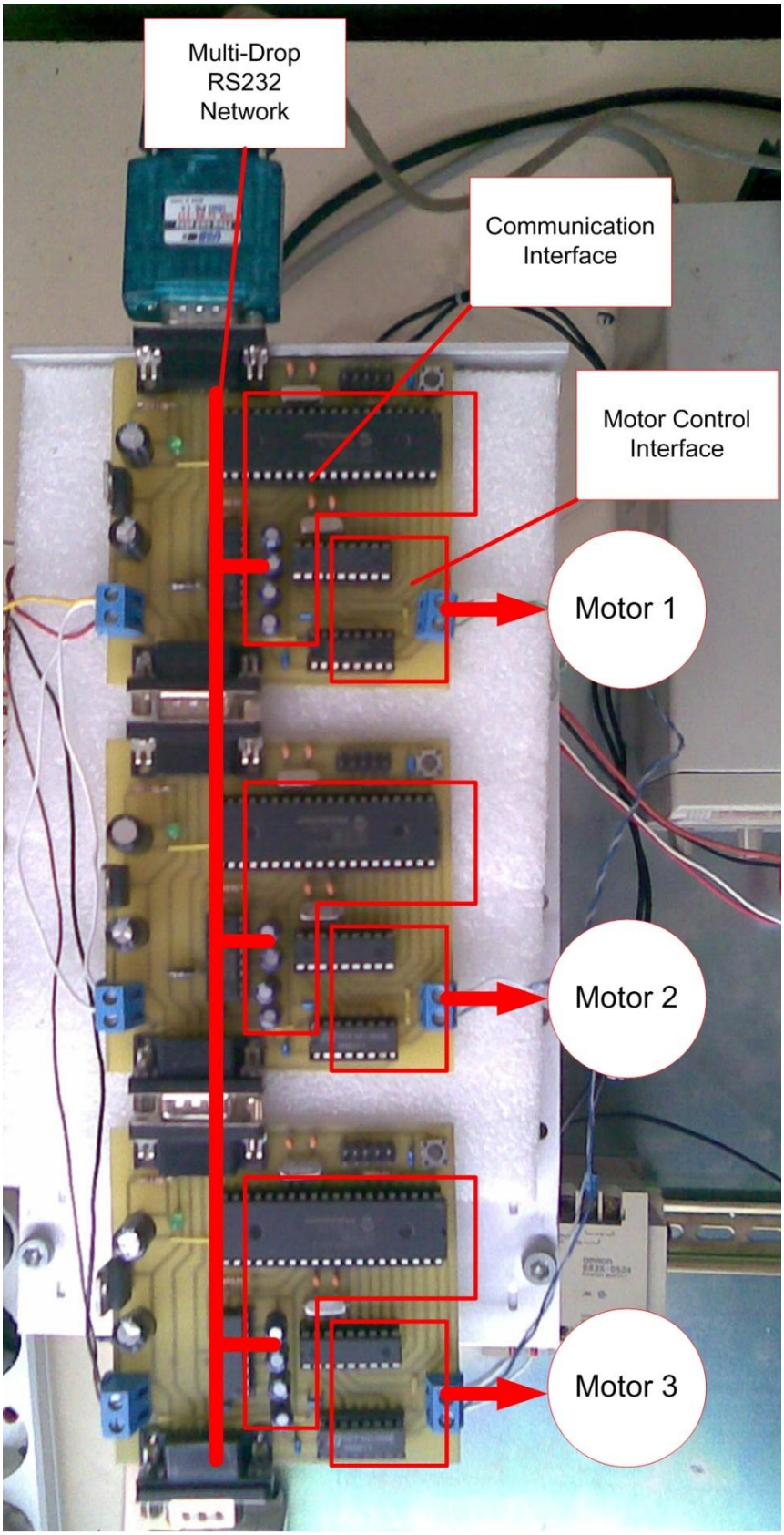


Figure A.11 Application of the third architecture

APPENDIX – B

Firmware of dsPIC30F4011

The dsPIC30F4011 is 16-bit digital signal controller which was designed for motor control and industrial applications. It is diversified as high-performance RISC CPU, which has wide range of peripheral interfaces.

