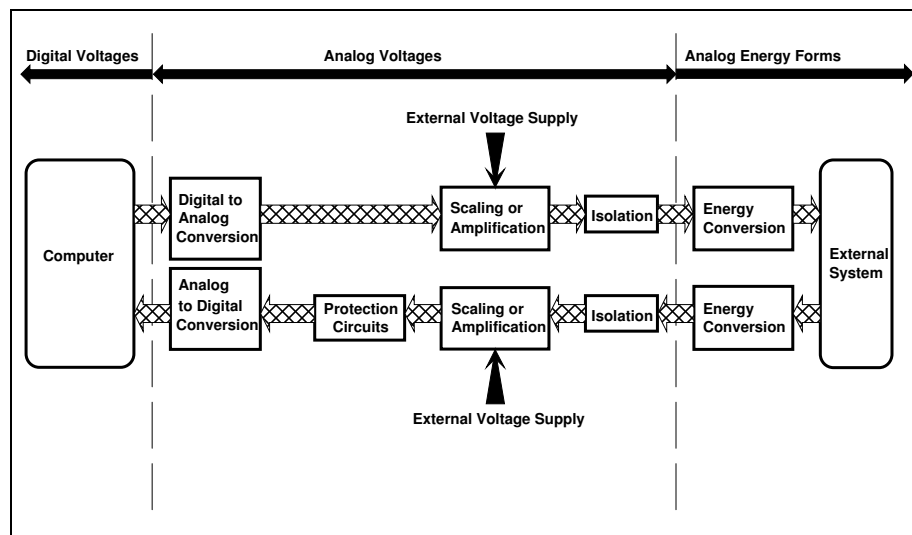

Chapter 1

An Overview of the Computer Interfacing Process

A Summary...

A qualitative description and overview of the entire computer interfacing process (shown below). The analog and digital computing domains and the basis of digital circuit design. Steps in computer interfacing including hardware design phases and software issues.



1.1 General Issues - Analog and Digital Computation

It is difficult to know whether the modern digital computer, and its Von Neumann architecture, were ever intended to interface to the real world. A computer is, by definition, a "reckoning machine" and certainly its original designers went to great lengths to make it into a device that could carry out repetitive computations and provide a limited amount of human reasoning. The analog computer, on the other hand, has always been closely associated with the control of physical systems, but has never been able to provide the reckoning ability that we associate with the digital computer. Analog computers no longer play any significant role in modern engineering, and so, this book is predominantly about digital computers and the problems that we face in connecting them to engineering systems.

It is interesting to note that when we use digital computers as islands of intelligence (that is, on their own), we find that their capabilities are only restricted by our own reasoning ability (that is, our ability to generate software) and by the speed at which computers can carry out the reasoning that we have instilled via our software. However, when we wish to interface computers to the real world (so that they can use their programmed reasoning to control a physical system) we find that our reasoning needs to be supplemented with an understanding of electronics, physics and engineering design principles before we can generate sensible solutions.

It is altogether likely that any computer programmer, with no knowledge of computer architecture or electronics, could create a working computer control system. Certainly, there are a sufficient number of commercially available, "black-box" solutions to assist in interfacing computers to external systems. The problem with using such solutions, without a proper understanding of the design principles behind them, is that sooner or later the seemingly minor problems that arise (system instability, unwanted spurious signals, irregular behaviour, etc.) become insoluble. This book has not been written with the intention of by-passing the ready-made solution, but rather, with the intention of helping you to understand the basis of these solutions and the problems that arise when using them.

The first issue that really needs to be addressed is that of the digital and analog computing domains. In all our "Newtonian" time-frames, the world is essentially analog in nature. Physical quantities do not change from one energy level to another in zero time - there is normally a continuous transition from one state to another, rather than a quantum variation. One may well ask why, if the world is essentially analog, have we chosen to discard the concept of analog computing and replace it with digital (quantum) computing. There are many reasons why analog computers were discontinued in the 1970s. These include:

- Accuracy
- Size
- Power consumption
- Cost.

Underlying all the problems in analog computing is the issue of representing quantities accurately. Consider an analog computer that is required to take in two numbers (between zero and ten) and add them together to achieve a given result. How is this achieved? We could represent each of the inputs and outputs with a voltage and use a circuit to electrically add the inputs to provide the required output. This is shown in Figure 1.1.

