SCHOOL OF ENGINEERING

INSTITUTE OF TECHNOLOGY, SLIGO

Evaluation of the Determinism of the Actuator Sensor Interface (ASI) on Programmable Logic Controllers (PLC), Personal Computers (PC) and Real-time Operating Systems (RTOS)

Submitted for the Degree of Master of Engineering

Research Student

Shane Loughlin BEng, MIEI, LCGI

Research Supervisors

Fergal Henry B.E. MEng Sc,

Brendan Mc Cormack B.E., MS. (Mech.), Ph.D

Submitted to the Higher Education and Training Awards Council

September 2004

ABSTRACT

Evaluation of the Determinism of the Actuator Sensor Interface (ASI) on Programmable Logic Controller (PLC), Personal Computer (PC) and Real Time Operating Systems (RTOS)

by

Shane Loughlin BEng, MIEI, LCGI

The purpose of this study was to evaluate the determinism of the AS-Interface network and the 3 main families of control systems, which may use it, namely PLC, PC and RTOS. During the course of this study the PROFIBUS and Ethernet field level networks were also considered in order to ensure that they would not introduce unacceptable latencies into the overall control system. This research demonstrated that an incorrectly configured Ethernet network introduces unacceptable variable duration latencies into the control system, thus care must be exercised if the determinism of a control system is not to be compromised. This study introduces a new concept of using statistics and process capability metrics in the form of C_{pk} values, to specify how suitable a control system is for a given control task. The PLC systems, which were tested, demonstrated extremely deterministic responses, but when a large number of iterations were introduced in the user program, the mean control system latency was much too great for an AS-I network. Thus the PLC was found to be unsuitable for an AS-I network if a large, complex user program is required. The PC systems, which were tested were non-deterministic and had latencies of variable duration. These latencies became extremely exaggerated when a graphing ActiveX was included in the control application. These PC systems also exhibited a non-normal frequency distribution of control system latencies, and as such are unsuitable for implementation with an AS-I network. The RTOS system, which was tested, overcame the problems identified with the PLC systems and produced an extremely deterministic response, even when a large number of iterations were introduced in the user program. The RTOS system, which was tested, is capable of providing a suitable deterministic control system response, even when an extremely large, complex user program is required.

ACKNOWLEDGEMENTS

A sincere thanks to Fergal Henry, Brendan McCormack , my supervisors, for their guidance and support.

Thanks to the staff of IT, Sligo for their support, in particular Ray Tobin Automation Technician and all the staff in the Research Department.

Thanks to the staff and management of SLControls for their support and the provision of funding, resources, and time which were contributed to this project.

Thanks to Laurie Reynolds of the Institute of Electrical Engineers (IEE) for the invitation to present a paper from my findings during this study at the prestigious Control 2004 event, at the University of Bath, U.K.

As always thanks to my parents, brothers and wife Siobhán for their continuing support.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

1 INTRODUCTION

The basic aim of this study is to determine if an acceptable level of determinism can be achieved on the Actuator Sensor Interface using the following control platforms:

- PLC – microprocessor based programmable logic controller.
- PC – Pentium based personal computer with a Windows operating system.
- PC – Pentium based personal computer with a Windows operating system and Real Time Extensions (RTE's).
- RTOS – Pentium based computer with a Real Time Operating System.

A benefit of this study will be the generation of comprehensive test data, which can be used to evaluate the performance of each of the above platforms, and to highlight their respective strengths and weaknesses.

The initial phase of this study, as outlined in Chapter 2, required a review of the relevant literature in order to acquire a mastery of the principles and theory of:

- Determinism
- AS-Interface network and relevant Field level networks
- PLC control systems
- PC based control systems
- RTOS based control systems

On completion of the above stated objectives, previous methodologies, which have been used by other researchers, were explored in Chapter 3, and a number of control systems were selected for test. Using the knowledge gained from previous research suitable tests were designed to stress test each of the control systems.

After the tests were designed and the control systems acquired and programmed, the tests were run in sequence. The results were gathered, and incorporated in Chapter 4. These results were subsequently analysed in order to determine the strengths and weaknesses of each control system. Statistics were explored as a methodology to provide a numerical method of quantifying the performance of different control systems.

Chapter 5 consists of a brief discussion of all of the control systems, which were tested, and their corresponding results. Conclusions were drawn in Chapter 6 which outline the author's interpretation of the results as found during the course of the study.

2 LITERATURE REVIEW

2.1 Determinism

A fundamental definition, which had to be established at the outset of this study, is the definition of determinism. It is a word which is used very flippantly and there are numerous different definitions [1], [2], [3], [4]. Determinism is not suited to being described in isolation. A much better understanding of determinism can be achieved, if it is explained in the context of why determinism is actually required. For this purpose, this study defines determinism as follows:

- *In a real time system the correctness of the computations are not only dependent on the logical correctness of the computations, but also upon the time at which the result is produced*
- *If the timing constraints of the system are not met, system failure is said to have occurred*
- *The operation of the above real time system is thus dependant on deterministic architecture and operating system*
- ***A deterministic architecture is an architecture where the worst-case response time can be stated with 100% certainty***

Real time systems can also be subdivided into 2 main categories, namely hard real time systems and soft real time systems. In hard real-time systems every event is serviced, and the task associated with that event is started and finished within a bounded period of time. In a soft real-time system some events may be dropped (i.e., never serviced) and the time required to service the event is not guaranteed to be bounded [5]. Figure 1 provides a diagrammatic explanation of the different methods in which a soft and hard real-time control system handles periodic tasks. The horizontal arrows that lead from the sampling instant to the real-time task are the event latency, or the delay from the time the task *should* begin execution to the time it *actually* begins execution.

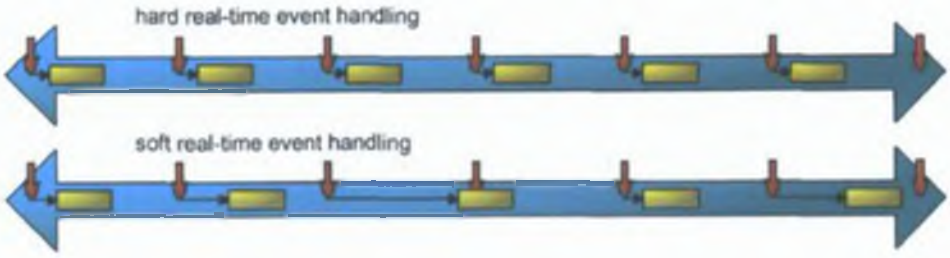


Figure 1: Soft real time versus Hard real time for a periodic task (Source: [5])

With a maximum number of slaves connected to an AS-I network, the worst-case response time is guaranteed to be 5ms. If the controller introduces an event latency, which is greater than 5ms, then there is a danger that critical events may be missed [6].

If the controller event latency cannot be accurately stated then, even though the AS-I is deterministic, the controller is not, and the total control system must be described as non-deterministic.

If on the other hand the controller event latency is deterministic and the event latency can be stated with 100% accuracy, then because it is known that AS-I is deterministic, it can be stated that the total control system can be described as deterministic. Unfortunately this does not mean that the total control system is suitable for the application in question. If the event latency introduced by the controller is much larger than the guaranteed latency of AS-I (5ms) then even though the control system is deterministic, it may be unusable due to an unacceptable event latency.

2.2 *Types of Control*

The requirements for determinism are dependant on the control function that must be achieved. For the purpose of this study the following shall be considered:

- PID Control
- Discrete Control

2.2.1 PID Control

PID Control is often cited as being the most common reason for requiring deterministic control, due the fact that its Integral and Derivative terms should be calculated at fixed intervals. This is a valid requirement but is often used by control system vendors to mislead, or highlight potential benefits with their product offering [5].

The Derivative term in a PID controller can cause instability in a process and should only be used if required [7]. The sampling interval has a much greater effect on the Derivative than the Integral term and the vast majority of controller functions can be achieved using P, and PI control [8]. This is substantiated by the fact that the Closed-Loop or On-Line tuning method proposed by Ziegler and Nichols in 1942, even goes as far as providing formulae for the settings of the PID terms for P, PI and PID controllers [7], [8].

For reliable PID control, the required sampling interval for the Process Variable should be approximately τ /10, were τ is the time constant of the system [8]. Most of the conventional process variables such as pressure, temperature, level and flow, which have been placed under automatic control are relatively slow, often with τ values of many seconds and even minutes [9]. Thus a sampling interval of 100ms should be more than adequate for most PID controller settings, while this value could be substantially increased for P and PI controllers.

2.2.2 Discrete Control

Discrete control systems are used to control processes which are digital by nature. They may have some analogue, and even PID control, but the overall process requires sequential control of digital actuators. The digital actuators typically utilize electrical, pneumatic or hydraulic principles.

In the early 1980's the only digital outputs for many of the PLC's used relays. The switch on time for relays introduced a latency of greater than 10ms [10]. This latency was a limiting factor for the speed of response of the process. During the 1990's, due to the falling cost of semiconductors, transistor based outputs became much more widely available. At this stage it became feasible to create designs with latencies less than 1ms [11]. This increase in speed allowed machine designers to create machines with much faster process responses.

The process variables for the most part consist of actuator position and product detection. The sensors for actuator position and product detection utilize a variety of operating principles such as hall effect, reed switch, pressure switch, photoelectric, etc. The response times of these sensors are typically less than 1ms [10], but many high performance sensors are commercially available with response times of less than 0.01ms [10].

2.3 Fieldbuses

Fieldbuses are industrial communication systems that use a range of media such as copper cable, fibre optics or wireless, with serial bit transmission for coupling distributed field devices (sensor, actuators, drives, transducers, etc.) to a central control or management system. Fieldbus technology was developed in the 1980's with the aim of replacing the commonly used central parallel wiring and prevailing analogue signal transmission (4-20mA or +/- 10V interface) with digital technology [14].

Due to different industry specific demands, geographical locations, and various market forces, several bus systems with varying properties were established in the market.

Figure 2 provides a very good overview of the main families of fieldbus. The first main separation is due to the type of control, which can be broadly separated into machinery and process [15]. Within the machinery section there are 3 main offerings, which are Modbus [16], ODVA [17], and PROFIBUS [18].

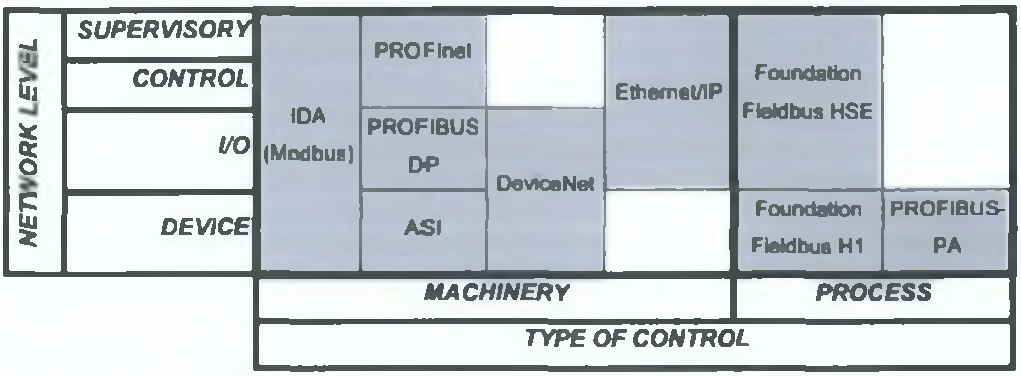


Figure 2: Principal Networks positioned by application (Source: [15])

The initial aims of this study limited the fieldbus investigation to the AS-Interface, which is the device level offering of the PROFIBUS organisation, as depicted in Figure 2. Kriesel and Madelung [6] state that the main limitations of AS-I are the small network size and limited distance that it can span (<100m). For larger industrial applications both Field and Cell level networks are required as outlined in Figure 3. This study would be incomplete without a brief investigation of both the Field and Cell level networks to ensure that they do not inadvertently introduce an event latency that has a negative effect on system performance.

PROFIBUS DP and Ethernet, which is the underlying technology upon which PROFInet is based, were selected as the Field and Cell level networks for investigation. This will allow the effect of the fieldbus on determinism to be quantified, at all layers of the manufacturing organisation, as depicted by Figure 3.

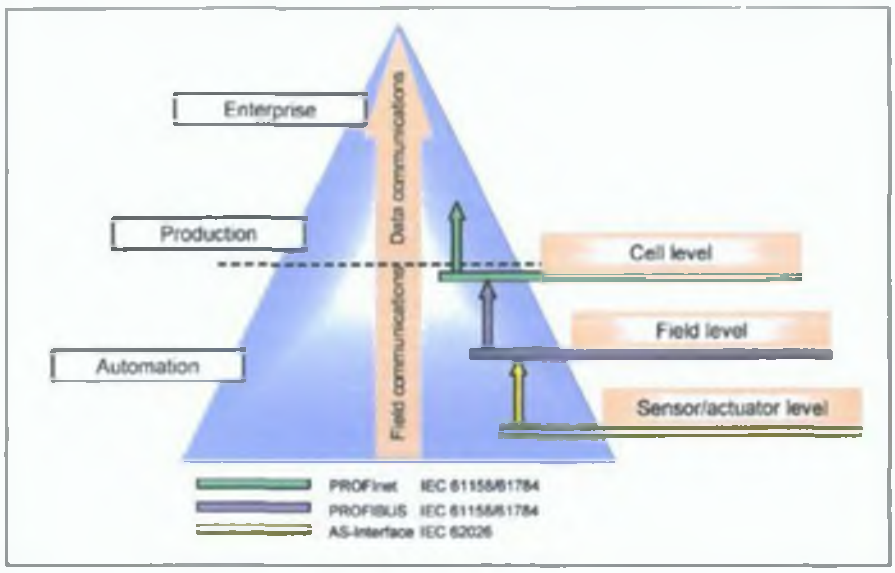


Figure 3: PROFIBUS Communications Model

2.3.1 AS-Interface

The AS-Interface is the simplest automation networking solution [19]. It offers a low-cost solution, which is required in networking. At the same time it provides power for the peripheral elements, transmission of data and diagnostic means throughout the whole system starting at the simple binary sensor up to the highest factory level.

The development of AS-Interface has been done by eleven competitive companies and was funded by the German government. For this development a consortium was founded in 1990 [20] [21]. The main features of the AS-Interface are outlined in Table 1.

FEATURE	DESCRIPTION
Data Transfer	Single-master system with cyclic polling
Addressing	Slaves receive a permanent address via the master or hand-held device
Network structure	Line, ring or tree topology
Transfer medium	Untwisted and unshielded two-wire cable for data and power (24V DC); typically up to 200mA per slave, up to 8A per bus
Cable length	100m max. scaleable using repeaters
Number of slaves	31 AS-Interface slaves max. per network
Number of sensors and actuators	Up to 4 sensors and 4 actuators per slave; maximum 248 binary participants per network
Telegrams	Telegram from the master containing the address; direct answer from the slave (single master operation)
Net data	4 bits master to slave and 4 bits slave to master
Cycle time with 31 slaves	5ms (decreases with decreased number of slaves)
Error detection	Effective detection and retransmission of incorrect telegram
Device interface of the AS-Interface chip	4 configurable inputs/outputs for data, together with 4 parameter outputs and 2 controller outputs (strobe)
Tasks of the master	Cyclic polling of all slaves; cyclic data transmission to and from the control unit (PLC, PC)
Management functions of the master	Initialization of the network, identification of the slaves, acyclic assignment of parameter values to the slaves, diagnostics of data transfer and slaves, error reports to the controller, addressing of replaced slaves

Table 1: Main features of the AS-Interface (Source: [6])

The AS-Interface replaces the complicated cable tree required by conventional wiring techniques by connecting all peripheral components using a single cable. Thus cabling expenditure is greatly reduced. One polling master and the respective slaves replace I/O cards. It is important to note that the AS-Interface, does not link complex devices such as control units and subsystems. Rather it links, normally under extreme environmental conditions, simple devices that often only have a data demand of 1 bit.

The AS-Interface uses *master-slave-access* with *cyclic polling*. In this procedure the master sends a request containing a certain slave address, and the polled slave with the address replies within the specified time, as outlined in Figure 4. From the point of view of the transmission system only one master and only one of the 32 slaves will participate in the data communications at a time. The AS-Interface data packets are short, simply structured and have a fixed length. Four useable data bits are exchanged between a master and every individual slave during one cycle. Therefore the data in both the master and the slave is updated after one cycle. The process image is exchanged between the master and the control unit via dual ported memory. Thus the data is available in the control unit after one cycle.

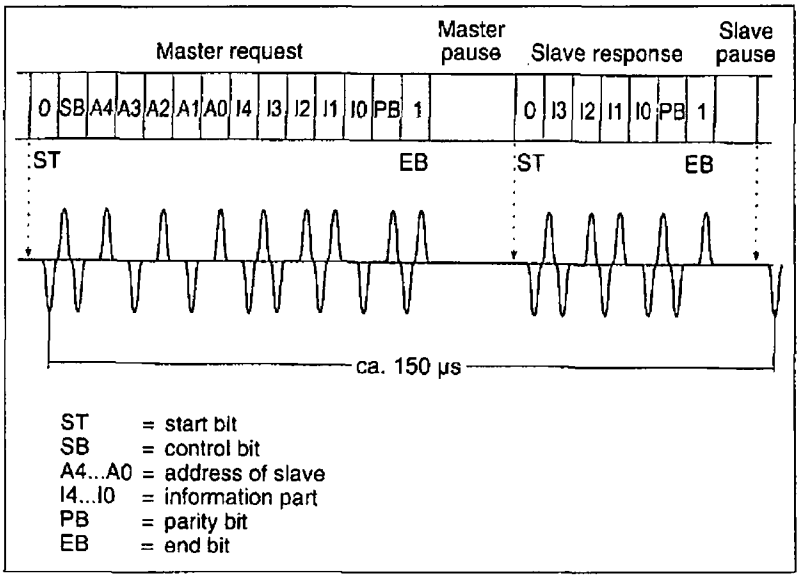


Figure 4: Structure of an AS-Interface message (Source [5])

From the point of view of determinism, one of the major advantages of the structure of the AS-Interface message is that the cycle time of the network automatically adapts to the number of connected slaves, in a clearly defined and measurable manner. If only 6 slaves are connected to the system a cycle time of approximately 1ms is achieved. In the case of a maximum configuration with 31 slaves it will be 5ms [6]. The master can also repeat individual messages when it receives no reply or no valid reply. It is not necessary to complete the full cycle.

The above polling method (cyclic polling), even under extreme fault conditions, is strictly deterministic. After 5ms new sensor data will be available to the control unit and new data will be transmitted from the control unit to the actuators. This satisfies the demand that most PLC systems have during real time processing [6].

On large automation systems, the AS-Interface limitation of 31 slaves with 4 bits of input data and 4 bits of output data cannot cater for the total input and output requirements. Another limitation of the AS-Interface is that the maximum network length of 100m can create an unacceptable constraint. As discussed previously a common method of overcoming the above constraints is to introduce a higher-level fieldbus network between the controller and more than one AS-Interface master. It is extremely important that such a fieldbus network is implemented correctly in order to ensure that the determinism of the control system as a whole is not compromised.