dsPIC30F4011 is used for communication interface. Generally, it incorporates the PIC16F88 for command generation and responsible for protocol implementation.

```
#include <30F4011.h>
#fuses HS
#fuses NOWDT,NOPROTECT,NOBROWNOUT
#use delay(clock=20000000))
#use rs232 (uart2, baud = 38400)
#use fast_io(E)
#use fast_io(D)
#bit DE = 0x02D4.2  // DE, RS485 TX Enable output: RD2
#bit SW1 = 0x02CE.14 // DCS Select #1 input: RC14
#bit SW2 = 0x02DA.8 // DCS Select #2 input: RE8
#bit SW3 = 0x02D4.1 // DCS Select #3 input: RD1
#bit SW4 = 0x02D4.3 // DCS Select #4 input: RD3
#bit SS = 0x02D4.0   // SS PIN
#bit LSW = 0x02E0.0  // LSW, limit switch connection: RF0
#bit RX = 0x02C8.6
#bit TX = 0x02C8.7
#bit SPI = 0x02C8.8
#bit ERR = 0x02CE.13
unsigned int8 r[1500];  // buffer
unsigned int8 d;
unsigned int8 _ID;   // ID
// myID is used to calculate device ID
// it uses the SWx PINs and DCS select DIP switch on the PCB
unsigned int8 myID(){
   unsigned int8 ID, dummy;
   ID = 0;
   dummy = SW4;
   ID += (dummy<<3);
   dummy = SW3;
   ID += (dummy<<2);
```

```
   dummy = SW2;
   ID += (dummy<<1);
   ID += SW1;
   ID = 120 + ID;
   return ID;
}
// go is used to send position command to motor interface
// it uses SPI interface
void go(unsigned int8 CN){
   unsigned int8 dummy;
   SPI = 0; // idicate SPI transaction has started
   // put SPI data while emptying the receive buffer
   dummy = spi_read(CN);
   while (!spi_data_is_in()){
      // since spi data is not received wait for data
   }
   dummy = spi_read();  // read SPI data
   SPI = 1; // idicate SPI transaction has completed
}
void main() {
   unsigned int8 r1, r2;
   unsigned int16 k; // time index
   int1 flag = 0;
   int1 Imaster = 0;
   //
   // PIN configurations
   set_tris_b(0x0000);
   set_tris_c(0x4000);
   set_tris_d(10);
   set_tris_e(0x0100);
   set_tris_f(17);
   //
   // setup SPI
   // setup as : spi master, data transmitted on H-to-L
   // sys. clock divided to 16 and set as SCK, slave select disabled
   setup_spi(SPI_MASTER|SPI_H_TO_L|SPI_CLK_DIV_16|SPI_SS_DISABLED);
   //
   //read out unit ID
   _ID = myID();
   //
   // reset indicator LEDs
   ERR = 1;
   RX = 1;
   TX = 1;
   SPI = 1;
   //
   // reset buffer flag
   flag = 0;
   while(1){
      if(kbhit()){
         //
```

```
// if serial data received
RX = 0;
r1 = getc();
r2 = getc();
RX = 1;
//
// if ID matches and flag = 0
// then prepare for data transfer
if((r1 == _ID) & (flag == 0)) {
   //
   // for r2, if ID matches dedicated as master
   if(r2 == _ID) Imaster = 1; // now im the master
   //
   // else dedicated as servant
   else Imaster = 0; // now im the servant
   DE = 1;  // enable TX
   TX = 0;
   putc(_ID);  //device answers PC's call for data transfer
   delay_us(250); // wait for transaction
   DE = 0;  // disable TX
   TX = 1;
   k = 0;   // reset time index
   RX = 0;
   // fill device buffer
   for(k=0;k<1500;++k){
      r[k] = getc();
   }
   RX = 1;
   DE = 1;  // enable TX
   TX = 0;
   putc(_ID);  // device invokes the PC: transfer completed
   delay_us(250);
   DE = 0;
   TX = 1;
   flag = 1;   // raise the buffer flag
   ERR = 0;
}
//
// if r1 = 'h' and ID matches go home position
else if ((r1 == 0x68) & (r2 == _ID)){   //h
   ERR = 0;
   //
   // go while Limit switch is detected
   while (!LSW) go(208);
   DE = 1;  // enable TX
   TX = 0;
   putc(_ID);  // device invokes the PC: home position
   delay_us(250);
   DE = 0;  // disable TX
   TX = 1;
   ERR = 1;
```

```
   }
   //
   // if r1 = 'j' and ID matches jog mode is ON
   else if((r1 == 0x6A) & (r2 == _ID )){   // j
      DE = 1;  // enable TX
      TX = 0;
      putc(_ID);  // device answers the PC's call
      delay_us(250);
      DE = 0;
      TX = 1;
      RX = 0;
      r1 = getc();   // r1 has to be device ID
      r2 = getc();   // get the feedrate data
      RX = 1;
      ERR = 0;
      //
      // while data received just go!
      while(!kbhit()){
         go(r2);
      }
      ERR = 1;
   }
   //
   // software synchronization, if r1 = 120, ID of the PC
   // buffer flag = 1 (buffer is full) execute buffer
   else if((r1 == 120) & (flag == 1)) {
      k = 0;
      for(k=0;k<1500;++k){
         r1 = getc();
         //SS = 0;
         go(r[k]);
         //SS = 1;
         // if device is deciated as master
         if(Imaster) {
            DE = 1;  // enable TX
            TX = 0;
            // it invokes the PC for next operation
            putc(_ID);
            delay_us(250);
            DE = 0;  // disable TX
            TX = 1;
         }
         // else just wait for master to finish transaction
         else delay_us(250);
      }
      flag = 0;   // reset the buffer flag
      ERR = 1;
   }
}
//
// else reset r1 and r2
```

```
        else {
            r1 = 0;
            r2 = 0;
        }
    }
```