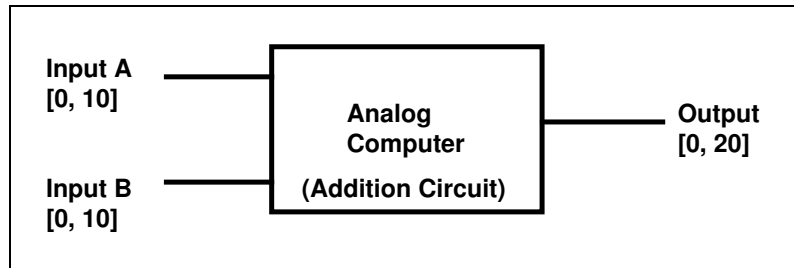


Figure 1.1 - Simple Analog Computation - Addition

From Figure 1.1, we would assume that if Input A is equal to 5 volts and Input B is equal to 2 volts, then the output would be equal to 7 volts. What happens however, if:

- Input A = 5.001322447 volts
- Input B = 1.999933821 volts ?

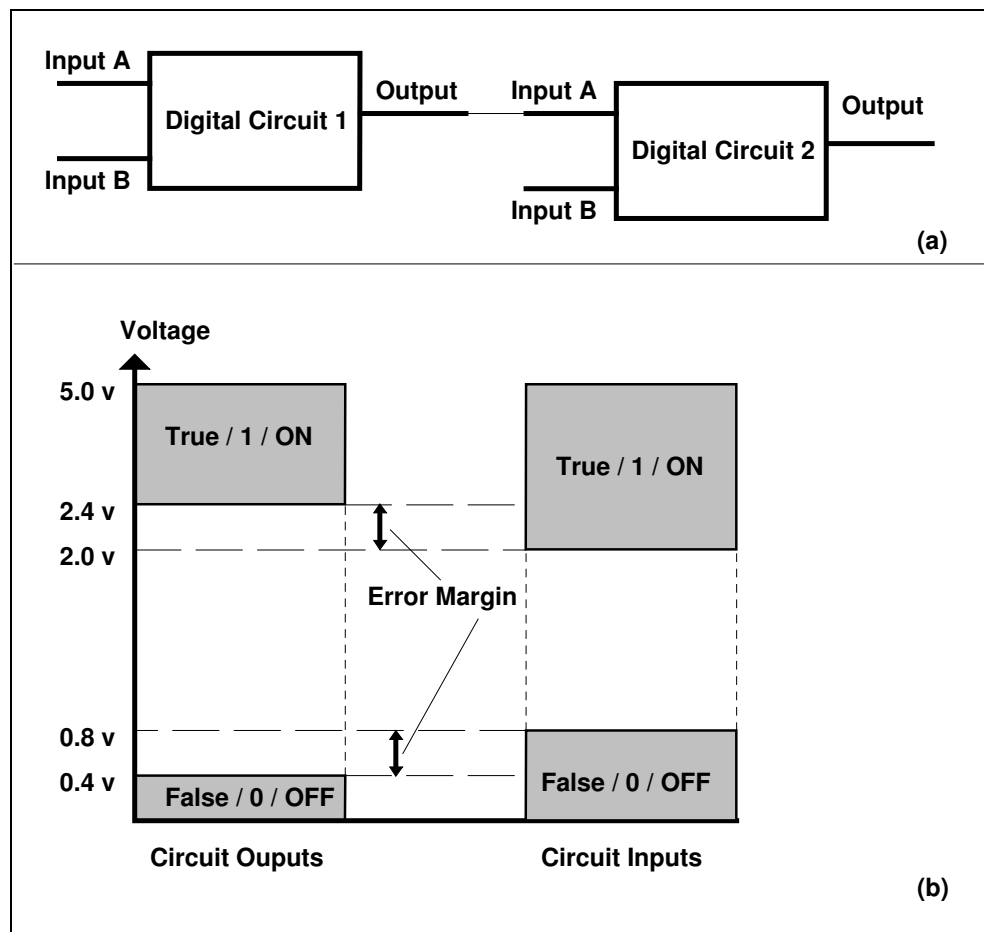
The answer to this question really depends upon how accurately we can design and fabricate our analog addition circuit. In engineering, we know that accuracy in setting or measuring energy levels normally equates to higher complexity and higher cost. This is certainly true in electronic circuits.

