THE USE OF MICROPROCESSORS IN THE CONTROL OF CHEMICAL PLANT WITH SPECIAL REFERENCE TO THE USE OF DISTRIBUTED PROCESSORS

A Thesis submitted for the degree of

Doctor of Philosophy

by

Azman Firdaus Shafii

Department of Chemical Engineering

University of Aston-in-Birmingham

April 1983

THE USE OF MICROPROCESSORS IN THE CONTROL OF CHEMICAL PLANT WITH SPECIAL REFERENCE TO THE USE OF DISTRIBUTED PROCESSORS

A. F. SHAFII Ph.D. APRIL 1983

SUMMARY

The advent of the microprocessor has created an enormous impact on both society and industry. In particular, it has provided the technological basis for far-reaching changes in industrial control and automation where attention is focussed on the observable trends in applying microprocessor-based systems to improve control system design and integrity. This has also led to a renewed interest in the implementation of modern digital control policies.

The research described here is concerned with the development and use of a linked twin processor system comprising an 8-bit Motorola M6800 microcomputer and a 16-bit Honeywell H316 minicomputer. The linked facility provides for a wide range of OFF- and ON-LINE data processing activities including interactive real-time data acquisition and control of chemical plants. The system can support two computer users, with independent on-line sampling frequencies and control configurations.

Two major software packages have been developed. The first comprises two real-time Executives, one for each processor and written in its assembly language, which handle the communication protocol between processors as well as input-output information to the process plant. In each computer the user may communicate interactively in BASIC. To this end, some practical demonstrations have been conducted on a binary distillation process and a double-effect evaporator.

The second software package is concerned with the digital simulation of an Extended Kalman Filter for state and parameter estimation of the distillation process. Although the simulation results are satisfactory, an on-line implementation for the present hardware is prohibited by relatively long computation times.

This research work has shown that although it is practical to apply microprocessor technology to process control problems the learning curve is steep and software development costs, in terms of man and machine hours, still remain the biggest consideration - a finding which is compatible with many independent initiatives.

Key words

Microprocessors, linked processor system, process control, Kalman filtering, simulation.

ACKNOWLEDGEMENT

The author is indebted to Dr. B. Gay for his supervision of the thesis, in particular for the guidance and idea generating sessions throughout the course of this research.

The author also wishes to acknowledge with thanks, the assistance given by the following:

The Public Services Department, Government of Malaysia, and the University of Malaya for the provision of a grant.

The Staff of the Malaysian Students Department, London; in particular Encik Wahid Md Don and Encik Ikmal Hijaz for their help in the 'less controllable' aspects of my stay in the U.K.

Messrs. F. M. Lea and D. Bleiby for their attention during computer system design development.

Miss Eileen Broughton for her excellent typing and retyping of manuscripts.

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF PLATES

NOMENCLATURE

A,B	Brambilla's constants
A_i, B_i, C_i	Constants for component i in the Antoine equation
A(t)	Plant driving matrix
B(t)	Plant input driving matrix
D(t)	Plant noise driving matrix
E	Mathematical Expectation operator
cov	Covariance
F	Feed rate (mol/hr.)
H	Vapour enthalpy (kJ/mol)
h	Liquid enthalpy (kJ/mol)
h_D	Liquid height in downcomer
h_{OW}	Liquid height over weir
h_W	height of weir
I	Unit matrix
J	Jacobian matrix
J_c	Cost function (to be minimised) in Kalman Filtering problems
K	Gain matrix (Kalman Filter)
K_i	Equilibrium relation ($K_i = y_i/x_i$) where x_i and y_i are liquid and vapour compositions of component i respectively
K_p	Proportional gain in DDC loop
K_I	Integral action
K_D	Derivative action
L	Liquid flow e.g. L_R is reflux rate
M	Measurement Matrix (Kalman Filter)
M_{RD}	Reflux drum hold-up
M_B	Reboiler hold-up

M_n	Tray liquid hold-up
P	Error covariance matrix (Kalman Filter)
Q	Process Noise Matrix (Kalman Filter)
R	Measurement Noise matrix (Kalman Filter)
V	Vapour throughput
x(t)	random state variable or vector
p(x,t)	probability density distribution of x at time t
$\bar{x}$	mean value of x
$\tilde{x}$	predicted value of x
$\hat{x}$	estimated value of x
z	measurement vector
z^{-1}	unit delay
v(t)	measurement noise (white, Gaussian and zero mean)
$\omega(t)$	process noise (white, Gaussian and zero mean)
u(t)	deterministic control input vector
Φ	State transition matrix
δ	Dirac δ-function or 'deviation'
Δt	Sampling interval (sometimes as T)
k	non-negative integer, usually denotes sampling instant in discrete time systems
f,g	non-linear functions
r(k)	set-point at time = $k\Delta t$
$\int$	integration operator

Subscripts

o	reference
i,j	with respect to component i or j
k	at time $k\Delta t$

Superscripts

-1	delay or inversion
T	transpose
o	reference
*	equilibrium value

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued) Page

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued) Page

TABLE OF CONTENTS (Continued) Page

TABLE OF CONTENTS (Continued) Page

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

CHAPTER ONE

INTRODUCTION

1. INTRODUCTION

The computer has opened the door to a large number of interesting and important applications ranging from on-line computer control of industrial processes or laboratory experiments, where the computer is the central component in the system, to miniaturised versions embedded in an ever increasing variety of consumer goods.

This research is basically concerned with one such class of computer applications, namely, the use of computers (a minicomputer and a linked microcomputer system) in the control of chemical engineering processes. This chapter covers the background and general concepts to such work and begins with a general perspective of computers and their uses, with special reference to microprocessors and microcomputers.

1.1 Early Development and Use of Computers

Evidence of early computation goes back to as early as 1700 B.C. when the Babylonians began using base 60 (sexagesimal) calculations from which came our units of hours, minutes and seconds.(1) Later, the abacus became extensively used in Asia and by 1630 the slide rule, automating the tasks of multiplication and division, had already become the most popular calculating tool in Europe.

As commerce and society became more 'sophisticated' in the 17th and 18th centuries, ancient calculating tools and aids proved inadequate so that many attempts were made to build mechanical calculating machines notably that of Pascal (1623-1662) and Leibniz (1646-1716). However, the person generally accepted as the father of today's computer(2) is Charles Babbage (1792-1871) who provided a model of a rudimentary computer in his Difference Machine (or Engine as he called it) in 1822 which he later improved into a more general purpose device called the

Analytical Engine. But it was not until a century later that similar ideas were realised by the pioneering work of Konrad Zuse[3] of Germany and independently, John Atanasoff[4] of the United States. Zuse's Z3 machine is believed to be the world's first general purpose, program controlled electro-mechanical computer. The Atanasoff machine, built in 1939, in the laboratories of Iowa State College, is widely regarded now as the world's first electronic computer.

The Atanasoff computer is particularly significant as computations were based on the binary number system (base 2) and the use of a machine regenerative memory.

But the most important pioneering effort was the design and building of the Electronic Numerical Integrator and Calculator (ENIAC) in 1943 through 1946, directed by John Mauchly and J. P. Eckert.[5] The ENIAC used electonic vacuum tubes instead of electro-mechanical relays (a 1000 times speed improvement) and is the first, large scale, fully electronic computer. By today's standards, the ENIAC was an enormous machine. The following features of the ENIAC explains why.

- components - 18,000 vacuum tubes
  70,000 resistors
  10,000 capacitors
- power consumption - 150 to 200 kW
- weight - 30 tons
- floor space occupied - 15,000 sq. ft.
- performance - 5,000 additions or subtractions per sec. 300 multiplications per sec.
- others - use of a central clock to synchronise operations. Use of flip-flops as the basic memory element. Low storage capacity (only 20 ten-decimal digit numbers).

The ENIAC was truly general purpose. Before retiring from active service in 1955, it processed 80,223 hours of work. Data input/output were on punched cards. Programming was done by wiring of component connections and was therefore formidable as a sound knowledge of machine operations was required.

Soon after the ENIAC was built, John von Neumann, himself a consultant in the ENIAC project, proposed the concept of the stored program computer. He suggested that instructions and data are better stored in the computer. Thus, instructions can be changed without manually re-wiring component connections and also since the instructions are stored as numbers, the computer could process instructions as if they were data. This made possible the automatic modification of instructions and alteration of their sequence. This concept led to the first fully complete stored program computer - the Electronic Delay Storage Automatic Calculator (EDSAC) built at Cambridge University in 1949 under the direction of M. V. Wilkes.(6)

The Universal Automatic Computer (UNIVAC)-1 in March 1951, largely due to the enterprising efforts of Mauchly and Eckert, turned out to be the world's first production-line digital computer. Instead of vacuum tubes, UNIVAC-1 used crystal diodes thereby foreshadowing the solid-state era. Its first installation was at the United States Census Bureau in 1951. Its first commercial installation was at the General Electric plant in Louisville, Kentucky.

The 1950's saw a flurry of activities not only in the development of hardware but also in the different levels of computer software. The invention of the transistor in 1948 eventually led to the production of the next generation of computers. These transistorised digital computers, coupled with the development of programming languages such as FORTRAN (1954), ALGOL (1958-1960) and COBOL (1959) greatly increased the use of computers especially in the data processing area.

In the same period, parallel milestones were also observed in industrial control and instrumentation. In the 1940s the application of vacuum-tube electronics to measuring instruments and the development of pneumatic force balance transmitters were forerunners of the industrial control rooms. The so-called pivotal year was 1958 when first-generation electonic control systems were introduced. Polymerization was reported as the first industrial process to be brought under closed-loop computer control.[7] It was reported that it took the company, Texaco, 2½ years to make the necessary preparations for the automation and a further five months to develop the computer model. The polymerization unit was connected to a Thompson-Ramo-Wooldridge RW-300 digital computer which gathered information from 110 sources, controlled 16 different streams, pressures and temperatures and sounded alarms if failure or danger threatened.

The use of direct digital control (DDC) in the process industry however, failed to match the euphoria it generated in the marketplace when it was first introduced in the early 1960s. A full scale DDC implementation was relatively rare despite its practical feasibility being demonstrated at several plant experimental trials.[8,9] One such application encompassed 100 loops in an ammonia-soda plant but as in other implementations especially in the petroleum industry, a complete set of analogue back-up units was always maintained. Apparently DDC went through a period of bad reputation largely due to the lack of sufficient understanding of both the digital computer and modern control theory on the side of its implementers.

The conventional DDC systems were characterised by a centralisation of computing power, both at the lowest (control) and highest (management) levels of the control system hierarchy. As a result, they suffer from the following setbacks which also precluded

DDC systems from wider acceptance.

a) heavy computational load on the control computer.

b) complex software and programming techniques required.

c) reduced individual loop information.

d) total system reliability is reduced.

e) individual loop performance must compete with that of the classical Proportional + Integral + Derivative (PID) analogue single-loop controller.

The appearance of the minicomputer based on integrated circuitry in the mid-1960s reduced some of the problems but in general computer usage was largely directed to open-loop plant optimisation operations,[10] management and plant-wide data processing for the mainframes bracket and also as a useful tool in process design and simulation. A useful introduction to the early development in computer process control is provided by Savas.[11]

1.2 Era of a New Computer Technology - The Microprocessor

Although in one sense, today's computers have changed little since the mid-1940's, there is an enormous difference between today's machines and those of only thirty years ago. Computer technology has basically entered into a fourth generation involving large-scale integration (LSI) methods to manufacture computers. The following dates roughly indicates the major technological advancements that have taken place.

1st Generation (1939-1954): Valve Computers

2nd Generation (1954-1965): Transistorised Computers

3rd Generation (1965-): Integrated Circuits (IC)

4th Generation (1971-): LSI and Very Large Scale Integration (VLSI)

In a report prepared for the Department of Industry (UK and Eire) by the Massachusetts Institute of Technology[12] it was reported that "In 1969, M. E. Hoff, an engineer for Intel Corporation ..... discovered that he could incorporate the entire central processing unit on a single silicon chip. By attaching two additional chips - one for input/output (I/O) and another for inscribing a program - Mr. Hoff had what amounted to a basic computer!". With that, the era of the microprocessor has arrived, promising with it, a most profound technological effect on the society and the world at large.

1.2.1 A Brief History of Microprocessors

Strictly speaking, a microprocessor may be defined as a Metal Oxide Semiconductor Large Scale Integration (MOS LSI) system that contains the arithmetic, logic and control units needed to form a complete digital processor. Whether it is realised on a single (i.e. monolithic) or on a small number of silicon-based chips, the microprocessor forms the central processing unit (CPU) of a new generation of digital computers. When this microprocessing unit is combined with memory, auxiliary circuits, power supply and control panel plus the minimum software into an integrated system, a microcomputer is produced. It is important to uphold this conceptual difference between a microprocessor and a microcomputer although the two terms have been used interchangeably in literature.

The microprocessor is a natural outgrowth of the semiconductor revolution which began when complete transistor circuits were first diffused on a single piece of silicon in the late 1950s. These early 'integrated' circuits contained the equivalent of only 10 to 20 transistors whereas today's chips may contain as many as 100,000 or more transistors. This is why microprocessors are the products of semiconductor manufacturers rather than existing computer manufacturers.

The first commercially available microprocessor was the 4-bit Intel 4004 introduced in 1971. As program storage is entirely in read-only memory (ROM) and data storage in random access memory (RAM), a microcomputer system based on the 4004 is a calculator-oriented system. This is not surprising as the 4004, comprising 1600 transistors on 0.25 cm^2 of silicon, was originally designed as a custom LSI part for the calculators made by the Japanese firm, Busicom. The 4-bit neatly allows Binary Coded Decimal (BCD) applications but in several ways this first product was very limited. The greatest integer which could be held is 15 and there is a lack of interrupt capability. Similar constraints were also exhibited by other 4-bit devices although the IMP-4 is micro-programmable.(13)

Eight-bit microprocessors were soon produced partly to overcome the limited performance of their 4-bit predecessors. The monolithic Intel 8008, introduced in 1972, was the first of these first-generation microprocessors. Although it has a 48-instruction set, a 16 x 1024 (16K) memory address capability and a limited interrupt facility, early users found it necessary to design considerable support and interface logic in their applications.(14) It was used mainly to fill in system requirements where either transistor/transitor logic (TTL) became too large and complex to be handled conveniently or minicomputers were too large and expensive.

As semiconductor technology advanced, second-generation microprocessors came into the market notably the Intel and Motorola family series. Unlike the Intel 4004 and 8008 chips which were fabricated using Positive MOS (PMOS) technology, second-generation devices like the Intel 8080, Motorola MC6800, Fairchild F8 (all introduced in 1974) and the Zilog Z80 (in 1978) were Negative MOS (NMOS) chips. They are three times faster and because of TTL compatible voltages these microprocessors emerged as the most popular devices.

1.2.2 Current State of the Art

It is difficult to assess accurately a technology which is in a high state of flux. Semiconductor advances have led to the production of more powerful eight and sixteen-bit single-chip microcomputers. The level of NMOS integration (memory cells/unit area) continues to double every year (the so-called Moore's Law, after Gordon Moore who first formulated this empirical growth pattern[15]) since the mid-1960s and with the availability of 64K RAMs and 256K ROMs, Japan is now on the way to produce 256K RAMs that would put her in a clear lead in semiconductor products.[16] Another significant trend has been the increasing use of Complementary MOS (CMOS) or CMOS/NMOS technology in processor and memory design. These devices are relatively fast (about 150 nanoseconds (ns) access time) but have a low power consumption. Also, an important development has been the production of Electrically-Erasable and Programmable ROMs (EEPROMs) that match industry standard Ultra-Violet EPROMs in density.

However, memory growth patterns and the popularity of microprocessor-based systems do not necessarily lead to completely new types of microprocessors. Very often, prototype designs are refined to meet the requirements of modern and structured high-level software. Thus, CPU designs, although still very much register-oriented as in early designs, are increasingly emulating various aspects of the stack architecture. Many processors have multiple index registers now giving rise to powerful and efficient addressing modes.

Take the Motorola series of 8-bit microprocessors. During 1977 and 1978, Motorola introduced several new chips such as the M6801, M6802, M6805 and M6809. The M6801 is one of the earliest VLSI microcomputers containing about 40,000 transistors. It is object code

compatible with the M6800 and include additional 16-bit instructions. The M6809 is a much more powerful processor. Its hardware architecture provides for two stack pointers, two index registers, a direct page register and 16-bit arithmetic. The major strengths of the M6809 lie in the addressing modes. For example, the Program Counter (PC) relative addressing mode permits position-independent code. In fact, this has allowed Motorola to sell mass-produced firmware in ROMs that can reside anywhere in the 64K address space. The direct page register allows quick access to any page in memory and simplifies the generation of software for multi-tasking operations. In some benchmark tests[17] the M6809 has shown a 2.7 times improvement in speed of the M6800, required 42% fewer instructions and used 33% less code. A comparative software analysis[18] of the M6809 showed that it is superior to the M6800 and the Z80.

But the real attention in recent years has been directed to the 16-bit microcomputers and development of supporting chips to meet the requirements of more powerful operating systems, block-structured high level languages and sophisticated computer graphics. Microcomputer support chips are developed basically to overcome three factors.[19] The first is the so-called 'Von Neumann bottleneck' problem in single processor systems. This has led to the concept of the co-processor or multi-processors. The second factor has been poor programmer productivity. This has led to the production of memory management units (MMU), virtual memory support and various peripheral controllers. And thirdly, the difficulty in interfacing microprocessors to analogue systems. This is partly solved now with the availability of a great variety of 8-bit Analogue-to-Digital and Digital-to-Analogue Converters (ADCs and DACs), compatible with microprocessor data busses, with conversion times of 10 to 500 microseconds (μs) and costing only £3 to £10.

1.2.3 The Microprocessor and Society

The real impact of microprocessor-based technology is yet to come, affecting both people and industry. The Optimist speaks about Continuous Technological Growth and Infinite Social Adaptability. The Pessimist speaks only about Zero Social Adaptability. What ever the situation is, every random sampling of microprocessor applications seems to reveal its protean utility.

There are now about 150 different computer manufacturers in the United States alone. The magazine, Office Systems[20], estimated the number of personal computers in the U.K. by the end of 1981 to be 176,000 (or one per 286 persons) as compared with 1,936,000 in the U.S. (or one per 125 persons). In a recent issue of the magazine 'What Micro?', a list of commercially available personal computers in the U.K. includes more than 200 different systems ranging in costs from under £500 to over £3500.[22]

And in the last one and a half years, over a 100 16-bit microcomputers have been launched on the world market. Of these, almost 50 are reportedly now available in the United Kingdom.[21] They form many of the more powerful personal computers and small business systems, and are much sought after by universities and other research establishments. The Motorola 68000 chip for example has a 16-bit address bus but uses a 32-bit internal architecture. It can run twice as fast as the DEC 11/70 microcomputer and since it is able to directly address up to 16 Megabytes of RAM, it is, in this respect, comparable to the IBM 370 mainframe computer!

The microprocessor is slightly over ten years old now. Over the period, it has become one of the most exciting technological innovations of this century. And the limits of NMOS technology has not yet been reached. In 1981, Hewlett Packard introduced their single-chip

32-bit processor using 450,000 transistors operating with an 18 MHz clock.[23] It can execute a 32-bit integer addition in 55 ns, a 32-bit integer multiply in 1.8 microseconds and a floating point multiply in 10.4 μs. More recently, an advanced-architecture microprocessor, the 16-bit AAMP, has been announced by Rockwell International.[24] Moving away from classical von Neumann and register-oriented designs, the AAMP is a VLSI CMOS/SOS (CMOS/Silicon on Sapphire) chip and has a language-directed stack architecture giving high throughput and high compiled code density.

1.3 Microprocessors in Control

The following factors have contributed to the application of microprocessors in the control industry.

1. Lower cost per function.
2. Flexibility.
3. Stability, Accuracy and Security.
4. Human Factors.
5. Advanced Control Capability.

As mentioned earlier, LSI technology has enabled thousands of digital circuits and special-function logic to be fabricated on a single chip at very low cost. Because the microprocessor can be programmed to perform different tasks, flexibility is more readily achieved than in equivalent analogue systems. Digital devices are more reliable and secure. Accuracy is only a function of bits used to represent a basic unit of information. Digital information can be processed and/or transmitted for data or report presentations and graphics display. This leads to improved operator-interface designs. Finally, the protean computing ability of the digital computer can be used to implement advanced control techniques which proved difficult in classical analogue systems.

The topic 'microprocessors in control' is very broad and can be discussed from many viewpoints such as its market growth, price change, control applications, theories, computations, computer architectures, software/firmware/hardware mixes, programming developments, LSI technologies, performances, reliabilities, etc. In the context of this work, it is more profitable to concentrate on microprocessors in process control.

1.3.1 Nature of Process Control Problems

Process Control is basically Large System Control. More specifically, it is fluid process control which involves the control of plants manufacturing homogeneous materials such as oil, chemicals, paper and concrete, etc. Because of the nature of the product, control manipulation is possible with simple valves or actuators. The state of the processed material can be inferred from simple measurements of continuous properties and most of the control variables are naturally bounded and self-stabilising.

Chemical processes are usually smoothly non-linear, highly uncertain, and of very high order, modelable only to a very gross approximation. A plant may involve thousands of measurements and actuators, and hundreds of control loops. Some plant complexes occupy many square miles of land. Thus, the very nature of the industry makes process control systems very complex and multi-level, and usually achieving a more complete level of automation than more other plants in the sense of produced value per employee. One operator may be responsible for 100 or more control loops but the training and intellect to cope with each loop per hour are slight. Human factors are therefore important.

If new, microcomputer-based process control techniques are introduced, then they must be accompanied with new operational techniques. Also the new process control system should be predictable in its failure modes which means a good manual operating philosophy (or any suitable contingency planning) must be provided.

1.3.2 Current Process Control Design

In the last decade, the image of computerised process control has improved after a bad start in the early 1960s. In the last decade, the design of the central control room has been extended to include better electronic instrumentation and microprocessor-driven measurements. There have been improved power supply systems and Cathode Ray Tube (CRT) operation distributed control. In the 1980-1990 period, the evolution of process control system technology is expected to continue with the expansion in direct digital control, intercoupling of complex digital devices and the increasing reliance on complex measurement. This is reflected by an overview of the present process control system design as illustrated in Figure 1.1.

In reference to this system overview, Stanton[25] has discussed some of the practical problems created by poor system maintenance and design, in spite of the availability of good measurement and computational equipment. Too many different hardware vendors for the system is cited as a common problem. It means proper interface support and protocol must be provided for data communication between field devices and control room equipment, and the digital computer.

Kane[26] also shared Stanton's view that poor maintenance of instruments can lead to costly experiences in a computer control environment. System technicians tend to overlook basic instrument

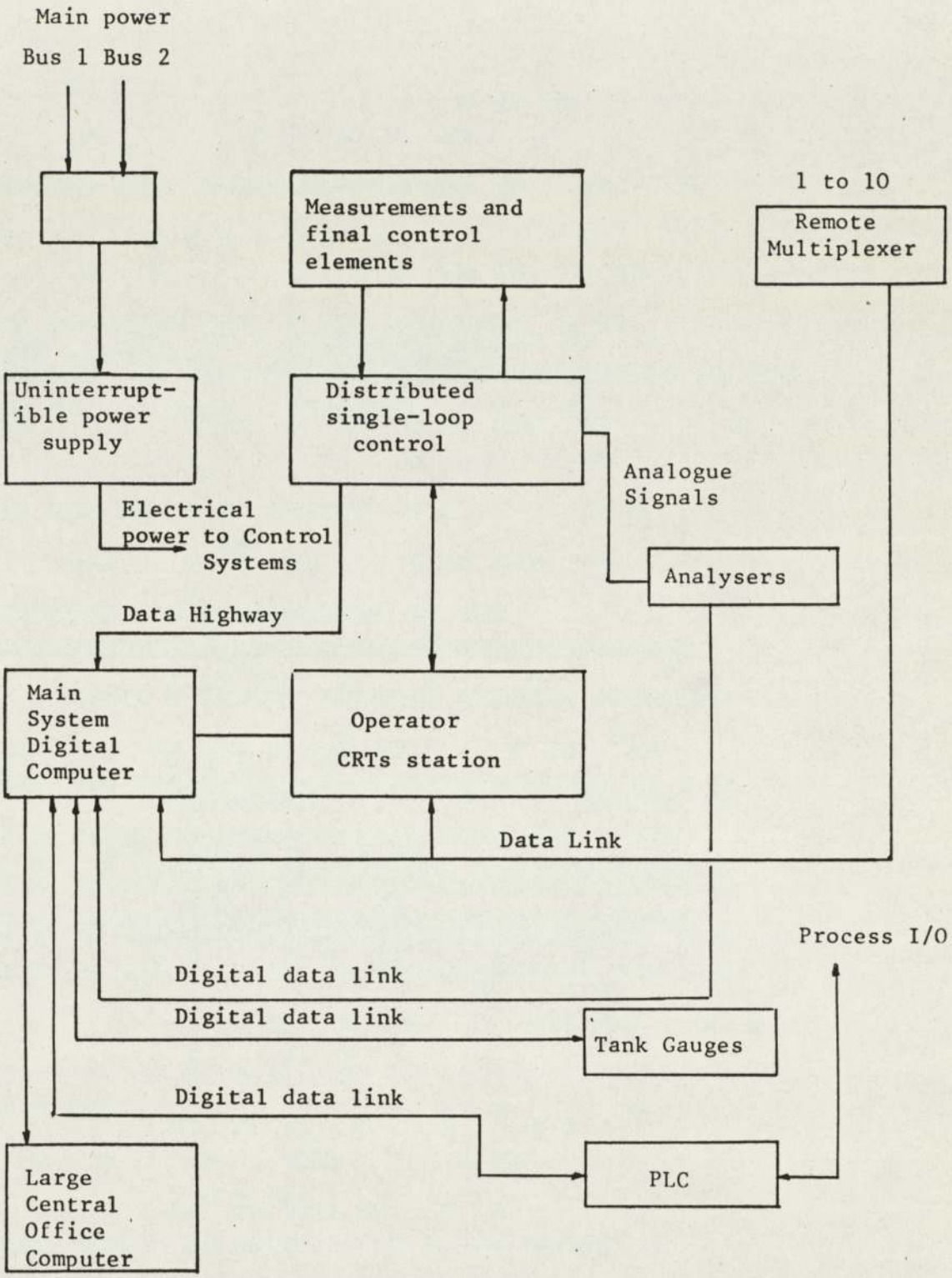


Figure 1.1. Present Process Control System Design.(26)

engineering and the real reason for poor system performance could be that an orifice plate is not installed correctly, a potential ground loop is overlooked or placing turbine meter runs in vertical pipe runs.

1.3.3 The Basic Process Control Loop

A block diagram of the classical feedback control loop is shown in Figure 1.2. Basically, a good control system should have the following features.

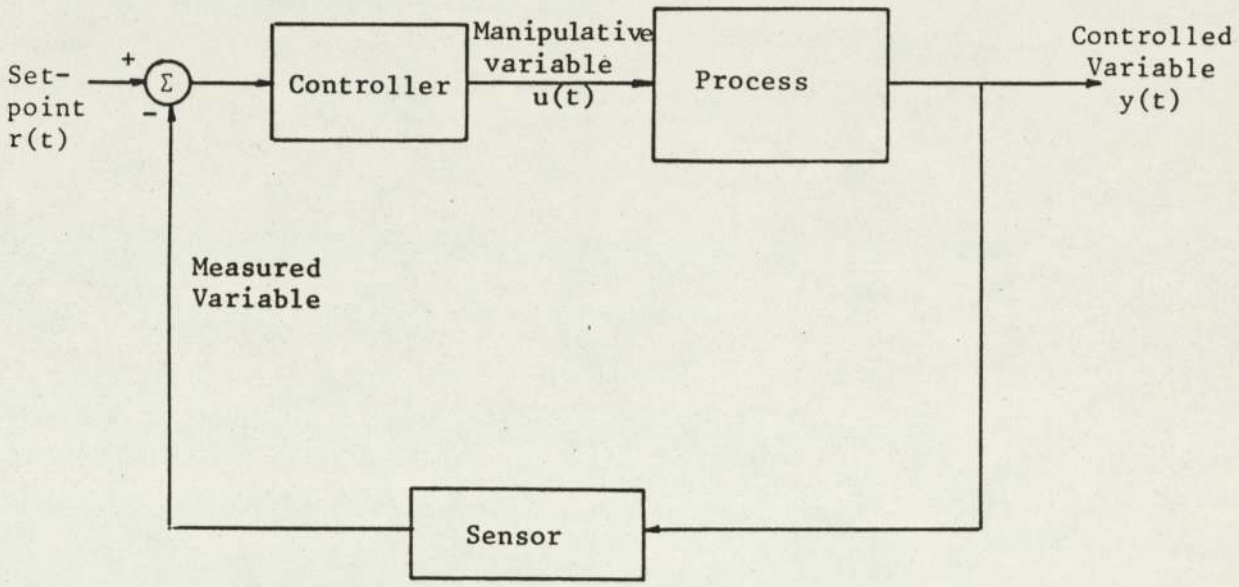


Figure 1.2 Block diagram of the classical feedback control loop

a) A good servo (set-point tracking) operation. Set-point changes should be fast and smooth.

b) A good regulatory function.

c) The control algorithm should be robust and reliable in the face of all possible disturbances and changes.

d) The controller itself should be reliable and easy to maintain.

e) The controller should be designable (tuneable) with a minimum of information concerning system disturbances and the system architecture.

The PID control algorithm, expressed in analogue form, has been the trademark of the process control industry since it evolved from the ingenious design of bellows and linkage mechanisms in pneumatic controllers some fifty years ago. 90% of all such controllers are said to be Proportional + Integral (PI). From the standpoint of universality and stability, PI represents a standard by which new control forms (for example, direct digital control algorithms) are measured.

When a microprocessor is used as the controller, it is only natural to write digital control algorithms that simulate analog control laws, the discrete PID algorithm whilst adding more flexibility for control loop interactions such as ratio-, cascade-, and feedforward control. As Figure 1.3 shows that the individual controllers gains K_P, K_I, K_D can be made independent of each other although they now become a function of the sampling time, T. To avoid the effects of sudden changes in the

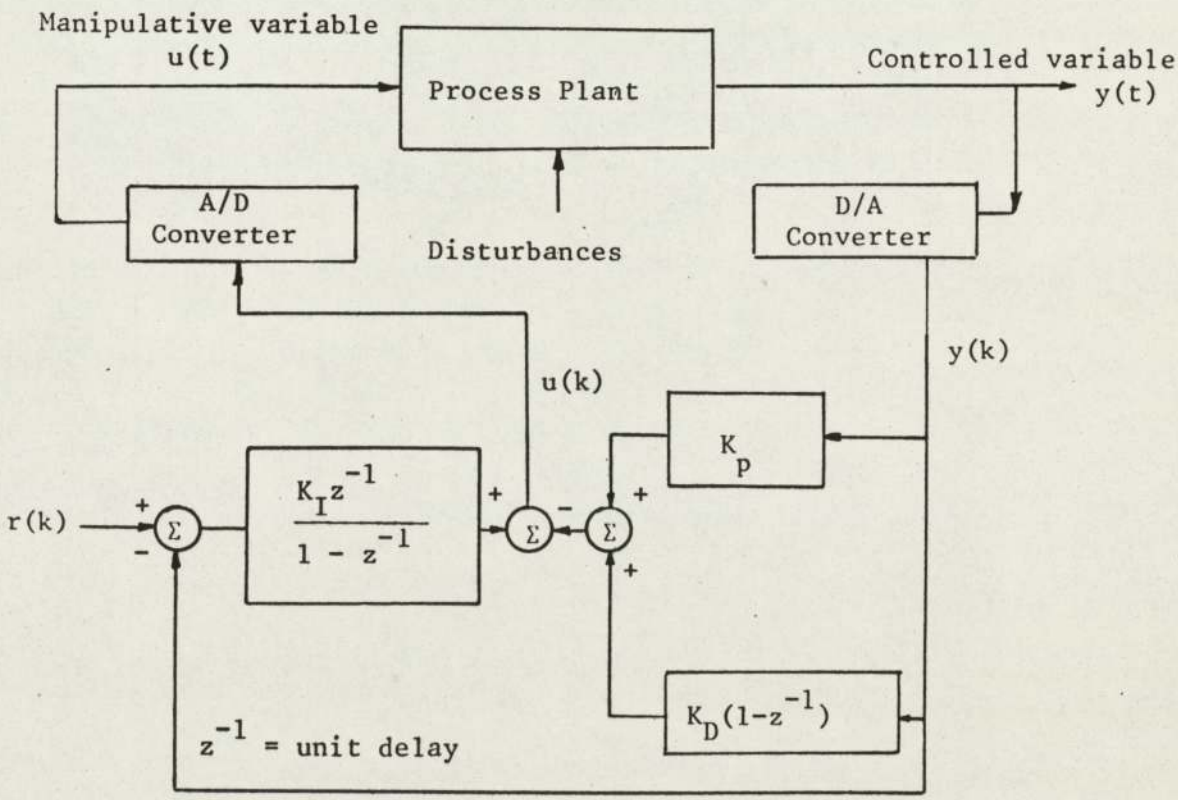


Figure 1.3 Direct Digital Control Loop (P+I+D)

manipulated variable when the set-point (remote supplied or internal) is suddenly changed (set-point and derivative kick), the set-point signal r(k) is only included in the I-action. With something like fifteen years experience in the PID-DDC algorithm implementation, it is reaching maturity. In all practical systems, the control algorithm must be packaged to provide certain key facilities:-

1. internal or remote set-point
2. bumpless transfer in auto/manual changeover
3. output control signal limiting
4. control signal conditioning
5. integral de-saturation and anti-reset windup
6. filtering of measured process variables
7. linearisation of measured process variables

Most analog controllers provide the first five facilities as standard and the last two facilities ought also to be included in any digital based control scheme. Figure 1.4 indicates how these facilities might be incorporated into a simple DDC package for a PID algorithm.

1.3.4 Recent Trends

It has often been said that the process industry has a good reputation for its tradition of holding off new technology. This cautious conservatism is to a certain extent acceptable for there is a need to safeguard vital and costly plant and equipment. New techniques do not really arouse practical interest unless they have a sufficient long and proven history of reliability. The other main reason is system component economics i.e. the analogue vs. digital trade-off. As a result of these two related factors, DDC never really challenged the primary controller market. Nevertheless, the experiences in implementing analogue and minicomputer-based systems have been invaluable.

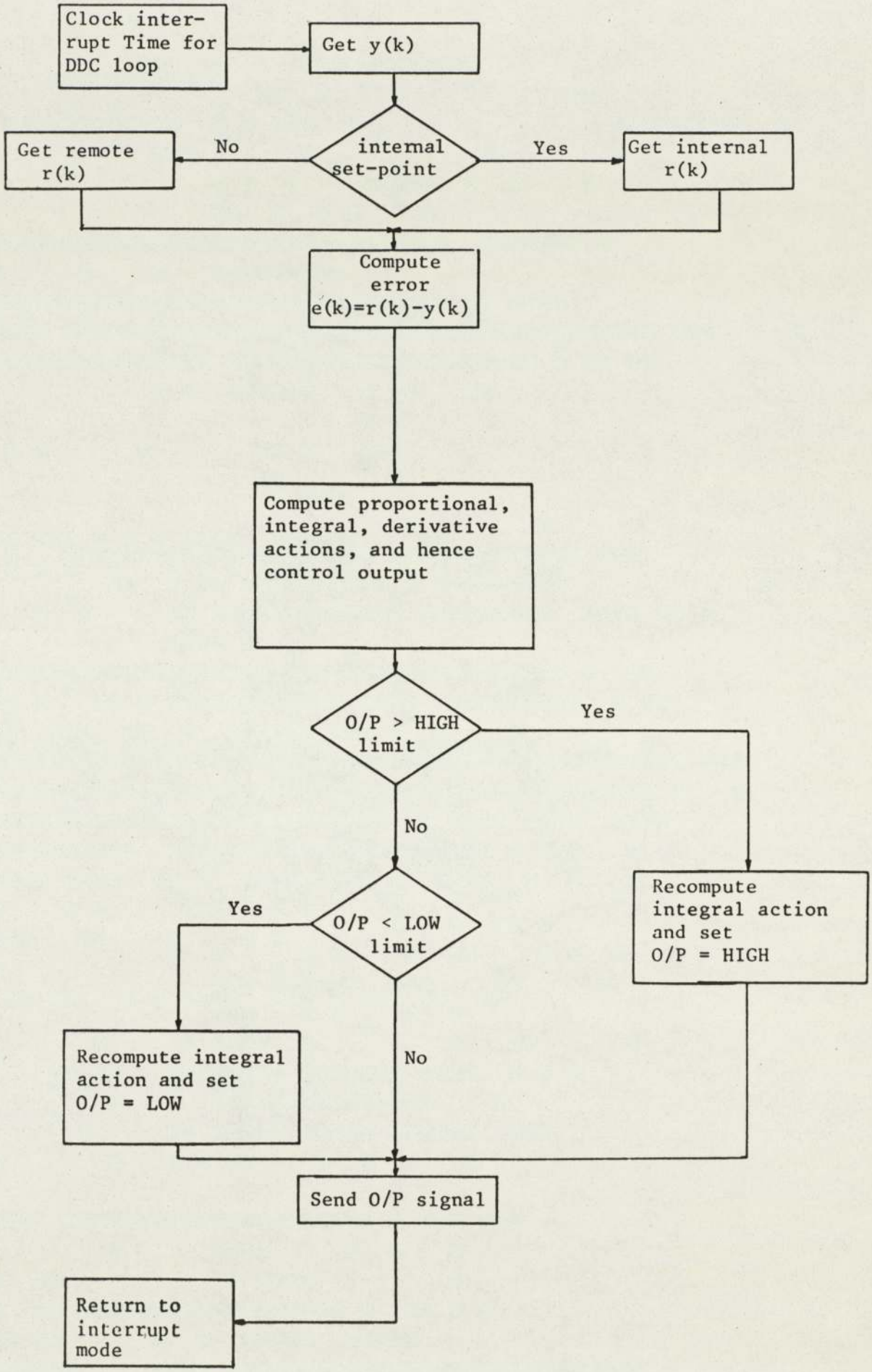


Figure 1.4 Flow Diagram of a Simple Digital PID Controller

Now that powerful microcomputing power is cheaply available and the increasing availability of suitable software, attitudes are beginning to change. The response has blossomed into two main directions:

1. The development of stand-alone, single-loop digital controllers.
2. Distributed Digital Control Systems.

Because of the theoretical and practical implications (and the so-called theory-applications gap) involved in such concepts, it warrants paying each a closer attention.

1.3.4.1 Stand-Alone, Single-Loop Digital Control

Classical single-loop control is built upon a sound theory and a long history of industrial applications. It has also been the bastion of the conventional PID analogue, feedback controller. It is also true that the cost, size and reliability of the analogue controller is improving but the nature of the algorithms that can be realised remains unaltered. The use of the microprocessor removes this limitation and allows more complex control algorithms to be implemented with relative ease.

Stand-alone, single-loop digital (SASLD) controllers are systems which have no central control computer and would have otherwise used independent analogue controllers. Such a system is a general purpose application for which a degree of adjustment or programmability is required to meet the needs of specific processes. Some studies in hardware and algorithms applicable to stand-alone controllers can be found in the literature.[27,28,29]

A SASLD controller would not offer much advantage over its analogue cousin unless it is field programmable in a readily understood

language (which would not be true if the controller is only ROM-based). Researchers in this field say that the SASLD controller can take up a computer format such as the Commodore's PET but with limited computing capability and memory to be economically attractive. Such requirements therefore place stringent requirements in the process models and control algorithms that can be used. These are as follows:

1. The parameters of the process model must be directly related to easily measured and easily computed process properties.
2. It must be possible to compute the controller parameters from the process model parameters with a minimum number of iterations.
3. The on-line control policy must be simple and fast.

Some modern digital control algorithms encompassing Finite-Time Settling Time (FTSC) and State Variable methods have been discussed by Anslander, et al.[31] They suggested that any transcendential routines and floating point calculations should be implemented on hardware rather than software methods.

1.3.4.2 Distributed Digital Control Systems

This trend is more evident as reflected in various meetings and journals in the control and engineering industries as well as the product trends in the controller market.[30,32-37] Actually, distributed control is not a new concept. Chemical plant complexes are inherently distributive in nature. The main agent of distributed control has in fact, been the analogue, feedback controller.

Distributed control is not a design concept. It is a control philosophy. In the early stages of process control, the instruments were located next to the process they were monitoring and the peripatetic

control man (the controller in fact!) would take measurements from the local instruments, adjust handwheels to correct deviations of many process variables and report back to his central management when things go out of hand. Today, local microprocessor-based instruments and controllers can perform these tasks much more effectively and efforts are being made to manufacture sufficiently rugged devices to survive the hostile environmental conditions.

Some control system designs and implementations based on the distributed control philosophy have been reported.[38,39] But above the level of direct or local (DDC) control, the systems do exhibit limitations. These include the ability to link to only one control operating system, and the dependence on one process computer.

A more general system design solution has been proposed by Kent Process Control Ltd.[40] A schematic diagram is shown in Figure 1.5. The system presents a 'Totally Distributed System' with total unit independence, functional and geographical distributions. The major benefits of such a design are as follows:

1. A single element failure would not bring down a significant sized area of controlled plant.
2. Reduced cost and size of units by functional distribution.
3. The system is expandable and contains no bottlenecks which restrict the data path or system operation.
4. Reduced wiring costs and signal noise by geographical distribution.
5. Maximum autonomy of distributed units.
6. System is evolutionary and can cope with changes in technology and market forces.

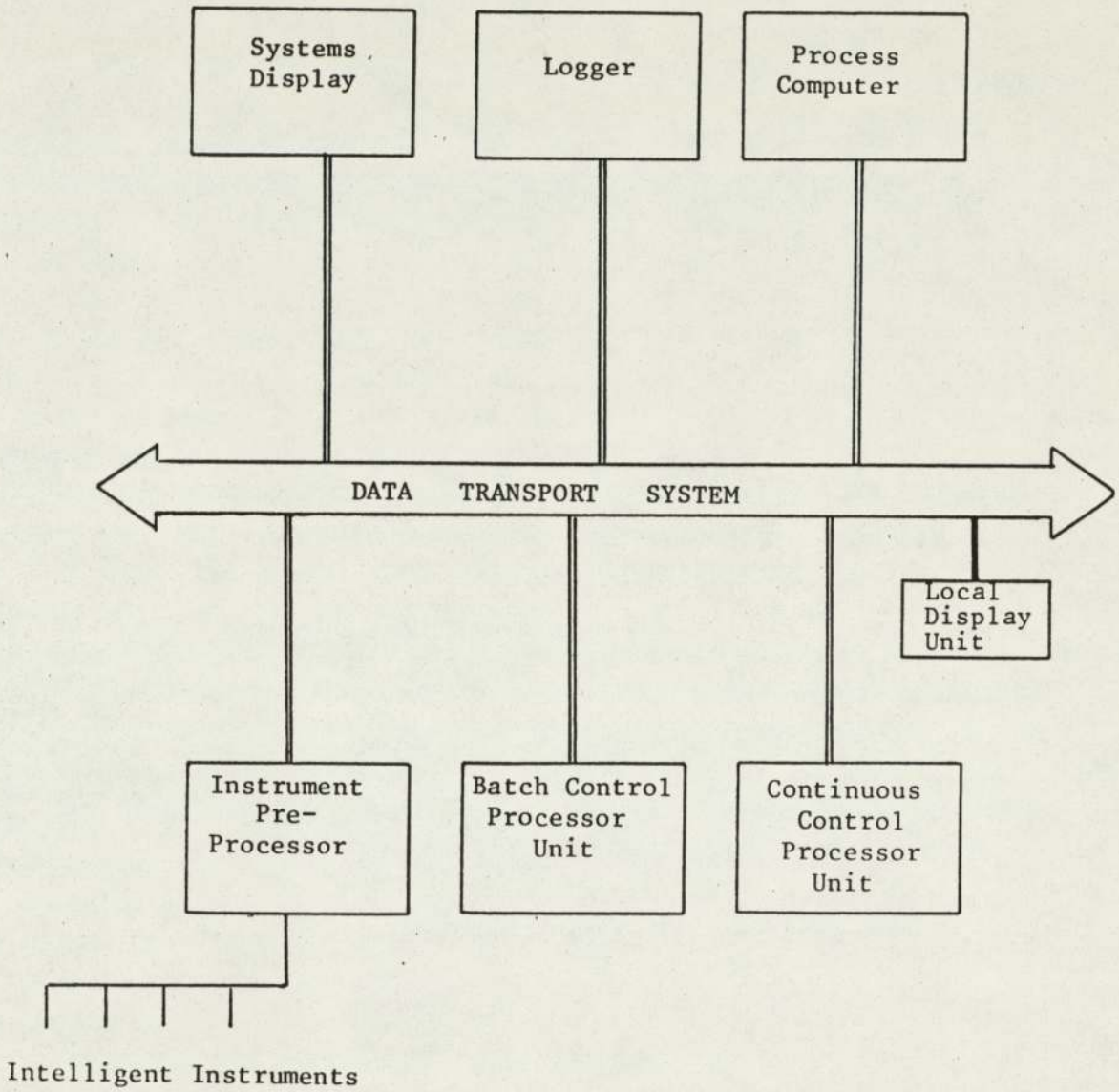


Figure 1.5 An Overview of a Totally Distributed System[40]

The system is basically made up of three main elements - the management units and the process units, linked by a secure multipath communications highway. Management units include operating stations, loggers and process computers and are usually associated with control rooms and process management functions. Process units, which have been

given a high degree of functional and supervisory powers for maximum autonomy, are normally located near to the plant they control. Physically, the process units are identical, but by implementing different software packages they can be used to carry out instrument pre-processing, batch or continuous control functions.

The system is also modular. Each module can support up to 32 management and process units in any combination to suit operational and managerial needs. Because these units can be located exactly where they are needed and instant total system access is possible, the system is said to be truly functionally and geographically distributed. Furthermore, the system can easily accommodate the SASLD controller discussed earlier. Some aspects of the control hierarchy of conventional DDC systems are also preserved.

It is believed that with improvements in data communications technology, and microprocessor support chips and software, such a system will be fully exploited.

1.4 Background to the Research Project

The Chemical Engineering Department of the University of Aston-in-Birmingham has had for some years a well developed interest in the real-time use of computers for data acquisition and processing and process control. The basic tool for these studies has been a Honeywell H316 minicomputer which through a Honeywell Analogue-Digital Input-Output System (HADIOS) has been linked to a number of items of process plant (distillation column, double-effect evaporator, resin manufacturing plant, chemical reactor, etc.).

Over the years, the original software has been developed and extended to support these applications. In particular, the BASIC interpreter has been modified to permit real-time use by a single user.

This is done via a collection of Assembly Language subroutines (the HADIOS Executive) callable from BASIC, to operate the various sub-interfaces of the HADIOS. A Graphics Library (originally written in FORTRAN by Tektronix Inc.) is also provided. The system therefore allows the user to quite conveniently write and modify his BASIC program on the Visual Display Unit (VDU) or an ASR-33 Teletypewriter (TTY), literally next to his plant, conduct process monitoring or control experiments and plot his results.

More specialised applications such as multivariable control, on-line estimation and plant dynamics studies would require the user to write additional subroutines (FORTRAN and/or Assembly Language) to suit his particular application. Also, the computer core memory is only 32K and the user may have to use the floppy disk storage facility for some software overlaying.

More recently, the Department purchased a M6800 microcomputer system. Through its 8 I/O Ports the M6800 can be connected to a variety of external devices including ADCs and DACs. The software of the system is disk-based with a good operating system (DOS). Of particular interest is the SD BASIC Compiler which can be interfaced to M6800 Assembly Language subroutines and will run in an interrupt environment. The M6800 micro-computer system has been used for small real-time tasks such as data acquisition and processing from instrumentation but when compared with the H316, it is very limited in the number and variety of inputs it can handle and in its supporting software. Again, only one user is supported.

The project was initiated because it seemed worthwhile to investigate the feasibility of a linked twin processor system through which, for instance two users could share hardware and software resources or, alternatively two parts of the same task could be divided between the computer system.

It was thought that the logical 'initial assault' on the problem was to provide the M6800 microcomputer system with its own Executive program, similar in function to the HADIOS Executive, and interfaced to the SD BASIC compiler. And if the proper hardware arrangements and software protocol for the linked twin processor system can be established, the M6800 user would then have independent access to HADIOS and hence his plant. To do this, the author had realised that the existing HADIOS Executive would have to be considerably modified. For efficient utilisation of microcomputer memory and to retain the feature that on-line M6800 users would only need to write his programs in SD BASIC, the M6800 Executive would have to be written in M6800 Assembly Language. Since this level of programming is more hardware dependent than others and that the author had no previous contact or experience in writing such software, much of the initial effort was directed to acquiring the basic skills in the low-level programming of both processors.

1.5 Objectives of the Thesis

The primary objective of this research is therefore to develop a linked H316-M6800 twin-processor system for real-time data acquisition and process control. The system should be easy to use and sufficiently robust for the general user. Furthermore, the system should be flexible enough to allow two independent users to share hardware and software resources or, alternatively two parts of the same control or data processing task could be divided between the computer system.

Secondary objectives are basically the applications of such a general purpose package. These are as follows:

1. To conduct practical demonstrations of the linked twin-processor system. The plant selected for this exercise is a distillation column. However, the package

should also permit the simultaneous access of two separate plants if desired.

2. To investigate the feasibility of constructing an advanced control policy based on an Extended Kalman Filtering estimation of the distillation column. The effort needed here is to design and implement a suitable filter for on-line estimation of several process variables and parameters, optimising where possible the advantages offered by the linked twin-processor facility. It is hoped that once an operational estimator is developed, an estimator-aided control policy can be implemented. In the pursuit of this particular objective, the Kalman filter's performance should be studied under simulated conditions. This means some modelling and digital simulation work on the Honeywell H316 minicomputer must be done.

CHAPTER TWO

APPLICATIONS OF MICROPROCESSORS

SOME OBSERVATIONS FROM A LITERATURE REVIEW

2. APPLICATIONS OF MICROPROCESSORS - SOME OBSERVATIONS FROM A LITERATURE REVIEW

The ubiquitous nature of the microprocessor in the field of control and instrumentation has already been introduced in Chapter 1. In this chapter, some observations from a broader look at microprocessor applications including several case studies, are presented.

A literature survey of microprocessor applications has been conducted to achieve the following objectives:

1. A broad base appraisal of microprocessor technology and its impact on society and industry.
2. Discovering the potential uses of microprocessors.
3. To have a good perspective of the various system design techniques used.
4. To understand the problems that arise when using microprocessors.

It is not the objective here to elucidate every reported application as each is usually accompanied by its own specific case history and motivation. Besides, there are just too many to consider. It is better to categorise the applications on the basis of functional complexity i.e. on the degree to which the flexibility and computing ability of the microprocessor is utilised.

Also, many applications reflect the specific hardware architecture of the microprocessor used and in general, the person conducting such a survey should at least be familiar with the basics of the technology. Otherwise, the less obvious points may easily be overlooked. Several good texts[41,42,43] are available for this purpose.

2.1 Overview of Microprocessor Applications

In an attempt to compile reported applications of LSI microprocessors and microcomputers for the period 1970-1974, Ward[44] managed to list only 97 references. Today, such a list would appear endless. As both the microprocessor industry and market began to be defined, new applications have appeared in a geometrical progression. It is already becoming difficult not to come across a microprocessor-based product on our way to work!

An early overview of microprocessor applications by Nichols[45] indicated that microprocessor usage was predominantly in the areas of industrial control and instrumentation, aerospace, computers and communication. Naturally, these cover military aspects as well. The picture is not much different today with perhaps an ever-increasing share going to consumer and business-oriented products. Microprocessors are also finding their way into products where electronics were not used before because the job could be done through electro-mechanical means. For example, they are being used to control traffic lights, household appliances and mixers. A person using a conventional elevator may soon find himself wishing a micro-chip to be in charge of lift operations.

Microelectronics has also allowed greater functional capability than would have been practical in previous design techniques primarily because the incremental cost for additional functions is very small in a microprocessor-based system. This has expressed itself most noticeably in the area of instrumentation where manufacturers are finding it practical to add such features as remote-control, auto-calibration, programmability, improved read-out and peripheral interfaces with little impact on product price. The 4-bit processors are the most popular with such applications and because of the enormous size of the market, the 4-bit processors, especially the Texas Instrument TMS 1000,

have also dominated microprocessor sales figures (number of units shipped per year).

Towards the complex end of microprocessor applications, the 8- and 16-bit microprocessors are normally used. Second and third generation 8-bit microprocessors not only form the basis of many popular microcomputer systems (Apple II, Sinclair ZX-81, etc.), they are also finding themselves increasingly embedded to reduce parts in a system and to enhance processing power. The most demanding data and word-processing applications however, require the power of the 16-bit devices. An application spectrum providing a quick perspective of microprocessor applications is found in an article by Biewer[46].

At the same time, the task of putting the general-purpose processor to solve specific problems fostered a host of supporting LSI peripheral chips which can be more complex than the microprocessor itself. Some like multipliers and arithmetic chips, are designed to enhance processing power; others like CRT and disk-memory controllers unburden the CPU from I/O chores. The newer generation of terminals and printers now have micro-based intelligence giving rise to more flexibility and local control, powerful displays and good quality prints. The more expensive VDUs often support high resolution animated graphics. But amid all the excitement, the production of software - the programs that tailor microprocessors to specific applications, could be the limiting factor in the proliferation of microcomputers. The situation is improving however, as many general-purpose microcomputer systems are now offering facilities with BASIC, PASCAL and also FORTRAN 77 and FORTH.

Detailed studies of a wide range of microprocessor applications abound in literature. The large amount of publications is also due to the fact that there are so many different types of microprocessor chips.

Often, a design solution is repeated with a different microprocessor architecture. A useful compilation covering the area of single chip controllers to more complex microprocessor systems is now available in book form[47]. The editors, Capece and Posa, have selected the more outstanding articles in the literature and grouped them into nine sections which include topics like Signal Processors, Peripheral Support Chips and Software for Microcomputers. In the MIT Report quoted in Chapter 1, case studies in heating, ventilation and air-conditioning controls to sewing machines and medical equipment are presented. The study observed several conclusions which are also compatible with those of other independent initiatives. For example, it showed that the motivations for the incorporation of microprocessors in each of the products studied are highly varied. Also, the fact that standard hardware and components can be tailored to meet each user's special requirements is cited as a major factor in the widespread use of microprocessors. Microprocessor-based products gain another added value - it lies in the efficiency in design and/or manufacturing which arise from replacing hardware with software. Finally, the study showed that in many cases, product performance rather than product cost, became the prime consideration of most manufacturers.

Naturally it is impossible to cover all sectors of microprocessor utilisation although review papers are available. Randle and Kerth[48] discussed microprocessors in instrumentation and described a design for the IEEE Standard 488 (GPIB) Interface Bus. Leventhal[49] provided some solutions to the special problems found in aerospace applications and Klig[50] gave an account of biomedical applications. The main thrust of the following discussion is therefore in the field of industrial applications.

2.2 Industrial Applications

The dual nature of the microprocessor - a piece of micro-electronics and a digital computer, provides the technological basis for widespread use in industrial environments. Since the micro chip is also a subset of a more general class of machines called Information Processors, its applications may also be viewed from the point of information handling. This can be broadly divided into data processing and control of industrial processes.

The data processing problem is one where large volumes of data are manipulated and the nature of the job may change from hour to hour. This is accomplished by using a system with a significant read/write memory and different programs to switch from job to job. The system demands the specialised techniques of program loading, interrupts, multi-processing and direct memory access (DMA), with the emphasis on productivity. Present microprocessors have shortcomings in satisfying the productivity requirement and make very poor computer replacements for data processing. The picture may change in the next few years if the trend in decentralised data processing continues. With better memory handling units, faster execution times and better software, medium size jobs may well be in the realm of the next generation of microprocessors. The emphasis is therefore on the control of industrial processes.

In an industrial situation, information taken from or transmitted to a process is usually of three types:

1. Analogue information: for example, voltages delivered by transducer measurements of level, flow, temperature, etc.
2. Logical information: contact closures or digital status contacts (detection of high or low levels, on/off etc.)

3. Digital information: information supplied by digital output measurements which include pulse (flowmeters) and tacho-meter type signals (frequency, angular position, speed, etc.)

The microprocessor can handle these types of information in a variety of functional complexity. From the point of user interaction and system programmability, two general roles can be identified: dedicated and general purpose systems.

2.2.1 Dedicated Microprocessors

Dedicated applications include the microprocessor replacements of hardwired logic, digital filtering and signal processing, and dedicated controllers. They are usually application and I/O-oriented with no or little user programmability.

The advantages of microprocessor logic over analog-based and hardwired components in the context of the processing plant has been discussed by Weissberger[51]. Increased flexibility and performance realised by firmware modules (ROM-based programs), reliability, and reduced storage, power and cooling requirements are listed as among the potential benefits. For example, a microprocessor dedicated to alarm limit checking and data logging could replace analogue recorders and annunciators. Sophisticated analogue equipment such as summing units, multiplexers, multipliers, dividers, lead lag circuits, timers and relays can be implemented as software modules. Furthermore, pre-package routines are available if the user does not wish to indulge in assembly language programming.

The fixed requirement of dedicated control also defines precisely the logic storage needed. Such control designs require only

ROMs and some scratchpad RAM to function. Often, dedicated control jobs can be done with less than 2K words of programmed logic. This has kept packaging requirements and costs to a minimum.

The need for greater flexibility has also led to the implementation of microprocessors as digital filters[52,53]. Although analogue filters have improved from bulky LC circuits to IC devices, they could not offer the advantages created by microprocessor-based designs which include immunity of their responses from changing component values caused by temperature or ageing, availability at very low cut-off frequencies, exactly reproducible responses and compatibility with digital transmission systems. The last point is expected to increase in importance due to the accelerating use of digital techniques in communication systems.

The implementation of a digital filter in a microprocessor is the physical realisation of a difference equation (the transfer function for the process). Microprocessors can easily be programmed to handle this as instruction sets usually have fast add, logical and shift instructions. Alternatively, the programming can be done via LSI hardware. An example of the first is reported by Allen and Holt[54] and a hardware filter fabricated in MOS LSI by Pye TMC has been described by Edwards[55].

The work of Allen and Holt warrants closer attention as it illustrates several aspects which are commonly encountered in microprocessor-based designs. The first consideration is speed which is of course not a restriction in analogue devices. The digital filter, like any sampled-data system, is band-limited in the sense that the range of frequencies it can handle is limited by the Nyquist sampling criterion. Thus, some form of approximation must appear when a digital equivalent of an analogue filter is derived. If filters with large

bandwidths are to be designed, then higher sampling frequencies are required which means programs must execute faster. Thus, on one hand filter characteristics can easily be altered (by changing coefficients in the difference equation), the advantage of flexibility is off-set by processor speed limitation.

The second problem is associated with wordlength. The digital filter was implemented on a M6800 based system incorporating an ADC/DAC and a Motorola Peripheral Interface Adapter (PIA) chip as illustrated in Figure 2.1. Note that the PIA A and B Sides have been programmed into an 8-bit input and output port respectively. A sampling interval of 1 ms is implemented by a software delay loop. At a sampling time, a CA2 control output is used to strobe the sampler which, upon obtaining a sample, holds it for its digital conversion. In the implementation of a low pass filter where the output y(kT) at time kT (T = sampling interval, k ε I where I is the set of non-negative integers) is given by:

$$y(kT) = aTx(kT) + \exp(-aT)y([k-1]T) \qquad (2.1)$$

x(kT) is the input at time kT

a is the inverse of the time constant in the filter transfer function, a multiplication subroutine had to be written as the M6800 has no special instruction for multiplication. Furthermore in an effort to reduce program size and increase the speed of the filter program, the 16-bit products were rounded to one byte causing some random errors in the output.

Also, the decimal coefficients aT and exp(-aT) could only be represented by as multiples of 2^{-7} (0.0078125) as this is the smallest magnitude other than zero for an 8-bit binary representation. These round-off errors not only alter the filter's response but as the complexity of the filter increases, the effect of this rounding can cause

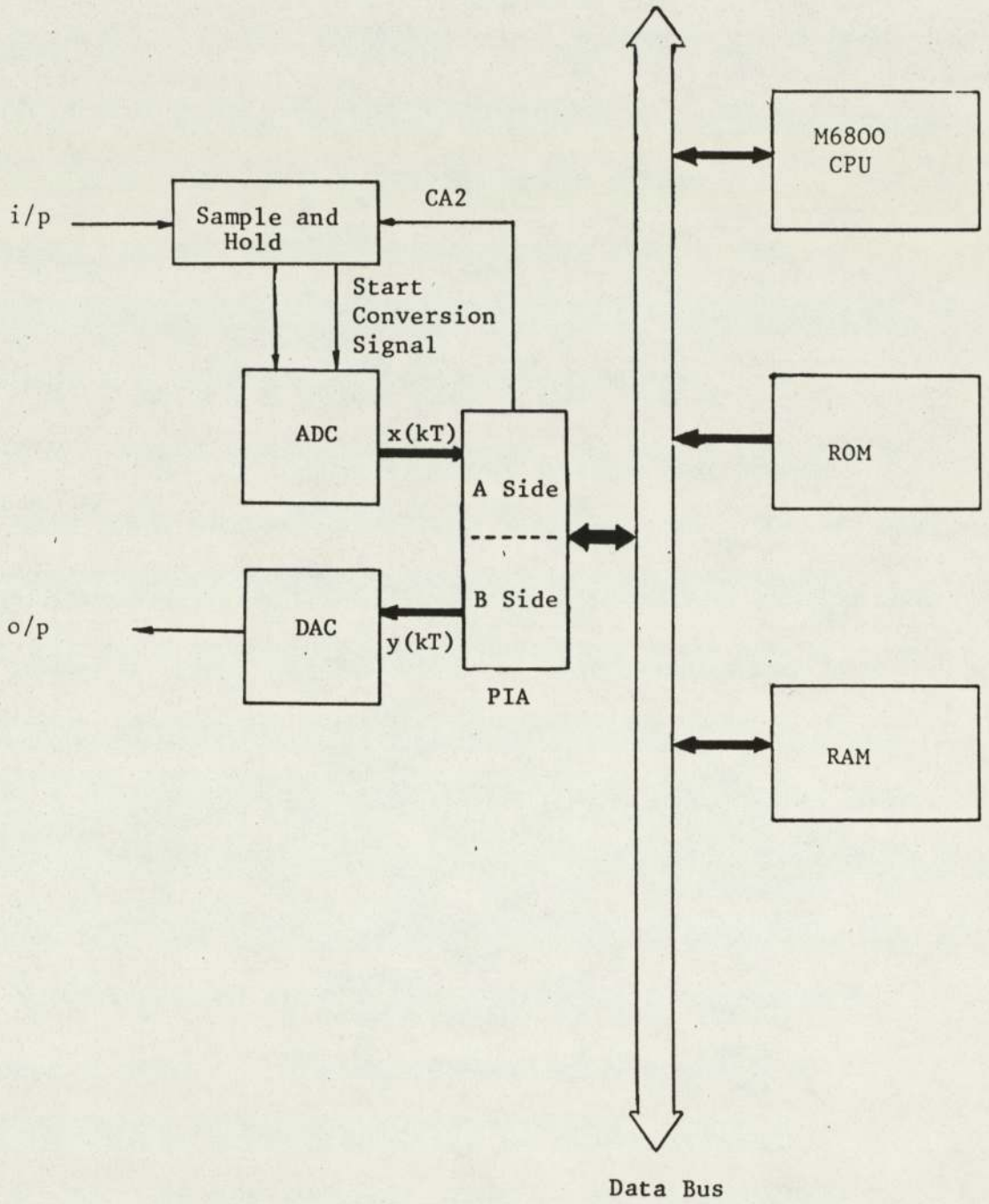


Figure 2.1 Overview of a microprocessor-based digital filter system design (54)

instability (56). Double precision arithmetic can be used but in this case the increase in program execution times makes it prohibitive.

The digital filter example clearly shows a situation where speed and accuracy can become critical factors in the evaluation of a design performance. As far as industrial signals are concerned, direct

microprocessor based signal conditioning and noise filtering is almost non-existent because an external, analogue pre-filter is usually required to eliminate signals above 5 Hz to prevent aliasing and interference from high frequencies which is possible in all sampled-data systems[57]. In data acquisition systems, pre-filtering is usually accomplished using a first-order, low pass hardware (analogue) filter. Any additional filtering is then best achieved by digital (including microprocessor) means.

The low pass filter also suits the nature of industrial processes. Because our ability to control most industrial processes is limited dynamically (and the processes themselves change state slowly because of time lags and delays), we are primarily interested in signals that are meaningful in the low frequency range. High frequency signal variations can therefore be treated as noise and discarded. These are normally electrical pick-ups concentrated in the 60 to 120 Hz range when fluorescent lighting is present and measurement noise (0.5 to 100 Hz) as a result of causes such as turbulence around flow sensors. Process noise, as a result of disturbances due to the process itself ranges from .005 to 1 Hz, rarely higher.

Finally, microprocessors are finding themselves increasingly embedded in various environments as controllers and data handlers. Embedded systems are those incorporating a microprocessor (or microprocessors) to accomplish specific logical or control tasks without which the system could not function. Usually, the product user is not aware of the presence of microprocessor intelligence in his vicinity. Applications of these sort range from microwave oven control to complex data handlers as in CRT terminals and minicomputer peripherals (disk memory controllers for example). Single loop process monitors are also

of this category and because the computing requirements are straightforward, most applications involve the 4- and 8-bit processors only.

An example of the latter is a M6800 based flow monitoring system described by McKay and Gross[58] for use in oil fields and gas processing plants. Sampling is done via a ADC-8S analogue to digital converter and the system's firmware on initialisation accepts orifice factors, transmitter ranges, fluid specific gravities, etc. thus allowing installations in different process environments.

2.2.2 General-Purpose Microcomputers

The general-purpose microcomputer system can be viewed as a system which is fully programmable and usually employs standard peripherals such as VDUs and floppy disk storage. A variety of functions are supported including scientific and business applications to instrument control and industrial automation.

General purpose systems are supplied with high level software options and a host of other support chips. This allows the buyer to shop around for suitable software packages (BASIC, FORTRAN compilers, Assemblers, etc.) to suit a particular application. Programs can be written, run and saved on a floppy disk. Alternatively, the microcomputer can be linked to a mainframe where program development can take place. Then it can be downloaded to the microcomputer for execution. The micro-computer system offers conventional computing resources at a comparatively low cost, making it within the financial reaches of university departments and industrial research establishments. As a result, the situation has spurred many research and development efforts into microcomputer applications to chemical engineering problems. For instance, at a recent Symposium where an earlier version of the software

package developed in this work[127] was also described several applications of commercially available microcomputer systems were reported.

The first describes interactive process flowsheeting implemented in BASIC to run on a 32K Commodore PET[59]. The authors claimed their results have shown that flowsheeting problems of significant size and complexity can be solved on a microcomputer although program execution time is admittedly slow. Another presentation described several applications of an Apple II microcomputer system[60]. Three applications were mentioned: monitoring a lab-scale fixed bed reactor in an attempt to reduce and control SO_2 emissions during the start-up of a sulphuric acid plant, the use of a DISA fibre-optic probe as an instantaneous direct measure of the phase content in a gas-liquid bubbling system and lastly the reading of a photograph of a field of mixed black-white solids produced from a static mixture by a light-wand. Again, application programs were written in BASIC.

Practising process engineers are also experimenting with the benefits of personal computing. In one such exercise bordering on process design, a steady state simulator of a distillation column (up to 5 components and a maximum of 20 theoretical stages) has been written in interpretive BASIC by Sucksmith[61]. The simulator, called MICROCHEM, for a 5 component run using NRTL liquid-activity coefficients and Redlich-Kwong vapour fugacities, used 22K of RAM and took about 5 hours. It indicates that present microcomputers can solve substantial practical problems such as distillation design but do not have the speed needed for more complex applications.

Microcomputing has certainly found a home in chemical engineering. The real benefits are yet to be seen when more powerful

16-bit devices and supporting software become fully integrated. In the mean time, there is considerable interest in implementing advanced control policies and an increasing commitment from industrial controller manufacturers to the need of programmable, microprocessor-based process controllers. The feasibility of applying modern controller designs will be looked at first before going on to a survey of several industrially proven microprocessor-based process controllers.

2.3 Implementation of Advanced Control Algorithms

Unlike the conventional analogue PID controller, the digital controller is easily configured to handle advanced, multi-variable and adaptive control algorithms. The development of the microprocessors has given further impetus in this direction. It is now economically possible to implement sophisticated control strategies[62-64].

One area which has caught the attention of microprocessor control designers is the development of self-tuning controllers. These controllers are designed to overcome the rather subjective tuning difficulties associated with conventional controllers and also to remove the heavy dependence of other digital controller designs on accurate plant models. The self-tuning regulator has been applied successfully in several industrial processes[65-68].

An early feasibility study of microprocessor-based self-tuners was detailed by Clarke et al. in 1975[69]. In a way, it was a response to the original self-tuning regulator of Astrom and Wittenmark[70] of 1973 which stimulated considerable interest as adaptive performance was shown to be possible with a relatively modest computational requirement. However, the design lacked flexibility in that the program was written in a 'medium-level language' which needed a macro-assembler for its generation, and hence was difficult to modify 'on site'. As a result,

a second microcomputer system was built, also based on the 8-bit Intel 8080 processor, to achieve system portability and providing a high-level language suitable for control. In view of the range of parameter values likely in self-tuning applications, the authors decided to represent numbers in floating point format comprising 3 bytes (7 for exponent, 16 for fractional part). As the 8080 has no overflow flag, the eighth bit of the first byte acts as a 'guard' bit for overflows. Hence values in the approximate range $\pm 10^{\pm 19}$ can be stored to a precision of 4½ decimal digits. Clarke and Frost[71] also developed a new high level language called Control BASIC which is used with the self-tuner. The system has been applied to several pilot plants including a batch chemical reactor and effluent pH control.

More recently, Sheirah et al.[72] reported what they described as a Universal Self-Tuning Controller. An initial design was built using 8-bit M6800 microprocessor components but it was found that the configuration limits the accuracy of computations. To improve on the accuracy, the software was re-written in 32-bit floating point arithmetic on an Intel 8085 microcomputer. Floating point numbers are therefore in the range $\pm 3.4 \times 10^{38}$ and the integers in the range $\pm 215 \times 10^{7}$ giving better results but at nearly ten-fold increase in computation time.

The design and implementation of an adaptive, identifier-based, single-loop controller using an Intel 8080A-based Microcomputer Development System (MDS) 800 is found in the work of Baradello[73]. Using a prototyping tool schematically shown in Figure 2.2, microcomputer programs were written on the PDP-11/40 using cross-assemblers and simulators, downloaded into the MDS-800 only for final verification. The software floating-point package used operates on real numbers represented by 3 bytes (1 bit for sign of number, 1 bit for sign of exponent, 6 bits for the exponent and a 16 bit mantissa). Typical addition and

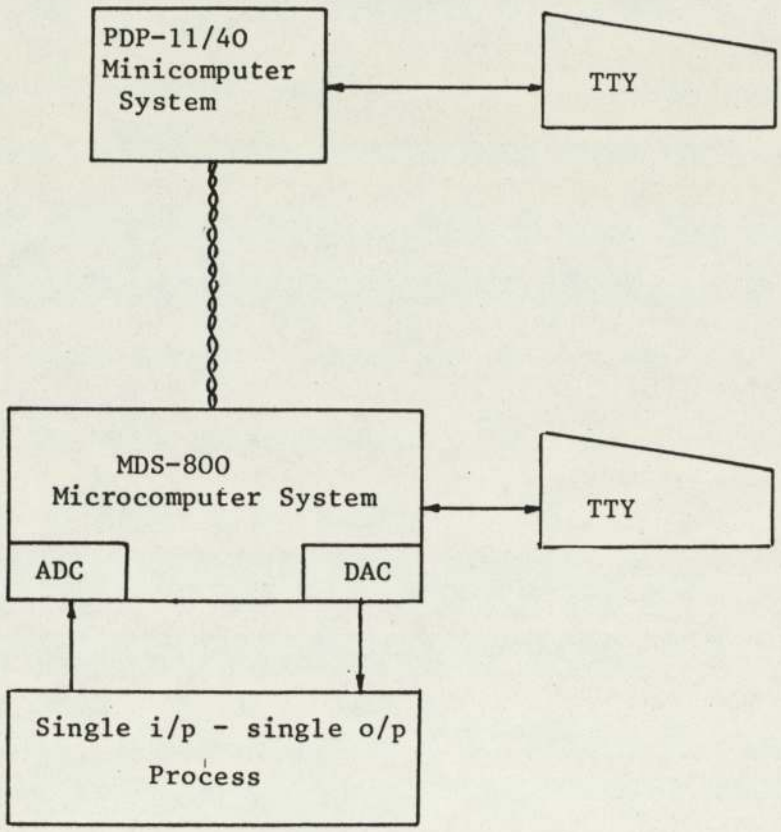


Figure 2.2 Single-loop Controller Prototyping Tool[73]

multiplication required 0.4 and 1.2 ms respectively accurate to ± least significant bit of the mantissa. However, real-time results were limited as the 'chemical process', up to 3rd order, was simulated on an EAI-TR20 analogue computer.