APPENDIX – C

Firmware of PIC16F88

The PIC16F88 is a powerful CMOS flash based 8-bit micro controller. 16F88 is dedicated to perform command signal generating that relieves considerable processing load on the dsPIC30F4011.

```
#include <16f88.h>
#fuses HS
#fuses NOWDT,NOPROTECT
#use delay(clock=20000000))
#bit DIR = 0x05.4   // DIR PIN connected to RA4
#bit SS = 0x06.5  // Busy
//
// direction command embeded inside the received byte
// calculate and set direction
unsigned int8 clcC(unsigned int8 d1){
   DIR = (d1>>7) & 0x01;
   d1 = d1 & 0x7F;
   return d1;
}
//
// calculate counts which need to be execute in the relevant index
unsigned int16 clcD(unsigned int8 r){
   unsigned int16 D;
   // half of the relevant period has just been calculated
   if (r > 0) D = 5000 / r;
   else D = 0;
   return D;
}
//
// execute received command by SPI
void go(unsigned int8 c){
   unsigned int16 D;
   if (c>0){
      c = clcC(c); // subtract direction information from received
      D = clcD(c);  // calculate period of command signal
      //
      // execute!
      while(c>0){
```

```
            output_high(PIN_B3);
            delay_us(D);
            output_low(PIN_B3);
            delay_us(D);
            --c;
         }
      }
   }
void main(){
   unsigned int8 d = 0, c = 0;
   // PIN configurations
   set_tris_a(0x00);
   set_tris_b(0x00);
   //
   // setup SPI
   // setup as : spi slave, data transmitted on H-to-L
   // slave select disabled
   setup_spi(SPI_SLAVE | SPI_H_TO_L | SPI_SS_DISABLED);
   while(1) {
      if(spi_data_is_in()){
         // get data from SPI
         d = spi_read(c);  //read data from SPI while writing c
         go(d);
         // set 30F4011, operation completed
         d = spi_read(c);  //read data from SPI while writing c
      }
   }
}
```