The alternative to generating accurate (expensive) circuits is digital computing, where we only allow electronic hardware to represent the binary numbers zero and one. Digital circuits allow us to carry out both arithmetic and a limited amount of human reasoning (tautology). The reasoning component is achieved by using zero to represent "false" or "off" and one to represent "true" or "on". The basis of digital circuits, and hence digital computing, is that voltage accuracy must not be permitted to dominate circuit design. As long as we can achieve voltages within certain ranges, then we can represent the only two numbers that we need to handle in digital computing.

The major design effort in the fabrication of digital semiconductor devices is therefore oriented towards achieving:

- High circuit speeds
- Small component sizes / High Component Density
- Low power dissipation
- Low circuit fabrication costs.

Figure 1.2 (a) shows two, cascaded digital circuits, each with two inputs and one output. Figure 1.2 (b) shows the allowable output and input voltage ranges for each of the circuits. An error margin exists between the output of one circuit and the input of the subsequent circuit, so that minor voltage variations that may occur do not affect the meaning of the information represented.



*Figure 1.2 - (a) Cascaded Digital Circuits
(b) Allowable Input and Output Voltage Ranges for Transistor to Transistor Logic (TTL) Digital Circuits*

The actual digital voltage ranges shown in Figure 1.2 (b) correspond to a family of digital circuits known as Transistor to Transistor Logic or TTL. This is the oldest of the widely used digital circuit technologies (dating back to the 1960s) and is important because most modern digital circuits are still designed with the same allowable input and output voltage ranges.

One of the major objectives of this book is to help you to come to terms with the design of modern digital computer systems, so that you can understand why it is so difficult to interface them to the outside world. The strategy chosen in order to achieve this is to begin with an overview of the role of computers in the industrial arena, so that you may gain some insight into the range of performance attributes required by modern computers. The next stage in the process is to introduce (or rekindle) the basics of modern electronic circuits, including transistors, thyristors and so on. This serves two purposes. Firstly, it will help you to understand how both analog and digital circuits are designed and the limitations of those circuits. Secondly, it will introduce you to the basic elements used in interfacing computers to the outside world.

As we progress through this text, we will examine how digital circuits are designed and how they are joined together to carry out binary arithmetic and limited "human" logic functions. Once we understand these concepts, we will move on to the development of digital storage devices (memory) and the digital state machine concept - the heart of modern microprocessor technology. We will then see how all the basic elements are brought together via an address and data bus structure to create what we now understand to be the modern digital computer.

This text book does not concern itself with the specifics of any one computer system or computer architecture. There are a number of reasons for this. Firstly, there are so many different microprocessor and computer system architectures that any reasonable treatment of their architecture would make this book unwieldy and out of date (by the time the information was compiled). Secondly, there are so many technical side-issues involved in the design of a commercial computer system that they tend to cloud the reader's understanding of the more important basic issues. Despite many manufacturers' claims to the contrary, this author believes that most modern computer systems are still only variations on a central theme. It is that theme which we shall explore in this book. You should then have enough knowledge to move on to the task of examining and understanding the specifics of a particular architecture in your own right. The same principle applies in terms of the treatment of microprocessors and Digital Signal Processors (DSPs) - they are both lumped together under the generic term of microprocessor. The assumption is that the DSP is a specialised form of microprocessor that has been optimised for low level control functions. Although most DSPs have, what is referred to as a "Harvard" architecture (rather than Von Neumann), in this book they are simply treated as a variations upon the central, Von Neumann theme.

1.2 Interfacing Computers Via Hardware

When we talk of interfacing computers to external systems, we are generally endeavouring to create closed control loops that enable a digital computer to generate (compute) some desired driving force, based upon the feedback obtained from the outside system. Sometimes, of course, we don't need a closed loop. This is particularly true if we only use a computer based device as a data logging or monitoring system, or if we use the computer to drive a system independently of the feedback (ie: as an open loop controller). However, regardless of the application, the same sorts of issues need to be addressed as a result of the incompatibilities between the "Newtonian-analog outside world" and the "digital computing world". The issues are shown schematically in Figure 1.3, together with the book chapters (shown in braces { }) that address them. You will see this diagram in various forms throughout this text, particularly on the front page of each chapter, where it will be used to remind you of the topics covered within that chapter. The topics most closely related to that chapter will be shown in bold lines and bold text, while the remainder will be shown in dotted lines and plain text.