In another research effort, Jensen[74] employed an M6800 microprocessor not only to identify but also to control to a set-point a distributed parameter system consisting of an insulated copper rod with variable heat flux at one end and a measurement at the other. A discrete-time model of the thermal system was integrated into an observer formulation where optimal control is achieved using feedback

estimates of the states. A least square technique was used to perform the on-line identification. Jensen strictly adhered to assembly language programming in formulating both the identification and control algorithms. Subroutines were written to perform 32-bit floating-point arithmetic in conjunction with a hardware multiplier, add and multiply vectors and matrices, for solving system equations using Gaussian reduction, and transposing matrices. According to Jensen, the assembly language route was chosen to break the tradition of under-utilisation of microprocessors by high-level programming which sacrificed speed and memory with no apparent improvement in the quality of the control effort.

2.4 Industrial Microprocessor-based Process Controllers

The relatively slow response of the process control industry in the use of microprocessors in the control of chemical plants has been discussed earlier. Nevertheless, the upward trend in computer control is certainly there. Despite the economic recession, a study of process control equipment markets in the United States[75] indicates a trend of 11.4% in the annual growth rate of process control computers and 32% for programmable controllers for the period 1981 to 1985.

Several reviews of the process controller market have been reported in the literature[36,76,77]. In general, the surveys indicate that the combination of analogue and conventional digital computer control is dominant. An example in the ac_2 (analogue control centre) marketed by Fisher Controls Company, which is designed to sandwich with their dc_2 (digital control centre) and supported by a modified basic programming called pc_2 program. Although each of these packages can cost up to £500,000, Fisher Controls claims to have installed this process controller hardware/software in more than 600 plants with about 1000 loops in diverse industries. For a detailed survey of other industrial products including

their costs range, Reference 77 should be consulted.

More specifically, microprocessor-based controllers have also appeared partly in response to changing trend in control system design and philosophy. Among the earliest to move away from centralised DDC is Honeywell Inc. who first field-tested their Total Distributed Control system TDC-2000 in 1975. The basic unit of the TDC-2000 is the Basic Controller, which comprises eight microcomputers (or computational slots) based upon the 16-bit CP 1600 microprocessor, which can be programmed from a control centre using any one of the 28 algorithms available. These include all functions associated with conventional instrumentation as well as computational devices from 1, 2 or 3-mode controllers with or without cascade or ratio to square root extraction, alarms and auto/manual switch. Currently, the TDC 2000 is an 8-loop shared controller although Honeywell Inc. is believed to be working on a 16-loop version. These controllers can be distributed around the plant as operational substations and linked to a control centre by a data highway. For these reasons, the manufacturer claims that the TDC-2000 is the first complete multimicroprocessor system to incorporate all the necessary aspects in a unified design and could therefore be considered a paradigm of future development in this field. The fact that by the end of 1978 several hundred TDC 2000 systems had been delivered involving more than 20,000 loops is an indication of its popularity.

Current industrial controllers basically offer similar facilities differing only in programmability, special functions and number of loops catered. The Mod III introduced by Taylor Instrument Limited was the first system to introduce an adaptive gain module for controlling non-linear parameters. The ACCO Bristol's microcomputer-based UCS 3000 offers conventional control algorithms as well as advanced

ones. For example adaptive control and optimisation can be implemented by the user. For this purpose, the UCS 3000 is equipped with a 16K RAM, and an EPROM module for the relatively fixed schemes.

However, many of these controller packages may not be suitable for certain process design requirements. In these cases, the companies concerned may prefer to carry out in-house developments using vendor supplied components instead. Monsanto Co. has described a M6800-based single loop controller for a simple temperature control application[78]. The system's control software resides in a 2K byte ROM and process variables are stored in a 128-byte RAM. A watchdog timer circuit shuts down the loop if the controller malfunctions. In its remote control mode, the controller can be addressed by a PDP-11/03 supervisory computer which provides new set-points, alarm-limits and tuning parameters. Another development was applied to pH control[79], always a difficulty with conventional analogue controllers due to the severe non-linearity of the neutralisation curve.

More complex industrial applications have also been reported by Mitchell[80] and Langill, et al.[81]. After having failed to find a suitable system on the market, Mitchell described how the Amoco Texas Refining Co. employed a M6800 microprocessor to control a cooling tower chemical addition and a Texas Instruments 990/4 microcomputer-based control of a cool-down cycle of a refinery coking unit. Langill described how microprocessors were used in a petroleum wax and sugar refinery recovery operations.

2.5 Problems in Using Microprocessors

The literature survey has shown that the microprocessor has brought both the era of opportunities and also of problems. This is because microprocessors are a new technology and for the first time

hardware and software is integrated into a single formation. Users are therefore introduced to the need for different methods of working. The problems of using microprocessors has been discussed by Carter[82]. This award-winning paper identifies the technical, manpower, commercial and sales and marketing aspects as problem areas but the treatment is only general. In fact, it is difficult to be specific as a difficulty encountered in one application may not be the case in another. This section builds upon the perspective generated by Carter's paper by giving some examples reported in the literature survey. The biggest single problem is software and this will be considered first.

2.5.1 Software Aspects

The generation of suitable software has not only been a major difficulty but has also captured the largest share of the costs in many implementations of microprocessor technology. The early users of microprocessors especially found developing and implementing programs difficult and expensive partly because microprocessor hardware design was at its infancy and did not take into account of software requirements and partly because of the lack of suitable software development tools.

Basically, there are three major software bases as described by Crutchley[83] in his four-part tutorial paper on microcomputer control. These are:

- process control language - high-level, user-oriented, requires minimum programming skill.
- standard high level programs - FORTRAN, BASIC, PASCAL, etc. Its effective use requires skilled programmes.
- assembly language - allows efficient use of processor time and memory but

users must learn how to program the particular processor.

Many companies already offer software based on process-control language which usually includes program linking and de-linking capabilities, a library of control and display programs and conversational input routines by which the user can easily build and modify control and other routines. Standard high level software is relatively rare on the industrial microcomputer scene as it usually requires more memory and not economical on 8-bit systems. Although a recent survey[84] of 400 leading American organisations in the instrumentation and control field was reported to have shown that FORTRAN at 50% usage still dominates the language bazaar of industrial multi-user systems (20% use PASCAL, 15% use C-language, 10% use assembly language and 5% others), this standard software is only likely to flood the scene when fully-developed 16-bit microcomputers become available. Until then, many users would have to invest considerable time and money learning to program in assembly language or buy vendor supplied ROM-based modules. In general the nature of the application and user resources dictate the software base to be used.

The literature also suggests that although process inputs (to the microcomputer) of 8 or 12-bit resolution (about 1 and .05% respectively) are generally satisfactory as transducer accuracy is rarely more than that, the internal representation of numbers operating on these inputs often require 3 or 4-byte floating point representation. This is particularly true when 8-bit microprocessors are used to implement advanced control strategies as in the case of self-tuning and identifier-based adaptive controllers. The cost has been increased user

program development and machine execution times. The trend however, is encouraging. Changes in microprocessor architecture and increased availability of supporting hardware chips are expected to create a more efficient high level software base for process control.

2.5.2 Hardware Selection

Hardware selection (microprocessor and supporting ROM, RAM and I/O chips) can be a nightmare if the user fails to calculate his system needs carefully. One of the biggest problems is at what level to start as microprocessor hardware is normally supplied at chip, board and system levels. Chip level is cheap to get started but the user must have the proper resources of design and technical support. Even here, choosing a suitable microprocessor is highly subjective. In particular, there is very little guideline on the relative difficulty in using different microprocessors for process control except, to the author's knowledge, in the research work of Dickey[85]. By using a set of 5 kernel programs said to be representative of computational tasks in process control systems, he developed a method of rating 6 microprocessors (M6800, Intel 8080, MCS 6502, IMP-8, Fairchild F8 and COSMAC 1801) in terms of relative difficulty of programming them in assembly language code. The method is based on measuring the logical complexity of each microprocessor instruction repertoire. Dickey's results, though a limited one, showed that the complexity measure of a microprocessor can be directly determined from a formal instruction set. In this ensemble, the M6800 was found to be the least difficult to program but in practice, other factors such as market rating, software and peripheral support and cost are the deciding factors.

The applications mentioned in this chapter emphasised mainly 8-bit microprocessors. This is partly because they were the first to affect the process control industry in a big way and partly because they are more

readily available with generally acceptable performances. When compared to 16-bit machines, they can be more efficient, on a bit for bit basis, in using memory. Ultimately, the choice of an 8 or a 16-bit processor becomes secondary when what really matters is the total system performance and features available.

2.5.3 System Integration

At the lower levels of the process control hierarchy where the microprocessor is currently more suitable, the microprocessor must be integrated into the analogue environment. The process of translating analogue signals into digital ones (and vice-versa) is likened to a process with some kind of transfer function. Thus, the discretisation process not only resulted in some loss of information but makes digital devices always slower than their electrical analogue counterparts.

Interfacing is another major problem area. The electrical interface is partially solved by making TTL compatible hardware but microprocessor designs have been known to suffer from electrical and radio frequency interferences. In many applications, filters and circuits were located near the sensors to minimise noise communicated to the microprocessor. But the greatest difficulty in microprocessor application design appears to be in the interfaces between the microprocessor and the mechanical parts of the system. Designing sensors and actuators is particularly demanding as it requires an intimate knowledge of how the system must perform. This perhaps explains why many firms have found that training their existing technical people in electronics is more desirable than attempting to bring in people expert in microprocessors and familiarise them with the user's needs.

2.5.4 Economics and Manpower

The main attraction for microprocessors has been the low cost for a given performance. But the overall cost in an application depends on a variety of factors including back-up instrumentation, equipment consolidation, peripherals and manpower training costs. Microprocessor controllers have often been said to be more flexible than conventional PID controllers. If we are talking about choosing between DDC and a PID controller, then the above comparison is not practical. This is because in practice, most computer-controlled plants have computer compatible analogue controllers which take over when the computer is down for maintenance or some other reasons. The situation is thus that microprocessor-based control is not justified on DDC only - the flexibility of DDC is just a handy advantage on going to computer control. The bulk of the credit in microprocessor-based control comes from the extra benefits generated by its programmed facilities. This would include communication with a supervisory computer, a richer loop information and status display or variable trend recording or some other background processing.

Applying microprocessors therefore needs appropriate manpower support if they are to bring in the maximum benefits. Two key figures - the design integrator, a person who knows a lot about the process or nature of the application, and the 'creative programmer' or software designer. Much of the long-term success of an application depends on the efficiency, versatility and reliability of the software created for the application.

2.6 Conclusions

The literature survey has shown that the technological basis for the application of microprocessors is already firmly established. However, the pace in the process control industry where the essence has been "that computers should control and controllers should compute" has been relatively slow although this is expected to change when better software support and 16-bit computers become readily available.

Microprocessors also introduce new difficulties and the need for different methods when developing a system design. Starting from chip level and programming in assembly language get the maximum out of a microprocessor but the user must supply adequate staff support. The learning curve with microprocessor-based designs is steep and it is wise to experiment on the gentler slopes first.

Generally, whenever the microprocessor is being applied, it has shown itself to be a cost effective alternative. However, many outstanding difficulties still remain. These are software development costs, interfacing problems and manpower/microprocessor productivity.

CHAPTER THREE

DEVELOPMENT OF A LINKED HONEYWELL H316-MOTOROLA M6800 TWIN PROCESSOR SYSTEM

3. DEVELOPMENT OF A LINKED HONEYWELL H316 - MOTOROLA M6800 TWIN PROCESSOR SYSTEM

3.1 Introduction

Central to system design and program development in this research are two departmental general-purpose computer systems: a ferrite core 16-bit Honeywell H316 minicomputer system and a semi-conductor-based 8-bit Motorola M6800 microcomputer system. Clearly, the two computers not only differ markedly in size, speed, system organisation and cost but needless to say, each has been a product of a different age in the evolution of computer technology. Originally acquired in the late sixties, the Honeywell 316 minicomputer system has now expanded to include the Honeywell Analogue Digital Input Output System (HADIOS) as its primary data acquisition facility. It is well equipped with supporting software and has been used in both batch scientific and real-time data acquisition and pilot plant control work. The M6800 microcomputer system on the other hand, was acquired in kit form and assembled in the late seventies, has less extensive supporting software and utilisation but nevertheless is sufficiently equipped with additional logic and peripheral interfaces for real-time data acquisition and control work.

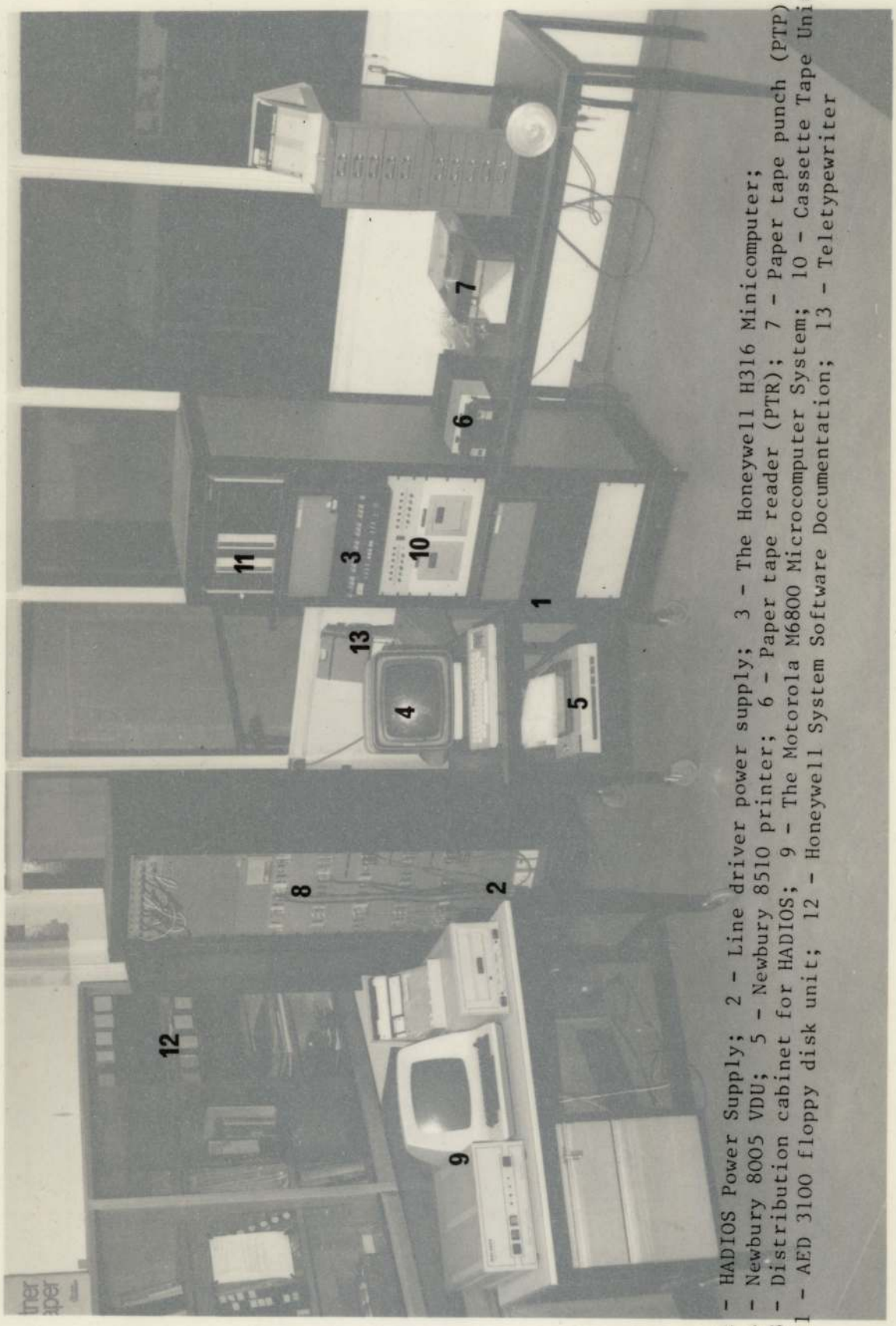
This chapter concentrates on the hardware architecture of both computer systems, the peripheral and interface devices, and finally the linking of both machines to form what will be described as the linked H316-M6800 twin processor system. The design and development of a feasible software protocol for the operation of such a system will be described in the next chapter.

3.2 The Honeywell H316 Minicomputer System

The Honeywell H316 is a second generation digital computer designed for both open shop and batch scientific applications and real-time on-line data processing and control. Its modular design, flexible I/O structure and command repertoire enables it to be tailored to a broad variety of applications both on- and off-line. The departmental Honeywell minicomputer facilities have expanded considerably from its basic configuration in a mainframe, a control panel and an ASR teletypewriter, to include a variety of peripheral devices and the HADIOS which can be linked to pilot plant signal conditioning boxes. The variety of applications has therefore included data reduction and formatting, process control, instrumentation, simulation and batch scientific and engineering computation.

3.2.1 Central Processing Unit and Peripheral Devices

Plate 3.1 shows the Honeywell minicomputer and peripherals as seen in the computer laboratory. A simplified block diagram of the CPU indicating the data storage registers, the control units and the I/O controls is shown in Figure 3.1. The random access memory shown as a single block, is a magnetic core store consisting of four modules of 4096 (4K) 16-bit words. The computer uses two's complement machine code, has a memory cycle time of 1.6 μs and I/O data handling at a maximum word rate of 156 KHz/s. Table 3.1 describes the functional units of the CPU and its I/O controls and Table 3.2 summarises the minicomputer's leading characteristics.



1 - HADIOS Power Supply; 2 - Line driver power supply; 3 - The Honeywell H316 Minicomputer;
4 - Newbury 8005 VDU; 5 - Newbury 8510 printer; 6 - Paper tape reader (PTR); 7 - Paper tape punch (PTP);
8 - Distribution cabinet for HADIOS; 9 - The Motorola M6800 Microcomputer System; 10 - Cassette Tape Unit;
11 - AED 3100 floppy disk unit; 12 - Honeywell System Software Documentation; 13 - Teletypewriter

Plate 3.1. The Honeywell H316 Minicomputer System and Peripherals


LR1

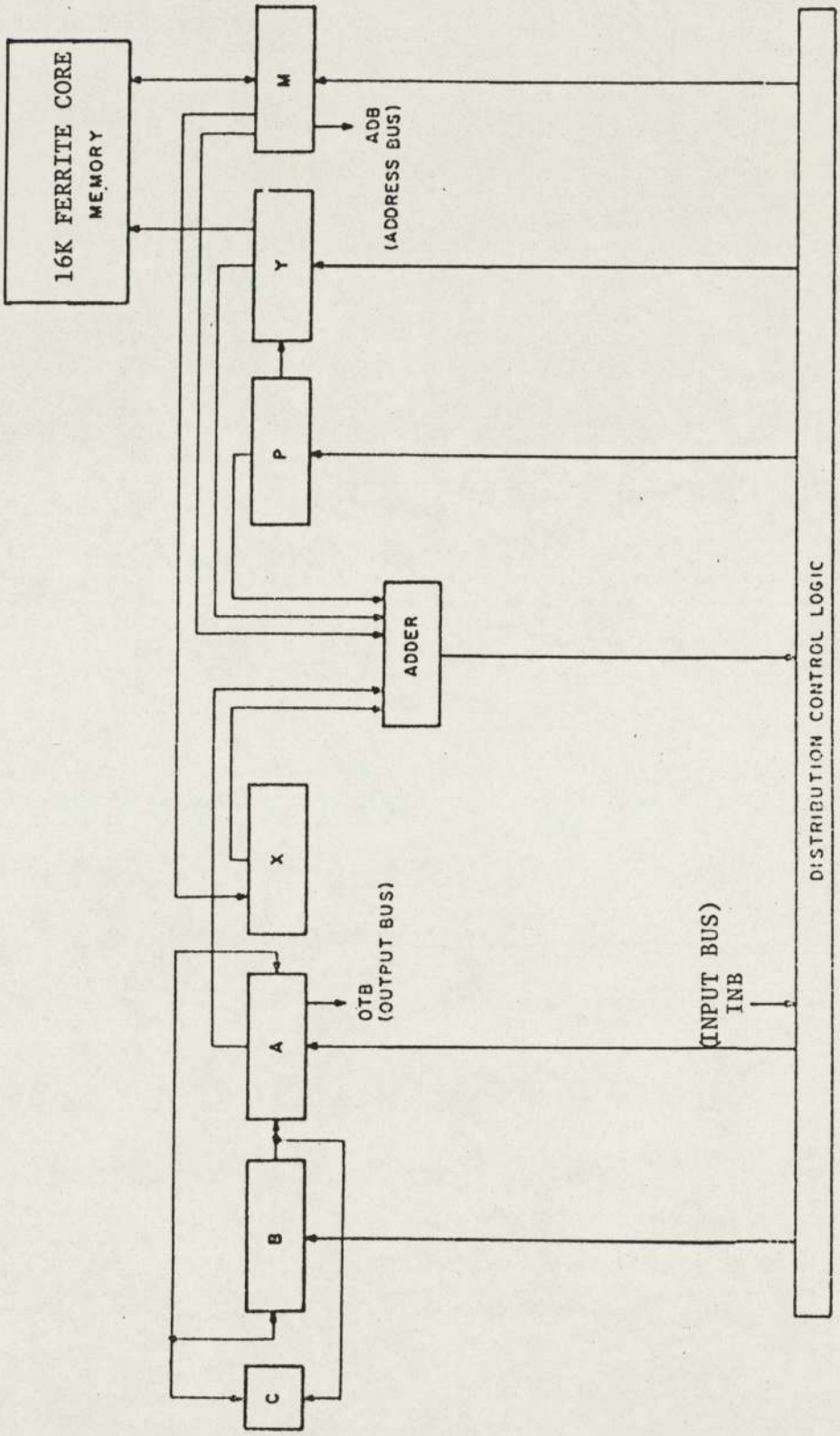


Figure 3.1 Block diagram of the Honeywell H316 Central Processing Unit

Table 3.1 Functional Units and I/O Controls of the Honeywell H316 CPU

A-Register (A)	A 16-bit primary arithmetic and logic register of the computer.
B-Register (B)	A 16-bit secondary arithmetic and logic register used primarily to hold arithmetic operands which exceed one word in length.
Adder	Performs the basic arithmetic processes of addition and subtraction.
M-Register (M)	A 16-bit memory buffer register used to transfer information to and from the core memory.
P/Y-Register (P/Y)	A 16-bit memory address register used to store the address for the memory.
C-bit (C)	A 7-bit indicator associated with the A- and B-registers that stores overflow status resulting from the execution of arithmetic instructions and stores the last bit shifted out of the A- or B-register during the execution of shift instructions.
Index Register (X)	A 16-bit register used for address modification. Usually, any memory write cycle addressing memory location zero also loads the X-register.
Output Bus (OTB)	Sixteen lines that transmit data from the computer A-register to an I/O device.
Input Bus (INB)	Sixteen lines that transmit data from an I/O device to the computer A-register.
Address Bus (ADB)	Ten lines used in conjunction with I/O devices. Bits 7-10 define the function to be performed by the I/O device. Bits 11-16 designate the I/O device to be used.

Table 3.2 The Honeywell H316 Leading Characteristics

Primary power	425 watts, 5.5 amps. at 115 vac ± 10% at 60 ± 2 Hz
Type	parallel binary, solid state
Addressing	single address with indexing and indirect addressing
Machine code	two's complement
Circuitry	integrated
Signal levels	logical ZERO: 0 volts logical ONE : 6 volts
Instruction complement	72 instructions (See Table A3.1, Appendix 3)
Memory Cycle Time	1.6 μs
Speed, Add	3.2 μs
Subtract	3.2 μs
Multiply (optional)	8.8 μs
Divide (optional)	17.6 μs
Standard memory	16K core organised in 512-word sectors
Memory Type	Coincident-current ferrite core
Standard interrupt	single standard interrupt line
I/O Modes	single word transfer single word transfer with priority interrupts
Standard I/O lines	10-bit address bus (4 function code, and 6 device address), 16-bit input bus, 16-bit output bus; external control and sense lines
Environment	room ambient temperature for computer less I/O devices: 0-45°C
Weight (less console)	120 pounds
Dimensions	17.88 in. x 24.5 in. x 14 in.
Cooling	filtered forced air

To date, the Honeywell minicomputer can support the following peripheral equipment:

1. A Tektronics 4010-1 VDU. This unit is capable of both graphical and character display. In the alphanumeric mode, it operates at a rate of 200 baud. Attached to the VDU is a hardcopy device which produces permanent copies of the display when these are required.
2. An ASR teletypewriter operating at 10 characters per second (cps).
3. A high speed paper tape reader operating at 200 cps.
4. A high speed paper tape punch operating at 75 cps.
5. A magnetic tape cassette unit, used for both input and output at a rate of 375 words per second.
6. An AED 3100P floppy disc storage unit.

More recently, several Newbury Model 8005 terminals and Model 8510 desk top dot matrix serial impact printers were acquired by the Department. These products are good examples of the impact of microprocessor technology as both contain ROM and RAM chips to provide greater intelligence, storage, and adaptability than conventional analogue or programmable logic-based devices.

The Newbury terminal has four modes of operations ranging from the standard alphanumeric I/O to graphical input mode. Local display and editing functions are provided to enhance page formatting and labelling. A RS232-based interface adapter with a smaller range of baud rate (110 to 9600) has been built at the minicomputer end to accommodate any one of the Newbury terminals. The terminal emulates the Tektronix VDU but with poorer screen resolution.

The printer is linked to the VDU through its own RS232 port and can also serve as a data communication terminal and a hardcopy unit

for VDU displays (for example, Graphics). Printing speed is 100 cps.

3.2.2 System Software

The definition of system software here is extended to include not only those programs supplied by the computer manufacturer (utilities, monitors and control programs) to control the operation of the computer but also any other standard software package provided by the computer manufacturer, instrument and interface manufacturers or organisations specialising in software. These may include language compilers, object loaders, I/O and library subroutines and operating systems. As a rule, system programs do not exactly meet the requirements of a specific computing objective so application programs are written, very often by the user himself. Application programs may or may not incorporate modified versions of available system software packages.

FORTRAN Compiler

The Honeywell FORTRAN IV compiler has been produced for 16-bit computers according to the American Standards Association specifications.(86) Normally, source programs are prepared using the Honeywell Text Editor in standard format. Object code output is normally directed to paper tape punch in relocatable mode.

DAP-16 MOD 2 Assembler

To avoid programming directly in machine code, a symbolic assembler, DAP-16 MOD 2, is provided by the manufacturer.(87) The assembler is a 'one for one' language, i.e. one symbolic instruction corresponds to one machine code operation, except in the case of pseudo-operations, which request action by assembler rather than

specifying an operation code. Assembly language programs are invaluable tools in writing real-time software packages or patching existing system programs to meet user/application requirements. Paper tape source programs are prepared using the Text Editor and usually assembled in the two-pass mode. Assembler and FORTRAN object code outputs are fully compatible. Tables A3.1 and A3.2 in Appendix 3 summarise the DAP-16 MOD 2 assembler language mnemonics and pseudo-operations respectively.

Object Leader

The object code produced by the DAP-16 assembler or FORTRAN compiler is processed by the object loader to form a core image in memory. As an instruction word of 16 bits requires 4 bits to represent a sufficient number of operands and a further 2 bits for indirect addressing and indexing, the maximum number of locations that can be accessed by direct addressing is 2^{10} (1024) or two sectors - the current sector of the instruction and usually the lowest sector in memory (base sector). By indirect addressing, the maximum number is 2^{14} (16384) as the index and indirect addressing bits are still required. This limitation means the user must have a reasonable assessment of cross-sector links acquired for a successful loading.

The loader therefore operate in two modes. In the desectorising mode, the loader handles all intersector references by generating indirect address links where necessary. These links are usually located in base sector, unless the assembler program specifies a location elsewhere by the SETB (Set Base) pseudo-operation. In the load mode, the loader assumes all intersector links are handled by the assembler program.

A successful loading generates a memory map indicating memory utilisation and program segment length and location. The core image may then be punched out as a self-loading system tape (SLST) using a

punch utility PAL-AP, stored on paper tape, cassette or stored on a floppy disc.

It was envisaged at the early stage of program development that much of the software to be used will occupy low and high sectors of H316 memory. A suitable self-contained loader, LDR-APM Rev. E, occupying locations '13050 through '16577, was therefore constructed. (See Section 3.1 of Appendix 3.)

BASIC-16

BASIC-16 is the Honeywell version of BASIC for 16-bit processors with memory size 4K or more. Interpretative in operation, the compiler provides the user with an interactive, problem-orientated high-level language. In standard form, communication with BASIC is from the teletype but a machine code modification permits I/O via the paper tape reader and punch.

An important refinement provided in BASIC-16 is the CALL statement, which enables up to 10 FORTRAN/DAP-16 subroutines to be accessed from a BASIC program. The general form of the statement is

$$\ell n \text{ CALL } (sn, a_1, a_2, \ldots, a_n)$$

where ℓn is the statement line number

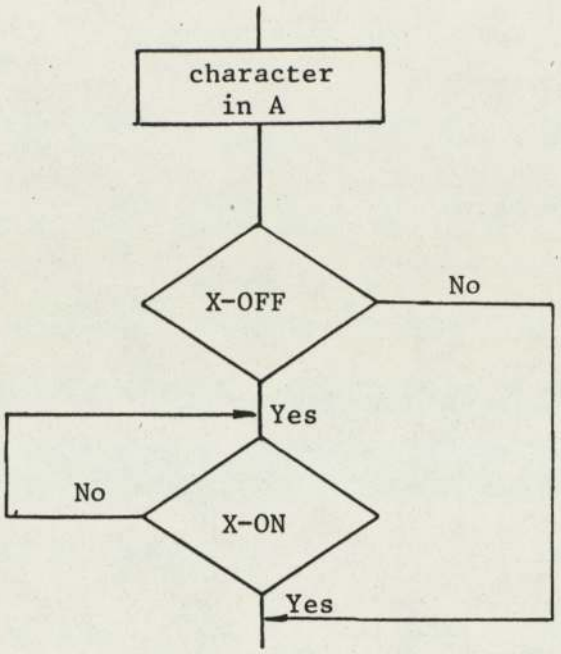
CALL is the statement operator

sn is the subroutine reference number (1 to 10)

a_1 to a_n are arguments to be passed to the subroutine called.

3.2.2.1 Machine Code Patch for use with the Newbury terminal and printer

A software patch was found to be necessary for satisfactory use of the Newbury terminal and printer. The flowchart shown below, puts the H316 in a waiting loop on sensing a 'VDU busy' signal from the



terminal. This 'busy' signal (X-OFF character) can be initiated by the printer if its input buffer is full. On receiving a 'ready' signal (an X-ON character) the H316 exits the waiting loop to proceed with normal program execution. In this way, no information is lost during printing. An example of its use with the H316 Text Editor is shown below.

```
Location          Mnemonic                    Machine code

'11270          TEST DAC    **              000000
'11271               STA    TEMP            04 1 311
'11272               OCP    '4              14 0 004
'11273               INA    '1004           54 1 004
'11274               JMP    *-1             01 1 273
'11275               CAS    XOFF            11 1 312
'11276               SKP                    100000
'11277               SKP                    100000
'11300               JMP    CONT            01 1 307
'11301               INA    '1004           54 1 004
'11302               JMP    *-1             01 1 301
'11303               CAS    XON             11 1 313
'11304               SKP                    100000
'11305               JMP    CONT            01 1 307
'11306               JMP    *-5             01 1 301
 11307          CONT LDA    TEMP            02 1 311
'11310               JMP*   TEST            -01 1 270
'11311          TEMP BSZ    1               000000
'11312          XOFF OCT    223             000223
'11313          XON  OCT    221             000221
```

3.2.3 The Honeywell Analogue Digital Input Output System (HADIOS)

The HADIOS hardware basically consists of a controller, connected to the I/O data and control lines, which generate subsidiary data, addresses and controls for up to 15 different subinterface cards. These subinterfaces can be analogue or digital devices and input or output. As Figure 3.2 schematically shows, the HADIOS is the main interface between the computer and a wide range of I/O devices in on-line applications. The HADIOS devices used are as follows:

3.2.3.1 High Level Analogue Inputs

This subinterface consists of a single channel analogue to digital converter (ADC), with a maximum conversion rate of 40 kHz, and connected to three 16-channel multiplexer units. Hence, there are 48 analogue inputs and these are numbered from 0 to 47. Figure 3.3 illustrates this schematically. The input signals (0 to 5 volts) are converted to a binary integer with ten bits resolution i.e. 0 to 1023.

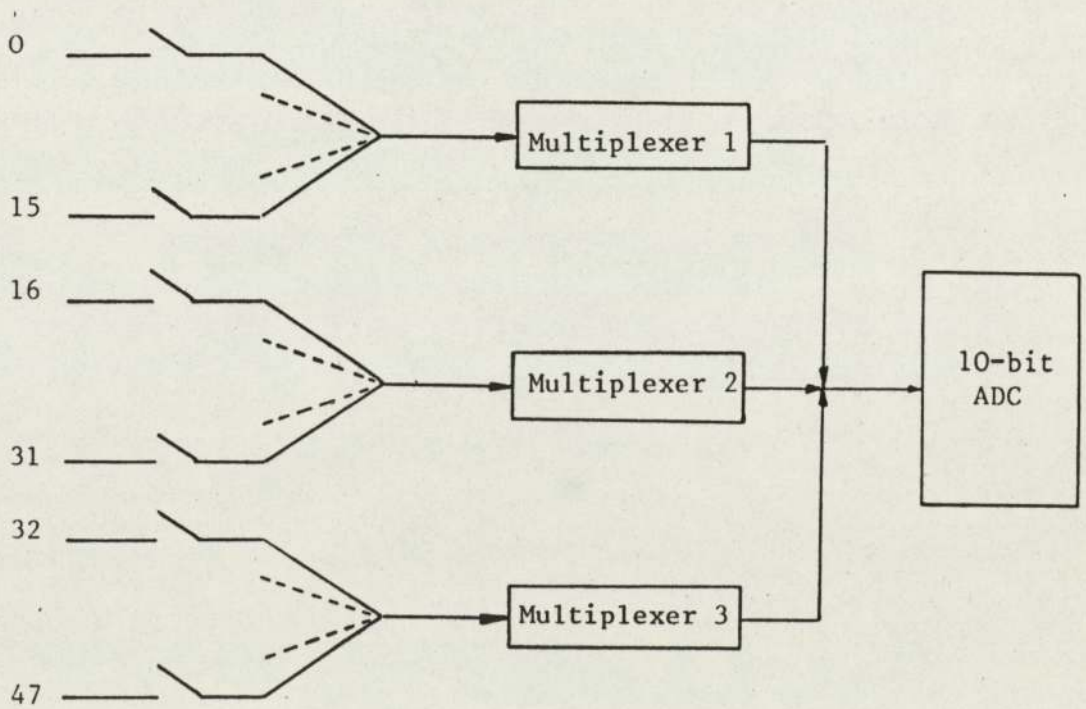


Figure 3.3 The Analogue Inputs Subinterface

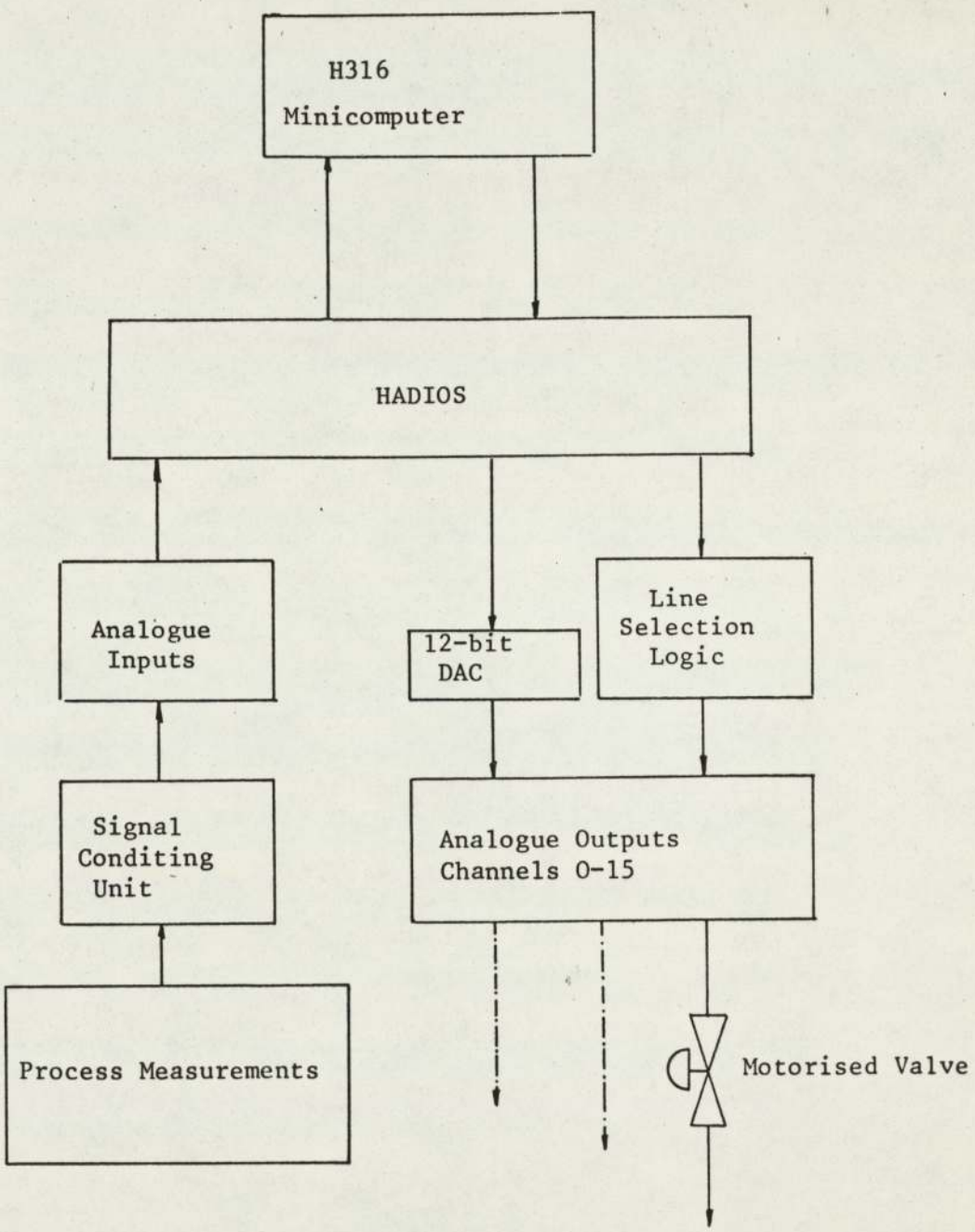


Figure 3.2 A Schematic Diagram of the Computer-Process Hardware Interface

3.2.3.2 High Speed Counter Inputs

A counter input provides a method of monitoring the number of changes of level of a digital input. The counter is incremented by a high active transition (a voltage transition from logic '0' to logic '1'). The current contents of a counter can be attained by a programmable command. Each counter can also be initialised to a preset value and programmed to interrupt the H316 central processor when half-full.

Each counter input subinterface card has an 8-bit register with a range of 0 to 255. When this subinterface is operated in the non-interrupt mode, the 8-bit register automatically returns to zero when a count of 255 is reached whereas in the interrupt mode when the register is half-full i.e. it contains 127, an interrupt request to the CPU is generated. There are 3 subinterface cards in service at present.

3.2.3.3 Logic-Level Non-Isolated Inputs

There are two digital inputs subinterface cards namely digital input A and B (DGIA and DGIB) respectively. Each card senses the voltage levels present on the 16 parallel user's input lines. The input signals are usually low impedance voltages switchable from logic '1' to logic '0'. The signals sensed are transferred to the A-register as a 16-bit pattern. By experimentation, it has been found that if any of the digital input lines are non-active from the user's point of view i.e. these lines are in open-circuit conditions, then the corresponding values in the A-register turn out to be logic '1's.

3.2.3.4 Logic-Level Outputs

The two digital subinterface cards A and B (DGOA and DGOB) provide a means of transferring digital data to peripheral equipment

external to, but near, the central processor. The data is held in MSI flip-flops in the subinterface cards and hence, remain valid until new data are output from the computer.

Either card can be used in conjunction with the digital to analogue converter (DAC) available. Sixteen analogue output channels (numbered 0 to 15) therefore provides a facility for outputting analogue voltages in the range 0 to 10 volts. However, each analogue output has the digital equivalent of only the 10 most significant bits of the original value in the A-register. This is because the six least significant bits are used mainly to make room for addressing up to 16 output channels.

3.2.3.5 Alarm Inputs

In general, the alarm inputs subinterface card channels externally-generated interrupts into the standard interrupt line of the H316 computer. There are 16 input channels on the card and the unit was configured to cause an interrupt request of the H316 on the occurrence of a low active transition (logic '1' to logic '0') on any one of the 16 input channels. When an interrupt is acknowledged, the state of the 16 inputs must be transferred to the computer A-register and a software sort carried out to find which particular input of the 16 channels has become active. The non-active lines (open-circuit conditions) register themselves as logic '1's in the computer A-register.

Summary of HADIOS Devices

To summarise the HADIOS then, the following facilities are provided:

48 analogue inputs connected to one ADC via three multiplexers

3 counter inputs

2 digital inputs

2 digital outputs, including one DAC giving 16 analogue outputs

1 alarm inputs unit providing 16 channels

3.2.4 The H316 Interrupt Structure

The H316 minicomputer has a powerful interrupt/break structure (a break is also hardwire-controlled but unlike an interrupt, it occurs between instructions or cycles of an instruction without affecting the contents of the program counter) which consists of nine levels. If two or more sources assigned to different levels request memory access simultaneously, they are executed in a priority sequence determined by hardware. Of main interest to this research is Level 6, the standard interrupt, to which most peripheral devices are connected by means of the priority interrupt line, PILOO, of the I/O bus.

When an I/O device or some special application hardware request service by forcing the party line PILOO to a particular state, an interrupt request is said to be present on the line and an interrupt request is said to be present on the line and an interrupt is pending. To generate an interrupt, however, the central processor must have executed an ENB (Enable Interrupt) instruction. Although, several sources may request interrupts simultaneously, only one interrupt is allowed at any one time.

The programmer can also control which devices request interrupts. This is done by setting the particular device interrupt mask corresponding to the A-register via the SMK '20 instruction. Location '63 is the dedicated location for the standard interrupt system and contains the interrupt vector.

For a device to cause an interrupt then, the following conditions must be met.

(i) The device must be ready.

(ii) The interrupt mask flip-flop must be set.

(iii) System interrupt must be enabled by an ENB instruction.

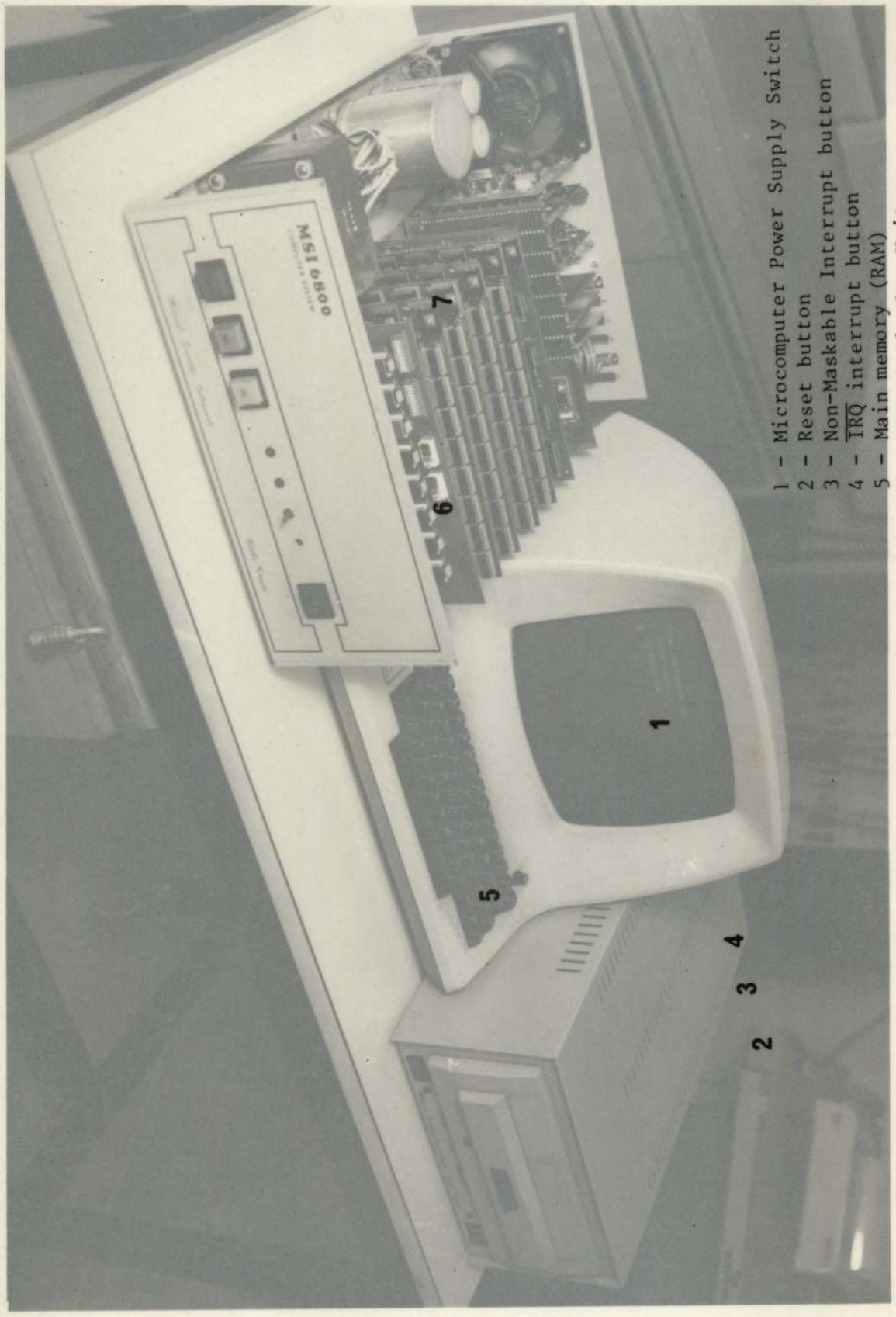
3.2.5 The H316 Real-Time Clock

The Real-Time Clock of the H316 minicomputer increments dedicated location '61 at a constant rate independent of the prime power source. Several frequencies (5 to 20 ms continuous) are available but a 20 ms frequency has been used throughout this work. When the contents of the counter pass from '177777 to '000000 and the device mask is set, an interrupt request is generated through the standard interrupt location '63.

The clock can operate in a non-interrupt mode and can therefore be used as a time base (i.e. real-time) in many applications. In the interrupt mode, the programmer can cause execution of his program either cyclically at a specified interval or at a specified time. Such a scheme is the basis of the operation of the HADIOS real-time executive program to be described in Chapter 4.

3.3 The Motorola M6800 Microcomputer System

This departmental microcomputer system started off from the Midwest Scientific Instruments M6800 Computer System Kit (which includes Chassis and Hardware, Power Supply, Mother Board and Connectors, MPU Board and Monitor, 8K RAM Memory Board, Interface Adapter Board and Serial Interface Board, all for £375 including V.A.T. in 1977) and now has a variety of supporting devices and software packages such as to become a general purpose microcomputer. Plates 3.2 and 3.3 show in varying detail the hardware of the M6800 microcomputer system.



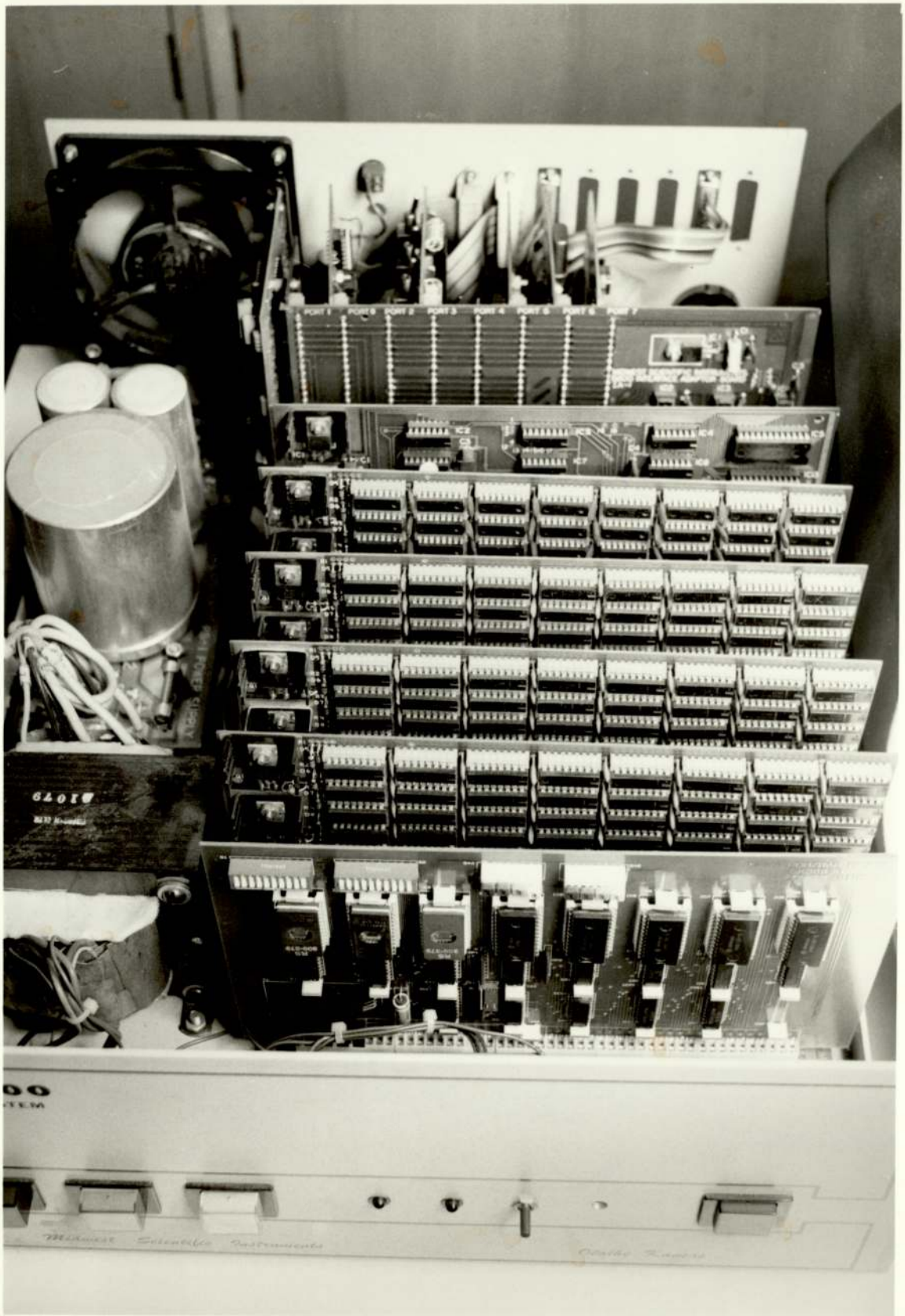
1 - Microcomputer Power Supply Switch
2 - Reset button
3 - Non-Maskable Interrupt button
4 - $\overline{IRQ}$ interrupt button
5 - Main memory (RAM)
6 - Visual Display Unit
7 - FD-8 Floppy Disk Unit

Plate 3.2. The Motorola M6800 Microcomputer System

MSI 6800

1 - M6800 Power Supply switch
2 - Power Supply (Voltage) Regulator
3 - Main Memory bank (RAM)

Plate 3.3 The M6800 System Kit (Assembled)

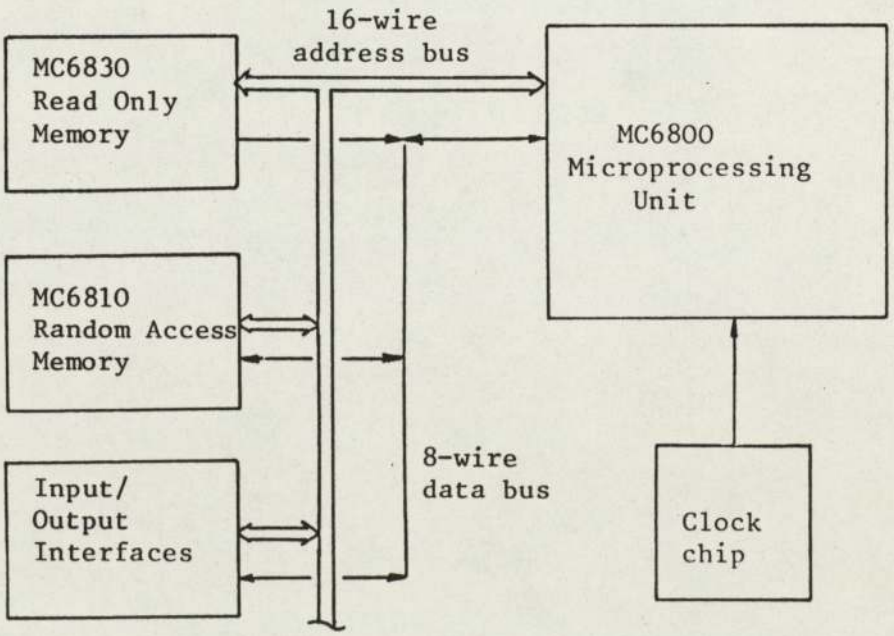


3.3.1 System Overview and Hardware Features

The basic Motorola M6800 microcomputer family consists of five parts:

1. The MC6800 microprocessing unit (MPU).
2. MC6830 masked programmed Read Only Memory (ROM) (1024 bytes of 8-bit each).
3. MC6810 Static Random Access Memory (RAM) (128 bytes of 8 bits each).
4. MC6820 Peripheral Interface Adapter (PIA) - for parallel data I/O.
5. MC6850 Asynchronous Communications Interface Adapter (ACIA) - for serial data I/O.

As shown in Figure 3.4, a complete microcomputer can be built by interconnecting the system components via a 16-wire address bus, an 8-wire data bus, and a 9-wire control bus, plus a clock so that the computer can function in a timely and orderly manner. The modularity of family components and the system's bus-oriented architecture allow easy configuration for a wide range of applications.



(Contol Bus not shown.)

Figure 3.4. The Motorola M6800 family components

For a component to be a member of the M6800 microcomputer family and to ensure compatibility, it must meet specific system standards. The standards must also make it convenient for other external devices to interface (or communicate) with the microprocessor. These standards, which apply to all M6800 components are as follows:

8-bit bidirectional data bus

16-bit address bus

3-state bus switching techniques

TTL/DTL level compatible signals

5 volt N channel MOS silicon gate technology

24 and 40 pin packages

Clock rate 100 KHz to 1 MHz

Temperature range of 0 to 70^{o}C.

The temperature range chosen is adequate for most industrial and commercial applications and is also the same as the standard TTL range.

The Data Bus

Since the basic word length of the M6800 is 8-bits (one byte), it communicates with other components via an 8-bit data bus. The data bus is bidirectional, and data is transferred into or out of the M6800 over the same bus. To accomplish this, a Read/Write line (one of the control lines) is provided.

An 8-bit data bus can also accommodate ASCII characters and packed BCD (two BCD numbers in one byte).

The Address Bus

These 16 output lines are used to address (through data generated by the MPU) devices external to the MPU and they are chosen for the following reasons:

1. For programming ease, the addresses should be multiples of 8-bits.
2. An 8-bit address bus would only provide 256 addresses but a 16-bit bus provides 65,536 distinct addresses (hexadecimal 0000 to FFFF) which is adequate for most applications.

Three-State Bus Switching Techniques

A typical digital line is normally either HIGH (normally at a logical '1' level), or LOW (normally at a logical '0' level). However, since microcomputer components are in general bus oriented in architecture, (a component chip hanging off the system bus, so to speak) and therefore share a common bus, three-state bus technology is necessary to allow one, and only one, selected component to drive the bus at any one time. This is done via the Read/Write ($R/\overline{W}$) Valid Memory Address ($\overline{VMA}$) control lines in conjunction with the address bus which make all unselected components force their three-state bus drivers to the high-impedance state (i.e. the third state). The transfer of information via the data bus can therefore be bidirectional over the same 8-wire bus.

5 Volts N Channel MOS Silicon Gate Technology

The M6800 requires only a single +5 volts supply for its operation. The use of a single standard voltage gives it a significant advantage over other microprocessors and MOS devices that require several different voltages for their operation.

3.3.2 The MC6800 Microprocessing Unit

The MC6800 microprocessor is the nucleus of the M6800 microcomputer system and is enclosed in a 40-pin package. As the block

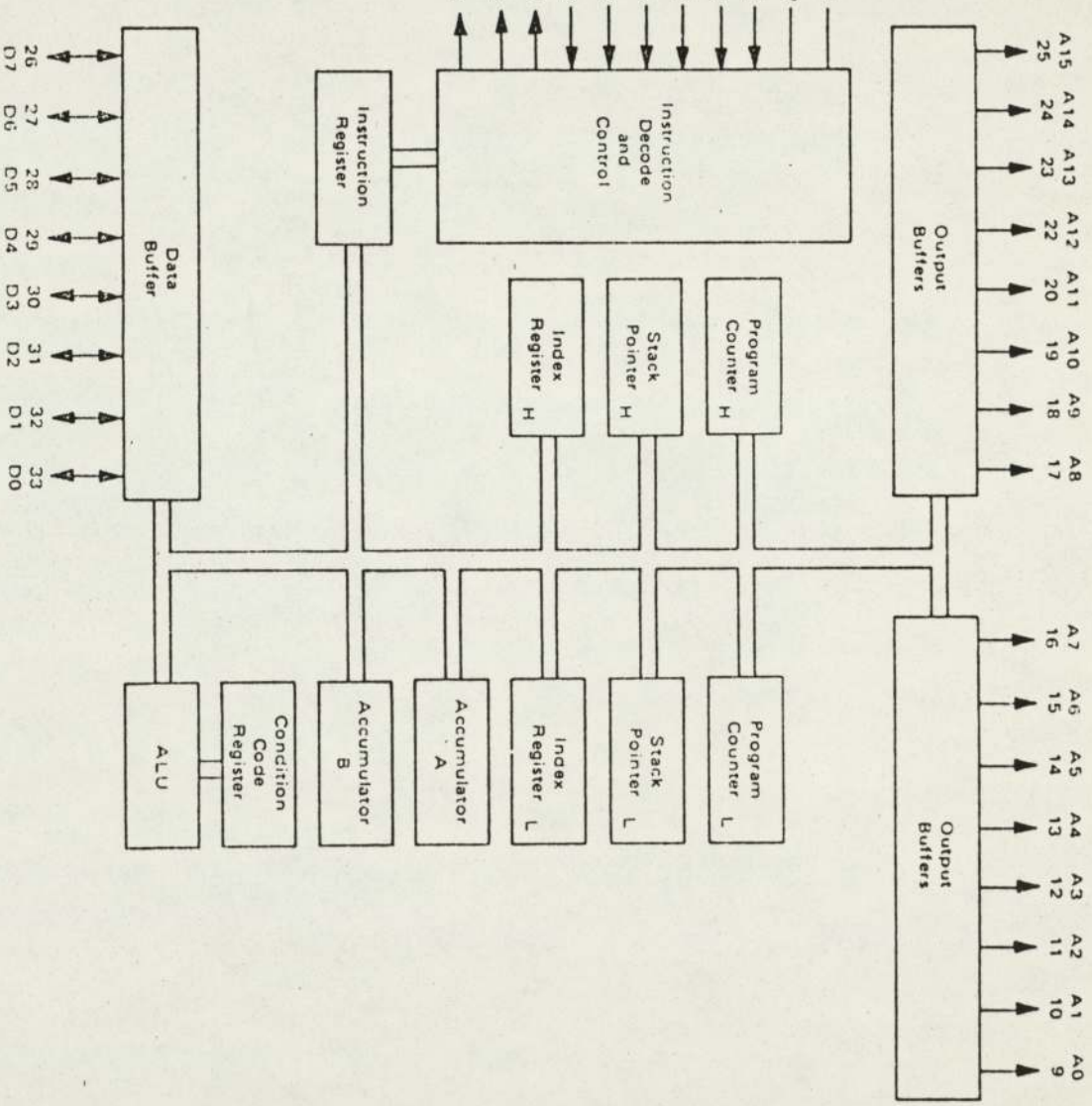


Figure 3.5 Block Diagram of the MC6800 Microprocessor[89]

diagram of the MPU in Figure 3.5 shows, the various internal registers are inter-connected to the instruction decode and control logic via an 8-bit internal bus. The figure also shows the nine control lines that communicate with the external devices, the address bus at the top and the data bus at the bottom.

The Arithmetic Logic Unit (ALU) and Hardware Registers

The Arithmetic Logic Unit of the MC6800 is an 8-bit, parallel processing, two's complement device. It includes the Condition Code (or processor status) Register.

Figure 3.6 shows the significance of the bits of the condition Code Register. The register is used by branch instructions to determine whether the MPU should execute an instruction located at some other address other than next in the sequence. For a detailed description of how each bit of the Condition Code Register is affected by branch and other instructions, the full instruction set of the M6800 in Table A3.3 of Appendix 3 should be consulted.

1	1	H	I	N	Z	V	C
7	6	5	4	3	2	1	0

C - Carry/Borrow

V - Overflow (two's complement)

Z - Zero result

N - Negative (bit 7 = 1)

I - Interrupt Mask

H - Half-Carry (bit 3 → bit 4)

Note: Bits 6 and 7 are not used and always set to a "1".

Figure 3.6 Condition Code Register

The other five internal registers which the user must be concerned with, are as follows:

A Accumulator (A)

B Accumulator (B)

Index Register (X)

Stack Pointer (SP)

Program Counter (P)

A Accumulator - An 8-bit register used as a temporary holding register for MPU operations performed by the ALU.

B Accumulator - As register A. Registers A and B allow operand to remain in the MPU. Instructions that can be performed using both accumulators (e.g. ABA, TAB and SBA) are therefore very fast as they do not require additional cycles to fetch the second operand.

Index Register - A 16-bit (two bytes) register, implemented with a high (H) and low (L) byte. It is primarily used to modify addresses when the indexed mode of addressing is employed. As with the A and B accumulators, the index register can be incremented, decremented, loaded, stored, or compared.

Stack Pointer - A 16-bit register, also implemented with a high (H) and low (L) byte, that contains a beginning address, normally in RAM, where the status and the register contents of the MPU during an interrupt, or the return address in executing a branch or jump to a subroutine instruction (BSR or JSR), can be stored.

Program Counter - A 16-bit register that contains the address of the next byte of the instruction to be fetched from memory. The program counter is automatically incremented by one when its current value is placed on the address bus.

System Clock

The M6800 microcomputer requires a two-phase non-overlapping clock capable of operating from the 5V system power. The clock synchronises the internal operations of the microprocessor, as well as all external devices on the bus. In this system, the MC6875 chip provides the two timing pulses Phase 1 and Phase 2 (ϕ_1 and ϕ_2) at 2 MHz.

3.3.3 Read Only Memory (ROM)

Read Only Memory for the M6800 microcomputer comes in 1024 (1K) eight-bit bytes, mask programmable MC6830 chips. Its bus organisation is shown in Figure 3.7.

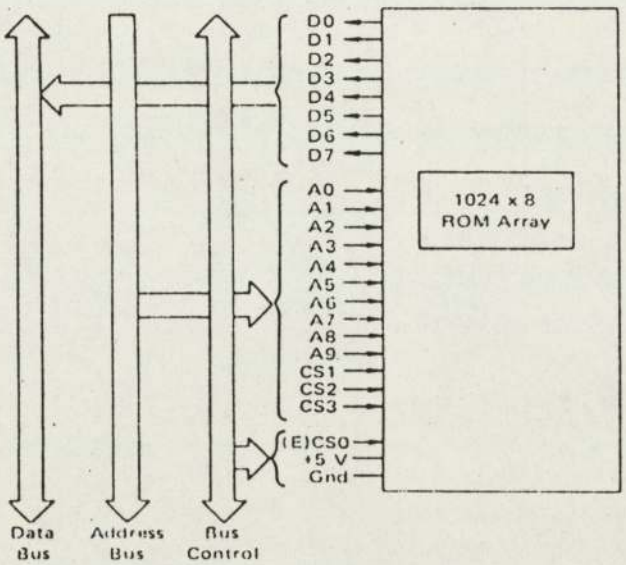


Figure 3.7 The MC6830 ROM Bus Interface (89)

Included are ten address lines, eight data lines and four chip select lines. Because this is a ROM, no R/$\overline{W}$ line is needed. The ten address lines are used by the MPU to select an eight-bit byte on the particular chip addressed. All ROM chips share the same ten address lines so chip select lines are used (defined by user and manufactured into the device) to decode individual chip address.

ROM units have specified memory access time of 700-900 nano-seconds and normally used to store permanent software. In this microcomputer system, a 1K ROM unit is used to house the MIKBUG monitor. A current version of FORTH has recently been programmed into a 4K EPROM memory thus enabling the user to access an efficient high-level language without the use of the floppy disk.

3.3.4 Random Access Memory (RAM)

The standard Random Access Memory chip for the M6800 microcomputer is the 128-byte, read/write MC6810. Its bus organisation is shown in Figure 3.8. Included are seven address lines, eight data lines, six chip select lines and a R/$\overline{W}$ line. The seven address lines are used by the MPU to select an eight-bit byte on the particular chip addressed. As in the case with ROMs, all RAM chips share the seven address lines and the chip select lines which are connected to other address and control lines, ensure that only one RAM chip is communicating with the system bus at any one time.

Because of its read/write nature, RAM chips provides the main memory for program and data storage. However, unlike Read Only Memory, RAM memory is volatile and all stored information is lost when the system power supply is off. For this reason, a floppy disk storage unit has been acquired for the M6800 microcomputer to house system and user-written software.

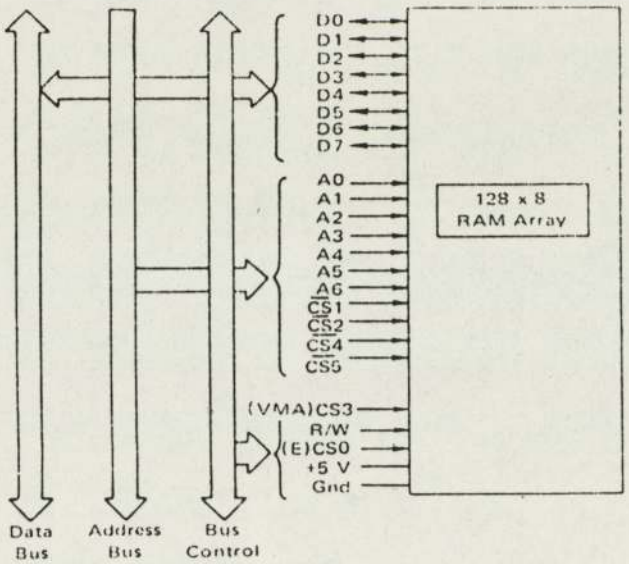


Figure 3.8 The MC6810 RAM Bus Interface[89]

3.3.5 Peripheral Interface Adapter (PIA)

The MC6820 Peripheral Interface Adapter chip is used to control parallel data transfers between the M6800 system and nearby external devices. A block diagram of a PIA is shown in Figure 3.9 which indicates that the PIA is a dual I/O port unit (a port is defined as an 8-bit parallel interface) with two similar sides labelled A and B. Each side contains three registers.

1. The Control/Status Register - controls the operation of its side of the PIA.
2. The Data Direction Register - determines the direction of data flow (input or output) on each I/O line.
3. The Peripheral Data Register - holds the I/O data going between the external system and the PIA.

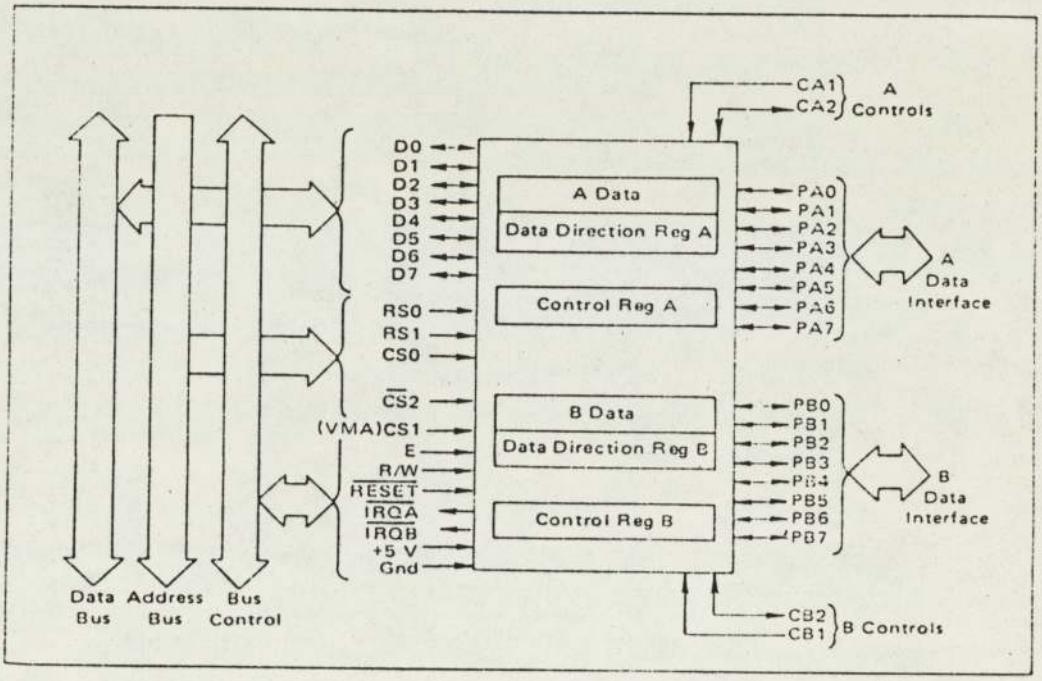


Figure 3.9 The MC6820 PIA Bus Interface (89)

These PIA registers are treated as a set of memory locations by the M6800 system. Data can be read or written into these registers as with any other memory location. In fact, the programmable nature of the PIA helps create a powerful and flexible I/O facility for the M6800 system, warranting a more detailed description of its operation. This can be subdivided into PIA interface lines, operating modes and initialisation.

(a) PIA Interface lines

The most important signal functions the user must be concerned with, are summarised in this section.

Peripheral Data Lines PA0-PA7 and PB0-PB7

Each of these eight data lines can be programmed to act either as an input or an output by setting a '1' in the corresponding bit in the data direction register if the line is to be an output or a '0' if it is to be an input.

Data Lines (D0-D7)

These eight bidirectional data lines permit transfer of data to/from the PIA and the MPU i.e. through the same data bus that the ROMs and RAMs share.

CA1 and CB1 Input Lines

These lines are only inputs to the PIA and set the interrupt request flags (bit 7) of the control registers (see Section on PIA Operating Modes). For this reason, CA1 and CB1 are often described as interrupt input lines.

CA2 and CB2 Control Lines

Each of these control lines can be programmed to act as either a peripheral output or interrupt input. The function of these lines is programmed with the respective control registers (bits 3, 4 and 5).