APPENDIX – D

Machine DLL

```
using System;
using System.IO.Ports;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace SPARCMILL_DCS_v1
{
    class SPARCMILL_DCS
    {
        SerialPort PORT = new SerialPort();

        Queue<byte> X = new Queue<byte>();
        Queue<byte> Y = new Queue<byte>();
        Queue<byte> Z = new Queue<byte>();
        bool ind = true;
        /// <summary>
        /// Constructor
        /// </summary>
        /// <param name="PortName">Name of the COM port</param>
        /// <param name="BRate">COM Speed</param>
        public SPARCMILL_DCS(string PortName, int BRate)
        {
            PORT.PortName = PortName;
            PORT.BaudRate = BRate;
            PORT.Parity = Parity.None;
            PORT.DataBits = 8;
            PORT.StopBits = StopBits.One;
            PORT.Handshake = Handshake.None;
        }

        #region functions

        /// <summary>
        ///
        /// </summary>
        /// <param name="x1"> displacement by means of mm</param>
        /// <param name="Fx1">feed by means of m/min</param>
        /// <returns></returns>
        public byte[] prepareData(double deltA, double f1)
        {
            if (deltA != 0)
            {
                deltA = deltA * 100;  // mm -> cnt conversation

                f1 = Math.Round(Math.Abs(f1 * 1000 / 6));
                if (f1 > 127) f1 = 127;
                // m/min -> cnt/100ms conversation
```

```
                double k = Math.Abs(deltA / f1);
                double t = Math.Truncate(k);
                k -= t;
                double f0 = k * f1;

                if (deltA < 0)
                {
                    f1 += 128;
                    f0 += 128;
                }

                byte[] data = new byte[Convert.ToInt32(t + 1)];

                data[0] = Convert.ToByte(f0);

                for (int i = 1; i < t + 1; ++i)
                {
                    data[i] = Convert.ToByte(f1);
                }
                return data;
            }
            else
            {
                byte[] b = new byte[1] { 0 };
                return b;
            }
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="dx"></param>
        /// <param name="dy"></param>
        /// <param name="dz"></param>
        /// <param name="f"></param>
        public void addtoQueue(double dx, double dy, double dz,
double f)
        {
            double fx = 0, fy = 0, fz = 0, r = 0;

            r = Math.Sqrt(Math.Pow(dx, 2) + Math.Pow(dy, 2) +
Math.Pow(dz, 2));

            fx = dx / r * f;
            fy = dy / r * f;
            fz = dz / r * f;

            byte[] datax = prepareData(dx, fx);
            byte[] datay = prepareData(dy, fy);
            byte[] dataz = prepareData(dz, fz);

            if ((datax.Length >= datay.Length) & (datax.Length >=
dataz.Length))
            {
                foreach (byte xk in datax) X.Enqueue(xk);
                foreach (byte yk in datay) Y.Enqueue(yk);
                foreach (byte zk in dataz) Z.Enqueue(zk);

                int dl = datax.Length - datay.Length;
                for (int q = 0; q < dl; ++q) Y.Enqueue(0x00);
```

```
                dl = datax.Length - dataz.Length;
                for (int q = 0; q < dl; ++q) Z.Enqueue(0x00);

            }
            else if ((datay.Length >= datax.Length) & (datay.Length
>= dataz.Length))
            {
                foreach (byte yk in datay) Y.Enqueue(yk);
                foreach (byte zk in dataz) Z.Enqueue(zk);
                foreach (byte xk in datax) X.Enqueue(xk);

                int dl = datay.Length - dataz.Length;
                for (int q = 0; q < dl; ++q) Z.Enqueue(0x00);

                dl = datay.Length - datax.Length;
                for (int q = 0; q < dl; ++q) X.Enqueue(0x00);

            }
            else if ((dataz.Length >= datax.Length) & (dataz.Length
>= datay.Length))
            {
                foreach (byte zk in dataz) Z.Enqueue(zk);
                foreach (byte xk in datax) X.Enqueue(xk);
                foreach (byte yk in datay) Y.Enqueue(yk);

                int dl = dataz.Length - datax.Length;
                for (int q = 0; q < dl; ++q) X.Enqueue(0x00);

                dl = dataz.Length - datay.Length;
                for (int q = 0; q < dl; ++q) Y.Enqueue(0x00);

            }

        }


        /// <summary>
        ///
        /// </summary>
        /// <param name="ID1">UNIT ID</param>
        /// <param name="ID2">id ID2 == UNIT ID then UNIT is MASTER
else UNIT is SERVANT</param>
        /// <param name="data"> 1500 element data array</param>
        public void sendData(byte ID1, byte ID2, byte[] data)
        {
            byte[] d = new byte[2];
            byte[] dmy = new byte[1];

            d[0] = ID1;
            d[1] = ID2;

            PORT.Open();
            PORT.Write(d, 0, 2);     // set DCS UNIT
            PORT.Read(dmy, 0, 1);    // get DCS UNIT
            PORT.Write(data, 0, 1500);  // put DCS UNIT
            PORT.Read(dmy, 0, 1);    // get DCS UNIT, now DCS UNIT
raises its buffer flag
            PORT.Close();
        }

        /// <summary>
```

```
        /// takes 1500 of byte arrays of wach axis
        /// and executes them
        /// </summary>
        /// <param name="deltaX"> X data array</param>
        /// <param name="deltaY"> Y data array</param>
        /// <param name="deltaZ"> Z data Array</param>
        public void eXecute()//byte[] deltaX, byte[] deltaY, byte[]
deltaZ)
        {
            byte[] d = new byte[2];
            byte[] dmy = new byte[1];

            byte[] datX = new byte[1500];
            byte[] datY = new byte[1500];
            byte[] datZ = new byte[1500];

            int k = 0;
            byte HOST = 120, servantX = 122, servantY = 123,
servantZ = 124;

            int l = X.Count;    // get length

            while (X.Count > 1)
            {
                //clear data arrays
                Array.Clear(datX, 0, 1500);
                Array.Clear(datY, 0, 1500);
                Array.Clear(datZ, 0, 1500);

                int lx = X.Count;

                if (l <= 1500)
                {
                    for (int i = 0; i < l; ++i)
                    {
                        datX[i] = X.Dequeue();
                        datY[i] = Y.Dequeue();
                        datZ[i] = Z.Dequeue();
                    }

                }
                else
                {
                    for (int i = 0; i < 1500; ++i)
                    {
                        if (X.Count < 1)
                        {
                            datX[i] = 0;
                            datY[i] = 0;
                            datZ[i] = 0;
                        }
                        else
                        {

                            datX[i] = X.Dequeue();
                            datY[i] = Y.Dequeue();
                            datZ[i] = Z.Dequeue();
                        }
                    }

                }
```

```
                sendData(servantX, servantX, datX);
                sendData(servantY, 0x00, datY);
                sendData(servantZ, 0x00, datZ);

                d[0] = HOST;
                d[1] = 0x00;
                PORT.Open();
                PORT.Write(d, 0, 2);

                for (k = 0; k < 1500; ++k)
                {
                    PORT.Write(dmy, 0, 1);

                    uPdateAll(datX[k], datY[k], datZ[k]);

                    PORT.Read(dmy, 0, 1);
                }

                PORT.Close();
            }
        }

        /// <summary>
        /// Jog function
        /// </summary>
        /// <param name="ID">unit ID</param>
        /// <param name="cnt">feed</param>
        public void Jog(byte ID, byte cnt)
        {
            byte[] d1 = new byte[2]; // for dummy readings
            byte[] d2 = new byte[1];   // for command indication on
DCS


            if (!PORT.IsOpen) PORT.Open();

            d1[0] = 0x6A;   // j
            d1[1] = ID;

            PORT.Write(d1, 0, 2);
            PORT.Read(d2, 0, 1);

            d1[0] = ID;
            d1[1] = cnt;

            PORT.Write(d1, 0, 2);

            PORT.Close();
        }

        /// <summary>
        /// jog stop function
        /// </summary>
        public void JogStop()
        {
            byte[] b = new byte[2];
            if (!PORT.IsOpen) PORT.Open();
            b[0] = 0x73;    // s
            b[1] = 0x73;    //s

            PORT.Write(b, 0, 2);
```

```
            PORT.Close();
        }

        #endregion functions


    }

}
```