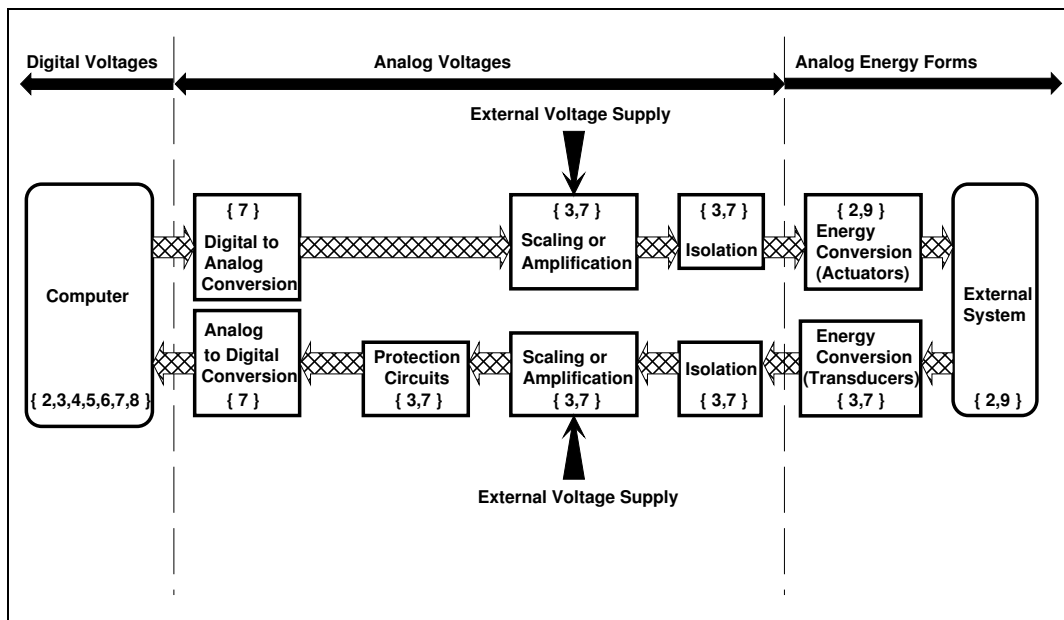


Figure 1.3 - Interfacing Digital Computers to External Systems

When observing Figure 1.3, the most important point to understand is that modern computers will only respond to incoming digital voltage signals and will only generate outgoing digital voltage signals. The levels of these digital voltages are normally in the order of those shown for TTL circuits (in Figure 1.2 (b)), although the actual levels depend upon the specific architecture of the digital circuits within the computer.

Modern computers do not respond to energy represented in any form other than voltage - generally, not even current. However, Figure 1.3 shows the generalised "outside world" as an external system which is propelled by continuous (analog) energy sources (mechanical, electrical, thermal, etc.) and feeds back analog signals that represent some form of energy (eg: temperature, voltage, current, pressure, stress, strain, position, velocity, acceleration, etc.) which is, in general, not a voltage. The closed loop of Figure 1.3 clearly represents some problems in terms of interfacing the small digital voltage signal world of the computer to the analog energy world outside. Signals fed back from the external system to the computer need to be:

- (i) Converted from their original form to voltage
- (ii) The raw voltage levels may need to be isolated from the external system
- (iii) The raw voltage levels need to be scaled to a level compatible with the needs of computer circuits
- (iv) In the event that the scaled voltages may suffer from unwanted spikes (as a result of spikes in the energy levels received from the outside world), protection circuits need to be considered to prevent high signals from damaging the computer
- (v) The scaled voltages need to be converted into a digital form.

The other major problem with connecting digital computers to the outside world is that the circuits within the computer are not only low in energy consumption, but also incapable of providing a large amount of energy. We know that signals within the computer are represented by digital voltages, somewhere in the order of those shown for TTL circuits in Figure 1.2 (b). However, we also need to appreciate that the major limitation of those circuits within a computer is that they are unable to provide large amounts of current. Typically, a digital circuit can provide less than a milli-Amp of current. This means that digital circuits can only provide a few milli-Watts of power. The actual amount depends upon the type of digital circuit but, in any event, is generally much less than the energy normally required to drive many external systems.