2.3.2 PROFIBUS DP

An extremely common method of expanding an AS-I network, with regard to both distance and total I/O count is to implement a field level network, such as Proibus DP as outlined in Figure 3.

PROFIBUS DP like the AS-Interface was designed to be deterministic. PROFIBUS DP also uses a polling mechanism between master and slave. The time it takes a slave to respond to a message from the master is the *reaction time*. Even if a PROFIBUS DP system receives many I/O signal changes at some point in time, there is no change in reaction time. Because ProfibusDP is deterministic, the reaction time can be accurately calculated [23].

A simplified calculation of system reaction time is available for a PROFIBUS DP whereby the reaction time is derived from the following parameters:

- TSDR (Station Reaction Time)
- The Transmission (Baud) Rate
- The Net Data Length specified
- Min_Slave_Interval (min time between two slave polling cycles)

A field level network consisting of a PROFIBUS DP Master and 31 PROFIBUS DP Slaves can achieve an update time of 1ms for the transmission of 8 bytes of input data and 8 bytes of output data, which is sufficient for 31 separate PROFIBUS DP to AS-I converters and 31 AS-I networks [24]. This clearly demonstrates that a PROFIBUS DP field level network will not introduce any significant latency to the control system.

2.3.3 Ethernet

Ethernet is by far the most widely used Local Area Networking (LAN) technology in the world today. In total, Ethernet outsells all other LAN technologies by a very large margin [25]. It is becoming increasingly common for Ethernet to be used on the factory floor and all indications appear that it is destined to become the preferred high-level fieldbus network.

Our discussion would not be complete, unless we investigated Ethernet to ensure that it can be configured in such a manner as to have a negligible effect on the determinism of the complete system.

The Ethernet Media Access Control (MAC) technology is called Carrier Sense Multiple Access with Collision Detection (CSMA/CD). CSMA/CD is the shorthand version for about seven different steps that make up an Ethernet transmission [25]. These steps are depicted in Figure 5.

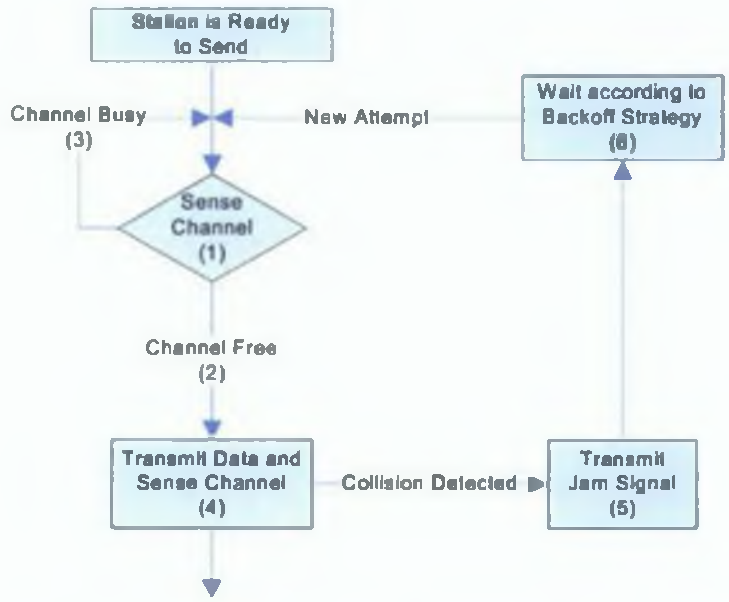


Figure 5: A flow diagram explaining CSMA/CD MAC (Source [25])

By definition a shared –media LAN transmission method also implies half-duplex. Half duplex means that a station is either transmitting or receiving, but not both at the same time. This is because in CSMA/CD a station has to listen to see if a channel is available, and only if it is, can a station start transmitting. When one station is transmitting, all others are listening. Thus it is an either-or situation for all stations on the LAN.

CSMA/CD MAC provided a very efficient method of operation for the coaxial cable upon which Ethernet was founded. Sharing the transmission media also brought with it collisions. Collisions are a very effective and efficient method of preventing overload, but are capable of producing unacceptable delays in a situation where determinism is important.

In the early 1990's the following advances were made:

- 10BASE-T wiring was introduced and offered the capability for separate transmit and receive data paths. Before the arrival of 10BASE-T wiring, coaxial cable didn't offer this capability. The use of only one electrical (coax) wire made simultaneous transmission and reception impossible
- The emergence of multipoint Ethernet bridges or switches meant that the physical media were no longer being shared by multiple users but were increasingly being used to connect 2 switches or a switch and a Network Interface Card (NIC) together in a point-to-point manner

In 1992, Kalpana seized this opportunity so that they could effectively double the speed of Ethernet by using full duplex transmission. Full duplex transmission means that a station can simultaneously transmit and receive. Kalpana started working with many other vendors to establish a de facto industry standard for full-duplex Ethernet over Unshielded Twisted Pair (UTP) wire.

The ability of full-duplex to simultaneously transmit and receive data is very clearly explained by Spurgeon [26], and is shown diagrammatically in Figure 6.

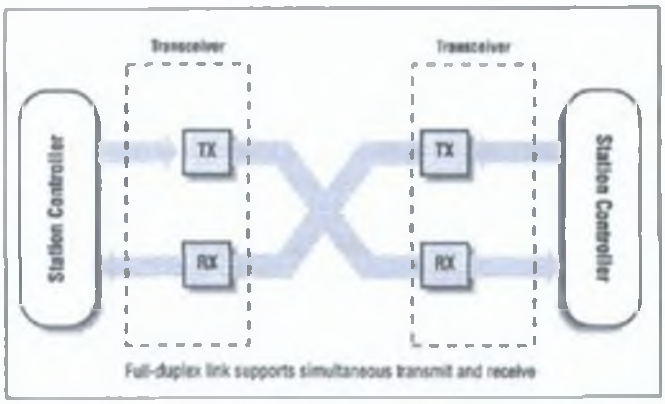


Figure 6: Full Duplex Operation (Source [26])

The CSMA/CD algorithm used on shared half-duplex Ethernet channels is not used on a link operating in full-duplex mode. A station on a full-duplex link sends whenever it likes, ignoring the carrier sense (CS). There is no Multiple Access (MA) since there is only one station at ether end of the link and the Ethernet channel between them is not the subject of access contention by multiple stations. Since there is no access contention, there will be no collisions either, so the station at each end of the link is free to ignore Collision Detection (CD) [26]. Thus the flow control diagram of full-duplex Ethernet is greatly simplified as depicted in Figure 7.

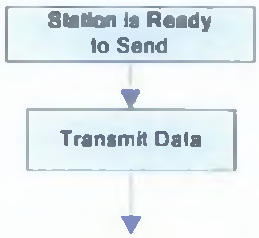


Figure 7: Full Duplex Flow Control Diagram

Breyer & Riley [25] provide an extremely concise diagram outlining the evolution of Ethernet as depicted in Figure 8.

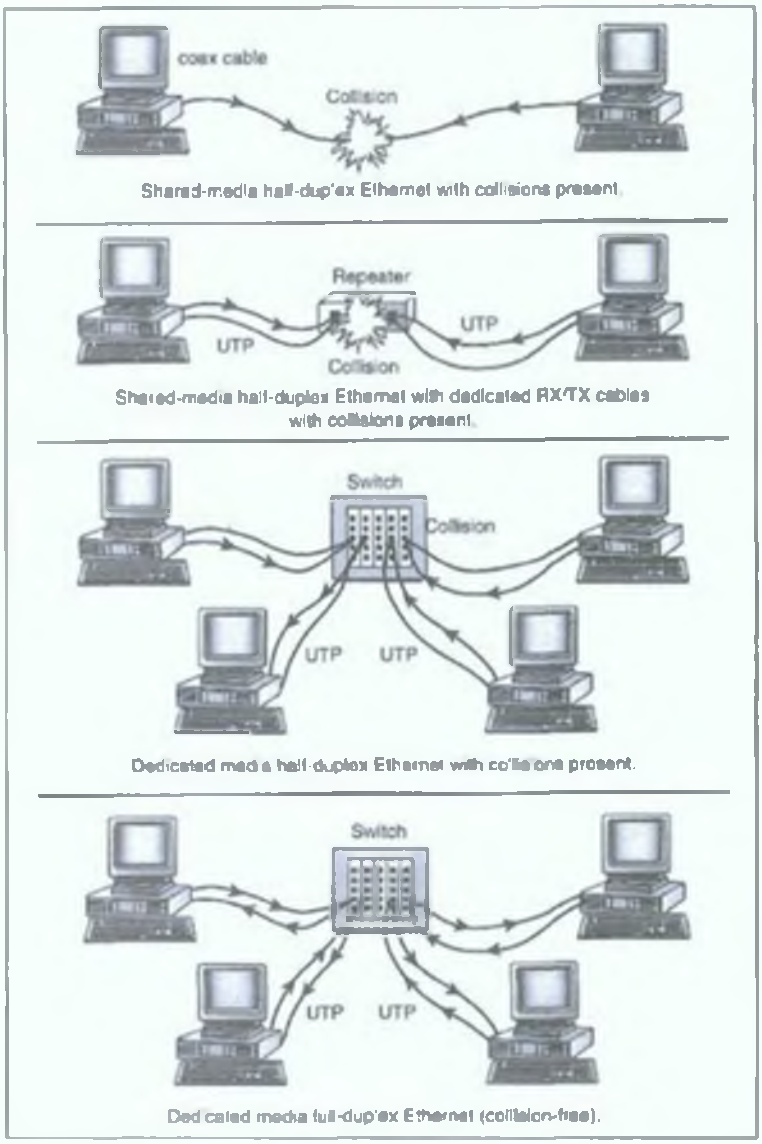


Figure 8: The evolution of Ethernet (Source: [25])

Thus the investigation of Ethernet can be summarised by stating that traditional half-duplex CSMA/CD Ethernet has inherent design issues that prevent it from providing a deterministic response. Emerging solutions for real-time Ethernet such as PROFInet-IRT V3 and EtherNet/IP combined with full-duplex configurations, facilitates the design of deterministic Ethernet networks, which are capable of providing similar and in some cases even better performance than a PROFIBUS DP fieldbus [24].

2.4 PLC

2.4.1 Background

Machine control was first achieved using electromechanical relays in the mid 1900's. Relay control provided reliable control for more than 70 years, with virtually no competition. In the early 1970's enormous advances were made in digital electronics. These advances together with the development of microprocessor based Programmable Logic Controllers (PLC's) meant that relay control systems were faced with true competition. A major advantage of the PLC, over Digital Logic was that it featured a ladder programming language modelled on relay circuitry. This eased the transition for maintenance personnel and played a very important role in ensuring that PLC's emerged as the best overall choice for a control system, unless the ultimate in operating speed, electrical noise immunity, or failsafe operation were required [27].

Throughout the 1970's and 1980's PLC's evolved very quickly, from simple discrete controllers, to much more powerful controllers capable of controlling complex processes. As analogue to digital conversion techniques became more and more sophisticated, it became possible to replace analogue control systems with PLC's capable of performing full Proportional, Integral and Derivative (PID) control. The mathematical capabilities of the PLC increased also and both Integer and Floating Point mathematical instructions became available, and some of the larger processors were even capable of handling complex ASCII strings. Examples of the complexity of the instruction set can be found in [28] & [29].

2.4.2 PLC Operation

A PLC executes in a cyclic manner as outlined in Figure 9. The length of time to read the inputs and update the outputs is normally relatively constant, after the device has been commissioned. The "Operate Program" duration is variable, depending on the code that has to be executed.

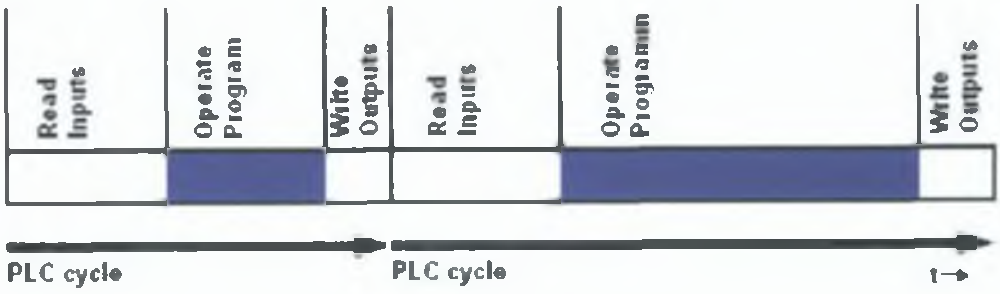


Figure 9: PLC Scan Cycle

On examination of the Instruction Set of a typical legacy non IEC61131-3 compliant PLC [28], it can be clearly demonstrated that the Instruction Execution Timing depends on:

- Whether the instruction is True or False
- Whether the instruction refers to an Integer or Real
- Where the Address is located in Data Memory
- The number of elements acted on per scan

Another noteworthy point is that the more complicated mathematical instructions are extremely slow to execute [28]. It is significant to note that one execution of a TAN function takes almost 0.5ms. In fact the True Execution Time of the TAN function can be as high as 600 times the False Execution Time. It is also important to note that there are program flow instructions such as Conditional Jumps, which the programmer can use.

Thus the length of time taken to scan the code can change dynamically, depending on the logic of the program. Such variation in execution time would undoubtedly seriously affect the determinism of the control system.

Figure 10 demonstrates that 1,000 (1k) PLC commands can execute in 15 µs on a Pentium III 600 PC processor. This is more than 4 times faster than a fast PLC, but it is more than 50 times faster than a standard PLC [30].

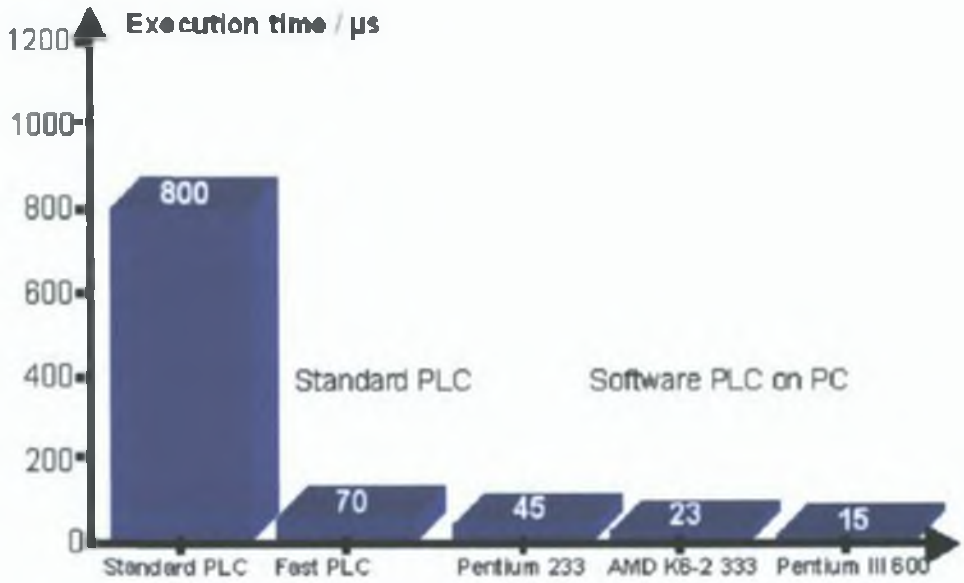


Figure 10: Execution time for 1k PLC commands (Source [30])

Even though conventional thinking suggests that the PLC must be deterministic due to its extremely large installed base on real-time control systems, it is extremely likely that the limitations of its processing power and instruction set could result in situations where the programmer compromises the determinism of the control system by the instructions that are used, and indeed the coding practices used.

2.5 PC (Personal Computer)

2.5.1 PC Hardware

Gordon Moore, one of the founders of Intel, wrote an article for the 35th anniversary issue of Electronics magazine published in April 1965 [31]. Moore had been asked to describe the future of electronics. His research team had recently doubled the capacity of a silicon chip. Balancing innovation and economic factors, Moore extrapolated that the number of transistors on a silicon chip could double each 18 months for the next decade. Professor Carver Mead of Cal Tech, later dubbed the prediction as "Moore's Law".

Moore's Law is also used to describe the law's results: the continuing exponential growth of digital capability and improved price/performance. Over the following 30 years, Moore's Law was proven to be accurate and has resulted in dramatic increases in performance, as depicted by Figure 11, while achieving enormous cost reductions, as depicted by Figure 12, on the PC platform.

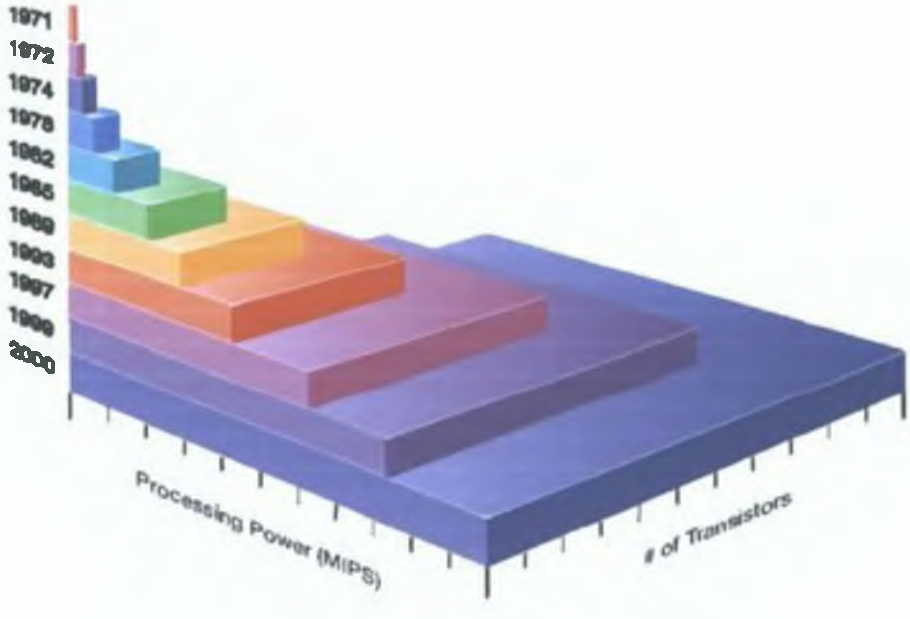


Figure 11: Moore's Law - effect on performance (Source [31])

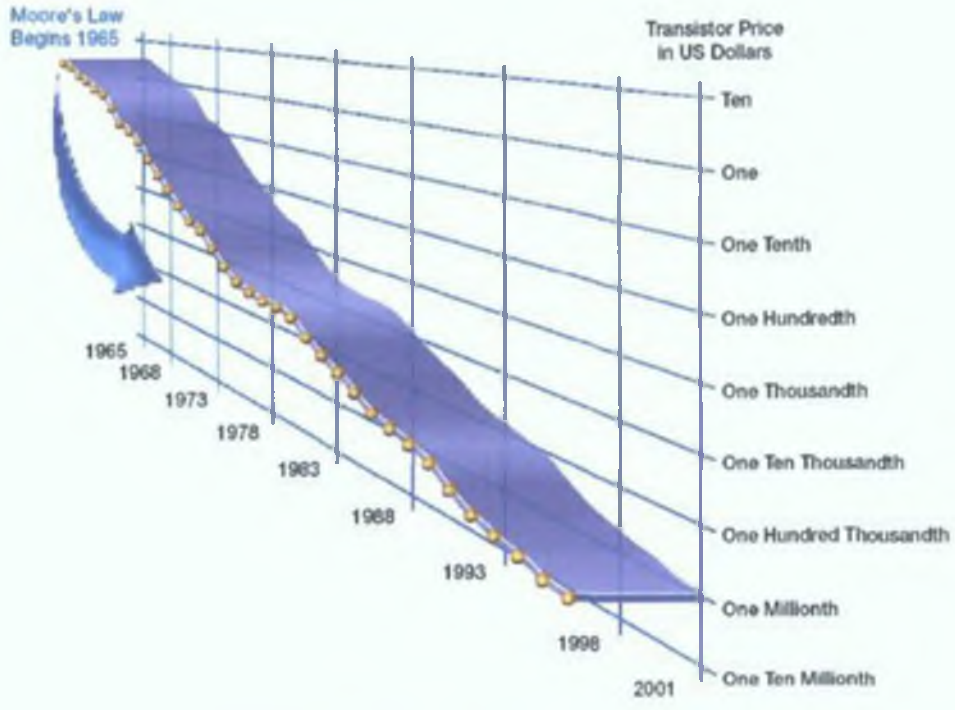


Figure 12: Moore's Law - effect on cost (Source [31])

Moore's Law has continued to be obeyed in the period 2000 – 2004, which is not contained in Figure 11 and Figure 12.