REFERENCES

[1] Taira T. and Yamasaki N.; "Functionally Distributed Control Architecture." "Nippon Kikai Gakkai Robotikusu, Mekatoronikusu Koenkai Koen Ronbunshu", 2003

[2] Yook J. K., Tilbury D. M. and Soparkary N. R.; "A Design Methodology for Distributed Control Systems to Optimize Performance in the Presence of Time Delays", IEEE, American Control Conference, 2001

[3] Luntz J. and Messner W.; "A Distributed Control System for Flexible Materials Handling", IEEE Control System Magazine, vol.17, no.1 ,1997

[4] Caspi, P., Mazuet C. and Paligot N. P.; "About the Design of Distributed Control Systems: The Quasi-Synchronous Approach", "Lecture Notes in Computer Science", Springer Berlin/Heidelberg, 2001

[5] Robert W., Fletcher M. and Norrie D. H.; "An Agent-Based Approach to Reconfiguration of Real-Time Distributed Control Systems", IEEE Transactions on Robotics and Automation, vol.10, no.1, 2002

[6] Chan H. and Ozguner U. "Closed-loop Control of Systems over a Communication Network with Queues", in Proceeding of the American Control Conference, vol.2, USA, pp: 811-815, 1994

[7] Koninckx R. and Brussel H.; "Closed-loop, Fieldbus-based Clock Synchronisation for Decentralised Control Systems" In "Reconfigurable Manufacturing Systems and Transformable Factories", pp: 213-235, Springer Berlin/Heidelberg, 2006.

[8] Ferrarini L., Veber C., Schwab C., Tangermann M. and Prayati A.; "Control Functions Development for Distributed Automation Systems Using Torero Approach", IFAC, 2005

[9] Peterson L., Austin D. and Chiristensen H.; "DCA: A Distributed Control Architecture for Robotics", IEEE/ RSJ, vol.11, no.2, 2001

[10] Angelov C. and Sierszecki K.; "Component-Based Design of Software for Distributed Embedded Systems", International Conference on Software and Systems Engineering ICSSEA, vol.3, no.1, 2003

[11] Cauffriez L., Ciccotelli J., Conrard B. and Bayart M.; “Design of Intelligent Distributed Control Systems: A Dependability Point of View”, Reliability Engineering and System Safety, pp: 19-32, 2004

[12] Aarsten A., Brugali D. and Menga G.; “Designing Concurrent and Distributed Control System”, “Communications of the ACM”, 1996

[13] Williams T.; “The Development of Reliability in Industrial Control Systems”, IEEE MICRO, 0272-1732/84/1200, 1984

[14] Yasuda G.; “Distributed Autonomous Control of Modular Robot Systems Using Parallel Programming”, “Journal of Materials Processing Technology”, pp: 357–364, 2003