$\overline{\text{IRQA}}$ and $\overline{\text{IRQB}}$

These two low-active lines are channelled together into the standard $\overline{\text{IRQ}}$ line and interrupt the MPU through the interrupt priority circuitry. In principle, if CA1, CA2, CB1 and CB2 all act as interrupt inputs, each PIA can generate four different $\overline{\text{IRQ}}$ interrupt requests to the MPU. Each input sets up its corresponding interrupt request flag

in the respective control registers (bit 6 and 7) and the MPU must carry out a software sort to determine a particular source. The interrupt request flags (bits 6 and 7) are cleared when the MPU needs the peripheral data register or on hardware reset.

(b) PIA Operating Modes

The operating mode of the PIA depends on the organisation of the control register words summarised in Table 3.3.

The PIA, like all the system component chips discussed thus far, is accessed via the chip select lines which are connected to the address lines of the MPU. The two register select lines (also connected to the MPU address lines) are than used to select one of the six internal registers. However, two input lines can only select one in four registers uniquely. This problem is conveniently solved by the logical status of bit 2 of the control registers. If bit 2 is a '1', then the peripheral data register is addressed. If bit 2 is a '0', then the data direction register is addressed. In fact, this is the only purpose of bit 2 of the control register.

Table 3.3 Organisation of the PIA Control Registers

	7	6	5 4 3	2	1 0
A Side	IRQA1	IRQA2	CB2 Control	DDRA or IORA	CA1 Control
	7	6	5 4 3	2	1 0
B Side	IRQB1	IRQB2	CB2 Control	DDRB or IORB	CB1 Control

The general purpose of each of the remaining bits is summarised as follows:

Bit 7 is the IRQA1/IRQB1 status bit. It is set by transitions on the CA1/CB1 input lines and will remain set until an MPU Read of Peripheral Data Register A/B.

Bit 6 is the IRQA2/IRQB2 status bit. It is set by the CA2/CB2 control lines and will remain set until an MPU Read of the Peripheral Data Register A/B.

Bit 5 determines whether CA2/CB2 control line is an input (0) or output (1).

Bit 4, if CA2/CB2 control line is an input, determines whether IRQA2/IRQB2 status bit (bit 6) is set by low active transitions on the CA2/CB2 line. If CA2/CB2 is an output, bit 4 determines whether CA2/CB2 control line is a pulse (0) or a logical level (1).

Bit 3 set to 1, if CA2/CB2 control line is an input, enables the $\overline{\text{IRQA}}/\overline{\text{IRQB}}$ signal to channel an interrupt request into the $\overline{\text{IRQ}}$ pin of the MPU, based on transitions on the CA2/CB2 control line. If CA2/CB2 is an output, bit 3 then takes on three different interpretations.

1. If bit 4 is 1, then CA2/CB2 control line will be output at all times with the level of bit 3.

If CA2/CB2 control line is a pulse (bit 4 is 0), bit 3 specifies an automatic handshaking sequence as follows:

2. If 0, bit 3 selects an interrupt handshaking.
3. If 1, bit 3 selects a programmed handshaking.

It is important to note that there are some differences in the handshaking logic associated with CB2 as compared to CA2. The PIA A side will only handle data input with handshaking and CA2 will be output low only after an MPU Read operation of the A Peripheral Data Register, while the PIA B side will only handle data output with handshaking and CB2 will be output low only after an MPU Write operation to

the B Peripheral Data Register. For this reason, the A side of a PIA is normally used as an input port and the B side as an output port.

Bit 1, when 1, enables the $\overline{IRQA1}/\overline{IRQB1}$ signal, via which an interrupt request is output to the MPU, based on transitions on CA1/CB1 control line.

(c) PIA Initialisation

On a power-on condition or a hardware reset, all the registers in the PIA will have been cleared. Because of these conditions, the PIA has been defined as follows:

1. All I/O lines to the external world are defined as inputs.
2. CA1, CA2, CB1, CB2 are defined as interrupt input lines that are low active.
3. All interrupts on the control lines are masked. Setting of interrupt flag bits will not cause $\overline{IRQA}$ or $\overline{IRQB}$ to go low.

For these reasons, each PIA must be initialised by formatting the control registers before applying it to meet any specific task.

3.3.6 The MC6850 Asynchronous Communications Interface Adapter (ACIA)

The MC6850 Asynchronous Communications Interface Adapter provides a means of interfacing the MPU to devices requiring an asynchronous serial data format, especially over very long distances. It can function either as a serial-to-parallel converter or as a parallel-to-serial converter. This means parallel data on the MPU data lines can be sent to the ACIA, a byte at a time, converted to a series of 1's or 0's, and then sent to a serial data device such as a teletype, CRT terminal or a printer. Likewise, data in the form of

1's and 0's can be received from an external source such as a keyboard or tape reader, converted by the ACIA to parallel data bytes and then placed on the microcomputer data bus.

Figure 3.10 shows a simplified block diagram of the ACIA. As before, chip selects lines cause a particular ACIA chip to be accessed and a combination of the R/$\overline{W}$ and Register Select lines selects one of the four internal registers. However, there are only two addressable memory locations as the Control and Status Registers share a common address and the Transmit Data and Receive Data Registers share the other one.

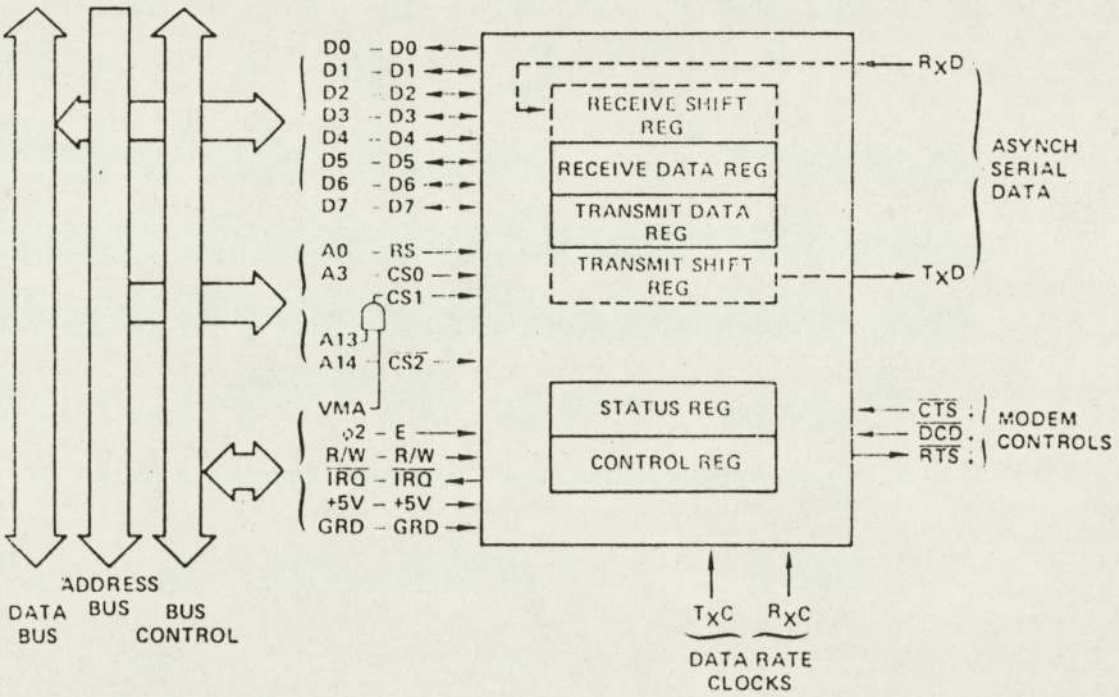


Figure 3.10 Block diagram of the MC6850 ACIA[89]

ACIA data transfer and control operations are more complex when compared to the PIA and as the user is usually less concerned with programming the ACIA, further details are not discussed here and system

manuals[88,89] should therefore be consulted. However, it is sufficient to note the following features:

1. The ACIA has no Reset input; a control code is used as a master reset.
2. The ACIA serial I/O logic requires an external clock signal (ϕ_2 of the system clock is usually used) in order to time the serial, asynchronous data stream which is either output or input.

3.3.7 The M6800 Input/Output (I/O) Ports

The M6800 I/O Ports provide the means to communicate with peripheral devices such as CRT terminals, printers and disk memories, via any of the interface adapters previously described. There are eight I/O ports, each of which is an eight-bit bidirectional port with control lines. Eight addresses per port is the standard assignment and as Table 3.4 shows, Port 0 resides with a base address of $F500.

Table 3.4 M6800 Input/Output Ports

I/O Port No.	Address Assignment
0	$F500-$F507
1	$F508-$F50F
2	$F510-$F517
3	$F518-$F51F
4	$F520-$F527
5	$F528-$F52F
6	$F530-$F537
7	$F538-$F53F

Hardware options are available however, for altering the base address of Port 0 and/or to assign four addresses per port.

3.3.8 Peripheral Devices

Supporting peripheral devices for the M6800 require some hardware configuration, provided basically by PIAs and ACIAs via the I/O Ports just described, for interfacing to the MPU.

A description of the available devices except the Interrupt Timer, which is discussed in Section 3.3.11, follows:

3.3.8.1 The Visual Display Unit (VDU) and the Teletypewriter (TTY)

The VDU/TTY is interfaced to the MPU by means of a single ACIA on Port 0 which has a base address of $F500. Since data handling is faster with the VDU, a switch is provided at the back of the M6800 panel to select the required baud rate. A Newbury Model 8510 bidirectional printer can be connected to the VDU to give hardcopy prints at 100 cps.

3.3.8.2 The FD-8 Floppy Disk Memory System

The FD-8 system comprises a single disk drive and controller and provides a cheap and reliable means of bulk storage for the M6800.

It is interfaced to the microcomputer by means of a single PIA chip on I/O Port 7 with a base address of $F538. One half of the chip is utilised as an eight bit bidirectional port for data flow and status information. The second half of the PIA is used as an output control port.

The controller board also contains approximately 3K of RAM memory which allows information to be transferred from controller to disk completely independently of processor speed.

3.3.8.3 The Analogue Input/Output Hardware

This hardware subsystem was built to provide the M6800 with the capability to perform data acquisition and process control functions.

It consists of a Control Logic Board and three devices, namely:

1. Mode Select Logic
2. An eight-bit Analogue to Digital Converter
3. An eight-bit Digital to Analogue Converter.

The unit is interfaced to the MPU through a PIA on Port 5 with a base address of SF528 and can be connected to a small analogue computer[90] and a level-sensor equipment[91].

Plate 3.2 shows the M6800 microcomputer with its supporting peripherals. In this research, the Analogue I/O Hardware is not used at all.

3.3.9 System Software

The M6800 is supplied with a family of software that permits program development and evaluation. Only the most important software packages are described.

The MSI-BUG Monitor

The MSI-BUG Monitor is a firmware program occupying locations $E000 through $E3FF. The ROM is connected to the M6800 system so that when a hardware reset occurs, the MSI-BUG Monitor is automatically entered. As such, the Monitor provides an immediate means of communications between the M6800 and the VDU/TTY.

The MSI-BUG program also has a 128-byte RAM, often referred to as scratchpad memory, for stack and temporary storage. This RAM has a base address of $F000. On reset, the Monitor automatically sets the MPU Stack Pointer to $F072.

The MSI-BUG Monitor provides useful functions which include listing, execution and debugging programs at he machine level. In particular, typing 'G EC00' on the terminal (G is the Execute User Program Function) causes the microcomputer to enter the Disk Operating

System (DOS) mode. A complete list of Monitor functions is found in the MSI-BUG Monitor MT-1 Manual.

The FD-8 Disk Memory System Software

The FD-8 software includes the Disk Operating System (DOS) monitor routine, disk driver routines, a routine called MINIDOS which reads/writes specified number of sectors to/from any desired address in memory to/from any desired track and sector location on a floppy disk, a routine called FDOS which allows the user to utilise floppy disk for system programs, user source and machine code programs, plus other utility programs which aid program development and documentation.

The DOS monitor and Disk Bootstrap are stored in ROM memory occupying locations $ECOO through $EFFF.

The TSC File Editor System

The TSC Text Editing System is intended for use with the FD-8 Disk Memory System and provides the user with an editor, the TSC File Editor, to enter and edit a M6800 Assembly Language or BASIC source program and save it on disk by filename or load it from a disk. For details on commands and use of the TSC File Editor, the TSC Editor for use with FD-8 Disk Memory System Manual should be consulted.

Software Dynamics BASIC (SD BASIC)

The Software Dynamics BASIC (SD BASIC) software allows the user to write programs in a familiar high-level language for scientific or engineering computation and real-time applications. SD BASIC is a compiler language, not an interpreter as are most BASICs. A source program is first compiled by SDBASCOM (SD BASIC Compiler) to produce an

output file which is M6800 Assembly Language compatible. This output file is then assembled using the M6800 Assembler to produce the binary code. This binary file can then be executed in conjunction with the system Run Time Package (RTP) and the Software Dynamics I/O Package (SDIOPACK).

SD BASIC compiling operation is therefore a two-pass process. If required, the compiled SD BASIC program can be merged with any other assembler routines using the RTP MERGE.B facility (a merger program) before proceeding to the second pass (assembling operation).

For the reasons just described, two main advantages are derived.

1. Programs are usually smaller as during execution, only the binary file and the run-time packages need be in core.
2. The run-time packages 'interpret' each line of assembled code and no line by line syntax analysis takes place. This leads to faster program execution.

The SD BASIC Compiler manual[92] should be referred to for a complete compilation of SD BASIC features. Among the useful ones are:

1. Individual memory locations (ROM or RAM) can be accessed.
2. SD BASIC programs are interruptible.
3. Assembly Language subroutines can be called from a SD BASIC program using the CALL facility.
4. SD BASIC accommodates file I/O operations in conjunction with the floppy disk unit.

3.3.10 The M6800 Interrupt Structure

The M6800 has a powerful interrupt structure which includes the following important aspects:

1. The stack concept.
2. Use of vectored interrupts.
3. Interrupt priority scheme provided by the microprocessor logic.

There are four kinds of interrupts to the M6800 MPU as summarised by order of priority in Table 3.5. The first three are hardware generated and the fourth, the Software Interrupt (SWI) is instruction initiated.

Table 3.5 Interrupts in the M6800 Microcomputer

Type	Location of Interrupt Vector in EPROM	Input Line to the MPU
Reset	$FFFE-$FFFF	$\overline{RES}$ via Pin 40
Non-Maskable Interrupt	$FFFC-$FFFD	$\overline{NMI}$ via Pin 6
Interrupt Request	$FFFA-$FFFB	$\overline{IRQ}$ via Pin 4
Software Interrupt	$FFF8-$FFF9	Instruction initiated

When an interrupt occurs, the instruction in progress is completed before the microprocessor begins its interrupt sequence. The first step in this sequence is to save machine status (i.e. program status) by storing the contents of the Program Counter, Index Register, A and B accumulators and the Condition Code Register on the stack in the order shown in Figure 3.11. The Stack Pointer (SP) which should always be pointing to a free location is decremented seven times to accommodate the seven MPU bytes. The interrupt mask bit (I) is next set to '1', which allows the service program to run without being interrupted by $\overline{IRQ}$ or SWI. $\overline{RES}$ and $\overline{NMI}$ are non-maskable interrupts and ignore the I bit. Then the MPU fetches the address of the service routine from the

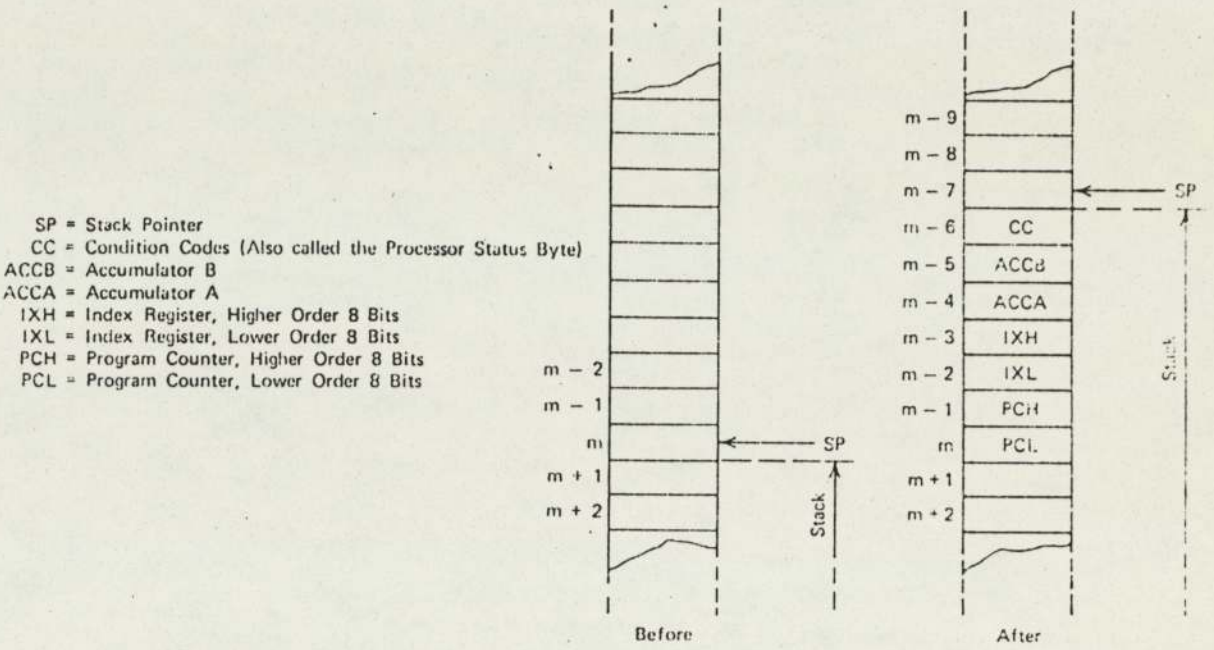


Figure 3.11 Saving machine status in the stack[89]

appropriate vector location and places it into the Program Counter.

Nested interrupts are permissible (in the case of $\overline{IRQ}$ and SWI, the I bit must be cleared using the CLI instruction) but the user must ensure the stack is large enough and common subroutines are reentrant.

The service program must end with the Return From Interrupt (RTI) instruction which restore machine register status, earlier stored on the stack.

$\overline{RES}$ interrupts are used to start a program from a power-down condition or to stop program execution by entering the MSI-BUG Monitor for debugging or to stop program execution by entering the MSI-BUG Monitor for debugging purposes. The $\overline{RES}$ line can also be connected to

any hardware devices that have a hardware reset and need to be initialised such as the PIA (the ACIA has no $\overline{RES}$ input line and must be software initialised).

$\overline{NMI}$ interrupts function as very high priority interrupts. Normally it is reserved for system recovery routine in the event of a power failure or where immediate user/peripheral action is required.

$\overline{IRQ}$ interrupts are usually used when peripheral devices must communicate with the MPU (data transfer, timers, etc.). Since a single PIA can generate interrupt requests from four difference sources, a software sort (e.g. a polling routine) must be carried out to isolate the active source.

SWI interrupts can be used in various programming situations such as for error indications and as a debugging aid.

3.3.11 The MP-T Interrupt Timer

The MP-T Interrupt Timer is a crystal-controlled, programmable oscillator/divider chip, interfaced to the microcomputer system via a PIA chip on I/O Port 6. Figure 3.11 shows the essential features of this arrangement. Only the B Side is used. The oscillator/divider's output is connected to the CB1 input line while its eight-bit data inputs are connected to the B-Side output data bus. These byte inputs to the timer are hexadecimal codes which governs the frequency of low active CB1 transitions (thereby generating $\overline{IRQ}$ interrupts of the MPU) at rates covering a range of one microsecond to one hour. The timer therefore provides a simple and accurate measure of the passage of real time.

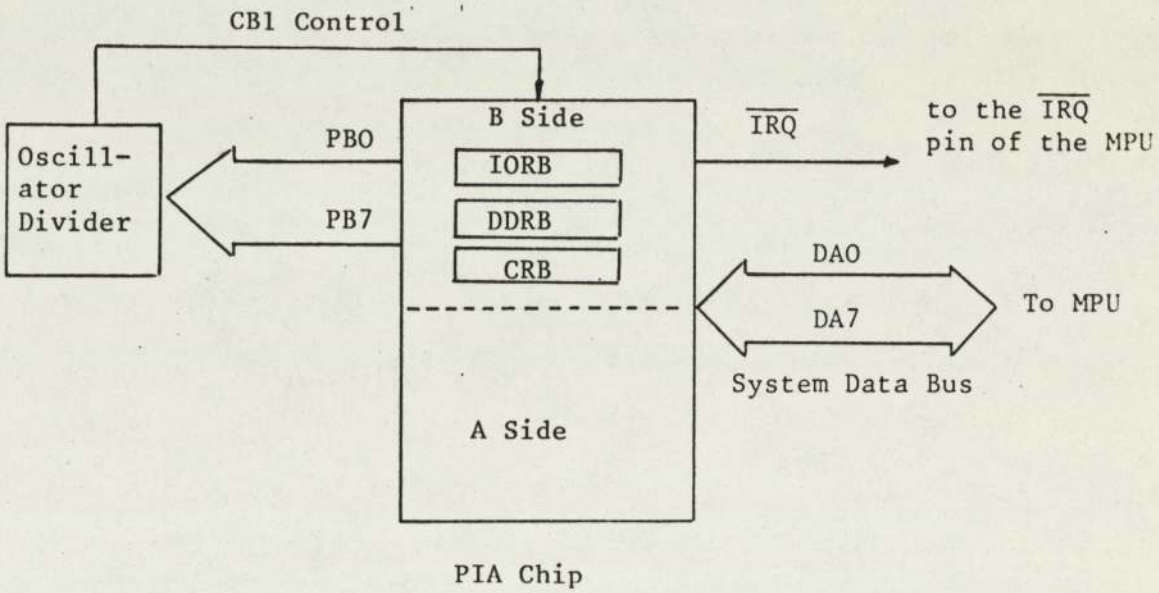


Figure 3.12 The MP-T Interrupt Timer Interface Organisation

3.4 The Linked H316-M6800 Twin Processor System

The H316 Minicomputer, through its HADIOS interface, has been the main tool for real-time data acquisition and control studies on a number of items of process plants (distillation column, double-effect evaporator, resin manufacturing plant, chemical reactor, etc.) in the Department. As a result, much of the original software has been developed and extended to support such applications. In particular, the BASIC-16 Interpreter is extended to permit real-time use and graphical display of results.

The M6800 microcomputer has also been used for real-time tasks but the applications are limited to using the Analogue I/O Hardware for data acquisition and processing from instrumentation.

The resulting software packages perform satisfactorily but in each case, operation is limited to

1. A single user.
2. One sampling frequency.

The M6800 is also limited in the number and variety of inputs and outputs it can handle, and has less supporting software.

For these reasons, a linked twin-processor arrangement was proposed, through which, for instance two users could share hardware and software resources or, alternatively two parts of the same task could be divided between the computer system.

3.4.1 System Hardware Overview

The basic idea here was to design a hardware arrangement in the simplest possible way and yet be sufficient for

1. either computer to be able to send an interrupt request to the other
2. 16-bit digital data to be transferred between the compilers (two 8-bit data bytes in the case of the M6800).

The resulting hardware arrangement for the linked twin-processor system is shown in Figure 3.13. It is unique in the sense that the M6800 access the process plant via the HADIOS and has to interrupt H316 processing to do so. Thus, in a way, the H316 behaves as the front end of the microcomputer.

Three HADIOS subinterfaces are used on the H316 side: a digital input (DGIB), a digital output (DGOB) and the alarm inputs. On the M6800 side, only a single PIA chip is required. The programmable

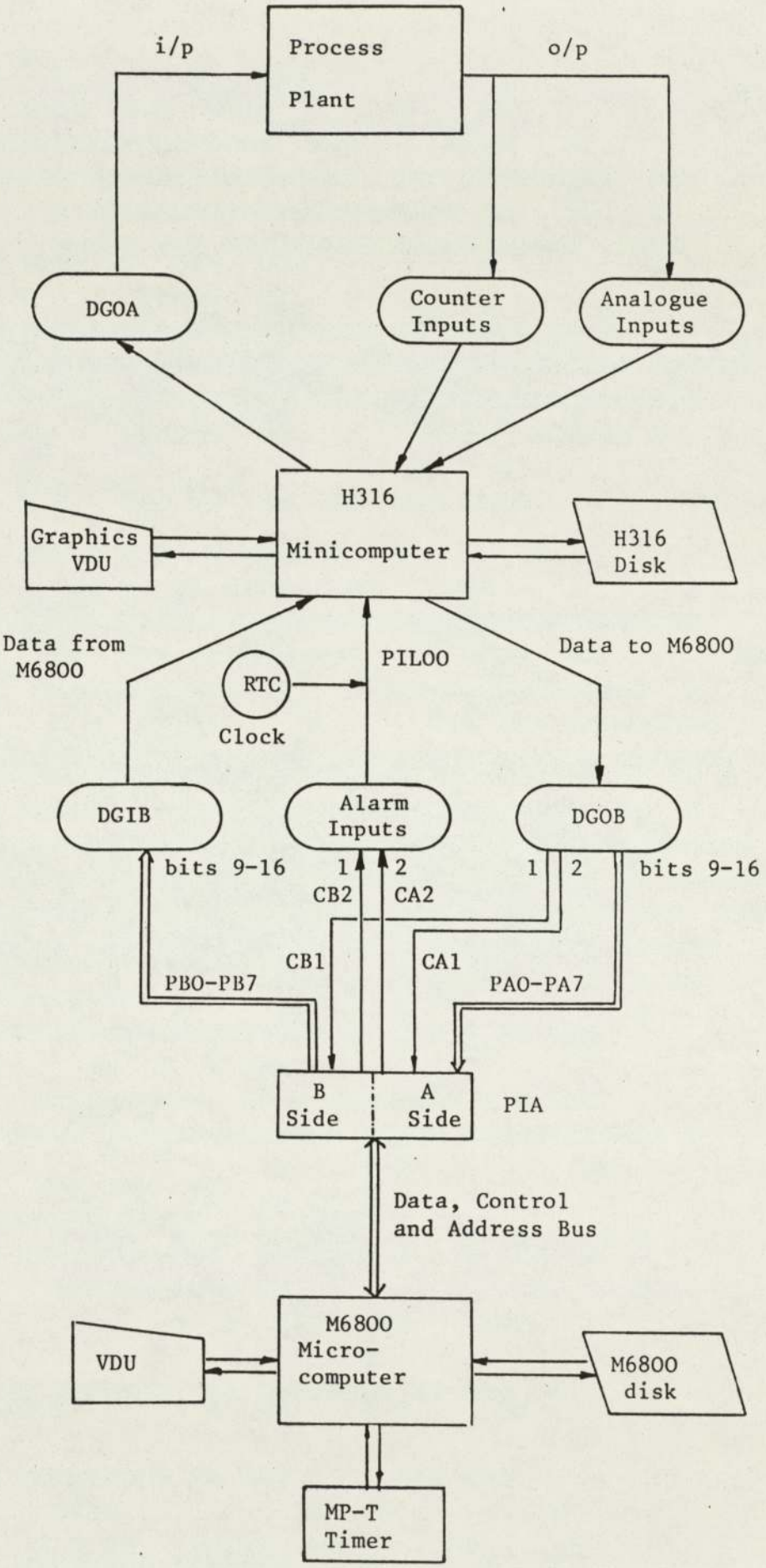


Figure 3.13 Hardware Arrangement for the Linked H316-M6800 Twin Processor System

nature of the PIA makes it very suitable for such an application.

3.4.2 Communication Protocol

The communication protocol was designed to meet both the system objectives and specific subinterface/adapter requirements. All the interface devices are TTL-compatible which means the output data lines (PB0-PB7) of the PIA B Side can be wired directly to the input lines (9 through 16)* of DGIB. Likewise, the input data lines (PA0-PA7) of the PIA A Side can be wired directly to the output data lines (9 through 16) of DGOB. Integer formatted data words can therefore be transferred between the computers via these I/O ports.

The M6800 needs to interrupt the H316 to perform the following:

1. Scan the process variables.
2. Output control settings.
3. Send data bytes to the H316 for further processing (special codes, further computation, output to paper tape or graphical display).

The first is accomplished via low active CA2 transitions on channel 2 of the alarm inputs. The other two are accomplished via low active transitions on channel 1 of the Alarm Inputs. The alarm inputs subinterface has been configured so as to generate an interrupt to the H316 when a low active transition occurs at any one of its sixteen input channels. Note that channels 3 through 16 of the Alarm Inputs are not used and remain in open circuit conditions.

By experimentation, it was found that the operating speed of the Alarm Inputs is about 100 Hz and the M6800 must wait in a sufficiently long delay loop before attempting to generate the next

---

*In the H316, the bits are numbered 1-16 from left to right.

CA2/CB2 interrupt. This timing problem was overcome by fitting a 10 μF capacitor between the solder turrets on the input stage of each Alarm Inputs channel used.

The CB2 interrupt acknowledged in the handshaking scheme is a low active CB1 transition generated from output pin 1 of DGOB. This transition sets the $\overline{\text{IRQB1}}$ flag and can therefore be detected by the M6800 software.

The PIA A Side, serving as the data input port to the M6800, is configured to operate in two modes. This is because the data transfer can be initiated either by the H316 or the M6800. The H316 may need to transfer data bytes to the M6800 immediately and it can interrupt the M6800 by a low active CA1 transition generated by the output line 2 of DGOB. Therefore, the A Side should have the option to be in CA1 interrupt mode The response to this interrupt is to convert the A Side to the handshaking mode, outputting a CA2 interrupt to the H316 every time the M6800 is ready for more data.

In a microprocessor scan, the A Side is always switched to the handshaking mode. Low active CA2 transitions interrupt the H316 and the data transfers acknowledged via the CA1.

The digital output DGOA is used (not simultaneously) by both computers to output control settings to the process plant.

3.5 Conclusions

This chapter has described mainly the hardware building blocks of the linked H316-M6800 twin-processor system and the arrangement shown in Figure 3.13 is the proposed architecture based on the resources available at the time of system design. One could use more PIAs on the M6800 side and allow more M6800 interrupts to the H316 via the remaining Alarm Inputs channels. Clearly, more complex and more

powerful data handling operations can be accommodated.

As described in Section 3.4.2, the communication protocol strongly depends on the hardware arrangement and software objectives for the given system. In fact, it directly affects the design and operation of the two related Assembly Language executive programs which handle the communication between the two processors.

CHAPTER FOUR

ON-LINE SOFTWARE DEVELOPMENT

4. ON-LINE SOFTWARE DEVELOPMENT

4.1 Introduction

In the context of this chapter, software development means the system design and development of the software packages required to operate the linked H316-M6800 twin-processor system for real-time data acquisition and process control. The software includes both system programs (standard routines or user modified) and user-written application modules.

The software support for the H316 minicomputer is already well developed. System programs are well documented and source listings (assembler listings) are available which make these programs easy to modify to suit a particular application. Over the years, various packages had been developed and in particular, attention is focussed on developing the BASIC-16 Interpreter to permit interactive real-time use and graphical display of results. To support these applications, many versions of the HADIOS Executive have been written (in DAP-16 Assembly Language) to operate the various HADIOS devices. The user's BASIC program may then call subroutines in the HADIOS Executive or the Graphics Library to meet his particular requirements.

On the other hand, system software documentation and listings for the M6800 are minimal. This means modifying them in the H316 sense is a non-trivial task. Also, it means that the detailed organisation and operation of important system packages such as the SD BASIC COMPILER and the RUN TIME PACKAGES are not fully understood. In one attempt to overcome this handicap, a DISSAMBLER was written to decode system binary programs into M6800 assembly language level codes[93]. The exercise on the RUN TIME PACKAGES for instance, was laboriously time-consuming and met with limited success.

Real-time use of the M6800 has been largely confined to the use of SD BASIC. This is because SD BASIC, like most BASICs, is a familiar language to most users and has many extended features suitable for such tasks. An SD BASIC program is interruptible; it can read or write into memory locations (for instance, testing an interrupt status flag) via its POKE and PEEK statements, manipulate bits via its AND, OR, SHIFT, XOR or COM functions, calls on M6800 Assembly Language subroutines, and communicate with data files stored on the floppy disk. In fact, a user may not find it necessary to interface assembly language subroutines to his BASIC program at all. This is generally true for most microprocessor BASICs since microprocessor software is usually more hardware dependent than others. But to make useful exploitation of this feature, the user must also have a reasonable understanding of the hardware building blocks of his microcomputer system.

The preceding discussions, at the same time, bring about another point. While each computer has sufficient software to operate individually in real-time mode, extra software is required to handle the communication protocol for the twin processor set-up. This is provided by two assembly language executive programs, one residing in each computer. This chapter describes the design and operation of such programs, and the use of existing software to achieve the stated objectives of the linked twin-processor system.

4.2 Software Objectives

The principal objective of previous software was to enable the user to program the H316 minicomputer in BASIC only so that programs would be easy to write and modify. The HADIOS Executive enables him to specify the sampling frequency, the devices and number of scans required. The system is flexible enough for the single user

to sit down at the VDU/TTY, in the immediate environment of his process plant, and conduct calibration or control experiments. Normally, his BASIC program would call subroutines (which were originally written in FORTRAN or H316 Assembly Language code) stored as machine code for further processing of plant information as these lead to faster execution times.

The M6800 user would also retain the interactive feature in spite of SD BASIC being a compiled language. This is because editing and recompiling programs is efficient as the language (unlike BASIC-16) is disk-based and supported by a good operation system (DOS).

The simplest logical approach for the linked twin-processor system is therefore to retain the facility for programming in BASIC and develop the software with the following objectives:

1. To continue the facility for an interactive BASIC program in the H316 to access a plant through HADIOS with an option for on-line graphics.
2. To provide a facility for a disk-based SD BASIC program in the M6800 to access the HADIOS hardware and the H316 graphics facility.
3. The HADIOS hardware to be divided between the two processors or accessed by both processors (except the counters). This would allow two plants or the same plant to be monitored and controlled.
4. Each processor to control its own real-time clock (hence two independent sampling frequencies are possible).
5. Either processor to be taken out of real-time mode and used off-line without affecting the other.

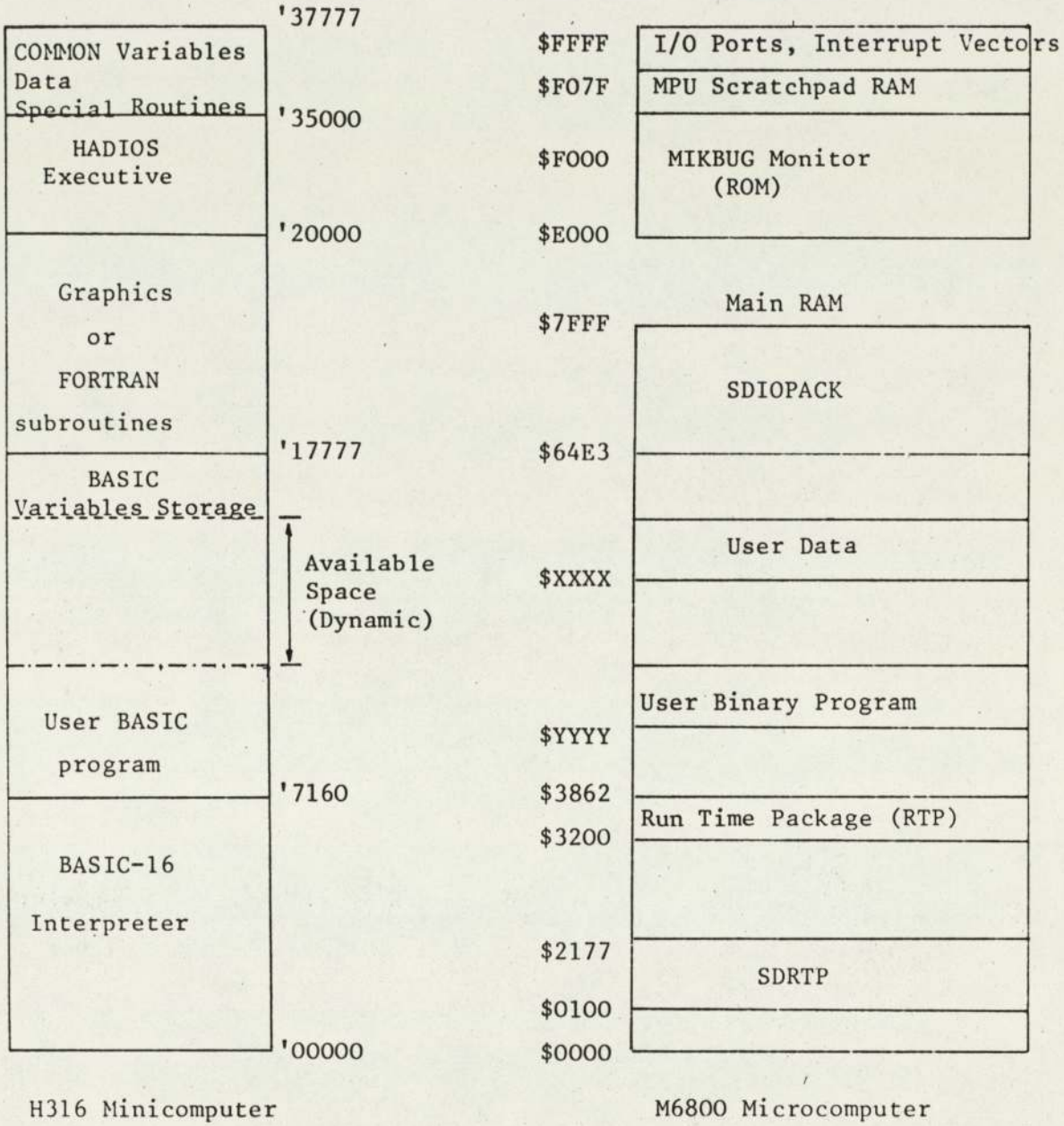
6. To provide a program overlay facility in conjunction with the AED3100 floppy disk unit for the H316 user.

The last objective is included to overcome the problem of user core-memory and design limitations. Since the M6800 may interrupt the H316 at any (permissible) time, the HADIOS Executive must always be core-resident. With additional space provisions for graphics, the H316 user may soon find that he is short of space to accommodate his BASIC program and data storage, and FORTRAN subroutines (which may be large in certain applications such as on-line parameter and state estimation) necessary for a given computing task. In view of the latter situation, the disk to core transfer facility is incorporated in the overall software design strategy.

4.3 Memory Allocation

Memory consideration is important in the initial stages of design as a limited core memory may result in the stated software objectives being more difficult to achieve. In this research, the problem was more critical in the H316 minicomputer. A consideration of memory utilisation in each computer (Figure 4.1) helps explain why.

Core memory for the H316 amounts to 16K organised into thirty-two (40 octal) 512-word sectors. System requirements mean the BASIC-16 Interpreter and the HADIOS Executive must be core-resident. The BASIC-16 Interpreter essentially occupies the lowest seven and a half sectors of memory (see Section 4.6 for greater details). It also requires a high octal address (HOA) to be specified to calculate and inform the user of the space available for his BASIC program and data storage. This means HOA should be made as high as possible. The HADIOS Executive and its FORTRAN library routines are relocatable code and could be loaded high up in memory.



$XXXX and $YYYY can be assigned using the SD BASIC DATA ORIGIN and PROGRAM ORIGIN facility respectively.

Typical values are:

$XXXX - $5000

$YYYY - $4000

Figure 4.1 Typical Memory Utilisation in the H316 and M6800 Computers

Memory allocation is also necessary for the GRAPHICS library, user-written FORTRAN subroutines, disk Read/Write routines, and any other data areas that used to remain core-resident. For these reasons, a value of '17777 is chosen for the HOA giving a BASIC user space sufficient for most applications. The HADIOS Executive is loaded high up in memory so that memory sectors '20 through '26 can be used for program segments loaded from the disk. (See Chapter 6 for a practical example.)

In the M6800, the user normally has most of RAM memory locations $0000 through $7FFF for program space. For real-time use, the SD BASIC Run Time Packages and the user's BASIC program interfaced to the M6800 Executive must be in core. The Run Time Package (RTP) and its Input Output Package (SDIOPACK) occupy locations $3200 through $3862 and $64E3 through $7FFF respectively. The software also uses page zero ($0 through $00FF) for direct addressing. Some other RAM locations are used for interrupt vectors ($FFF8-$FFFF), I/O Ports address assignments ($F530-$F53F) and CPU scratchpad ($F000-$F072). The remaining area of RAM is available to the user.

4.3.1 Source Tape Preparation

In the course of this research, programs and subprograms have been written in SD BASIC, Honeywell BASIC, standard FORTRAN, DAP-16 MOD 2 and M6800 Assembly Languages. In each computer system, a text editor was used.

4.3.1.1 H316 Software

As the H316 is not provided with a disk-based operating system, the H316 Text Editor was used to prepare source programs on paper tape or cassette tape. DAP-16 MOD 2 programs were normally assembled in the two-pass mode. FORTRAN subroutines were compiled

using the FORTRAN Compiler Revision E. The object output from both operations are compatible and are loaded into core memory using a self-contained loader, LDR-APM Revision E.

Since the BASIC interpreter uses most of sector zero (the usual base sector) for address constants, buffers and base requirements, H316 source programs were written in the de-sectorised mode. FORTRAN modules are kept small (within a sector) and perform no I/O operations. The starting base address for each sector is specified at loading time. For DAP-16 MOD 2 programs, the SETB (Set Base) pseudo-operation may be used. The programmer must ensure that sufficient space is reserved for base. A base crossing into the next sector is detected by the loader (an 'MO' - memory overflow message is output to the VDU) but not base which is over-written by program code. An estimate of program size is helpful but a successful program load usually requires some initial dummy loads or trial and error procedures.

At one stage, it was thought that the Honeywell Series 16 FORTRAN Translator could be used to de-sectorise FORTRAN subprograms. The Translator allows in-line DAP-16 MOD 2 coding so that the SETB and ORG (Program Origin) pseudo-instructions can be inserted. However, the author found the method less attractive as too many iterations were required to de-sectorise large FORTRAN programs (as they are in estimation studies). Furthermore, the locations of blocks of COMMON storage in the various program units must be defined at translate-time. This is because the names given by the FORTRAN programmer to particular COMMON blocks cannot be passed on for recognition by the loader.

Another problem was concerned with subroutine reentrancy. The FORTRAN library routines are non-reentrant which means certain interrupt response code (such as M6800 interrupts) must have its own collection of FORTRAN library routines. This requires more space for

code duplication. The BASIC interpreter has its own library routines (the BASIC MTH-PAK). They are also non-reentrant but very similar to equivalent FORTRAN mathematical routines. For this reason, they are fully exploited and used by subroutine or interrupt response code when BASIC statements are not in execution (such as the RTC interrupt response and Graphics). A list of the BASIC MTH-PAK routines used is found in Table A4.10 of Appendix 4. Note that several routines are not available in the MTH-PAK. These include ARG$ and F$AT (used in argument transfers) and D$11 (integer division).

4.3.1.2 M6800 Software

The disk-based operating system (DOS) in the M6800 makes it easy to write and compile programs. The TSC Text Editor and the program files are all stored on the disk. For large source files (such as the M6800 Executive), a memory patch using the MIKBUG command 'M' was necessary to increase the editor's workspace. In the H316, a similar patch was not possible. The HADIOS Executive source tape editing had to be done in two parts.

Another advantage in the M6800 is that unlike the H316, the M6800 emulates some aspects of stack architecture. The M6800 stack grows toward the low end of memory (on a last-in first-out (LIFO) basis) and the instruction set provides fast 'PSH' and 'PUL' (push and pull) instructions operating on registers A and/or B for stack manipulation. While in the H316 the user has to specify the buffer to save the machine status of an interrupted program (in fact one buffer for each source of interrupt), processor status in the M6800 is automatically 'pushed' on to the stack on interrupt. The programmer is therefore relieved of this chore but has to make sure that the stack must not grow too large and overwrite some instruction or data bytes. Further-

more, the H316 programmer must be careful not to use common register-oriented instructions such as LDA (Load A register) or STA (Store A register) before saving the status of processor keys (the C-bit, shift count etc.) in the interrupt response code.

At the assembly language level, the M6800 instruction set also offers greater flexibility than DAP-16 MOD 2. The H316 is very much single-register (A-register) oriented. The index register is useful for temporary storage and addressing and many control instructions (SNZ-skip if non-zero, SPL-skip if plus, etc.) work only with the A-register. On the other hand, two 8-bit registers (A and B) and a 16-bit index register are available to the M6800 programmer. Branch instructions (BMI-branch is minus, BLS-branch if less or equal, etc.) work with all the registers. Furthermore, the index register itself can be indexed and is therefore useful in retrieving a 16-bit value from a 16-bit address. Ample examples of the points just mentioned are found in the respective real-time executives.

The WAI (Wait for Interrupt) instruction however, does not work in this version of the M6800 microcomputer system. Due to the specific hardware set-up of the interface adapters, a WAI causes the VMA (valid Memory Address) line to become inactive and disenables all incoming $\overline{\text{IRQ}}$ interrupts.

Also, the TSC Text Editor can be patched with the following assembly code to punch out (at the TTY) H316 compatible source tapes. The M6800 can therefore act as an intelligent terminal to the H316.

Finally, in writing SD BASIC programs for use with the M6800 executive the programmer must avoid variable names which have already been specified as labels in the Executive. Data structures for integers and reals are also different in SD BASIC. The address of a variable and its numerical value (real or integer) each takes up six

A Machine Code patch of the TSC Text Editor to enable the preparation of H316 compatible source program tapes.

```
         NAM   MCASM
         OPT   O
         ORG   $7FDO
OUTEEE   EQU   $E1D1
         STAA  TEMP
         EORA  #$8O
         JSR   OUTEEE
         LDAA  TEMP
         CMPA  #$OD
         BNE   RETN
         LDAA  #$8A
         JSR   OUTEEE
RETN     RTS
TEMP     RMB   1
         END
```

bytes as indicated below. Memory grows from left to right.

Address of Numeric Value

0 * * *	ADDRESS

Integer Value

0 * * *	INTEGER

$0 \leq \text{INTEGER} \leq 65536$

Decimal Floating Point Values

EXP	BCD	BCD	BCD	BCD	BCD

The BCD bytes are base 100 digits. Floating point zero if defined by EXP = 0 and all BCDs zero.

Because main memory is volatile in the M6800 and data areas may contain 'anything' on a power-on condition, it is therefore sensible to initialise those variables which is assumed by the M6800 Executive to be in a real or integer format, at the beginning of the program and sustained throughout program execution. However, SD BASIC is flexible enough to allow an array to have a mixture of real and integer elements.

4.4 The HADIOS Executive Revision 03

The HADIOS Executive Rev 03 is a considerably modified version of the previous version. It is written in H316 Assembly Language (DAP-16 MOD 2) and contains subroutines callable from a BASIC program, and all the interrupt service routines. In this way, the details and low-level operations of HADIOS devices remain hidden from the general user who would be programming mainly in BASIC and FORTRAN.

The organisation of the Executive is best understood by first listing its specific functional objectives which are as follows:

1. To enable regular access of HADIOS hardware at the required frequency from either computer.
2. To handle and service all interrupts to the H316 minicomputer (real-time clock, counters, CA2 and CB2 control, and special data transfers to/from the M6800).
3. To provide a real-time base for the calculation of flowrates from counter inputs (for both computers).
4. To provide an idling loop to wait for clock interrupts.

5. To provide a dispatcher table which contains pointers for special/background processing. An option is provided whereby the H316 may be exclusively used by the M6800 user (graphics and/or data processing).
6. To enable either computer to output control settings to the process plant at any desired time.
7. To terminate H316 scanning of HADIOS devices when the required number of scans has been done.
8. To handle error conditions while not disrupting any microprocessor operation.

These objectives are achieved by a combination of subroutines callable from BASIC, and interrupt response routines. The use of the subroutines will be discussed first with reference to an assembly listing of the HADIOS Executive provided in Table A4.1 of Appendix 4.

4.4.1 HADIOS Subroutines

The user communicates with HADIOS by CALLs from his BASIC program as follows:

CALL (1, A(0), B(0)

Subroutine 1 is the basic scanning routine. Arrays A (dimensioned 13) and B (dimensioned 86) contain the user's parameters and variables.

A(0) Scanning interval in seconds

A-Array

A(1) Devices required

= 1 Analogue Inputs

= 2 Counter 1

= 4 Counter 2

= 8 Counter 3

The required set of devices is selected by setting A(1) equal to the sum of the appropriate values above. This allows easy decoding of the HADIOS devices required by ANA (logical AND with A-Register) instructions in the Executive.

A(2) Number of scans required, including the initial scan at time zero.

A(3) First analogue channel.

A(4) Last analogue channel.

A(5) A delay element in the Analogue Inputs sampling code. The specified analogue channels (A(3) through A(4)) are scanned as many times as possible over a period of one H316 clock resolution (20 ms) and the cumulative sum for each channel is divided by that ensemble number. The smaller A(5) is, the smaller the delay after each multiplexing operation and therefore the larger the ensemble number. This averages out the noisy signals giving the effect of a simple filter. A(5) = 33 is a typical value and zero is not recommended.

The Executive returns the current ensemble number in A(5) after every Analogue Inputs scan and stores the initial value of A(5) internally.

A(6) Counter 1 scan type.

= 0 Non-interrupt mode.

= 1 Counter 1 interrupts enabled.

A(7) Counter 1 preset value (0-255).

A(8) Counter 2 scan type.

= 0 Non-interrupt mode.

= 1 Counter 2 interrupts enabled.

A(9) Counter 2 preset value (0-255).

A(10) Counter 3 scan type

= 0 Non-interrupt mode

= 1 Counter 3 interrupts enabled

A(11) Counter 3 preset value (0-255)

A(12) A program flag in the HADIOS Executive.

= 0 Normal. The H316 scans at its own clock frequency.

= 1 In this mode, the HADIOS devices come under exclusive control of the M6800 user. The H316 clock is not interrupt enabled and only the following elements of array A need to be specified:

A(1) = integral multiple of M6800 sampling intervals.

A(2) = total number of scans required

The Executive also stores values of Analogue and/or Counter Inputs (specified by the M6800 user) into the appropriate B array elements. The H316 BASIC program can therefore access these values at the interval specified by A(1) and subject them to further mathematical treatment and/or graphics. In fact, the treated data can also be communicated to the M6800 via another CALL to Subroutine 4.

A(13) A program flag in the HADIOS Executive.

= 0 Normal, ON-LINE MODE*.

= 1 The H316 is in OFF-LINE mode. The H316 is not clock interrupt driven for a plant scan but may communicate with the M6800 which may or may not be in OFF-LINE mode. If the M6800 is in OFF-LINE, then the H316 has to interrupt the M6800 through subroutine 4 for a H316 to M6800 data transfer.

The subroutine of A(1) and A(2) is the same as when A(12) = 1.

---

*ON-LINE means with respect to the process plant only.

B-Array

B(0)-B(47) Analogue Inputs channel readings (each averaged over the ensemble number as returned in A(5)).

B(48) Counter 1 interrupt time (sec.). The HADIOS Executive Rev 03 (unlike previous versions) upgrades the interrupt time at every counter interrupt cycle. It is proposed that in general this gives a better measurement of the 'instantaneous' value at scanning time (see Section 4.4 on Counter Code Modifications). This modification applies to all counters i.e. B(51) and B(54) as well.

B(49) Counter 1 contents at scanning time.

B(50) Number of Counter interrupts during the last scanning interval.

B(51) Counter 2 interrupt time (sec.).

B(52) Counter 2 contents at scanning time.

B(53) Number of Counter 2 interrupts during the last scanning interval.

B(54) Counter 3 interrupt time (sec.).

B(55) Counter 3 contents at scanning time.

B(56) Number of Counter 3 interrupts during the last scanning interval.

B(57)-B(86) Storage area for data transferred from the M6800. (0-32767).

At the BASIC level, subroutine 1 is entered only once. The addresses of A(0) and B(0) are passed to the Executive via locations PARS and PARS+1 respectively. All other array locations can therefore be addressed as displacements from this base taking into account that all BASIC variables are REAL in the FORTRAN sense and occupy two computer words each.

The location of the next BASIC line after CALL (1,A(0),B(0)) is also stored internally. It serves as the entry point into BASIC from the DISPATCHER waiting loop (a scanning interrupt or an M6800 interrupt when A(12) or A(13) is non-zero).

Basically Subroutine 1 initialises some program flags and the required HADIOS devices. In the interrupt mode, the clock and the counters are initialised by a routine called the Common Interrupt Initiator. The program then enters the DISPATCHER loop to wait for the first clock interrupt which comes after about 100 ms.

CALL (2)

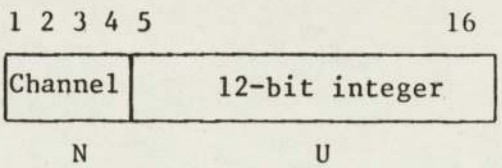
This CALL causes the program to wait in the DISPATCHER loop for the next scan (clock interrupt of A(12) = 0 or the relevant M6800 interrupt if A(12) or A(13) = 1) if all scans requested via A(2) have not been done. Else the BASIC program continues from after the statement CALL(2).

CALL (3,N,U)

This subroutine converts digital output signals into analogue outputs via DGOA and extra hardware (a 12-bit DAC and a Zero-Order Hold).

N Analogue output channel (0 - 15).

U Digital equivalent of the analogue output. For $0 \leq U \leq 32767$, the analogue output is 0 to 10 volts. As DGOB is normally interfaced to the PIA A Side, only DGOA is used by this subroutine. The set-up word output to the DGOA is shown below:



The DGOA can be accessed by both computers. To prevent a simultaneous service, part of the code has been made interrupt inhibited.

CALL (4,I,M,D(0))

This subroutine is used to transfer data stored in the D array to the M6800. If I = 0 then the M6800 must be in ON-LINE mode and the subroutine merely places the data in a temporary buffer (BUF1 BSZ 30) to be picked up by the M6800 at its next process scan, else I = 1, which causes the routine to immediately interrupt the M6800 via a low active CA1, thus initiating the data transfer.

M — Number of H316 computer words to be transferred including D(0) in one transfer operation (M ≤ 30).

D(0) — Array locations D(0)-D(M-1) are consecutively occupied by the M data words.

4.4.2 Interrupt Handling

All interrupt requests to the H316 are channelled into the standard interrupt line, PIL00, and sorted out by the HADIOS Executive. When an interrupt occurs (for conditions, see Section 3.2.4), the program automatically executes a JST* '63 (a JUMP and STORE through dedicated location '63) to begin the servicing routine. The address SKST of the Executive is placed in location '63 so that the software sort may begin from location SKST+1.

The sorting out is accomplished by a sequential testing of possible sources using the SKS (skip if sense line set) instruction. Once identified, program status is saved in a five-word buffer of the particular interrupt Link by the Common Interrupt Handler (as opposed to the stack concept in the M6800). The program status consists of five words:

Contents X and A-Registers.

Program Keys (C-bit, Shift Count, Double Precision Mode Indicator, etc.).

Contents of the B-Register.

Return Address.

After an interrupt response code has been executed, the program exits through a Common Interrupt Return which restores the original status of the interrupted program. System requirement means that some parts of the interrupt response code are necessarily interrupt enabled/inhibited.

Figs. 4.2a and 4.2b respectively show the flowchart of decoding a particular interrupt source and the saving of program status by the Common Interrupt Handler. Note that in sorting out the M6800 interrupts, the keys are saved first before manipulating the A-register.

4.4.2.1 The H316 Real-Time Clock (RTC)

The H316 clock interrupt is primarily used to initiate a scan of the required devices. The code is fairly straightforward except that some manipulations of the contents of some locations associated with the counter response code is done to maintain the correct measure of real time. This is because the counter response code also uses location '61 as a time base (see Section 4.4.3).

Note that the clock resolution is 20 ms. The BASIC user merely specifies A(0) in seconds. The Executive multiplies it by 50 first, then truncates the result to the nearest integer two's complement multiple of 20 ms. On a clock interrupt, the clock is normally reset to this initial value.

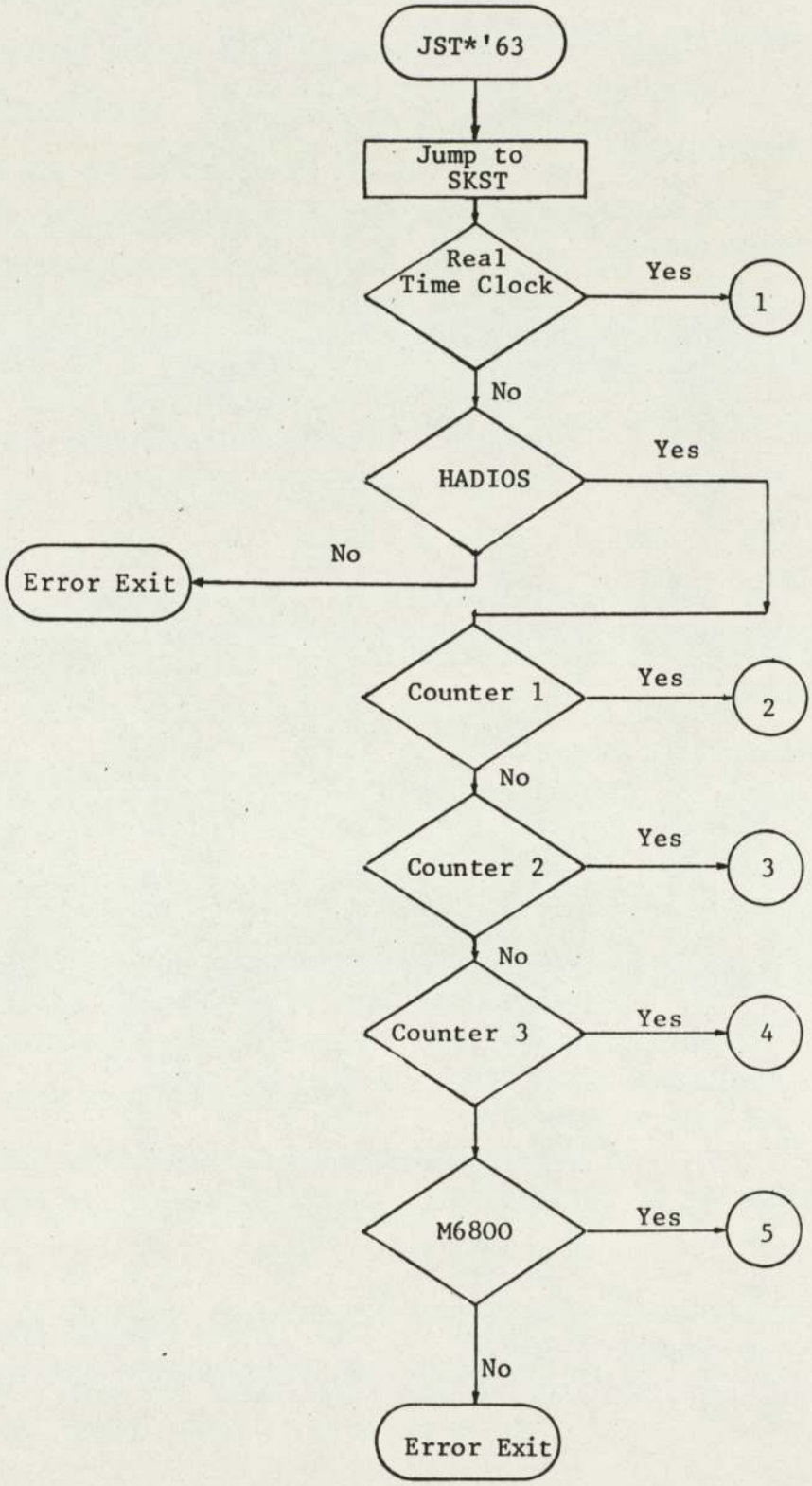


Figure 4.2a Interrupt Source Identification in the H316 Minicomputer

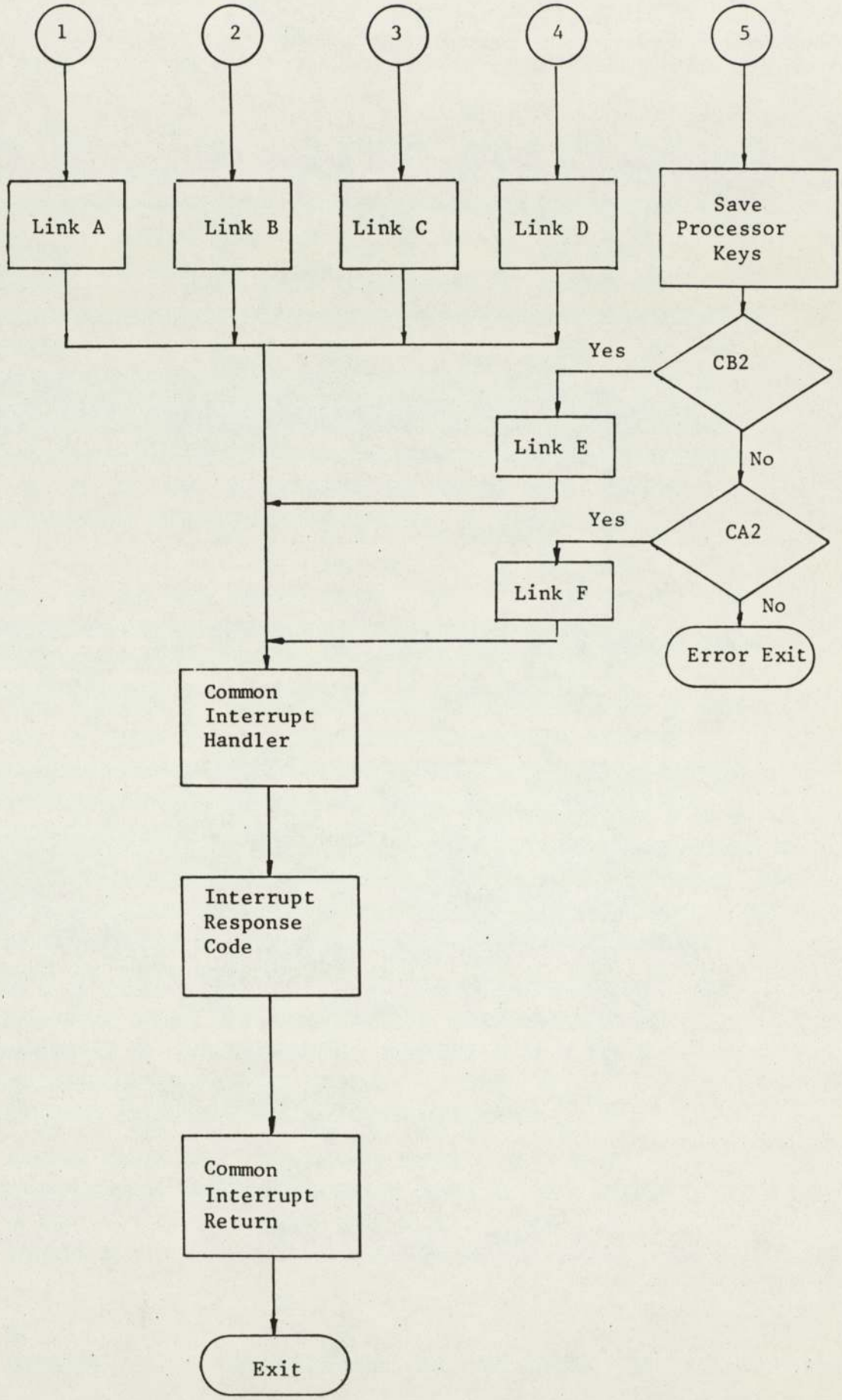


Figure 4.2b Interrupt Handling in the H316 Minicomputer

4.4.2.2 Counters

Each counter input is driven by pulses received from a flowmeter which provides an alternative method of flow measurement in comparison to the usual pressure-differential manometer/transducer. When programmed in the interrupt mode, a counter interrupts the H316 when its internal register is half-full (127). The modification to the counter interrupt response code is described in greater detail in Section 4.4.3.

Although both computers can access any of the counters, at any one time the usage is mutually exclusive.

4.4.2.3 CB2 Interrupts

Because the M6800 is interfaced to the H316 via a single PIA, the CB2 and the CA2 interrupt response codes make heavy use of steering flags or semaphores to achieve different computing tasks. The CB2 interrupt in particular, is used to provide four basic functions.

1. Initialisation

Normally, the HADIOS Executive Package Revision 03 (BASIC Interpreter + Graphics + HADIOS Executive Rev 03) is started by entering at location zero (relative) in the Executive. This is because the Common Interrupt Initiator needs to configure the H316 to make it CB2 and CA2 interruptible right from the beginning. A jump is then made to location '1000 for the initialisation of the BASIC Interpreter.

The M6800 can then interrupt the H316 to access the required HADIOS devices. The first ten CB2 interrupts are normally assignments of HADIOS parameters specified by the user in SD BASIC (see Section 4.5.1).

2. Control Outputs

A 16-bit digital equivalent analogue output can be sent by the M6800 user using two CB2 transfers. The HADIOS Executive then outputs

the value via the specified channel of DGOA to the process plant. Because any byte containing 254 or 255 is a special code to the Executive, the maximum value for the least significant byte of the 16-bit control word is 253. In other words, the maximum 16-bit control output from the M6800 is limited to $7FD.

3. M6800 to H316 Data Transfer

The HADIOS Executive can receive up to thirty words (60 bytes) from the M6800 at any one time through sixty CB2 transfers. This operation must be preceded by another two CB2s which send a byte of 254 (a steering flag) and the number of words to be transferred.

4. Error Code

In the event of an error situation that would require the M6800 program to be aborted, a data byte of 255 is sent to the H316 via a single CB2 transfer. This causes the HADIOS Executive to initialise the H316-M6800 communication protocol, without disrupting any H316 software operation. Such an error situation could either arise from a genuine program execution error which is detected by the M6800 Executive or a user interference which stops M6800 operation when the NMI button is pressed.

Figure 4.3 shows a flowchart of the CB2 Interrupt structure. Note that a CB2 interrupt is always acknowledged by a CB1 in the handshake protocol.

4.4.2.4 CA2 Interrupts

The PIA A Side is also configured in the handshake mode except that when the M6800 is in the OFF-LINE mode, bit 1 of the control register is set, thus allowing the H316 to interrupt the M6800 via a CA1 transition.

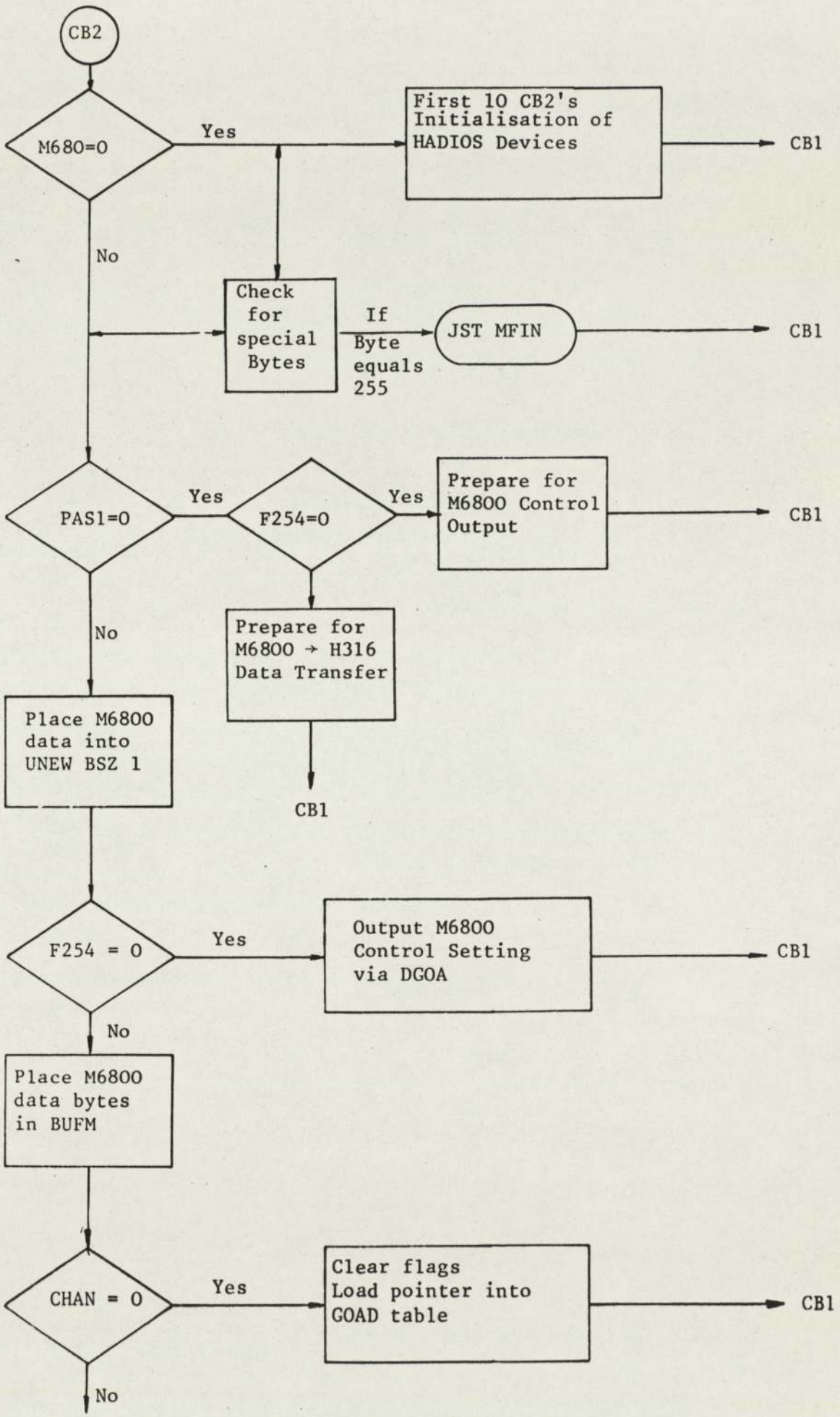


Figure 4.3 The CB2 Interrupt Flowchart

CA2 interrupts are used in the transfer of data bytes from the H316 to the M6800 which arise in three different situations as described below. As in the CB2 interrupt response code, steering flags are used to achieve different computing tasks as only a single CA2 interrupt line is available.

1. M6800 Scanning Routine

In a M6800 scan, two CA2 interrupts are required to retrieve a HADIOS device input (analogue or counter). Each byte of the input word is sent together with a CA1 acknowledge. The CA1 transition does not cause an $\overline{\text{IRQ}}$ interrupt to the M6800 as bit 1 of the PIA A Side control register has been earlier reset to 0. During such a scanning operation, the A Side remains in this non-interruptible mode. Note that this operation also picks up any H316 data that is 'waiting' to be transferred.

2. H316 to M6800 Data Transfer

When the M6800 is not scanning (OFF-LINE mode), the H316 may transfer data to the M6800, initiated by a CA1 interrupt through the PIA A Side. The CA2 interrupt to the H316 is now used as an acknowledge signal. Note that in this operation, only the first CA1 is an interrupt to the M6800. The following transitions are handled in the same way as in a M6800 scanning routine except that the role of CA1 and CA2 are now reversed.

3. Error Code

An error situation may also be detected in the HADIOS Executive which only affect M6800 operation. These error codes are sent to the M6800 via any of the two data transfer operations just described. A fuller treatment of error handling by the HADIOS Executive is described in Section 4.4.4.

Figure 4.4 shows a flowchart of the CA2 Interrupt structure. Note the two ways in which the CA2 (hence the CA1) can be used .

4.4.3 Counter Code Modifications

The counter interrupt response code and counter inputs handling routines have been modified to achieve the following objectives:

1. A search for a more accurate value of the 'instantaneous' flowrate.
2. Counters under M6800 control to depend also on the H316 clock (location '61) for measurement of real time.

A relatively simple scheme has been devised to achieve both objectives. It is based on the continuous running of the H316 clock. When H316 scans are not required, the clock is enabled to run only in the non-interrupt mode i.e. the transition from -1 to 0 in location '61 does not generate an interrupt request via PILOO.

There are two ways of calculating flowrates, assuming the relation between flowmeter pulse rate and flow has been determined. For counter 1, the formulae are as follows; in units of pulses received per second.

1. Rate = (127-A(7))/B(48)
2. Rate = ((127-A(7)) * B(50) + B(49) - A(7))/A(0)

Formula 1 is an 'instantaneous' measure of flowrate if the counter interrupt time B(48) is calculated as close as possible to the sampling time. Formula 2 is an average measure over the entire previous scanning interval, hence the inclusion of the term A(0).