2.5.2 PC Software

From the start Microsoft have dominated the PC operating system software market. The Microsoft operating system product families fall into 2 major groups, which are:

- Descended from DOS (Windows 1.0-3.11, Windows 9x, Windows ME)
- Descended from Windows NT (Windows 2000, Windows XP)

Windows 1.0-3.11 and Windows 9x are not suitable for use in Real Time Systems [32]. But some vendors such as Rockwell [9] provide automation systems based on Windows NT, therefore this study would not be complete without a detailed examination of Windows NT.

2.5.2.1 Windows NT

Within Windows NT and operating systems descended from Windows NT (e.g. Windows 2000 and Windows XP), user applications are defined as processes. Windows NT is a pre-emptive operating system that allows multiple processes (applications) to run at the same time. A process has a number of properties associated with it. For real time applications, one of the most important properties is the priority class (such as real-time) that define the basic priority at which the application will run. The priority model within Windows NT includes 32 priority levels, of which 16 are reserved for the operating system and real-time processes [33].

Each process maintains a private address space to ensure that it will not interfere with other processes, and each process has a base priority class. As shown in Figure 13 below, real-time applications can run with a base priority class of 31 (highest priority), 24 and 16. Typically real-time applications can run with a base priority of 24.

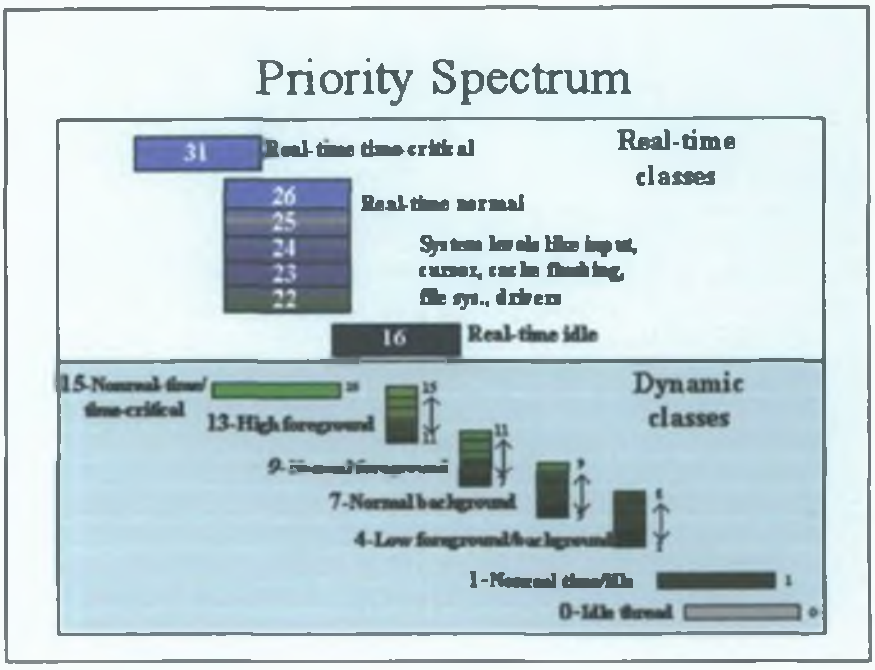


Figure 13: Windows NT Priority Classes (Source [33])

A program is a static sequence of instructions, whereas a process is a set of resources reserved for the thread(s) that execute the program. A thread is the entity within a process that Windows NT schedules for execution. Without it the process's program can't run [34]. Thus each process must have one or more threads associated with it, within the same address space; there each thread represents an independent portion of that process. These threads inherit the properties associated with each process, including the priority level.

To the Win32 Application Programming Interface (API), each thread has a priority based on a combination of its process priority class and its relative thread priority. The mapping from the Win32 priority to the internal Windows NT numeric priority is shown in Table 2 [34].

The priorities shown in Table 2 list the base priority of a thread, which is derived from the process' priority class. The thread's priority can be programmatically changed, within defined limits, by calling the function SetThreadPriority [34]. However, a thread's priority can change as the thread executes. The system can boost a thread's priority higher as time goes on and reduce the priority back down to the base, though windows NT will never reduce a thread lower that its base priority.

For example, a process running at real-time class 24 can have threads that run anywhere between classes 26-22, depending on their own independent priority. These threads will always stay within the real-time priority class.

		Win32 Process Priority Classes			
		Real time	*High*	*Normal*	*Idle*
Win32 Thread Priorities	Time Critical	31	15	15	15
	Highest	26	15	10	6
	Above normal	25	14	9	5
	Normal	24	13	8	4
	Below normal	23	12	7	3
	Lowest	22	11	6	2
	Idle	16	1	1	1

Table 2: Mapping of Win32 to NT Numeric Priorities (Source [34])

According to Microsoft; an application that is running the Real-Time priority class (a base priority of 16) can potentially take so much of the available CPU resources that no resources will be available for other processes or threads. This includes a possible "starving" of both the mouse and the keyboard. This implies that the mouse may become unavailable to click another application to execute an action. It also implies that the keyboard may not respond if you try to press CTRL+ESC to get the Task List and cancel the application. Microsoft even go so far as to say:

> "Although Windows NT has good real-time capabilities; it is not designed to compete with a special purpose real-time system. If you need such a system, obtain the special real-time hardware and supporting operating system" [36].

In 1998 Rockwell Automation provided a very comprehensive white paper outlining why they chose to use the Windows NT Operating Systems without real time extendions such as Radisys InTime, VenturCom RTX and Hyperkernel, for their PC-architecture-based soft control [9]. This is a very interesting development, because no other PLC manufacturer took this stance, and it directly contradicts Microsoft's recommendations.

Rockwell Automation claims that running soft control in the Real Time (RT) class prevents other applications from "severely" affecting determinism. This statement is based on the assumption that lower priority applications will not cause Ring 0 device drivers to perform operations. It is extremely likely that lower priority applications will require device drivers to perform operations such as reading and writhing to disk, network communications, audio, etc. [37]. Rockwell warn that if the periodic tasks are set for very short time periods, a situation can occur where the keyboard and mouse are not recognised by Windows because Windows is spending all of its time executing the real-time tasks of their software. This would appear to point directly to the warning given by Microsoft stating that applications running RealTime priority class can actually cause a potential "starving" of both the mouse and keyboard [38].

2.6 RTOS

In the early 1990's a number of RTOS's were commercially available. At that time one of the largest purchasers of control systems was the General Motor Power Train Group (GMPTG). At this time GMPTG took a strategic decision to drive PLC providers to move to Open Control Systems (OCS) as outlined below:

> *"Historically, divisions that are now parts of GM Powertrain Group (GMPT) have been considered to be leaders and innovators in the field of industrial control. The basic architecture of a Programmable Logic Controller (PLC) was described by a group of GM engineers in the early sixties." [39]*

> *"Currently, most CNC, motion and discrete control applications within the automotive industry incorporate proprietary control technologies. There are difficulties associated with using proprietary technologies such as vendor-dictated pricing structures, non-common interfaces, higher integration costs and the requirement of specific training for troubleshooting and operation. Controller elements, a modularity concept, and higher-level requirements for various elements of an open modular architecture controller are stated to convey the definitions of open, architecture controller in the context of automotive applications. Satisfying these requirements will enable an open, modular controller to be* ***economical, maintainable, open, modular*** *and* ***scaleable*** *thus meet the manufacturing needs in the automotive industry." [40]*

GMPTG ruled out the use of proprietary RTOS's and endorsed the development of commercially available Real Time Extensions (RTE's) to augment the Microsoft Windows NT Operating system [41]. GMPTG conducted a series of exhaustive tests on the 3 main commercially available RTE's, which were HyperKernel from Imagination Systems Inc., INtime from RadiSys and RTX from VenturCom. GMPTG concluded that Windows NT in isolation should not be used for control, but that any of the above RTE's used correctly with Windows NT is capable of providing a deterministic OCS capable of providing hard real time control [41].

In the late 1990's the RTE providers did a lot of further work [42], [43], [44], [45], [46], [47], [48], but the majority of the tests that were carried out were subsets of the tests carried out by the GMPTG [41]. A lot of the resulting papers were extremely commercially oriented, and were specifically aimed at achieving credibility for a particular suppliers product offering.

In 1998 Rockwell Automation produced a white paper, which directly contradicted the use of RTE's and attempted to discredit the work done by the RTE providers by stating [9]:

> *"Several vendors opted to collect one data point on the Windows NT operating system, present it out of context, and then claim that it was unacceptable for control. These vendors have committed to a path of proprietary extensions that will limit their ability to adopt standard Microsoft technologies."*

This paper very clearly outlines some of the potential dangers of proprietary extensions, at the time that it was published (1998), but over the past 5 years, VentureCom [49], and others, have demonstrated that many of Rockwell Automation's causes for concern were not justified, and VentureCom currently provide comprehensive support for all of Microsoft's recent Operating Systems including Windows NT, 2000 and XP.

VentureCom's RTX product is implemented as a collection of libraries (both static and dynamic), a real-time subsystem (RTSS) realised as a Windows XP kernel device driver, and an extended Hardware Abstraction Layer (HAL) as outlined in Figure 14. The subsystem implements the real-time objects and scheduler. The libraries provide access to the subsystem via a real-time Application Programming Interface (API), known as RtWinAPI. RtWinAPI provides access to these objects. The RTWinAPI can be called from within the standard Win32 environment as well as from within RTSS. Using RtWinAPI from Win32 does not provide the determinism available with RTSS, but it does allow for much of the application development to be done in the Win32 environment. All that is required to convert a Win32 program to an RTSS program is to re-link with a different set of libraries [49].

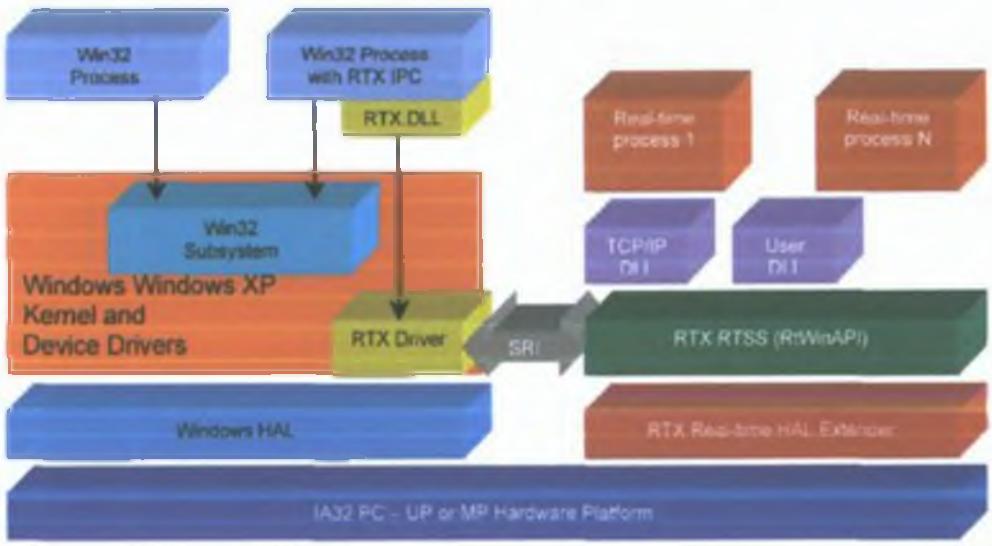


Figure 14: RTX Architecture

Architectures such as the RTX allow Windows operating systems (based on Windows NT) and a real time operating system, based on RTE's to co-exist on one platform.

Even though GMPTG attempted to drive the development of the OCS onto the Windows NT platform augmented with RTE's, there are many other platforms, which cannot be omitted from this study.

An alternative methodology, to gain the business benefits outlined by the GMPTG was supplied by a German company called 3S [50]. 3S created a programming tool for industrial controllers and PLC components based on the international standard IEC 61131-3 [51], called CoDeSys (short for Controller Development System). This is an innovative approach because the CoDeSys programming environment is a platform and manufacturer independent programming system. As outlined in Figure 15, CoDeSys allows the programmer to program different PLC's from different manufacturers in the same environment. This functionality allows code portability across multiple platforms, and directly challenges the monopolistic market of the PLC manufacturers as defined by ARC [52].

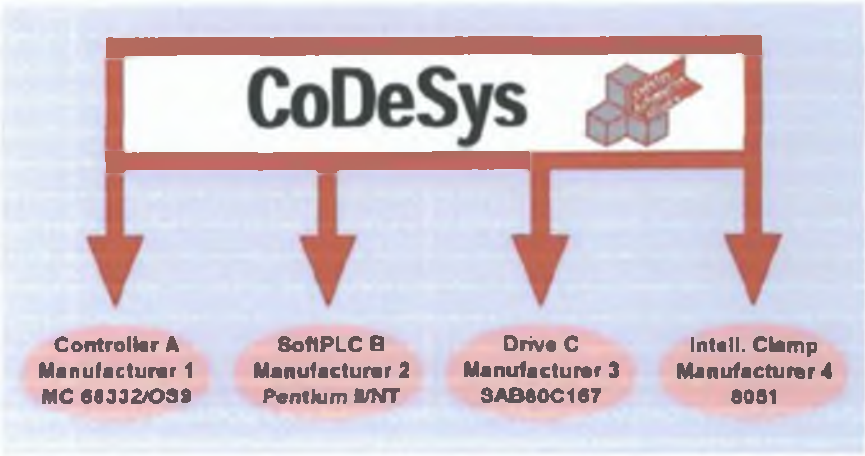


Figure 15: CoDeSys Vendor Independence (Source [50])

CoDeSys SP is a runtime kernel for PLC's. There are 4 different types of CoDeSys SP, but the most powerful is CoDeSys SP 32 Bit Full (CSP32F) for 32-bit processors with multi tasking OS (VxWorks, WinCE & Linux). It is worth noting that the CSP32F can be implemented on the Motorola MC680x0, Motorola MC683xx, Motorola ColdFire, Intel 80x86, Intel Pentium x ARM and Power PC processor families.

CoDeSys also supports implementations on Windows operating systems. CoDeSys SP RTE is a SoftPLC, which runs under Windows. It uses a standard industrial PC with the Windows NT, 2000 or XP operating system. The real time kernel (based on VXWorks) guarantees a deterministic behaviour with jitter in the millisecond (ms) region.

3S created an Automation Alliance of independent automation vendors [53], who have developed OCS based on their products. At present there are more than 29 independent vendors with product offerings based on the SP, and more than 21 vendors with product offerings based on the SP RTE. This represents an extremely large number of product offerings, on both RTOS, and Windows operating systems with RTE's, which are relevant to this study.

3 METHODOLOGY

3.1 Previous work done

Many of the papers produced in the late 1990's [42], [43], [44], [45], [46], [47], [48], are too commercially focussed to provide the basis for a study such as this, but the original study carried out by GMPTG [41] remains a valid study with independent test plans.

3.1.1 GMPTG Test Plan

3.1.1.1 Overall Aim

The overall aim of the GMPTG tests [41] was to determine if hard real-time event handling could be achieved on the Windows NT operating system with Real Time Extensions (RTE's).

If any substantial event latencies (the delay from the time the task *should* begin execution to the time it *actually* begins execution) are introduced into the control system then it is only capable of providing soft real-time event handling, as outlined in Figure 1, and is not capable of providing a deterministic response. It is important to note that some latency is quite acceptable, and in fact all control systems demonstrate some form of event latency. Only when the latency exceeds pre-defined limits, can the system be deemed to be incapable of hard real-time event handling.

3.1.1.2 Test Methodology

The GMPTG engineers created an application, which was installed on the system under test. The application toggled a digital output between +5V and 0V at 0.5ms intervals (as per Figure 16). The GMPTG engineers installed National Instruments digital output cards in the systems under test and connected the digital outputs to a National Instrument Data Acquisition System as shown in the simplified diagram in Figure 17. The Data Acauisition system measured the event latency of the digital output, and thus the event latency of the executing code, by measuring the mark and space of the digital output waveform.

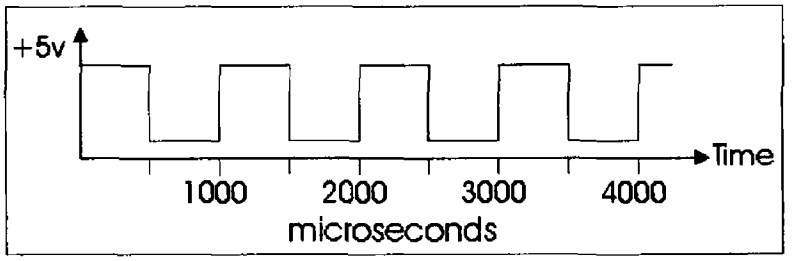


***Figure 16:* GMPTG Tests Digital Output Waveform (Source:[41])**

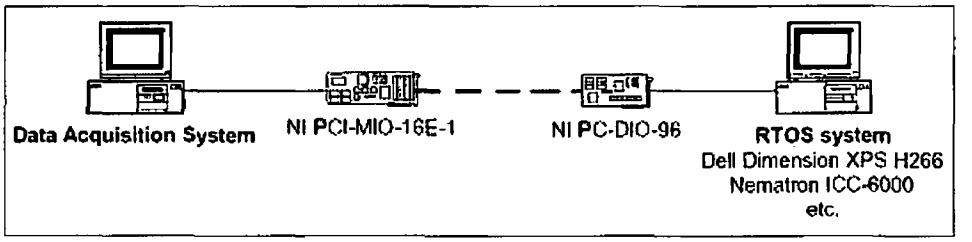


Figure 17: Simplified GMPTG Test Hardware (Source:[41])

While the application was running on the system under test artificial loads and fault conditions were simulated on the system under test, to determine the worst-case event latency of the system, and thus state the determinism of the system. These artificial loads included:

- WinTach running
- A large compile with Visual C++
- Faulty device drivers
- Normal desktop applications
- Faulty programs which cause NT to crash

3.1.1.3 Conclusions

The method used by GMPTG of strobing the digital output on and off at predetermined intervals and measuring the mark space ratio in a Data Acquisition System was selected as an ideal methodology for the tests to be carried out in this study.