Consider the case where a computer is required to control a power station. The external system is a generator driven by a rotating turbine. The signals fed back to the computer may include items such as turbine speed, output voltages, etc. Based upon these signals, the computer needs to provide a controlled driving force that will cause the turbine to rotate at a given speed - clearly the computer is not going to supply the Mega-Watts of mechanical power that are required to rotate the turbine. Rather, the "driving force" produced by the computer is a very low level electrical signal that is used to drive actuators, valves, etc. that control the flow of steam or water to the generator turbine. The problem is then how to convert the computer's controlling signals into some form of energy that can ultimately drive the external system.

Assuming then, that the computer circuits are unable to provide the required driving energy for an external system, there are a number of interfacing steps that may need to be carried out in order to achieve the required outcome. These include:

- (i) Conversion of the computer's internal digital voltages to analog form
- (ii) Amplification of the analog voltages to higher energy levels
- (iii) Isolation of the computer circuits from the external system
- (iv) Conversion from the analog voltage form to the required final energy (mechanical energy, thermal energy, etc.).

Chapter 3 of this text actually has two objectives. The first is to introduce you to the basic electronic elements so that you can understand how these are used in order to create modern digital computers. The second objective of the chapter is to introduce you to the basic electronic elements used in the computer interfacing process. One of the key elements in all digital and analog circuits is the transistor. Chapter 3 shows how the transistor can be used to create basic digital circuit elements and also how it can be used as an analog amplifier. In both cases, the transistor is shown to be a device whose function is to control the flow of energy from a fixed, external electrical energy supply in such a way that the outputs of transistor circuits are related to the inputs by some predefined characteristic. This concept is fundamental to both computing and interfacing.

Following an examination of the basic elements of modern computer architecture, in Chapters 4, 5 and 6, Chapter 7 continues the hardware interfacing theme by examining the basic elements of closed-loop control systems. As you may gather from Figure 1.3, Chapters 3 and 7 are closely tied together. Chapter 9 completes the study of interfacing hardware by examining one of the classic mechatronic systems and benchmark control problems - the servo motor. The purpose of this chapter is to bring together, in hardware and software, the sort of elements that have been introduced during the course of this book. Electromagnetic actuators (motor drives) have been singled out from other mechatronic devices for special attention in this text. This is because of their enormous importance in modern industrial systems, most notably robots, CNC machines, transfer lines and flexible manufacturing systems (FMSs). As a result of their widespread application, inherent complexity and the fact that they embody the concepts of mechatronics and interfacing as a whole, they have been allocated an entire chapter.

One major issue that isn't dealt with in this book is the hardware interfacing of computers via networks. This is an important subject in its own right and has therefore been dealt with in the complementary text book, "Data Communications and Networking for Manufacturing Industries".

1.3 Interfacing Computers Via Software

The problems associated with interfacing computers to external systems are often accentuated when the software development phase commences. It is then that the designer realises the limitations of the hardware and has to make the decision of whether to continue on the existing course or perhaps revise the original hardware design altogether.

In most applications where computers are interfaced to external systems (particularly mechatronic systems), the critical issue is whether incoming signals can be processed quickly enough and, in a closed loop, whether outputs can be generated quickly enough, to ensure the stable operation of the combined system. The most efficiently coded software cannot overcome inadequacies of the processing and interfacing hardware, but poorly written software can certainly exacerbate hardware problems. The objective therefore, in most time-critical applications, is to develop software that can carry out a given task within a specified time. This is referred to as real-time software development and has an added dimension of difficulty that isn't found in many other software development tasks - that is, the computational time factor.

There are a number of features which are common to most pieces of software related to computer interfacing. These include:

- (i) Time-critical input/output routines that read incoming data from interfacing hardware and write outgoing data to interfacing hardware
- (ii) Control routines that process incoming data through some set of algorithmic or decision making criteria in order to generate an output response
- (iii) User input/output routines that enable a system user to interact with the computer and, thereby, the external system to which it is interfaced.

These elements are shown schematically in Figure 1.4, which needs to be studied for a few moments in order to understand the interaction between the various components. Of particular significance is the so-called "operating system" of the computer control system, which is a piece of software that acts as a "shell" in which all the other software elements execute. In the majority of applications, the operating system will be a commercially produced piece of software. However in low level environments, such as application-specific microprocessor controls, the system designer may write a very limited form of operating system to achieve a given task.