In previous versions of the HADIOS Executive, B(48) is determined for the first counter interrupt after every HADIOS scan but its value is only read at the next scan. While this is acceptable if the

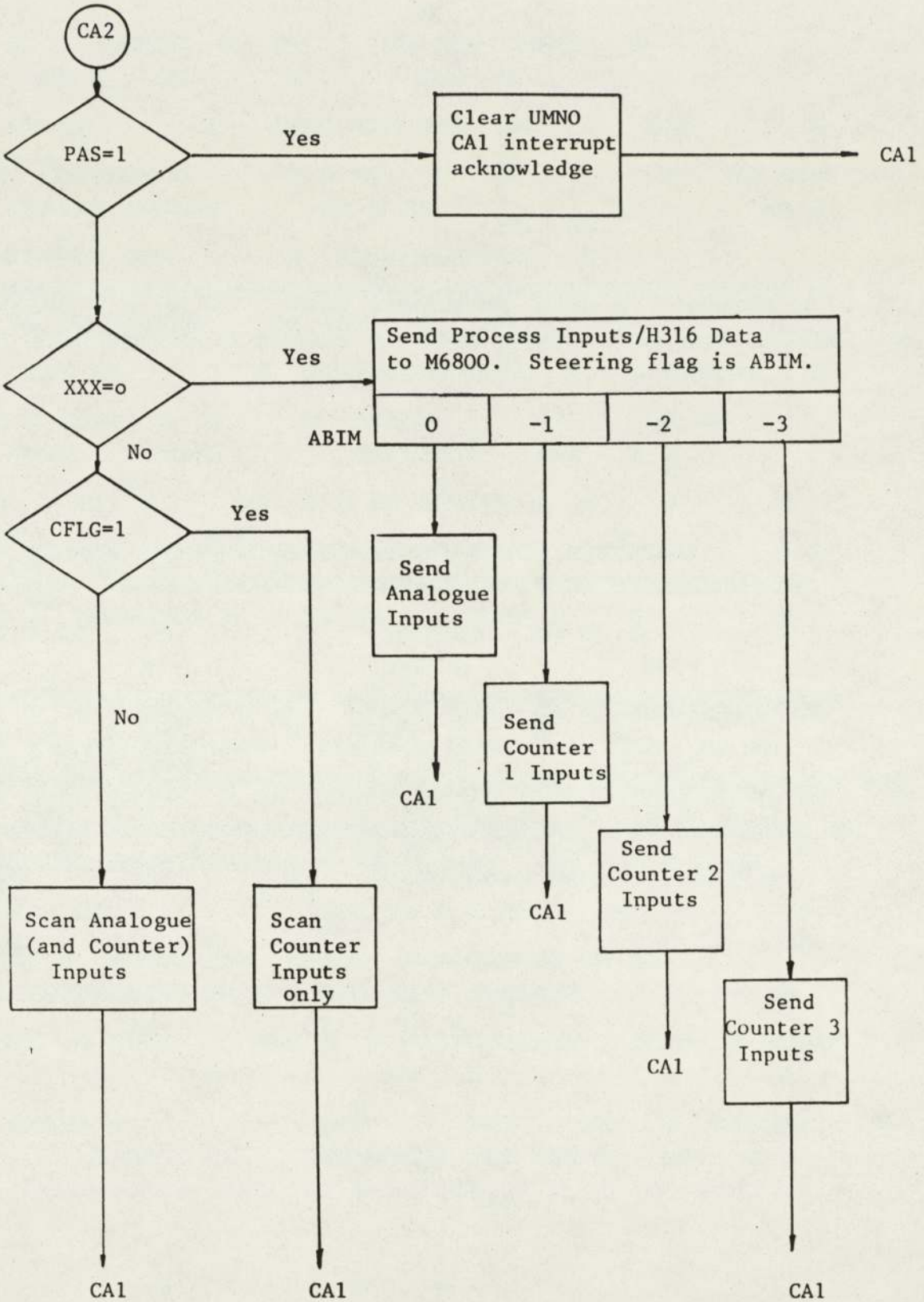


Figure 4.4 The CA2 Interrupt Flowchart

flow does not change appreciably over the scanning interval, misleading results can occur if the flow regime changes markedly. For this reason, the modified code upgrades location ITM1 (precursor to B(48)) at every counter 1 interrupt.

The modified code, like all interrupt response codes, has been kept as short as possible. The main problem in writing a workable code was in meeting the following specific requirements.

1. All counters depend on location '61 for measurement of real-time.
2. Use of a counter by either computer is mutually exclusive.
3. The contents of location '61 changes abruptly (by software) on a H316 scan or an error condition.

Tests conducted with a laboratory pulse generator showed that the modified code is a better approach to the measurement of flow while at the same time satisfying the requirements of the linked twin-processor system.

4.4.4 Error Handling and Sense Switch Usage

A proper error handling scheme is vital in the HADIOS Executive if it is to support a general, user-interative and real-time facility. The fifth software objective in Section 4.2 in fact requires that the H316 must not be in the HALT mode else M6800 real-time operation would have no practical meaning.

The existing scheme whereby an error condition would cause the program to exit to the BASIC command mode is extended to include several more error conditions. The error message has the format ERROR XY LINE NNN, where NNN is the BASIC line number and XY is one of the following:

BK - A program BREAK has occurred in the course of program execution. Sense switch 1 (S.S.1) has been set. Note that a BREAK prior entry to the Executive does not adjust location '61.

CE - The H316 user has specified a counter which is already under M6800 control.

GE - Relevant only when A(12) = 1. The H316 was still executing the interscan BASIC processing when the next relevant M6800 scanning interrupt occurs. It is therefore similar to error TF.

NC - The HADIOS controller has interrupted the CPU but the interrupting device was not a counter.

NM - An Alarm Input interrupt has occurred but it was not a CA2 or CB2 from the M6800.

NR - An unidentified interrupt has occurred.

RI - A Real to Integer conversion error has occurred in a library routine, i.e. the conversion of a real number outside the range -32768 to 32767 has been attempted.

TF - The H316 clock interrupt frequency is too small and BASIC code was still being executed when the next scanning interrupt arrives.

UI - Program execution has been terminated via a user interference i.e. S.S.2 has been set and detected in the DISPATCHER waiting loop.

Sense Switch Usage

S.S.1 Normally used for terminating a BASIC program. If the switch is set when H316 is in communication with the M6800, then program execution jumps to BASIC command mode via the HADIOS Executive error exit routine.

S.S.2 This switch setting is tested in the DISPATCHER waiting loop. If set, program returns to the BASIC command mode.

S.S.3 The INPUT statement reads data from cassette or paper tape.

S.S.4 The PRINT statement writes data to cassette or paper tape.

There are other error conditions that do not disrupt H316 operation. When they occur, the HADIOS Executive simply communicates the condition, to the M6800 at the next earliest scan of analogue or counter inputs. The M6800 Executive then takes the appropriate action to inform its user (see Section 4.5.3).

The HADIOS Executive also uses some FORTRAN library routines for its own use (for example FSAT, FSER, etc.) although the use of such routines has been minimised since many of the routines in the BASIC Interpreter MATHEMATICS PACKAGE are similar and therefore can be used. All these routines are non-reentrant and cannot be used simultaneously by two or more calling programs. This also means that FORTRAN subroutines callable from BASIC (for example the GRAPHICS segment or the DISK READ/WRITE facility) must be provided with their private collection of library routines. FORTRAN generated errors are therefore possible and these are distinguished from the other errors by the characters 'FT'.

4.4.5 Limitations

In providing a general purpose package such as the HADIOS Executive Package Revision 03, some flexibility is normally lost.

In particular, the sampling frequency A(0) is fixed on entry and remains constant throughout an on-line run whereas more intelligent executives would include a facility for variable sampling speeds in response, or in adaptation, to rapidly changing plant operating conditions.

Another limiation is that is is only possible to scan a single group of sequentially arranged analogue channels. If one is to scan channels 0 to 7 and channels 36 to 47 in one scanning routine then the

user has no choice but to scan channels 0 to 47. This means some time is wasted in scanning the irrelevant channels. To overcome this problem, channels for a single process plant are grouped together as far as possible.

The maximum ensemble number for any channel is 32. Recall that a group of sequentially arranged analogue channels is scanned as many times as possible over one clock resolution (20 ms). At the same time, the cumulative sum for each analogue channel is stored in a buffer. The ADC is a 10-bit device and the maximum digital equivalent for a 5 volt input is 1023. For thirty-two samples at this input level (which is practicable), the cumulative sum is 32736 which is within the upper limit for a 16-bit two's complement integer word. Clearly, the cumulative sum for thirty-three samples would be interpreted as a negative number, leading to meaningless results. This upper bound on the ensemble number is not included in the previous HADIOS Executive packages.

Since the HADIOS Executive and the Disk Read/Write routine must be core-resident, five locations of the BASIC Interpreter's CALL table (which is now located in the Executive) are permanently occupied. Therefore, only up to five other subroutine addresses can be accommodated. However, with the disk facility, different program segments can share the same address as long as only one program segment needs to be in core at any one time. The problem of CALL-table size is therefore regarded as an academic observation.

4.5 The M6800 Executive

The M6800 Executive is written in M6800 Assembly Language and followed similar lines to the HADIOS Executive. It proved to be reasonably straightforward to write given the previous experience on the H316 in writing such software.

As in the HADIOS Executive, the M6800 can be understood by first considering the specific functional objectives and discussing them in reference to a source listing provided in Table A4.2 of Appendix 4.

The M6800 Executive is designed to meet the following objectives:

1. To enable the M6800 to access HADIOS hardware at the required frequency.
2. To handle all interrupt requests to the M6800 MPU (i.e. $\overline{\text{NMI}}$, and device generated $\overline{\text{IRQ}}$, including batch data transfers to/from the H316).
3. To provide an idling loop to wait for the M6800 scanning interrupt (via the MP-T Interrupt Timer).
4. To provide a dispatcher table which contains pointers for special/background processing (for example, moving data from a buffer into an SD BASIC array).
5. To enable the M6800 user to output control settings to the process plant.
6. To terminate M6800 scanning when the required number of scans has been done.
7. To handle error conditions while not disrupting H316 operation.

These objectives are achieved by a combination of subroutines callable from SD BASIC, and the interrupt service routines. Together, they form the Executive which must be assembled together with the user's SD BASIC program.

The M6800 Executive is normally stored as a file on floppy disk. For real-time use, the user merely appends it to an application SD BASIC program. The arrays used are A (dimensioned 11) and B (dimensioned 86).

As with the HADIOS Executive, the M6800 Executive can operate either in the ON-LINE or OFF-LINE mode. The user HADIOS parameter and storage values take the same significance with the following differences.

1. Array A must be an integer array. For array B, only the elements corresponding to the channels scanned [B(A(3)) - B(A(4)) and H316 data transferred [B(57) - B(57 + M - 1)] must remain in integer format.
2. The control outputs must be converted into integer format first (using the INT function) before conveying them to the M6800 Executive.
3. The counter contents used by the M6800 (potentially B(49), B(52) and B(55)) are scanned as multiples of the RTC clock resolution (20 ms). The user must convert these to the required units through program.

As in the HADIOS Executive Package, there are basically four subroutines. However, a call to a house-keeping subroutine (CALL SUBO) must be made first. This prints out the high SD BASIC address (in hexadecimal) and the high address for the entire program segment (SD BASIC and M6800 Executive). This allows the programmer to examine data locations which by default (the data storage area can be assigned elsewhere via the DATA ORIGIN statement) begin at the top of the SD BASIC program segment.

SUBO also requests the user to specify which mode he wants the M6800 Executive to operate in. In the ON-LINE mode, the MP-T Timer is used to initiate M6800 scanning interrupts and the Executive is used in a way similar to the HADIOS Executive under normal operation. In the OFF-LINE mode, the MP-T Timer is not used at all. The M6800 may not independently access a process plant but can be programmed to engage

in asynchronous data transfer operations as part of a distributed computing activity between the two processors.

CALL SUB1 (A(0),B(0))

This is the basic scanning routine. It is similar to CALL (1,A(0),B(0)) in the H316 except that SUB1 is called once in every scan. Note that the addresses of the arrays A and B are transferred on to the M6800 stack in the reverse order as they appear in the argument list.

CALL SUB2

This subroutine call is similar to CALL(2) in the H316 except that if the number of M6800 scans required is still not reached, the program always return to SD BASIC where it jumps into the DISPATCHER loop (via SUB1) to wait for the next scanning interrupt.

CALL SUB3(N,U)

This subroutine is similar to CALL(3,N,U) in the H316 except that in the M6800, N and U must represent integer values. As in CALL SUB1 (A(0),B(0)), the argument addresses are pushed on to the stack in the reverse order (see Figure 4.5). Recall that in SD BASIC, the address of a variable (as the numerical value itself) takes up six bytes. When a CALL is executed, the index register points to the first byte of the address of the last argument transferred while register A contains the number of arguments involved.

N = output channel number of DGOA (0-15)

U = 0 to 32767 corresponding to 0-10 volts

CALL SUB4(M,D(0))

This subroutine transfers M consecutive data words (two bytes each) located in INTEGER array D which is dimensioned 29 (buffer size defined by BUFM in the HADIOS Executive). The first CB2 output in the

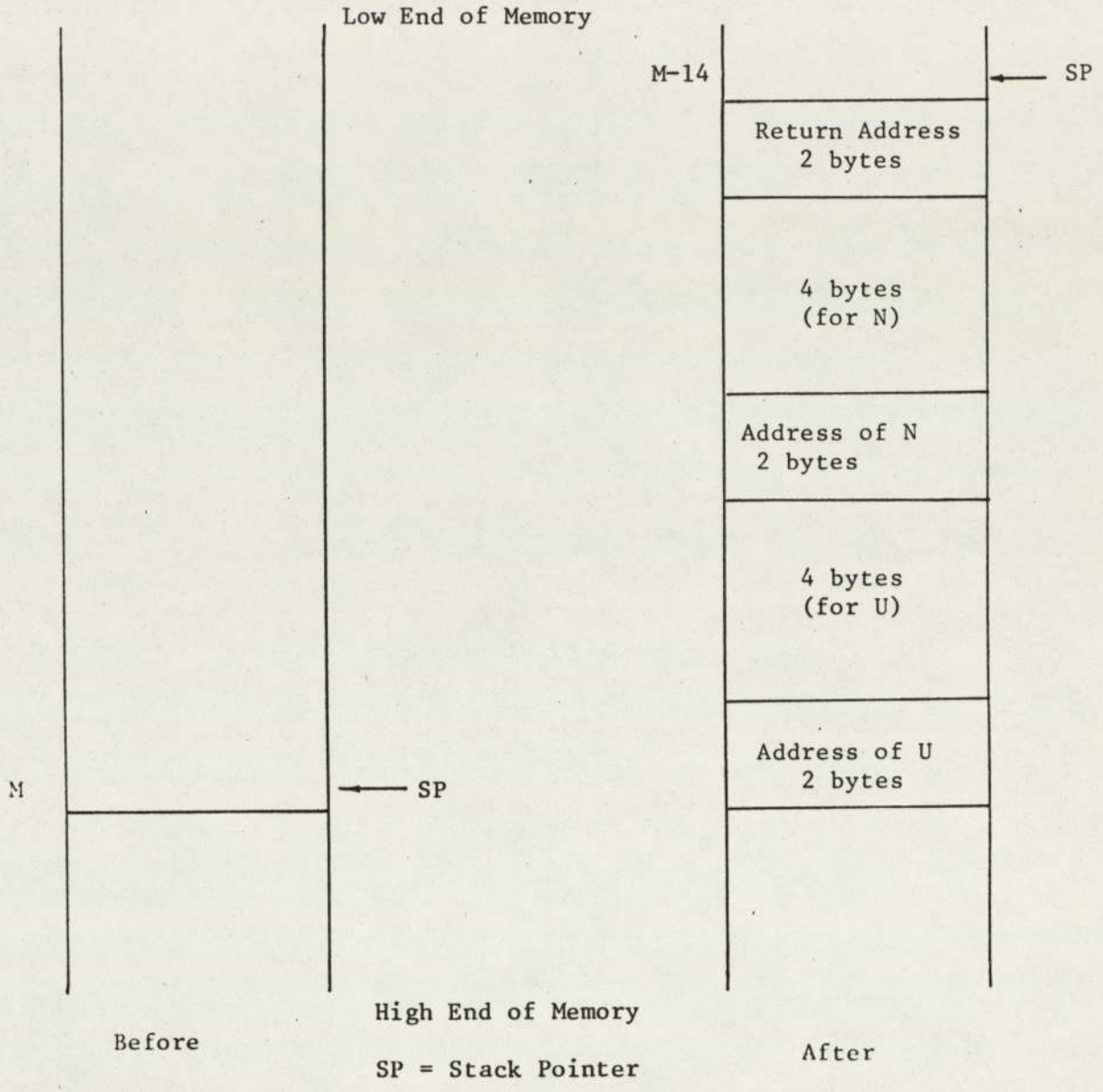


Figure 4.5 M6800 stack before and after CALL SUB3(N,U)

sequence transfers a code byte (254) which prepares the HADIOS Executive for the subsequent data transfer. Like all data transfers to the H316, the CB2 interrupt is used to inform the H316 that a data byte is available and the H316 acknowledges receipt via a CB1.

4.5.1 Interrupt Handling

A M6800 Executive basically handles $\overline{\text{IRQ}}$, $\overline{\text{NMI}}$ and SWI interrupts. $\overline{\text{IRQ}}$ interrupts are generated by the MP-T Timer (every second if the

reserved SD BASIC variable TIMER is set to $06, an integer multiple of which is the sampling interval) or by the H316 via a low active CA1 transition to initiate a H316 to M6800 batch data transfer operation.

Unlike the HADIOS Executive, the M6800 Executive does not have to allocate buffer locations to store the machine state of the interrupted program. This is automatically done by hardware which conserves the status on to the stack. The interrupt handling routine is therefore straightforward and as each type of interrupt ($\overline{IRQ}$ or $\overline{NMI}$) is channelled through its own dedicated vector, the Executive only has to poll the expected sources to identify the active one.

Figure 4.6 shows the interrupt handling flowchart as used in the M6800 Executive. The polling method is used to sort out the $\overline{IRQ}$ interrupts, the MP-T Timer being tested first (IRQB1 flag set?) as it is the more likely one under normal ON-LINE conditions.

There is only one $\overline{NMI}$ interrupt source i.e. via the NMI button at the computer front panel. It is used to cause a user interrupt in a manner similar to the functions of sense switches 1 and 2 in the H316.

The SWI interrupt is used as a run-time debugging aid. In using the system, spurious CB1s (or CA1s) may be generated when data is transferred from the DGOA to the PIA A Side. Because the PIA B Side is usually programmed in the handshake mode, a spurious low active CB1 sets the IRQB1 flag. This premature setting causes the next STAA IORBO (send a low active CB2 to the Alarm Input subinterface) instruction to be ineffective. Subsequent CB2s are effective as the IRQB1 flag would have been reset by a LDAA IORBO instruction. As a result, the M6800 Executive 'thinks' that it has sent out all the required CB2s whereas the HADIOS Executive has actually received one less and the effect on the CB2 interrupt response code is of course, disastrous. Two SWI instructions

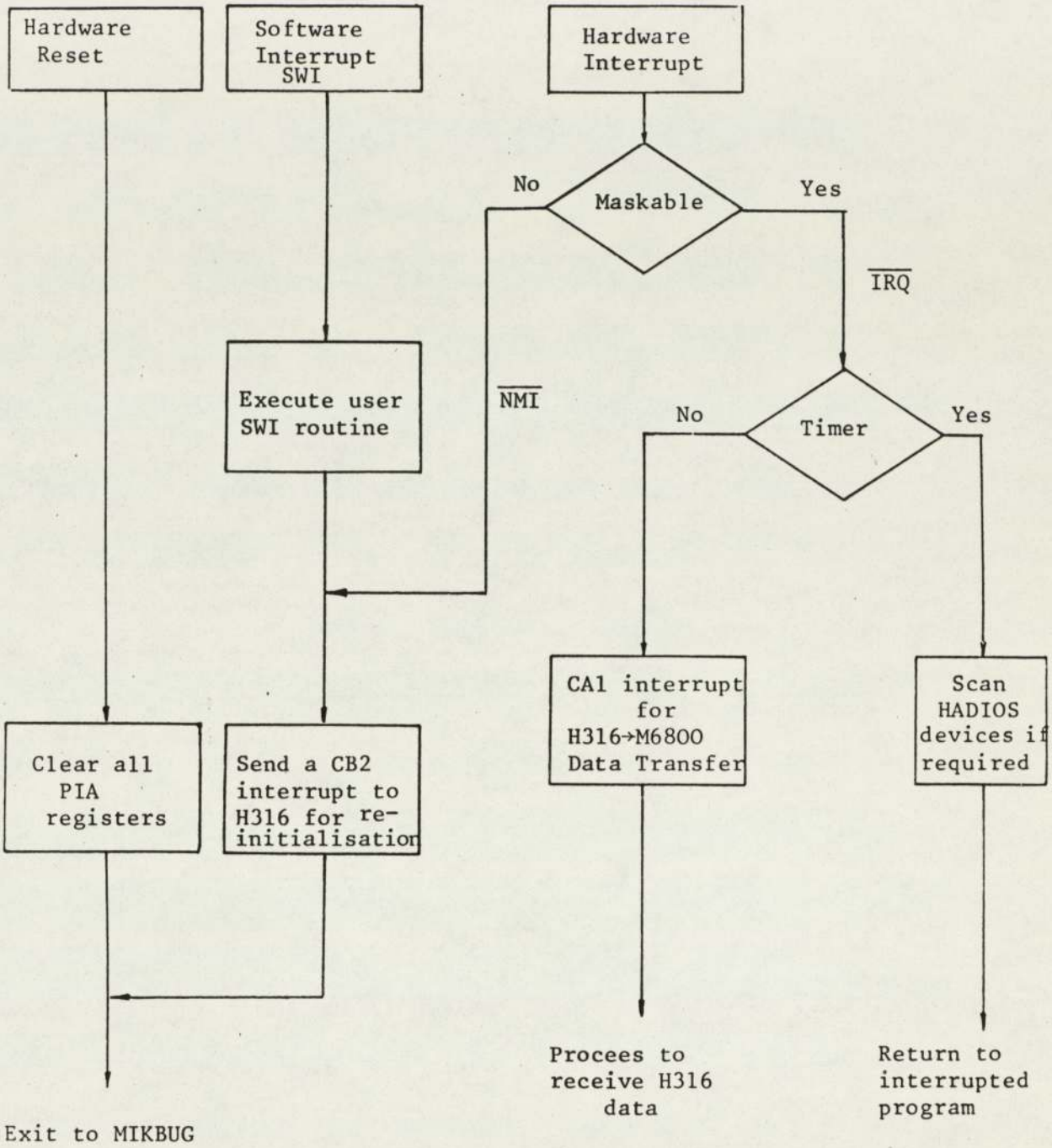


Figure 4.6 Interrupt Handling Flowchart in the M6800 Executive

are used to check a spurious CB1 (or a CA1). The first SWI interrupt response prints a message and restores the MIKBUG SWI vector. The second, which is MIKBUG's own, prints the register contents before passing control to the monitor.

4.5.2 Error Handling

The SWI error handling routine has been described earlier. It means an interface hardware check is necessary. As far as the M6800 user is concerned, the other error conditions may encompass the following:

1. Initialisation errors detected by the M6800 Executive.
2. SD BASIC Run Time Package (RTP) errors.
3. FORTRAN errors in the library routines used by the M6800 interrupt response code of the HADIOS Executive.
4. A user interference via the NMI button.

There are basically two types of errors: those that necessitate initialisation of CB2/CA2 response code in the HADIOS Executive and those that do not. In the first category, when an error is detected, a special data byte (255) is sent to the H316 via a CB2. All relevant program flags and buffer locations in the HADIOS Executive are set to their initial values so that the M6800 can re-start its communication protocol at any subsequent time.

Run-time (SD BASIC) errors are channelled to the M6800 Executive via SUB99 (EN,LN) where EN and LN are assigned the error and line (last line number encountered) numbers respectively, and the SD BASIC "ON ERROR GOTO" facility. Run-time errors in the HADIOS Executive which concerns the M6800 include the specifying of a counter currently being used by the H316 user and FORTRAN library routine errors. These errors are only passed on to the M6800 Executive for recognition at the next scanning interrupt.

User interference must be via the NMI button once the M6800-H316 protocol has started. This is because a hardware reset clears all M6800 CPU and PIA registers as well which means there is no way to initialise the HADIOS Executive for M6800 interrupts except by stopping H316 program execution and patching the flags and buffers by hand.

SD BASIC can recover from input errors by responding with a "INPUT ERROR?" message and ignores superfluous input data before the carriage return key is pressed. Each CTRL/O (the CONTROL and O keys together) can also be used to 'delete' a previous character input.

The M6800 Executive error messages are self-explanatory (refer to the assembler listing in Table A4.2 of Appendix 4). Of course a more user friendly package can be written at the expense of more programming effort and space requirements. In the context of this work, that extra feature does not seem warranted.

4.5.3 Limitations

As the M6800 Executive is designed along the lines of the HADIOS Executive and to provide the M6800 user with the ability to access the HADIOS system, most of the limitations discussed in Section 4.4.5 also apply to the M6800 Executive. Other constraints are reflective of the fact that the M6800 is a linked system to the H316-HADIOS hardware via a single PIA chip. The M6800 user has no access to the plant without using some H316 machine time. This and other constraints caused by limited hardware and supporting software will be discussed in a later chapter. However, M6800 sampling times can go well below 1 sec. (H316 minimum) by specifying the appropriate MP-T Timer control code through the variable TIMER in SD BASIC.

4.6 Modifications to the BASIC-16 Interpreter

The original BASIC interpreter is still available but its utility is limited Input/Output is only via a TTY and it cannot run in an interrupt environment. To meet the requirements of the twin-processor system several modifications are necessary. These patches are easily made since an assembler source listing of the interpreter is available.

Figure 4.7 gives an overview of the memory utilisation by the interpreter. The initialisation routines (INIT.A) are only used once and can therefore be used to store program text. The user's BASIC program is stored in a compressed form to facilitate interpretive action during program execution.

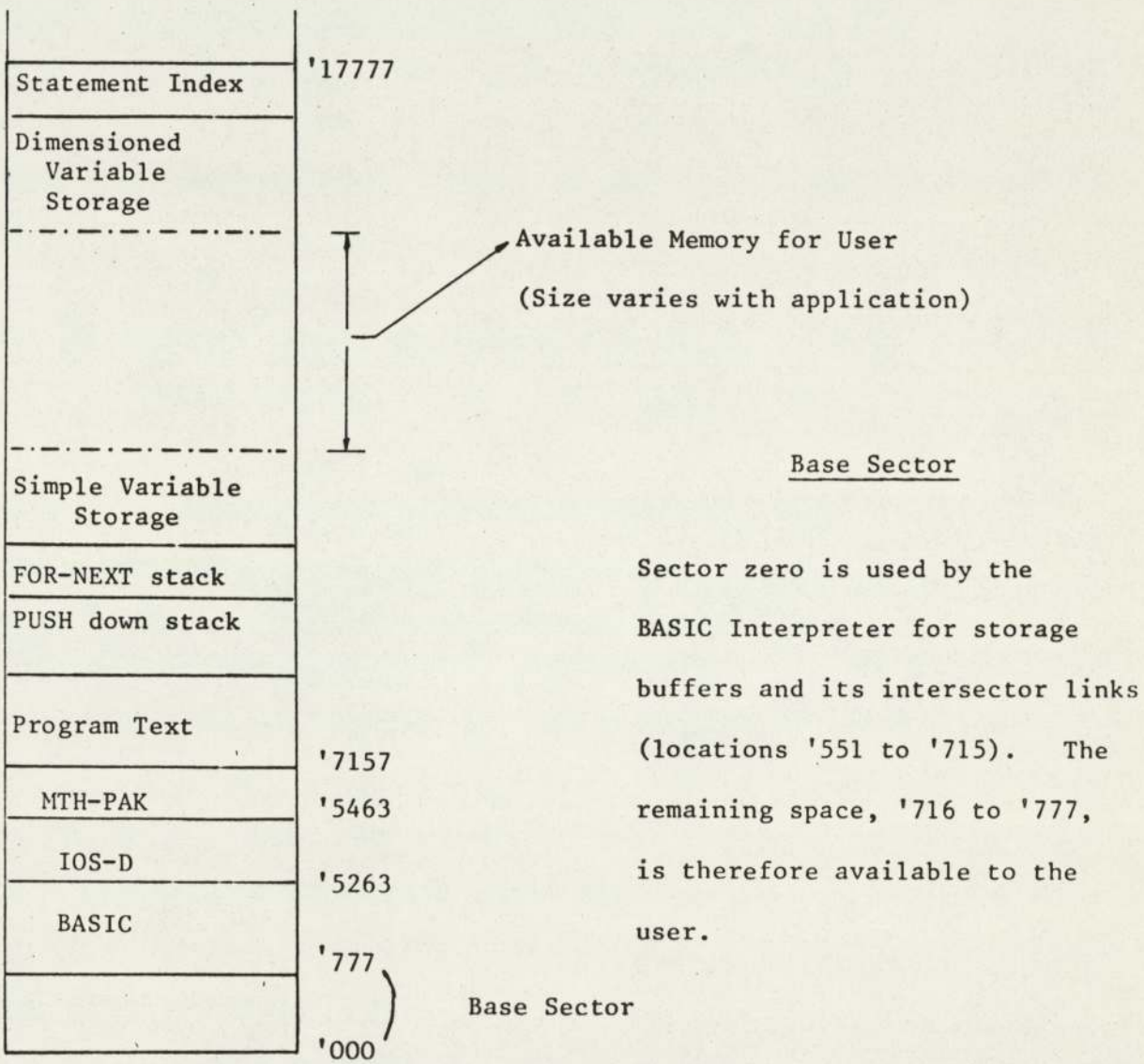


Figure 4.7 Memory Utilisation by the H316 BASIC Interpreter

The ADT1-8 Table

The ADT1-8 Table (assigned dimensioned variable table) has been shifted to location '757 to '766 to allow the use of locations

'61 and '63 for real-time applications. This is done by the assembly language program listed in Table A4.3 of Appendix 4.

Initialisation

Locations '166 through '180 are patched so that an initialisation, the message "HADIOS EXEC 03 1983" is output to the VDU. Other locations patched are indicated below:

Location		Original	Patch (Mnemonic + Code)	
'7240		JST TYPE	NOP	'101000
'7241		XAC HMAM	NOP	'101000
'7242	INO3	JST TYPE	NOP	'101000
'7243		XAC AYOH	NOP	'101000
'7245		LDA C241	JMP '7301	'003301

where '7301 contains the instruction LDA HOA, and HOA (location '7367) is the High Octal Address initialised for this system to 17777.

The I/O MOD

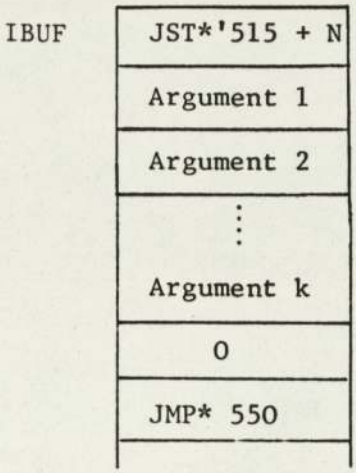
The I/O MOD program patch allows the BASIC interpreter to read in data via a paper tape reader (PTR) when an INPUT statement is executed and Sense Switch 3 set. Similarly, if sense Switch 4 is set, a PRINT statement will output characters or data to the paper tape punch (PTP). The I/O MOD as listed in Table A4.4 of Appendix 4 is written in Relocatable format. In this work, the I/O MOD is loaded into locations '720 through '752 (the loader usually loads from an even address location). This is because location '550 through '715 are used by the interpreter for its base requirements.

The CALL statement

The BASIC interpreter keeps a ten-word CALL table in locations '516 through '527 where the addresses of subroutines callable from BASIC are normally hand-patched. Alternatively a software method can be adopted

as done in the HADIOS Executive (refer to its CALL table).

When BASIC encounters a CALL statement, basically three things are done as illustrated below before a jump to IBUF is made from location '4012.



1. JST* '515 + N instruction is placed at the start of the calling sequence i.e. in the input buffer, IBUF.
2. If more arguments follow after N, then their addresses are placed in IBUF+1 onwards.
3. A zero word is inserted after the k^{th} argument for FORTRAN compatibility and the sequence ends with an entry into BASIC through location '550.

Therefore if the HADIOS executive is to maintain its own CALL statement processor and table, then location '4012 should be patched with a JMP* '716 where '716 contains the beginning address of such a code.

Error routines

The normal entry-point of the BASIC error handling routine is '5243, accessible from all parts of the interpreter outside sector 5 through a base link in '551. However, the linked processor system requires a certain amount of housekeeping prior exiting to the command mode via '5243. The address of the HADIOS Executive error housekeeping routine (ERCL) is therefore patched into location '551. The following patches have still to be made as they are jumps to the error routine from sector 5 itself where the base link is not necessary.

Location	Original	Patch
'5133	JST '5243	JST* '551
'5742	"	"
'5744	"	"

The program 'BREAK' needs a further consideration since a 'BREAK' can occur before or after the program has entered the Executive. If the break is made before entry then no house-keeping by the Executive is necessary else the 'BREAK' is processed like any other error. The 'BREAK' exit to the command mode in location '3267 (JMP CMOD) is therefore patched with a JMP* '717 where '717 contains the address of the 'BREAK' handling routine in the Executive.

Special locations ('63 and '2540)

Location '63 is patched with the standard interrupt vector (SKST). Location '2540 has a more subtle significance. The relevant code is shown below to make the point. However, the 'locate simple variable' routine (LSV) returns the program counter to '5237 is the research is not successful

Location	Original
'5236	JST LSV
'5237	EX15 JST ERR
'5240	BCI 1,UV
'5241	JMP EX14

thereby reporting an 'Undefined Variable' error, or to '5240 if the search was successful. Location '5240 is therefore interpreted as an instruction in the latter context. The Honeywell documentation apparently acknowledges this programming mistake and says that the "BCI 1,UV" (or '152726) will be interpreted as an ERA (Exclusive OR) instruction.

However, program safety cannot be assured as both bit 1 (the indirect address flag) and bit 2 (the index bit or tag) are both set. If the index register contains a bit 1 set then it indicates another level of indirect addressing, and the process can remain in an indirect loop indefinitely. For this reason, location '5240 is patched with '052726 i.e. with the indirect addressing bit reset.

BASIC IOS-D Patch

The following 'Jump and Stores' were placed into the two locations of the BASIC IOS-D module to handle the software patch required when using the Newbury terminal and printer.

Location	Mnemonic
'5344	JST TEST
'5365	JST TEST

where TEST = '753 (See also Table A7.3 of Appendix 7.)

4.7 H316 Tektronix Graphics Package

The Graphics Library routines have been grouped together to form a single segment forming subroutine 5. A steering subroutine called GRAPH(124) listed in Table A4.5 in Appendix 4, allows the user to access the different graphical operations from a single CALL(5,N,C(0)) by specifying parameters N and elements of the C array (dimensioned 7) as follows:

N = 1 Enter graphic mode
= 2 Set the windows
= 3 perform graphic functions (draw, move, etc.)
= 4 to invoke the cursor facilities
= 5 output alphanumeric characters to screen
= 6 leave graphic mode

For N = 3 A(0) = 1 dark move
= 2 move and draw the point
= 3 draw a line connecting the initial and final point

The x and y coordinate values are conveyed through A(5) and A(6) respectively. A(0) = 4,5 and 6 also perform similar operations if Δx and Δy are given.

A(1),A(2) = minimum value and range of x coordinate

A(3),A(4) = minimum value and range of y coordinate

A(5),A(6) = values of x and y to be plotted

A(7) = to output text on the screen or to invoke the cursor facility.

The Graphics Package was originally written for the Tektronix 4010 terminal. Using the more advanced Newbury 8510 multi-page terminal, two control operations are needed to enter and leave its graphic input mode.

C(7) = 29, N = 5, CALL(5,N,C(0)) - enter Newbury 8510 graphics mode

C(24) = 24, N = 5, CALL(5,N,C(0)) - leave graphics mode to return to standard mode.

The graphics package was constructed as a self-contained program segment. This would allow the segment to be stored on floppy disk and loaded into core when required. The graphics package uses some COMMON areas and these have been assigned to the top of memory ('37506 - '37777) where they remain core-resident. It was decided that to maximise use of available memory, the graphics segment should be loaded into sectors '20 through '26. To achieve this, the 37 subroutine object tapes making up the basic graphics facility, are divided into suitably-sized groups using the 'OBCHOP' utility. The package requires its own FORTRAN library routines although some space is saved by using BASIC MTH-PAK routines where possible. The FORTRAN error routine F$ER requires a different patch depending whether the graphics is used in conjunction with the HADIOS Executive or not. (See Section 4.9.)

The loading procedure of the graphics package with the resulting memory map are described in Tables A4.6 and A4.7 of Appendix 4.

4.8 Special FORTRAN and Utility Routines (FER, FHT, BASIC MTH-PAK POINTERS and SU10

F$ER and F$HT

Recall that the M6800 interrupt response code of the HADIOS Executive requires its own FORTRAN library routines. While this in general means duplication, the F$ER and F$HT error routines need modifications to suit their special functions.

F$ER and F$HT used by the H316 clock interrupt response code or graphics need a patch such that they exit via the HADIOS Executive error handling routine (ERCL). If the Executive is not being used such as in off-line simulation and/or graphics, then the routine exits via the BASIC error routine.

F$ER and F$HT used by M6800 interrupt response code must not disrupt any H316 program execution. They are therefore modified so that on exit, error codes (ERRM and MCOD) are eventually passed on to the M6800 (at the next M6800 scanning interrupt) for recognition.

BASIC MTH-PAK Pointers

Many of the BASIC MTH-PAK routines are similar to their equivalent FORTRAN counterparts. They can therefore be used to minimise duplication. There is one difference as far as the real-to-integer routine (C$21) is concerned. The BASIC C$21 reports an error (via ERCL, patched into location '551 of BASIC) if the conversion results in a value outside the H316 integer range. The FORTRAN C$21 apparently does not. The various MTH-PAK pointers are taken from the BASIC interpreter and passed on to the loader in the form of assembly language EQUs.

SU10

This DAP-16 MOD 2 routine is designated as subroutine 10 and its

beginning address therefore occupies location '527 in the BASIC CALL table.

The routine is called from BASIC as CALL(10,X1,X2,X3) where

X1=0 Output 24 frames of paper tape (leader)

X1≠0 Using the H316 RTC as a timer

X2=0 start the clock with a large negative number

X2≠0 stop the clock

X3 = time elapsed in seconds.

Assembler listings of all the above routines are found in Table A4.8 through Table A4.11 respectively in Appendix 4.

4.9 Software Execution Times

In many on-line control applications, computer speed is often a vital factor in achieving good performances. The faster a digital controller determines the error status of a controlled variable, the quicker calculated (and optimised) corrective actions can be made.

This section compares the software execution speeds of the H316 and M6800. The specified performance based on machine level instructions is discussed first, followed by results from a simple BASIC and FORTRAN program benchmark test.

Specified Performance

Computer performance is not easy to define in exact terms, as it is determined by both hardware and software architectures of the machine which in turn are function of a rapidly changing technology. Nevertheless, a comparison of machine code execution times is a useful indication.

Since the H316 and the M6800 are basically stored-program computers, instructions are fetched from memory and decoded by the internal logic of the CPU in a clock synchronised operation. The nominal memory cycle time in the H316 is 1.6 μs which means loading a value (LDA) into the A-register, adding it with the contents of a memory location (ADD) and then storing the result (STA) would take 9.6 μs as each of these instructions takes 2 cycles or 3.2 μs. In terms of cycle times, the M6800 is therefore a faster machine since with a clock frequency of 2 MHz, it can load (4 cycles), add (4 cycles) and store (5 cycles) in the extended addressing mode in 6.5 μs. However, if a 16-bit data byte is involved, the M6800 basically repeats the process and that would require 13 μs. Hence, on a per bit basis the H316 is faster.

The variation in instruction execution speed is more marked in the M6800. There are 5 addressing modes generating 1, 2, and 3-byte instructions which, as a result gives rise to 197 different instructions. Generally, those instructions in the indexed mode are among those with the higher cycle times required. The SWI instruction has the longest cycle time (6 μs) as the instruction needs to be decoded and then the seven byte machine status pushed on the stock.

BASIC Benchmark Test

As the user(s) of the system under study would be programming in BASIC in either computer, it would be useful to compare the execution times in running a short BASIC program. This can be done by using the RTC (in the H316) and the MP-T Interrupt Timer (in the M6800) to mark the passage of real time.

Listings of the three benchmark programs, with their associated subroutines if any are shown in Tables A4.12 to A4.15 respectively in Appendix 4. Each program comprises the same number of looping operations (IF...THEN...ELSE only in SD BASIC), similar GOSUBs and arithmetic operations. The outermost loop index (FOR...NEXT in BASIC, DO loop in FORTRAN) N and the variable VALUE (or V) must be specified by the user. Note that VALUE can be integer or floating-point formatted in SD BASIC F1 and F2 in the SD BASIC program are MP-T Timer control codes. When F1 = $03, the timer interrupts the M6800 every 1 ms thus its time base. Altogether each program was run 10 times covering values of N from 0 to 4 (or 1 to 5 for the FORTRAN program) and Table 4.1 shows the execution times obtained.

Table 4.1 Benchmark program execution times in the H316 and M6800 (secs.)

N	VALUE	M6800	H316 BASIC	H316 FORTRAN
0	2	3.361	8.92	.42
0	1.556	4.406	8.96	.42
1	2	6.694	17.78	.84
1	1.556	8.810	17.88	.82
2	2	10.082	26.66	1.22
2	1.556	13.295	26.82	1.24
3	2	13.443	35.54	1.64
3	1.556	17.811	35.76	1.62
4	2	16.803	44.42	2.06
4	1.556	22.020	44.68	2.04

Note: For FORTRAN program $1 \leq N \leq 5$

The following observations can be made:

1. SD BASIC runs 2 to 2.5 faster than BASIC but about 8 times slower than FORTRAN code implemented on the H316.
2. In SD BASIC, the execution times increases by about 30% if floating-point arithmetic operand is being used. The difference in the H316 is very slight.

The first observation is due to the fact that SD BASIC is a compiled language. An SD BASIC source program is compiled into a series of coded numbers (not syntax) which are interpreted at run time. In the H316, the BASIC interpreter has to carry out a line syntax (although stored in compressed form) before execution takes place. Compiled FORTRAN is much faster as the object code produced is usually an efficient machine code program.

The second observation is partly explained by referring to the structures of integer and decimal floating point variables in SD BASIC (Section 4.3.2.2). Because operations are carried out on the fractional and exponential parts of a floating point number, more cycles are used.

4.10 Construction of the HADIOS Executive Package Rev 03

The BASIC Interpreter, the HADIOS Executive and the Graphics Package can now be combined together to form a general purpose software package which has options for graphics and communication with the linked M6800 microcomputer system.

To construct the package, the three SLSTs (BASIC + HADIOS + GRAPHICS) were first prepared. The BASIC Interpreter is already available in SLST form. It only required the modifications described

in Section 4.6. The Graphics Package was loaded into locations '20006 through '26774 (see Table A4.7 of Appendix 4). The first six locations of sector '20 has been reserved for a dummy subroutine called KOMMON which comes into significance in constructing the Kalman Filtering simulation package but this aspect will be discussed in Chapter 6.

Finally, the HADIOS Executive was loaded and together with subroutine 10 (SU10) and the required FORTRAN library routines. The core image eventually occupies about six sectors i.e. from '20000 to '34762. The loading procedure and the resulting memory map are fully described in Tables A4.16 and A4.17 respectively in Appendix 4.

The addresses of subroutines 5 and 10 (GRAPH and SU10) are then patched into locations '522 and '527 of the BASIC Interpreter's CALL table. The final package therefore consists of a two-part SLST tape namely the BASIC Interpreter ('63-'7415) and Graphics/HADIOS Executive ('20000-'34762). Copies were made on paper tape and on floppy disk.

4.11 Conclusions

This chapter has described the philosophy and the technical effort required in providing a general purpose software package for the linked H316-M6800 twin-processor system. A user's manual including several demonstration programs has been prepared elsewhere[94] although some examples and potential operating problems will be discussed in Chapter 7. The package is relatively simple to use and also provides a facility for on- or off-line distributed data processing.

Table 4.2 below summarises the four operating modes of the system. Note that the term ON- or OFF-LINE is with respect to the process plant.

Table 4.2 Operating Modes of the linked H316-M6800 Twin-Processor System

H316	M6800	Remark
ON-LINE	ON-LINE	When the M6800 is in off-line mode, the
ON-LINE	OFF-LINE	PIA A Side is configured into the interrupt
OFF-LINE	ON-LINE	mode and the H316 sends data via a CA1
OFF-LINE	OFF-LINE	interrupt. Otherwise, the PIA A and B Sides are always in the handshake mode.

Inter-processor data transfers and control outputs are possible in all modes of operation. Also, it is stated earlier (Section 4.2) that the use of floppy disks to overlay program segments is incorporated in the software design strategy. The implication of this facility on the HADIOS Executive Package Rev. 03 will be discussed in a later chapter.

CHAPTER FIVE

THEORETICAL DEVELOPMENT OF A KALMAN FILTERING APPLICATION

5. THEORETICAL DEVELOPMENT OF A KALMAN FILTERING APPLICATION

Estimation theory plans an important role in the application of modern control theory to industrial systems. This has been attributed to the fact that in many industrial processing systems, the total state vector can seldom be measured and the number of plant outputs is much less than the number of states. In other cases where the state vector is measurable, the measurements are often corrupted by significant experimental error which precludes its use in control system design. In addition, the measurement process introduces delays and time lags into the control loop while the process itself is subjected to random, unmodelled disturbances.

This chapter describes the application of a sequential estimation technique which was first formulated by Kalman[95] and later with Bucy[96] at the turn of the 1960s. Known as the Kalman filter, this estimation technique has been widely used in aerospace and electrical systems, and to a lesser extent in chemical process systems.

The discussion which follows will only present the Kalman filter in a formal way so as to highlight the basic features of an operational filter. This is because a full and rigorous treatment of these topics requires a thorough background in the theory of stochastic processes as well as classical, deterministic optimal control. The texts written by Doobs[97], Bryson and Ho[98] and Jazwinski[99] appear to be the standard references for this purpose.

5.1 Introduction to Kalman Filters

Quite apart from its theoretical importance, the Kalman filter is now regarded as a highly practicable technique for state and parameter

estimation. For a linear dynamic system, the theory is on a rigorous basis but an extension of the filter, by means of local linearisation, has been a useful approximation when applied to non-linear systems.

The fundamental idea in a Kalman filtering problem is to determine the state variables for a given process from only a knowledge of the outputs (plant data) and the inputs (controls, disturbances, etc.). The underlying assumption is that the process is modelled by a first-order, linear, vector-differential equation excited by an additive random noise and that some measurements which are some combinations of the state variables, are also available, corrupted by experimental error. This is schematically shown in Figure 5.1.

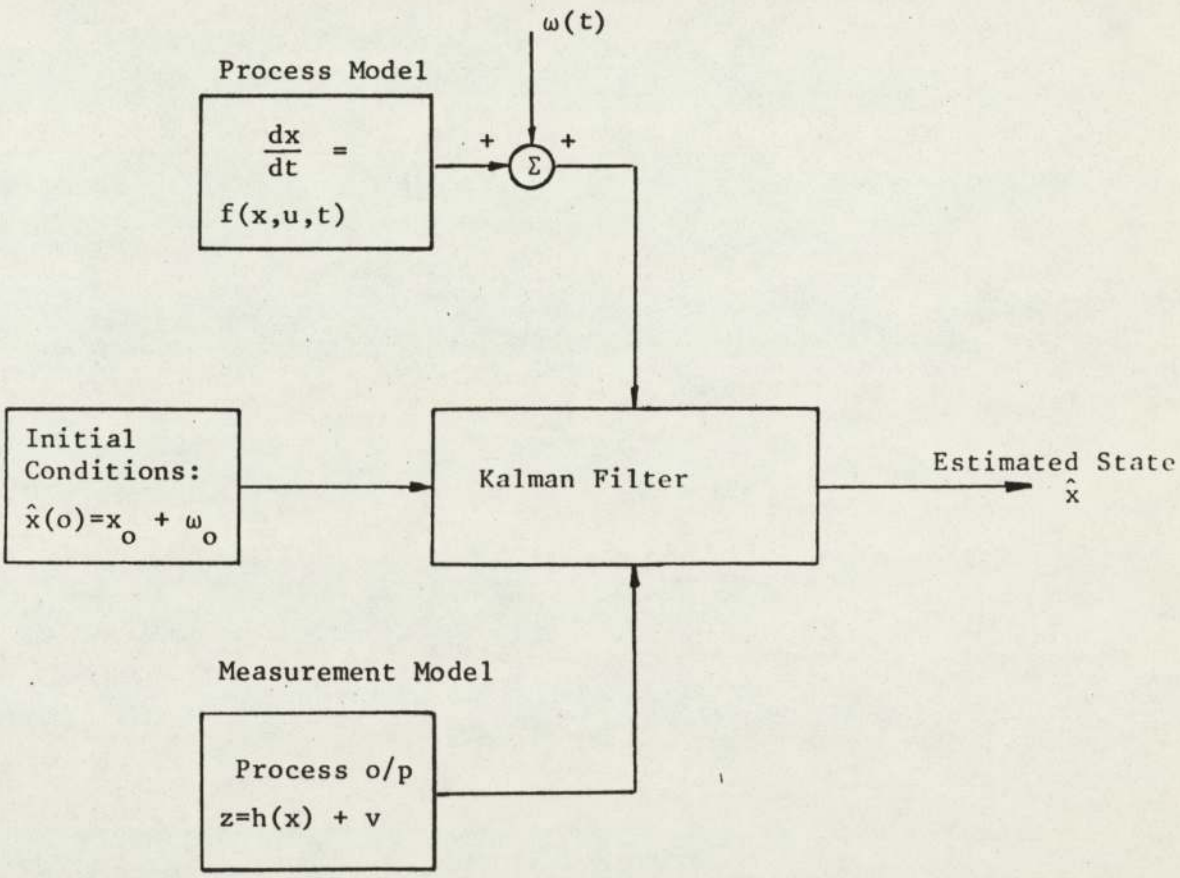


Figure 5.1 Schematic Formulation of a System for Kalman Filtering Applications

The need for some measurements may be intuitively obvious. Since x (and z) is now a random variable, and in general a function of t, it is possible to envision a time evolution of the distribution of possible process states resulting from the stochastic model. This is provided by its probability density function p(x,t) and for the one-dimensional case this is shown schematically in Figure 5.2a. However, if measurements z are available, then it is possible to consider

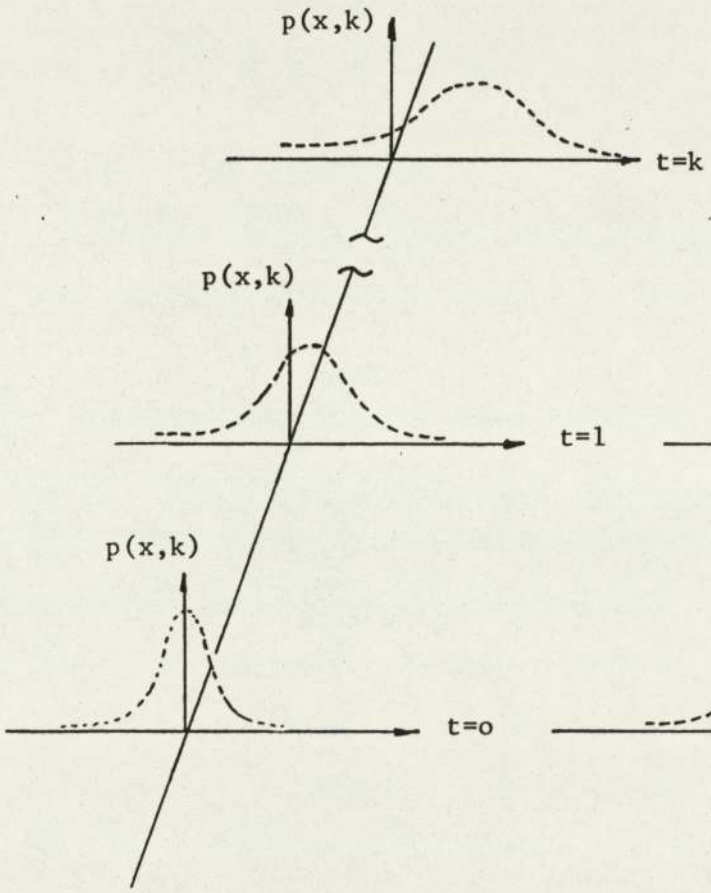


Figure 5.2a Time evolution of p(x,t) without observations

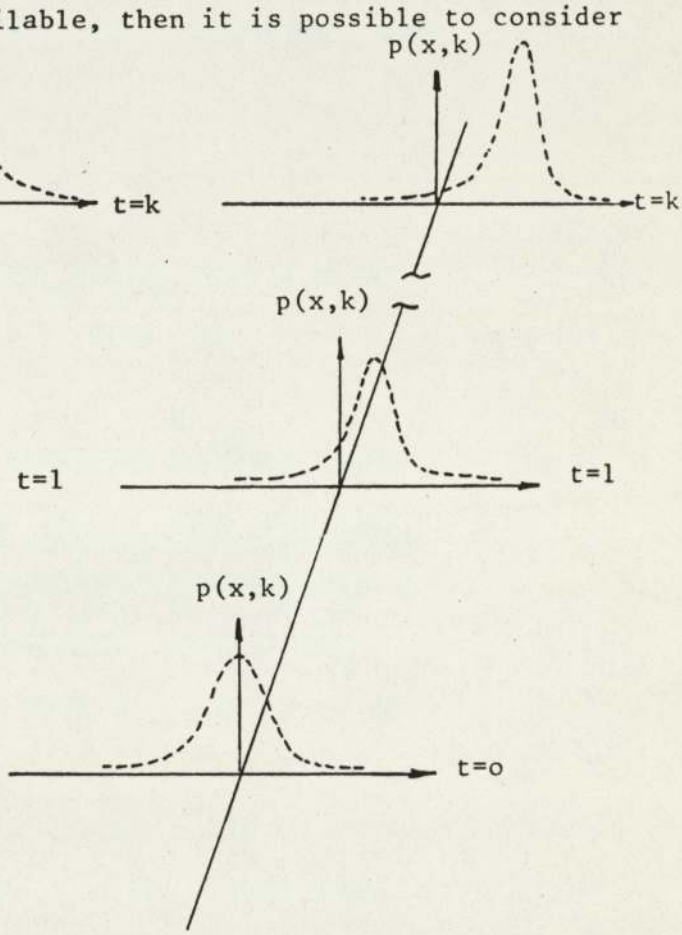


Figure 5.2b Time evolution of p(x,t) with plant observations

the conditional probability distribution p(x(t)/z), which is the probability distribution of the state given the set of measurements $z(t')$, $0 \leq t' \leq t$. With proper use of plant information one can devise a procedure to improve the estimates with time as shown in Figure 5.2b.

The Kalman filter algorithm is one such procedure to obtain the best (optimal) estimate of the true state according to some predefined cost functional. The only difference is that for a linear system, the Kalman filter is the best of all from the class of both linear and non-linear estimators. In general, a different optimisation criterion yields a different estimate (maximum likelihood, minimum variance, least squares, etc.) but if the noise is Gaussian, they are all identical.

The engineering practice of using a probabilistic approach to modelling and uncertain measurements is adopted because there is no other alternative approach that can match it in terms of its extensive mathematical theory and sophistication. In particular, attention is focussed on the use of continuous, Gaussian, white noise. This means its density function is uniquely defined by its mean and variance which in this case, is infinite. Thus, past measurements do not help in improving future predictions or estimates. In other words, the design is based on a 'worst case' situation (infinite variance) and future predictions, often by complex algorithms and heavy computational load, are minimised.

Furthermore, primary macroscopic sources of random phenomena are independently Gaussian since these macroscopic random effects may be thought of as a superposition of very many microscopic random effects, regardless of individual statistical properties. The assumption is therefore reasonably practical in most cases.

5.1.1 Linear Systems

5.1.1.1 Continuous-time Systems

The process and measurement models assumed are as follows:

$$\frac{dx(t)}{dt} = A(t)x(t) + B(t)u(t) + D(t)\omega(t) \qquad (5.1)$$

$$z(t) = M(t)x(t) + v(t) \tag{5.2}$$

where $x(t)$ is the state vector (nx1)

$u(t)$ is the deterministic control input vector (ℓx1)

$\omega(t)$ is the process noise vector (rx1)

$z(t)$ is the measurement vector (mx1)

$v(t)$ is the measurement noise vector (mx1)

$A(t)$ is the system matrix (nxn)

$B(t)$ is the input driving matrix (nxℓ)

$D(t)$ is the process noise driving matrix (nxr)

$M(t)$ is the measurement matrix (mxn)

In fact without the noise terms, one can immediately see the structural similarity with optimal control problems as using the control effort $u(t)$, one can force the time evolution of $x(t)$ so as to minimise the departure from a known reference trajectory $x_o(t)$.

The following assumptions are usually used to solve the problem.

1. The initial state vector is Gaussian with known mean and covariance.

$$E\{x(0)\} = \bar{x}(0) \tag{5.3}$$

$$\text{cov}[X(0),x(0)] = E\{(x(0) - \bar{x}(0))(x(0) - \bar{x}(0))^T\} = P(0,0) \tag{5.4}$$

and $P(0,0) = P^T(0,0)$, non-negative definite.

E is the mathematical expectation operator and superscript T denotes transpose.

2. The process driving noise $\omega(t)$ is white, Gaussian, with zero mean and known covariance, $\forall t \geq 0$.

$$E(\omega(t)) = 0 \quad \forall t > t_o \tag{5.5}$$

$$\text{cov}[\omega(t),\omega(\tau)] = E\,[\omega(t)\omega^T(\tau)] = Q(t)\delta(t-\tau) \tag{5.6}$$

where $Q(t) = Q^T(t)$, non-negative definite $\forall t \geq 0$.

$\delta\,(t-\tau)$ is the Dirac delta-function

The measurement is also white, Gaussian, with zero mean and known covariance matrix $\forall t$, i.e.

$$E(v(t)) = 0 \quad \forall t \tag{5.7}$$

$$\text{cov}[v(t), v(\tau)] = E[(v(t).v^T(\tau)] = R(t)\delta(t-\tau) \tag{5.8}$$

with $R(t) = R^T(t) \quad \forall t$

Also, the processes $x(0)$, $v(t)$ and $\omega(t)$ are mutually independent, i.e.

$$\text{cov}[x(0), \omega(t)] = 0 \qquad \forall t \tag{5.9}$$

$$\text{cov}[x(0), v(t)] = 0 \qquad \forall t \tag{5.10}$$

$$\text{cov}[\omega(t), v(\tau)] = 0 \qquad 0 \leq \tau \leq t \tag{5.11}$$

The solution to the continuous problem of (5.1) and (5.2) is well known and is given below:

$$x(t) = \Phi(t,t_o)x(t_o) + \int_{t_o}^{t} \Phi(t,\tau)B(\tau)u(\tau)d\tau + \beta(t) \tag{5.12}$$

where $\Phi(t,t_o)$ is the state transition matrix (nxn) given by

$$\Phi(t,t_o) = \exp(A\Delta t) \text{ where } \Delta t = t-t_o. \tag{5.13}$$

$\beta(t)$ is an n-vector, zero mean, Gaussian white noise.

5.1.1.2 Discrete-time Systems

In discrete-time, the Kalman filtering problem is formulated as follows: Consider the linear, discrete-time, multi-variable system defined by

$$x(k+1) = \Phi(k+1,k)x(k) + B(k)u(k) + \omega(k) \tag{5.14}$$

$$z(k+1) = M(k+1)x(k+1) + v(k+1) \tag{5.15}$$

where the symbols carry the same significance as before except that (5.14) and (5.15) are valid only at the instant k of time.

The system is shown in signal flow formation in Figure 5.3.

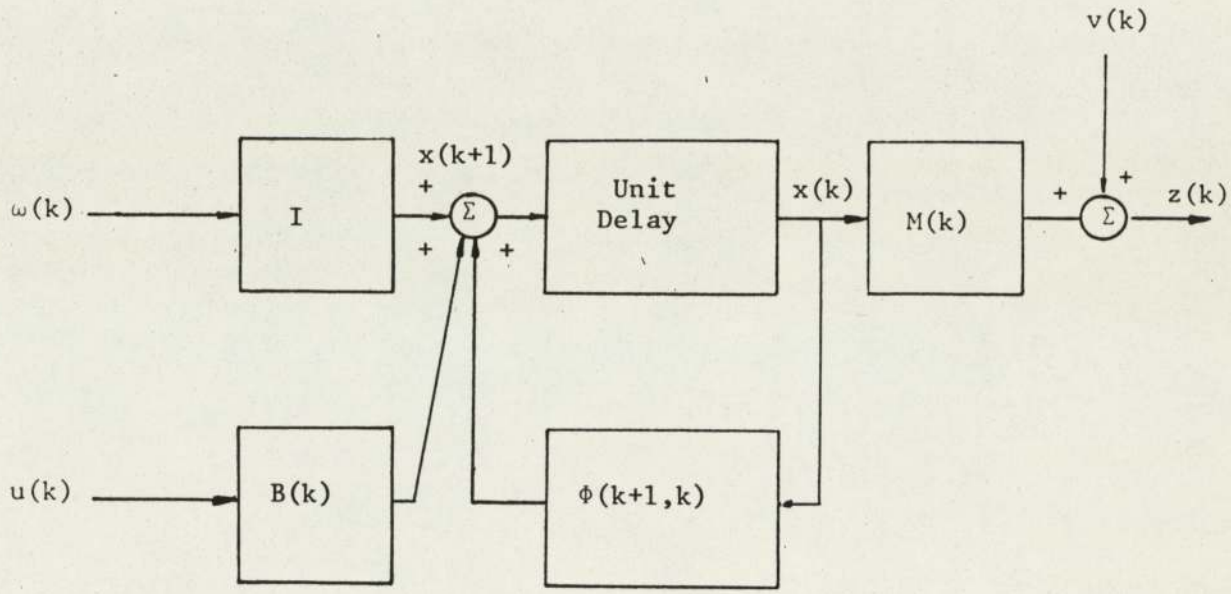


Figure 5.3 Signal Flow diagram for the Discrete-time System Formulation

Furthermore, the noise is zero-mean, Gaussian, white and stationary.

$$E[\omega(k)\omega^T(j)] = Q\delta_{jk} = Q, \quad j=k \qquad (5.16)$$
$$= 0, \quad j\neq k$$
$$E[v(k)v^T(j)] = R\delta_{jk}$$

Also, to solve equation (5.14), an initial estimate of x(0) is required. Since x(0) is now Gaussian, this is given by its mean and covariance respectively:

$$E[x(0)] = \hat{x}(0) \qquad (5.17)$$

$$E[(x(0) - \hat{x}(0))(x(0) - \hat{x}(0))^T] = P(0,0) \qquad (5.18)$$

As before, $\omega(k)$, $v(k)$ and $\hat{x}(0)$ are mutually independent.

The Kalman filtering problem is then to determine the estimate $\hat{x}(k+1,k+1)$ so as to minimise the following quantity J_c.

$$J_c = \frac{1}{2} (x(k,k) - \hat{x}(k,k))P^{-1}(k,k)(x(k,k) - \hat{x}(k,k))$$

$$+ \frac{1}{2} \sum_{j=o}^{k-1} \{[z(j+1) - M(j+1)\hat{x}(j+1,j)]^T R_{j+1}^{-1} [z(j+1)-M(j+1)\hat{x}(j+1,j)]\} \quad (5.19)$$

where superscript -1 denotes matrix inversion, and $\hat{x}(k,j)$ is the estimate obtained at time k given the set of observations through time j.

The problem can be solved in several ways[99-101] yielding the following recursive set of prediction and estimation matrix relations for the case of unit process noise driving matrix and open-loop situation (u=0).

Prediction step:

$$\tilde{x}(k+1,k) = \Phi(k+1,k)\hat{x}(k,k) \quad (5.20)$$

$$P(k+1,k) = Q(k+1,k)P(k,k)\phi^T(k+1,k) + Q \quad (5.21)$$

Estimation step:

$$K(k+1) = P(k+1,k)M^T(k+1)\cdot\{M(k+1)P(k+1,k)M^T(k+1) + R\}^{-1} \quad (5.22)$$

$$\hat{x}(k+1,k+1) = \tilde{x}(k+1,k) + K(k+1)\{z(k+1) - M(k+1)\tilde{x}(k+1,k)\} \quad (5.22a)$$

$$P(k+1,k+1) = \{I - K(k+1)M(k+1)\} P(k+1,k) \quad (5.23)$$

or $P(k+1,k+1) = \{I - K(k+1)M(k+1)\} P(k+1,k).$

$$\{I - K(k+1)M(k+1)\}^T + K(k+1)RK^T(k+1) \quad (5.24)$$

where $R, Q, \hat{x}(0,0)$ and $P(0,0)$ are given and I is the unit matrix (nxn).

Kalman[95] has shown that provided the system remains observable and controllable (complete observability is a sufficient condition for the existence of a steady-state solution and complete controllability will ensure that the steady-state solution is unique), the estimate $\hat{x}(k,k)$ and the filter algorithm are stable for all k.

Clearly, the Kalman filter equations (5.20-5.24) are easily implemented on a digital computer. Since the algorithm is recursive, storage requirements are kept to a minimum. In fact, the storage and computational requirements per iteration have been discussed by Mendel[102].

If the order of the system is high, say, in the order of 50 state variables, so that the on-line storage and computation times become prohibitive, multi-level filters can be designed such as the two-level form due to Noton[103].

In this research work, the prediction step of equation (5.20) is accomplished by the integration of the filter process model thus removing any inaccuracies due to the computation of $\Phi(k+1,k)$ and equation (5.24) is used instead of (5.23) as Aoki[104] has shown via a sensitivity and error analysis of the filter that (5.24) is better conditioned for numerical computation since the right hand side is the sum of two symmetric positive definite matrices. This will ensure the symmetry and positive definiteness of the estimated error covariance matrix $P(k+1,k+1)$ which is the inherent assumption in (5.19). On the other hand, (5.23) is at best the difference of two positive definite matrices.

Unfortunately there is no systematic way of choosing R,Q, and the a priori information $x(0,0)$ and $P(0,0)$, so as to achieve a desired filter performance. Much depends on the user applying judicious estimates after coming to grips with the physics of the problem. However, qualitative and semi-theoretical guidelines are available.

The initial state estimate $\hat{x}(0,0)$ and its error covariance matrix $P(0,0)$ determine the basic speed of response of the filter. The magnitude of the initial state error $(x(0,0) - \hat{x}(0,0))$ will cause an initial error in the covariance matrices which results in an initial error in the gain or weighting matrix $K(k+1)$. The initial error increases the time required for the filter to reach steady state. Similarly if $P(0,0)$ is large, the filter gain $K(k+1)$ will initially be large. This also increases the time to reach steady state as a high gain filter would rely more on current noisy measurement residuals, $z(k+1) - M(k+1)\hat{x}(k+1,k)$.

Equation (5.22) therefore determines the response characteristics of the filter. If the gain is large, the filter relies more on current observations to determine the estimates. When the gain is small, the memory weighting is large i.e. the filter tends to use past information, paying less attention to new measurements.

The error covariance P(k,k) basically describes how uncertainty propagates in time. It gives a measure of the spread of the distribution of $\hat{x}(k,k)$ about the true state x(k,k) which is unknown (except in simulation). Normally, the magnitudes of the elements of P(k,k) will increase initially, often reaching a maximum before stabilising the steady state values.

The assignments of the noise intensity matrices Q and R are also interpreted in a similar way. If the dynamic model of the plant (or certain sections of it) is subjected to modelling errors, then the elements of Q can be increased (or corresponding elements of it) so as to 'mask' out the effects of error propagation in steps (5.21) and (5.22). A large Q causes a high gain and the effect has been mentioned earlier. The values of R are relatively easier to determine as one is usually less uncertain about his sensor equipment. Normally, accurate measuring instruments are reflected in small values of R. This could also mean that we want the filter to rely more on measurements to make up for a process dynamics which is not well understood (as propagated by the integration of the process model). Large values of R therefore tend to make the filter gain small which causes current measurements to be disregarded. Clearly, the assignment of $\hat{x}(0,0)$, P(0,0), Q and R constitutes what is known as, in the parlance of today, 'tuning' the Kalman filter.

5.1.2 Non-linear Systems - the Extended Kalman filter

It must be remembered that the Kalman filter is a linear, (discrete-time) finite-dimensional system. For a non-linear application, the system process (and measurement) models must be linearised about a known and suitable reference trajectory and the deviation variables are then processed through the filter equations (5.20-5.24).

Consider the multivariable system

$$\frac{dx}{dt} = f(x,u,t) + \omega(t) \tag{5.25}$$

$$z = h(x) + v(t) \tag{5.26}$$

where f is a non-linear lumped-parameter model of the plant,

h is a non-linear function of the state vector x(t), and the rest of the symbols have been defined earlier.

By linearising (5.25) and (5.26) about a reference vector x_o (Taylor Series expansion and omitting second and higher order terms), one can obtain the following linear system for the discrete-time, open-loop case:

$$\delta x(k+1,k) = \Phi(k+1,k)_{x_o}\, x(k,k) + \omega(k) \tag{5.27}$$

$$\delta z(k+1) = M(k+1)_{x_o}\, \delta x(k,k) + v(k) \tag{5.28}$$

where $$\Phi(k+1,k)_{x_o} = I + J\Delta t \tag{5.29}$$

is the first order Taylor Series truncation of the matrix exponential expansion of equation (5.13) and J is the Jacobian matrix

$$\left.\frac{\partial f[x(k,k)]}{\partial x(k,k)}\right|_{x_o}$$

$$\Delta t = t_{k+1} - t_k$$

$$M(t+1) = \left.\frac{\partial h[x(k,k)]}{\partial x(k,k)}\right|_{x_o} \tag{5.30}$$

Since at time t_{k+1}, the 'best' knowledge of the plant can only come from the latest estimate $\hat{x}(k,k)$, it is not surprising that $\hat{x}(k,k)$ is often chosen to be the reference trajectory. The implication is that linearisation would have to be done at every sampling instant to obtain new values of Φ and M with obvious overhead on the computation. The filter therefore provides the deviation estimates and the state estimates at time t_{k+1} which is simply given by

$$\hat{x}(k+1,k+1) = \hat{x}(k,k) + \widehat{\delta x}(k+1,k+1) \tag{5.31}$$

Note that the estimates obtained may no longer be optimal. In fact, it can be shown that conditions of optimality is guaranteed only for an infinite dimensional system[101] hence the name sub-optimal filters. Furthermore, divergence and bias effects may effect the numerical stability of the filter algorithm. The main reason for these observations is the violation of the linearity assumption. Although, the sequence of relinearisation about a new estimate as soon as it becomes available does help to prevent large estimation errors from propagating through time, there is still no way of predicting that the linearised process (and measurement) models are valid at each statistical experiment where a statistical experiment is defined as the realisation of one of the many possible states at the sampling time. This is because we do not have specific control over the outcome of a white noise process and therefore we cannot guarantee that the system does not deviate significantly from the region in which the linearisation is more or less valid.

Higher order filters can be formulated by retaining more terms in the Taylor series expansion and/or the time interval can be made smaller to make the system 'more linear' but often these measures do not meet the requirements of on-line applications.

Figure 5.4 shows the flow diagram for the application of a discrete-time, extended Kalman filter on a digital computer.

5.1.3 Chemical Engineering Applications

Relatively speaking, there have been few applications of Kalman filtering theory to industrial process estimation and control. Some of the reasons are coupled to the same inertial effects that the digital computer has to overcome to be acceptable to the chemical process industry. Other factors include the fact that accurate process models are rarely available, and the statistical nature of observation errors and random inputs to large industrial processes is difficult to predict. Even where reliable plant models are known, the system is often non-linear and too complex for on-line repetitive calculations. Nevertheless, one would expect that with more powerful and cheaper microcomputers increasingly available on the market, low-order filters or even single-loop single variable filters may soon find their way in.

One of the earliest reported chemical engineering applications of the Kalman filter was by Noton and Choquette[105,106] in the on-line identification of a reactor train for the Polymer Corporation in Canada. In this pioneering work, the computer was initially used to make off-line computations during open-loop experiments. Later, closed-loop experiments were conducted and significant improvements over manual control were reported.