It is important to note that the main focus of the GMPTG work was to evaluate just NT based Real Time Extensions and Windows NT. It did not include any tests with regard to PLC's or RTOS's. It is also extremely focused on loading the system under test with tests such as "Normal desktop applications" that are only relevant to Windows NT. The WinTach program represented an extremely controllable method of simulating loads on a Windows NT system, but unfortunately at the time of this study it was no longer commercially available. An alternative benchmarking software application called BurnInTest from Passmark [54] was selected, and used during this study. BurnInTest provided a comprehensive method of simulating variable loads on the CPU, Memory (RAM), 2D Graphics, 3D Graphics, Disks (A: and C:), Network, CD/DVD and USB ports.

The GMPTG tests do not focus on the effect of the processing power and instruction set of the system under test. The processing power and instruction set limitations can have an adverse effect on the determinism of a system and as such had to be tested during this study.

3.2 Control System Product Selection

There are many products based on the PLC, PC and RTOS platforms but for the purpose of this study, the tests have been limited to the comparison of 5 diverse control systems. The control systems products were selected as outlined in Table 3. The control systems have been named with the following syntax:

XXX_YYY_ZZZ

Where:

- ***XXX*** *is the control system family (i.e. PLC, PC or RTOS)*
- ***YYY*** *is the supplier of the operating system or the operating system*
- ***ZZZ*** *is the controller supplier or the application software*

Visual Basic was selected as opposed to Allen Bradley's Softlogix [37] in order to control all Process and Thread priorities, to ensure that they never occupied the Time Critical Priority [33], [34], as outlined in Table 2, during the test.

SYSTEM	DESCRIPTION	VENDOR	PRODUCT
PLC_Rockwell_AB	Microprocessor based Rockwell / Allen Bradley (AB) PLC	Rockwell, Allen Bradley	1785-L60L-C
PLC_3S_IFM	Microprocessor with 3S PLC runtime kernel, and IFM controller	IFM	Controller E
PC_3S_Beckhoff	Pentium processor with Windows 2000 operating system and 3S runtime kernel for Windows, and Beckhoff controller	Beckhoff	TwinCAT
PC_WinNT_VB	Pentium processor with Windows 2000 operating system Visual Basic application	Microsoft	Visual Basic
RTOS_3S_ELAU	Pentium processor with 3S PLC runtime kernel and ELAU controller	ELAU	MaX4

Table 3: Control Systems Product Selection

3.3 Control System Cost

The cost of each control system product is outlined in Table 4 and Table 5. The cost is extremely important, because in the absence of good technical selection criteria, cost is often used for selection. The costs have been split into development and runtime, because development costs are only incurred once, while runtime costs are incurred every time a control system is implemented.

The costs of input and output hardware have not been included due to the many options available. The communications protocols supported as standard by the control system have been included. This is a very important factor, because communications cards can cost from €500 to €1500 if they are to be included in the control system at a later date [16], [17], [18]. The cost differential between the control systems is enormous, with the most expensive system (PLC_Rockwell_AB) costing more than 23 times the cheapest system (PLC_3S_IFM).

SYSTEM	SOFTWARE
*PLC_Rockwell_AB	€2605
PLC_3S_IFM	€365
PC_3S_Beckhoff	€770
PC_WinNT_VB	€520
RTOS_3S_ELAU	€2970

Table 4: Control Systems Development Environment Costs

SYSTEM	COMMUNICATIONS	LICENSE	PROCESSOR	TOTAL
*PLC_Rockwell_AB	DH+, RS232	€0	€13895	€13895
PLC_3S_IFM	PROFIBUS DP, AS-I, RS232	€0	€593	€593
PC_3S_Beckhoff	Ethernet, RS232	€770	**€1000 to €3500	€1770 to €4200
PC_WinNT_VB	Ethernet, RS232	€0	**€1000 to €3500	€1000 to €3500
RTOS_3S_ELAU	Ethernet, RS232, RS485, PROFIBUS DP, CAN	€95	€3970	€4065

Table 5: Control Systems Runtime Costs

* Valid at time of purchase (2002) for processor, this does not include Rack or Power Supply.

** The processor is a PC which is commercially available for less than €1000, but the environment may be harsh and require an industrial PC which is available for €3500.

3.4 Control System Constraints

Each control system has its own inherent constraints, which affect its operation, as outlined in Table 6. The tests carried out in this study have been designed to exaggerate the effect of the constraints on each system, and ensure that the control system is capable of providing a deterministic response, compatible with the time constraint of an AS-I network.

A constraint peculiar to Visual Basic when it is used as a control platform is the latencies, which can be introduced when ActiveX and OCX controls are used [55]. Both ActiveX and OCX controls are independent program modules that can be accessed by other programs in a Windows environment. Many programmers don't consider the fact that these controls have full access to the Windows operating system and even though they may provide an excellent solution to a requirement such as graphing or trending, they may have an adverse effect on system performance.

SYSTEM	CONSTRAINT	EFFECT
PLC_Rockwell_AB	Instruction Set, Slow Execution Speeds	This control platform will have a long scan time if Iteration or Floating Point mathematics is utilised by the Programmer
PLC_3S_IFM	Instruction Set, Slow Execution Speeds, Faulty Slaves	This control platform will have a long scan time if Iteration or Floating Point mathematics is utilised by the Programmer. This controller is also an AS-I master, thus faulty slaves may affect the control system
PC_3S_Beckhoff	CSMA/CD, System Loads	This control platform with Ethernet architecture will be prone to latencies if CSMA/CD is used. Other applications running on the PC may affect system performance
PC_NT_VB	ActiveX, OCX, Process Priority, System Loads	It is extremely likely that ActiveX and OCX components will introduce latencies of variable duration. The priority process and threads of the control application and other applications running on the PC may effect system performance
RTOS_3S_ELAU	None identified	No constraint has been identified for this control platform. None of the constraints applicable for the other control platforms apply.

Table 6: Control System Constraints

3.5 Design of Test to stress Control System constraints

Table 7 outlines the probability of the constraints of each of the control systems affecting the determinism of the control system.

SYSTEM	PID	NETWORK	ITERATION	FLOATING POINT MATHS	BurnIn Test	ActiveX
PLC_Rockwell_AB	Low		High	High		
PLC_3S_IFM	Low	Low	High	High		
PC_3S_Beckhoff	Low	High	Medium	Medium	Medium	
PC_WinNT_VB	Low		Medium	Medium	High	High
RTOS_3S_ELAU	Low		Medium	Medium		

Table 7: Probability of affecting Determinism

A major limiting factor of the microprocessor based PLC's of the 1980's and 1990's is the execution time for the instruction set. There is excellent evidence available to demonstrate that traditional PLC's can be as much as 50 times slower than an Open Control System (OCS) [5], [30]. Over the years the PLC Ladder programming language has been expanded to include mathematical instructions but due to the unsuitability of the control platform these instructions are extremely slow to execute [28].

With the evolution of Six Sigma [56] and Overall Equipment Effectiveness (OEE) [57] it has become increasingly important that the controller is capable of gathering statistical data on the variables it is controlling. It is not uncommon for process engineers to request the minimum, maximum, mean and standard deviation of particular process variables. Gathering such statistical data on variables would be extremely demanding for a PLC, because it requires both floating-point mathematics and numerous iterations through the sample data. It is extremely interesting to note that the instruction set on the mid range traditional PLC's does not even support iteration instructions such as FOR loops [29]. The FOR loop instruction is only available on the large traditional PLC's [28].

Even though PID control was evaluated as having a low probability of affecting determinism, it was included in the test application due to the fact that it is very often associated with determinism. Code was written to simulate and control a PID process for each of the control systems. Every attempt was made to keep the test code as close to identical as possible across all of the various platforms. The test code consisted of the following main code modules:

- PID
- Statistics
- Visualisation (including graphing where possible)

Graphing was specifically selected because it is an essential requirement for controller tuning and it also provided the opportunity to evaluate on the PC_WinNT_VB platform, the effect that ActiveX's or OCX's, which require a lot of PC resources and inter process switching, have on the determinism of a Visual Basic application.

Table 8 outlines the programming languages that were used for each of the code modules on each of the systems under test. A point worth noting is that the OCS's offer by far the most comprehensive programming environments, due to their compliance with IEC-61131 [51], integrated visualisation and trending as standard.

CODE MODULE	PC_WinNT_VB	PLC_Rockwell_AB	PC_3S_Beckhoff	PLC_3S_IFM	RTOS_3S_ELAU
Program Structure	Visual Basic	Ladder	Sequential Function Chart	Sequential Function Chart	Sequential Function Chart
Statistics	Visual Basic	Ladder	Structured Text	Structured Text	Structured Text
PID	Visual Basic	Ladder	Structured Text	Structured Text	Structured Text
Random Number	Visual Basic	Ladder	Structured Text	Structured Text	Structured Text
Visualisation	Visual Basic	Not Available	CoDeSys	CoDeSys	CoDeSys
Graphing	ActiveX	Not Available	Trace	Trace	Trace

Table 8: Programming Languages

The test configurations that were adopted are outlined in Figure 18 and Figure 19 and a photograph of the test systems is contained in Figure 20. Due to the high cost of fieldbus devices only 2 of the control systems under test utilized fieldbus based I/O. Ethernet based I/O configured on an office CSMA/CD Ethernet network was used on the PC_3S_Beckhoff control system, while an AS-I slave was used on the PLC_3S_IFM. Standard I/O was utilized on all the other control systems. In the case of the PC_NT_VB, this was achieved by using one of the parallel port digital outputs.

With the exception of visualisation on the PLC_Rockwell_AB, identical applications ran on each system under test. Because the PLC_Rockwell_AB was a microprocessor based PLC it did not have visualisation available.

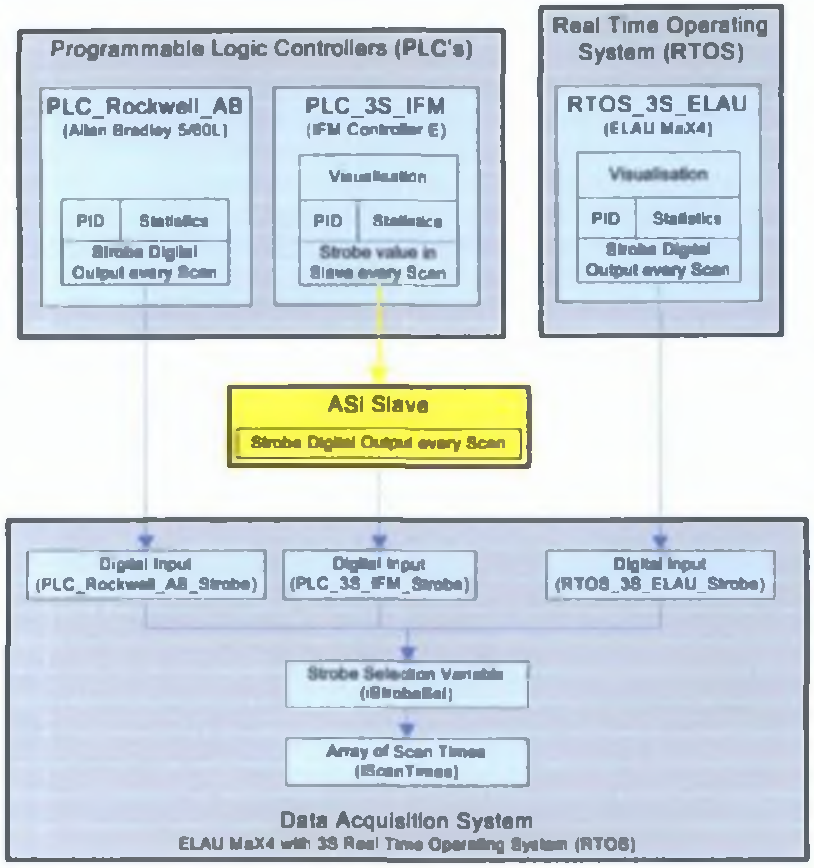


Figure 18: PLC & RTOS Test configuration

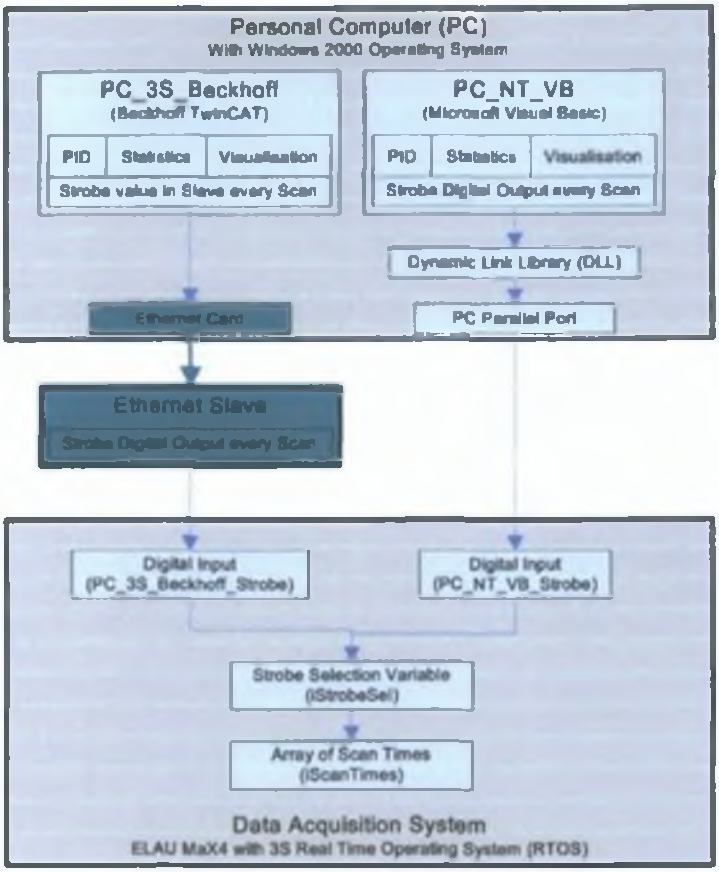


Figure 19: PC Test configuration

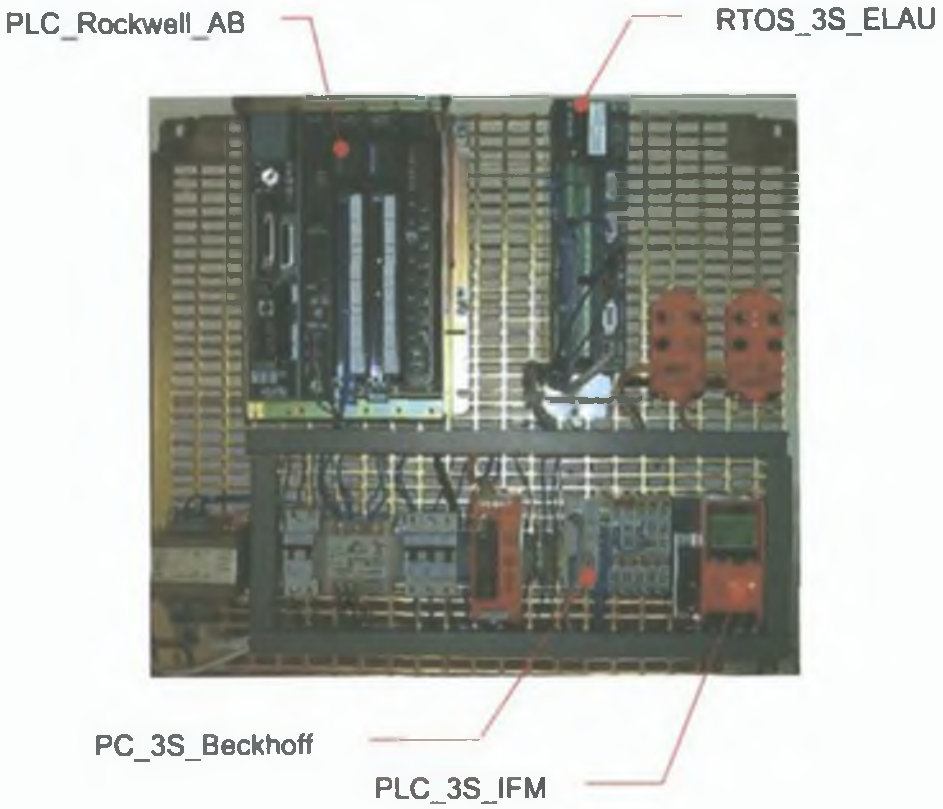


Figure 20: Photograph of test equipment

The execution of the test code is explained graphically in Figure 21. The test code consist of 4 main parts as follows:

- PID Controller Simulation
- Visualisation
- Statistics Iterations
- Strobe the Digital Output

In the PID Controller Simulation module a random number generator is used to create a continuously varying Process Variable. The process variable is supplied to the PID Code module, and applied to the PID algorithm, and a suitable controller output is produced.

On the PC_WinNT_VB platform, the visualisation is an integral part of the code. All of the 3S programming environments contain a visualisation package as an integral part of their programming environment. The PLC_Rockwell_AB platform does not support any visualisation functions.

In the Statistics Iterations module, iterations of statistical calculations were used to stress test each platform. The basic statistics calculations consisted of 2 iterations through an array of 500 floating point values of the Process Variable. On the first pass the Sum, Minimum and Maximum values are calculated. When the first pass is complete it is possible to calculate the Mean. When the Mean is known, the Standard Deviation can be calculated by performing a second pass through the array. The code was also constructed to allow for any number of iterations of the statistics to be performed.

The 'Strobe Digital Output' module, toggled the digital output every time that it was executed. This generated a pulse waveform output similar to Figure 16, but the On time and Off time of the signal varies depending on the length of time that the controller required to execute all of the other code modules.

This On time and Off time were measured directly by the Data Acquisition System and recorded in milliseconds (ms). The 'On time' and 'Off time' were a direct measurement of the latency of the control system under test, thus the term Control System Latency was adopted. The control system latency consists of the Scan Time (i.e. the length of time that the controller required to execute its user program) and all other latencies such as network load, and the length of time required for the transistor in the digital output to switch on or off.

Initially the test duration was set for 1 hour as per the GMPTG tests [41], but this would have taken an enormous amount of time to complete all of the tests in the required timeframe. Using statistical methods it was possible to analyze the performance of each system under test even if the test only ran for a duration of 60 seconds.