An operating system is not only a shell in which software can execute but it is more importantly a manager that controls each program's access to resources such as disk-drives, screens, memory locations, etc. We will examine the important role of the operating system in some detail in Chapter 8, before we move on to other issues of software development and software selection.

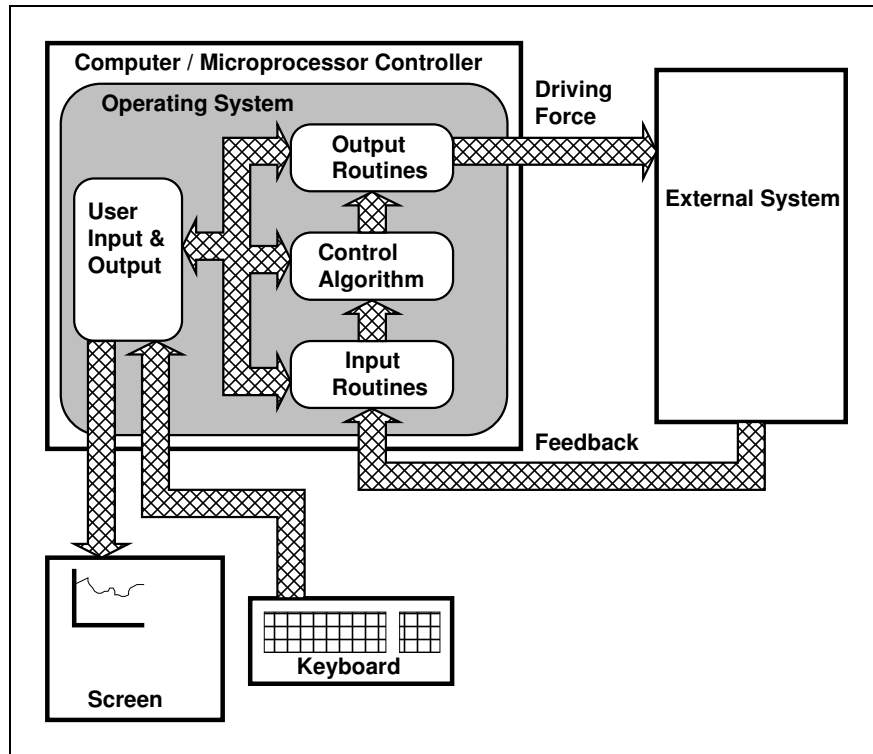


Figure 1.4 - The Computer Interfacing Software Development Process

Attempting to provide the specifics of how pieces of software, such as input/output routines, etc. should be written is akin to telling people how long to cut a piece of string - it all depends upon the specific application. More importantly, the software development strategy depends upon the specifics of the functions provided by the operating system itself.

The basic software routines illustrated in Figure 1.4, although common, are not universal to all applications. For example, the user input/output routines sometimes don't exist on a very low level microprocessor control system whose only purpose is to carry out some predefined task whenever activated. Output routines are not required in data logging applications and inputs are not required in open-loop control systems. However, in as much as both these tasks are common to most mechatronic control systems, they will be reviewed together in Chapter 8.

Given that there are a wide range of applications in computer interfacing, the issue arises of how we can tackle the subject of software development within the confines of a single book or chapter. The answer is that we clearly cannot cover all the issues involved in software development. More importantly, it must be remembered that this book has not been written with the intention of superseding the countless, existing text books on programming languages and packages. In this book, our objective is to examine the process of selecting development platforms and packages to achieve a set of criteria.

Some years ago, one might have argued that the selection of software development platforms was a trivial task - simply use low-level assembly language for time-critical tasks and high level languages (such as C or Pascal) for user input/output and common control algorithm development. However, this decision is no longer clear cut as a result of the introduction of "windows" type operating system environments and so-called "software development engines" (such as spread-sheets, databases, etc.). These are over and above the novel, 1980s attempts at computer programming, including neural networks, expert systems, etc. As we go through Chapter 8, we shall examine the task of software development with these new tools in mind, so that we can establish a decision making process for the selection and design of software.

At the end of Chapter 9, we will see how hardware and software can be brought together in the design of the traditional servo motor control, implemented via both analog and digital techniques.