Other applications soon followed notably those by Coggan and Noton[107] and Sargent and co-workers[109]. In particular, Coggan

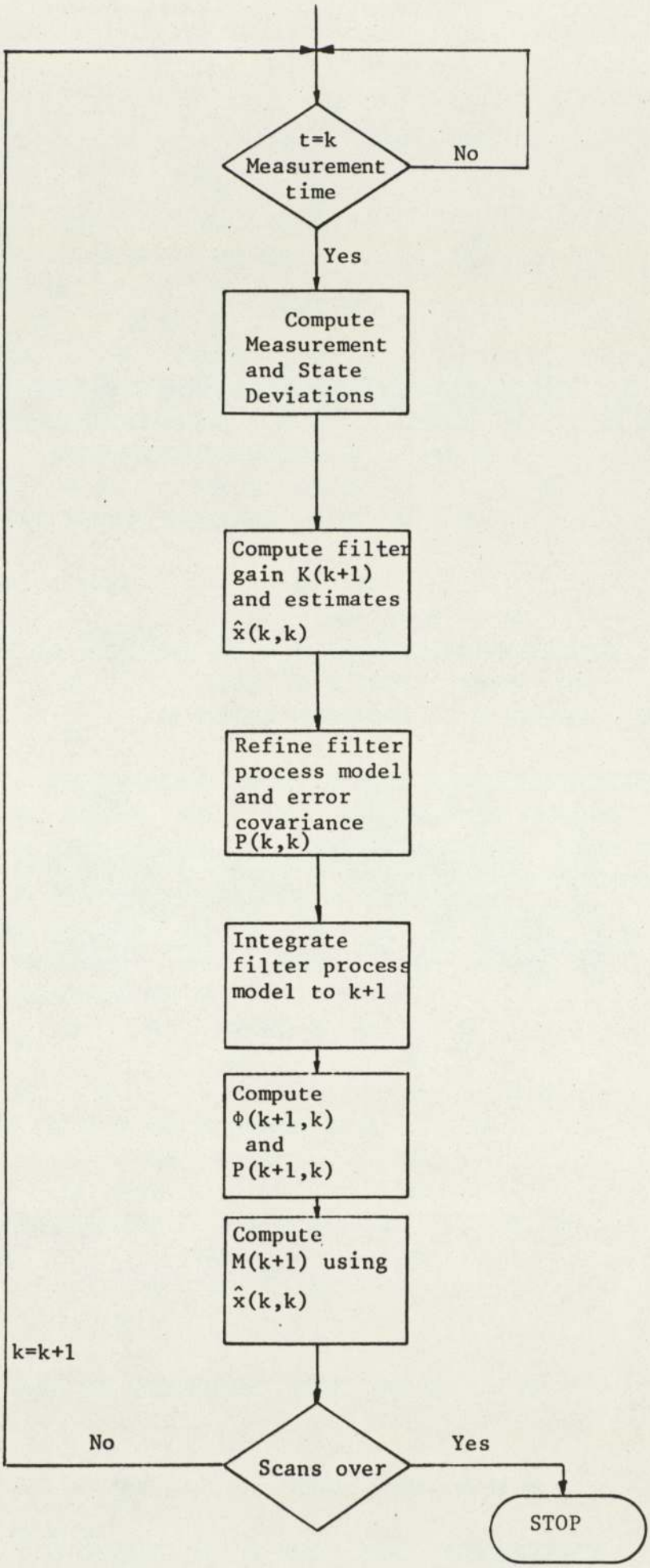


Figure 5.4 Flow diagram of the discrete-time Extended Kalman Filter

Wilson[108] suggested a model reduction technique, prior to estimation, to minimise the number of state variables required to describe a system. They also reported an on-line implementation of a 10^{th} order system on a 4K microcomputer including an investigation of computational times per filter cycle for various dimensions of the state vector[110]. Sargent and Goldman[108] described the application of an extended Kalman filter to state and parameter estimation of a simulated distillation column and a fixed-bed catalytic reactor with superimposed Gaussian and rectangular noise. The work though informative is based on disturbance-free process models i.e. Q = 0. A comprehensive review of other applications in the process industry, especially in open-loop situations, can be found in the work of Webb[111].

However, the most informative compilation of the application of modern control theory and estimation, applied to chemical processes, has been done by Seborg and Fisher[112]. The research mainly conducted on a double-effect evaporator demonstrated that the incorporation of the Kalman filter into a multi-variable control design results in significantly better process control. Payne[113] and Coleby[114] also worked on the double effect evaporator using non-stationary forms of the filter. They implemented their on-line algorithms on a 16-bit H316 minicomputer for open-loop conditions.

More recently, Dahlqvist[115] and Brosilow et al.[116-118] have applied the filtering exercise to distillation columns. Daie[119] has shown, under simulated process conditions, that an estimator-aided feed-forward (EAFF) controller delivers a better performance than a digital PID controller in tackling large, load disturbances that enter a binary distillation process.

An industrial scale application has been described by Wick[120]. It involves the on-line estimation of centre temperature profiles of steel ingots in a soaking pit. An off-line model-fitting was first conducted and the resulting model used in a locally linearised form for on-line state estimation. A MODCOMP II/2 process control computer was used to collect plant measurements connected by a data link to a host computer where the filter algorithm resides.

5.2 Design of a State and Parameter Estimator for a Distillation Column

During the development of the HADIOS and M6800 executives, Daie[119] completed a research effort which included the modelling, and state and parameter estimation of a binary distillation process. The liquid used was a mixture of trichloroethylene and tetrachloroethylene. Although Daie successfully constructed a model for the process which was checked against experimental data, estimation and some work on an estimator-aided feedforward (EAFF) control of the distillation process were done on much more powerful mainframe computers (the ICL 1904S and Manchester University CDC 7600) for simulated conditions. Typically, 22 first-order differential equations were used to describe the process dynamics of the 11-plate column and its filter model. The matrices involved were therefore large (the error covariance matrix P(k,k) would be 26 by 26 if 22 states and 4 parameters were being estimated) and calculations time-consuming. In particular, when Daie tried to use constant plate hold-ups for his filter process model he was forced to raise the number of process measurements to at least six for an operational filter.

It therefore seemed appropriate to extend Daie's work and investigate the feasibility of applying such a filter to an on-line

situation using the linked H316-M6800 twin processor developed in this work. In doing this, several considerations were immediately obvious to the author.

The first of these is available storage space in the H316 minicomputer. Initial calculations showed that since the BASIC interpreter, the HADIOS Executive and some data areas would have to be core-resident, it is not practical to implement any of Daie's original filters. For example, the 26 by 26 P(k,k) matrix alone would need 1352 words (about 2½ sectors) of storage space. The problem dimensionality must therefore be reduced and even at the early stages, it was envisaged that the floppy disk facility would have to be used for program overlay.

The second conerns computational speed. In his simulation work, Daie proposed a sampling interval of 18 seconds. Longer sampling intervals led to the deterioration of filter performance. Even with a reduced dimensionality, the meaningful minimum being the estimation of 11 plate liquid compositions from 3 temperature measurements i.e. for open-loop estimation or stochastic feedback 'optimal' control, it was unlikely that a sampling interval of 18 seconds could be achieved. In earlier on-line Kalman filtering work on a double-effect evaporator, Payne[113] required 120 seconds for an 8^{th}-order system and Webb[111] required 100 seconds for a 7^{th} order system. The bulk of computation time was taken up by the integration of the filter model in the prediction step of the filter algorithm.

The final computational aspect concerns accuracy. The single precision integer and floating-point number formats of the Manchester CDC 7600 and H316 are compared in the tables below.

It is clear that the reduction in going from the two mainframes to the process control minicomputer is very significant. The effect is that a filter that works well in the CDC may not necessarily maintain

Table 5.1 Number representation in the mainframe and minicomputers

	CDC 7600	H316	H316 Double Precision
Integer:			
Number of bits	60	16	32
Approximate range	$\pm 2^{59}$	± 32768	$\pm 2^{30}$
Floating Point:			
Number of bits	60	32	48
Range	10^{-293} to 10^{332}	$\pm 10^{38}$	$\pm 10^{38}$
Accuracy	approx. 14 decimal digits	2^{-23} ($\simeq$ 7 decimal digits)	2^{-39}

its performance in a much smaller minicomputer. Double-precision numbers may be used with the H316, each consuming another 16 bit word and improving the range and accuracy as shown in the last column of Table 5.1, but the overhead in computational time makes it prohibitive given the scale of the on-line application requirements.

The filter algorithm and integration of filter process model would be best done in FORTRAN given the findings of Section 4.9. BASIC statements would be mainly used to initialise the estimator, scaling and conditioning plant outputs, plotting in conjunction with the Tektronix Graphics package and other I/O tasks. These would save some previous computation time while maintaining a reasonable level of user interaction with his on-line program. In order to grasp the magnitude of the problems involved, simulation experiments would be necessary and these are detailed in Chapter 6. The rest of the discussion in this chapter is directed to describing the experimental background to the filter exercise, followed by the theoretical development in designing a suitably-sized

Kalman state, or state and parameter, estimator for the binary distillation process.

5.2.1 Apparatus and Process Description

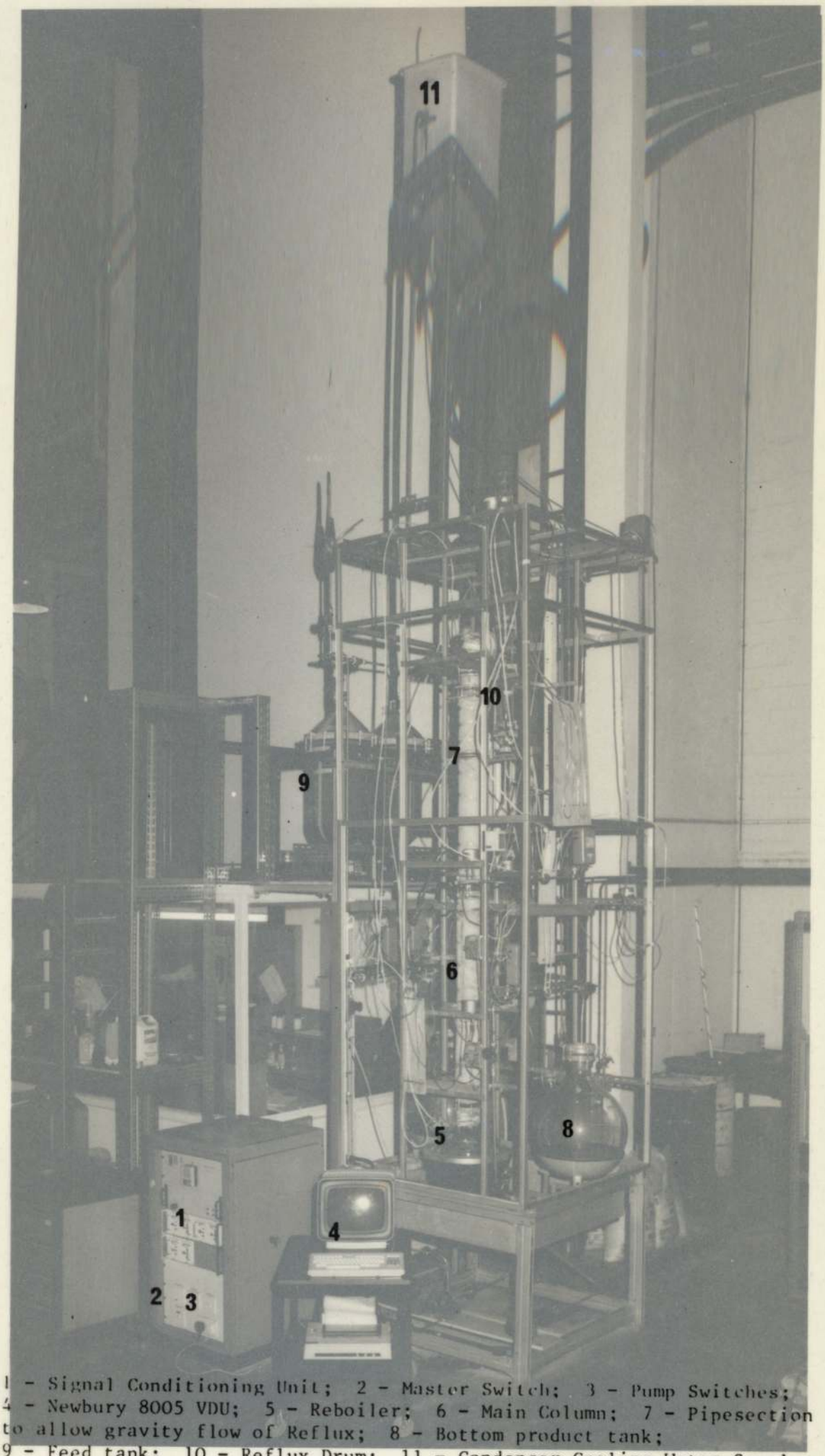
5.2.1.1 Introduction

The distillation column and its accessories were donated to the Department by the I.B.M. (U.K.) Ltd., together with some parts of its instrumentation and data conditioning equipment. A schematic diagram of the column is shown in Figure 5.5 and a laboratory view is shown in Plate 5.1. The construction of the column has been described in detail by Daie. Only the main features will be mentioned here.

At present, the column hardware includes:

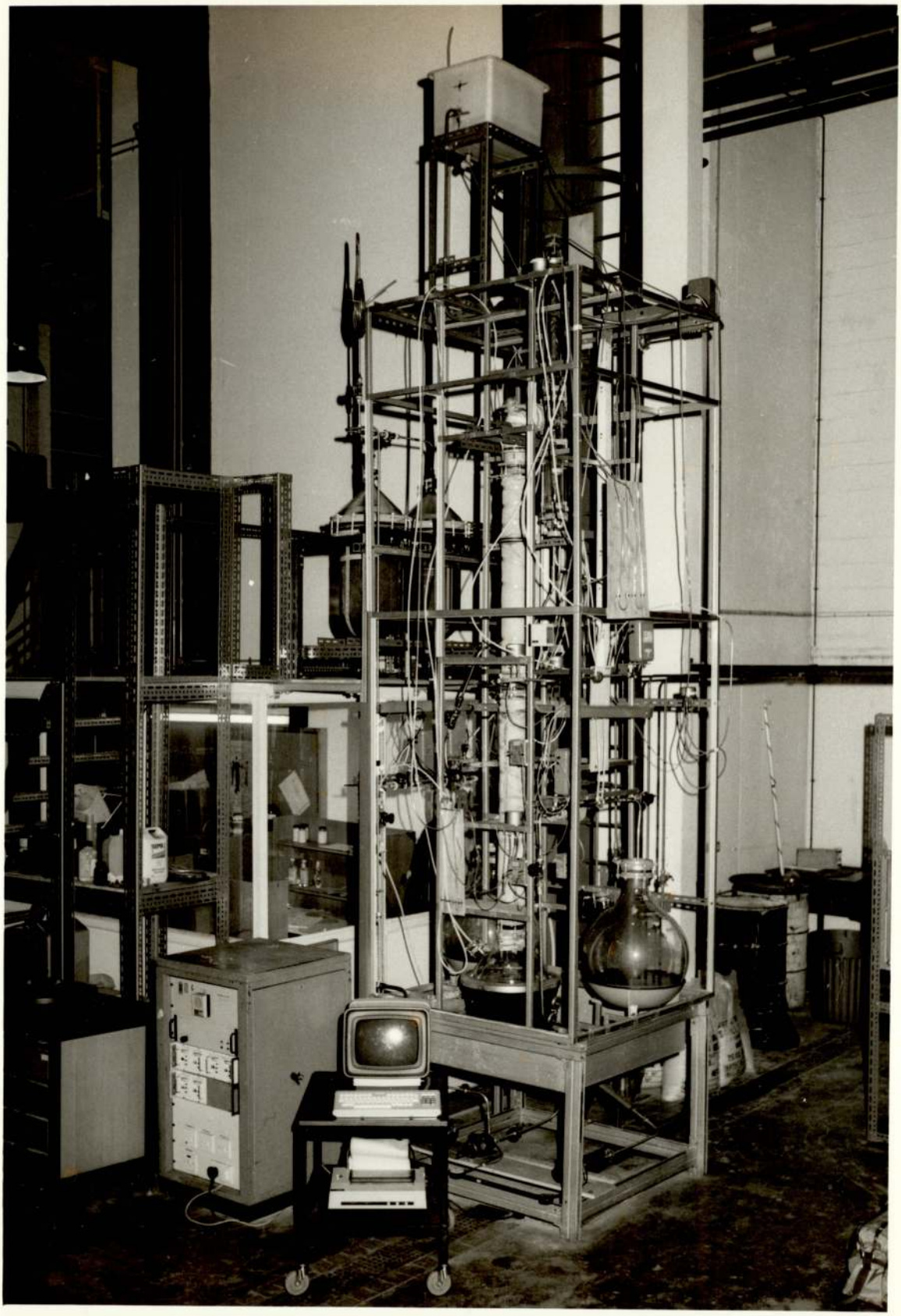
1. an enriching and a stripping section with six and four 3-inch diameter sieve trays respectively and one temperature measurement well per tray.
2. a 3-inch diameter standard glass total condenser plus the reflux drum.
3. a 2.4 kw, double-circuit Isomantle electric heater.
4. a 1-metre long, 3-inch O.D. glass pipe section, above the enriching section.
5. two cylindrical feed tanks.
6. two spherical product tanks - for top and bottom product respectively.
7. three stainless steel centrifugal pumps.

The column was designed by IBM to be an all-purpose distillation column capable of handling corrosive liquids. For this reason, the materials used for its construction and piping were limited to stainless steel, glass and P.T.F.E. All the pipelines are made of ½-inch O.D. heavy gauge stainless steel. However, the vapour line connecting the



1 - Signal Conditioning Unit; 2 - Master Switch; 3 - Pump Switches; 4 - Newbury 8005 VDU; 5 - Reboiler; 6 - Main Column; 7 - Pipesection to allow gravity flow of Reflux; 8 - Bottom product tank; 9 - Feed tank; 10 - Reflux Drum; 11 - Condenser Cooling Water Supply

Plate 5.1 A Laboratory View of the IBM Distillation Column



column to the condenser is made of a 3-inch diameter glass section.

The main characteristics for a typical tray are listed in Table 5.2. A small heat exchanger is installed in the bottom line to prevent boiling liquid reaching the pump thus avoiding possible cavitation problems. The entire column is also lagged with ½-inch standard fibre glass lagging.

Table 5.2 Main characteristics of a typical tray

Tray diameter	0.0762 m.
Downcomer diameter	0.0105 m.
Tray thickness	0.0002 m.
Diameter of the Perforations	0.00011 m.
Number of Perforations	145
Weir Height	.0003 m.

5.2.1.2 The Motorised Valves

As indicated by Figure 5.5, five motorised values are provided on the column. Each value can be driven manually from the remote signal condition unit or from the computer through the use of CALL(3,N,U) (or SUB3(N,U) in the M6800). The range of the valve input signal is 0 to 10 volts from fully closed to fully open stem positions. Each valve can be calibrated in situ by outputting a voltage signal from the computer and measuring the volumetric flowrate by the stopwatch and bucket method. The valve characteristics can therefore be expressed into the form

output voltage = f (volumetric flowrate)

or volumetric flowrate = g (output voltage)

where f and g are generally non-linear functions. The polynomial fit for each valve is given in Appendix 5. Valve 4, the condenser cooling water valve, was not calibrated as it was always set to fully open during experimental runs.

5.2.1.3 The Reboiler

The reboiler is essentially a 2.4 kw isomantle electric heater which can be driven by a maximum of 10 volt input from the computer or manually from the remote signal conditioning unit. The reboiler characteristics have been studied by Daie and may be used for the purposes of this research. These are as follows:

$$Q = .3312 - 0.17724V + .0792V^2 - .00524V^3 \qquad (5.31)$$

or alternatively

$$V = 3.62 - 3.86Q + 6.59Q^2 \qquad (5.32)$$

where Q is the heat transferred kJ/s

V is the input voltage, 0 to 10 volts.

These correlations were found for a 12 kg reboiler liquid contents. Clearly, the heat transfer area (hence Q) changes with various amounts of reboiler liquid, hence necessitating a correction factor. The actual heat transferred Q_A is then given by

$$Q_A = \frac{1}{R} Q \qquad (5.33)$$

where R = h (mass of liquid in the reboiler, M kg)

$$= -0.9183 + 0.485M - 0.0442\ M^2 + .00191\ M^3$$

5.2.1.4 Temperature measurements

Altogether five thermocouples were used to monitor the feed, reflux and three tray temperatures. The feed and reflux stream thermocouples are of the Lee-Dickens type and accurate to $\pm$ 1°C. They derive their power supply from the mains. On the other hand, the three

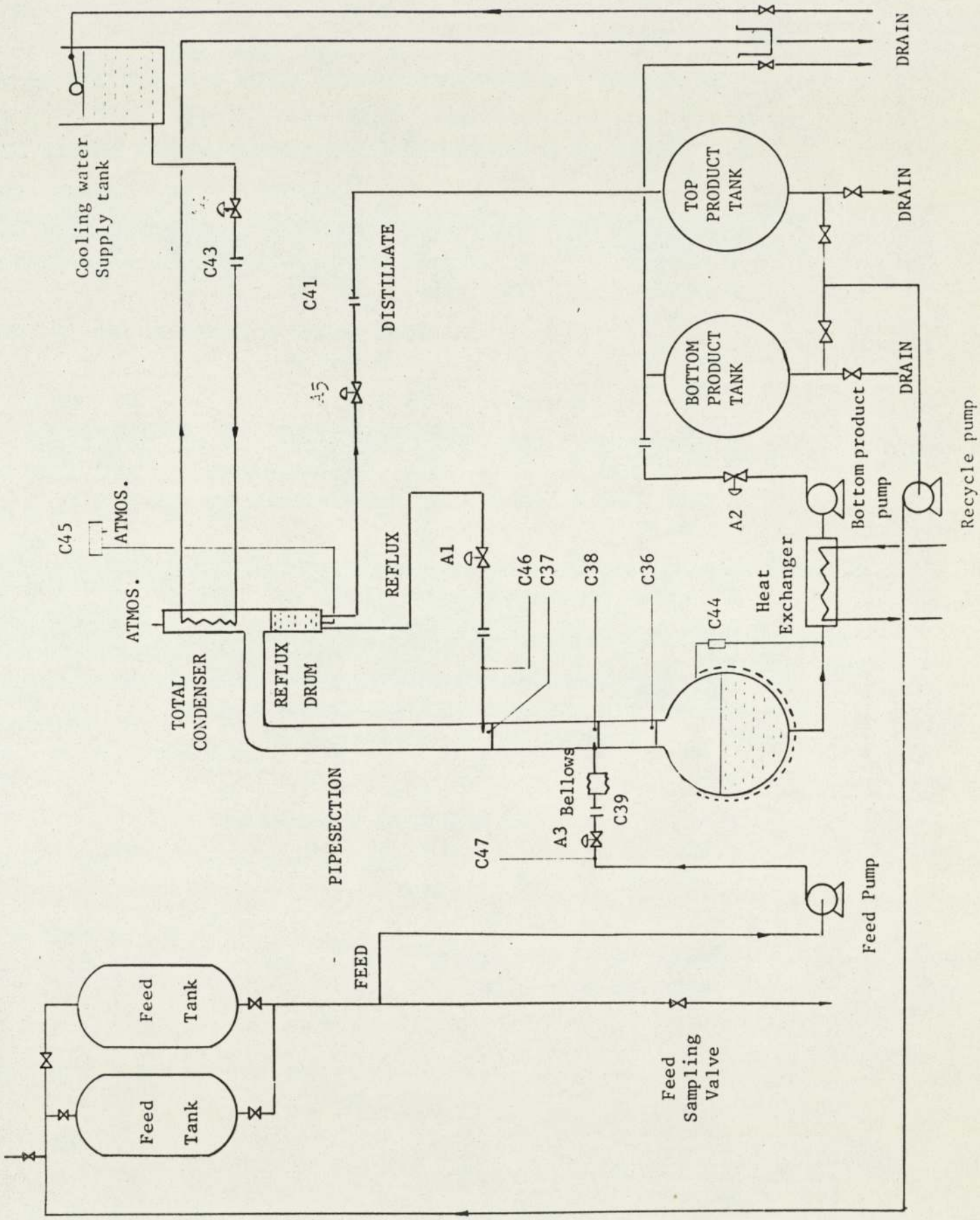


Figure 5.5. A Schematic diagram of the IBM Distillation Column

tray thermocouples are accurate to ± .1°C and battery powered. The characteristics of the three tray thermocouples, normally the top, feed and bottom trays, and the feed and reflux streams are given in Appendix 5.

5.2.1.5 Continuous Level Measurements

The reflux drum and reboiler levels need to be monitored continuously for open-loop and control experiments. This is done via two air differential pressure transducers, isolated from the liquid by an air-lock. Transducer characteristics of the form

Hold up (volume or mass) = z (output voltage)

were found to be linear and they are given in Appendix 5. The correlations are:

Reboiler: $M_B = -60.0 + 40.1V$ (5.34)

Reflux Drum: $M_{RD} = -.05864 + .541V$ (5.35)

5.2.1.6 Flow Measurements

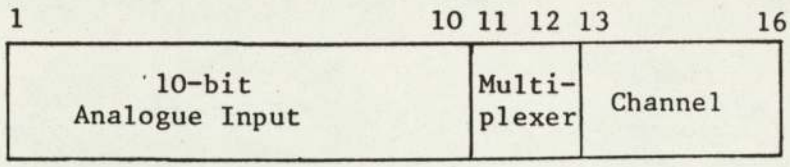
In the original column set-up, four pressure differential transducers were provided for the measurement of feed, bottoms, reflux and distillate flowrates. The flowrate could be correlated to the transducer output (0 to 5 volts) and a correlation obtained in a way similar to previous transducer calibrations. However, the use of these transducers were grossly inaccurate and unreliable and they were therefore not used for flow measurements.

The high inaccuracy (up to ± 50 cm^3/min for a 150 cc/min flow) is basically due to the low flowrates available because of the limitations on the reboiler heater capacity. Flowrates are computed via valve opening characteristics instead.

5.2.1.7 Remote Signal Conditioning Unit

The H316-HADIOS hardware is connected to the column via a remote signal conditioning interface as shown schematically in Chapter 3, Figure 3.2. A total of 12 analogue inputs and 6 digital outputs are used as it stands at present. The channel numbers assinged are given in Table 5.3 below.

The remote signal conditioning unit prepares transducer electrical outputs (usually in the order of millivolts) for H316-HADIOS compatibility. Thus, all analogue outputs are converted into the range 0 to 5 Volts D.C. As the ADC set-up word (shown below) requires 2 bits to differentiate any one of four multiplexers (at present only three are used) and a further 4 bits to decode any one of 16 channels per multiplexer,



only a 10-bit ADC can be used to represent an analogue input giving an accuracy of about 0.1%. Also contained in the unit are:

1. line drivers for the transmission of digital signals.
2. power supplies for the transducers, motorised valves and reboiler.
3. the analogue output device.
4. the manual/automatic switches for the values and the reboiler required to start up the column.

The signal conditioning unit was built by Departmental electricians and is situated adjacent to the plant for convenience as well as to minimise the effects of electrical noise during the

Table 5.3 Analogue Inputs and Digital Output Channel Numbers

Analogue Inputs	Location and Function
C36	Lowermost tray, thermocouple T_{10}
C37	Topmost tray, thermocouple T_1
C38	Feed tray, thermocouple T_7
C39	Feed flow, pressure transducer (PT)
C40	Bottoms flow, PT
C41	Distillate flow, PT
C42	Reflux flow, PT
C43	Condenser cooling water, PT
C44	Reboiler Level Indicator, PT
C45	Reflux Drum Level Indicator, PT
C46	Reflux stream, thermocouple
C47	Feed stream, thermocouple
Digital Output Channel	
A1	Reflux stream, motorised valve (MV)
A2	Bottoms, MV
A3	Feed stream, MV
A4	Condenser cooling water, MV
A5	Distillate, MV
A6	Reboiler Heat

transmission of signals to the computer.

5.2.1.8 Process Operation

A typical process operation includes start-up, manual/computer control and shut-down. The recommended procedure is as follows[119].

1. Turn on the mains power supply by switching on the MASTER switch of the remote signal conditioning unit.
2. Put all valves and reboiler heater into manual mode.
3. Switch on the H316 computer, the HADIOS power supply (switch 1 in Plate 3.1) and the digital output line driver power supply (switch 2 in Plate 3.1).
4. Load the HADIOS Executive Package Rev 03.
5. Empty the reboiler and reflux drum.
6. Recycle the feed stock a couple of time to ensure good mixing of components.
7. Disconnect all transducer connections at the pressure transducers to allow air into the transducers.
8. Manually open and close the valves to ensure that they are in working condition.
9. Fill the reboiler to the required amount, connecting the transducer connections as soon as the dead zone is covered (about the first 200 cc.).
10. Connect all transducer connections making sure that all U-tubes and air-locks are intact.
11. Turn on the cooling water for the condenser and the bottoms product line heat exchanger.
12. Switch the reboiler heater on and run the column at total reflux for at least 20 minutes to ensure that the trays are filled (the reflux valve fully opened).

13. Switch the feed pump on and open the feed valve gradually to allow the feed into the column.
14. Set the reflux valve to the required opening.
15. Switch the thermocouples on and switch to automatic mode of operation to let the computer take over.
16. Allow about 30 minutes for the column to reach steady state after which disturbances can be introduced and experimentation can begin.

Shut-down

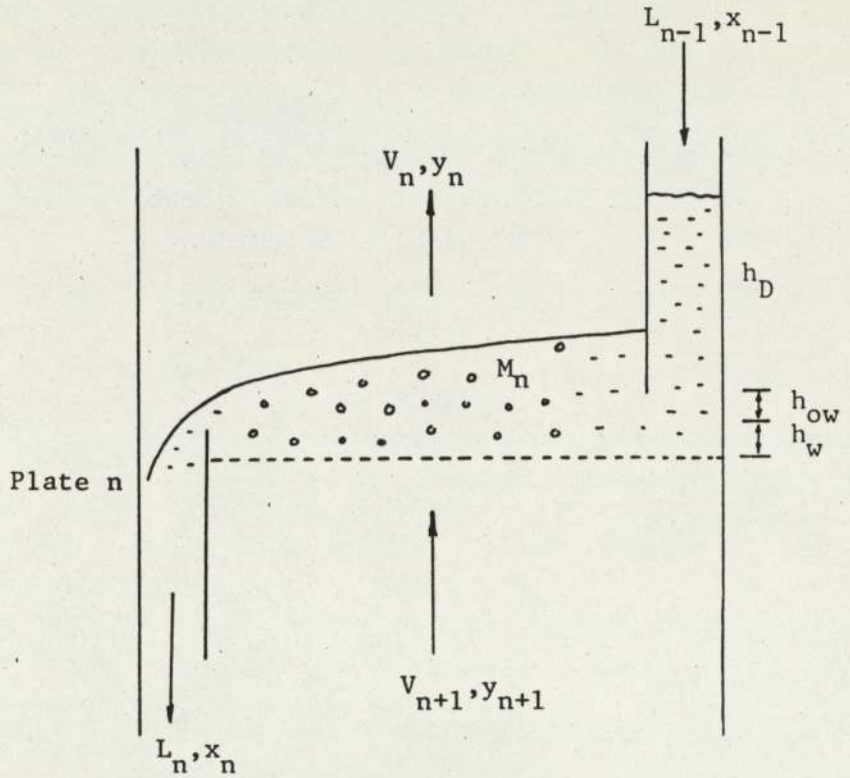
1. Switch the heater, all pumps, manual and motorised valves off, in that order.
2. Switch the thermocouples off.
3. Switch off the HADIOS, computer system and associated peripherals.
4. Switch the MASTER-switch off.
5. Allow the column to cool down for at least 30 minutes and turn off the cooling water taps.

5.2.2 Formulation of Process Models

Two mathematical models of the binary distillation process are formulated - one to simulate the actual column operation and the other to be used as the filter process model. Both models are based on the work of Daie with some modifications to adapt them to the storage and other computing requirements of the H316 minicomputer.

Essentially, distillation behaviour can be approximated by modelling a typical tray with a defined vapour-liquid equilibrium and stepping this up from a single tray to a column with a number of similar trays. The models of the reboiler and condenser can then be added to complete the mathematical description of the system.

A typical tray is shown in Figure 5.6 with the following assumptions made:



h_D = liquid height in the downcomer

h_{ow} = liquid height over the weir

h_w = height of the weir

M_n = plate liquid hold-up

L_n, V_n = liquid and vapour flow rates respectively

x_n, y_n = liquid and vapour compositions of the MVC

Figure 5.6 Schematic Diagram of a Typical Tray

1. Liquid on the tray is perfectly mixed and incompressible.
2. The time constants for the fluid and thermal transfers are negligible compared with that for the mass transfer.

3. Vapour hold-up is negligible compared with liquid hold-up.

4. The column is adiabatic.

5. Vapour and liquid are in thermal equilibrium (but not in phase equilibrium).

Total continuity: $$\frac{d}{dt} M_n = (L_{n-1}-L_n) + (V_{n+1}-V_n) \quad (5.36)$$

Component continuity for the more volatile component (MVC):

$$M_n \frac{d}{dt} x_n = L_{n-1}(x_{n-1}-x_n) + V_n(x_n-y_n) + V_{n+1}(y_{n+1}-x_n) \quad (5.37)$$

Energy equation: $$V_n = (L_{n-1}h_{n-1} + V_{n+1}H_{n+1} - L_n h_n)/H_n \quad (5.38)$$

Fluid Dynamics: $$L_n = (M_n - A)/B \quad (5.39)$$

where A,B are Brambilla constants[121]

Actual vapour composition for MVC: $$y_n = y_{n+1} + E_n(y_n^* - y_{n+1}) \quad (5.40)$$

where y_n^* is the vapour composition is equilibrium with x_n (through bubble-point calculations)

E_n is a Murphree vapour-phase efficiency term for tray n.

1. Model of the Feed Tray

Total continuity: $$\frac{d}{dt} M_n = (L_{n-1}-L_n) + (V_{n+1} - V_n) + F \quad (5.41)$$

where F is the feed rate (mol/hr.)

Component continuity for the MVC:

$$M_n \frac{dx_n}{dt} = L_{n-1}(x_{n-1} - x_n) + V_n(x_n - y_n) + V_{n+1}(y_{n+1} - x_n) + F(x_F - x_n) \quad (5.42)$$

where x_F is the feed composition (mol fraction of MVC)

Energy Balance equation: $V_n = (L_{n-1}h_{n-1} + V_{n+1}H_{n+1} - L_n h_n + Q_F)/H_n$ (5.43)

where $Q_F = F\Sigma q_j x_j$ and $q_j = C_{p,j}(T_F - T_o)$

T_F = feed temperature, T_o = reference temperature (273K)

Fluid Dynamics: $L_n = (M_n - A)/B$

Actual vapour composition for MVC:

$$y_n = y_{n+1} + E_n(y_n^* - y_{n+1}) \quad (5.44)$$

5.2.2.1 The Actual Process Model - Model I

2. Model of a Typical Tray

With reference to Figure 5.6, the following energy balance can be written for a typical tray n:

$$\frac{d}{dt}(M_n U_n) = (L_{n-1}h_{n-1} + V_{n+1}H_{n+1}) - (L_n h_n + V_n H_n) \quad (5.45)$$

where
- U = internal energy term (kJ/mol)
- M_n = molar liquid hold-up
- L_n = liquid rate from the n^{th} tray (mol/hr.)
- h_n = liquid enthalpy (kJ/mol)
- V_n = vapour rate from the n^{th} tray (mol/hr.)
- H_n = vapour enthalpy (kJ/mol).

The accumulation term, $\frac{d}{dt}(M_n U_n)$, is negligible compared with the heat fluxes through the tray giving the following equation for the calculation of vapour rate leaving each tray.

$$V_n = \frac{L_{n-1}h_{n-1} + V_{n+1}H_{n+1} - L_nH_n}{H_n} \qquad (5.46)$$

where $h_n = \Sigma\, h_j x_j$, x_j is liquid mole fraction of component j

$H_n = \Sigma\, H_j y_j$, y_j is vapour mole fraction of component j

$$h_j = \int_{T_o}^{T} C_{p,j}\, dT$$

$$H_j = h_j + \lambda_j$$

T_o = a reference temperature (273K)

λ_j = latent heat of vaporisation of the j^{-th} component

$C_{p,j}$ = heat capacity equation for component j (function of temperature, T)

3. Model of the Reboiler

Total continuity: $$\frac{d}{dt} M_n = (L_{n-1} - L_o) - V_n \qquad (5.47)$$

where L_o = bottoms rate (mol/hr.)

Component continuity for the MVC: $$M_n \frac{d}{dt} x_n = L_{n-1}(x_{n-1} - x_n) + V_n(x_n - y_n) \qquad (5.48)$$

Energy equation: $$V_n = (L_{n-1}h_{n-1} - L_o h_o + Q)/H_n \qquad (5.49)$$

where Q = heat duty (kJ/hr.)

Fluid dynamics: $$L_n = (M_n - A)/B \qquad (5.50)$$

Actual vapour composition: $$y_n = x_n + E_n (y_n^* - x_n) \qquad (5.51)$$

4. Model of the Condenser and Reflux Drum

Total continuity: $$\frac{d}{dt} M_D = V_1 - L_R - D \quad (5.52)$$

where L_R = the reflux rate (mol/hr.)

D = the distillate rate (mol/hr.)

Component continuity: $$M_D \frac{d}{dt} x_D = V_1(y_1 - x_D) \quad (5.53)$$

where M_D = reflux drum molar hold-up

x_D = the reflux composition for MVC

y_1 = topmost plate vapour composition

5.2.2.2 The Filter Process Model - Model II

In the work of Daie, it was observed that the percentage variations of plate liquid hold-ups are small when compared with corresponding variations in liquid or vapour compositions. In view of this and the limitations imposed on the H316 minicomputer, the filter process model is a much simplified version of Model I with the following assumptions:

(i) Constant molar hold-ups.

(ii) Equimolal overflow: $V_{n-1} = V_n = V_{n+1} = V$.

(iii) the feed and reflux streams enter the column as saturated liquids at their bubble points.

(iv) constant column pressure.

(v) negligible time-lags in the reboiler and condenser responses.

1. Model of a Typical Tray

Total continuity: $$L_{n-1} = L_n = L \quad (5.54)$$

where L = reflux rate, L_R, is in the enriching section

= L_R + F is in the stripping section.

Component continuity for the MVC:
$$M_n \frac{dx_n}{dt} = L_{n-1}(x_{n-1}-x_n) + V(y_{n+1}-y_n) \quad (5.55)$$

Actual vapour composition:
$$y_n = y_{n+1} + E_n(y_n^* - y_{n+1}) \quad (5.56)$$

2. Model of the Feed Tray

Total continuity:
$$L_n = L_{n-1} + F \quad (5.57)$$

Component continuity for the MVC:
$$M_n \frac{dx_n}{dt} = L_{n-1}x_{n-1} - L_n x_n + V(y_{n+1}-y_n) + Fx_F \quad (5.58)$$

Actual vapour composition:
$$y_n = y_{n+1} + E_n(y_n^* - y_{n+1}) \quad (5.59)$$

3. Model of the Reboiler

Total continuity:
$$L_o = L_{n-1} - V \quad (5.60)$$

Component continuity:
$$M_n \frac{dx_n}{dt} = L_{n-1}x_{n-1} - L_n x_n - Vy_n \quad (5.61)$$

Energy equation:
$$V = Q/\lambda \quad (5.62)$$

where λ = average latent heat of vaporization for the mixture

Actual vapour composition:
$$y_n = y_n^* E_n \quad (5.63)$$

4. Model of the Condenser and Reflux Drum

Total continuity:
$$L_R = V - D \quad (5.64)$$

Component continuity for the MVC:

$$M_D \frac{dx_D}{dt} = V(y_1 - x_D) \tag{5.65}$$

5.2.2.3 Tray Efficiencies

It was also observed in Daie's work that the calculations of Murphree plate efficiencies based upon the A.I.Chem.E. method[122] were not only time-consuming but that the resulting efficiencies did not vary much over the simulated process operating conditions and disturbances. In this work, the tray efficiencies for the two models are those obtained off-line for a typical operating condition. This will be discussed again in Chapter 6.

5.2.3 The Kalman Filter Algorithm

For the purposes of this research work, the discrete-time Kalman Filter algorithm can be restated as a set of prediction and estimation steps as follows:

Prediction:

$$\tilde{x}(k+1,k) = \hat{x}(k,k) + \int_{t_k}^{t_{k+1}} f(\hat{x}(k,k))dt \tag{5.66}$$

$$P(k+1,k) = \phi(k+1,k)P(k,k)\phi^T(k+1,k) + Q \tag{5.67}$$

Estimation:

$$K(k+1) = P(k+1,k)M^T(k+1).\{M(k+1)P(k+1,k)M^T(k+1)+R\}^{-1} \tag{5.68}$$

$$\hat{x}(k+1,k+1) = \tilde{x}(k+1,k) + K(k+1).\{z(k+1) - M(k+1)\tilde{x}(k+1,k)\} \tag{5.69}$$

$$P(k+1,k+1) = \{I-K(k+1)M(k+1)\}P(k+1,k)\{I-K(k+1)M(k+1)\}^T + K(k+1)RK(k+1)^T \tag{5.70}$$

To start the filtering computation, R,Q together with $\hat{x}(0,0)$ and $P(0,0)$ must be known at time zero (k=0).

The distillation models described earlier are clearly non-linear and the extended Kalman filter applies. The approach taken is to process the state and measurement deviations (from a known reference trajectory) through the linear filter algorithm and convert them back into actual estimates. This procedure is described below where the reference trajectories are derived from the previous estimates.

1. Set k=0: Estimate $\hat{x}(k,k)$ and $P(k,k)$ and derive $\hat{z}(k)$ from the filter process model.
2. Compute ϕ based on $\hat{x}(k)$.
3. Compute predicted state vector $\tilde{x}(k+1,k)$ at next sampling instant by integration of the non-linear differential equations (filter process model) starting from $\hat{x}(k,k)$.
4. Compute the predicted state deviations and the measurement deviations as new measurements z(k+1) become available:

$$\tilde{\delta x}(k+1,1) = \tilde{x}(k+1,k) - \hat{x}(k,k) \tag{5.71}$$

$$\delta z(k+1,k) = z(k+1) - \hat{z}(k) \tag{5.72}$$

5. Process the deviations through the linear filter to obtain $x(k+1,k+1)$.

$$\hat{\delta x}(k+1,k+1) = \tilde{\delta x}(k+1,k) + K(k+1).\{\delta_z(k+1) - M(k+1)\ \tilde{\delta x}(k+1,k)\} \tag{5.73}$$

6. Compute $\hat{x}(k+1,k+1)$ using $\hat{x}(k+1,k+1) = \hat{x}(k,k) + \hat{\delta x}(k+1,k+1)$ (5.74)
7. Compute $\hat{z}(k+1)$ using the filter model. Note, $\hat{x}(k+1,k+1)$, $\hat{z}(k+1)$ now become the new reference trajectories.
8. Set k=k+1, jump to step 2 and continue.

5.2.4 The State Vector x_{nx1}

A typical problem was to estimate 11 plate liquid compositions and 4 process variables regarded as parameters. The resulting state vector is therefore 15x1:

$$x^T = (x_1, \ldots\ldots\ldots x_{11}, F, x_F, V, L_R) \tag{5.75}$$

where F, x_F, V and L_R are feed rate, feed composition, boil-up and reflux rates respectively.

5.2.5 The State Transition Matrix Φ_{nxn}

The state transition matrix ϕ can be calculated given the latest estimates $\hat{x}(k,k)$ as follows:

From equation (5.29),

$$\phi(k+1,k) = I + J(k)\ \delta t \tag{5.76}$$

where J(k) is the Jacobian matrix $\left.\frac{\partial f(x_j)}{\partial x_j}\right|_k$ is computed at the latest estimated state.

$$\frac{\partial f(x_j)}{\partial x_j} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix} \tag{5.77}$$

From the filter process model, and noting the equilibrium relationship $y_i = K_i x_i$ and that the feed plate is plate 7:

1. $$f_1 = \{L_R(x_D - x_1) + V(y_2 - y_1)\} / M_1 \tag{5.78}$$

$$\frac{\partial f_1}{\partial x_1} = \{K_1(L_R - V) - L_R\}/M_1 \tag{5.79}$$

$$\frac{\partial f_1}{\partial x_2} = VK_2/M_1 \tag{5.80}$$

$$\frac{\partial f_1}{\partial x_j} = 0 \qquad \forall_j = 3,11 \tag{5.81}$$

In addition, if F, x_F, V and L_R are treated as pseudo-state variables or parameters and being estimated:

$$\frac{\partial f_1}{\partial F} = 0 \tag{5.82}$$

$$\frac{\partial f_1}{\partial x_F} = 0 \tag{5.83}$$

$$\frac{\partial f_1}{\partial V} = (y_2 - y_1)/M_1 \tag{5.84}$$

$$\frac{\partial f_1}{\partial L_R} = (y_1 - x_1)/M_1 \tag{5.85}$$

2. $$f_2 = \{L_R(x_1 - x_2) + V(y_3 - y_2)\}/M_2 \tag{5.86}$$

$$\frac{\partial f_2}{\partial x_1} = L_1/M_2 \tag{5.87}$$

$$\frac{\partial f_2}{\partial x_2} = -(L_R + VK_2)/M_2 \tag{5.88}$$

$$\frac{\partial f_2}{\partial x_3} = VK_3/M_2 \tag{5.89}$$

$$\frac{\partial f_2}{\partial x_j} = 0 \quad \forall j = 4,11 \tag{5.90}$$

$$\frac{\partial f_2}{\partial F} = 0 \tag{5.91}$$

$$\frac{\partial f_2}{\partial x_F} = 0 \tag{5.92}$$

$$\frac{\partial f_2}{\partial V} = (y_3-y_2)/M_2 \tag{5.93}$$

$$\frac{\partial f_2}{\partial L_R} = (x_1-x_2)/M_2 \tag{5.94}$$

3. $$f_3 = \{L_R(x_2-x_3) + V(y_4-y_3)\}/M_3 \tag{5.95}$$

$$\frac{\partial f_3}{\partial x_1} = 0 \tag{5.96}$$

$$\frac{\partial f_3}{\partial x_2} = L_R/M_3 \tag{5.97}$$

$$\frac{\partial f_3}{\partial x_3} = -(L_R + VK_3)/M_3 \tag{5.98}$$

$$\frac{\partial f_3}{\partial x_4} = VK_4/M_3 \tag{5.99}$$

$$\frac{\partial f_3}{\partial x_j} = 0 \quad \forall j = 5,11 \tag{5.100}$$

$$\frac{\partial f_3}{\partial F} = 0 \tag{5.101}$$

$$\frac{\partial f_3}{\partial x_F} = 0 \tag{5.102}$$

$$\frac{\partial f_3}{\partial V} = (y_4 - y_3)/M_3 \tag{5.103}$$

$$\frac{\partial f_3}{\partial L_R} = (x_2 - x_3)/M_3 \tag{5.104}$$

4. $$f_4 = \{L_R(x_3 - x_4) + V(y_5 - y_4)\}/M_4 \tag{5.105}$$

$$\frac{\partial f_4}{\partial x_j} = 0 \quad j = 1,2 \tag{5.106}$$

$$\frac{\partial f_4}{\partial x_3} = L_3/M_4 \tag{5.107}$$

$$\frac{\partial f_4}{\partial x_4} = -(L_R + VK_4)/M_4 \tag{5.108}$$

$$\frac{\partial f_4}{\partial x_5} = VK_5/M_4 \tag{5.109}$$

$$\frac{\partial f_4}{\partial x_j} = 0 \quad j = 6,11 \tag{5.110}$$

$$\frac{\partial f_4}{\partial F} = 0 \tag{5.111}$$

$$\frac{\partial f_4}{\partial x_F} = 0 \tag{5.112}$$

$$\frac{\partial f_4}{\partial V} = (y_5 - y_4)/M_4 \tag{5.113}$$

$$\frac{\partial f_4}{\partial L_R} = (x_3 - x_4)/M_4 \tag{5.114}$$

5. $$f_5 = \{L_R(x_4-x_5) + V(y_6-y_5)\}/M_5 \tag{5.115}$$

$$\frac{\partial f_5}{\partial x_j} = 0 \quad \forall j = 1,3 \tag{5.116}$$

$$\frac{\partial f_5}{\partial x_4} = L_R/M_5 \tag{5.117}$$

$$\frac{\partial f_5}{\partial x_5} = -(L_R + VK_5)/M_5 \tag{5.118}$$

$$\frac{\partial f_5}{\partial x_6} = VK_6/M_5 \tag{5.119}$$

$$\frac{\partial f_5}{\partial x_j} = 0 \quad \forall j = 7,11 \tag{5.120}$$

$$\frac{\partial f_5}{\partial F} = 0 \tag{5.121}$$

$$\frac{\partial f_5}{\partial x_F} = 0 \tag{5.122}$$

$$\frac{\partial f_5}{\partial V} = (y_6-y_5)/M_5 \tag{5.123}$$

$$\frac{\partial f_5}{\partial L_R} = (x_4 - x_5)/M_5 \tag{5.124}$$

6. $$f_6 = \{L_R\ (x_5 - x_6) + V(y_7 - y_6)\}/M_6 \tag{5.125}$$

$$\frac{\partial f_6}{\partial x_j} = 0 \qquad \forall j = 1,4 \tag{5.126}$$

$$\frac{\partial f_6}{\partial x_5} = L_R/M_6 \tag{5.127}$$

$$\frac{\partial f_6}{\partial x_6} = -(L_R + VK_6)/M_6 \tag{5.128}$$

$$\frac{\partial f_6}{\partial x_7} = VK_7/M_6 \tag{5.129}$$

$$\frac{\partial f_6}{\partial x_j} = 0 \quad \forall j = 8,11 \tag{5.130}$$

$$\frac{\partial f_6}{\partial F} = 0 \tag{5.131}$$

$$\frac{\partial f_6}{\partial x_F} = 0 \tag{5.132}$$

$$\frac{\partial f_6}{\partial V} = (y_7 - y_6)/M_6 \tag{5.133}$$

$$\frac{\partial f_6}{\partial L_R} = (x_5 - x_6)/M_6 \tag{5.134}$$

7. $$f_7 = \{L_R x_6 - L_7 x_7 + V(y_8 - y_7) + F x_F\}/M_7 \quad (5.135)$$

where $L_j = L_R + F, \forall j = 7,11$

$$\frac{\partial f_7}{\partial x_j} = 0 \quad \forall j = 1,5 \quad (5.136)$$

$$\frac{\partial f_7}{\partial x_6} = L_R/M_7 \quad (5.137)$$

$$\frac{\partial f_7}{\partial x_7} = -(L_7 + K_7 V)/M_7 \quad (5.138)$$

$$\frac{\partial f_7}{\partial x_8} = VK_8/M_7 \quad (5.139)$$

$$\frac{\partial f_7}{\partial x_j} = 0 \quad \forall j = 9,11 \quad (5.140)$$

$$\frac{\partial f_7}{\partial F} = (x_F - x_7)/M_7 \quad (5.141)$$

$$\frac{\partial f_7}{\partial x_F} = F/M_7 \quad (5.142)$$

$$\frac{\partial f_7}{\partial V} = (y_8 - y_7)/M_7 \quad (5.143)$$

$$\frac{\partial f_7}{\partial L_R} = (x_6 - x_7)/M_7 \quad (5.144)$$

8. $$f_8 = \{L_7 x_7 - L_8 x_8 + V(y_9 - y_8)\}/M_8 \quad (5.145)$$

$$\frac{\partial f_8}{\partial x_j} = 0 \quad \forall j = 1,6 \tag{5.146}$$

$$\frac{\partial f_8}{\partial x_7} = L_7/M_8 \tag{5.147}$$

$$\frac{\partial f_8}{\partial x_8} = -(L_8 + VK_8)/M_8 \tag{5.148}$$

$$\frac{\partial f_8}{\partial x_9} = VK_9/M_8 \tag{5.149}$$

$$\frac{\partial f_8}{\partial x_j} = 0 \quad \forall j = 10,11 \tag{5.150}$$

$$\frac{\partial f_8}{\partial F} = (x_7 - x_8)/M_8 \tag{5.151}$$

$$\frac{\partial f_8}{\partial x_F} = 0 \tag{5.152}$$

$$\frac{\partial f_8}{\partial V} = (y_9 - y_8)/M_8 \tag{5.153}$$

$$\frac{\partial f_8}{\partial L_R} = (x_7 - x_8)/M_8 \tag{5.154}$$

9. $$f_9 = \{L_8 x_8 - L_9 x_9 + V\,(y_{10} - y_9)\}\,/M_9 \tag{5.155}$$

$$\frac{\partial f_9}{\partial x_j} = 0 \quad \forall j = 1,7 \tag{5.156}$$

$$\frac{\partial f_9}{\partial x_8} = L_8/M_9 \tag{5.157}$$

$$\frac{\partial f_9}{\partial x_9} = -(L_9 + VK_9)/M_9 \tag{5.158}$$

$$\frac{\partial f_9}{\partial x_{10}} = VK_{10}/M_9 \tag{5.159}$$

$$\frac{\partial f_9}{\partial x_{11}} = 0 \tag{5.160}$$

$$\frac{\partial f_9}{\partial F} = (x_8 - x_9//M_9 \tag{5.161}$$

$$\frac{\partial f_9}{\partial x_F} = 0 \tag{5.162}$$

$$\frac{\partial f_9}{\partial V} = (y_{10} - y_9)/M_9 \tag{5.163}$$

$$\frac{\partial f_9}{\partial L_R} = (x_8 - x_9)/M_9 \tag{5.164}$$

10. $$f_{10} = \{L_9 x_9 - L_{10} x_{10} + V(y_{11} - y_{10})\}/M_{10} \tag{5.165}$$

$$\frac{\partial f_{10}}{\partial x_j} = 0 \quad \forall j = 1,8 \tag{5.166}$$

$$\frac{\partial f_{10}}{\partial x_9} = L_9/M_{10} \tag{5.167}$$

$$\frac{\partial f_{10}}{\partial x_{10}} = -(L_{10} + VK_{10})/M_{10} \tag{5.168}$$

$$\frac{\partial f_{10}}{\partial x_{11}} = VK_{11}/M_{10} \tag{5.169}$$

$$\frac{\partial f_{10}}{\partial F} = (x_9 - x_{10})/M_{10} \tag{5.170}$$

$$\frac{\partial f_{10}}{\partial x_F} = 0 \tag{5.171}$$

$$\frac{\partial f_{10}}{\partial V} = (y_{11} - y_{10})/M_{10} \tag{5.172}$$

$$\frac{\partial f_{10}}{\partial L_R} = (x_9 - x_{10})/M_{10} \tag{5.173}$$

11. $$f_{11} = (L_{10}x_{10} - L_{11}x_{11} - Vy_{11})/M_{11} \tag{5.174}$$

$$\frac{\partial f_{11}}{\partial x_j} = 0 \quad \forall j = 1,9 \tag{5.175}$$

$$\frac{\partial f_{11}}{\partial x_{10}} = L_{10}/M_{11} \tag{5.176}$$

$$\frac{\partial f_{11}}{\partial x_{11}} = -(L_{11} + VK_{11})/M_{11} \tag{5.177}$$

$$\frac{\partial f_{11}}{\partial F} = x_{10}/M_{11} \tag{5.178}$$

$$\frac{\partial f_{11}}{\partial x_F} = 0 \tag{5.179}$$

$$\frac{\partial f_{11}}{\partial V} = -y_{11}/M_{11} \tag{5.180}$$

$$\frac{\partial f_{11}}{\partial L_R} = \frac{x_{10}}{M_{11}} \tag{5.181}$$

12. $$f_{12} = \frac{dF}{dt} = 0 \tag{5.182}$$

$$\frac{\partial f_{12}}{\partial x_j} = 0 \quad \forall j = 1,15 \tag{5.183}$$

13. $$f_{13} = \frac{dx_F}{dt} = 0 \tag{5.184}$$

$$\frac{\partial f_{13}}{\partial x_j} = 0 \quad \forall j = 1,15 \tag{5.185}$$

14. $$f_{14} = \frac{dV}{dt} = 0 \tag{5.186}$$

$$\frac{\partial f_{14}}{\partial x_j} = 0 \quad \forall j = 1,15 \tag{5.187}$$

15. $$f_{15} = \frac{dL_R}{dt} = 0 \tag{5.188}$$

$$\frac{\partial f_{15}}{\partial x_j} = 0 \quad \forall j = 1,15 \tag{5.189}$$

5.2.6 The Measurement Vector z_{mx1}

The measurement vector comprises five plate temperature and two flow measurements:

$$z^T = (T_1, T_3, T_5, T_7, T_9, F, L_R) \tag{5.190}$$

5.2.7 The Measurement Matrix M_{mxn}

Since the state variables are not measured (on-line composition measurement is normally a more involved task), a functional relationship h is needed to relate temperature T to plate liquid composition x, i.e. $T = h(x)$.

Consider the following:

On a tray,
by Raoult's Law:
$$P_1(T) + P_2(T) = x_1 P_1^o(T) + x_2 P_2^o(T) = P \qquad (5.191)$$

where P = total pressure, mm.Hg.

P_i = partial pressure of component i, mm.Hg.

P_i^o = vapour pressure of i, mm.Hg.

Thus,
$$x = \frac{P - P_2^o(T)}{P_1^o(T) - P_2^o(T)} \qquad (5.192)$$

Further, by using Dalton's Law:
$$P_i = y_i P \qquad (5.193)$$

and the Antoine relationship
$$\log P_i^o(T) = A_i - \frac{B_i}{C_i + T} \qquad (5.194)$$

it can be shown that

$$\frac{\partial T}{\partial x} = \frac{\{P_1^o(T) - P_2^o(T)\}^2}{-\frac{\partial P_2^o}{\partial T}\{P_1^o(T) - P_2^o(T)\} - \{\frac{\partial P_1 O(T)}{\partial T} - \frac{\partial P_2^o(T)}{\partial T}\}\{P - P_2^o(T)\}} \qquad (5.195)$$

where
$$\frac{\partial P_i^o(T)}{\partial T} = \frac{2.303\ B_i}{(C_i + T)^2} P_i(T).$$

Thus, the measurement matrix, M consists of zero elements everywhere except

$$M(1,1) = \frac{\partial T_1}{\partial x_1} \quad , \quad M(2,3) = \frac{\partial T_3}{\partial x_3} \quad , \quad M(3,5) = \frac{\partial T_5}{\partial x_5}$$

$$M(4,7) = \frac{\partial T_7}{\partial x_7} \quad , \quad M(5,9) = \frac{\partial T_9}{\partial x_9}$$

$M(6,12) = 1.$ and $M(7,15) = 1.$

If instead of temperatures, compositions were measured, then the non-zero M-matrix elements described above are all unity.

5.2.8 Filter Initialisation and Tuning

To start the filter operation $\hat{x}(0,0)$, $P(0,0)$, R and Q must be specified. An off-line simulation of a Mcabe-Thiele analysis of the distillation for a typical operating condition, yields the temperature, hold-up, liquid and vapour, and composition profiles which can be used to initialise the actual and filter process models for simulation purposes. Typical values for $P(0,0)$, R and Q are given below:

$P(0,0)$ = diag (0.001, ... 0.001, .04, 0.01, .04, .04)

R = diag (.01, .01, .01, .01, .01, 1.0, 1.0)

Q = diag (.01, .01, .01, .01, .01, .01, .005, .005, .005, .005, .005, 1.0, .15, 1.0, 1.0)

The significance of these numbers will be discussed in the next chapter.

5.2.9 Conclusion

The theoretical background of a Kalman filtering application has been described in this chapter although the emphasis has been on simulation on the H316 minicomputer. However, simulation and off-line

modelling are normally the first steps in an on-line estimation work. The preceding discussions have indicated that considerable model development in the context of limited computing resources is necessary before applying a Kalman filter to a process plant. In this case, the plant dynamic models are more or less established. The case may be similar in an actual industrial situation for good plant management would have some record of a plant's dynamic behaviour which may have been required for other design or control purposes.

The next move is to examine the filter performance in simulation on the H316 minicomputer before an on-line attempt is made. This is the essence of the material covered in the next chapter.

CHAPTER SIX

TOTAL SIMULATION PACKAGE - PROCESS MODELS AND ESTIMATION

6. TOTAL SIMULATION PACKAGE - PROCESS MODELS AND ESTIMATION

The ideas formulated in Chapter 5 are put together into a digital simulation package which is described in this chapter.

6.1 Software Objectives

The total simulation package includes modules which allow dynamic response studies of process models as well as estimation for a fixed dimensional system in an interactive manner. The objectives of the package are basically two-fold:

1. To investigate the feasibility of implementing on-line estimation and control of a distillation process using the linked H316-M6800 twin processor system.
2. To allow the user to become familiar with the physics of the process and the operational features of a Kalman filter.

In terms of program structure and philosophy, the package is basically similar to that implemented by Daie. In terms of software resources used and computing constraints, it is completely different. In fact the dimensionality of the filtering problem had to be reduced to suit the requirements of the H316 minicomputer system.

The simulation package allows the user to investigate the process dynamics of both the filter and actual process models 'simultaneously' thus helping the designer to make judicious assignments to the Q matrix for a filtering experiment. For example by disturbing certain parameters in the process and noting the response of the state vector, one can derive a feeling for the 'stiffness' of the column.

The prime concern however is speed and to a lesser extent, storage. The numerical integration process would be expected to consume much of machine time in model response studies on estimation. The

package allows the user to investigate the possibility of using larger integrator step lengths for a meaningful application, thereby shortening computation times for a given sampling interval.

The simulation package is also constructed with an on-line application in mind. In other words, use of storage areas which would normally be occupied by core-resident modules such as the HADIOS Executive, is minimised. It was quickly learnt that the use of floppy disks for program overlay is inevitable.

The estimation modules allow the user to tune his filter and study aspects of numerical stability and filter divergence. Having found a reasonable filter operating range, he can then start formulating his on-line package.

6.2 The Simulation Package

Although the simulation package allows the user to conduct dynamic response and digital control studies in an interactive manner, estimation is restricted to a particular filtering problem only. This is because the COMMON areas used by different program modules have fixed addresses for a given problem dimension. Thus, several packages are actually available, each having a different state and/or measurement vector dimensions.

6.2.1 Software Requirements

It was decided that the user is interfaced to the fixed modules of the simulation package via interpretive BASIC. This would allow him to specify important variables such as integrator step-length, filter tuning and graphics package parameters, etc. with ease. The fixed modules of the package i.e. the process models, the filter algorithm, the integrator, etc. are implemented in FORTRAN, basically for faster

execution. BASIC mathematics library routines are used as far as possible.

Note that the BASIC interface has provided some degree of flexibility in the sense that the integrator step-length can be altered during computation and one is free to specify the type, magnitude and time of a process disturbance. For studying controller designs implemented at the BASIC level, two controller outputs (reflux rate and reboiler duty) have been provided. Input/output tasks are exclusively handled by BASIC statements.

6.2.2 The Need for Program Overlay and Construction of the Package

The problem of using the floppy disk for secondary storage on the H316 was first tackled by Jordon and Gay[123] and was first used in an on-line environment by Mukesh[124]. The existing software allows program segments, previously stored on the disk using the utility routine A$D14, to be overlayed thus greatly increasing the size of available memory. However, its core image occupied unfavourable locations. A new core image was constructed for the purpose of the simulation package. The procedure is described in Table A6.1 of Appendix 6.

Because the simulation package is too large for the available core, it was decided to split the package into 5 segments, one segment of which will be in core at any one time. The construction of the five segments are described in detail in Tables A6.2 to A6.6 of Appendix 6. Basically, the schematic representation of the package is shown in Figure 6.1. The names listed which are not defined in Figure 6.1 refer to the FORTRAN subroutines used in the total package. Clearly, several subroutines can now share the same starting address as indicated by

'0	'7160	'20000	'27000	'33143	
BASIC	User Space	KOMMON, INIT, INIT1, INIT2, INIT3, P10F10, TRANS, MATMUL, MATTPX	COMN, MATADD, SDBUB, DIAADD, FDTX, SDINT, KALMA1, MATTPS, INTPAS, INTJS, MATHS, DTOC	COMMON variables	1
"	"	KOMMON, DCOL, DCOL1, DCOL2, DCOL3, DCOL4, DCOL5, DCOL6, STEP, SWAVE, RAMP, PERTUB.	"	"	2
"	"	KOMMON, SIMUL, SIMULX, VECTOR, KALMAN, MATINV, P10F10, DIAMUL, DIASUB, MATTPX, TRANS, PKK, MATVEC	"	"	3
"	"	GRAPHICS Package (37 subroutines + MATHS.)	"	"	4
"	"	KOMMON, KOMN	"	"	5

Figure 6.1 Schematic Construction of the Total Simulation Package

Table 6.1, which shows the contents of the BASIC CALL table.

A description of all the FORTRAN subroutines is given later in Section 6.2.4. DTOC is a FORTRAN interface routine for the slightly

modified Read/Write Sector (RWSC) routine of the disk utility A$D14.

Table 6.1 Subroutine Assignments in the BASIC CALL table

No.	Location in BASIC	Subroutine	Address
1	'516	DTOC	'30000
2	'517	KOMN,GRAPH,SIMUL,DCOL,INIT	'20006
3	'520	P1OF1O	'24000
4	'521	COMN	'27000
5	'522	KALMAN	'22000
6	'523	PKK	'26000
7	'524	PERTUB	'25000
8	'525	DISTUB (not in estimation)	'23000
9	'526	*	-
10	'527	SU1O (location varies)	'33620

where * means unused location.

6.2.3 Steady-State Profiles

6.2.3.1 Mcabe-Thiele Calculations

Due to storage limitations and since steady-state profiles for a given process condition are used only once by the filter and actual process models, steady-state calculations were done in BASIC. The first program, listed in Table A6.7 of Appendix 6, calculates the time-invariant, average Antoine constants, 3 for each component.

The equation used is that of equation (5.194).

$$\log P_i^o(T) = A_i - \frac{B_i}{C_i + T}$$

The second program, listed in Table A6.8 of Appendix 6 does a Mcabe-Thiele analysis of the column for a given operating condition. Thus, for a given feed rate, temperature and composition, reflux ratio and temperature, top and bottom product specifications, top or bottom product rate, the program calculates the heat duty and the following profiles:

1. Plate liquid and vapour concentration profiles for the MVC.
2. Plate hold-ups (mol.)
3. Plate temperatures (^oC)
4. Liquid rates (mol./hr.)
5. Plate Murphree vapour-phase efficiency, E_{mv}.

The reflux ratio is calculated using a Wegstein algorithm to speed up convergence. The boil-up rate becomes the vapour rate through the column which is constant. The plate efficiency calculations are based on the A.I.Ch.E. method[122], originally developed for binary mixtures on bubble cap trays and was later extended to other types of trays[125]. For a detailed description of the method, references (119), (122) and (125) should be consulted.

6.2.3.2 Refined Steady States

The Mcabe-Thiele calculations provide approximate profiles for the actual and filter process models (models I and II respectively). As indicated in the program in Table A6.8, the model used to calculate these profiles is basically model I except that an option for plate

efficiency calculations is offered and that the vapour rate throughout the column is constant. Since model II is a much simpler model than model I, it is not surprising that when the two models are initialised with the Mcabe-Thiele profiles, the state vector in model II moves to a new point in space driving its differential equations to zero while the state generated by model I basically needs little adjustment. This point is shown by the corresponding variations in composition, vapour and temperature profiles in Figures 6.2 to 6.3. The final steady-states are termed refined steady-states and may be used to initialise the filter or the dynamic simulation modules by punching them out first on to paper-tape.

6.2.4 FORTRAN Subroutines

Excluding the graphics steering routine, GRAPH, and the utility SU10, the total simulation package contains 38 FORTRAN subroutines, 10 of which are accessible from BASIC. The modules and their functions are described below. Their source listings are found in Tables A6.9 to A6.11 of Appendix 6, where the arguments in the CALL statements are also defined.

KOMMON - This is a non-executable routine. Basically it contains all the COMMON blocks used in package. Its purpose is to align the COMMON variables in all the segments correctly. For this reason, it is the first object module loaded in the construction of any of the 5 segments that constitutes the simulation package.

KOMN - BASIC statement - CALL(2,C(1),L6,L7,L8,L9).

It is called to retrieve array values (P,Q,R,M and matrix inverse in filter simulation experiments) which are returned columnwise in the vector C.

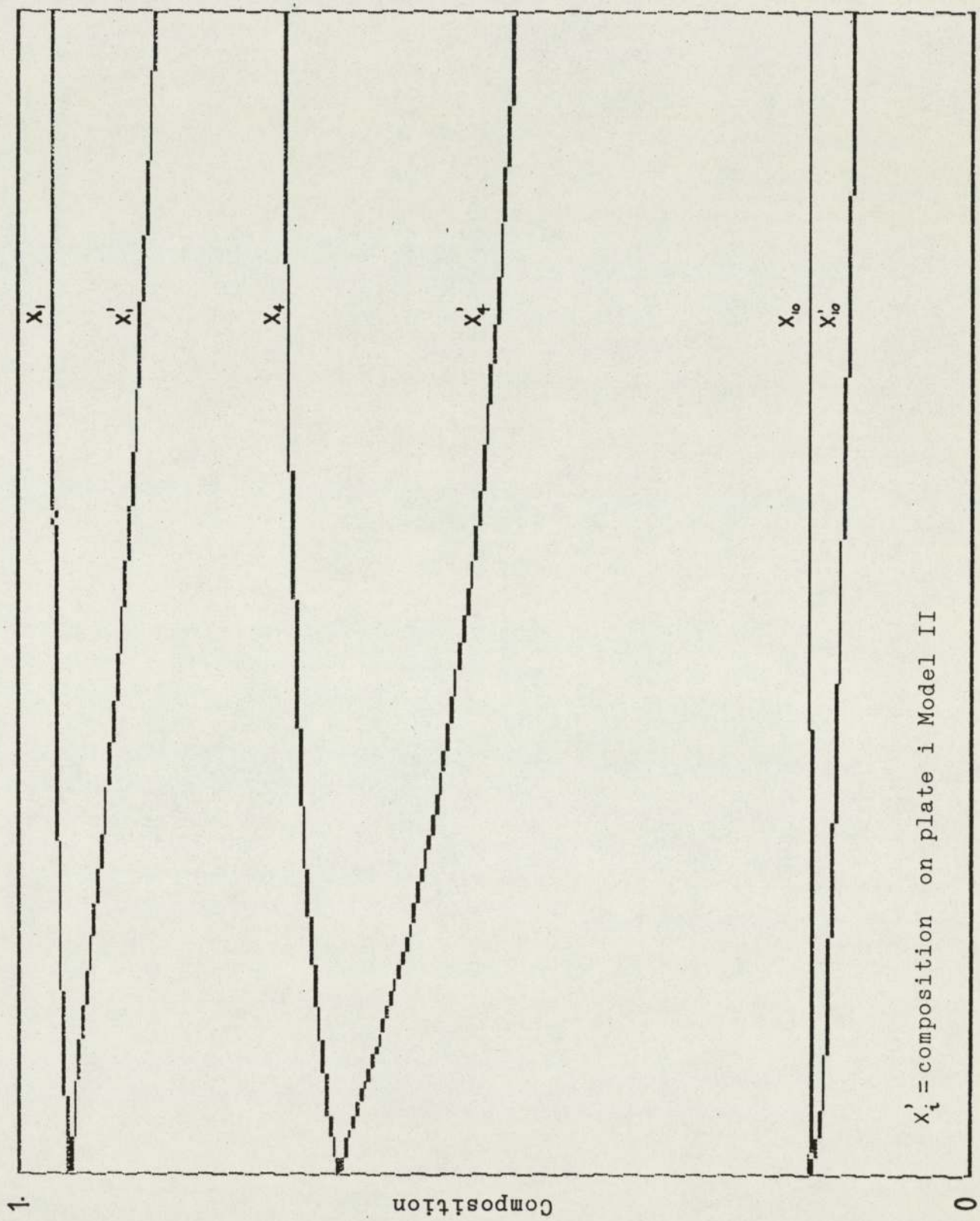


Figure 6.2 Refinement of Steady States in Models I and II - Compositions

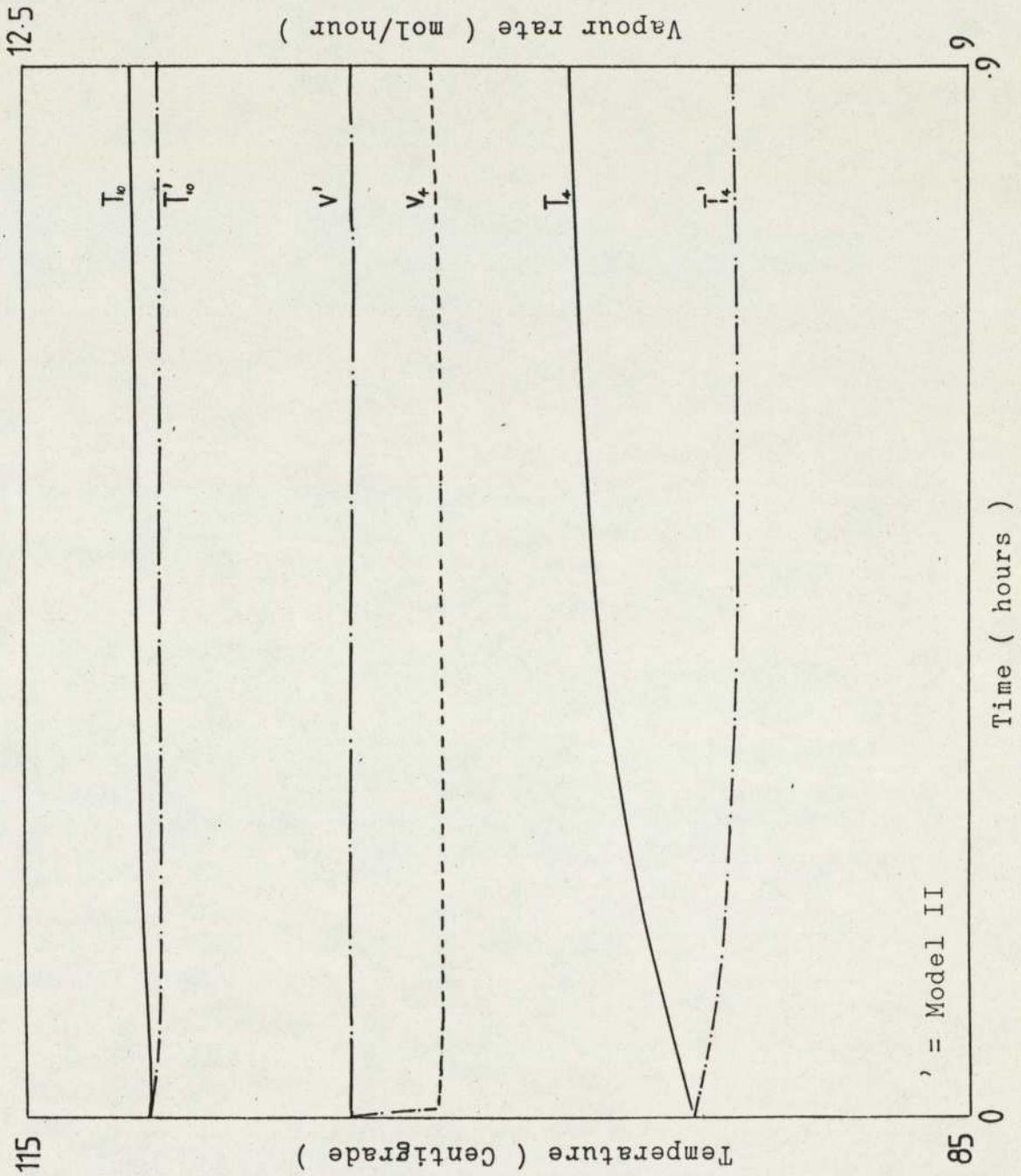


Figure 6.3 Refinement of Steady States in Models I and II - Temperatures and Vapour Rates

COMN - BASIC statement - CALL(4,C(1),D(1),E(1),RO,QO,E2,E3)

This routine is among the 10 FORTRAN modules that are core resident. It is used to retrieve current profiles (compositions, flowrates, temperatures, etc.) of the two process models. The estimated profiles are returned in vector E.