[15] D’Andrea R. and Dullerud G. E.; “Distributed Control Design for Spatially Interconnected Systems”, IEEE Transactions on Automatic Control, vol.48, no.9, 2003

[16] Nilsson J.; “Real-Time Control Systems with Delays”, PhD. Thesis, Department of Automatic Control, Lund Institute of Technology, 1998

[17] Wyeth G., Kennedy J. and Lillywhite J.; “Distributed Digital Control of a Robot Arm”, in Proceedings of the Australian Conference on Robotics and Automation: ACRA, pp: 217 – 222, 2000

[18] Zongying S., Wenli X., Xu W. and Peigang J.; “Distributed Hierarchical Control System of Humanoid Robot THBIP-1”, in Proceedings of the 4th World Congress on Intelligent Control and Automation, vol.2 , pp: 1265-1269, 2002

[19] Schafer T. and Chevalier M.; “Distributed Motor Control Using the 80C196KB”, AP-428 Application Note, Intel Corporation, 1993

[20] Brennan R.W. and Norrie D.H.; “Metrics for Evaluating Distributed Manufacturing Control Systems”, “Computers in Industry”, pp: 225–235, 2003

[21] Crnosija P., Kuzmanovic B. and Ajdukovic S.; “Microcomputer Implementation of Optimal Algorithms for Closed-Loop Control of Hybrid Stepper Motor Drives”, IEEE Transactions on Industrial Electronics, vol.47, no.6, pp. 1319-1325, 2000

[22] Alford C.O. and Sledge R.B.; “Microprocessor Architecture for Discrete Manufacturing Control, Part I: History and Problem Definiton”, IEEE Transactions on Manufacturing Technology, IEEE Transactions on. 07/1976; 5(2):43- 49

[23] McFarlane D.; “Modular Distributed Manufacturing Systems and the Implications for Integrated Control”, in Proceedings of IEE Colloquium on Choosing the Right Control Structure, 1998

[24] Gerkey B. P., Vaughan R.T., Stoy K., Howard A., Sukhatme G.S. and Mataric M.J.; "Most Valuable Player: A Robot Device Server for Distributed Control", IEEE, vol.51, no.2, pp: 1226-1231, 2001

[25] Malezzoni C., Ferrarini L. and Carpanzano E.; "Object-oriented Models for Advanced Automation Angineering", "Control Engineering Practice", pp: 957-968, 1999

[26] Yook J. K. and Tilbury D. M.; "Performance Evaluation of Distributed Control Systems with Reduced Communication", in Proceedings of the Allerton Conference on Communication, Control, and Computing, 2000

[27] Christian F. and Fetzer C.; "Probabilistic Clock Synchronization", in Proceedings of the Thirteenth Symposium on Reliable Distributed Systems, Oct, 1994

[28] Gaderer G., Loschmidt P. and Sauter T.; "A Novel Simulator for Clock Synchronized Distributed Systems", Control Fieldbus Systems and Their Applications, IFAC, 2005

[29] Benitez-Pierez H. and Garcia-Nocetti F.; "Reconfigurable distributed control using smart peripheral elements", "Control Engineering Practice", pp: 975–988, 2003

[30] Yatkin V. and Hanisch H.M.; "Verification of Distributed Control Systems in Intelligent Manufacturing", Journal of Intelligent Manufacturing, vol.14, no.1, pp: 123-136, 2003

[31] Kowalewski S., Engell S., Preubig J. and Stursberg O.; "Verification of Logic Controllers for Continuous Plants Using Timed Condition/Event-system Models" Automatica, vol.6, no.1, pp: 505-518, 1999

[32] Moore P.R.; "Virtual Engineering: An Integrated Approach to Agile Manufacturing Machinery Design and Control", Mechatronics, vol.13, no.10, pp: 1105–1121, 2003

[33] Thompsona H. A., Benitez-Pereza H., Leea D., Ramos-Hernandeza D.N., Fleminga P.J. and Legge C.G.; "A CANbus-based Safety-critical Distributed Aeroengine Control Systems Architecture Demonstrator", Microprocessors and Microsystems, vol.9, no.11, pp: 345–355, 1999

[34] Kim H.S., Lee J.M., Park T. and Kwon W. H.; "Design of Networks for Distributed Digital Control Systems in Nuclear Power Plants", International Topical Meeting on Nuclear Plant Instrumentation, Controls, and Human-Machine Interface Technologies, CHINA, 2000

[35] Lian F., Moyne J. and Tilbury D.; “Network Design Consideration for Distributed Control Systems”, IEEE Transactions on Control Systems Technology, vol.10, no.2, pp: 297-307, 2002