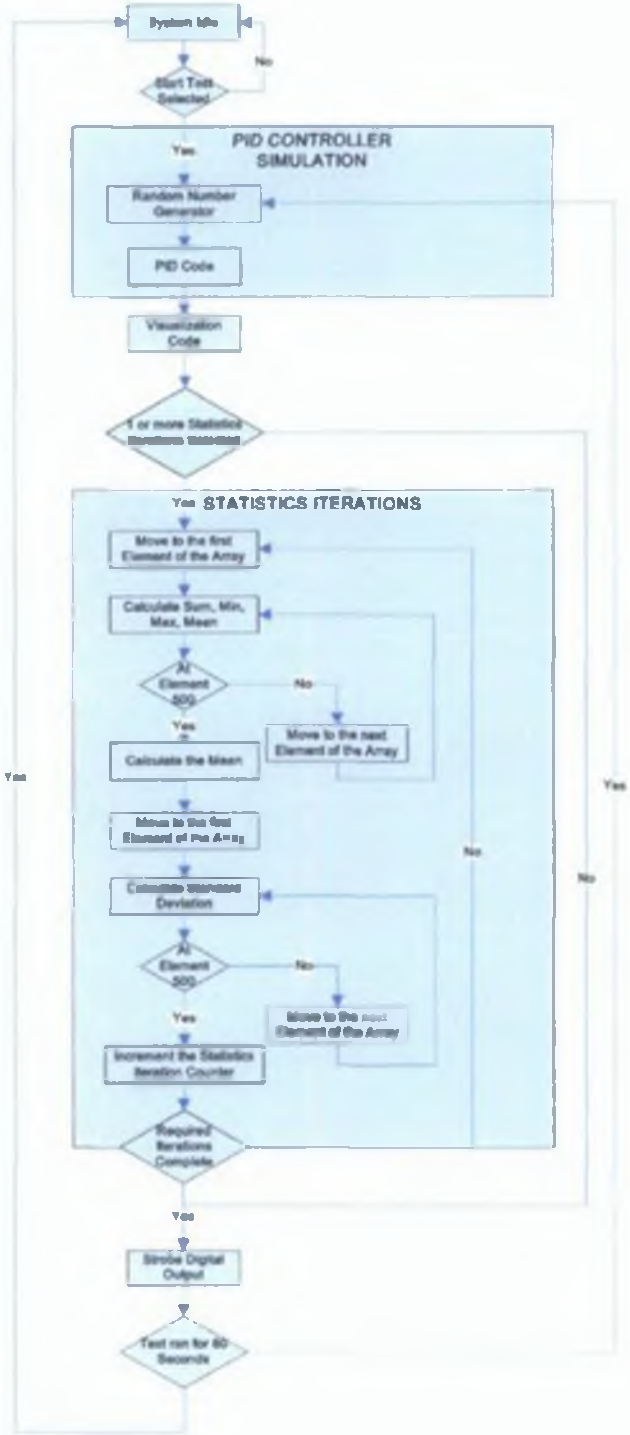


Figure 21: Test code execution flowchart

4 RESULTS

4.1 Collection of Initial Results

The initial results were collected and the control system latencies, which were measured in ms, were inserted into the relevant interval bins. This resulted in a count per interval bin as per the GMPTG tests [41]. An example of the test results tabulated in this format can be found in Figure 22 below. The Std Dev, σ, was calculated according to Equation 1, where x_i is the value of the current observation and $\bar{x}$ is the mean of the observations and *n* the number of samples in the range.

$$\sigma = \sqrt{\frac{\sum_{1}^{n}(x_i - \bar{x})^2}{n-1}}$$

Equation 1: Standard Deviation Equation (Source [56])

Control System Latency (Numerical & Statistics)

	COUNT
0 to 1 ms	13
1 to 2 ms	10
2 to 3 ms	7
3 to 4 ms	22
4 to 5 ms	35
5 to 6 ms	140
6 to 7 ms	1594
7 to 8 ms	1158
8 to 9 ms	1740
9 to 10 ms	13

	COUNT
0 to 9 ms	4732
10 to 19 ms	525
20 to 29 ms	103
30 to 39 ms	11
40 to 49 ms	7
50 to 59 ms	
60 to 69 ms	2
70 to 79 ms	1
80 to 89 ms	3
90 to 99 ms	1

	COUNT
0 to 99 ms	5385
100 to 199 ms	2
200 to 299 ms	
300 to 399 ms	
400 to 499 ms	
500 to 599 ms	
600 to 699 ms	
700 to 799 ms	
800 to 899 ms	
900 to 999 ms	

STATISTICS	
Min	1
Max	121
Average	9 81232597
Std Dev	4.638375853

Figure 22: Results based on GMPTG format

With the very large volume of tests across all of the different control systems, it very quickly became difficult to evaluate the data in numerical format. Better readability was achieved when the results were displayed in graphical format. The data was plotted using a linear Y Axis as demonstrated in Figure 23 and a logarithmic Y Axis as demonstrated in Figure 24. The logarithmic scale had the effect of highlighting the values in every interval bin regardless of the magnitude of the value.

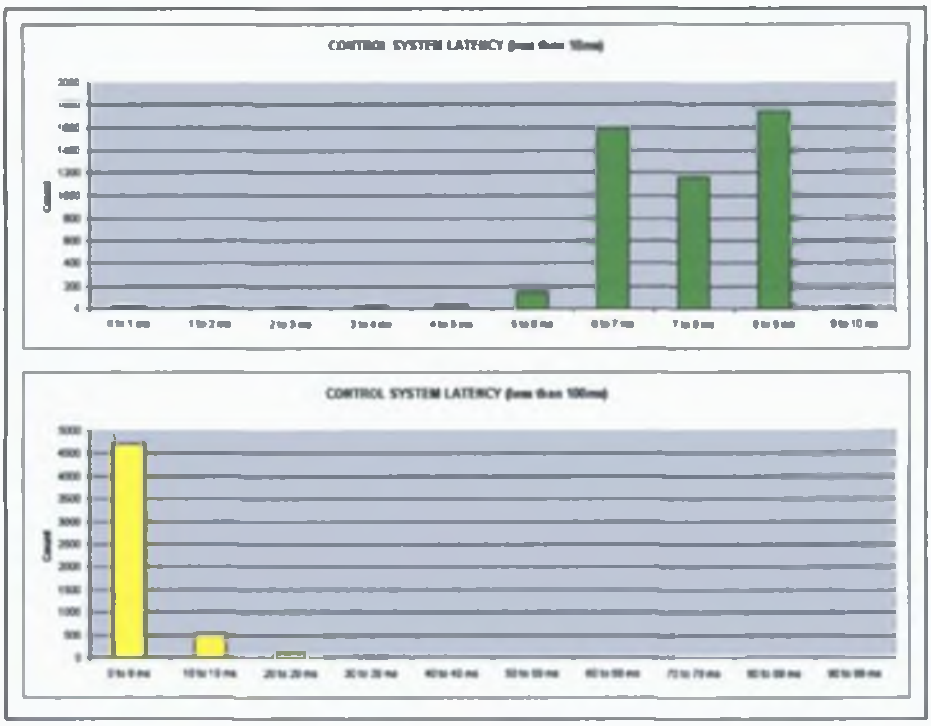


Figure 23: Results displayed in Graphical Format, Linear Y Axis

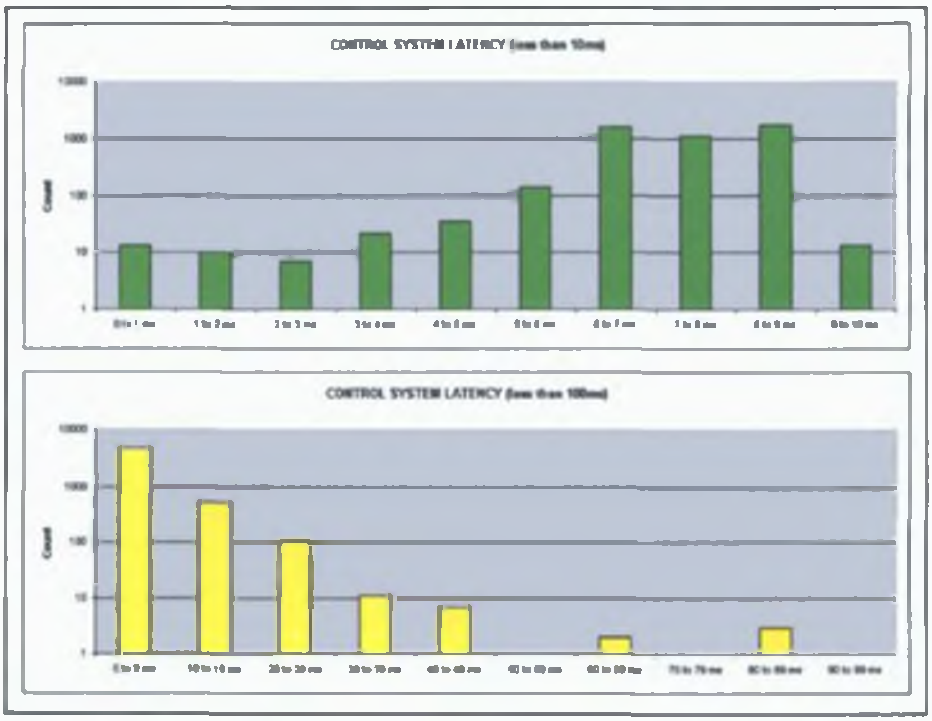


Figure 24: Results displayed in Graphical Format, Logarithmic Y Axis

As the tests were conducted it became apparent that a lot of data was being gathered about the performance of the various systems but there was no objective measurement by which to compare the acceptability of their performances.

Caplen outlined in Figure 25 the percentage of readings between each standard deviation on a normal distribution [56]. An attempt was made to use 6σ and 8σ values but these variables are not useful to determine if a control system is suitable in isolation. Some of the platforms demonstrate an extremely low σ, and as such are extremely deterministic, but the mean is extremely high thus the control system is unsuitable for the control task.

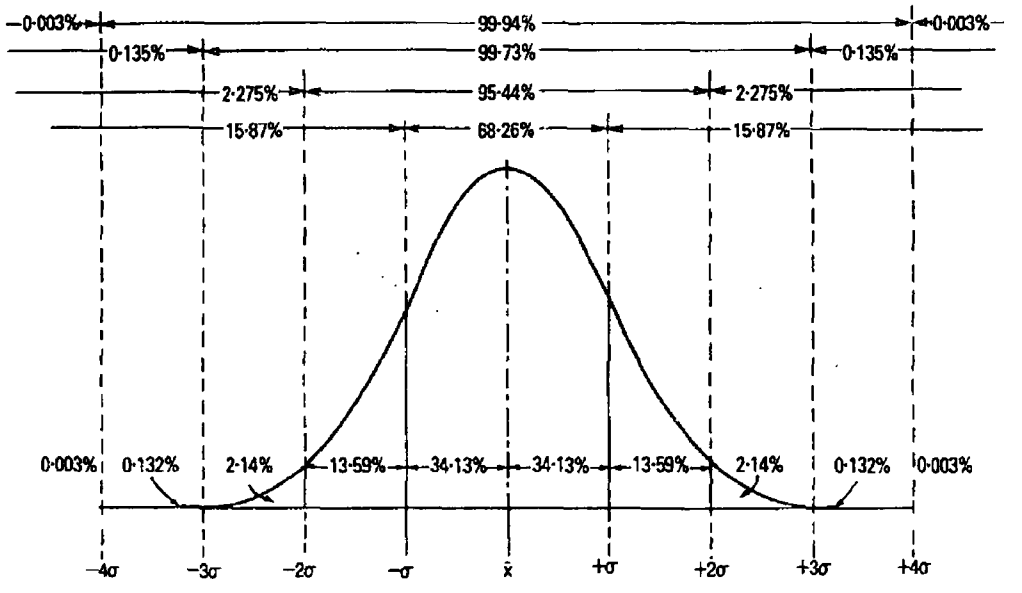


Figure 25: Percentage of readings between each Standard Deviation (Source [56])

4.1.1 Capability Index

Caplen [56] explained that it is often useful to have a simple capability index (C_p) to show how capable or otherwise a process is with respect to its specification limits. A capable process is one where almost all of the measurements fall within the specification limits. Caplen went on to explain C_p as outlined in Equation 2

$$\text{Capability Index} = Cp = \left[\frac{(\text{Total Tolerance})}{(6\,\text{Standard Deviations})}\right]$$

Equation 2: Capability Index Equation (Source [56])

4.1.2 Relative Capability

BS 5700 [57] introduced 2 new terms:

- U = Upper specification limit
- L = Lower specification limit

which combined with Equation 2 facilitated the use of a numeric range to indicate relative capability as follows

- High relative capability >1.33
- Medium relative capability 1.00 to 1.33
- Low relative capability < 1.00

The 3 capabilities listed above are displayed graphically in Figure 26. This methodology provided an ideal solution by providing one figure to indicate if the control system was performing in an acceptable manner or not.

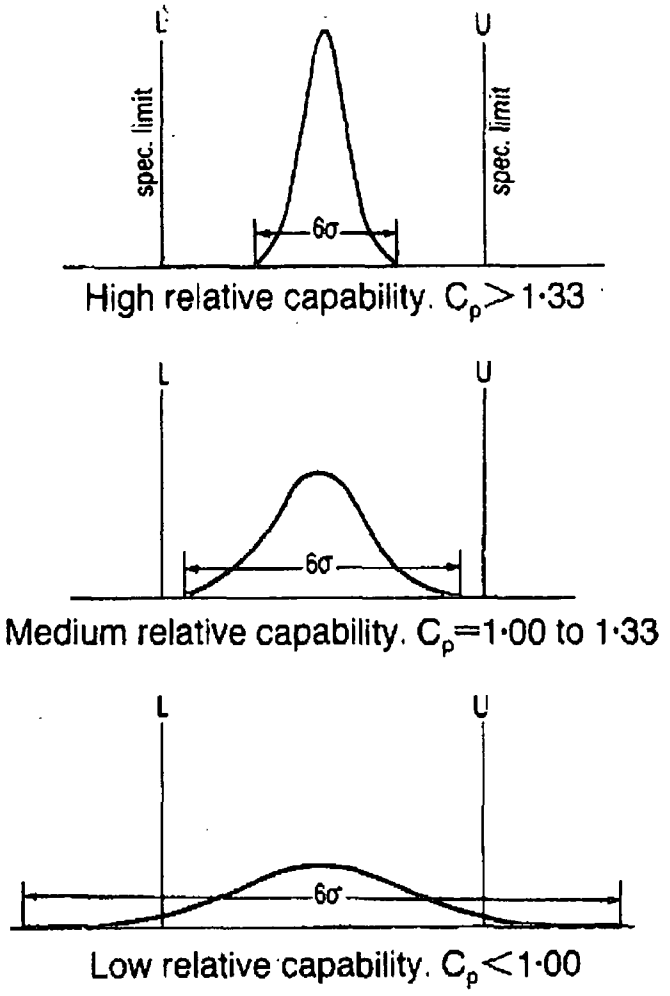


Figure 26: The 3 Relative Capabilities (Source [56])

BS 5700 [58] also points out that when it is not possible to set the process mean midway between the specification limits, a single C_p value is misleading, and it is necessary to compute two C_{pk} values, one for each limit, and use the lower of the two. The equations for each of these C_{pu} and C_{pl} are outlined in Equation 3 and Equation 4 respectively.

$$C_{pu} = \frac{U - \bar{x}}{3\sigma}$$

Equation 3: Capability Index Upper (Source [56])

$$C_{pl} = \frac{\bar{x} - L}{3\sigma}$$

Equation 4: Capability Index Lower (Source [56])

Because it is not possible for us to influence the process mean in these tests C_{pk}, C_{pu} and C_{pl} are more relevant than a single C_p variable.

For the purpose of these control system tests an extremely aggressive set of capability categories were created. This is justifiable due to the criticality of the variable (event latency) being measured. The 4 capability categories were created as follows:

- An extremely suitable control system $C_{pk} > 2.00$
- A suitable control system $1.33 < C_{pk} < 2.00$
- An unsuitable control system $1.00 < C_{pk} < 1.33$
- An extremely unsuitable control system $C_{pk} < 1.00$

Even though it was relatively easy to perform all of the capability calculations, the preferred option was to display the results in graphical format similar to Figure 23 and Figure 24. Unfortunately the results when plotted in Excel were unsatisfactory due to the fact that the chart does not provide a good method of creating a high quality histogram.

A suitable ActiveX, which provides extremely high quality histograms was sourced at a software company called vdisoft [59] and incorporated into a Visual Basic Application.

4.1.3 Setting the Specification Limits

4.1.3.1 Initial Setting

Initially the specification limits U and L were set at 0ms and 15ms respectively and the Target was set to 7.5ms. An initial test was conducted on the RTOS_3S_ELAU, with no statistics iterations, and it produced the data contained in Figure 27 and Table 9. This demonstrated that this controller was well capable of providing a very deterministic response, with an extremely low standard deviation, but the C_{pl} and C_{pk} values were much lower than the C_{pu} value. There appeared to be a skewing effect, which required more investigation, before the tests were started.

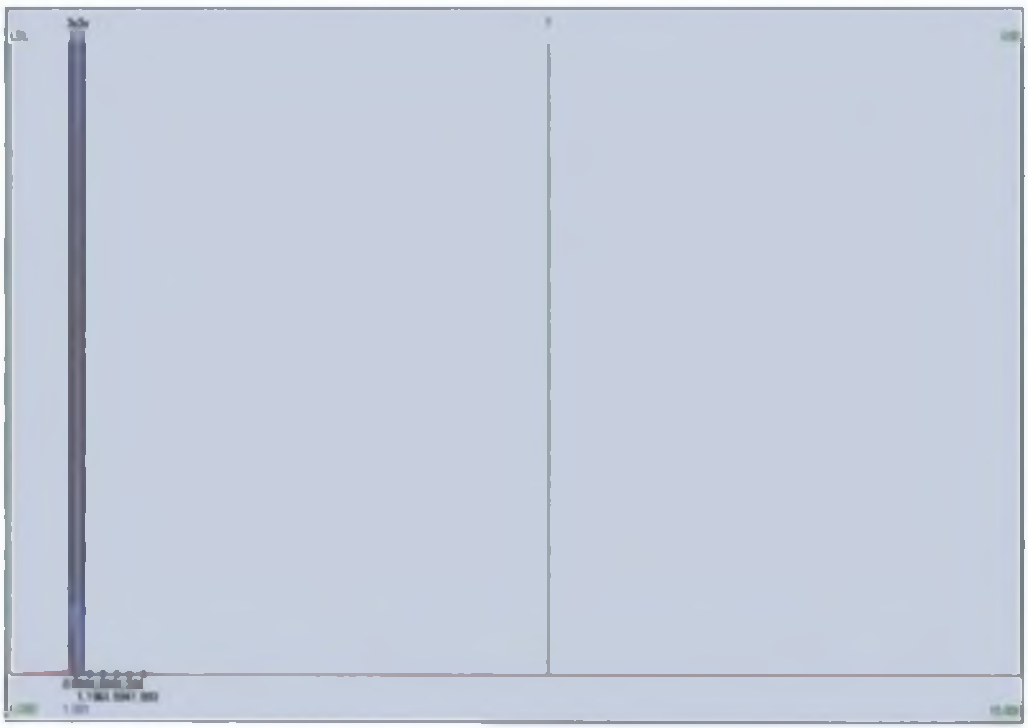


Figure 27: RTOS_3S_ELAU Initial Test Capability Graph

N	C_p	C_{pk}	C_{pu}	C_{pl}	Mean	Standard Deviation
29977	90.288	12.048	168.528	12.048	1.001	0.028

Table 9: RTOS_3S_ELAU Initial Statistics

4.1.3.2 Skewing of the C_{pk}

Even though the capability results are extremely high, they are quite misleading. 1ms is a much better performance than the selected target of 7.5ms, but it does represent a deviation from the target and the C_{pl} value demonstrates this by producing a value of 12.048, while the C_{pu} produces a value of 168.528. BS 5700 dictates that the C_{pk} value must be the lowest of the C_{pu} and C_{pl} values and as such is correct in producing a C_{pk} of 12.048 [58]. Thus when the controller performs better than the target the C_{pk} actually produces a worse result than if all of the values were exactly on the target.