Since it can be called anytime (or anywhere) in the BASIC program to change the reflux rate (RO) and heat input (QO) to the actual process model, the routine can also be used to output the control vector {RO,QO} in digital control simulation studies.

INIT - BASIC statement - CALL(2,C(1),D(1),E(1),E1,E2,E4,V1)

This is an initialisation routine. Other related routines are INIT1, INIT2 and INIT3.

DCOL - BASIC statement - CALL(2,T1,T2,Z1)

This, in conjunction with DCOL1, DCOL2, DCOL3, DCOL4, DCOL5 and DCOL6, forms the actual process dynamic model. Note that T1 is time, T2 is integrator step-length and Z1 is integration-method option. The variables are specified in BASIC which allows easy changes to be made.

PERTUB - BASIC statement - CALL(7,T1,S(1),R(1),W(1),F1,F2)

This routine simulates actual process disturbances at the desired times (step, ramp or sinusoidal of up to 5 process variables: feed rate, temperature and composition, reflux rate and temperature). It therefore calls subroutines STEP, RAMP or SWAVE where appropriate.

DISTUB - This is similar to PERTUB except that the disturbances affect only the filter process model. It is usually used in modelling studies of model II.

SDBUB - This routine calculates bubble points and vapour equilibrium compositions. It includes a Newton-Raphson iteration with an adjustable limit in the bubble point accuracy.

SDINT - This routine, in conjunction with INTPAS and INTJS forms the integrator. Numerical integration methods offered are simple and modified Euler methods only.

SIMUL - BASIC statement - CALL(2,T3,T4,Z2,T9)

Together with SIMULX, they constitute the filter process dynamic mode. Note that T3 (time), T4 (integration step length) and Z2 (integration method option) can be different to the ones used in Model I when both are being 'simultaneously' integrated. T9 is a steering flag. If zero, normal integration; else the secondary variables (flowrates, temperatures, etc.) are updated with the latest estimated state vector.

P1OF1O - BASIC statement - CALL(3,S9)

This routine, in conjunction with TRANS (transition matrix calculations) computes the predicted error covariance matrix P(k+1,k). S9 is the sampling intervals in hours.

PKK - BASIC statement - CALL(6)

This routine computes the estimated error covariance matrix P(k+1,k+1).

KALMAN - BASIC statement - CALL (5,Y(1),U(1),K(1),M(1),L(1),P9,D9)

Together with KALMA1, they constitute the main Kalman filter estimation steps. Noisy process measurements are passed to the FORTRAN subroutines via vector Y.

FDTX - This routine is used to compute temperature-composition derivatives required for the measurement matrix (equation 5.195 in Chapter 5) of the extended Kalman filter.

The rest are matrix and vector manipulation routines.

MATMUL - Matrix Multiplication. The product matrix is returned to the calling subroutine via either one of the matrices in the argument list.

VECTOR - Vector addition or subtraction.

MATINV - Matrix inversion based on the Gauss-Jordan method.

MATVEC - Matrix-Vector multiplication.

DIAMUL - Matrix multiplication, the first being a diagonal matrix.

DIASUB - Matrix subtraction, the second being a diagonal matrix.

MATTPS - Matrix transpose.

MATTPX - Matrix transpose but unlike MATTPS, it puts the answer in the matrix to be transposed.

MATADD - Matrix addition.

DIAADD - Matrix addition where one of the matrices is diagonal.

6.3 Simulation of the Dynamic Response of the Process Models

The total simulation package allows the user to conduct dynamic response studies of both process models simultaneously. A typical flow-chart for such a simulation exercise is shown in Figure 6.4. Each model may be initialised with its steady-state profiles and then subjected to multiple process disturbances. Since it is useful to know how well the filter process model approximate the 'true' plant (Model I), each model is integrated using the same step length and subjected to the same disturbances at the same points in time.

6.3.1 Response to a single step disturbance in feed rate

When the process models are subjected to a single 40% step change in feed rate, the typical responses for plate liquid compositions of the MVC are shown in Figures 6.5 and 6.6. The response of plate temperatures are shown in Figure 6.7. In this case, the models have been initialised with their individual refined steady states and the integration step length, using the simple Euler method, is .0001 hour.

The results show that dynamically, the filter process model (Model II) is a reasonable approximation to the 'true' plant although its steady-state behaviour is markedly different. For example, the liquid composition on plate 4 settles to about .84 in Model I whereas in the

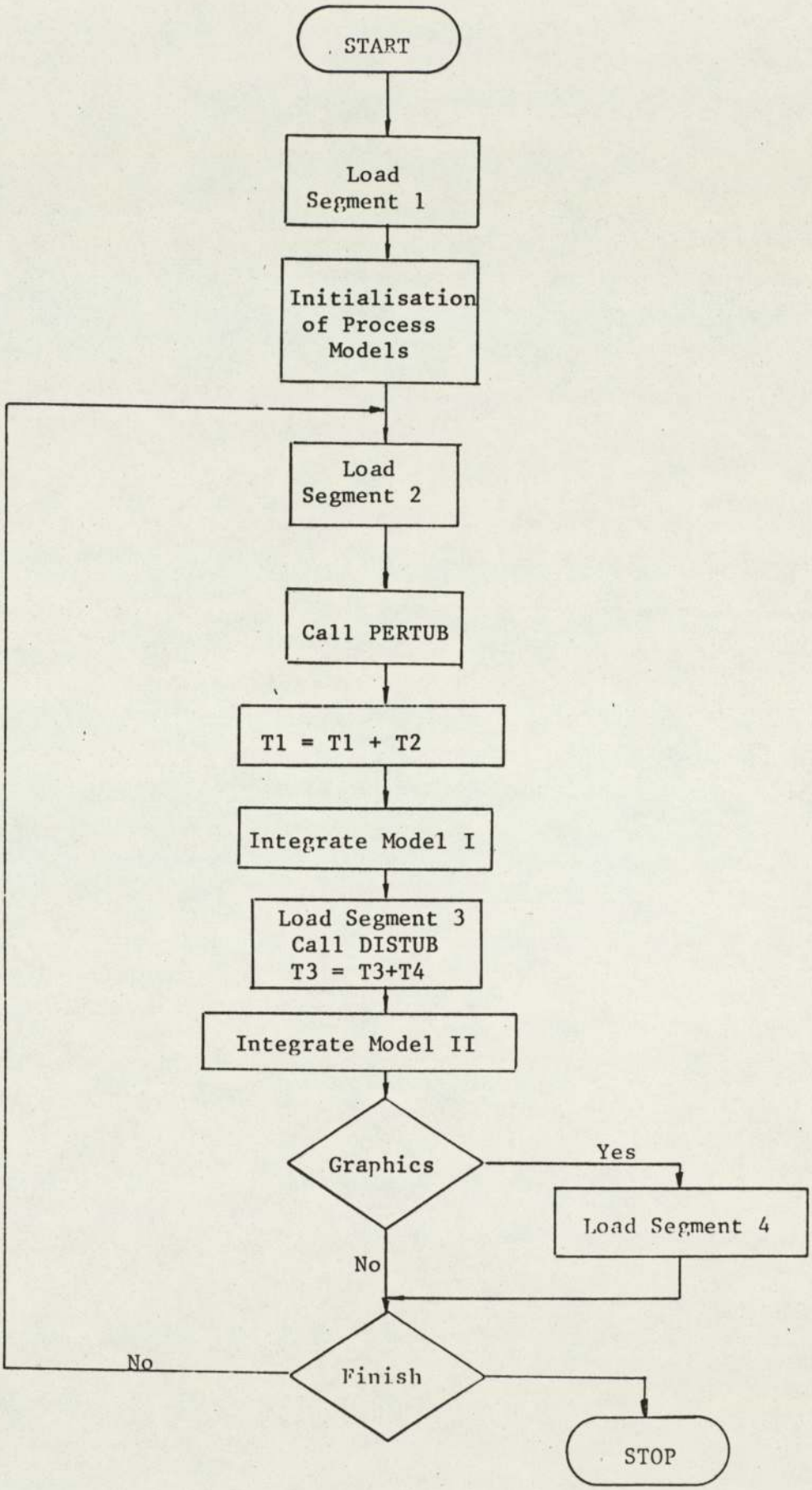


Figure 6.4 Flowchart for Simulation of Dynamic Response of Process Models

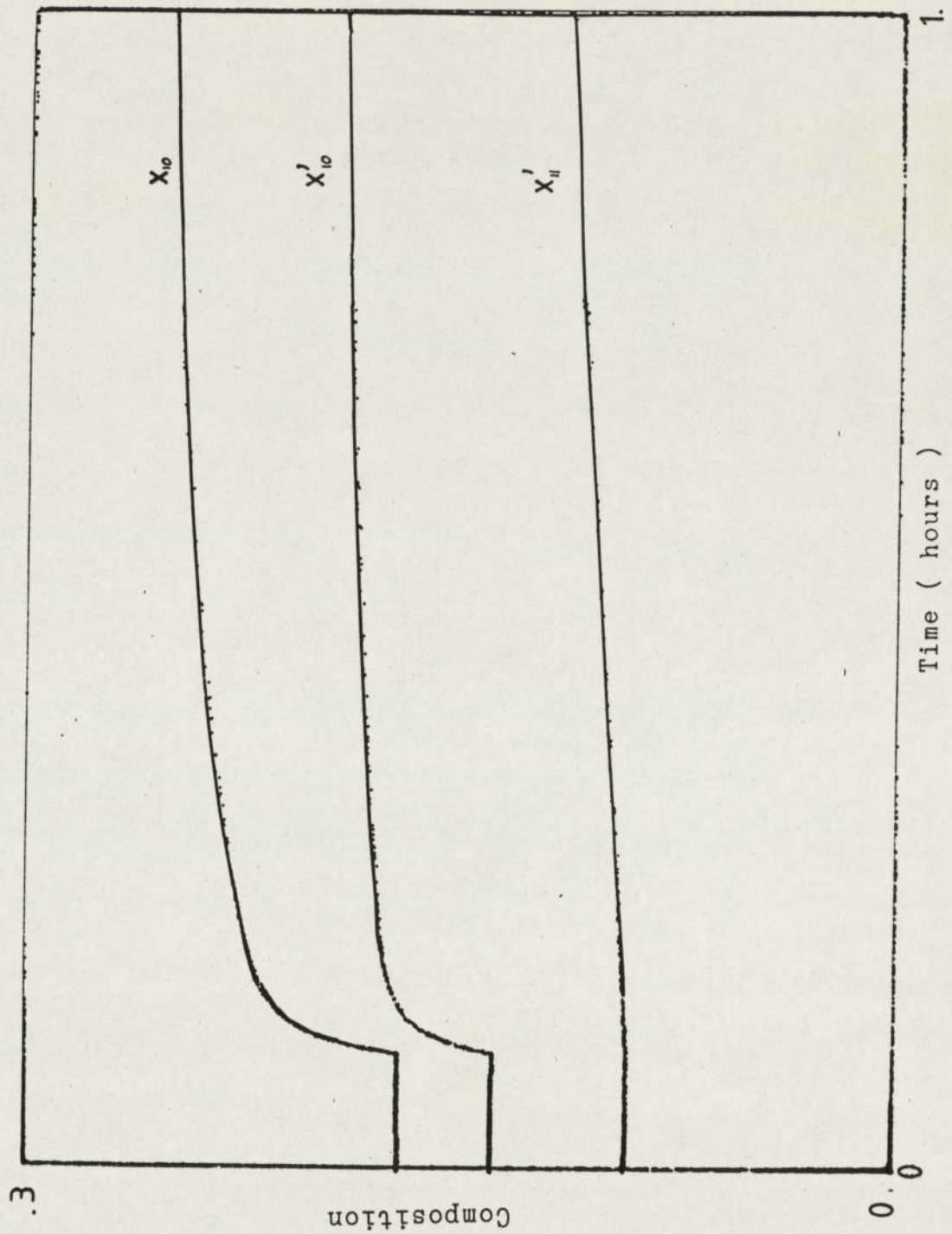


Figure 6.5 Response of tray liquid compositions due to a 40% step change in feed rate

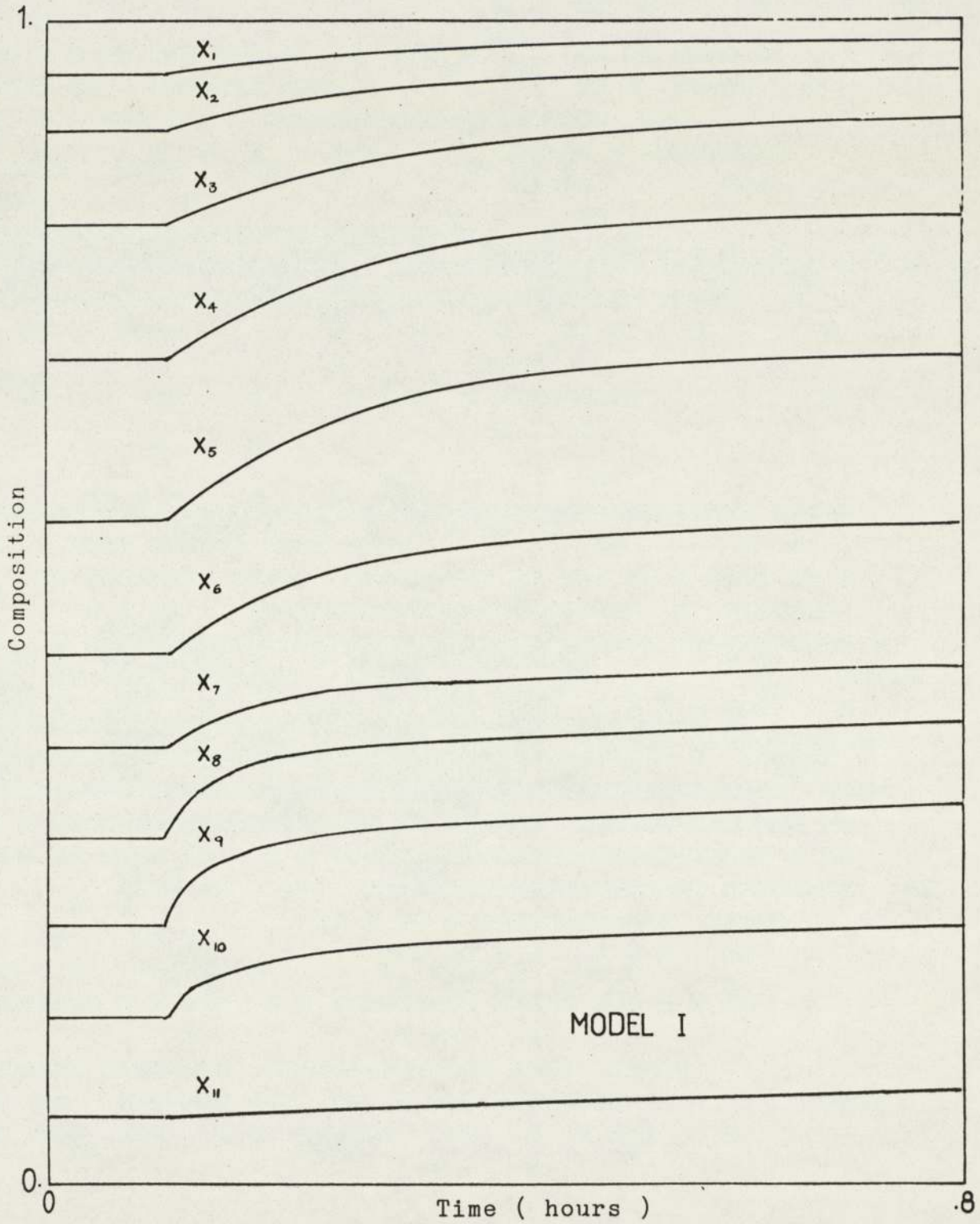


Figure 6.6 Response of tray liquid compositions due to a 40% step change in feed rate

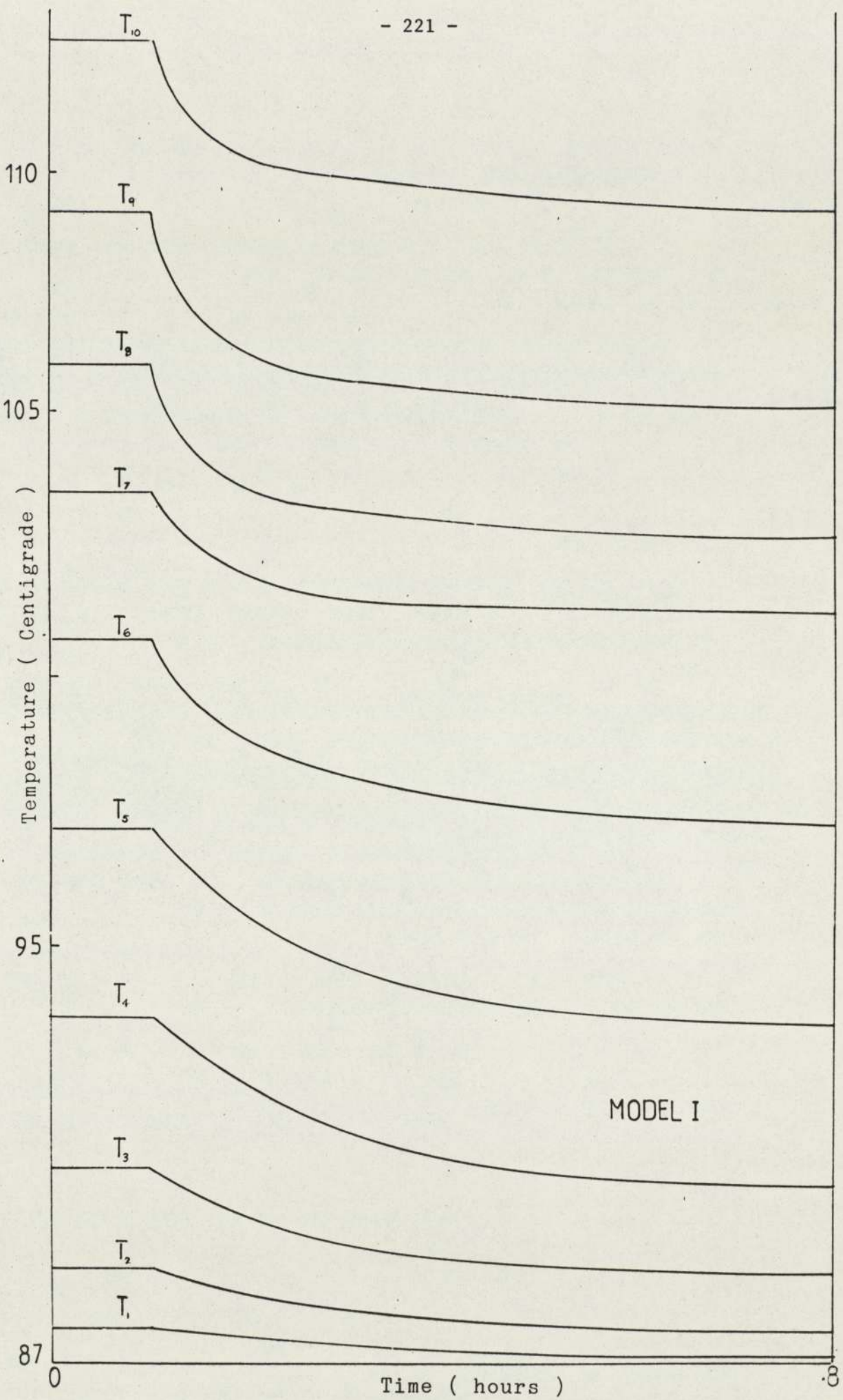


Figure 6.7 Response of Tray Temperatures due to a 40% step change in feed rate

filter process model, the corresponding value is about .61. The bottom product compositions are hardly affected because of the dampening effect of the relatively large reboiler hold-ups. Also, the responses above the feed plate (plate 7) are slower, increasingly so as one moves up the column and away from the feed plate, than those in the stripping section. This is because distillation is a stagewise process and the effect of a feed rate disturbance is translated into changes of compositions and internal flowrates which eventually work their way up the column.

6.3.2 Response to multiple disturbances in feed rate and composition

When both models are initialised with the steady-state profiles of Model I, and then subjected to multiple disturbances in feed rate and composition, the typical responses in plate liquid compositions for the MVC are shown in Figure 6.8. Clearly, in the stripping section, the behaviour is reasonably well modelled but above the feed plate the response from Model II is unsatisfactory. The results in Figures 6.2 and 6.3 suggest that because Model II is initialised with the steady state profiles of Model I, the driving force pulling Model II to its steady state values is significant (in fact the drive is stronger above the feed plate) thus swamping the effect of the load disturbances.

6.4 Kalman Filtering Simulation

In terms of software development carried out in this research, a considerable portion has been directed to the simulation of the Kalman filtering application formulated in Chapter 5. Several simulation packages are available, each differing in problem dimension in the sense of number of states and/or parameters to be estimated, and the number of process measurements. Only two such packages are discussed as they adequately reflect the typical findings and problems encountered. A

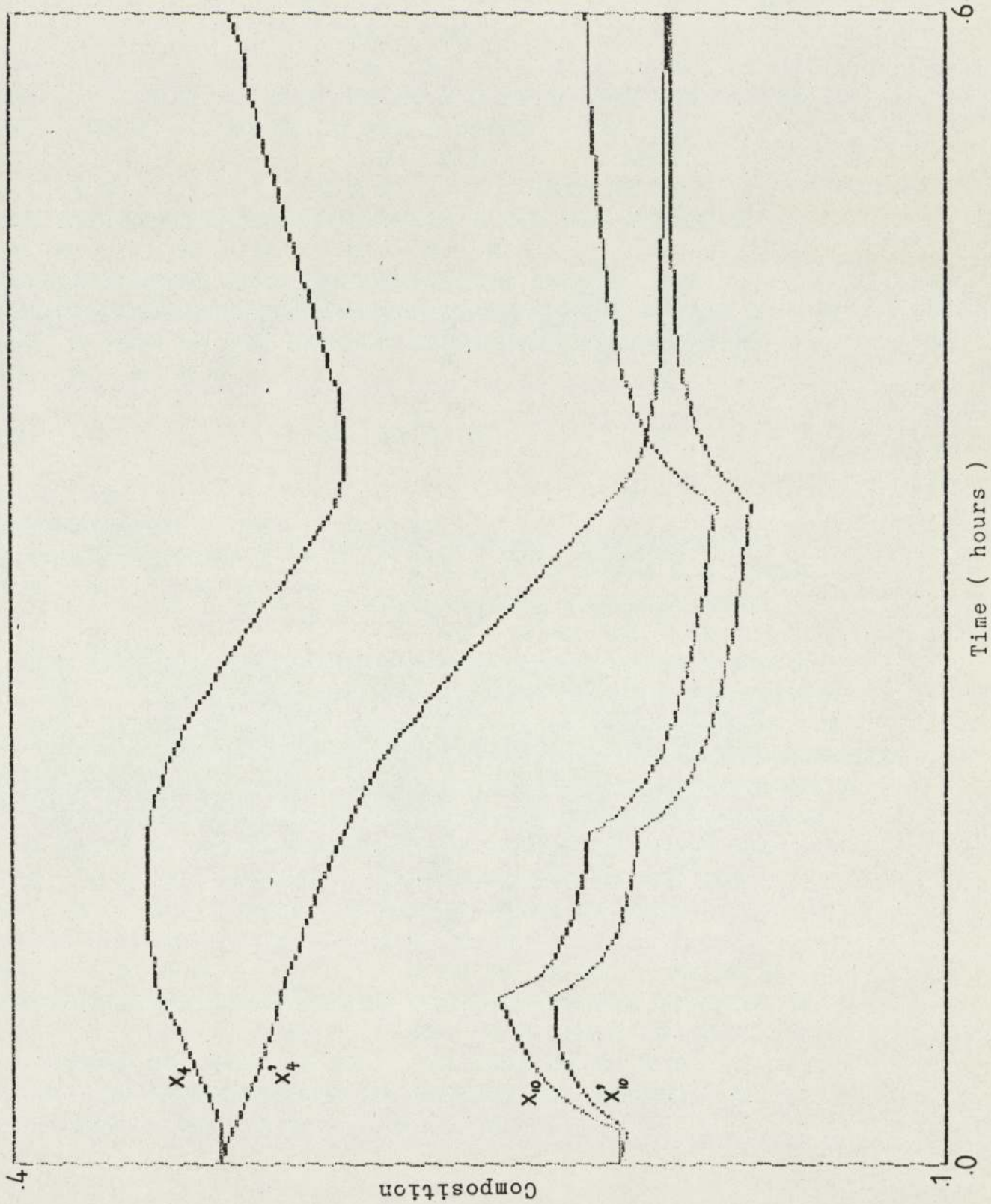


Figure 6.8 Response of Tray liquid compositions due to multiple disturbances in feed rate and composition

flowchart for such a simulation experiment is shown in Figure 6.9.

6.4.1 State and Parameter Estimator: EKF1

Given the limited size of core memory in the H316 minicomputer the original filtering problem considered by Daie[119], i.e. a 26 by 26 process model, a 22 by 22 filter process model and 3 measurements, has been reduced to the following:

a 15 by 15 process model (11 states and 4 parameters)

an 11 by 11 filter process model

and 7 process measurements.

The number of measurements, plate liquid compositions or temperatures, has been increased to compensate for the poorer predictions resulting from a simplified filter process model. The theoretical development for this exercise has already been covered in Chapter 5.

6.4.1.1 Initial estimate of the state

The refined steady-state profiles for Model I are usually used as an estimate of the state vector at time zero. Typically, this is for the separation:

Feed Rate	$F = 10$	mole/hr
Feed composition	$x_F = .4$	mole fraction of MVC
Top composition	$x_D = .95$	mole fraction of MVC
Bottom composition	$x_B = .05$	mole fraction of MVC
Reflux ratio	$R = 2.0$	

6.4.1.2 The Initial Error Covariance matrix, $P(0,0)_{15x15}$

P(0,0) is diagonal and its typical value is given by

$$P(0,0) = \text{diag}\ (.001,\ldots\ldots\ldots\ldots\ldots\ldots,.001,.04,.01,.04,.04)$$

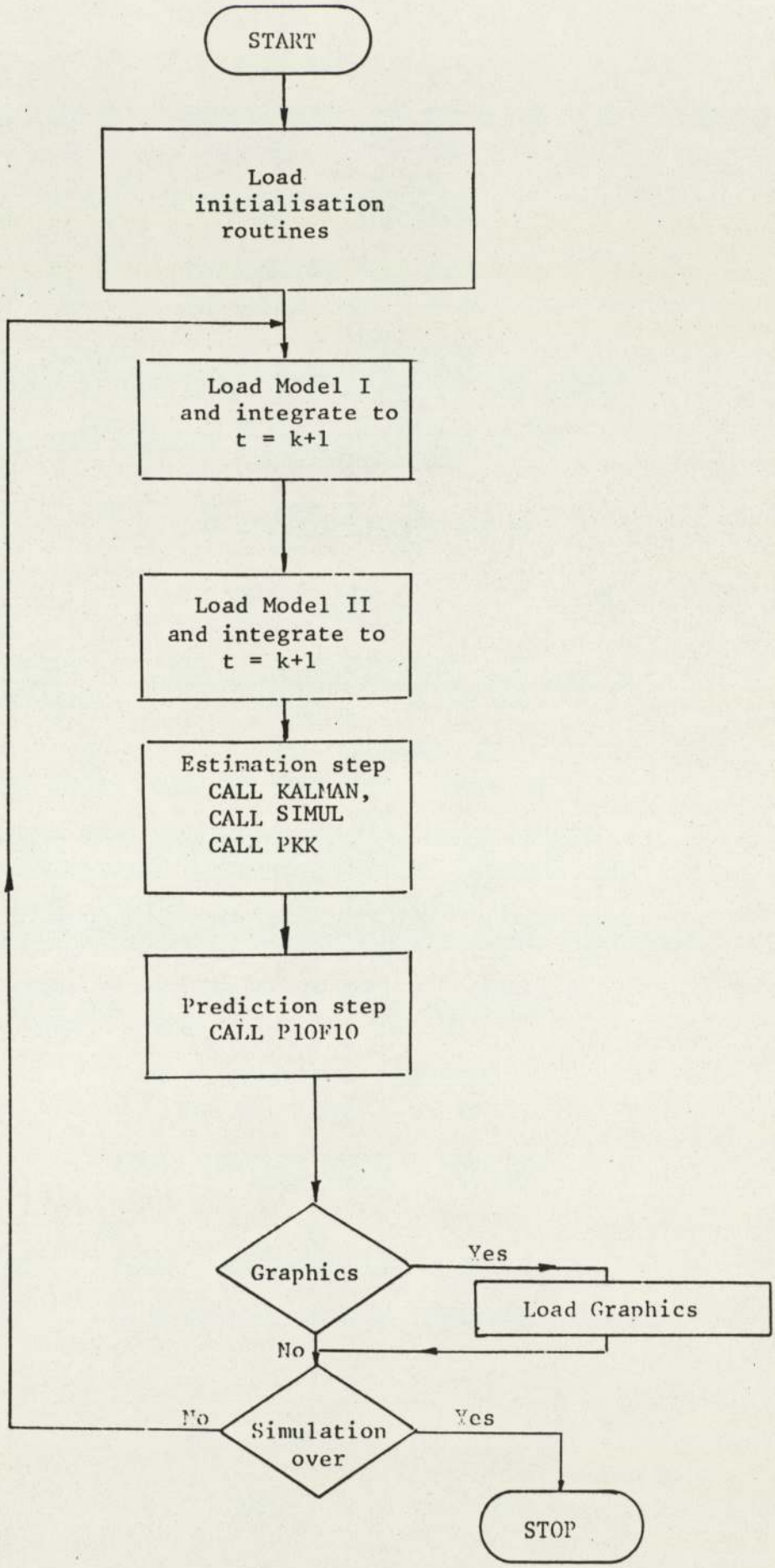


Figure 6.9 Flowchart for Simulation of a Kalman Filtering Application

This means that the standard deviations of the initial composition estimates, feed rate, feed composition, reflux rate and boil up rate are about 3%, 2%, 10%, 3% and 2% respectively.

6.4.1.3 The System Noise Matrix, Q_{15x15}

The system noise matrix is also diagonal and its typical value is given by

Q = diag (.01,.01,.01,.01,.01,.01,.005,.005,.005,.005,.005,1.0,.15,1.0,1.0)

Thus, during the prediction step, a noise term of longer amplitude is added on to the error covariances of the compositions above the feed plate as this region is less well modelled as evident earlier.

6.4.1.4 The Measurement Noise Matrix R_{7x1}

This is given by

R = diag (.01,.01,.01,.01,.01,1.0,1.0)

The standard deviations of the thermocouples, feed rate and reflux rate measurements are therefore about $.1^{o}C$, 10% and 16% respectively.

6.4.1.5 Simulated Gaussian Noise

Since the pseudo-random number generator is BASIC has an approximately flat probability distribution profile, a crude Gaussian noise is used based on 50 numbers generated by the RND function. The temperature measurements are corrupted by a sequence characterised by $(0,\sqrt{.01})$ and the flow measurements by $(0,\sqrt{0.5})$.

6.4.1.6 Observability

The discrete observability matrix L_{od} for this estimation problem for up to k sampling instants is given by

$$L_{od}\ [\Phi^T(1,0)M^T(1), \Phi^T(2,0)M^T(2), \ldots\ldots, \Phi^T(k,0)M^T(k)] \qquad (6.1)$$

The condition for observability for k sampling instants is that L_{od} has rank n[126], where n is 15. Since Φ and M vary with time, and L_{od} grows with k, the task may be carried out by some numerical means. It can be shown however, that since the feed flowrate F is being estimated in EKF1, a feed tray temperature (or composition) measurement must be included in the measurement vector. If not, the system is unobservable for all k.

6.4.1.7 The Effect of Sampling Interval

It was observed that estimation performance is sensitive to the choice of sampling interval, Δt. Figures 6.10, 6.11 and 6.12 graphically illustrate this point for sampling intervals of .002, .004 and .005 hour respectively. The rest of the filter parameters are the same. The filter became unstable as reflected in large values in the error covariance matrix when a sampling interval greater than 0.005 hour was used. The results in Figures 6.10, 6.11 and 6.12 show that although the filter's performance is acceptable, a combination of a less well modelled enriching section and a larger sampling interval has resulted in a larger bias in the composition estimates above the feed plate. On one hand, the larger Δt generates a poorer state transition matrix Φ. The other contribution to the loss of performance is due to the fact that if the integration parameters (method and size of step-length) are kept constant, model predictions are likely to be less accurate since more step-lengths are required over a larger Δt.

Figures 6.13 and 6.14 show, for a sampling interval of .003 hour, that a different initial estimate affects only the initial response of the filter. The first disturbance occurs at t = .1 hour which means that the filter is able to track the composition on plate 10 after about

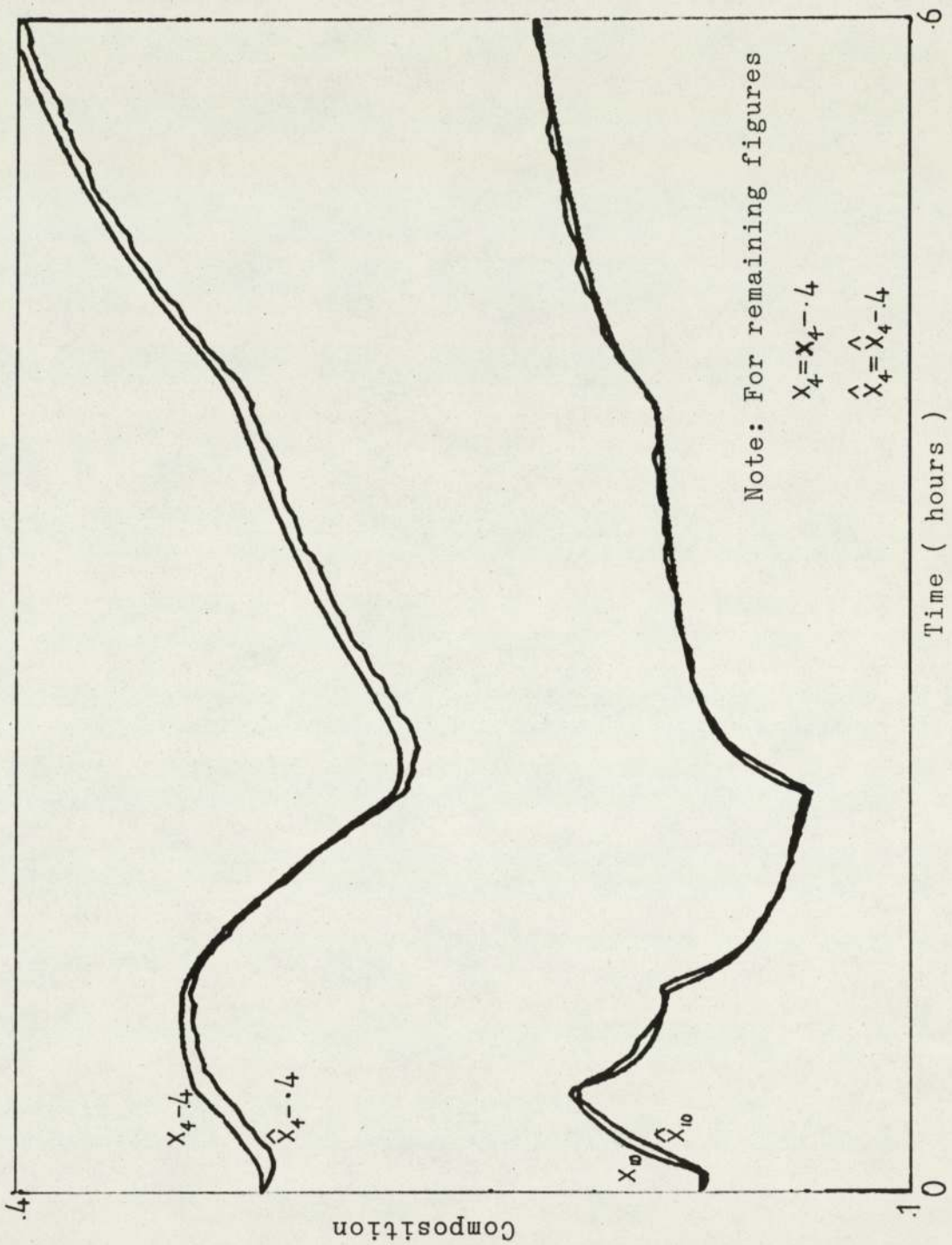


Figure 6.10 Estimation with EKF1: Δt = .002 hour

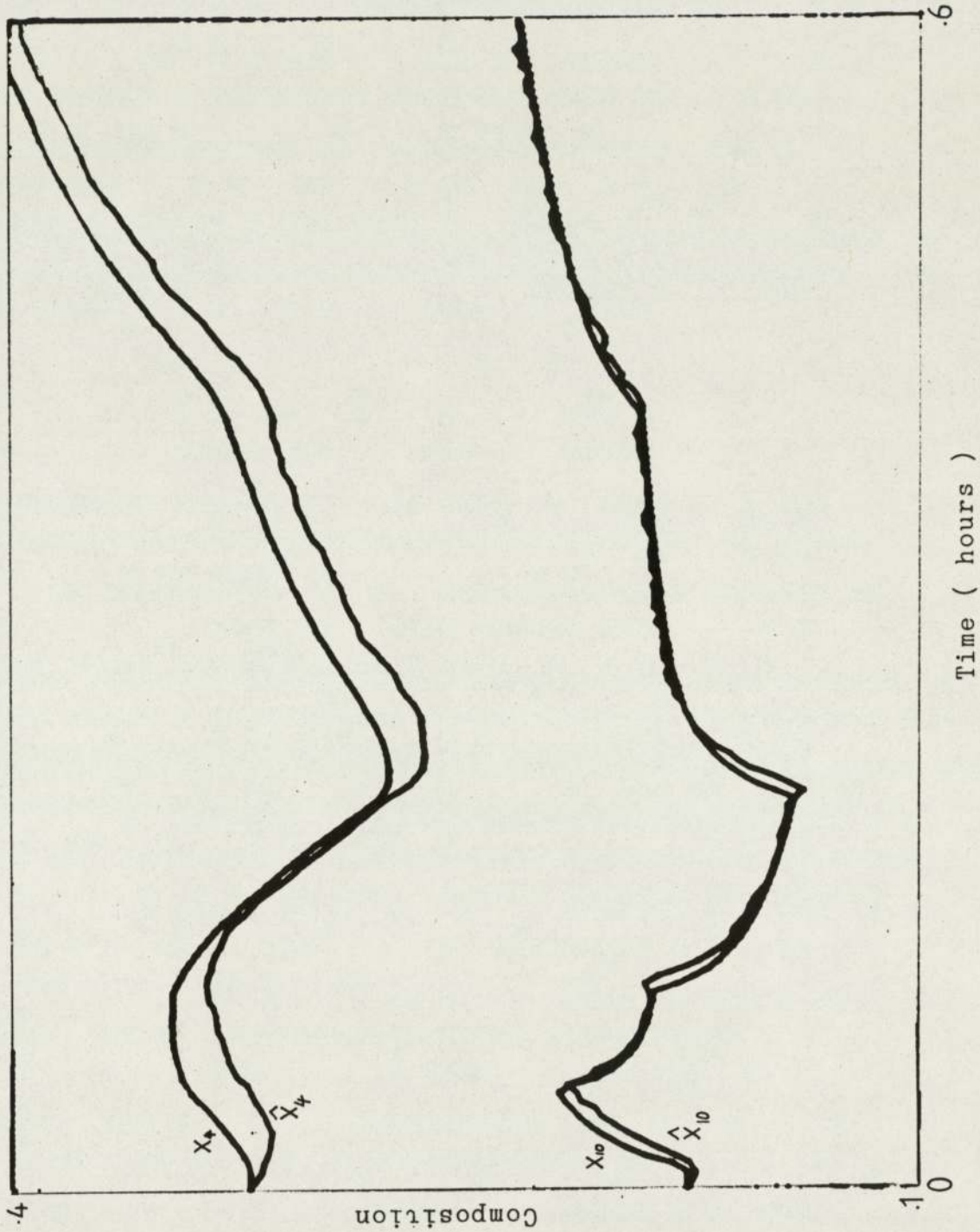


Figure 6.11 Estimation with EKF1: Δt = .004 hour

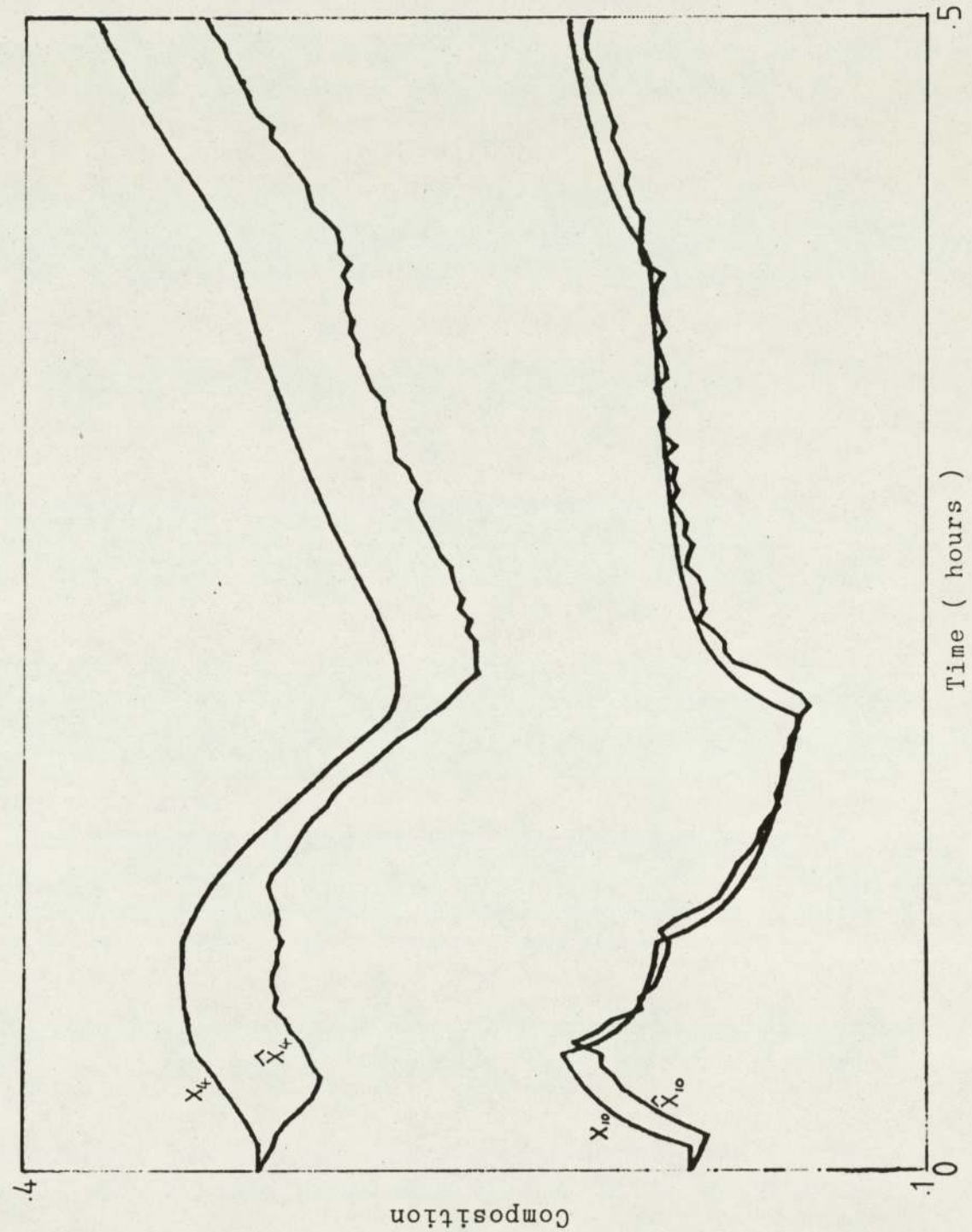


Figure 6.12 Estimation with EKF1: Δt = .005 hour

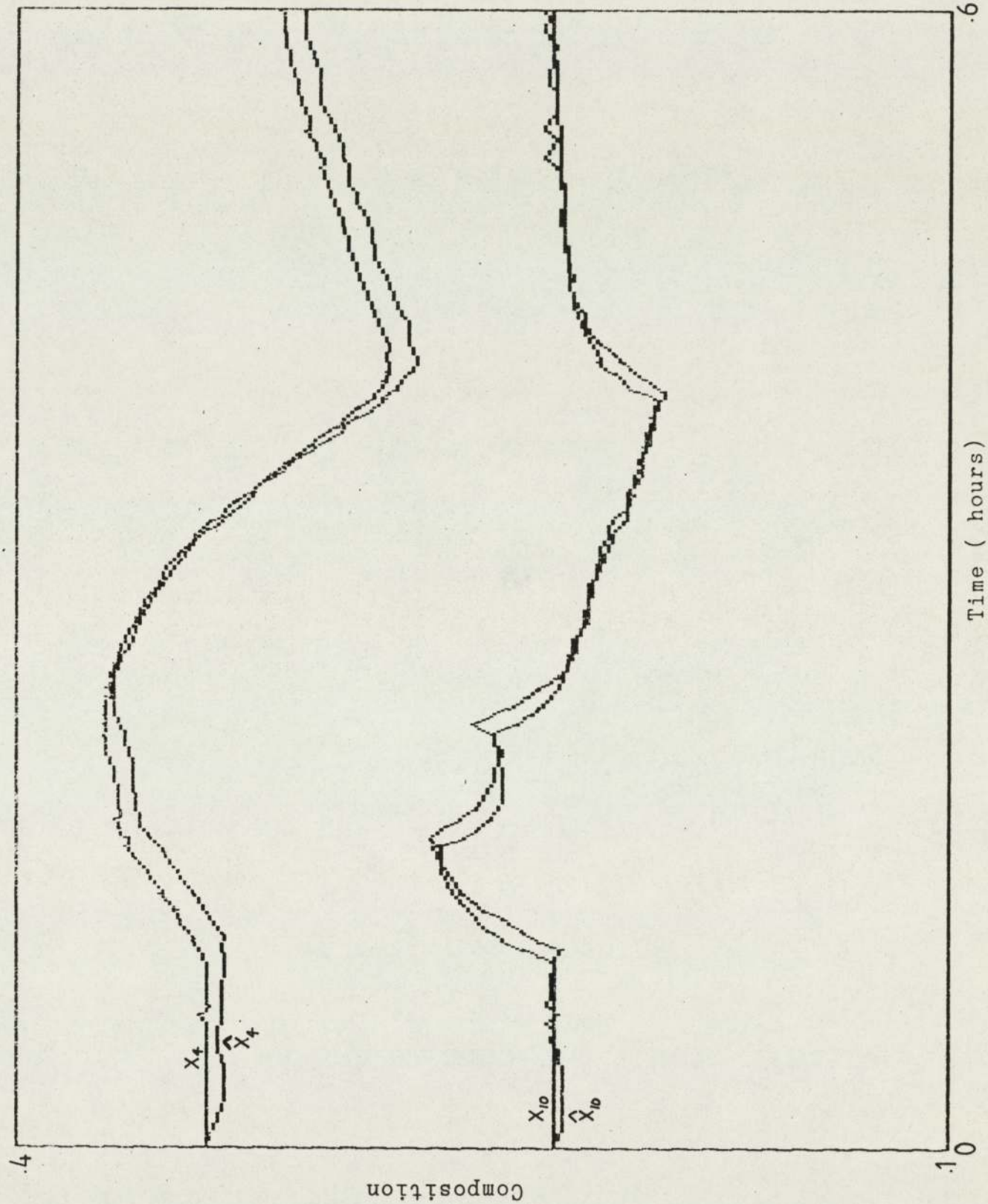


Figure 6.13 EKF1: Models initialised with Model I steady states

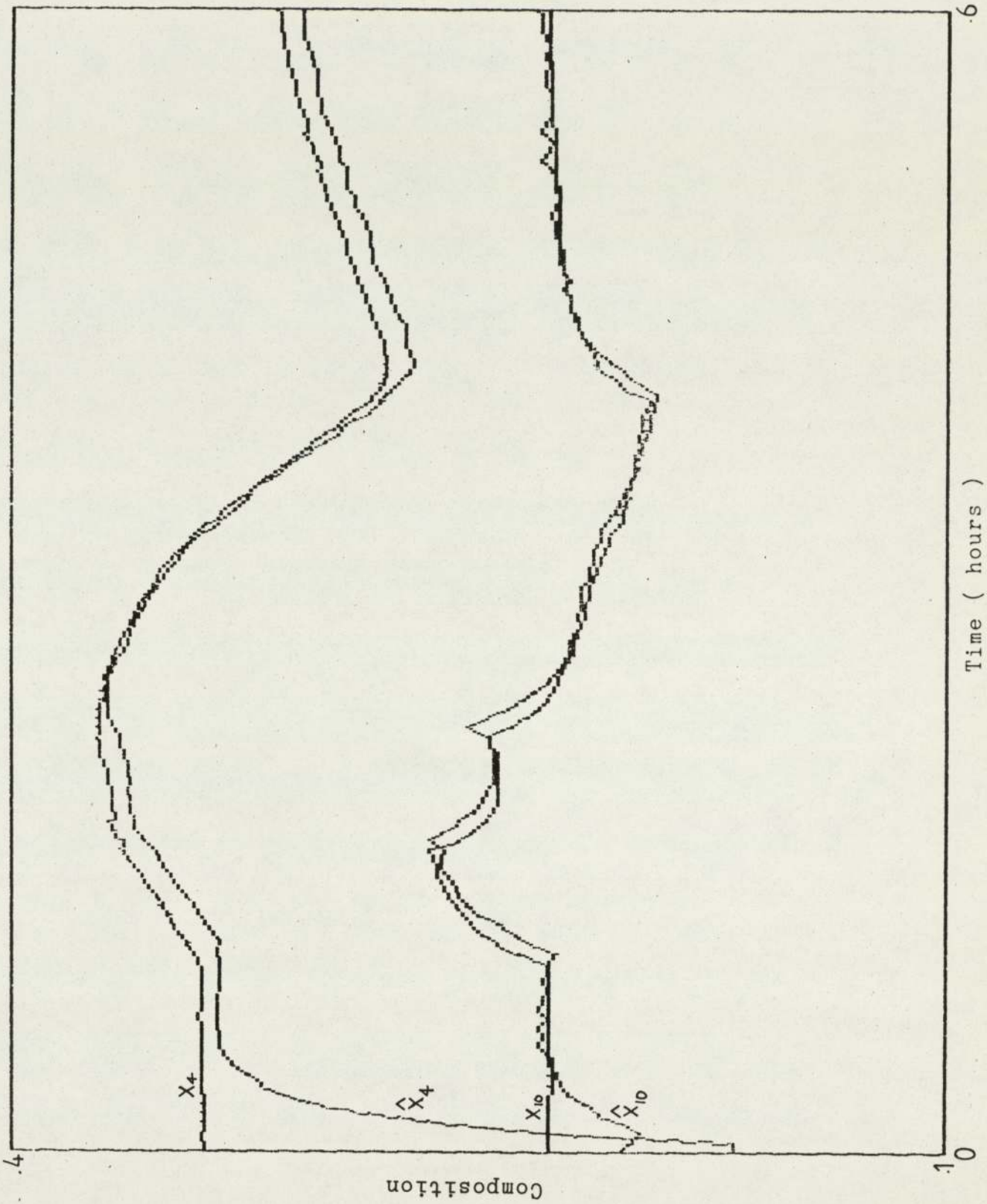


Figure 6.14 EKF1: Models initialised with own steady states

15 plant observations. Examination of the filter gain K(k+1) during this period, shows relatively large values of elements corresponding to the five temperature measurements. This is because for the same measurement noise matrix R, the filter weights the measurements in Figure 6.14 more heavily than the case in Figure 6.13 as model predictions are less reliable during these transient conditions.

6.4.1.8 Estimation of Process Parameters F, x_F, L_R, V

The motive in estimating the four parameters F, x_F, L_R and V is linked both to estimation and control. The calibration experiments in Chapter 5 Section 5.2.1.6 have demonstrated the difficulties associated with the orifice-plate based flow transducers used to measure F and L_R such that flows are determined from valve openings instead. These are therefore pragmatic solutions and it may be possible that the flow values obtained may not be accurate enough. For instance, the values sometimes suffer from mechanical and electrical strictions and the flow for a given valve opening is not calibrated for varying upstream pressure. In the latter case, the reflux rate may not be sensitive since under normal operating conditions, the reflux drum level would be under regulatory contol. However, this is not so for the feed rate. For a complete run, the change in liquid level in the feed tank can vary by as much as a metre and the effect of decreasing head in feed rate entering the column is therefore significant. Estimating F and L_R therefore provides the 'software' alternative.

Varying feed composition x_F is relatively more difficult to track and even more so the boil-up rate, V. The filter not only provides estimates but generates additional benefits in the sense that it helps in understanding the internal dynamics of the column. The other bonus is control. A 'degrees of freedom' analysis for a typical

column operation yields only two manipulative variables to effect the state vector for the system. Normally, L_R and the heat duty q are chosen since they represent convenient energy inputs to the column. Although q is not estimated directly, its effect, V, is. It is therefore possible to envisage some feed forward control applications where the effect of a feed disturbance on L_R and V (hence, q) can be predicted by the filter and the necessary manipulative measures are than made.

For the case of EKF1, with conditions the same as in Figures 6.13 and 6.14 except that the sampling time is .005 hour, the estimates of F, x_F, L_R and V are shown in Figures 6.15 through 6.17. In particular, in Figure 6.17 the estimated boil-up rate $\hat{V}$ is compared with the 'true' boil-up rate and the vapour rate leaving the top plate.

Figure 6.15 shows that the filter is producing reasonably good estimates of F and L_R despite large variations in feed rate. Part of the reason is due to the fact that both flows are also measured. The reflux rate remain constant as the process is under open-loop, steady-state (with respect to reflux and reboiler level) conditions.

Figure 6.16 shows that the estimated feed composition $\hat{x}_F$ is able to track unmeasured variations in x_F but the filter seems unable to eliminate a consistent positive bias.

On the other hand, the estimated boil-up rate $\hat{V}$ is relatively insensitive to changing process conditions. This is shown in Figure 6.17. The filter process model assumes a constant vapour throughput up the column. While the actual variations in the stripping section (as evident in a data print-out) are relatively small, vapour rates above the feed plate appear to be more markedly affected by changes in feed rate. This behaviour is absent in the filter process model and since it is significant, it is suggested as a cause for filter instability especially at larger sampling intervals. In fact, using relatively larger Q elements

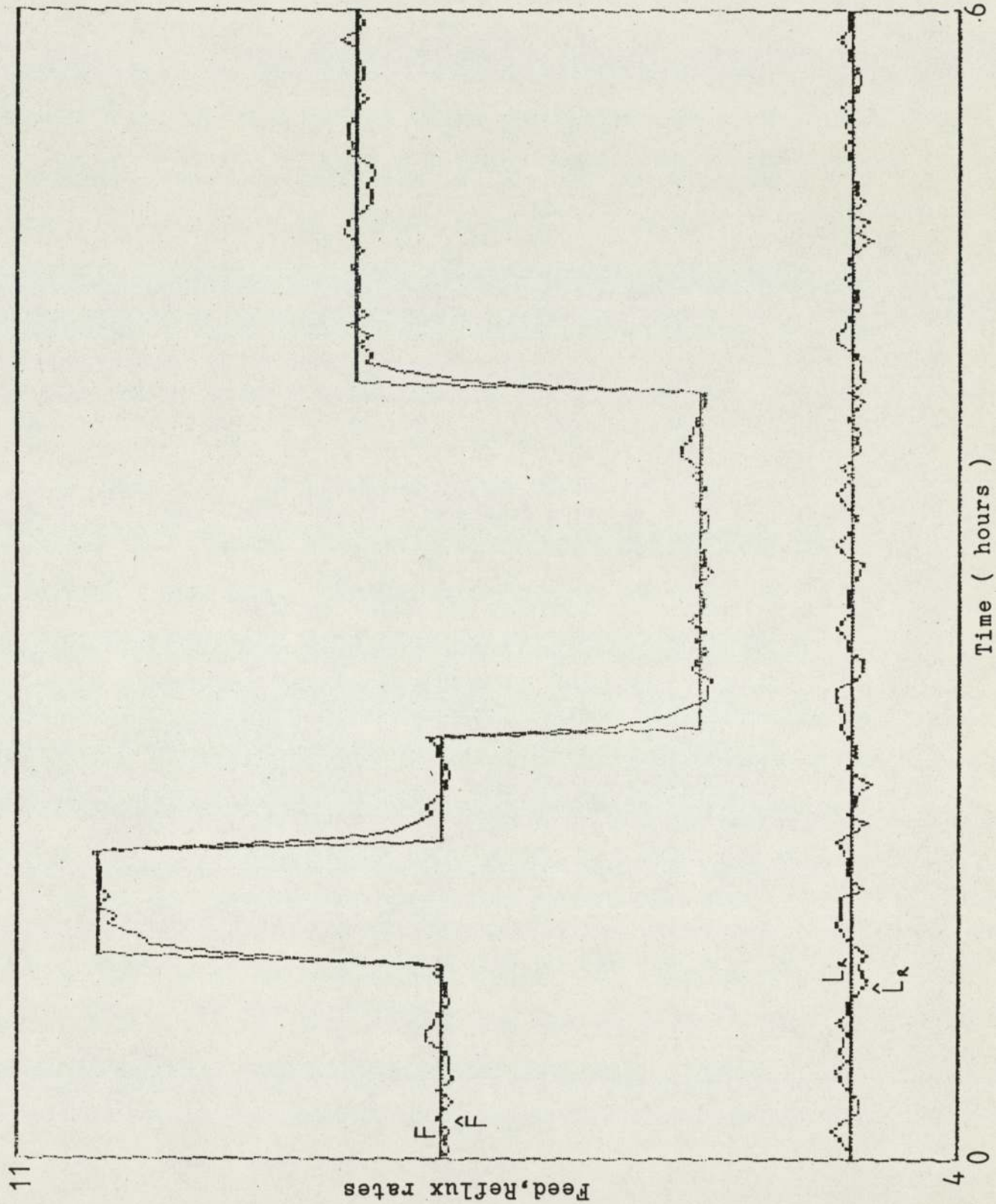


Figure 6.15 Parameter estimation with EKF1: Feed and Reflux rates

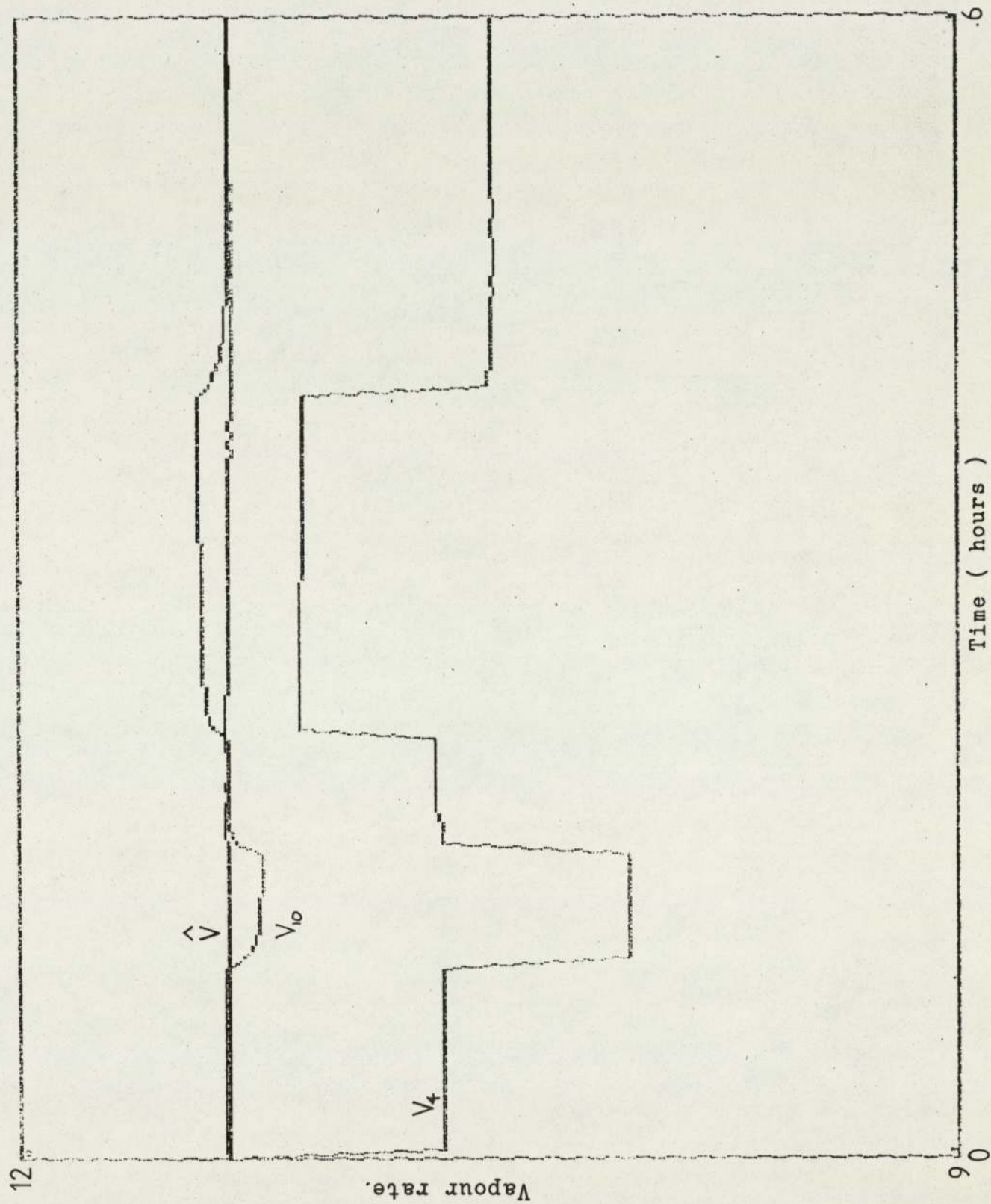


Figure 6.17 Parameter estimation with EKF1: Vapour rates

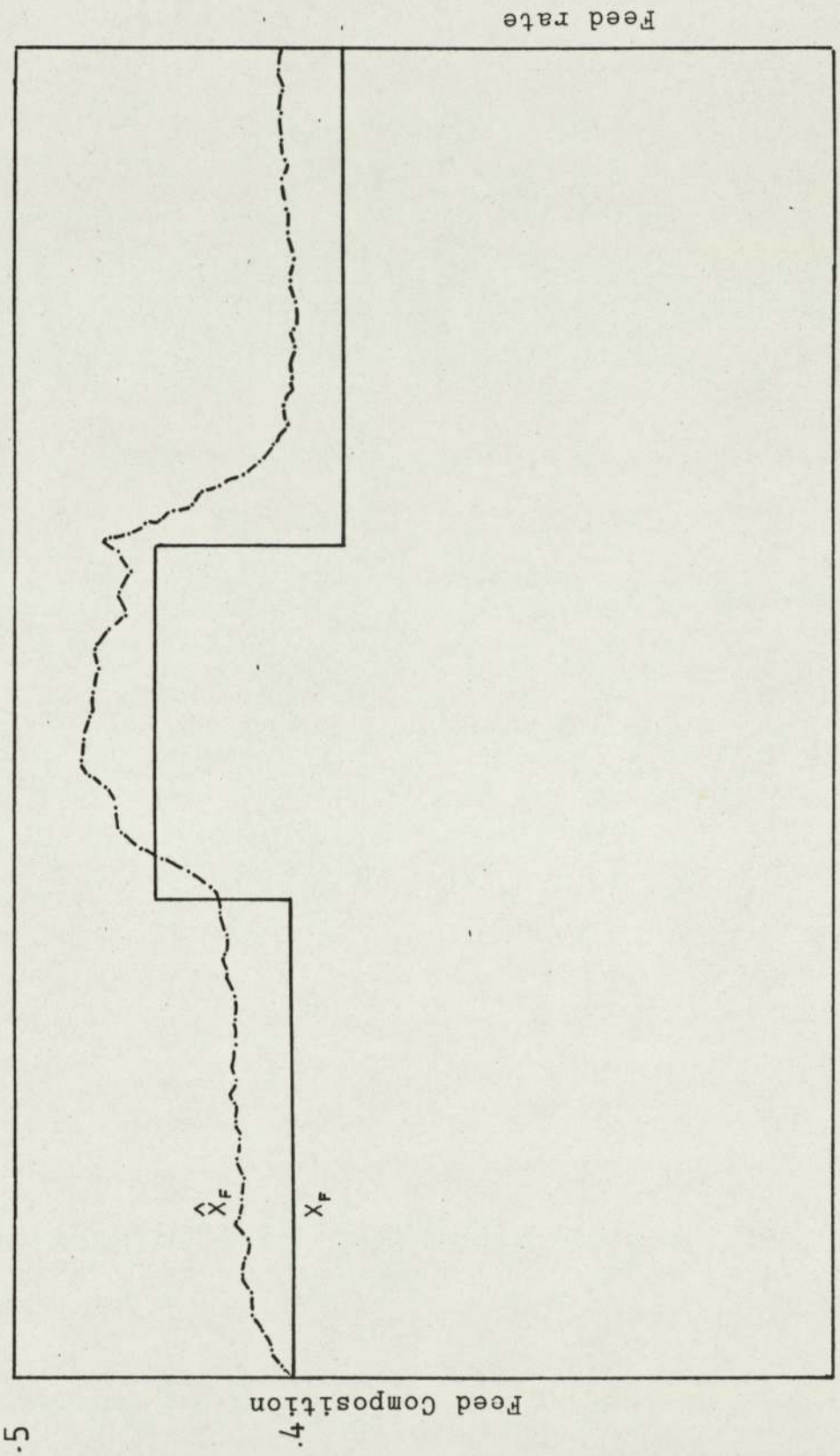


Figure 6.16 Parameter estimation with EKF1: Feed composition

corresponding to the compositions above feed plate also did not seem to improve the performance in any significant way.

6.4.1.9 The BASIC program

The BASIC program used in EKF1 is listed in Table A6.12 of Appendix 6. With a slight modification, the same program can also be used to simulate the dynamic response of process models as described by the flowchart in Figure 6.4.

6.4.2 State Estimator using three tray temperature measurements: EKF2

A more practical filtering problem as a first step toward an on-line application is the estimation of 11 plate liquid compositions using 3 temperature measurements. The memory maps for the four segments (the Graphics segment is the same as for EKF1) are listed in Tables A6.13 to A6.16 of Appendix 6. The feed flow rate and composition, reflux rate, reflux drum and reboiler levels were fixed.

6.4.2.1 Filter initialisation

As in EKF1, the steady-state profiles of Model I are used as the initial estimates. The matrices $P(0,0)_{11x11}$, R_{3x3} and Q_{11x11} are typically as follows:

$P(0,0)$ = diag (.001,.....,.001)

Q = diag (.009,.009,.009,.009,.009,.009,.004,.004, .004,.004,.004)

R = diag (.01,.01,.01)

6.4.2.2 The Measurement Vector and Measurement Noise Matrix

It was decided to spread the temperature measurements through the column. As the feed tray temperature is considered important, the remaining two thermocouples are allocated to the top (plate 1) and

bottom (plate 10) trays respectively.

The measurment vector z_{3x1} is therefore

$$z = (T_1, T_7, T_{10})^T \tag{6.2}$$

and the measurement noise matrix M_{11x3} at time $t = (k+1)\Delta t$ is a matrix with elements zero except for the following:

$$M(1,1) = \frac{\partial T_1}{\partial x_1} \tag{6.3}$$

$$M(2,7) = \frac{\partial T_7}{\partial x_7} \tag{6.4}$$

$$M(3,10) = \frac{\partial T_{10}}{\partial x_{10}} \tag{6.5}$$

As in EKF1, a crude Gaussian noise characterised by the sequence $(0, \sqrt{.01})$ is added to the temperature measurements at each sampling instant.

6.4.2.3 Effect of changing integration methods

The previous simulation results for EKF1 are obtained when both process models are integrated using the simple Euler method with the critical step length of .005 hour or 18 seconds (this figure is critical only for the integration of Model I). The estimation result for EKF2 is shown in Figure 6.18. The first four disturbances are due to feed rate and the fifth which occurs after .4 hour is due to a 12.5% step change in feed composition. In the stripping section, the estimate and the actual values are virtually indistinguishable while above the feed plate the behaviour is reflective of earlier difficulties except that the filter seems to respond better to a disturbance in feed composition.