[36] Walsh G. C., Ye H. and Bushnell L. G.; “Stability Analysis of Networked Control Systems”, IEEE Transactions on Control Systems Technology, vol.9, no.2, 2002

[37] Yuan X., Melhem R., Gupta R., Mei Y. and Qiao C.; “Distributed Control Protocols for Wavelength Reservation and their Performance Evaluation”, Photonic Network Communications, vol.5, no.2, pp: 207-218, 1999

[38] LIU D., NING P. and LI R.; “Establishing Pairwise Keys in Distributed Sensor Networks” ACM Transactions on Information and System Security, pp: 41-77, 2005

[39] Tan K., Lee T. H. and Soh C. Y.; “Internet-Based Monitoring of Distributed Control Systems—An Undergraduate Experiment”, IEEE Transactions on Education, vol.15, no.7, 2002

[40] Tsang J. and Beznosov K.; “A Security Analysis of the Precise Time Protocol”, Vol. 4307/2006, in Information and Communications Security, 50-59. Springer Berlin / Heidelberg, 2006

[41] Hsieh C.,Wan P., and HSU P.; “CAN-Based Motion Control Design”, SICE Annual Conference, pp: 2504-2509, 2003

[43] Kopetz H.; “A Comparison of CAN and TTP”, Annual Reviews of Control, pp: 177-188, 2000

[44] Carvalho F.C., Freitas E. P., Pereira C.E. and Ataide F.H.; “A Time Triggered Area Network Platform with Essentially Distributed Clock Synchronization”, Information Control Problems in Manufacturing, INCOM-2006

[45] Modbus Application Protocol Specification, “ *www.modbus-ida.com* “, Last Checked: 08.06.2009

[46] Lafortune S.; “On Decentralized and Distributed Control of Partially Observed Discrete Event Systems”, Advances in Control Theory and ApplicationsLNCIS353, pp.171-184, 2007

[47] J.G. Engelstad; “Robust Control of Delayed Systems”, PhD. Thesis, University of Florida, 2000

[48] Ray, A.; “Introduction to Networking for Integrated Control Systems”, IEEE Control Systems Magazine, 1989 0272-170818910100-0076

[49] Lian F.L., Moyne J.R. and Tilbury D.M.; “Performance Evaluation of Control Networks for Manufacturing Systems”, in Proceedings of ASME Dynamic Systems and Control Division, vol.22, no.2 pp: 179-186.1999

[50] Lian F.L., Moyne J.R. and Tilbury D.M.; “Performance Evaluation of Control Networks: Ethernet, ControlNet, and DeviceNet”, Control Systems Magazine, IEEE, vol.21, no.1, pp: 66-83, 2001

[51] MAXIM, Application Note 723, “Selecting and Using RS-232, RS-422, and RS-485 Serial Data Standards”, 2000

[52] Riedl M., Diedrich C., Naumann F. and Matzekat I.; “Event Driven Applications for Automation Area”, Information Control Problems in Manufacturing, INCOM, 2006

[53] LIU F., YAO Y., HE F. and CHEN S.; “Stability Analysis of Networked Control Systems with Time-varying Sampling Periods”, Journal of Control Theory and Applications, 10.1007/s11768-008-7186-8 ed.: 22-25, 2008

[54] Zhang W., Branicky M.l S. and Phillips S.; “Stability of Networked Control Systems”, IEEE Control Systems Magazine, 0272-1708/01 ed., 2001

[55] Yook J.K., Tilbury D.M. and Soparkar N.R.; “Trading Computation for Bandwidth: Reducing Communication in distributed Control Systems Using State Estimators”, IEEE Transactions on Control Systems Technology, vol.24, no.2, pp: 503-518, 2002

[56] Rs-422 and RS485 Application Note, www.bb-elec.com, Last Checked: 08.06.2009

[57] RS485/RS422 Circuit Implementation Guide, www.analog.com, Last Checked: 08.06.2009

[58] Fleisig R.V. and Spence A.D; “A Constant Feed and Reduced Angular Acceleration Interpolation Algorithm for Multi-axis Machining”, Computer-Aided Design, pp: 1-15, 2001

[59] Yang M.Y. and Hong W.P.; “A PC–NC Milling Machine with New Simultaneous 3-axis Control Algorithm”, International Journal of Machine Tools & Manufacture, vol.4, no.1, pp: 555–566, 2001

[60] Sanchez-Reyes J.; “A Simple Technique for NURBS Shape Modification”, IEEE Computer Graphics and Applications, vol.14, no.3, pp: 52-59, 1997