To solve the above phenomenon, every reading that was less than 7ms was deemed to have met its target and the Data Acquisition System code was modified to place this reading in the 7ms interval bin as opposed to the interval bin (i.e. 1 to 6) in which it actually occurred. This modification is explained diagrammatically in Figure 28 and Figure 29. This modification eliminated the skewing of the C_p and C_{pk}

4.1.3.3 Final Setting

The specification limits were reduced in order to ensure that the control system latency could not exceeded 2 x AS-I updates, without being detected. AS-I has a guaranteed worst-case response of 5ms, thus a U specification limit of 2 x 5ms = 10ms would be appropriate. The U specification limit was reduced by 1ms to 9ms to allow for a margin of resolution error. Thus the specification limits U and L were then set to 5ms and 9ms respectively and the Target was set to 7ms.

For completeness the 8σ and Mean + 4σ values were collated in the results. The 8σ value provided an extremely useful indication of the variability of the control system latency, while the Mean + 4σ value provided a statistical estimate of the maximum value of the control system latency.

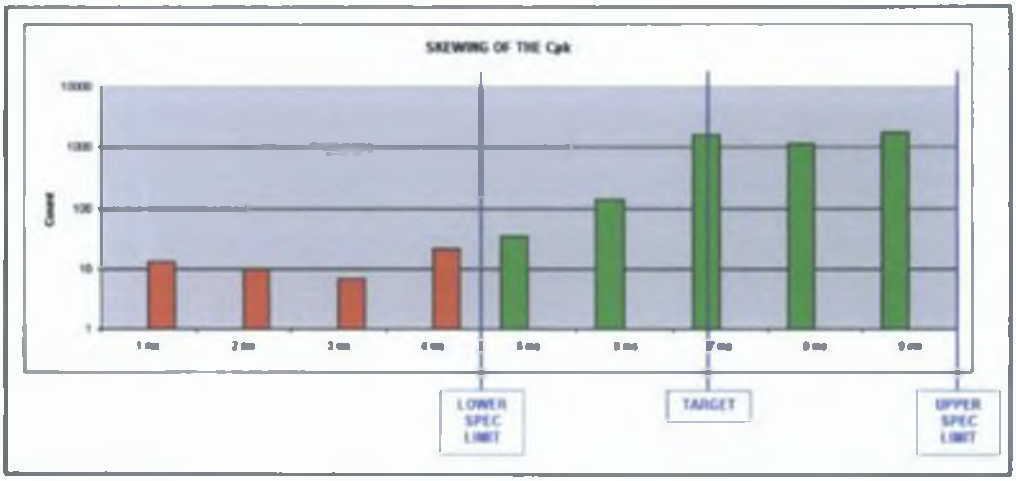


Figure 28: Skewing of the C_{pk}

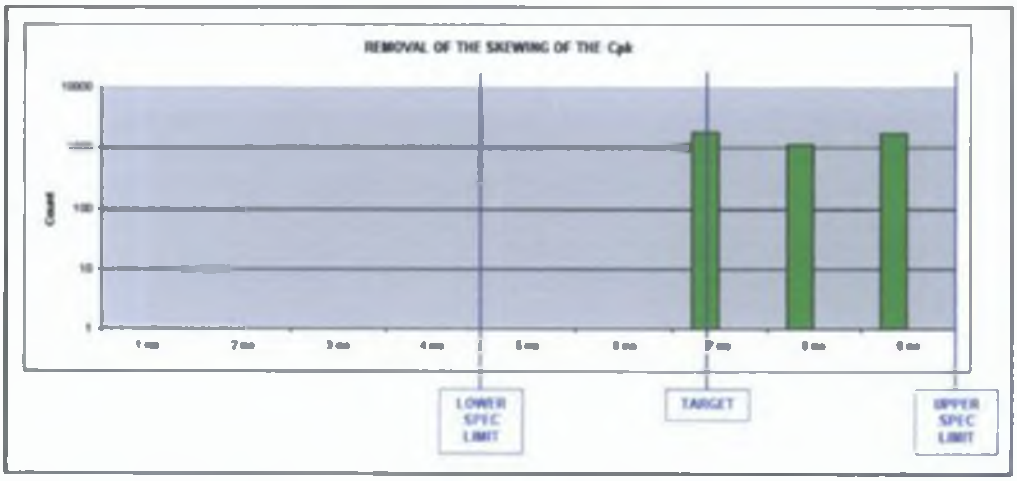


Figure 29: Removal of the Skewing of the C_{pk}

4.2 PLC_Rockwell_AB

The PLC_Rockwell_AB test results are depicted in Table 10 and Figure 30. These results clearly demonstrate that there is an extremely low deviation in the Control System Latency, especially when there are no statistics iterations. However the performance is extremely poor when iterations are required, and even though the results are extremely deterministic the mean value is well outside the acceptable timing constraints of an AS-I network.

Statistics Iterations	Min	Mean	Max	Cpk	Standard Deviation (σ)	8σ	Mean + 4σ
0	7.00	7.00	13.00	14.60	0.05	0.40	7.2
1	90.00	91.41	93.00	-38.47	0.71	5.68	94.25
3	268.00	269.23	271.00	-121.89	0.71	5.68	272.07
5	445.00	447.03	449.00	-197.37	0.74	5.92	449.99
10	890.00	890.74	892.00	-460.99	0.64	5.12	893.30

Table 10: PLC_Rockwell_AB Test Results

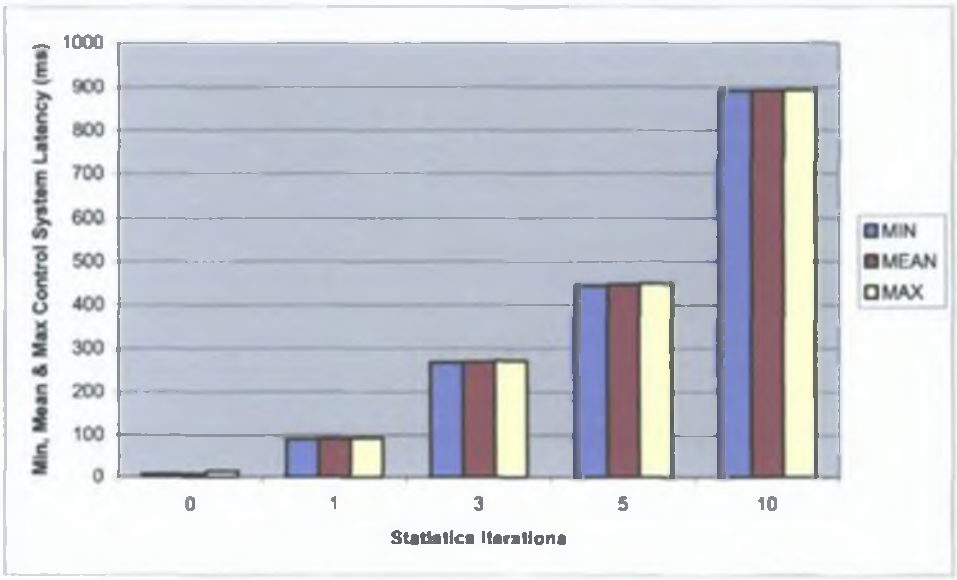


Figure 30: **PLC_Rockwell_AB Min, Mean and Max Results**

4.3 PLC_3S_IFM

The PLC_3S_IFM tests produced very similar results to the PLC_Rockwell_AB tests. These results are contained in Table 11 and Figure 31. These tests also demonstrate that this controller has an extremely low deviation in the Control System Latency, especially when there are no statistics iterations. However the performance is extremely poor when iterations are required. In fact this controller is incapable of performing more than 3 iterations, thus there are no entries for the 5 and 10 iterations rows in Table 11. These tests were carried out with the AS-I network configured for 32 nodes but with only one node connected in order to simulate maximum network loading. Even under maximum network loading the controller was quite capable of providing an excellent C_{pk} of 41.56 when no iterations were performed.

Statistics Iterations	Min	Mean	Max	Cpk	Standard Deviation (σ)	8σ	Mean + 4σ
0	7.00	7.00	8.00	41.56	0.02	0.16	7.08
1	83.00	85.32	88.00	-25.39	1.00	8.00	89.32
3	261.00	263.30	266.00	-89.05	0.95	7.60	267.10

Table 11: PLC_3S_IFM Test Results

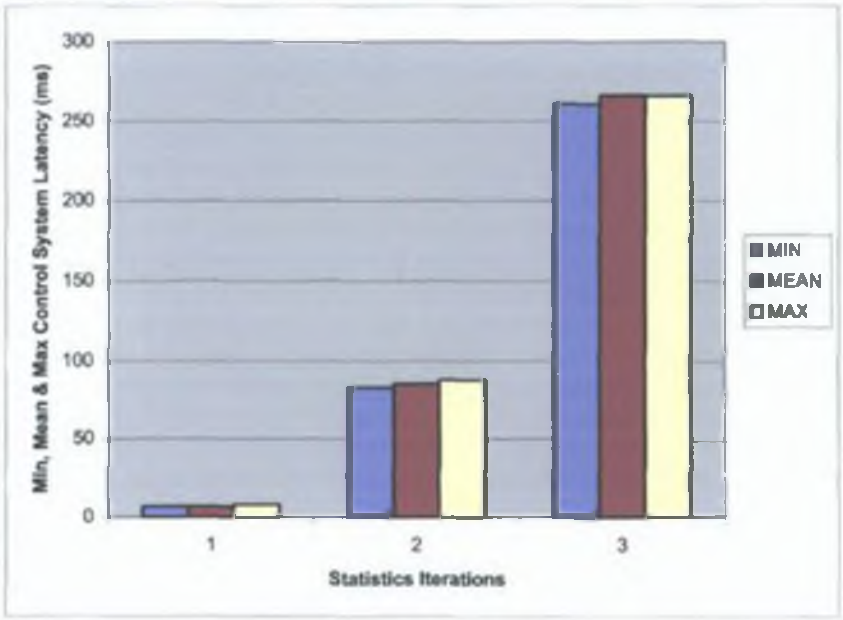


Figure 31: PLC_3S_IFM Min, Mean and Max Results

4.4 PC_3S_Beckhoff

The PC_3S_Beckhoff tests were run with the no loading on the PC (Table 12 and Figure 32), and 100% loading on the PC (Table 13 and Figure 33). The results obtained were dramatically different to the PLC test results. The controller was much less capable and the deviation was much larger. Even though the loading had a measurable effect on performance, the controller was not capable, and as such cannot provide a deterministic response, even with no loading.

The minimum was much larger than expected. Even under extreme loading it would have been reasonable to expect that some of the control system latencies would have been in the 7ms to 10ms range. Upon investigation the Beckhoff digital output card had a response time of 3ms, which undoubtedly introduced latency into the control system. The fact that the digital output was connected using Ethernet configured as CSMA/CD with a standard switch would also have contributed greatly to the variation in the Standard Deviation and the extremely large maximum values that were found.

In order to ensure that there was nothing wrong with the code, a second digital output was strobed in exactly the same manner as the first, but it was connected back as a digital input to the PC_3S_Beckhoff system. Using this technique is was possible to establish that there was a variable event latency between the controller switching its DO on and the actual hardware switching on.

Even though it is outside the scope of this study, it becomes evident that such operation could cause serious malfunction of the control system and highlights the potential lack of determinism on an incorrectly configured Ethernet based control system.

Statistics Iterations	Min	Mean	Max	Cpk	Standard Deviation (σ)	8σ	Mean $+4\sigma$
0	12.00	14.11	166.00	-0.45	3.78	30.24	29.23
1	12.00	14.03	29.00	-1.09	1.54	12.32	20.19
3	12.00	14.56	618.00	-0.13	14.05	112.40	70.76
5	12.00	14.03	32.00	-1.05	1.59	12.72	20.39
10	12.00	18.72	986.00	-0.08	40.23	321.84	179.64
15	14.00	29.41	961.00	-0.25	27.72	221.76	140.29

Table 12: PC_3S_Beckhoff with no Loading Test Results

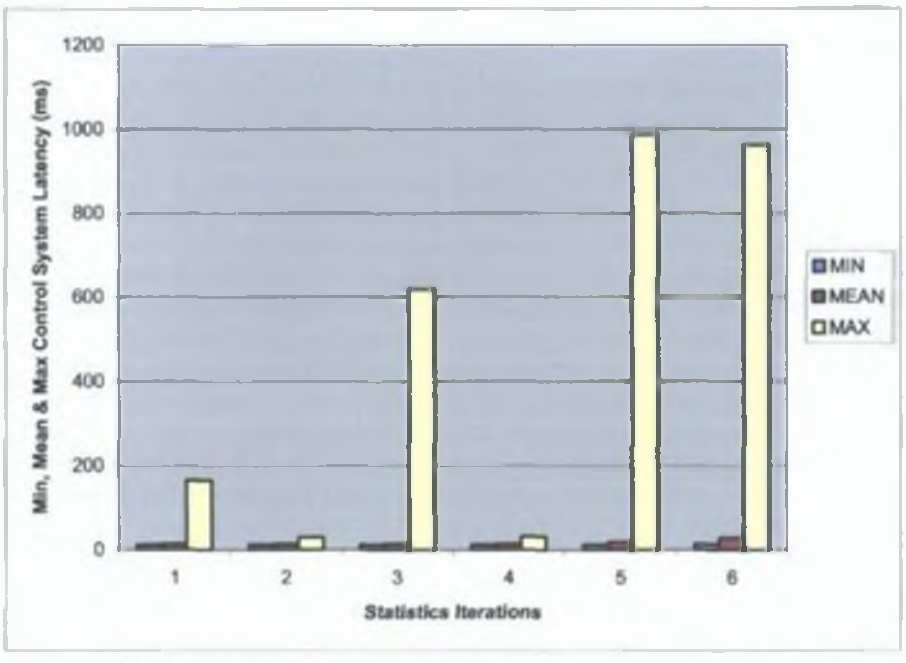


Figure 32: PC_3S_Beckhoff with no Loading Min, Mean and Max Results

Statistics Iterations	Min	Mean	Max	Cpk	Standard Deviation (σ)	8σ	Mean $+4\sigma$
0	12.00	14.23	226.00	-0.32	5.42	43.36	35.91
1	12.00	15.75	724.00	-0.08	29.39	234.12	133.31
3	12.00	14.54	324.00	-0.21	8.97	71.76	50.42
5	11.00	15.61	787.00	-0.08	27.53	220.24	125.73
10	13.00	19.83	946.00	-0.07	50.29	402.32	220.99
15	14.00	28.83	807.00	-0.26	25.23	201.84	129.75

Table 13: PC_3S_Beckhoff with 100% Loading Test Results

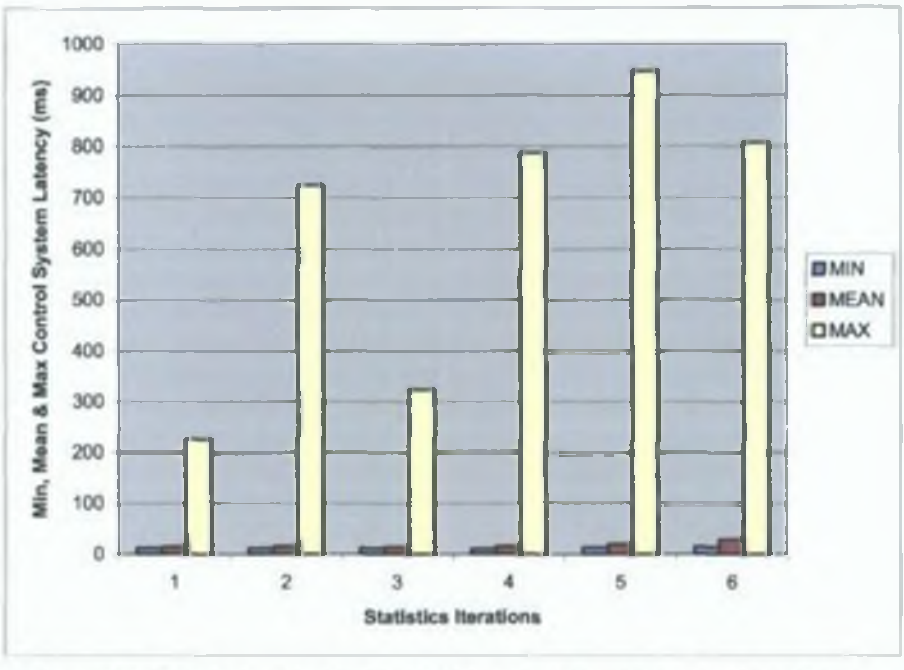


Figure 33: PC_3S_Beckhoff with 100% Loading Min, Mean and Max Results

4.5 PC_WinNT_VB

The testing of the PC_WinNT_VB required the largest body of work. It became apparent very soon that it was not possible for a Visual Basic application on Windows NT platform to provide deterministic performance. This came as no surprise because Microsoft warned of this [33], [35], [36], [38] and the GMPTG test results clearly demonstrate it [41]. Nevertheless Rockwell claimed that by elevating the priority of a task from Normal to RealTime priority that Windows NT was capable of providing satisfactory control [9]. Thus this study would not be complete if Rockwell's suggestion was not investigated.

When the BurnInTest application is not running, there is no load on the PC. This scenario produced the results found in Table 14 and Figure 34. These results clearly demonstrate that the performance of the VB application is unacceptable, but highlights very little improvement in the average performance of the Visual Basic application when it is running as a RealTime task. On closer examination of Table 14 it becomes apparent that the standard deviation (sigma) is sometimes better when the task is run at RealTime priority, but this improvement is erratic, and not reproducible, thus should not be considered.

When a standard Graphing ActiveX is activated in the VB application the performance of the controller is drastically reduced as demonstrated in Table 15 and Figure 35. This clearly demonstrates that ActiveX's can have an extremely detrimental effect on the performance of a VB application regardless of the priority at which the task is running, and as such should not be used in conjunction with an application, which has a controlling function.

When the application is running as a RealTime task, the Graphing ActiveX is not activated, and a varying BurnInTest load is applied to the application, there is only a marginal effect on the control system latency as demonstrated by Table 16, and Figure 36.