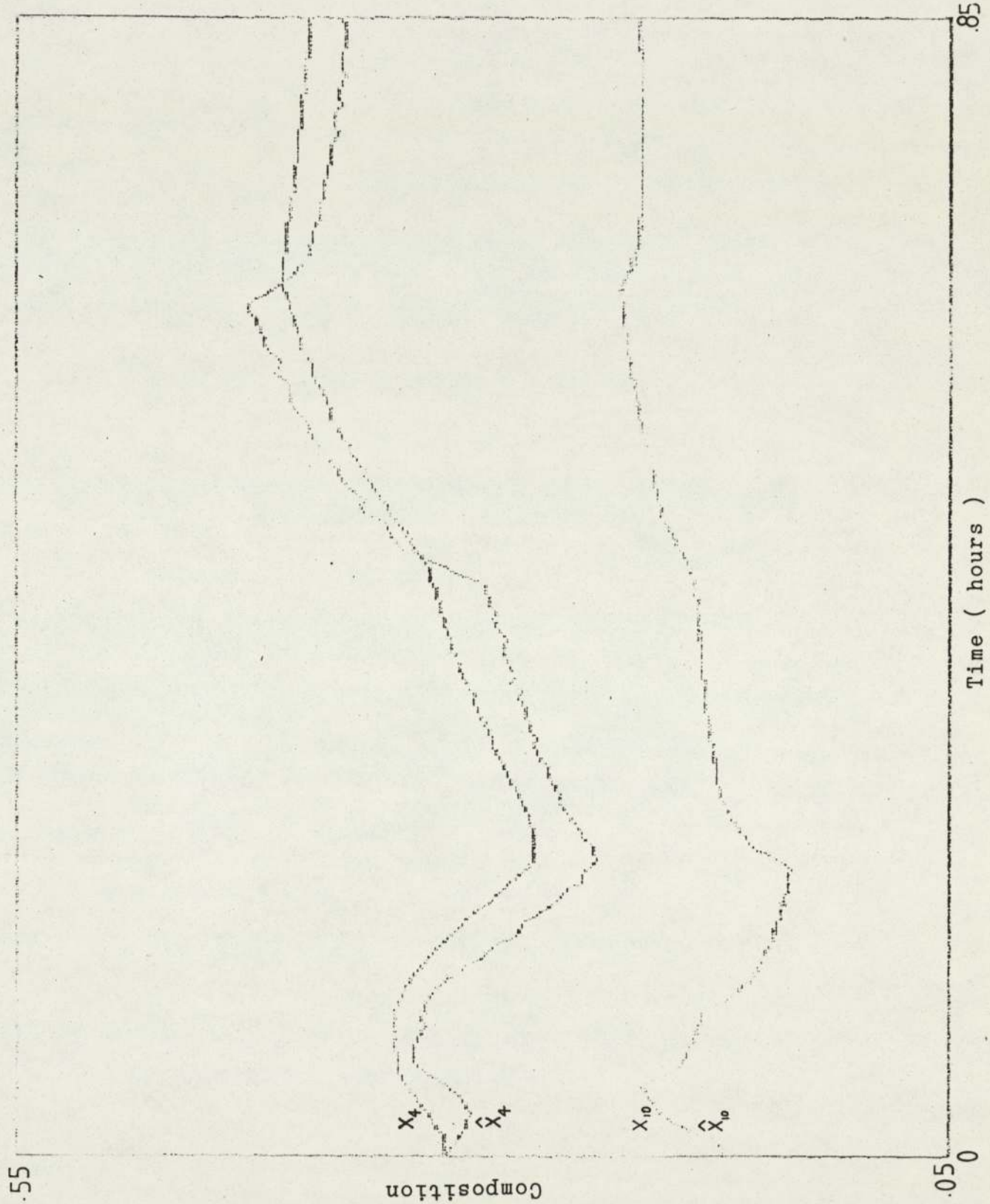


Figure 6.18 Estimation with EKF2: Δt = .0001 hour, Simple Euler

A similar behaviour is indicated in Figures 6.19 and 6.20 where the filter process model is integrated using the modified Euler method with larger step-lengths. While the performance in the stripping section is unchanged, a slight deterioration is indicated by the use of a larger integration step-length, which in Figures 6.19 and 6.20, are 10 and 25 times that used in Figure 6.18.

6.5 Considerations for an On-line Filter: EKF3

The results obtained from the simulation of EKF2 suggest that an on-line version of the filter, EKF3, should be investigated. The various aspects to be looked at include memory utilisation, execution times, filter initialisation and some real-time requirements which would have to be satisfied.

6.5.1 Memory utilisation

The basic memory configuration for the on-line package is similar to that for EKF2 except that the HADIOS Executive is now incorporated. Noisy measurements are made at each process scan. The true state of the plant is now unknown.

A feasible memory utilisation using only two segments is shown schematically in Figure 6.21. The Graphics package, if required, forms the third segment. Since Model I is irrelevant in the memory utilisation of EKF3, the storage requirements are less critical. For instance, the COMMON base have gone up from '34735 to '35413 thus enabling the HADIOS Executive ('27000 through '34762) to be accommodated. In fact, in EKF3 none of the estimation FORTRAN modules needs be core-resident. However, the disk overlay routine DTOC has to be divided into two parts. The main body resides in sector '35 while the rest are loaded into the unused locations of the HADIOS Executive. The memory allocation is detailed

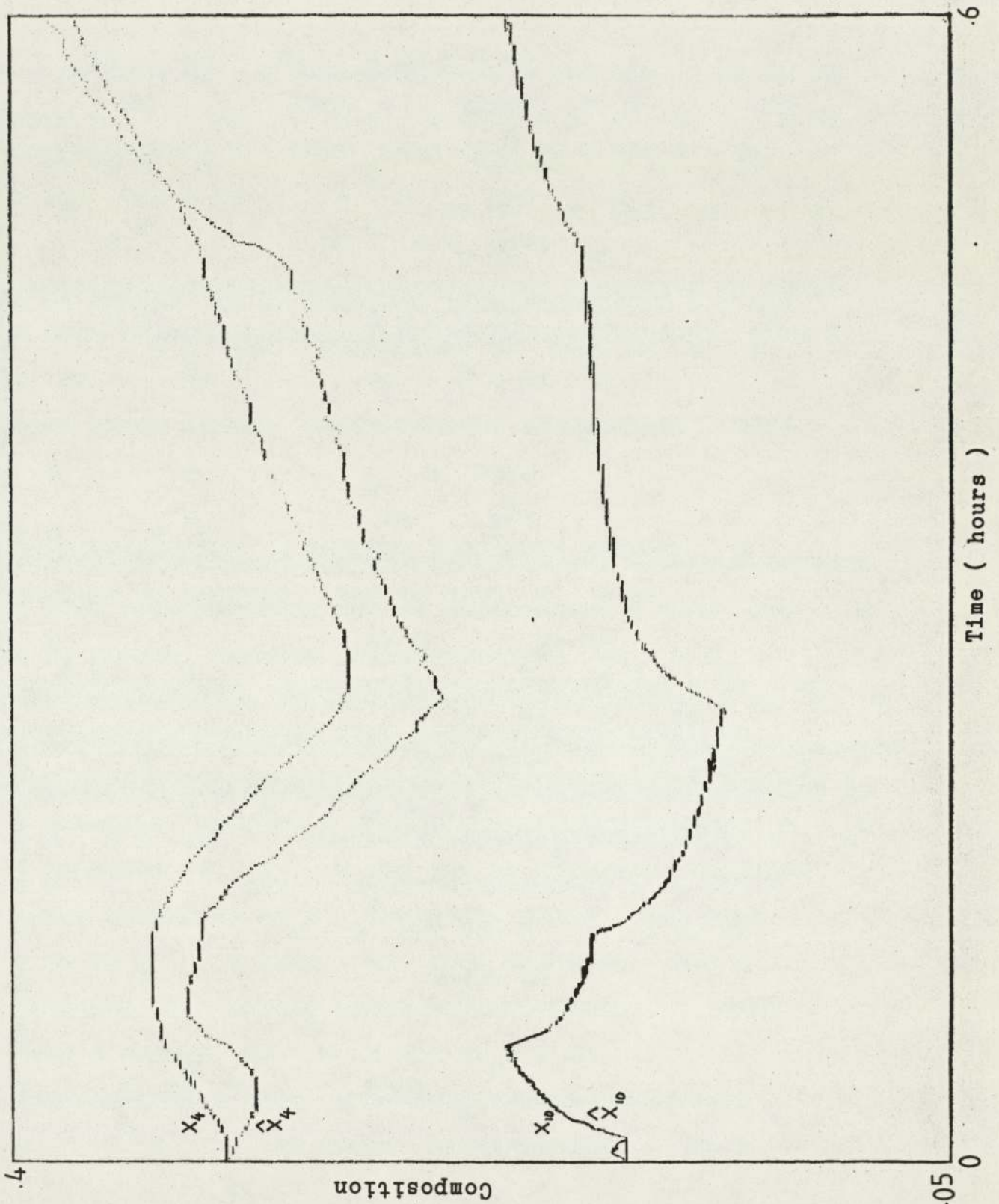


Figure 6.19 Estimation with EKF2: Δt = .001 hour, Modified Euler

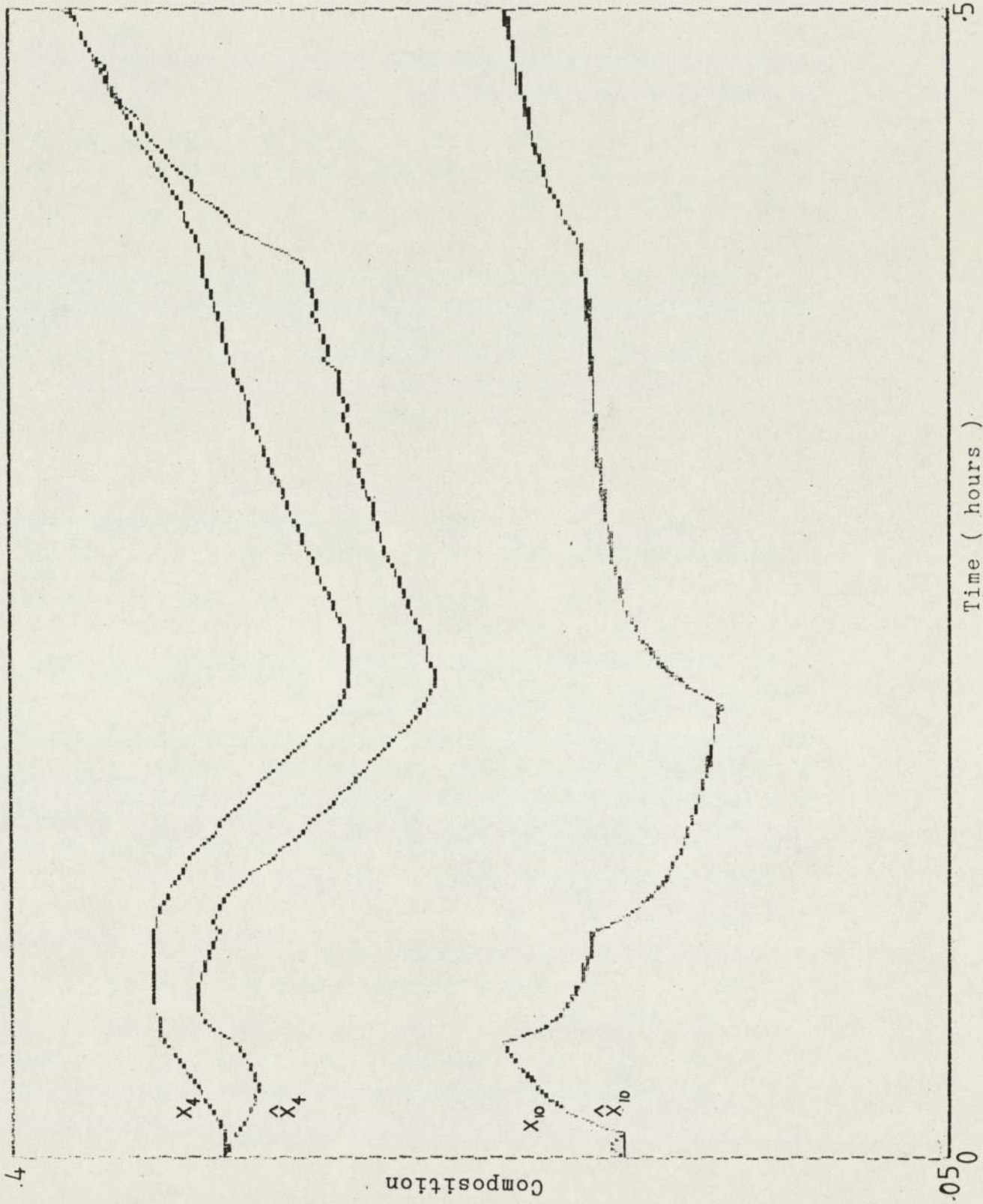


Figure 6.20 Estimation with EKF2: Δt = .0025 hour, Modified Euler

'0	'7160	'20000	'27000	'35413
BASIC	User Space	KOMMON, SIMUL, SIMULX, P10F10, MATMUL, TRANS, MATTPX, DIAADD, SDBUB, SDINT, INIT, INIT3, INTPAS, INTJS	HADIOS EXECUTIVE Rev 03, DTOC, INIT1, INIT2 + MATHS	COMMON variables
"	"	KOMMON, KOMN, COMN, MATMUL, FDTX, KALMAN, KALMA1, MATTPX, MATTPS, DIAADD, MATVEC, MATINV, PKK, DIASUB, MATADD, DIAMUL	"	"

Figure 6.21 Schematic Construction of EKF3

further in the corresponding memory maps found in Tables A6.17 to A6.19 of Appendix 6.

6.5.2 Execution Times

Besides memory requirements, the other vital factor in an on-line implementation is machine execution time. Faster program execution times would permit smaller sampling intervals thereby improving estimation performance. Using the timer routine SU10 in conjunction with EKF2, the execution times required in a typical estimation exercise are shown in Table 6.2 below.

For an on-line application, the total time incurred would be the sum due to 3, 4 or 5, 6, 7, 8, 9 and, if plotting is required, 10 and 11. Further time would also be required for a certain number of I/O tasks.

Table 6.2 H316 Execution times in a Kalman Filtering application

	Function	Time (secs)
1.	Loading Segment 2	3.1
2.	Integration of Model I (per step)	3.8 Simple Euler
3.	Loading Segment 3	3.1
4.	Integration of Model II (per step)	2.3 Simple Euler
5.	Integration of Model II (per step)	6.7 Modified Euler
	Estimation step:	
6.	Subroutine KALMAN (inversion, etc.)	2.2
7.	Refinement of filter process model	2.2
8.	Calculation of P(k,k)	9.8
9.	Calculation of P(k+1,k)	8.1
10.	Loading Segment 4 (Graphics)	2.8
11.	Plotting 4 variables	0.44

Note: 3,4 or 5,6,7,8,9 are necessary for ON-LINE.

The time for steps 3 and 6 to 9 which are relatively constant is about 25 seconds which means the user can only minimise program execution times by a suitable choice of integration procedure.

By trial and error it was found that the critical step length for the integration of the filter process model using the simple Euler method is approximately .0012 hour i.e. about 12 times the critical step-length for Model I. Using the modified Euler method, a step-length of .0025 hour has been found to be acceptable as indicated earlier in Figure 6.20. It can be shown that because the first method requires more integration step lengths (although 3 times faster per step) for a nominal sampling interval, it uses much more computation time and hence becomes a second choice.

The minimum sampling interval for EKF3 is therefore about $(25 + 7\ell)$ seconds where ℓ is a positive integer and 7 is a rounded figure for step 5. Using a step-length of .0025 hour, the meaningful value for ℓ is 12 or greater since the integration time of about $12 \times 7 = 84$ seconds and the 25 second estimation/overlaying step can be accommodated within a sampling interval of $12 \times .0025 \times 3600 = 108$ seconds. A step-length of .003 hour can be used, giving poorer predictions, but ℓ would be reduced to 7 and the sampling interval to about 76 seconds.

Unfortunately, the filter could not cope with such values of Δt, even when $\Delta t = 27$ seconds as shown by Figure 6.22. The estimates start to oscillate after about 15 sampling instants and filtering operation breaks down. The chief reason for instability is attributed to a poor value of Φ as Δt enters directly into its first-order Taylor series truncation. Under the circumstances, EKF3 is therefore unlikely to be practically feasible.

6.6 Conclusions

Basically, this chapter has shown that a major simulation exercise is feasible on the H316 minicomputer although at considerable programming effort and long execution times. The simulation results have shown that while it is possible to design an operational filter for a given process, an on-line implementation is not practical unless further steps are taken to reduce the sampling interval, Δt, or alternatively a better approximation of Φ is found.

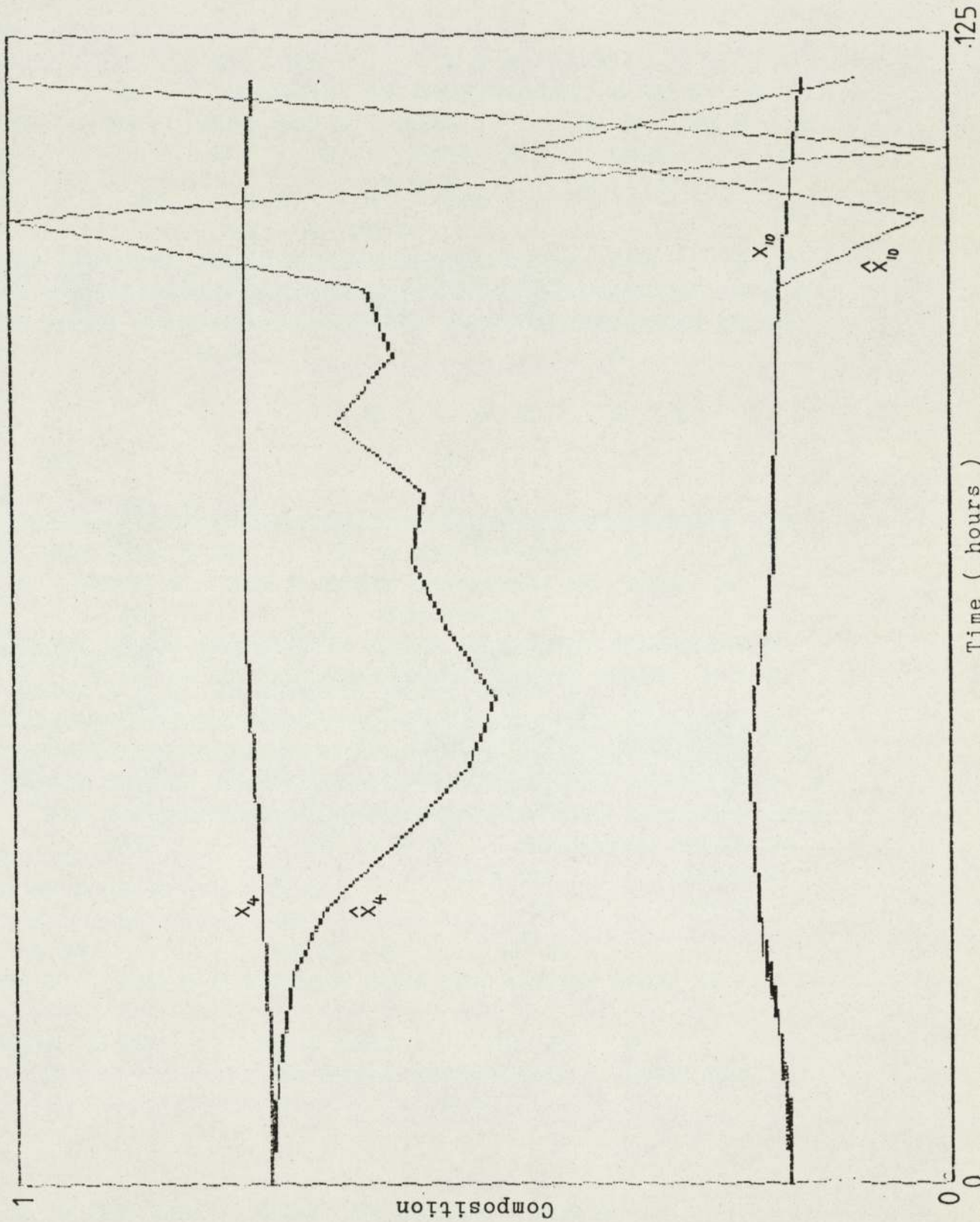


Figure 6.22 Estimation with EKF2: Δt = .0075, Modified Euler

CHAPTER SEVEN

EXPERIMENTAL RESULTS AND DISCUSSIONS

7. EXPERIMENTAL RESULTS AND DISCUSSIONS

The three major aspects of this research emanating from the main and secondary objectives stated in Chapter 1, Section 1.4 will be discussed in the following sections. In view of what has been done these aspects can now be restated as follows:

1. Construction of the linked H316-H6800 twin processor system.
2. Using the software package for the linked-processor system.
3. Kalman filtering feasibility studies.

7.1 Construction of the Linked H316-H6800 Twin processor System

It must be noted that the development of the linked twin processor system has been done within the context of the hardware resources available. It turned out that the extra hardware interfaces required is minimal (a single PIA chip at the micro-end and an Alarm Inputs subinterface for the HADIOS) and with sufficient technical support the data and control links proved to be rather straightforward. However, it also turned out that this solitary dependence on a single PIA has both been a major factor in the logic design of the software protocol and defines the limit of the facilities the linked processor system can offer. The case of data transfers between the computers clearly illustrates these points.

7.1.1 Constraints due to a single PIA chip

During system development, it was desired that the H316 should be able to send data immediately to the M6800 even if the latter is on-line. The purpose may be to alert the microcomputer to undertake

some immediate and specific computing task. To achieve this, the PIA A Side would have to be configured in the interrupt mode so that the H316 may interrupt the M6800 via a CA1 active transition. The micro may or may not be required to acknowledge the data transfer accompanying this interrupt. If it does, then the CA2 is likely to be used. However, since the micro is in on-line mode, a split second situation can arise whereby the mini sends a CA1 just before the micro sends out its first CA2 to interrupt the mini for a process scan (a frequency which is independent of and unknown to the mini). The mini is therefore interrupted on the wrong footing so to speak and the protocol may never recover. For this reason, the PIA A Side has been made CA1 interruptible in the off-line mode only. The M6800 still retains the ability to interrupt and send data to the H316 via the PIA B Side output port. Thus, when the M6800 is on-line, the action of CALL (4,I,M,D(0)) in the H316 is only to place the data {M,D(0),D(1),..., D(M-1)} in a temporary buffer for M6800 collection at its next process scan. The effect of the data D(0),D(1),...., etc. on the computation in the M6800 is therefore felt at the most, one M6800 sampling interval later.

Clearly, a second PIA chip removes some of these limitations. In fact, it can be dedicated to process alarm signals and I/O parallel data transfers between the M6800 and the H316 or any other piece of equipment capable of doing so. The M6800 microcomputer system must provide a Port address for it while at the H316 and, unused Alarm inputs interrupt lines and DGOB output pins are readily available. The software effort in either executive would include extra interrupt polling steps and their corresponding response codes and no major revamping of existing logical paths is necessary.

7.1.2 Alternative Designs in linking the two Processors

During system design, alternative approaches to provide a linked twin processor have been considered. Three technically feasible designs can be identified. For the sake of discussion, they will be called Method 1, 2 and 3. The first two are RS232-based and are therefore serial data methods and the third is a completely parallel data technique.

7.1.2.1 Serial Data Methods

The first alternative design is schematically shown in Figure 7.1.

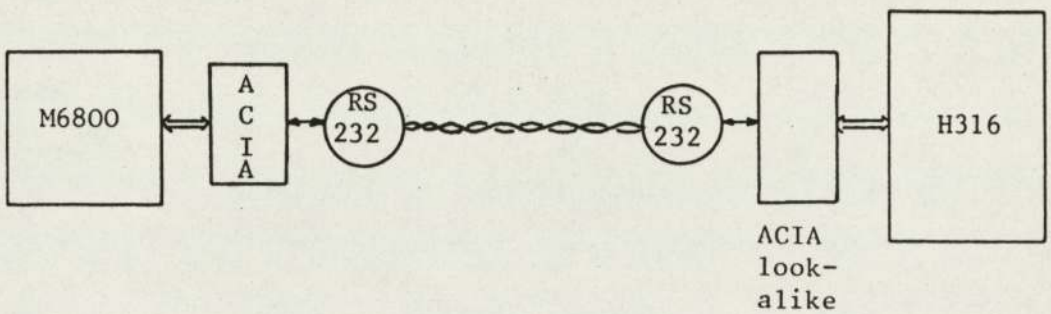


Figure 7.1 A Serial Data Link for the H316-M6800 system: Method 1

An ACIA chip is interfaced to the M6800 data and address busses and basically converts 8-bit data bytes into a serial format suitable for an RS232 output port. At the H316 end, another RS232 will receive the transmitted data and channel it to an ACIA look-alike which is interfaced to the minicomputer system bus. This ACIA look-alike needs to be developed in-house as the logic involved must be upgraded,

acceptable and unambiguous for the H316 digital circuitry which is basically DTL-based (Diode-Transistor Logic).

Although the RS232 links provide a simple approach in the sense that it only involves three wires (Transmit, Receive and Common) and the standard hardware usually accepts only 'true' signals, the method is inherently slow. The advantage of being able to transmit and receive ASCII data (eight bits with perhaps one or two start/stop bits), thus permitting the transfer of floating point numbers and strings, programs, etc., is off-set by the need to develop sufficient software modules to check for incorrect data. This is because the serial link does not check the contents of the data to be transferred.

Avoiding the use of the ACIA look-alike at the H316 end leads to Method 2 schematically shown in Figure 7.2. Here, one avoids the extra logic design but is forced to use the only RS232 port available on the H316 which is currently used with the Newbury 8005 terminal. As far as the H316 is concerned the M6800 is now just another (complicated, intelligent) VDU.

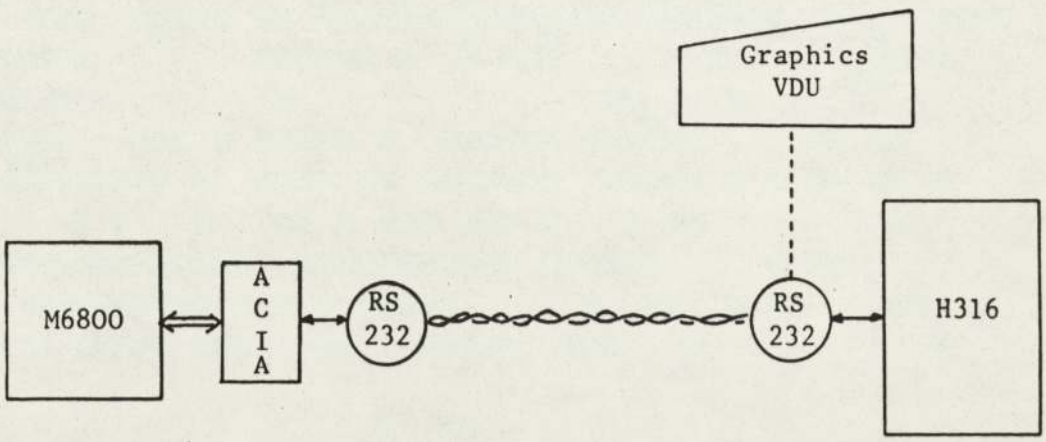


Figure 7.2 A Serial Data link for the H316-M6800 system: Method 2

7.1.2.2 Parallel Data Link

As an alternative to the parallel data link used in this research, Method 3, schematically shown in Figure 7.3, avoids the use of the HADIOS system altogether.

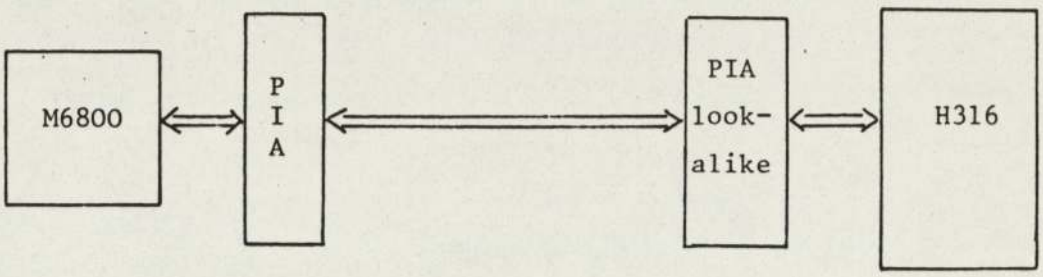


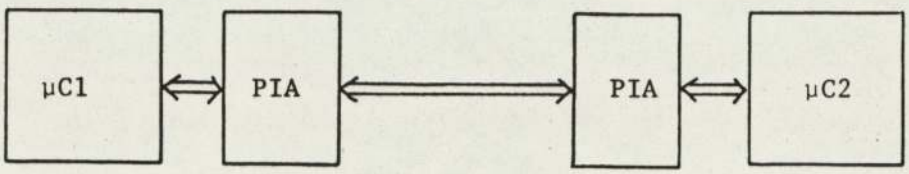
Figure 7.3 A Parallel Data Link for the H316-M6800 system: Method 3

This method requires the in-house development of a PIA look-alike at the H316 end. The advantage of fast data transfer rates and avoiding the HADIOS which was not originally designed for processor to processor communication of this sort, are again off-set by the need for a considerable and careful digital design to cope with the different (and older) DTL technology of the H316. In addition, control signals specific to the H316 data and address bus, and the necessary low-level software support would have to be provided.

Summary

The preceding discussions have shown that the method used in this research is a close second to the 'best' technique of Method 3.

The extra logic design is minimum and data transfer rates are reasonable. Of course, as Figure 7.4 shows, such links for a microcomputer to microcomputer system is relatively straightforward. Basically, two



µC = Microcomputer

Figure 7.4 A Microcomputer to Microcomputer Link

PIAs are required. Each microcomputer is merely another I/O device to the other.

7.1.3 The M6800 interfaced directly to a Process Plant

The present M6800 microcomputer system can be configured to be interfaced directly to a process plant via family compatible ADC/DAC chips. In fact, a limited study[90,91] has shown that it is practically feasible. In the context of this research, the M6800 system could have been interfaced directly to any of the process plant signal conditioning units via an analogue card. Figure 7.5 shows a diagrammatic view of two possible arrangements.

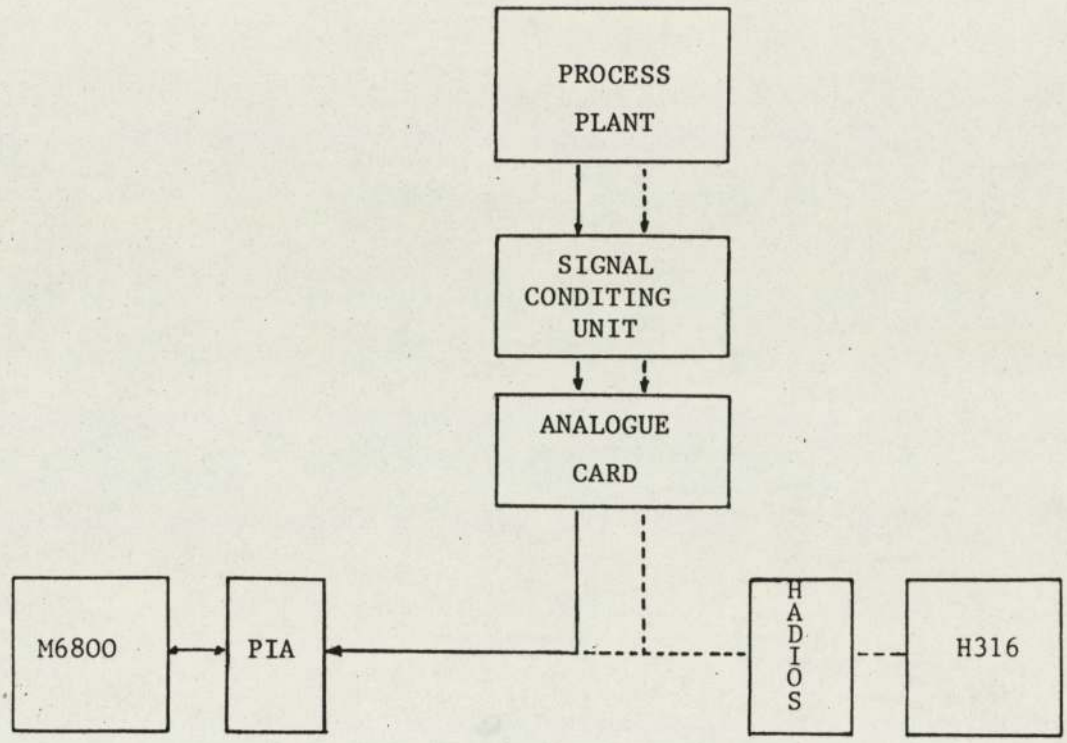


Figure 7.5 The M6800 system interfaced directly to a Process Plant

The dark line configuration is closer to industrial practice as it removes microprocessor dependence on a host minicomputer for its data acquisition and control operations. The function of the analogue card is to prepare the plant analogue inputs for microprocessor (and ADC) compatibility and channels microprocessor digital or analogue control outputs to the plant. The ADC typically can be an 8-bit ADCS-80 or its 12-bit version obtainable at a very reasonable price.

The dotted line configuration adds more flexibility and distributed control power on top of the existing twin processor facility. A separate PIA would have to be provided but much more powerful control schemes can be designed if future development of this research is to blossom in this direction.

7.1.4 Noise and Spurious Interrupts

Besides having to cope with new methods of working (requiring

familiarity with microprocessor technology), the system designer is also faced with signal interferences which may consume many man-hours of program development effort. A particular problem that plagued the linked processor system developed from time to time is that created by spurious interrupts generated by CA1 or CB1 control inputs to the PIA.

Low-active CB1 transitions are less problematic since the PIA B Side is an output port and data is normally sent out of the M6800. The PIA A Side is quite a different story. As Figure 3.12 in Chapter 3 shows, bits 1 and 2 (CB1 and CA1 lines respectively) and bits 9 to 16 of the DGOB are used for data transfer operations. It was observed (with the aid of a CRT oscilloscope and a digital logic analyser) that occasionally the CA1 input line is energised, generating spurious spikes, when certain data patterns appear on the data lines (bits 9 to 16). The downward edge of this spurious pulse upsets the program logic in the M6800 Executive by setting the IRQA1 flag prematurely if the PIA A Side is in the normal handshake mode and creates a spurious $\overline{\text{IRQ}}$ interrupt request otherwise (M6800 in off-line mode).

The exact cause of the problem was never identified but a temporary solution by providing a separate screened cable for the CA1 control input line seems to eliminate the problem. Possible causes may include signal mismatch due to improper impedance at either end, unfiltered voltage spikes from the DGOB or even faulty chips. Such 'random assaults' on the system integrity can be monitored by running special test or debugging programs from time to time especially before a major on-line exercise.

Other disturbances are more predictable. They are mainly noise generated from switching mains power supplies for the HADIOS and its digital output drivers (switches marked 1 and 2 in Plate 3.1 of Chapter 3). These phenomena may be pathological to the power supply

and noise rejection designs of the existing hardware but it is not a serious problem. As suggested in the operating manual[94], it is best to switch on all relevant power supplies first before loading the system and application software.

7.2 Using the Software Package for the Linked processor Facility

In using the linked H316-M6800 twin processor system, the user must realise that the M6800 is a linked computer, completely dependent on H316 machine cycles for its process data acquisition and control. This may be seen as an inherent disadvantage as in practical industrial situations, one would wish the microcomputer to be as autonomous as possible - both in the hardware and software sense of the word. The linked-processor facility developed in this work is therefore unique in the sense that it is a pragmatic solution in optimising the human and material resources available.

7.2.1 Development of and Running User Application Programs

The detailed steps involved in program development have been described by way of examples in the user's manual of Reference 94. Only the important operational points will be mentioned here.

Basically, the user enters the HADIOS Executive Rev 03 package at location '27000. The Executive initialises the HADIOS for M6800 CA2/CB2 interrupts and starts the H316 Real-Time Clock (location '61) in the non-interrupt mode before jumping to location '1000 for normal BASIC interpreter initialisation. Location '61 which is incremented once every 20 ms is used as a general-purpose real-time base. If the user strictly does not want to allow M6800 interrupts, then he should enter the package at location '1000 which is the normal entry point of the Interpreter.

H316

In the H316, the user may write programs in BASIC with FORTRAN and DAP-16 MOD2 Assembly Language subroutines. The disk overlay facility can be used if core storage is insufficient. If he does not wish to use the on-line graphics package, then before entering the package, the High Octal Address (HOA) for the BASIC interpreter (location '7367) should be patched with '26777, thus providing him with an extra seven sectors of core space (the graphics and the Executive are self-contained segments).

Note that CALL (3,N,U) and CALL (4,I,M,D(0)) are independent of CALL (1,A(0),B(0)) and CALL (2) which are mutually related. This allows the user to conduct control valves calibrations in situ or transfer data to the M6800 in off-line distributed processing.

In on-line work, sampling intervals can be kept small by reducing inter-scan BASIC processing to a minimum. However, developing FORTRAN subroutines may take hours as there is no operating system as such for the H316. Program debugging is normally done by switching the H316 into single-instruction (SI) or Memory Access Mode (MA) and examining bit patterns in individual memory locations. The HALT instruction in location '31046 of the Executive, though undesirable in real-time software, is deliberately included for detecting a fatal error. H316 execution also halts if its instruction decoder fails to interpret a given bit pattern as a recognisable operation code. The most common execution errors arising out of using FORTRAN and DAP-16 MOD-2 subroutines include those caused by improper allocation of base when loading in the desectorised mode, and improper use of the index register and indirect addressing mode.

M6800

Although a combination of SD BASIC, the M6800 Executive, additional and disk-based file I/O facilities provides a wide range of on-line applications, the major constraint is still speed, and a lack of general purpose mathematical routines. The SD BASIC could not be modified in (Honeywell) BASIC sense, and its mathematical package is non-reentrant (as is BASIC MTH-PAK) and inaccessible from user assembly language subroutines. If FORTRAN is supported then it may be possible to emulate the Tektronix Graphics routines for use with the Newbury terminal. With a modified RS232, the baud rate can be increased from the present 300 to a maximum of 9600.

In the M6800, CALL SUB3(N,U) and CALL SUB4(M,D(0)) are normally used between CALL SUB1(A(0),B(0)) and CALL SUB2. If the user wants to use only CALL SUB3(N,U) and/or CALL SUB4(M,D(0)) then the steering flag M680 in location '27001 of the HADIOS Executive should be manually set to non-zero before initialising the package.

Also, in a normal M6800 operation (off- or on-line mode) the reset ($\overline{\text{RES}}$) button may be used to stop program execution provided the first CB2 interrupt (via the first CALL SUB1(A(0),B(0)) has not been sent out else the NMI button should always be used. This is because a hardware reset also clears all PIA registers and the M6800 Executive would not be able to re-initialise the microprocessor response code in the HADIOS Executive.

7.2.2 Use of Counters - Some Operational Constraints

As indicated in Chapter 4, Section 4.4.3, a counter may be used by a H316 on a M6800 user. The modified counter handling code does give better estimates of flowrates provided certain precautions are taken.

A counter, driven by a turbine flowmeter or otherwise, has two operating modes. In the non-interrupt mode, the user must ensure that his maximum flowmeter output pulse rate does not exceed 255 within a sampling interval (the user can minimise this by presetting the counter to zero at each sampling instant). Failing to do this would lead to incorrect readings of counter contents during scanning time since the 8-bit counter register overflows to zero.

The situation is slightly more involved when the counter is used in the interrupt mode. Consider the schematic timing diagram of Figure 7.6 showing instances of a counter and computer scanning interrupts. Clearing the last counter interrupt time, ITM_N, is the best estimate, from

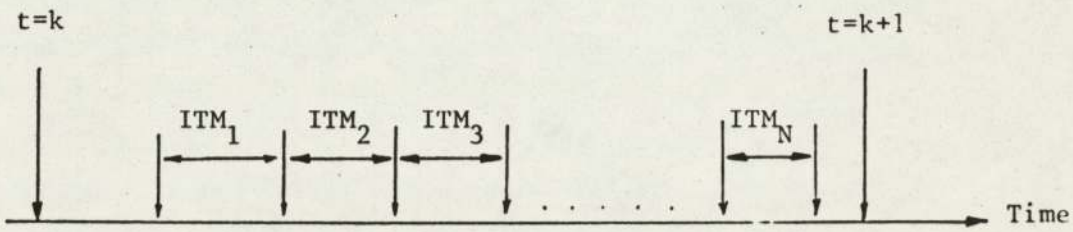


Figure 7.6 A Timing Diagram for Counter and Computer Scanning Interrupts

the set of $\{ITM_j, j=1,N\}$, of instantaneous flowrate at time k+1 if ITM_N does not include significant time delay which may arise if the counter tries to interrupt during system interrupt inhibited. In fact, significant error is embedded in ITM_N if it includes the 20 ms delay when a computer is sampling analogue inputs. This is a plausible situation

as for example in the double-effect evaporator exercise (see Section 7.2.3.2) where the maximum flowrate used corresponds to about 200 Hz (4 counter increments per 20 ms) the error for an interrupt time of one second (corresponding to a counter preset value of (127-50) = 77) is about 8%. Since the preset value is fixed for a given run, the percentage decreases with decreasing flowrate but the preset value must not be so low that the number of counter interrupts becomes less than two. The counter flowrate formula 1 in Section 4.4.3 becomes meaningless in this situation. The user can minimise these difficulties through suitable choices of counter preset values for a known flow operating range.

In addition, the values of counter interrupt times B(48), B(51) and B(54) are meaningless (they are zeroed in the case of the micro) at the first process scan (time zero) and should be ignored.

The preceding discussions suggest that using the counters in the non-interrupt mode with a preset value of zero is more preferable. As a general rule in real-time software design, the system should not be burdened by handling extra interrupts unless it is absolutely necessary. A multiple interrupt situation is always a potential source of erros, fatal or otherwise.

The counter experiments also indicates the disadvantage of going digital in this specific case. A better counter set up would contain in-built logic (e.g. an embedded micro) to calculate the counter inputs and then merely presents them for collection during a process scan. In this way, the computer saves valuable processor time and system reliability improved.

7.2.3 On-line Demonstration Experiments

The litmus test for a real-time data acquisition and control software is actually to use it on-line to a process plant. The linked

processor facility has been used on the IBM distillation column and the double-effect evaporator. The experience is summarised below.

7.2.3.1 Microprocessor Control of Reboiler and Reflux Drum Hold-ups in the IBM Distillation Column

The experiment described here was chosen because it demonstrates a suitable role of the M6800 microcomputer in using the linked processor system for a possible open-loop, on-line estimation exercise.

Basically, the H316 is placed in the OFF-LINE mode, i.e. A(12) = 1, while two P or P+I controllers are set up in the M6800 which also scans the plant (Analogue Inputs channels 36 through 47) every one or two seconds. Because A(12) = 1 in the H316 BASIC program, process information is available at each M6800 scan. The H316 can then use this information for further data processing or plotting.

The SD BASIC and BASIC programs required for the exercise are listed in Tables A7.1 and A7.2 of Appendix 7. Note that the SD BASIC program sends the two M6800 controller outputs [D(0),D(1) in SD BASIC, into B(57) and B(58) respectively in the H316] during every sampling interval. Also, at a specified time, the H316 program places a non-zero value into B(57) in SD BASIC to instruct the M6800 to affect a step change in the feed valve to the column.

The column operation is first started using the procedure described in Chapter 5, Section 5.2.1.8. After initialising the software package, the simple BASIC program below

```
10 INPUT N,V
20 U = 32767*V/10
30 CALL (3,N,U)
40 GOTO 10
```

can be used to bring the column approximately to steady state. The various valve readings (N=1 to 5) are noted and can be used as initial valve settings in the SD BASIC program thus allowing a bumpless transfer.

The arrangement allows the evaluation of M6800 controller performance by plotting control and manipulative variables on the H316 VDU. To do this, the value of A(1) in the H316 program must be set to 1 so that the plotting interval is also the M6800 scanning interval.

Figure 7.7 shows such a plot where the M6800 controller outputs are indicated by the two histograms. The controllers, though needful of proper tuning, seem able to keep the two levels reasonably constant. The reboiler level in particular is considerably noisy due to the boiling liquid. The noise in the reflux drum level is mainly due to condensed vapour droplets which fall directly on to the liquid surface. The spurious dots and occasionally missing segment are due to, presently, unsolved difficulties when using the Newbury 8005 VDU in an interrupt environment. Apparently, the VDU 'loses' a character, or plots a different or several different ones instead, if the H316 is interrupted (by the M6800 or a counter) at the moment when it is about to output a character to the VDU. The plotting action is initiated by the first CA2 interrupt in a process scan. Since during each sampling interval the M6800 interrupts the H316 a total of 42 times (28 for analogue inputs, 8 for control outputs and 6 for M6800 to H316 data transfer), it is not possible to pin-point which interrupts are responsible.

This phenomena also causes editing problems in BASIC when the micro is connected on-line to the plant. A machine code patch for the BASIC IOS-D module has been made in an attempt to recover a possibly 'lost' character during an input mode but outstanding problems remain.

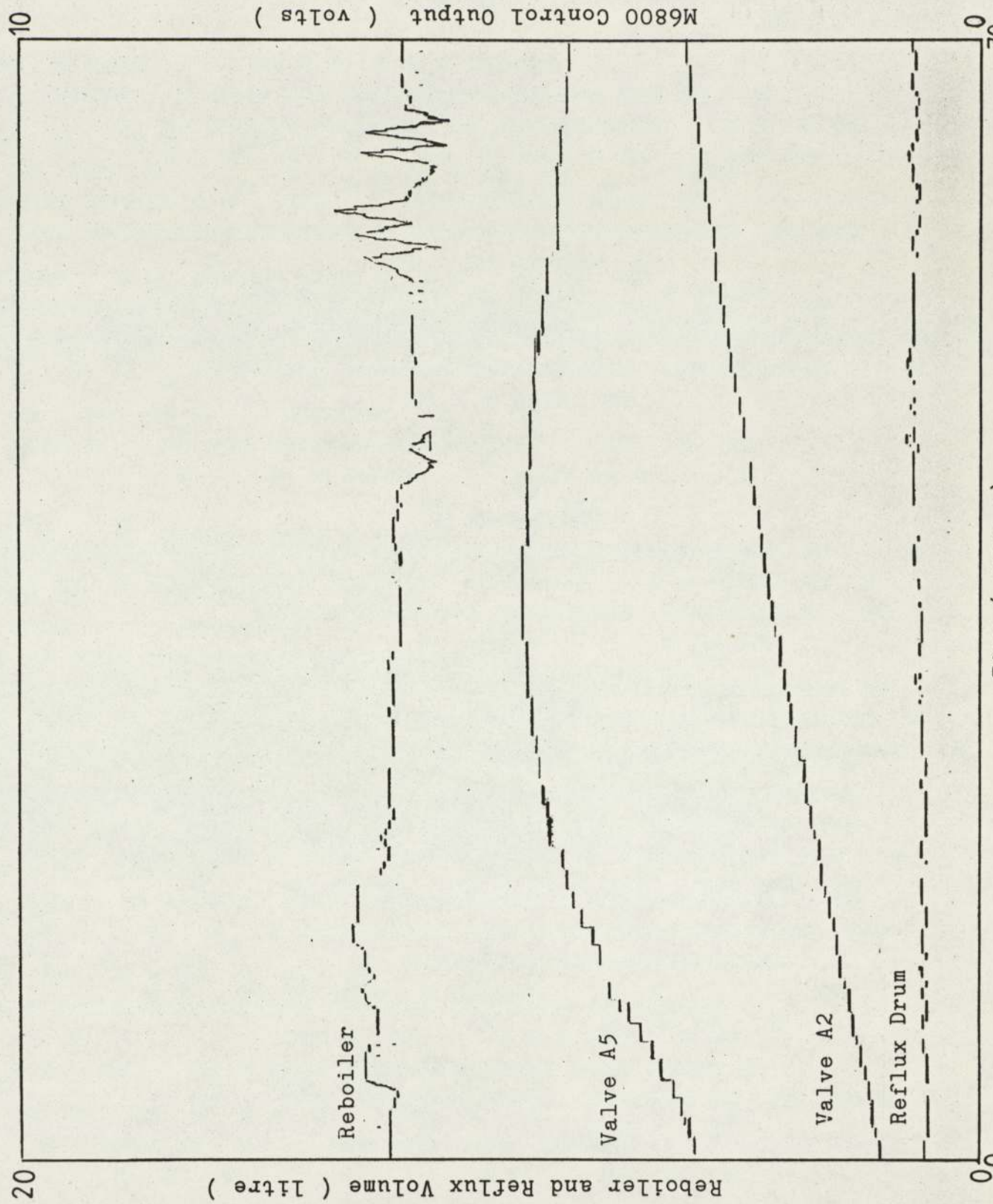


Figure 7.7 M6800 Control of Reboiler and Reflux Drum Levels

The patch, located in sector zero, is described in Table A7.3 of Appendix 7.

When the column is kept under steady-state conditions by microprocessor control action and the column subjected to an approximate step disturbance in feed rate, the response of the three tray temperatures are shown in Figure 7.8.

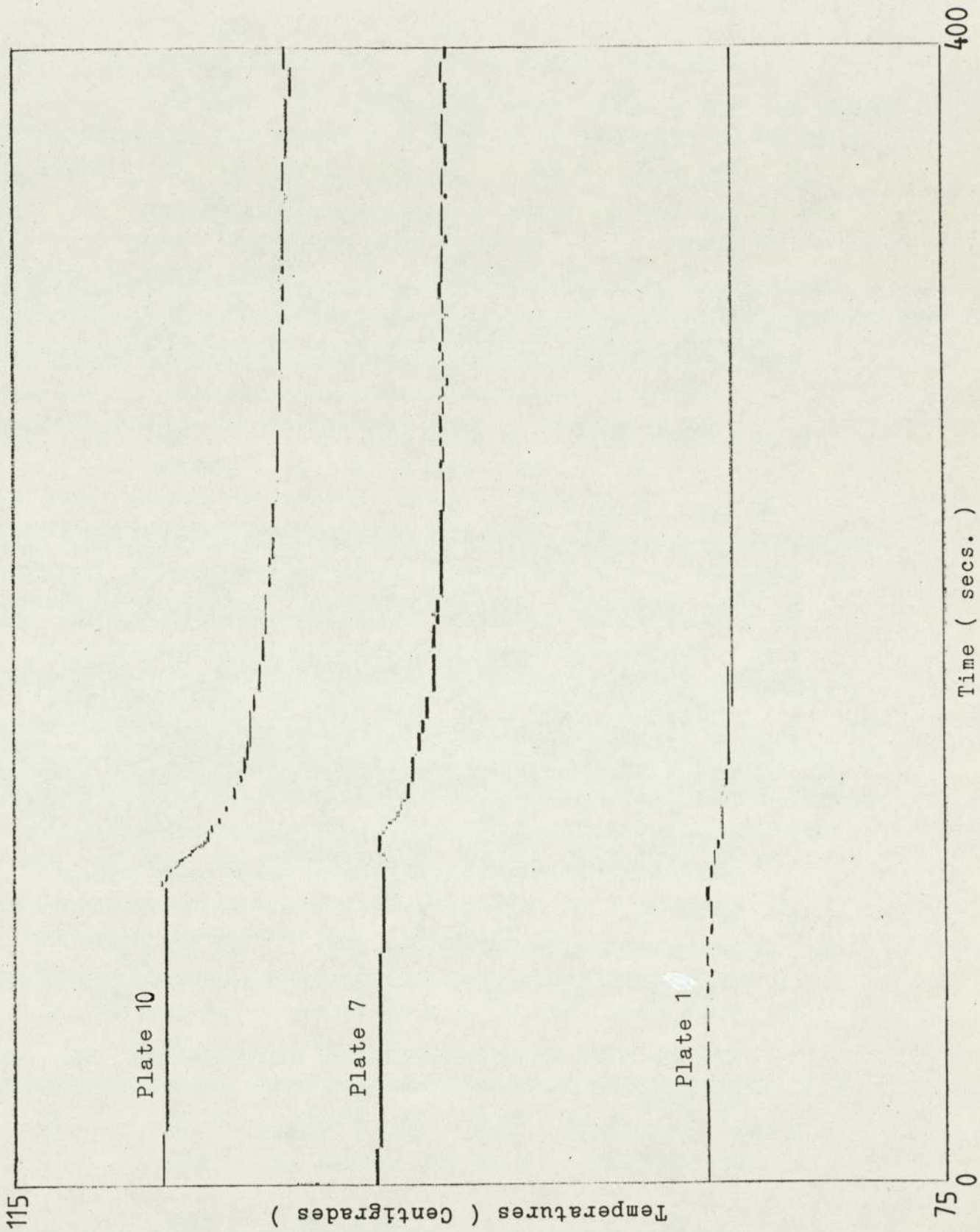


Figure 7.8 Response of tray temperatures due to a step change in feed rate

7.2.3.2 Microprocessor Control of Temperatures in a Double-Effect Evaporator

A parallel research effort in the Department is undertaken by Nawari[128] and is concerned with the modelling and control of a double-effect evaporator. The process model developed is to be ultimately incorporated in the design of a Wiener-Hopf controller[129,130] applied to a multivariable case. It was decided to demonstrate the feasibility of using the microprocessor in such a control scheme.

The control problem is schematically shown in Figure 7.9. It is proposed that the microprocessor regulates the cyclone outlet liquid temperature T_b and the feed preheater outlet temperature T_o using P+I action. In an actual Wiener-Hopf implementation, the set-points for the M6800 controllers would be generated by a Wiener-Hopt controller model residing in the H316. The H316 in this sense acts as a supervisory computer in a basically servo-control type of set-up. Since the Wiener-Hopf controller is yet to be fully developed and the prime consideration is to test the practicality of the control system architecture, a P+I model is used to generate the new set-points instead. Figure 7.10 illustrates the control philosophy in block diagram form.

The temperature T_o is regulated by controlling (to a set-point) the preheater outlet flowrate. The microprocessor measures the flow via a counter, driven by a turbine flowmeter, and adjusts the downstream valve (high pressure to close) accordingly. The temperature T_b is regulated by controlling (to a set-point) the input stream flowrate to the 1st effect. The microprocessor measures the steam rate via an analogue input channel connected to an orifice plate-based pressure transducer and adjusts the downstream valve (high pressure to open)

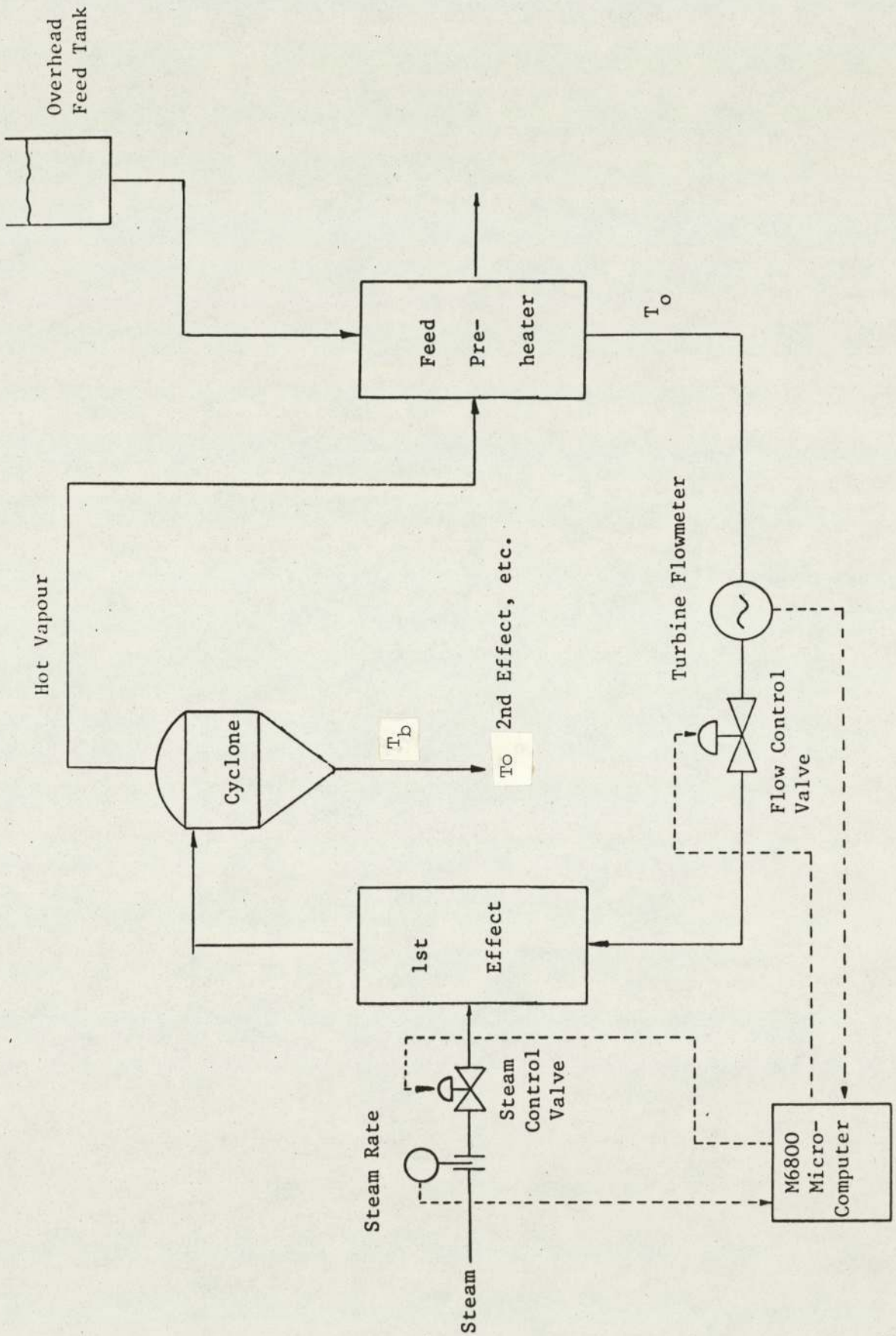


Figure 7.9 Schematic diagram of the Control Problem in the Double-Effect Evaporator

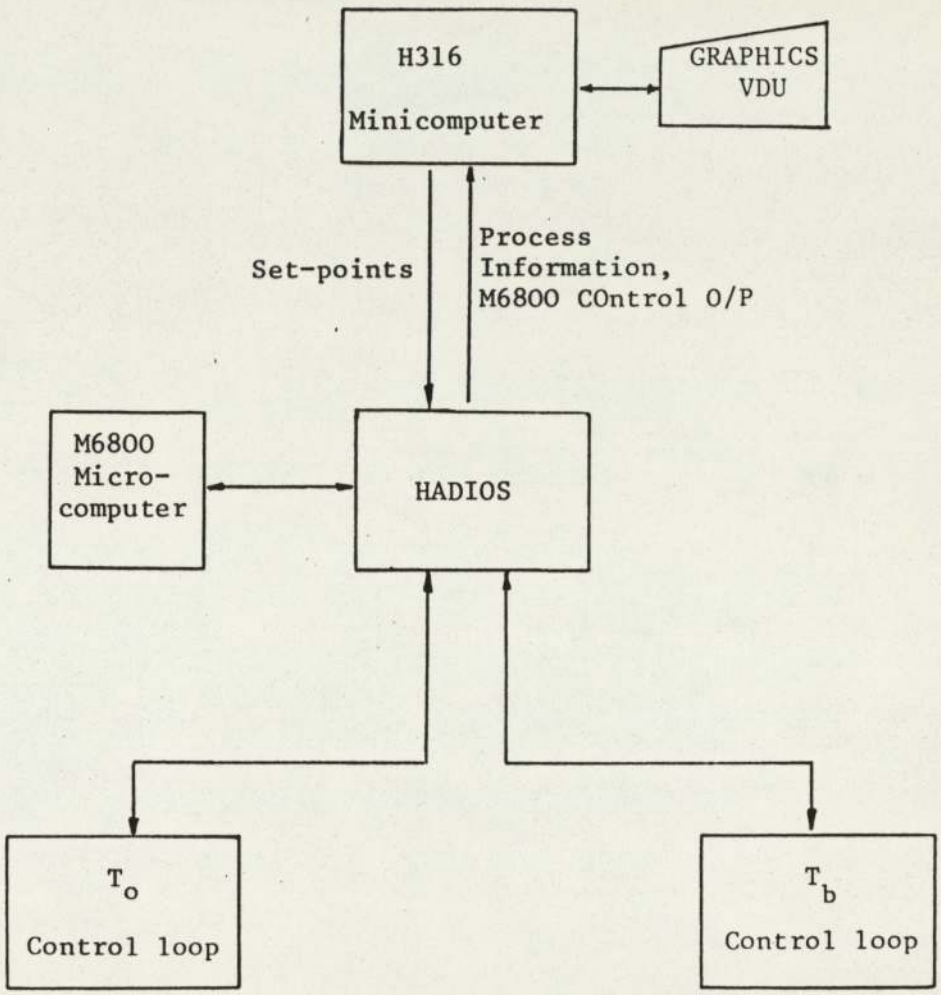


Figure 7.10 Block Diagram of a Double-Effect Evaporator Control System Design

accordingly. Because the hot vapour outlet from the 1st effect is used to heat the feed, the outlet temperature of which is being regulated by adjusting liquid flow which then feeds back into the 1st effect, the two control loops are inherently interactive.

The SD BASIC and BASIC programs for a control experiment is listed in Tables A7.3 and A7.4 of Appendix 7. Figure 7.11 shows the microprocessor controller performance. As in the distillation case, the H316 is in OFF-LINE mode. The sampling interval is 2 seconds to allow the plotting of microprocessor control outputs (histograms) at every process scan. When microprocessor control outputs are not plotted, the sampling interval can be reduced to one second and the H316 may send new set-points to the M6800 once in an integral multiple of the latter's sampling interval. Figure 7.11 shows that even with set-points generated by a relatively straightforward P+I model, T_o is reasonably well regulated. The 'pulse' at about 250 seconds is due to a deliberate flow disturbance introduced by affecting manually a secondary valve in the preheater outlet line. The regulation of T_b is less satisfactory and needs further controller adjustments.

Note on On-line Experiments

Admittedly, the number of on-line experiments is limited. This is because most of the time in this work has been directed at system and software development. However, the experiences with the distillation column and the double-effect evaporator demonstrated the practicality of the linked H316-M6800 twin processor facility for real-time applications.

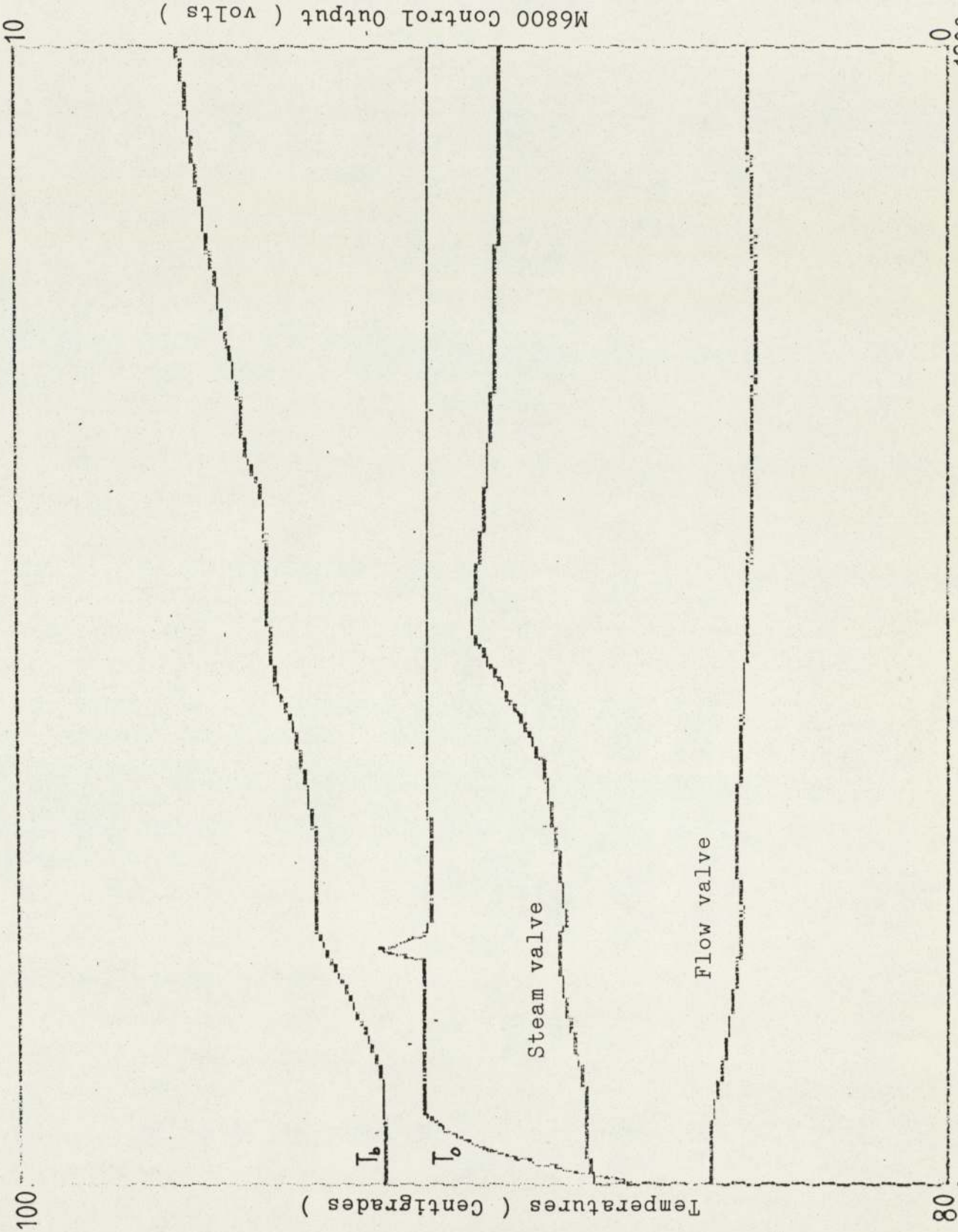


Figure 7.11 M6800 Control of Temperatures in the Double-Effect Evaporator

7.3 The Total Simulation Package

In this section the results from the total simulation package described in Chapter 6 are reviewed.

7.3.1 Overview of the Simulation experiments

The simulation experiments, besides being a design aid, have created a better understanding of the problems associated with the application of an Extended Kalman filter. The Departmental H316 minicomputer was chosen for such purposes because the simulation exercise was conducted with an on-line application in mind. Thus, the memory utilisation and software design of the package are reflective of this point.

Although the techniques of interfacing BASIC, FORTRAN and DAP-16 MOD2 programs are well established in the H316, thereby permitting the design of interactive simulation packages, there is virtually no operating system. The disk operating system, built around the A$D14 routine is primitive and handles only batch transfers. In the existing arrangement, it is not possible to edit programs using the Text Editor and store them directly on to floppy disks. A relatively minor modification to a FORTRAN subroutine would require the complete segment which it is in to be reconstructed. A change in the size of any COMMON block in any of the FORTRAN subprograms requires the entire simulation package to be reconstructed. This is to preserve the correct address alignments since the same variables may be referred to in different program segments. Changing the dimension of the filtering problem therefore requires a major effort. It is an unattractive feature of H316 operation which the user has to cope with.

The other aspect of the H316 that limits the running of simulation experiments is its slow speed by today's minicomputer standards.

A typical simulation run for EKF2 for instance, requires about 9 hours for a real problem time of 0.7 hour. The execution times for EKF1 are longer still. As a result, the simulation package was normally left to run overnight.

7.3.2 Dynamic Process Models

The results from the dynamic response studies have revealed certain weaknesses of the filter process model. This is particularly relevant in the enriching section which is basically two thirds of the column. Earlier, Figure 6.8 has shown the inability of the filter process model to predict the significant variations in vapour rates in this region. This weakness is partly associated with the assumptions of equimolal overflow and a simple heat transfer reboiler model incorporated in its formulation. The effect of a load disturbance on vapour throughput is only propagated through the relation $V = Q/\lambda$ (equation 5.62) and since changes in reboiler compositions are relatively small (see equation 5.61) due to the large hold-up, the changes in λ (hence, V) are therefore minimal.

There is less difficulty with composition changes due to disturbances in feed rate and/or feed composition. The model of the feed tray (equation 5.58 in particular) ensures that the feed tray composition x_7 is affected and hence the disturbance propagated throughout the column. In general, Model II is perturbed to a lesser extent as is evident from the response of tray compositions starting from their respective steady states, but which modelling aspect this is due to is difficult to pin-point.

In the bubble point calculations, the values of activity coefficients for the conditions tested were found to be in the range

0.94 to 1.0. To save some space and execution times, the activity coefficients could be simply assigned to unity for the filter process model without seriously affecting the prediction of equilibrium vapour compositions. In the bubble point iteration routine, if the limiting accuracy (EPSIL in subroutine SDBUB) is significantly less than .001 (on 0.1% of $1^{o}C$) say, EPSIL = .01, then the composition predictions becomes unsatisfactory. In fact, by doing so, the saving in computation times is not significant (about 1 second over a typical sampling interval).

7.3.3 Estimation

The simulation study has shown that the application of an Extended Kalman filtering problem on a minicomputer involves the careful consideration of memory space available, the software resources and process modelling. If the filter is to be implemented on-line than an appraisal of program execution times is necessary.

Since a Kalman filter essentially manipulates information from a process model and plant measurements, the quantiative nature of these dual contributions being reflected in the filtering equations 5.20 through 5.24, a sufficiently accurate filter process model is therefore desirable. In fact, a better model results in a better Φ and hence, better filter operation. However, at the moment there is no theoretical method of saying how good a particular model should be.

A similar situation arises in selecting the number of process measurements. With more measurements, one may be able to compensate for the effect of some modelling deficiencies but it also increases the demensionality of the filtering problem. In the on-line implementation, extra sensors would have to be installed. A more crucial problem associated with measurements is the question of system observability.

In this work, the discrete observability matrix L_{od} helps in the selection of measurements so that one is assured the system is not unobservable at all times. On the other hand, unless it is computationally determined, one also cannot guarantee observability at each sampling instant. Thus, it pays to have as accurate a filter process model as possible since in regimes of unobservability, the filter tends to weight model predictions more heavily and 'ignores' the measurements.

In general, the states associated with the measurement variables have been better estimated. The estimates usually converge to steady values and the corresponding elements in the error covariance matrix are small in magnitude. It is when the filter starts to be unstable that elements of the error covariance matrix become unduly large and physically meaningless (for example, the composition error covariances becoming greater than unity).

7.3.4 On-line estimation and the linked twin processor system

The simulation results with EKF2 have shown that the on-line filter EKF3, is only practical if the sampling intervals do not exceed 18 seconds.* The study suggests that filtering operation breaks down due to the poor approximations of the true state transition matrix. Little can be done about program execution times except through the use of hardware techniques and more efficient matrix manipulations. The other approach is to find a better Φ which is not too sensitive to the Δt.

EKF3 has also been developed in line with the requirements of the linked H316-M6800 twin processor system. In an on-line situation, the microcomputer can be given the task of regulating the reflux drum and reboiler levels, and other fast flow loops which

require relatively high sampling frequencies. It is therefore suggested that the H316 operates in the OFF-LINE mode. Although process information is available at every M6800 process scan, the estimator uses the measurements at a suitable multiple of micro-computer sampling interval.

The estimates can be calculated as soon as process measurements became available. This is because the filter gain and the predicted error covariance matrices can be precomputed. Such a situation is particularly important in a stochastic control application since corrective actions should be output to the plant as soon as the latest estimates become available. The linked processor facility can cope with all these requirements quite readily.

* As shown in Section 6.5.2, the computational overhead for estimation requires a sampling interval of at least 76 seconds.

CHAPTER EIGHT

CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

8. CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

This research has shown that microprocessor technology can be readily applied to problems of industrial process control by proper consideration of both the hardware and software resources available at the time of system design. If the application is a first one for a particular person, then it also involves a learning process which can consume considerable time and development effort.