[61] Gallinaa P., Scuorb N. and Mosettia G.; “Delayed-reference Control (DRC) Applied to Machining Operations”, International Journal of Machine Tools & Manufacture, vol.47, no.3, pp: 1386–1392, 2005

[62] Xinhua J. and Xingwu C.; “Design on Odd-even Steps Third Order Approach Interpolation Algorithm for Logarithmic Curve”, in Proceedings of the 17th World Congress The International Federation of Automatic Control, vol.5, no.4, pp: 2174-2179, CHINA, 2008

[63] Lai Y. L., Chen J. H. and Hung J. P.; “Development of Machinable Ellipses by NURBS Curves”, International Journal of Mechanical Systems Science and Engineering, vol.12, no.6, pp:29-36, 2000

[64] Beaulieu N.C.; “Extrema of sin x/x function”, Electronics Letters, 1995

[65] Glavonjic M., Milutinovic D., and Zivanovic S.; “Functional Simulator of 3-axis Parallel Kinematic Milling Machine”, The International Journal of Advanced Manufacturing Technology, 10.1007/s00170-008-1643-x, 2008

[66] Tang, Y., Landers R. G. and Balakrishnan S.N.; “Hierarchical Optimal Force–Position–Contour Control of Machining Processes”, Control Engineering Practice, pp: 909–922, 2006

[67] Erkorkmaz K. and Altintas Y.; “High Speed CNC System Design Part I: Jerk Limited Trajectory Generation and Quintic Spline Interpolation”, International Journal of Machine Tools & Manufacture, vol.14, no.2, pp: 1323–1345, 2001

[68] Erkorkmaz K. and Altintas Y.; "High Speed CNC System Design Part III: High Speed Tracking and Contouring Control of Feed Drives”, International Journal of Machine Tools & Manufacture, vol.15, no.3, pp: 1637–1658, 2001

[69] Omirou S. L. and Barouni A. K.; “Integration of New Programming Capabilities into a CNC Milling System”, Robotics and Computer-Integrated Manufacturing, vol.24, no.3, pp: 518–527, 2005

[70] Hu, W.; “Interpolation Algorithm Based on Central Angle Division”, International Journal of Machine Tools & Manufacture, vol.18, no.5, pp: 473–478, 2002

[71] Xia Q. and Rao M.; “Knowledge Architecture and System Design for Intelligent Operation Support Systems”, Expert Systems with Applications, pp: 115–127, 1999

[72] Ivanenko S. A., Makhanov S. and Munlin M.; “New Numerical Algorithms to Optimize Cutting Operations of a Five-axis Milling Machine”, Applied Numerical Mathematics, pp: 395–413, 2003

[73] Qiu H., Ariura Y. and Ozaki H.; “Optimum Circular Interpolation for Plane Curve Contours”, International Conference on: Systems, Man and Cybernetics, 1995. Intelligent Systems for the 21st Century, vol.2, CANADA, pp: 1279-1284, 1995

[74] Xu X.W.; “Realization of STEP-NC Enabled Machining”, Robotics and Computer-Integrated Manufacturing, pp: 144–153, 2006

[75] Koninckx B. and Van Brussell H.; “Real-Time NURBS Interpolator for Distributed Motion Control”, CIRP Annals - Manufacturing Technology, pp: 315-318, 2002

[75] Koninckx B. and Van Brussell H.; "Real-Time NURBS Interpolator for Distributed Motion Control." CIRP Annals - Manufacturing Technology, pp: 315-318, 2002

[76] Omirou S.L.; "Space Curve Interpolation for CNC Machines." Journal of Materials Processing Technology, pp: 343–350, 2003

[77] Samad Z. and Tusof A.; "Reference Pulse CNC Interpolator Based on Eclosing Line for 2D Parametric Curve." Jurnal Teknologi, 67–78.

[78] Koninckx R. and Brussel H. Van.; "Closed-loop, Fieldbus-based Clock Synchronisation for Decentralised Control Systems." In Reconfigurable Manufacturing Systems and Transformable Factories, Springer Berlin/Heidelberg, 2006.

[79] Sandvik Tool Catalog, www.sandvik.com, Last Checked: 08.06.2009

[80] CCS IDE, www.ccsinfo.com, Last Checked: 08.06.2009

[81] Serial Communication, www.ti.com, Last Checked: 08.06.2009

[82] TCP/ IP Protocol, www.linfo.org/tcp_ip.html, Last Checked: 08.06.2009

[83] MACRO.ORG, www.macro.org, Last Checked: 08.06.2009

[84] SERCOS, www.sercos.com, Last Checked: 08.06.2009

[85] FANUC LTD., www.gefanuc.com, Last Checked: 08.06.2009