Statistics Iterations	Priority	Min	Mean	Max	Cpk	Standard Deviation (σ)	8σ	Mean + 4σ
0	Normal	7.00	9.66	37.00	-0.09	2.52	20.16	19.74
1	Normal	7.00	12.05	428.00	-0.13	7.72	61.76	42.93
3	Normal	7.00	16.68	410.00	-0.32	8.05	64.40	48.88
5	Normal	12.00	21.33	415.00	-0.36	11.39	91.12	66.89
10	Normal	23.00	33.91	426.00	-0.47	17.77	142.16	104.99
15	Normal	34.00	45.85	513.00	-0.59	20.77	166.16	128.93
0	RealTime	7.00	10.20	41.00	-0.15	2.61	20.88	20.64
1	RealTime	7.00	13.01	32.00	-0.61	2.19	17.52	21.77
3	RealTime	7.00	18.13	358.00	-0.35	8.64	69.12	52.69
5	RealTime	12.00	22.47	554.00	-0.34	13.15	105.20	75.07
10	RealTime	23.00	33.37	370.00	-0.84	9.66	77.28	72.01
15	RealTime	34.00	44.60	385.00	-1.06	11.21	89.68	89.44

Table 14: PC_WinNT_VB Test Results without ActiveX and no Load

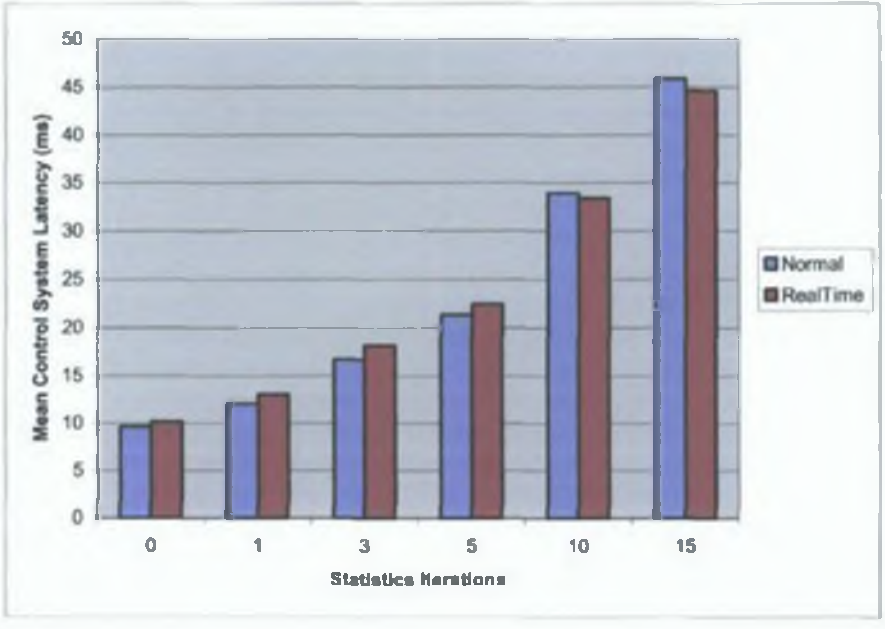


Figure 34: PC_WinNT_VB Mean Results without ActiveX and No Load

Statistics Iterations	Priority	Min	Mean	Max	Cpk	Standard Deviation (σ)	8σ	Mean + 4σ
0	Normal	124.00	141.79	354.00	-2.87	15.44	123.52	203.55
1	Normal	127.00	152.22	749.00	-0.77	62.21	497.68	401.06
3	Normal	131.00	162.21	946.00	-0.68	74.66	597.28	460.85
5	Normal	134.00	160.01	823.00	-0.84	59.78	478.24	399.13
10	Normal	145.00	177.49	952.00	-0.71	78.88	631.04	493.01
15	Normal	157.00	198.67	713.00	-0.82	77.47	619.76	508.55
0	RealTime	128.00	137.79	644.00	-1.63	26.27	210.16	242.87
1	RealTime	127.00	135.88	161.00	-9.86	4.29	34.32	153.04
3	RealTime	128.00	141.03	158.00	-10.40	4.23	33.84	157.95
5	RealTime	135.00	142.32	153.00	-14.99	2.98	23.84	154.24
10	RealTime	144.00	157.49	685.00	-1.81	27.29	218.32	266.65
15	RealTime	152.00	162.35	702.00	-1.81	28.29	226.32	275.51

Table 15: PC_WinNT_VB Test Results with ActiveX and no Load

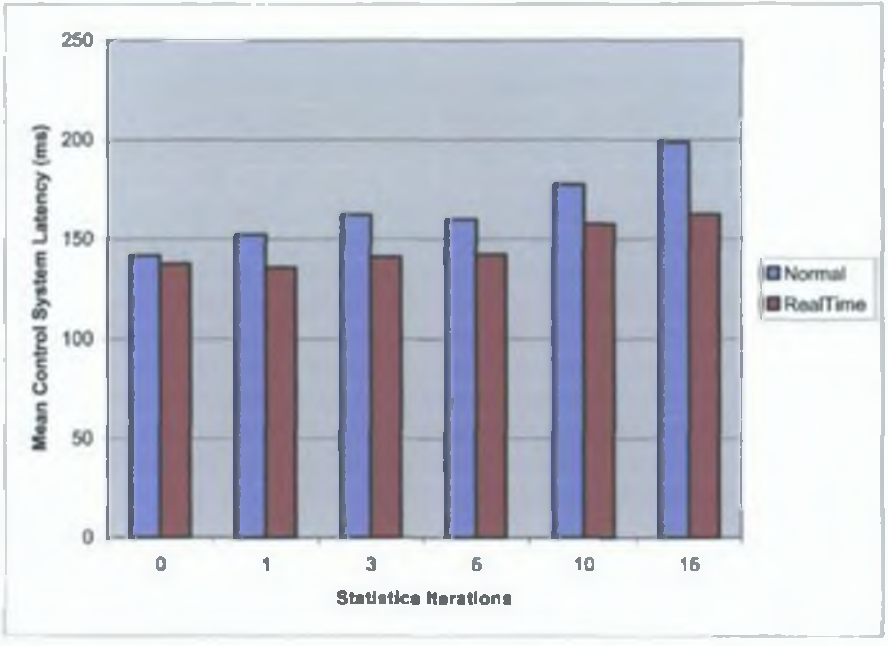


Figure 35: PC_WinNT_VB Mean Results with ActiveX and No Load

Statistics Iterations	% Load	Min	Mean	Max	Cpk	Standard Deviation (σ)	8σ	Mean + 4σ
0	0%	7.00	10.20	41.00	-0.15	2.61	20.88	20.64
1	0%	7.00	13.01	32.00	-0.61	2.19	17.52	21.77
3	0%	7.00	18.13	358.00	-0.35	8.64	69.12	52.69
5	0%	12.00	22.47	554.00	-0.34	13.15	105.20	75.07
10	0%	23.00	33.37	370.00	-0.84	9.66	77.28	72.01
15	0%	34.00	44.60	385.00	-1.06	11.21	89.68	89.44
0	50%	7.00	10.22	51.00	-0.15	2.72	21.76	21.10
1	50%	7.00	13.00	560.00	-0.15	9.18	73.44	49.72
3	50%	7.00	18.68	529.00	-0.32	10.13	81.04	59.20
5	50%	12.00	22.86	471.00	-0.47	9.88	79.04	62.38
10	50%	23.00	35.25	565.00	-0.60	14.66	117.28	93.89
15	50%	32.00	40.00	638.00	-0.72	16.00	128.00	104.00

Table 16: PC_WinNT_VB Test Results, RealTime priority without ActiveX and varying Load

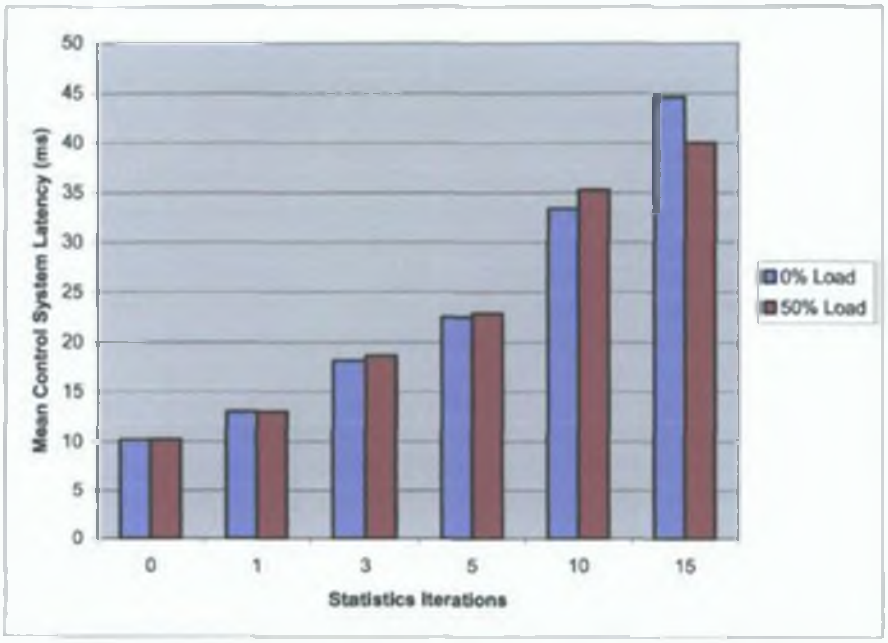


Figure 36: PC_WinNT_VB Test Results, RealTime priority without ActiveX and varying Load

4.6 RTOS_3S_ELAU

The RTOS_3S_ELAU test results are summarized in Table 17 and Figure 37 and demonstrate optimum performance over the full range of the statistics iterations. These tests demonstrate that there is virtually no deviation of the control system latency time on this controller. The maximum resolution of the Data Acquisition System was 1ms. Therefore the fact that the minimum was 7ms and the maximum was 8ms, may actually demonstrate that this controller was capable of providing deterministic control, with a reproducibility of better than 1ms, even under an extremely large number of iterations.

Statistics Iterations	Min	Mean	Max	Cpk	Standard Deviation (σ)	8σ	Mean $+4\sigma$
0	7.00	7.00	8.00	115.42	0.01	0.08	7.04
1	7.00	7.00	8.00	115.39	0.01	0.08	7.04
3	7.00	7.00	8.00	81.25	0.01	0.08	7.04
5	7.00	7.00	8.00	72.21	0.01	0.08	7.04
10	7.00	7.00	8.00	51.67	0.01	0.08	7.04
15	7.00	7.02	8.00	5.04	0.13	1.04	7.54

Table 17: RTOS_3S_ELAU_Test Results

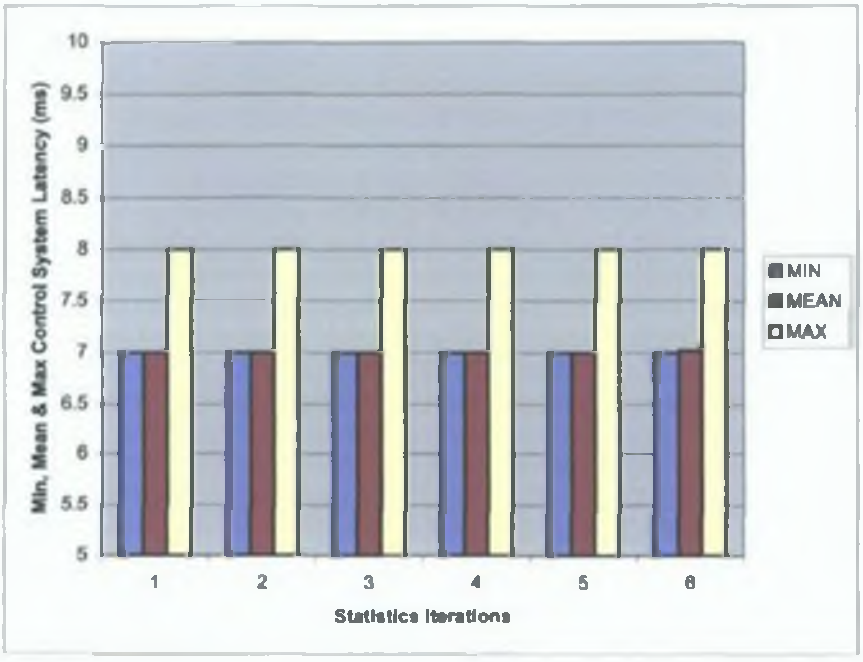


Figure 37: RTOS_3S_ELAU Min, Mean and Max Results

5 DISCUSSION

5.1 Determinism of the Fieldbuses

5.1.1 Determinism of AS-I

The results gathered in Table 11 clearly demonstrate that PLC_3S_IFM's AS-I network is capable of providing a deterministic response of better than 5ms when it has 31 missing slaves. It achieved an excellent C_{pk} of 41.56 and a Mean + 4σ value of 7.08ms. These results were achieved on an AS-I network with 31 missing slaves. These conditions represent the maximum number of errors that can exist on a AS-I network and thus simulates an extreme loading condition. Because the results on the PLC_3S_IFM were so conclusive and all AS-I masters have to pass the same stringent certification tests it was not necessary to incur the additional costs of inserting AS-I master cards on the PC and RTOS based systems [6]. Thus we confirm that the AS-I network is capable of providing a deterministic response even under extreme fault conditions.

5.1.2 Determinism of Ethernet

Even though it is not a requirement of this study to evaluate the determinism of Ethernet as a network, it is an inherent part of the PC_3S_Beckhoff control system under test and as such required some investigation. The Ethernet I/O of the PC_3S_Beckhoff was connected via a normal hub, on an Ethernet network configured as CSMA/CD in full duplex. This configuration represents a typical Information Technology (IT) configuration, but the best result which could be achieved was an unacceptable C_{pk} of –0.45 and Mean + 4σ value of 29.23ms. Thus this Ethernet network configuration is not suitable as a fieldbus network for a field level network above an AS-I network. Other configurations may be suitable but were not tested during this study [25], [26]. Special design considerations are also required for Ethernet, when it is implemented in industrial installations, if a deterministic response is required [60].

5.2 PLC

Both the PLC_Rockwell_AB and PLC_3S_IFM clearly demonstrated an extremely deterministic response. When no statistical iterations were required the PLC_Rockwell_AB achieved a C_{pk} of 14.60 and Mean + 4σ value of 7.2ms, while the PLC_3S_IFM achieved a C_{pk} of 41.56 and Mean + 4σ value of 7.08ms, clearly demonstrating that both platforms are excellent platforms for an AS-I network.

When the statistical iterations were introduced the limitations of the processing power of the PLC were highlighted. The results were extremely dramatic, and the PLC_3S_IFM was actually incapable of providing more than 3 iterations without actually going into a fault condition whereby it could not actually execute its user program. It is very important to note that even under these extreme loading conditions that the PLC achieved a deterministic response, with the standard deviation never exceeding 1.00. However it did not have the processing power to execute the iterations in an acceptable time limit for the AS-I network. When 1 statistical iteration was introduced, the PLC_Rockwell_AB's performance dropped to an unacceptable C_{pk} of –38.47 and Mean + 4σ value of 94.25ms. The PLC_3S_IFM's performance was only slightly better with a C_{pk} of -25.39 and Mean + 4σ value of 89.32ms.

These tests clearly demonstrate that the PLC is an excellent platform for providing a deterministic response but the designer must take into account the processing that is required for the user program or the overall control system latency may be increased to unacceptable limits for an AS-I network.

An item worthy of mention is the fact that the PLC_3S_IFM achieved a marginally better performance than the PLC_Rockwell_AB even though it is less than 1/20th of the cost, and has PROFIBUS DP and AS-I communications capability. Thus the PLC 'price versus performance ratio' has improved considerably in the last 20 years.

5.3 PC

5.3.1 PC_3S_Beckhoff

It may be possible for the PC_3S_Beckhoff to provide a deterministic response but not enough work was done in this area to clearly demonstrate this. What was clearly demonstrated is that the incorrect hardware selection and/or configuration of an Ethernet based PC_3S_Beckhoff system can seriously compromise the determinism of a system.

5.3.2 PC_WinNT_VB

This work clearly demonstrates that the Windows based operating system, without an RTE is totally unsuitable for providing deterministic control. The Standard Deviation gathered from these tests ranges from an unacceptable 2.52 to a dreadful 118.24. In extreme cases the Mean has exceeded 600ms and the Maximum has exceeded 964ms, which undoubtedly represents a control system failure.

It is possible to improve the performance of the PC_VB_APP by raising the priority of the process from 'normal' to 'realtime'. This is clearly demonstrated by the fact that under 100% loading both the Standard Deviation and Mean responses for processes with realtime priority can be as much as 40 times better than a process with normal priority. Even though this appears to substantiate Rockwell Automation's claims that hard real time extensions are not required [9], the response of a realtime process is not at all deterministic and Microsoft's advice not to use the Windows operating system for realtime control [33], [35], [36], [38], should be strictly adhered to. A major finding of this work was the effect of the use of graphing ActiveX within a VB application. Regardless of the priority of the process (normal or realtime), an ActiveX that refreshes the display caused a dramatic degradation of system performance and rendered the application incapable of achieving its control function.

5.4 RTOS

The performance of RTOS_3S_ELAU was excellent. Its Standard Deviation was an order of magnitude better than any of the other systems.
When 0 statistics iterations were required, the RTOS_3S_ELAU produced a C_{pk} value of 115.42, which is 7.9 times better than the PLC_Rockwell_AB, and 2.7 times better than the PLC_3S_IFM.

Even with 15 statistics iterations being calculated, the RTOS_3S_ELAU was capable of achieving a C_{pk} of 5.04, a standard deviation of 0.13ms and a Mean + 4σ value of 7.54. This C_{pk} of 5.04 is more than twice the value of C_{pk} that has been specified in this study as the requirement for an extremely suitable control system.

The RTOS_3S_ELAU provides an ideal performance for a deterministic network such as the AS-I network and when subjected to statistics iterations has the processing power to out-perform any of the other systems that have been put under test.

5.5 A Significant Observation

A significant observation, which can be made, is that in some instances the Maximum control system latency values exceed the Mean + 4σ values. +/- 4σ represents 99.94% of the population (as per Figure 25) [56]. If a test is to produce 1 sample, whose result falls outside of this population then 1 sample must be less than 0.06% of the total number of samples taken during this particular test. Thus the minimum number of samples required to produce 1 sample with a value outside the Mean + 4σ value is 1667 samples.

Table 15 outlines the test results on the PC_WinNT_VB for the control system latency with the graphing ActiveX activated and no BurnInTest load. By adding the sampling data to some of the Table 15 results a table such as Table 18 can be produced. Table 18 clearly demonstrates that for the majority of the tests the Maximum control system latency values exceed the Mean + 4σ values by a very large amount, but there are not enough samples for 0.06% of the samples to produce 1 sample. Thus the σ calculation, and Maximum control system latency must be investigated further to ensure that there is no error in the results.

Mean + 4σ	Max	Max - (Mean+ 4σ)	Samples	0.06% of N
203.55	354.00	150.45	427	0.26
401.06	749.00	347.94	398	0.24
460.85	946.00	485.15	359	0.22
399.13	823.00	423.87	372	0.22
493.01	952.00	458.99	329	0.20
508.55	713.00	204.45	286	0.17
242.87	644.00	401.13	409	0.25
153.04	161.00	7.96	446	0.27
157.95	158.00	0.05	429	0.26
154.24	153.00	-1.24	409	0.25
266.65	685.00	418.35	385	0.23
275.51	702.00	426.49	373	0.22

Table 18: Table 15 results and Sampling information

Figure 38 demonstrates a frequency distribution of a re-run of one of the tests contained in Table 15. When the frequency distribution is analysed it does not display a normal distribution. It consists of a number of normal distributions with rogue values. In this instance there was 1 sample in isolation, which produced a control system latency of 533ms. (Note: for the purpose of readability I have manually changed the number of samples with a latency of 533ms to 10. No other readings have been changed). This value of 533ms may be statistically insignificant, but could have catastrophic effects on a control system.

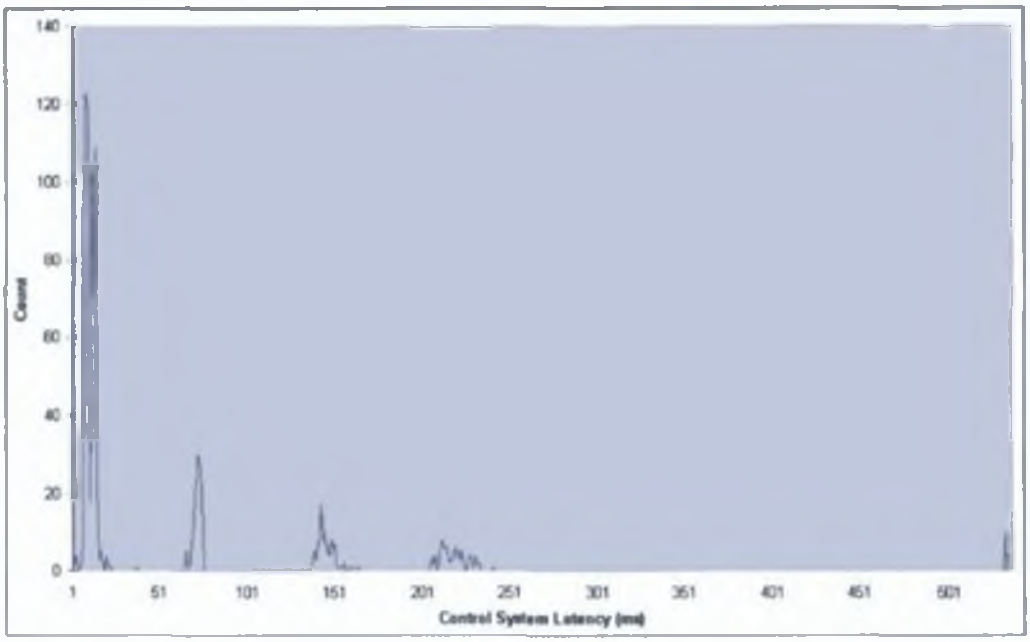


Figure 38: PC_WinNT_VB frequency distribution

Table 19 consists of the PLC_Rockwell_AB results from Table 10 and the sampling data. This data is a complete contrast to the data contained in Table 18. The only time that the Maximum control system latency exceeds the Mean + 4σ value is when there are an enormous number of samples and up to 10 samples could fall outside the +/- 4σ range. Figure 39 demonstrates a frequency distribution of a re-run of one of the tests contained in Table 10. This frequency distribution displays a normal distribution, with no rogue values.

Mean + 4σ	Max	Max - (Mean+ 4σ)	Samples	0.05 % of N
7.2	13.00	5.8	17281	10.37
94.25	93.00	-1.25	660	0.40
272.07	271.00	-1.07	225	0.14
449.99	449.00	-0.99	136	0.08
893.30	892.00	-1.3	68	0.04

Table 19: Table 10 results and Sampling Information

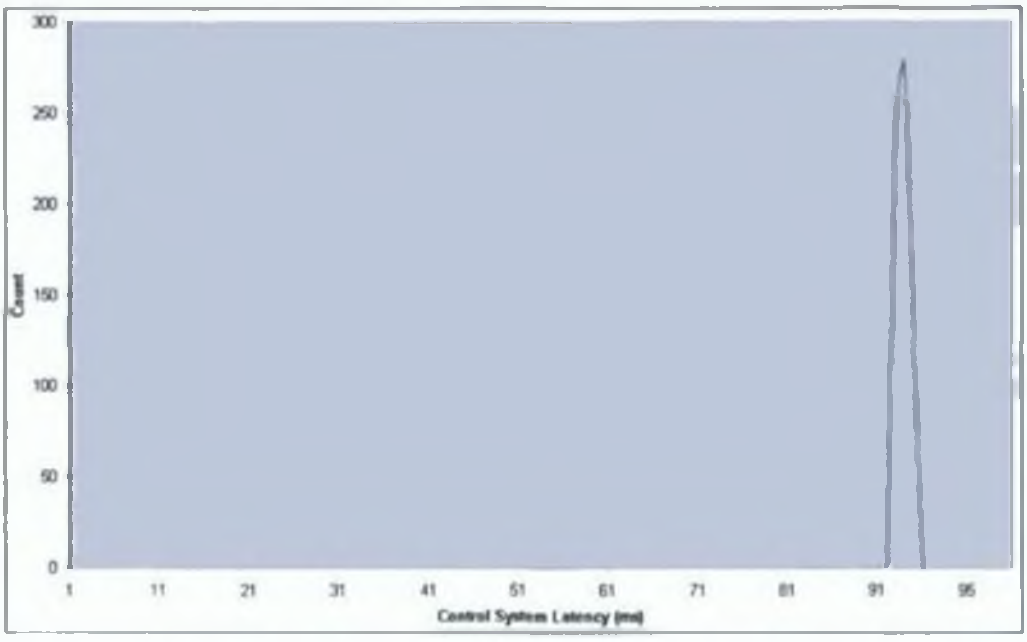


Figure 39: PLC_Rockwell_AB frequency distribution

All of the maximum values for the RTOS_3S_ELAU control system latencies, as outlined in Table 17, exceed the Mean + 4σ value. This would tend to point to the danger that the RTOS_3S_ELAU may demonstrate a non normal distribution in the same manner as the PC based systems. On closer inspection it becomes apparent that this is due simply to a rounding error. The interval bin has a resolution of 1ms, while the Mean + 4σ value is calculated to an accuracy of 2 decimal places. Thus we can state that the RTOS_3S_ELAU displays a normal distribution, in the same manner as the PLC control systems, and in direct contrast to the PC control systems.

5.6 Further Work

Unfortunately every project has a limited scope, duration and resource limit. If extra resources had been available it would have been useful to quantify the determinism of the AS-I network when the maximum number of correctly working slaves are connected [6], even though the literature survey clearly states that the AS-I network is deterministic.

An extremely interesting area, which was outside the scope of this study, is the evaluation of determinism across various different Ethernet configurations. A lot of information is available from Ethernet component providers, demonstrating how determinism can be achieved but quantifying how each of the potential configurations performs would undoubtedly produce useful results.

The tests that were performed on the PLC were based on only 2 PLC's that are microprocessor based. There are many PLC's now available, such as the Allen Bradley ControlLogix, that are actually PC based [37]. It is extremely likely that platforms such as these are capable of providing the processing power to provide a deterministic response, even under extreme loading such as the statistics iterations. Future work should expand these tests to evaluate more PLC, PC and RTOS platforms.

The effect of the ActiveX in the PC_VB_APP was extremely dramatic. It is not possible at this stage to state whether it was due to inter-process switching or possibly graphics requirements. A lot of work could be concentrated in this area to determine exactly what caused this phenomenon.

The RTOS_3S_ELAU clearly out-performed all of the other systems under test. It would be extremely interesting to design a test capable of stress testing the RTOS_3S_ELAU to quantify exactly how much better it is than the other systems.

5.7 Things that could have been done differently

In this study the ELAU MaX4 was used as the Data Acquisition System. In this configuration it was capable of providing a resolution of 1ms. In hindsight higher accuracy results could have been achieved by using the National Instruments Data Acquisition card as per the GMPTG tests.

6 CONCLUSIONS

After evaluating the determinism of the AS-I network, PLC systems, PC systems and RTOS systems the following conclusions can be drawn:

The following conclusions can be drawn from the study of the fieldbus systems:

- The AS-I network, even under extreme fault conditions of 31 faulty slaves is deterministic, and has a worst-case response of 5ms
- The PROFIBUS DP network is also deterministic and will not introduce any latency, if used as a field level network above an AS-I network.
- An incorrectly configured Ethernet network is extremely non-deterministic
- The literature states that a correctly configured Ethernet network is deterministic, but this configuration was not tested during this study

Both of the PLC systems, which were tested, PLC_Rockwell_AB, and PLC_3S_IFM, demonstrated very similar results, thus the following conclusions can be drawn for both PLC systems:

- The PLC systems are extremely deterministic
- The PLC systems control system latency exhibits a normal frequency distribution
- The instruction set is very limited
- The PLC systems do not have sufficient processing power for a lot of iteration as simulated by the statistics iterations in the tests
- Even though the PLC systems are deterministic, due to its lack of processing power, it is not suitable as a control system for applications where a lot of iteration is required
- The PLC systems are suitable control systems for implementation with an AS-I network, if the user program does not require a lot of processing

The conclusions from the PC systems is limited to the PC_WinNT_VB. This is due to the fact that more tests would need to be done on the PC_3S_Beckhoff before accurate conclusions could be drawn. With the PC_WinNT_VB the following conclusions can be drawn:

- The PC_WinNT_VB system is extremely non-deterministic
- The PC_WinNT_VB system control system latency exhibits a non-normal frequency distribution, with many observations, which do not belong to the curve
- Modifying the process priority from Normal to RealTime does not significantly improve the performance of the system
- The PC_WinNT_VB is not suitable for implementation with an AS-I network regardless of the user program

From the RTOS system, which was tested (RTOS_3S_ELAU), the following conclusions can be drawn:

- The RTOS system is extremely deterministic
- The RTOS system has more than enough processing power for applications, which require a lot of iteration as simulated by the statistics iterations in the tests
- The RTOS system was the only system, which was capable of providing a suitable control system latency for an AS-I network when the maximum number of statistics iterations calculations were being performed

Also statistics must be used prudently. Assumptions should not be made that systems follow a normal distribution. This can only be determined after testing has been completed. A very useful rule of thumb which was identified is that if the number of samples is less than 1667 and the maximum value exceeds the Mean + 4σ, then in all likelihood the system does not have a normal distribution.

7 BIBLIOGRAPHY

[1] http://www.industrialnetworking.co.uk/mag/v7-1/andy.html

[2] http://www.chipcenter.com/circuitcellar/september99/c99wp2.htm

[3] http://cthulhu.control.com/alist/archive.old/msg01959.html

[4] http://msdn.microsoft.com/library/default.asp?url=/library/en-s/dnwbgen/html/msdn_realtime.asp

[5] RadiSys Corporation. "Determinism and the PC Architecture - Applying PC Hardware to Real-Time, Applications Technical Paper". July 1998.

[6] Kriesel W, Madelung O, "AS-Interface, The Actuator-Sensor-Interface for Automation". Hanser. 1999.

[7] Smith, C.A, Corripio, A.B. "Principles and Practices of Automatic Process Control". 1997.

[8] Shinskey, F.G. "Process Control Systems, Application, Design, and Tuning". 1998.

[9] Rockwell Automation. "White Paper, Using the Windows NT Operating System for Soft Real-time Control – Separating Fact from Fiction". 1998.

[10] Radionics. "Industrial Components Catalogue". March 2003 – February 2004.

[11] Warnock, I.G. "Programmable Controllers operation and application". 1988.

[12] Keyence. "Sensors Vision and Measurement". 2003.

[13] Omron. "Sensing & Safety general catalogue". 2003.

[14] PROFIBUS Nutzerorganisation e.V. PNO. "PROFIBUS TECHNOLOGY AND APPLICATION, System Description". October 2002.

[15] Polsonetti, C. "INDUSTRIAL ETHERNET PROTOCOLS: THE NEXT BATTLEGROUND". http://ethernet.industrial-networking.com/articles/i12protocols.asp. November 2002

[16] www.modbus.org

[17] www.odva.org

[18] www.profibus.com

[19] Kriesel W, Gibas P, Riedel M, Blanke W. "Feldbus als Mehrebenenkonzept. messen, steuern, regeln" Berlin 33(1990)4, S. 150-153

[20] Shiff, A. "AS-I: Technischer Losungsansatz; Vortrag auf dem Fachgesprach "Bussysteme" des VDI/VDE-Technologiezentrum Informationstechnik". Berlin im Februar 1991 (als Sprecher des Lenkungsausschusses des AS-I-Konsotriums)

[21] Madelung, O.W. "Acktuator Sensor Interface. Grundforderungen an ein neues Buskonzept". Sensor Report (1992) Heft1.

[22] Madelung O.W. "AS-I ersetzt nicht nur den Kabelbaum. Das einfache Feldbussystem fur binare Peripherieelemente". Electronik plus. Heft „ Feldbusse. Franzis-Verlag, Munchen 1992.

[23] Acromag. "Technical Reference, Introduction to ProfibusDP". 2002.

[24] Hirschmann Network Systems. "White Paper, Real Time Services (QoS) in Ethernet Based Industrial Automation Networks"

[25] Breyer R., Riley S. "Switched Fast and Gigabit Ethernet, Third Edition, Understanding, Building and Managing High-Performance Ethernet Networks". 1999.
[26] Spurgeon, C.E. "Ethernet, The Definitive Guide". 2000.
[27] Warnock, I.G. "Programmable Controllers, Operation and Application". 1988.
[28] Allen Bradley. "PLC-5 Programmable Controllers, Instruction Set Reference". November 1998.
[29] Allen Bradley. "SLC 500™ Instruction Set". April 1999.
[30] Beckhoff TwinCAT®. "The Windows Control and Automation Technology. PC Control Introduction" March 2001.
[31] Intel. "Expanding Moore's Law, The Exponential Opportunity". 2002
[32] http://www.microsoft.com/windows/WinHistoryDesktop.mspx
[33] Microsoft Corporation. "Real-Time Systems and Microsoft Windows NT". June 29, 1995.
[34] Solomon, D.A. "Inside Windows NT". Microsoft Press. 1998.
[35] Microsoft MSDN Library. "Article Q10280, Realtime Priority Applications and Windows NT". May 9, 1997.
[36] Microsoft MSDN Library. "Article Q94265, Windows NT and Real-Time Operating Systems". May 6, 1997.
[37] Rockwell Automation. "Allen Bradley SoftLogix5800 Systems User Manual". August 2002.
[38] Microsoft MSDN Library. "Article Q10280, Realtime Priority Applications and Windows NT". May 9, 1997.
[39] Chrysler, Ford, G M. "Requirements of Open, Modular Architecture Controllers for Applications in the Automotive Industry, Version 1.1". December 13,1994.
[40] GMPTG. "Open Modular Architecture Controls at GM Powertrain, Technology and Implementation, Version 1.0". May 14 1996.
[41] Frampton, N., Tsao, J. Yen, J. "Hard Real-time Extensions of NT© Evaluation Report, Test Plan and Phase 1 & 2 Results". April 17, 1998.
[42] RadiSys Corporation. "Determinism and the PC Architecture - Applying PC Hardware to Real-Time, Applications Technical Paper". July, 1998.
[43] RadiSys Corporation. "Real-Time OPC Utilizing INtime to Implement Deterministic OPC Servers, White Paper". April, 1998.
[44] RadiSys Corporation. "INtime Interrupt Latency Report, Measured Interrupt Response Times, Technical Paper". November, 1998.
[45] RadiSys Corporation. "Windows NT Thread Latency Measurements Measured Response Times for the REAL_TIME_PRIORITY_CLASS Technical Paper". December, 1998.
[46] Radisys Corporation. Fischer, P. "Reliable Control with Windows NT". 1998.
[47] Radisys Corporation. Fischer, P. "Distributed Real Time Systems with Windows NT and INTime". 1998.
[48] Nematron. "PC-Based Control Technology, An In-Depth Look". 2000.

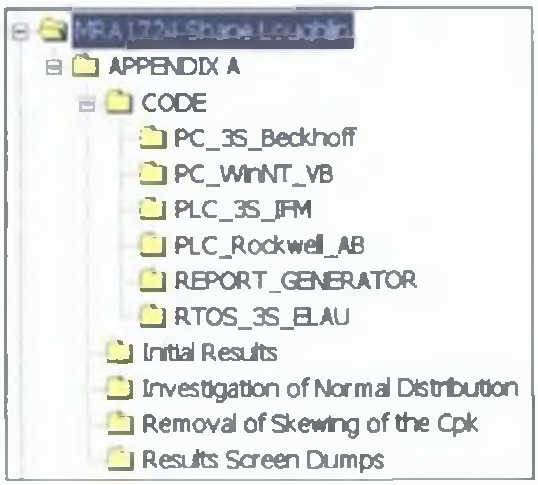
[49] VentureCom, O'Keefe, J.A.. "Venturecom RTX™ Enabling Microsoft® Windowst® Windows XP and Windows XP Embedded for Hard Real-Time". 2002.
[50] 3S. "Product Brochure, 3s – Smart Software Solutions GmbH". 2004 .
[51] PLCopen. "IEC61131-3: a standard programming resource". March 2004.
[52] http://www.arcweb.com/research/auto
[53] www.automation-alliance.com
[54] http://www.passmark.com/products/bit.htm
[55] Balena, F. "Programming Microsoft Visual Basic 6.0". 1999.
[56] Caplen, R.H. "A practical Approach to Quality Control". 1988.
[57] Loughlin, S. "A hollstic approach to Overall Equipment Effectiveness (OEE)". IEE Computing and Control Engineering, December / January, 2003 / 2004.
[58] BS 5700. "1984 Guide to process control using quality control chart methods and cusum techniques". 1984.
[59] http://www.vdisoft.com
[60] Lounsbury, B., Westerman, J. "Ethernet: Surviving the Manufacturing and Industrial Environment White Paper". 2001.

8 APPENDIX A – Source Code and Test Results

Appendix A contains all of the source code and test results used during the course of this project. Appendix A is supplied in electronic format on the attached Compact Disk (CD) labelled 'MRA1724 Shane Loughlin' where MRA1724 is the HETAC assigned registration number for this project. The CD contains the following sub directories:

- 'CODE' contains all of the test code that was implemented on each of the control system platforms.
- 'Initial Results' contains an example of the first results that were produced.
- 'Investigation of Normal distribution' contains the data that was used to investigate the normal distributions on the PC_WinNT_VB and the PLC_Rockwell_AB.
- 'Removal of Skewing of the C_{pk}' contains the diagrams that were used to depict how the skewing of the C_{pk} was removed.
- 'Results Screen Dumps' contains all of the actual screen dumps from the Report Generator application, which was used to collate the data from the control platforms during the execution of the tests.

A diagram of the layout of the directories on the CD is shown below:



An electronic copy of this thesis is contained in the root directory of the CD and is labelled 'MRA1724 Shane Loughlin.pdf'

The following software is required to view, program or execute the code in the CODE sub-directories:

- PC_3S_Beckhoff – Beckhoff TwinCat PLC Control V2.9.0
- PC_WinNT_VB – Microsoft Visual Basic 6.0
- PLC_3S_IFM – CoDeSys V2.2.5.2
- PLC_Rockwell_AB – RSLogix 5.20.10
- REPORT_GENERATOR – Microsoft Visual Basic 6.0
- RTOS_3S_ELAU – ELAU EPAS-4 Version 14.02

NOTE: These versions of software are the versions that were used during the course of this project. It is reasonable to assume that these applications will be upwardly compatible to newer versions of the same applications.