8.1 The Linked H316-M6800 Twin Processor System

The linked H316-M6800 twin processor system developed in this work is an example where the use of a microcomputer has considerably extended an existing real-time data acquisition and process control facility thereby creating a more powerful and flexible computing base for research and experimentation. In particular, the system is sufficiently general-purpose to support two users with independent sampling frequencies and control configurations. Alternatively, a single user may use the linked facility for a wide range of data processing activities including OFF- and ON-LINE implementations. With proper consideration of timing and resource limitations, the user may also program the facility to provide for distributed processing and control of chemical plants. The experience with the double-effect evaporator, though limited, has indicated that advanced control strategies can also be readily implemented.

Like many other independent initiatives in microprocessor applications, hardware and software aspects are integrated and considered important factors in system design. In this research, the design of the interfacing requirements and the communication protocol have been found to be relatively straightforward and additional hardware has been kept to

a minimum. Attention is focussed on the use of a single PIA at the microcomputer end. In fact, the design has demonstrated the power and programmability of a single peripheral interface chip which is typical of microprocessor-based devices. Clearly, more powerful designs can be constructed if more peripheral interfaces are used.

The principal software objective in the linked processor system is to allow interactive access to a chemical plant. In either processor, BASIC provides the means for this. The general user therefore need not be too concerned with two-level programming in a particular application. However, more specialised applications normally require additional FORTRAN and assembly languages routines. As the simulation of a Kalman Filtering application shows, this involves the careful consideration of memory utilisation, the software resources available and program execution times.

The development of the linked facility also highlighted some problems which are commonly encountered in the application of microcomputers. Spurious signals and power supply transients were the main problems since interfacing problems were minimum as both the HADIOS and M6800 interfaces are TTL compatible. Nevertheless, the service of a microelectronics staff has been found to be indispensible.

8.2 Simulation of a Kalman Filtering Application

In an attempt to apply an Extended Kalman Filter to a binary distillation process, this work has also demonstrated the value of interactive digital simulation, both as a design aid and as a means of understanding the mechanics of the filtering problem. In particular, the use of the floppy disk overlay facility has permitted the combined parameter and state estimation study for up to a 15th order system.

For a sampling interval of 18 seconds and below, the filter's performance has been found to be satisfactory.

More critical than space limitations turned out to be the speed of the H316 minicomputer. An on-line attempt of an 11th order state estimator was not possible because the sampling interval required for a stable operation was too small.

8.3 Recommendations for Further Work

8.3.1 The linked processor facility

The linked processor facility developed in this work has created more avenues for research into process control problems, in particular the implementation of microprocessor-based controllers. Because of available human and material resources at the time of system design, the communication protocol depends heavily on the use of a single PIA for data I/O to and from the M6800. It is suggested that an extra PIA be added to the system to relieve this burden. The additional PIA could be used for data transfers between the two computers or even interfaced directly to an item of process plant.

Admittedly, the number of on-line experiments using the linked facility is limited. A more exhaustice series of experiments should be carried out to check the robustness of the software developed because a piece of real-time software is almost impossible to be 'bug' free. Using it on-line can help define the operating conditions and precautions, the adherence to which would normally result in satisfactory use of the linked processor system.

The recommendations to improve the operation of the IBM Distillation Column and its associated instrumentation have already been suggested by Daie in his work. The recurring problem is still the lack of properly designed flow transducers and to a lesser extent,

stiction in the motorised valves. It was also observed that the rate of condensation of top vapour product is relatively low. This is due to the limited reboiler heater capacity and the phenomenon of parasitic reflux in the connecting pipe above the enriching section. Because of this, the distillate and reflux valves could not be set to large opening values as this would simply empty the reflux drum in a matter of several minutes. Any major control work on the column should therefore consider these operating limitations very carefully.

8.3.2 Kalman Filtering Application

Since it was shown that the on-line state estimator EKF3 works reasonably well in simulation, methods of finding a better state transition matrix Φ and keeping the sampling interval small should be sought so that an on-line implementation becomes practical. A more accurate Φ may be obtained by formulating a better filter process model or considering higher-order terms in its Taylor series expansion. Alternatively, an adaptive design such as the one experimented by Webb[111] could be considered. The last two approaches may incur considerable overheads in execution times. This means programs should execute faster. It is recommended that a hardware-based multiplication/divide option or a floating point processor be used with the H316. This should solve many of the difficulties described in the filtering exercise.

The possibility of the M6800 doing some of the interscan calculations should also be considered. One may start with the preparation of a library of general purpose mathematical routines, written in M6800 Assembly Language. Matrix and vector manipulations can then be achieved using the library package. Alternatively, some of the microcomputing load can be done in SD BASIC. If future work is

to develop in this direction then it is best to experiment with small order systems first. In the final analysis, the question of timing and accuracy, and general aspects of concurrent programming can be important considerations.

8.3.3 Possible re-design of the estimation procedure

The possibility of redesigning the estimation procedure based on a constant gain extended Kalman Filter should also be investigated. Such a filter would reduce the computational overhead since the calculations of the error covariance and filter gain matrices are no longer necessary. The optimum gain would still have to be determined by trial and error although the steady-state values obtained in this work may be used as starting values. The method would also reduce the size of the sampling interval thus increasing the possibility of an on-line application.

APPENDIX

3.1 Construction of a self-contained '16000' loader - Memory map

```
*LOW    10050
*START  13000
*HIGH   13600
*NAMES  34007
*COMN   37777
*BASE   13571
*BASE   12774
*BASE   11776
*BASE   10774
I$MC    10215**
I$CB    10227**
INHALT  13444

MF
```

An SLST of the loader is punched out to paper tape

Subroutines I$MC and I$CB were not loaded since in using the loader, the use of magnetic or card input medium is not expected.

Table A3.1 DAP-16 MOD2 INSTRUCTIONS

Mnemonic	Meaning	Mnemonic	Meaning
ADD	Binary add to A register	LDA	Load A
ALR	Logical left rotate of A	LDX	Load Index Register
ALS	Arithmetic left shift of A	LLL	Long left shift of A and B registers
ANA	Logical AND to A	LRL	Long nightshift of A and B
ARR	Logical right rotate A	MPY	High speed multiply
AOA	Add one to A	NOP	No operation
ARS	Arithmetic right shift of A	OCP	Output control pulse
CAS	Compare and skip	OCT	Octal constant
CRA	Clear A	OTK	Output keys
DIV	High speed divide	SKP	Unconditional skip
ENB	Enable CPU interrupts	SKS	Skip if sense line set
ERA	Exclusive OR to A	SMI	Skip if A minus
HLT	Halt	SMK	Set mask
IAB	Interchange A and B	SNZ	Skip if A $\neq$ 0
IMA	Interchange A and memory location	SPL	Skip if A > 0
INA	Input to A from peripheral	SSM	Set sign minus
INH	Inhibit CPU interrupts	STA	Store A
INK	Input keys	STX	Store index register
IRS	Increment, replace and skip	SUB	Subtract
JMP	Unconditional Jump	SZE	Skip if A = 0
JST	Jump and store current location	TCA	Two's complement A

PROGRAMMING INSTRUCTIONS FOR HADIOS

HADIOS CONTROLLER

SKS 'XXYY - skip if controller not interrupting
- YY is device address of controller (70)
- XX is controller highway address line (17)

DIGITAL INPUT

INA 'IIYY - input to A register
- II is the digital input highway address line
  DGIA - II = 11
  DGIB - II = 10

DIGITAL OUTPUT

OTA 'OOYY - Output from A register
- OO is the digital output highway address line
  DGOA - OO = 13
  DGOB - OO = 12

COUNTER INPUTS

INA 'CCYY - Input to A register from counter 8 bit register. Counter register is automatically cleared after an INA
- CC is the counter input highway address line
  Counter 1 CC = 04
  Counter 2 CC = 06
  Counter 3 CC = 02

OTA 'DDYY - Output from A register to present the counter 8 bit register
- DD is the second counter input highway address line
  Counter 1 DD = 05
  Counter 2 DD = 07
  Counter 3 DD = 03

SKS 'CCYY - Skip if counter not interrupting

OCP 'CCYY - Enable half-full counter interrupts

OCP 'DDYY - Reset counter interrupt mode

ANALOGUE INPUTS

OCP 'AAYY - Starts ADC conversion cycle
- AA is the first analogue input highway address line
- Corresponds to DATA in the HADIOS Executive
- AA = 00

INA 'AAYY - Input data to the A register is conversion has completed; data enters the 10 MSB's of A.
- The previous contents of A must be cleared first.

OTA 'BBYY - Output set up word to ADC.
- BB is the second analogue input highway address line (01).

The set up word format is as follows:

0 000 010 000 MMM NNN
where MM = 00, 01, 02 (multiplexer number)
N NNN = 00 to '15 (channel number)
bit 6 is set to logic '1' to force the ADC to be in the sequential address mode.

CONTROLLER INTERRUPT BIT

In an interrupt mode, the HADIOS controller interrupt bit must be set with a SMK '20 instruction issued with bit 13 set in the A register.

ALARM INPUTS

INA 'LLYY - Input status of alarm input channels into A register.
- LL is the device input highway address line (14).

OPA 'LLYY - Sets the alarm inputs option into 'input' mode.

SKS 'LLYY - Skip if alarm inputs not interrupting.

Table A3.2 Summary of DAP-16 MOD2 Pseudo-Operations Used

Mnemonic	Meaning
ABS	Instruction following to have absolute addresses
BCI	Binary coded information
BSZ	Block set set to zero
DAC	Define address constant
ORG	The location of the next instruction or data
REL	Instruction following to have relocatable addresses (relative)
SETB	Set base
VFD	Variable Field Descriptor
**	Zero address code
*	Indirect operation (when in operation code)
*	Address of this location (when in address field)
'222	Octal constant
= 9	Literal constant

Table A3.3 M6800 Instruction Set[131]

ACCUMULATOR AND MEMORY INSTRUCTIONS

ACCUMULATOR AND MEMORY OPERATIONS	MNEMONIC	IMMED OP	IMMED ~	IMMED #	DIRECT OP	DIRECT ~	DIRECT #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	INHER OP	INHER ~	INHER #	BOOLEAN/ARITHMETIC OPERATION (All register labels refer to contents)	5 H	4 I	3 N	2 Z	1 V	0 C
Add	ADDA	8B	2	2	9B	3	2	AB	5	2	BB	4	3				A + M → A	↕	•	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	5	2	FB	4	3				B + M → B	↕	•	↕	↕	↕	↕
Add Acmltrs	ABA													1B	2	1	A + B → A	↕	•	↕	↕	↕	↕
Add with Carry	ADCA	89	2	2	99	3	2	A9	5	2	B9	4	3				A + M + C → A	↕	•	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	5	2	F9	4	3				B + M + C → B	↕	•	↕	↕	↕	↕
And	ANDA	84	2	2	94	3	2	A4	5	2	B4	4	3				A • M → A	•	•	↕	↕	R	•
	ANDB	C4	2	2	D4	3	2	E4	5	2	F4	4	3				B • M → B	•	•	↕	↕	R	•
Bit Test	BITA	85	2	2	95	3	2	A5	5	2	B5	4	3				A • M	•	•	↕	↕	R	•
	BITB	C5	2	2	D5	3	2	E5	5	2	F5	4	3				B • M	•	•	↕	↕	R	•
Clear	CLR							6F	7	2	7F	6	3				00 → M	•	•	R	S	R	R
	CLRA													4F	2	1	00 → A	•	•	R	S	R	R
	CLRB													5F	2	1	00 → B	•	•	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	5	2	B1	4	3				A − M	•	•	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	5	2	F1	4	3				B − M	•	•	↕	↕	↕	↕
Compare Acmltrs	CBA													11	2	1	A − B	•	•	↕	↕	↕	↕
Complement, 1's	COM							63	7	2	73	6	3				M → M	•	•	↕	↕	R	S
	COMA													43	2	1	A → A	•	•	↕	↕	R	S
	COMB													53	2	1	B → B	•	•	↕	↕	R	S
Complement, 2's (Negate)	NEG							60	7	2	70	6	3				00 − M → M	•	•	↕	↕	①	②
	NEGA													40	2	1	00 − A → A	•	•	↕	↕	①	②
	NEGB													50	2	1	00 − B → B	•	•	↕	↕	①	②
Decimal Adjust, A	DAA													19	2	1	Converts Binary Add. of BCD Characters into BCD Format	•	•	↕	↕	↕	③
Decrement	DEC							6A	7	2	7A	6	3				M − 1 → M	•	•	↕	↕	④	•
	DECA													4A	2	1	A − 1 → A	•	•	↕	↕	④	•
	DECB													5A	2	1	B − 1 → B	•	•	↕	↕	④	•
Exclusive OR	EORA	88	2	2	98	3	2	A8	5	2	B8	4	3				A ⊕ M → A	•	•	↕	↕	R	•
	EORB	C8	2	2	D8	3	2	E8	5	2	F8	4	3				B ⊕ M → B	•	•	↕	↕	R	•
Increment	INC							6C	7	2	7C	6	3				M + 1 → M	•	•	↕	↕	⑤	•
	INCA													4C	2	1	A + 1 → A	•	•	↕	↕	⑤	•
	INCB													5C	2	1	B + 1 → B	•	•	↕	↕	⑤	•
Load Acmltr	LDAA	86	2	2	96	3	2	A6	5	2	B6	4	3				M → A	•	•	↕	↕	R	•
	LDAB	C6	2	2	D6	3	2	E6	5	2	F6	4	3				M → B	•	•	↕	↕	R	•
Or, Inclusive	ORAA	8A	2	2	9A	3	2	AA	5	2	BA	4	3				A + M → A	•	•	↕	↕	R	•
	ORAB	CA	2	2	DA	3	2	EA	5	2	FA	4	3				B + M → B	•	•	↕	↕	R	•
Push Data	PSHA													36	4	1	A → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
	PSHB													37	4	1	B → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
Pull Data	PULA													32	4	1	SP + 1 → SP, M_{SP} → A	•	•	•	•	•	•
	PULB													33	4	1	SP + 1 → SP, M_{SP} → B	•	•	•	•	•	•
Rotate Left	ROL							69	7	2	79	6	3				M	•	•	↕	↕	⑥	↕
	ROLA													49	2	1	A	•	•	↕	↕	⑥	↕
	ROLB													59	2	1	B	•	•	↕	↕	⑥	↕
Rotate Right	ROR							66	7	2	76	6	3				M	•	•	↕	↕	⑥	↕
	RORA													46	2	1	A	•	•	↕	↕	⑥	↕
	RORB													56	2	1	B	•	•	↕	↕	⑥	↕
Shift Left, Arithmetic	ASL							68	7	2	78	6	3				M	•	•	↕	↕	⑥	↕
	ASLA													48	2	1	A	•	•	↕	↕	⑥	↕
	ASLB													58	2	1	B	•	•	↕	↕	⑥	↕
Shift Right, Arithmetic	ASR							67	7	2	77	6	3				M	•	•	↕	↕	⑥	↕
	ASRA													47	2	1	A	•	•	↕	↕	⑥	↕
	ASRB													57	2	1	B	•	•	↕	↕	⑥	↕
Shift Right, Logic	LSR							64	7	2	74	6	3				M	•	•	R	↕	⑥	↕
	LSRA													44	2	1	A	•	•	R	↕	⑥	↕
	LSRB													54	2	1	B	•	•	R	↕	⑥	↕
Store Acmltr.	STAA				97	4	2	A7	6	2	B7	5	3				A → M	•	•	↕	↕	R	•
	STAB				D7	4	2	E7	6	2	F7	5	3				B → M	•	•	↕	↕	R	•
Subtract	SUBA	80	2	2	90	3	2	A0	5	2	B0	4	3				A − M → A	•	•	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	5	2	F0	4	3				B − M → B	•	•	↕	↕	↕	↕
Subtract Acmltrs	SBA													10	2	1	A − B → A	•	•	↕	↕	↕	↕
Subtr. with Carry	SBCA	82	2	2	92	3	2	A2	5	2	B2	4	3				A − M − C → A	•	•	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	5	2	F2	4	3				B − M − C → B	•	•	↕	↕	↕	↕
Transfer Acmltrs	TAB													16	2	1	A → B	•	•	↕	↕	R	•
	TBA													17	2	1	B → A	•	•	↕	↕	R	•
Test, Zero or Minus	TST							6D	7	2	7D	6	3				M − 00	•	•	↕	↕	R	R
	TSTA													4D	2	1	A − 00	•	•	↕	↕	R	R
	TSTB													5D	2	1	B − 00	•	•	↕	↕	R	R

ADDRESSING MODES

COND. CODE REG.

LEGEND:

- OP Operation Code (Hexadecimal);
- ~ Number of MPU Cycles;
- # Number of Program Bytes;
- + Arithmetic Plus;
- − Arithmetic Minus;
- • Boolean AND;
- M_{SP} Contents of memory location pointed to be Stack Pointer;
- ⊕ Boolean Inclusive OR;
- ⊕ Boolean Exclusive OR;
- $\overline{M}$ Complement of M;
- → Transfer Into;
- 0 Bit = Zero;
- 00 Byte = Zero;
- H Half-carry from bit 3;
- I Interrupt mask
- N Negative (sign bit)
- Z Zero (byte)
- V Overflow, 2's complement
- C Carry from bit 7
- R Reset Always
- S Set Always
- ↕ Test and set if true, cleared otherwise
- • Not Affected
- CCR Condition Code Register
- LS Least Significant
- MS Most Significant

JUMP AND BRANCH INSTRUCTIONS

JUMP AND BRANCH OPERATIONS	MNEMONIC	RELATIVE OP	RELATIVE ~	RELATIVE #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	INHER OP	INHER ~	INHER #	BRANCH TEST	5 H	4 I	3 N	2 Z	1 V	0 C
Branch Always	BRA	20	4	2										None	•	•	•	•	•	•
Branch If Carry Clear	BCC	24	4	2										C = 0	•	•	•	•	•	•
Branch If Carry Set	BCS	25	4	2										C = 1	•	•	•	•	•	•
Branch If = Zero	BEQ	27	4	2										Z = 1	•	•	•	•	•	•
Branch If ≥ Zero	BGE	2C	4	2										N ⊕ V = 0	•	•	•	•	•	•
Branch If > Zero	BGT	2E	4	2										Z + (N ⊕ V) = 0	•	•	•	•	•	•
Branch If Higher	BHI	22	4	2										C + Z = 0	•	•	•	•	•	•
Branch If ≤ Zero	BLE	2F	4	2										Z + (N ⊕ V) = 1	•	•	•	•	•	•
Branch If Lower Or Same	BLS	23	4	2										C + Z = 1	•	•	•	•	•	•
Branch If < Zero	BLT	2D	4	2										N ⊕ V = 1	•	•	•	•	•	•
Branch If Minus	BMI	2B	4	2										N = 1	•	•	•	•	•	•
Branch If Not Equal Zero	BNE	26	4	2										Z = 0	•	•	•	•	•	•
Branch If Overflow Clear	BVC	28	4	2										V = 0	•	•	•	•	•	•
Branch If Overflow Set	BVS	29	4	2										V = 1	•	•	•	•	•	•
Branch If Plus	BPL	2A	4	2										N = 0	•	•	•	•	•	•
Branch To Subroutine	BSR	8D	8	2											•	•	•	•	•	•
Jump	JMP				6E	4	2	7E	3	3				See Special Operations	•	•	•	•	•	•
Jump To Subroutine	JSR				AD	8	2	BD	9	3					•	•	•	•	•	•
No Operation	NOP										01	2	1	Advances Prog Cntr Only	•	•	•	•	•	•
Return From Interrupt	RTI										3B	10	1		———			(10)		———
Return From Subroutine	RTS										39	5	1	See special Operations	•	•	•	•	•	•
Software Interrupt	SWI										3F	12	1		•	S	•	•	•	•
Wait for Interrupt	WAI										3E	9	1		•	(11)	•	•	•	•

INDEX REGISTER AND STACK MANIPULATION INSTRUCTIONS

INDEX REGISTER AND STACK POINTER OPERATIONS	MNEMONIC	IMMED OP	IMMED ~	IMMED #	DIRECT OP	DIRECT ~	DIRECT #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	INHER OP	INHER ~	INHER #	BOOLEAN/ARITHMETIC OPERATION	5 H	4 I	3 N	2 Z	1 V	0 C
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	6	2	BC	5	3				$(X_H X_L) - (M/M+1)$	•	•	(7)	↕	(9)	•
Decrement Index Reg	DEX													09	4	1	$X - 1 \rightarrow X$	•	•	•	↕	•	•
Decrement Stack Pntr	DES													34	4	1	$SP - 1 \rightarrow SP$	•	•	•	•	•	•
Increment Index Reg	INX													08	4	1	$X + 1 \rightarrow X$	•	•	•	↕	•	•
Increment Stack Pntr	INS													31	4	1	$SP + 1 \rightarrow SP$	•	•	•	•	•	•
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	6	2	FE	5	3				$M \rightarrow X_H, (M+1) \rightarrow X_L$	•	•	(4)	↕	R	•
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	6	2	BE	5	3				$M \rightarrow SP_H, (M+1) \rightarrow SP_L$	•	•	(5)	↕	R	•
Store Index Reg	STX				DF	5	2	EF	7	2	FF	6	3				$X_H \rightarrow M, X_L \rightarrow (M+1)$	•	•	(6)	↕	R	•
Store Stack Pntr	STS				9F	5	2	AF	7	2	BF	6	3				$SP_H \rightarrow M, SP_L \rightarrow (M+1)$	•	•	(9)	↕	R	•
Index Reg → Stack Pntr	TXS													35	4	1	$X - 1 \rightarrow SP$	•	•	•	•	•	•
Stack Pntr → Index Reg	TSX													30	4	1	$SP + 1 \rightarrow X$	•	•	•	•	•	•

CONDITION CODE REGISTER MANIPULATION INSTRUCTIONS

CONDITIONS CODE REGISTER OPERATIONS	MNEMONIC	INHER OP	INHER ~	INHER #	BOOLEAN OPERATION	5 H	4 I	3 N	2 Z	1 V	0 C
Clear Carry	CLC	0C	2	1	$0 \rightarrow C$	•	•	•	•	•	R
Clear Interrupt Mask	CLI	0E	2	1	$0 \rightarrow I$	•	R	•	•	•	•
Clear Overflow	CLV	0A	2	1	$0 \rightarrow V$	•	•	•	•	R	•
Set Carry	SEC	0D	2	1	$1 \rightarrow C$	•	•	•	•	•	S
Set Interrupt Mask	SEI	0F	2	1	$1 \rightarrow I$	•	S	•	•	•	•
Set Overflow	SEV	0B	2	1	$1 \rightarrow V$	•	•	•	•	S	•
Acmltr A → CCR	TAP	06	2	1	$A \rightarrow CCR$	———			(12)		———
CCR → Acmltr A	TPA	07	2	1	$CCR \rightarrow A$	•	•	•	•	•	•

CONDITION CODE REGISTER NOTES:

(Bit set if test is true and cleared otherwise)

(1) (Bit V) Test: Result = 10000000?

(2) (Bit C) Test: Result = 00000000?

(3) (Bit C) Test: Decimal value of most significant BCD Character greater than [illegible] (Not cleared if previously set.)

(4) (Bit V) Test: Operand = 10000000 prior to execution?

(5) (Bit V) Test: Operand = 01111111 prior to execution?

(6) (Bit V) Test: Set equal to result of N ⊕ C after shift has occurred.

(7) (Bit N) Test: Sign bit of most significant (MS) byte of result = 1?

(8) (Bit V) Test: 2's complement overflow from subtraction of MS bytes?

(9) (Bit N) Test: Result less than zero? (Bit 15 = 1)

(10) (All) Load Condition Code Register from Stack. (See Special Operations)

(11) (Bit I) Set when interrupt occurs. If previously set, a Non Maskable Interru[illegible] required to exit the wait state.

(12) (All) Set according to the contents of Accumulator A.

Table A4.1 Assembler listing of the HADIOS Executive Rev. 03

```
                        *                                   PAGE      1
0001                    *
0002                    * HADIOS EXECUTIVE REV 03 for the
0003                    * linked H316-M6800 twin processor system.
0004                    * Listing of version dated 15.2.1983
0005                    * Prepared by A.F.B. SHAFII
0006                    *
0007                    * Normally loaded from '27000 onwards....
0008                    * Must be core-resident as M6800 has
0009                    * independent access of HADIOS system.
0010                    *
0011                    * Software Package should be entered at
0012                    * '27000 .. If user does not want any MICRO
0013                    * interrupts then package should be entered
0014                    * at P = '1000
0015                    *
0016                         REL
0017                         ENT   HADIOS,PDCS        Entries for
0018                         ENT   SKST               Loader memory
0019                         ENT   CALM               map ..
0020                         ENT   CAL0
0021                         ENT   IFLG
0022                         ENT   ERRM
0023                         ENT   MCOD
0024                         ENT   ERCL
0025                         ENT   CB2
0026                         ENT   CA2
0027                         ENT   ABUF
0028                         ENT   MBUF
0029                         ENT   C123
0030                         ENT   CTRS
0031                         ENT   MCTR
0032                         ENT   TIME
0033                         ENT   TABL
0034                         ENT   ICT1
0035                         ENT   M0
0036                         ENT   CA22
0037                         ENT   SUB1
0038                         ENT   SUB2
0039                         ENT   SUB3
0040                         ENT   SUB4
0041                         SETB  BAS0
0042                    *
0043                    * General form of HADIOS I/O Commands is
0044                    * YYY 'XX70 where YYY is an I/O command ..
0045                    * XX is device Line address and 70 is HADIOS
0046                    * Controller address ..
0047                    *
0048         000070     DATA EQU   '0070
0049         000170     ANAG EQU   '0170
0050         000270     CTR3 EQU   '0270
0051         000370     SET3 EQU   '0370
0052         000470     CTR1 EQU   '0470
0053         000570     SET1 EQU   '0570
0054         000670     CTR2 EQU   '0670
0055         000770     SET2 EQU   '0770
0056         001070     DGIB EQU   '1070
0057         001170     DGIA EQU   '1170
```

```
                                *
0058            001270      DGOB EQU   '1270
0059            001370      DGOA EQU   '1370
0060            001470      ALRI EQU   '1470
0061            001770      HAD  EQU   '1770
0062                        *
0063            000020      CLK  EQU   '20
0064                        *
0065                        * SIP = BASIC statement INDEX POINTER
0066                        * SBP = BASIC statement  BYTE POINTER
0067                        * IBUF is BASIC input BUFFER
0068                        * CJST is one location before BASIC 'CALL'
0069                        * table ... See CALL statement PROCESSOR
0070                        * of BASIC Interpreter.
0071                        *
0072            000220      SCLK EQU   '220
0073            000034      SIP  EQU   '34
0074            000037      SBP  EQU   '37
0075            000230      IBUF EQU   '230
0076            000515      CJST EQU   '515
0077            004064      SS   EQU   '4064
0078                        *
0079                        * BASIC Maths. Package Pointers...Library
0080                        * routines are similar to corresponding
0081                        * FORTRAN Library routines ..
0082                        *
0083            000675      C12  EQU   '675
0084            000676      C21  EQU   '676
0085            000653      L22  EQU   '653
0086            000654      H22  EQU   '654
0087            000670      D22  EQU   '670
0088            000674      M22  EQU   '674
0089                        *
0090                        *Start CLOCK in non-interrupt mode
0091                        *Initialise CA2,CB2 Interrupts and program
0092                        *flags
0093                        *
0094 00000      100000      PDCS SKP
0095 00001      000000      M680 BSZ   1
0096 00002      0 02 00635       LDA   NMAX
0097 00003      0 04 00061       STA   '61
0098 00004      14 0020          OCP   CLK
0099 00005      0 02 00650       LDA   MCKK
0100 00006      0 04 02572       STA   ABF0
0101 00007      0 10 00247       JST   CII
0102 00010      0 000156         DAC   LNRE-2
0103 00011      0 001637         DAC   CB2
0104 00012      0 10 00247       JST   CII
0105 00013      0 000171         DAC   LNRF-2
0106 00014      0 002160         DAC   CA2
0107 00015      000401           ENB
0108 00016      0 02 00745       LDA   =-10
0109 00017      0 04 01552       STA   A20
0110 00020      0 01 01000       JMP   '1000
0111                        *
0112                        * Interrupt entry point --- via loacation'63
0113                        * Save processor KEYS first before decoding
0114                        * interrupts via ALARM INPUT sub-interface..
```

```
                                *                                PAGE        3
0115                            * A device may interrupt iff it is ready,its
0116                            * interrupt MASK flip-flop is set and the
0117                            * system interrupt is enabled by the ENB
0118                            * instruction .
0119                            *
0120 00021   0 000000    SKST   DAC    **
0121 00022   34 0020             SKS    CLK
0122 00023   0 01 00074          JMP    LNKA
0123 00024   34 1770             SKS    HAD
0124 00025   0 01 00030          JMP    CONT
0125 00026   0 10 03603          JST    ERCL
0126 00027   147322             BCI    1,NR
0127                            *
0128                            * HADIOS interrupting
0129                            *
0130 00030   34 0470     CONT   SKS    CTR1
0131 00031   0 01 00107          JMP    LNKB
0132 00032   34 0670             SKS    CTR2
0133 00033   0 01 00122          JMP    LNKC
0134 00034   34 0270             SKS    CTR3
0135 00035   0 01 00135          JMP    LNKD
0136 00036   34 1470             SKS    ALRI
0137 00037   0 01 00042          JMP    MC68
0138 00040   0 10 03603          JST    ERCL
0139 00041   147303             BCI    1,NC
0140 00042   0 13 00071  MC68    IMA    AINT
0141 00043   000043             INK
0142 00044   000005             SGL
0143 00045   0 04 00072          STA    SAVK
0144 00046   14 1470             OCP    ALRI
0145 00047   54 1470             INA    ALRI
0146 00050   0 01 00047          JMP    *-1
0147 00051   0 04 00073          STA    MIKR
0148 00052   100400             SPL
0149 00053   0 01 00060          JMP    *+5
0150 00054   0 02 00072          LDA    SAVK
0151 00055   171020             OTK
0152 00056   0 13 00071          IMA    AINT
0153 00057   0 01 00150          JMP    LNKE
0154 00060   0 03 00744          ANA    ='40000
0155 00061   101040             SNZ
0156 00062   0 01 00065          JMP    *+3
0157 00063   0 10 03603          JST    ERCL
0158 00064   147315             BCI    1,NM
0159 00065   0 02 00072          LDA    SAVK
0160 00066   171020             OTK
0161 00067   0 13 00071          IMA    AINT
0162 00070   0 01 00163          JMP    LNKF
0163 00071   000000     AINT   BSZ    1
0164 00072   000000     SAVK   BSZ    1
0165 00073   000000     MIKR   BSZ    1
0166                            *
0167                            * CLOCK interrupt link
0168                            *
0169 00074   0 10 00176  LNKA   JST    CIH
0170 00075   000000             BSZ    5
0171 00102   000001             OCT    1
```

```
                                *                                       PAGE        4
0172 00103    0 000075          DAC   LNKA+1
0173 00104    0 000000    LNRA  DAC   **
0174 00105    0 35 00103        LDX   *-2
0175 00106    0 01 00222        JMP   CIR
0176                            *
0177                            * Counter interrupt links
0178                            *
0179 00107    0 10 00176  LNKB  JST   CIH                   Counter 1
0180 00110    000000            BSZ   5
0181 00115    000010            OCT   10
0182 00116    0 000110          DAC   LNKB+1
0183 00117    0 000000    LNRB  DAC   **
0184 00120    0 35 00116        LDX   *-2
0185 00121    0 01 00222        JMP   CIR
0186                            *
0187 00122    0 10 00176  LNKC  JST   CIH                   Counter 2
0188 00123    000000            BSZ   5
0189 00130    000010            OCT   10
0190 00131    0 000123          DAC   LNKC+1
0191 00132    0 000000    LNRC  DAC   **
0192 00133    0 35 00131        LDX   *-2
0193 00134    0 01 00222        JMP   CIR
0194                            *
0195 00135    0 10 00176  LNKD  JST   CIH                   Counter 3
0196 00136    000000            BSZ   5
0197 00143    000010            OCT   10
0198 00144    0 000136          DAC   LNKD+1
0199 00145    0 000000    LNRD  DAC   **
0200 00146    0 35 00144        LDX   *-2
0201 00147    0 01 00222        JMP   CIR
0202                            *
0203                            * M6800 interrupt links
0204                            *
0205 00150    0 10 00176  LNKE  JST   CIH                   CB2 interrupt
0206 00151    000000            BSZ   5
0207 00156    000010            OCT   10
0208 00157    0 000151          DAC   LNKE+1
0209 00160    0 000000    LNRE  DAC   **
0210 00161    0 35 00157        LDX   *-2
0211 00162    0 01 00222        JMP   CIR
0212                            *
0213 00163    0 10 00176  LNKF  JST   CIH                   CA2 interrupt
0214 00164    000000            BSZ   5
0215 00171    000010            OCT   10
0216 00172    0 000164          DAC   LNKF+1
0217 00173    0 000000    LNRF  DAC   **
0218 00174    0 35 00172        LDX   *-2
0219 00175    0 01 00222        JMP   CIR
0220                            *
0221                            * Common Interrupt Handler
0222                            *
0223 00176    0 000000    CIH   DAC   **
0224 00177   -0 15 00176        STX*  CIH
0225 00200    0 35 00176        LDX   CIH
0226 00201    1 13 00001        IMA   1,1
0227 00202    000043            INK
0228 00203    000005            SGL
```

```
                                *                                PAGE        5
0229 00204    1 04 00002        STA   2,1
0230 00205    0 02 00021        LDA   SKST
0231 00206    1 04 00004        STA   4,1
0232 00207    1 02 00005        LDA   5,1
0233 00210    140401            CMA
0234 00211    0 03 00264        ANA   MASK
0235 00212    0 04 00264        STA   MASK
0236 00213    74 0020           SMK   '20
0237 00214    1 02 00007        LDA   7,1
0238 00215    0 04 00651        STA   CIHA
0239 00216    000201            IAB
0240 00217    1 04 00003        STA   3,1
0241 00220    000011            DXA
0242 00221   -0 01 00651        JMP*  CIHA
0243                            *
0244                            * Common Interrupt Return
0245                            *
0246 00222    1 02 00003 CIR    LDA   3,1
0247 00223    000201            IAB
0248 00224    1 02 00005        LDA   5,1
0249 00225    140401            CMA
0250 00226    001001            INH
0251 00227    0 03 00264        ANA   MASK
0252 00230    1 05 00005        ERA   5,1
0253 00231    0 04 00264        STA   MASK
0254 00232    74 0020           SMK   '20
0255 00233    1 02 00004        LDA   4,1
0256 00234    0 04 00245        STA   CIRA
0257 00235    1 02 00000        LDA   0,1
0258 00236    0 04 00246        STA   CIRX
0259 00237    1 02 00002        LDA   2,1
0260 00240    171020            OTK
0261 00241    1 13 00001        IMA   1,1
0262 00242    0 35 00246        LDX   CIRX
0263 00243    000401            ENB
0264 00244   -0 01 00245        JMP*  CIRA
0265 00245    000000      CIRA  BSZ   1
0266 00246    000000      CIRX  BSZ   1
0267                            *
0268                            * Common Interrupt Initiator
0269                            *
0270 00247    0 000000    CII   DAC   **
0271 00250   -0 35 00247        LDX*  CII
0272 00251    0 12 00247        IRS   CII
0273 00252    1 02 00000        LDA   0,1
0274 00253    140401            CMA
0275 00254    0 03 00264        ANA   MASK
0276 00255    1 05 00000        ERA   0,1
0277 00256    74 0020           SMK   '20
0278 00257    0 04 00264        STA   MASK
0279 00260   -0 02 00247        LDA*  CII
0280 00261    0 12 00247        IRS   CII
0281 00262    1 04 00002        STA   2,1
0282 00263   -0 01 00247        JMP*  CII
0283 00264    000000      MASK  BSZ   1
0284 00265    000000      IFLG  BSZ   1
0285                            *
```

```
                                *                                PAGE        6
286                             * DISPATCHER -- Waiting loop in HADIOS EXEC.
287                             * Background job pointers are placed here.
288                             * SENSE SWITCH 2 is tested for a user
289                             * 'Interrupt' to BASIC command mode.
290                             *
291 00266    0 35 00743 DISP LDX    =-5
292 00267    1 02 00313      LDA    GOAD+5,1
293 00270    100040          SZE
294 00271    0 01 00301      JMP    DISQ
295 00272    100010          SR2
296 00273    0 01 00277      JMP    STOP
297 00274    0 12 00000      IRS    0
298 00275    0 01 00267      JMP    DISP+1
299 00276    0 01 00266      JMP    DISP
300 00277    0 10 03603 STOP JST    ERCL
301 00300    152711          BCI    1,UI
302 00301    0 04 00305 DISQ STA    DISS
303 00302    140040          CRA
304 00303    1 04 00313      STA    GOAD+5,1
305 00304   -0 01 00305      JMP*   DISS
306 00305    000000     DISS BSZ    1
307 00306    000000     GOAD BSZ    5
308                             *
309                             * BASIC entry point via '716 ( JMP* '716 )
310                             * Location CJST contains JST* '515
311                             * Location IBUF contains JST* '515+N
312                             * where N is the Subroutine ref. no.
313                             * For other Subroutines: DAC IBUF in TABL
314                             * Addresses in BASIC CALL table from
315                             * location '522 onwards should be patched
316                             *
317 00313    0 02 00230 CALL LDA    IBUF
318 00314    0 07 00515      SUB    CJST
319 00315    0 04 00000      STA    0
320 00316   -1 01 00316      JMP*   TABL-1,1
321 00317    0 000331   TABL DAC    SUB1
322 00320    0 000601        DAC    SUB2
323 00321    0 000652        DAC    SUB3
324 00322    0 004070        DAC    SUB4
325 00323    0 000230        DAC    IBUF
326 00324    0 000230        DAC    IBUF
327 00325    0 000230        DAC    IBUF
328 00326    0 000230        DAC    IBUF
329 00327    0 000230        DAC    IBUF
330 00330    0 000230        DAC    IBUF
331                             *
332                             * Subroutine 1 --- CALL(1,A(0)(B(0))
333                             * Flag CAL0=1 if H316 in on-line mode
334                             * else zero..
335                             *
336                             * Counter can be used by H316 or M6800
337                             * If MCI (I=1,2,3) is non-zero then Counter
338                             * I is exclusively under micro control
339                             *
340                             * For Counter I,ITMI is the
341                             * interrupt time (N*20 ms) and is upgraded
342                             * every interrupt cycle -- an instantaneous
```

```
                                *                                PAGE      7
0343                            * value..
0344                            *
0345                            * If H316 is off-line ( A(12) or A(13)=1 )
0346                            * Subroutine does not modify location '61
0347                            *
0348 00331    0 02 00034 SUB1   LDA    SIP
0349 00332    0 04 00641        STA    SIP1
0350 00333    0 02 00037        LDA    SBP
0351 00334    0 04 00642        STA    SBP1
0352 00335    0 02 00636        LDA    NEXT
0353 00336    0 04 00340        STA    *+2
0354 00337    0 01 00341        JMP    *+2
0355 00340    0 000000          DAC    **
0356 00341   -0 10 00640        JST*   FAT
0357 00342    000002            DEC    2
0358 00343    000000     PARS   BSZ    2
0359 00345   -0 10 00653        JST*   L22
0360 00346   -0 000343          DAC*   PARS
0361 00347   -0 10 00674        JST*   M22
0362 00350    0 000646          DAC    F50
0363 00351   -0 10 00676        JST*   C21
0364 00352    0 01 00570        JMP    ERI
0365 00353    140407            TCA
0366 00354    0 04 00637        STA    CINT
0367 00355    0 02 00343        LDA    PARS
0368 00356    0 06 00742        ADD    =10
0369 00357    0 04 01533        STA    REP+3
0370 00360    0 12 00343        IRS    PARS
0371 00361    0 12 00343        IRS    PARS
0372 00362    0 35 00741        LDX    =-13
0373 00363    0 15 00645        STX    KT
0374 00364   -0 10 00653        JST*   L22
0375 00365   -0 000343          DAC*   PARS
0376 00366   -0 10 00676        JST*   C21
0377 00367    0 01 00570        JMP    ERI
0378 00370    0 35 00645        LDX    KT
0379 00371    1 04 04342        STA    CTRS+13,1
0380 00372    0 12 00343        IRS    PARS
0381 00373    0 12 00343        IRS    PARS
0382 00374    0 12 00000        IRS    0
0383 00375    0 01 00363        JMP    *-10
0384 00376    0 02 04341        LDA    CTRS+12           A(13)=1 ?
0385 00377    100040            SZE
0386 00400    0 01 01554        JMP    A7+1
0387 00401    0 02 04340        LDA    CTRS+11           A(12)=1 ?
0388 00402    101040            SNZ
0389 00403    0 01 00407        JMP    *+4
0390 00404    0 02 00740        LDA    =-1
0391 00405    0 04 00633        STA    ARR0
0392 00406    0 01 00422        JMP    GPAR
0393 00407    0 02 00737        LDA    =1
0394 00410    0 04 03522        STA    BYTE+1
0395 00411    0 04 00632        STA    CAL0
0396 00412    0 04 00631        STA    HEND
0397 00413    0 02 00061        LDA    '61
0398 00414    0 04 00630        STA    L61
0399 00415    0 10 00706        JST    MTES
```

```
                                *                                       PAGE        8
0400 00416   001001                INH
0401 00417   14 0220               OCP    SCLK
0402 00420   0 02 00634            LDA    FRST
0403 00421   0 04 00061            STA    '61
0404                            *
0405 00422   0 35 00743 GPAR     LDX    =-5                 Clear Dispatch
0406 00423   140040               CRA                        table..
0407 00424   0 04 00265            STA    IFLG
0408 00425   0 04 01573            STA    SCAN
0409 00426   1 04 00313            STA    GOAD+5,1
0410 00427   0 12 00000            IRS    0
0411 00430   0 01 00426            JMP    *-2
0412                            *
0413                            * Interrupt initialisation routines
0414                            * H316 clock first.Check for doubly specifie
0415                            * counters.If A(12)=1 H316 off-line but scan
0416                            * as A(1) multiples of micro scan interval..
0417                            * if A(13)=1 both computers off-line.
0418                            *
0419 00431   0 02 04340            LDA    CTRS+11
0420 00432   100040               SZE
0421 00433   0 01 00566            JMP    INT
0422 00434   0 10 00247            JST    CII
0423 00435   0 000102             DAC    LNRA-2
0424 00436   0 001113             DAC    ACLK
0425 00437   14 0020              OCP    CLK
0426 00440   001001               INH
0427                            *
0428                            * Counter 1 --- Device 2
0429                            *
0430 00441   0 02 04325            LDA    CTRS
0431 00442   0 03 00736            ANA    =2
0432 00443   101040               SNZ
0433 00444   0 01 00455            JMP    A1
0434 00445   0 02 01622            LDA    MC1
0435 00446   101040               SNZ
0436 00447   0 01 00452            JMP    *+3
0437 00450   0 35 00736            LDX    =2
0438 00451   0 10 00572            JST    CERR
0439 00452   0 10 00506            JST    CT1
0440 00453   0 004333             DAC    CTRS+6
0441 00454   0 004332             DAC    CTRS+5
0442                            *
0443                            * Counter 2 --- Device 4
0444                            *
0445 00455   0 02 04325 A1       LDA    CTRS
0446 00456   0 03 00735            ANA    =4
0447 00457   101040               SNZ
0448 00460   0 01 00471            JMP    A2
0449 00461   0 02 01626            LDA    MC2
0450 00462   101040               SNZ
0451 00463   0 01 00466            JMP    *+3
0452 00464   0 35 00735            LDX    =4
0453 00465   0 10 00572            JST    CERR
0454 00466   0 10 00526            JST    CT2
0455 00467   0 004335             DAC    CTRS+8
0456 00470   0 004334             DAC    CTRS+7
```

```
                                *                                PAGE        9
0457                            *
0458                            * Counter 3 --- Device 8
0459                            *
0460 00471    0 02 04325 A2     LDA   CTRS
0461 00472    0 03 00734        ANA   =8
0462 00473    101040            SNZ
0463 00474    0 01 00566        JMP   INT
0464 00475    0 02 01632        LDA   MC3
0465 00476    101040            SNZ
0466 00477    0 01 00502        JMP   *+3
0467 00500    0 35 00734        LDX   =8
0468 00501    0 10 00572        JST   CERR
0469 00502    0 10 00546        JST   CT3
0470 00503    0 004337          DAC   CTRS+10
0471 00504    0 004336          DAC   CTRS+9
0472 00505    0 01 00566        JMP   INT
0473                            *
0474                            * CT1 --- Counter 1 initialisation
0475                            *
0476 00506    0 000000    CT1   DAC   **
0477 00507   -0 35 00506        LDX*  CT1
0478 00510    0 12 00506        IRS   CT1
0479 00511    1 02 00000        LDA   0,1
0480 00512    74 0570           OTA   SET1
0481 00513    0 01 00512        JMP   *-1
0482 00514   -0 35 00506        LDX*  CT1
0483 00515    1 02 00000        LDA   0,1
0484 00516    101040            SNZ
0485 00517    0 01 00524        JMP   *+5
0486 00520    0 10 00247        JST   CII
0487 00521    0 000115          DAC   LNRB-2
0488 00522    0 001000          DAC   ICT1
0489 00523    14 0470           OCP   CTR1
0490 00524    0 12 00506        IRS   CT1
0491 00525   -0 01 00506        JMP*  CT1
0492                            *
0493                            * CT2--- Counter 2 initialisation
0494                            *
0495 00526    0 000000    CT2   DAC   **
0496 00527   -0 35 00526        LDX*  CT2
0497 00530    0 12 00526        IRS   CT2
0498 00531    1 02 00000        LDA   0,1
0499 00532    74 0770           OTA   SET2
0500 00533    0 01 00532        JMP   *-1
0501 00534   -0 35 00526        LDX*  CT2
0502 00535    1 02 00000        LDA   0,1
0503 00536    101040            SNZ
0504 00537    0 01 00544        JMP   *+5
0505 00540    0 10 00247        JST   CII
0506 00541    0 000130          DAC   LNRC-2
0507 00542    0 001031          DAC   ICT2
0508 00543    14 0670           OCP   CTR2
0509 00544    0 12 00526        IRS   CT2
0510 00545   -0 01 00526        JMP*  CT2
0511                            *
0512                            * CT3--- Counter 3 initialisation
0513                            *
```

```
                                *
0514 00546    0 000000    CT3   DAC    **
0515 00547   -0 35 00546        LDX*   CT3
0516 00550    0 12 00546        IRS    CT3
0517 00551    1 02 00000        LDA    0,1
0518 00552   74 0370           OTA    SET3
0519 00553    0 01 00552        JMP    *-1
0520 00554   -0 35 00546        LDX*   CT3
0521 00555    1 02 00000        LDA    0,1
0522 00556   101040            SNZ
0523 00557    0 01 00564        JMP    *+5
0524 00560    0 10 00247        JST    CII
0525 00561    0 000143          DAC    LNRD-2
0526 00562    0 001062          DAC    ICT3
0527 00563   14 0270           OCP    CTR3
0528 00564    0 12 00546        IRS    CT3
0529 00565   -0 01 00546        JMP*   CT3
0530                            *
0531 00566    000401      INT   ENB
0532 00567    0 01 00266        JMP    DISP
0533 00570    0 10 03603  ERI   JST    ERCL
0534 00571    151311            BCI    1,RI
0535 00572    0 000000    CERR  DAC    **
0536 00573    0 02 00000        LDA    0
0537 00574    140401            CMA
0538 00575    0 03 04325        ANA    CTRS
0539 00576    0 04 04325        STA    CTRS
0540 00577    0 10 03603        JST    ERCL
0541 00600    141705            BCI    1,CE
0542                            *
0543                            * Subroutine 2 --- CALL(2)
0544                            * IFLG is normally set to 1 in the clock
0545                            * interrupt response code.
0546                            * If scans=A(2) then retutn to BASIC after
0547                            * CALL(2) else goto DISPATCHER.
0548                            *
0549 00601    001001      SUB2  INH
0550 00602    0 02 00034        LDA    SIP
0551 00603    0 04 00643        STA    SIP2
0552 00604    0 02 00037        LDA    SBP
0553 00605    0 04 00644        STA    SBP2
0554 00606    140040            CRA
0555 00607    0 04 00265        STA    IFLG
0556 00610    0 02 01573        LDA    SCAN
0557 00611    0 07 04326        SUB    CTRS+1
0558 00612    101400            SMI
0559 00613    0 01 00616        JMP    *+3
0560 00614    000401            ENB
0561 00615    0 01 00266        JMP    DISP
0562 00616    0 02 04340        LDA    CTRS+11
0563 00617    0 04 00000        STA    0
0564 00620    140040            CRA
0565 00621    0 04 00631        STA    HEND
0566 00622    0 04 04340        STA    CTRS+11
0567 00623    0 04 04341        STA    CTRS+12
0568 00624    0 02 00000        LDA    0
0569 00625    100040            SZE
0570 00626    0 01 04000        JMP    GFIN
```

```
                                *                                PAGE        11
0571 00627   0 01 03620          JMP   HFIN
0572 00630   000000      L61     BSZ   1
0573 00631   000000      HEND    BSZ   1
0574 00632   000000      CAL0    BSZ   1
0575 00633   000000      ARR0    BSZ   1
0576 00634   177777      FRST    DEC   -1
0577 00635   100000      NMAX    DEC   -32768
0578 00636   0 000231    NEXT    DAC   IBUF+1
0579 00637   000000      CINT    BSZ   1
0580 00640   0 000000    FAT     XAC   F£AT
0581 00641   000000      SIP1    BSZ   1
0582 00642   000000      SBP1    BSZ   1
0583 00643   000000      SIP2    BSZ   1
0584 00644   000000      SBP2    BSZ   1
0585 00645   000000      KT      BSZ   1
0586 00646   041544      F50     DEC   50.
     00647   000000
0587 00650   0 004312    MCKK    DAC   MCTR
0588 00651   000000      CIHA    BSZ   1
0589                     *
0590                     * Subroutine 3 -- CALL (3,N,U)
0591                     * DG0A is common to H316 and M6800
0592                     * 16-bit output truncated to 12-bits,allowin
0593                     * for DAC channel selection ( 0 < N < 15 )
0594                     *
0595 00652   001001      SUB3    INH
0596 00653   0 02 00636          LDA   NEXT
0597 00654   0 04 00656          STA   *+2
0598 00655   0 01 00657          JMP   *+2
0599 00656   0 000000           DAC   **
0600 00657  -0 10 00640          JST*  FAT
0601 00660   000002             DEC   2
0602 00661   000000      N       BSZ   1
0603 00662   000000      U       BSZ   1
0604 00663  -0 10 00653          JST*  L22
0605 00664  -0 000661           DAC*  N
0606 00665  -0 10 00676          JST*  C21
0607 00666   0 01 00570          JMP   ERI
0608 00667   0 04 00705          STA   CH3
0609 00670  -0 10 00653          JST*  L22
0610 00671  -0 000662           DAC*  U
0611 00672  -0 10 00676          JST*  C21
0612 00673   0 01 00570          JMP   ERI
0613 00674   0415 77             ALS   1
0614 00675   000201             IAB
0615 00676   0 02 00705          LDA   CH3
0616 00677   0400 74             LRL   4
0617 00700   000201             IAB
0618 00701   74 1370            OTA   DG0A
0619 00702   0 01 00701          JMP   *-1
0620 00703   000401             ENB
0621 00704  -0 01 01572          JMP*  CL06
0622 00705   000000      CH3     BSZ   1
0623                     *
0624 00706   0 000000    MTES    DAC   **          MTES and MCX
0625 00707   0 02 01622          LDA   MC1         are Counter
0626 00710   101040             SNZ               related routine
```

```
                                *
0627 00711    0 01 00714        JMP    *+3
0628 00712    0 10 00727        JST    MCX
0629 00713    177775            DEC    -3
0630 00714    0 02 01626        LDA    MC2
0631 00715    101040            SNZ
0632 00716    0 01 00721        JMP    *+3
0633 00717    0 10 00727        JST    MCX
0634 00720    177776            DEC    -2
0635 00721    0 02 01632        LDA    MC3
0636 00722    101040            SNZ
0637 00723    0 01 00726        JMP    *+3
0638 00724    0 10 00727        JST    MCX
0639 00725    177777            DEC    -1
0640 00726   -0 01 00706        JMP*   MTES
0641                            *
0642 00727    0 000000    MCX   DAC    **
0643 00730   -0 35 00727        LDX*   MCX
0644 00731    0 10 04266        JST    TIME
0645 00732    0 12 00727        IRS    MCX
0646 00733   -0 01 00727        JMP*   MCX
0647 00734    000010            FIN
     00735    000004
     00736    000002
     00737    000001
     00740    177777
     00741    177763
     00742    000012
     00743    177773
     00744    040000
     00745    177766

0648          000746      BAS0  EQU    *
0649                            ORG    '1000
0650                            SETB   BAS1
0651                        *
0652                        * Counter interrupt response code -- 1
0653                        *
0654 01000    14 0570       ICT1  OCP    SET1
0655 01001    0 02 01622          LDA    MC1
0656 01002    101040              SNZ
0657 01003    0 01 01006          JMP    *+3
0658 01004    0 02 04317          LDA    MCTR+5
0659 01005    0 04 04333          STA    CTRS+6
0660 01006    0 02 04333          LDA    CTRS+6
0661 01007    74 0570             OTA    SET1
0662 01010    0 01 01007          JMP    *-1
0663 01011    14 0470             OCP    CTR1
0664 01012    0 12 01030          IRS    FLG1
0665                        *
0666 01013    0 02 00061          LDA    '61
0667 01014    0 04 00000          STA    0
0668 01015    0 07 01026          SUB    TIM1
0669 01016    0 06 01025          ADD    ADT1
0670 01017    0 04 01027          STA    ITM1
0671                        *
0672 01020    0 15 01026          STX    TIM1
0673 01021    140040              CRA
```

```
                                *
0674 01022   0 04 01025         STA   ADT1
0675                            *
0676 01023   0 10 00117         JST   LNRB
0677 01024   0 01 01000         JMP   ICT1
0678 01025   000000        ADT1  BSZ   1
0679 01026   000000        TIM1  BSZ   1
0680 01027   000000        ITM1  BSZ   1
0681 01030   000000        FLG1  BSZ   1
0682                            *
0683                            * Counter response code -- 2
0684                            *
0685 01031   14 0770        ICT2  OCP   SET2
0686 01032   0 02 01626          LDA   MC2
0687 01033   101040             SNZ
0688 01034   0 01 01037         JMP   *+3
0689 01035   0 02 04321         LDA   MCTR+7
0690 01036   0 04 04335         STA   CTRS+8
0691 01037   0 02 04335         LDA   CTRS+8
0692 01040   74 0770            OTA   SET2
0693 01041   0 01 01040         JMP   *-1
0694 01042   14 0670            OCP   CTR2
0695 01043   0 12 01061         IRS   FLG2
0696                            *
0697 01044   0 02 00061         LDA   '61
0698 01045   0 04 00000         STA   0
0699 01046   0 07 01057         SUB   TIM2
0700 01047   0 06 01056         ADD   ADT2
0701 01050   0 04 01060         STA   ITM2
0702                            *
0703 01051   0 15 01057         STX   TIM2
0704 01052   140040            CRA
0705 01053   0 04 01056         STA   ADT2
0706                            *
0707 01054   0 10 00132         JST   LNRC
0708 01055   0 01 01031         JMP   ICT2
0709 01056   000000        ADT2  BSZ   1
0710 01057   000000        TIM2  BSZ   1
0711 01060   000000        ITM2  BSZ   1
0712 01061   000000        FLG2  BSZ   1
0713                            *
0714                            * Counter response code -- 3
0715                            *
0716 01062   14 0370        ICT3  OCP   SET3
0717 01063   0 02 01632          LDA   MC3
0718 01064   101040             SNZ
0719 01065   0 01 01070         JMP   *+3
0720 01066   0 02 04323         LDA   MCTR+9
0721 01067   0 04 04337         STA   CTRS+10
0722 01070   0 02 04337         LDA   CTRS+10
0723 01071   74 0370            OTA   SET3
0724 01072   0 01 01071         JMP   *-1
0725 01073   14 0270            OCP   CTR3
0726 01074   0 12 01112         IRS   FLG3
0727                            *
0728 01075   0 02 00061         LDA   '61
0729 01076   0 04 00000         STA   0
0730 01077   0 07 01110         SUB   TIM3
```

```
                                *                                PAGE      14
0731 01100   0 06 01107          ADD   ADT3
0732 01101   0 04 01111          STA   ITM3
0733                            *
0734 01102   0 15 01110          STX   TIM3
0735 01103   140040             CRA
0736 01104   0 04 01107          STA   ADT3
0737 01105   0 10 00145          JST   LNRD
0738 01106   0 01 01062          JMP   ICT3
0739 01107   000000       ADT3  BSZ   1
0740 01110   000000       TIM3  BSZ   1
0741 01111   000000       ITM3  BSZ   1
0742 01112   000000       FLG3  BSZ   1
0743                            *
0744                            * H316 Clock interrupt response code
0745                            *
0746 01113   140040       ACLK  CRA
0747 01114   0 04 00630          STA   L61
0748 01115   0 10 01576          JST   HTES
0749 01116   0 02 03522          LDA   BYTE+1
0750 01117   101040             SNZ
0751 01120   0 01 01124          JMP   *+4
0752 01121   140040             CRA
0753 01122   0 04 03522          STA   BYTE+1
0754 01123   0 01 01125          JMP   *+2
0755 01124   0 10 00706          JST   MTES
0756 01125   14 0220             OCP   SCLK
0757 01126   0 02 00637          LDA   CINT
0758 01127   0 04 00061          STA   '61
0759 01130   14 0020             OCP   CLK
0760 01131   000401             ENB
0761 01132   0 02 00101          LDA   LNKA+5
0762 01133   0 07 01547          SUB   BSTP
0763 01134   100400             SPL
0764 01135   0 01 01574          JMP   STP
0765 01136   0 02 00265          LDA   IFLG
0766 01137   100040             SZE
0767 01140   0 01 01574          JMP   STP
0768 01141   0 12 00265          IRS   IFLG
0769                            *
0770                            * Counter Inputs --- H316 only
0771                            *
0772 01142   001001       A3    INH
0773 01143   0 02 04325          LDA   CTRS
0774 01144   0 03 01677          ANA   =2
0775 01145   101040             SNZ
0776 01146   0 01 01207          JMP   A4
0777 01147   140040             CRA
0778 01150   54 0470             INA   CTR1
0779 01151   0 01 01150          JMP   *-1
0780 01152   000201             IAB
0781 01153   0 02 04333          LDA   CTRS+6
0782 01154   74 0570             OTA   SET1
0783 01155   0 01 01154          JMP   *-1
0784 01156   000401             ENB
0785 01157   0 02 00344          LDA   PARS+1
0786 01160   0 06 01676          ADD   =48
0787 01161   0 06 01676          ADD   =48
```

```
                              *
0788 01162    0 04 01200          STA   B48
0789 01163    140442              OCT   140442
0790 01164    0 04 01172          STA   B49
0791 01165    140442              OCT   140442
0792 01166    0 04 01204          STA   B50
0793 01167    000201              IAB
0794 01170   -0 10 00675          JST*  C12
0795 01171   -0 10 00654          JST*  H22
0796 01172    0 000000    B49     DAC   **
0797 01173    0 02 01027          LDA   ITM1
0798 01174   -0 10 00675          JST*  C12
0799 01175   -0 10 00670          JST*  D22
0800 01176    0 000646            DAC   F50
0801 01177   -0 10 00654          JST*  H22
0802 01200    0 000000    B48     DAC   **
0803 01201    0 02 01030          LDA   FLG1
0804 01202   -0 10 00675          JST*  C12
0805 01203   -0 10 00654          JST*  H22
0806 01204    0 000000    B50     DAC   **
0807 01205    140040              CRA
0808 01206    0 04 01030          STA   FLG1
0809                          *
0810 01207    000401      A4      ENB
0811 01210    001001              INH
0812 01211    0 02 04325          LDA   CTRS
0813 01212    0 03 01675          ANA   =4
0814 01213    101040              SNZ
0815 01214    0 01 01255          JMP   A5
0816 01215    140040              CRA
0817 01216    54 0670             INA   CTR2
0818 01217    0 01 01216          JMP   *-1
0819 01220    000201              IAB
0820 01221    0 02 04335          LDA   CTRS+8
0821 01222    74 0770             OTA   SET2
0822 01223    0 01 01222          JMP   *-1
0823 01224    000401              ENB
0824 01225    0 02 00344          LDA   PARS+1
0825 01226    0 06 01674          ADD   =51
0826 01227    0 06 01674          ADD   =51
0827 01230    0 04 01246          STA   B51
0828 01231    140442              OCT   140442
0829 01232    0 04 01240          STA   B52
0830 01233    140442              OCT   140442
0831 01234    0 04 01252          STA   B53
0832 01235    000201              IAB
0833 01236   -0 10 00675          JST*  C12
0834 01237   -0 10 00654          JST*  H22
0835 01240    0 000000    B52     DAC   **
0836 01241    0 02 01060          LDA   ITM2
0837 01242   -0 10 00675          JST*  C12
0838 01243   -0 10 00670          JST*  D22
0839 01244    0 000646            DAC   F50
0840 01245   -0 10 00654          JST*  H22
0841 01246    0 000000    B51     DAC   **
0842 01247    0 02 01061          LDA   FLG2
0843 01250   -0 10 00675          JST*  C12
0844 01251   -0 10 00654          JST*  H22
```

```
                                *
0845 01252    0 000000    B53   DAC    **
0846 01253    140040            CRA
0847 01254    0 04 01061        STA    FLG2
0848                            *
0849 01255    000401      A5    ENB
0850 01256    001001            INH
0851 01257    0 02 04325        LDA    CTRS
0852 01260    0 03 01673        ANA    =8
0853 01261    101040            SNZ
0854 01262    0 01 01323        JMP    A6
0855 01263    140040            CRA
0856 01264    54 0270           INA    CTR3
0857 01265    0 01 01264        JMP    *-1
0858 01266    000201            IAB
0859 01267    0 02 04337        LDA    CTRS+10
0860 01270    74 0370           OTA    SET3
0861 01271    0 01 01270        JMP    *-1
0862 01272    000401            ENB
0863 01273    0 02 00344        LDA    PARS+1
0864 01274    0 06 01672        ADD    =54
0865 01275    0 06 01672        ADD    =54
0866 01276    0 04 01314        STA    B54
0867 01277    140442            OCT    140442
0868 01300    0 04 01306        STA    B55
0869 01301    140442            OCT    140442
0870 01302    0 04 01320        STA    B56
0871 01303    000201            IAB
0872 01304   -0 10 00675        JST*   C12
0873 01305   -0 10 00654        JST*   H22
0874 01306    0 000000    B55   DAC    **
0875 01307    0 02 01111        LDA    ITM3
0876 01310   -0 10 00675        JST*   C12
0877 01311   -0 10 00670        JST*   D22
0878 01312    0 000646          DAC    F50
0879 01313   -0 10 00654        JST*   H22
0880 01314    0 000000    B54   DAC    **
0881 01315    0 02 01112        LDA    FLG3
0882 01316   -0 10 00675        JST*   C12
0883 01317   -0 10 00654        JST*   H22
0884 01320    0 000000    B56   DAC    **
0885 01321    140040            CRA
0886 01322    0 04 01112        STA    FLG3
0887 01323    000401      A6    ENB
0888                            *
0889                            * Analogue Inputs scan --- H316 only
0890                            *
0891 01324    0 02 04325        LDA    CTRS
0892 01325    0 03 01671        ANA    =1
0893 01326    101040            SNZ
0894 01327    0 01 01553        JMP    A7
0895 01330    0 35 01670        LDX    =-48
0896 01331    140040            CRA
0897 01332   -0 04 01431        STA*   AB48
0898 01333    0 12 00000        IRS    0
0899 01334    0 01 01332        JMP    *-2
0900 01335    0 02 04331        LDA    CTRS+4
0901 01336    140407            TCA
```

```
                                *                                PAGE        17
0902 01337    0 04 01430            STA    HCTS
0903 01340    0 02 04330            LDA    CTRS+3
0904 01341    0 06 01432            ADD    AB48+1
0905 01342    0 04 01433            STA    AB48+2
0906 01343    0 02 04330            LDA    CTRS+3
0907 01344    0 07 04327            SUB    CTRS+2
0908 01345    141206                AOA
0909 01346    140407                TCA
0910 01347    001001                INH
0911                          *
0912                          * Sampling spaced over subintervals of 20 ms
0913                          * Effect of a simple filter...
0914                          *
0915 01350    0 04 01550 HPC1 STA    CHXR
0916 01351    140040                CRA
0917 01352    0 04 01514            STA    ENS
0918 01353    0 02 00061            LDA    '61
0919 01354    0 04 01426            STA    HREG
0920 01355    0 02 00061            LDA    '61
0921 01356    0 04 01427            STA    HREG+1
0922 01357    0 05 01426            ERA    HREG
0923 01360    101040                SNZ
0924 01361    0 01 01355            JMP    *-4
0925 01362    0 35 01550 ESBL LDX    CHXR
0926 01363    0 02 04327            LDA    CTRS+2
0927 01364    0 04 01515            STA    STUP
0928 01365    0 02 01515 MUX  LDA    STUP
0929 01366    74 0170               OTA    ANAG
0930 01367    0 01 01366            JMP    *-1
0931 01370    14 0070               OCP    DATA
0932 01371    140040                CRA
0933 01372    54 0070               INA    DATA
0934 01373    0 01 01372            JMP    *-1
0935 01374    0404 72               LGR    6
0936 01375   -0 06 01433            ADD*   AB48+2
0937 01376   -0 04 01433            STA*   AB48+2
0938 01377    0 12 01515            IRS    STUP
0939 01400    0 12 00000            IRS    0
0940 01401    0 01 01365            JMP    MUX
0941 01402    0 12 01514 HPC2 IRS    ENS
0942 01403    0 02 01667            LDA    =31
0943 01404    0 07 01514            SUB    ENS
0944 01405    101400                SMI
0945 01406    0 01 01410            JMP    *+2
0946 01407    0 01 01414            JMP    *+5
0947 01410    0 02 00061            LDA    '61
0948 01411    0 05 01427            ERA    HREG+1
0949 01412    101040                SNZ
0950 01413    0 01 01421            JMP    DLY
0951 01414    0 02 01514            LDA    ENS
0952 01415   -0 10 00675            JST*   C12
0953 01416   -0 10 00654            JST*   H22
0954 01417   -0 001533             DAC*   REP+3
0955 01420    0 01 01516            JMP    A5A
0956 01421    0 02 01430 DLY  LDA    HCTS
0957 01422    0 04 01426            STA    HREG
0958 01423    0 12 01426            IRS    HREG
```

```
                                *                                PAGE      18
0959 01424   0 01 01423          JMP   *-1
0960 01425   0 01 01362          JMP   ESBL
0961 01426   000000      HREG    BSZ   2
0962 01430   000000      HCTS    BSZ   1
0963 01431   1 001514    AB48    DAC   ABUF+48,1
0964 01432   1 001435            DAC   ABUF+1,1
0965 01433   000000              BSZ   1
0966 01434   000000      ABUF    BSZ   48
0967 01514   000000      ENS     BSZ   1
0968 01515   000000      STUP    BSZ   1
0969                     *
0970                     * Analogue Data processing --- H316
0971                     *
0972 01516   000401      A5A     ENB
0973 01517   0 02 04327          LDA   CTRS+2
0974 01520   0 06 01544          ADD   BF0
0975 01521   0 04 01545          STA   ADBF
0976 01522   0 02 00344          LDA   PARS+1
0977 01523   0 06 04327          ADD   CTRS+2
0978 01524   0 06 04327          ADD   CTRS+2
0979 01525   0 04 01546          STA   DBF
0980 01526   0 35 01550          LDX   CHXR
0981 01527   0 15 01551          STX   X1
0982 01530  -0 02 01545 REP      LDA*  ADBF
0983 01531  -0 10 00675          JST*  C12
0984 01532  -0 10 00670          JST*  D22
0985 01533   0 000000           DAC   **
0986 01534  -0 10 00654          JST*  H22
0987 01535  -0 001546           DAC*  DBF
0988 01536   0 12 01545          IRS   ADBF
0989 01537   0 12 01546          IRS   DBF
0990 01540   0 12 01546          IRS   DBF
0991 01541   0 12 01551          IRS   X1
0992 01542   0 01 01530          JMP   REP
0993 01543   0 01 01553          JMP   A7
0994 01544   0 001434    BF0     DAC   ABUF
0995 01545   000000      ADBF    BSZ   1
0996 01546   000000      DBF     BSZ   1
0997 01547   0 000021    BSTP    DAC   SKST
0998 01550   000000      CHXR    BSZ   1
0999 01551   000000      X1      BSZ   1
1000 01552   000000      A20     BSZ   1
1001                     *
1002                     * Update H316 scan ---- VEC2=CL06
1003                     *
1004 01553   0 12 01573 A7       IRS   SCAN
1005 01554   0 02 00641          LDA   SIP1
1006 01555   0 04 00034          STA   SIP
1007 01556   0 02 00642          LDA   SBP1
1008 01557   0 04 00037          STA   SBP
1009 01560   0 02 04341          LDA   CTRS+12
1010 01561   100040             SZE
1011 01562  -0 01 01572          JMP*  CL06
1012 01563   0 02 04340          LDA   CTRS+11
1013 01564   100040             SZE
1014 01565  -0 01 01572          JMP*  CL06
1015 01566   0 02 01572          LDA   CL06
```

```
                                *
1016 01567   0 04 00307         STA   GOAD+1
1017 01570   0 10 00104         JST   LNRA
1018 01571   0 01 01113         JMP   ACLK
1019 01572   004013       CL06 OCT   4013
1020 01573   000000       SCAN BSZ   1
1021 01574   0 10 03603   STP  JST   ERCL
1022 01575   152306             BCI   1,TF
1023 01576   0 000000     HTES DAC   **
1024 01577   0 02 04325         LDA   CTRS
1025 01600   0 03 01677         ANA   =2
1026 01601   101040             SNZ
1027 01602   0 01 01605         JMP   *+3
1028 01603   0 10 00727         JST   MCX
1029 01604   177775             DEC   -3
1030 01605   0 02 04325         LDA   CTRS
1031 01606   0 03 01675         ANA   =4
1032 01607   101040             SNZ
1033 01610   0 01 01613         JMP   *+3
1034 01611   0 10 00727         JST   MCX
1035 01612   177776             DEC   -2
1036 01613   0 02 04325         LDA   CTRS
1037 01614   0 03 01673         ANA   =8
1038 01615   101040             SNZ
1039 01616   0 01 01621         JMP   *+3
1040 01617   0 10 00727         JST   MCX
1041 01620   177777             DEC   -1
1042 01621   -0 01 01576        JMP*  HTES
1043 01622   000000       MC1  BSZ   4
1044 01626   000000       MC2  BSZ   4
1045 01632   000000       MC3  BSZ   4
1046 01636   000000       TZER BSZ   1
1047                      *
1048                      * M6800 interrupts response code
1049                      * CALM=1 means M6800 in communication mode
1050                      * ie. ON or OFF-line else CALM=0
1051                      *
1052 01637   000401       CB2  ENB
1053 01640   0 02 00001         LDA   M680
1054 01641   100040             SZE
1055 01642   0 01 02460         JMP   CB22
1056 01643   54 1070           INA   DGIB
1057 01644   0 01 01643         JMP   *-1
1058 01645   141050            CAL
1059 01646   -0 04 02572        STA*  ABF0
1060 01647   0 04 02367         STA   CFLG+1
1061 01650   0 10 04021         JST   NCEK
1062 01651   0 12 02572         IRS   ABF0
1063 01652   0 12 01552         IRS   A20
1064 01653   0 01 02635         JMP   CB1
1065 01654   0 02 01666         LDA   =-10
1066 01655   0 04 01552         STA   A20
1067 01656   0 02 02372         LDA   REAL+2
1068 01657   0 04 02415         STA   REP1+3
1069 01660   0 02 01671         LDA   =1
1070 01661   0 04 02151         STA   CALM
1071 01662   0 04 00001         STA   M680
1072 01663   0 04 02365         STA   XXX
```

```
                                *
1073 01664   140040             CRA
1074 01665   0 01 02000          JMP   SEC2
1075 01666   177766             FIN
     01667   000037
     01670   177720
     01671   000001
     01672   000066
     01673   000010
     01674   000063
     01675   000004
     01676   000060
     01677   000002

1076         001700       BAS1  EQU   *
1077                             ORG   '2000
1078                             SETB  BAS2
1079 02000   0 04 03525   SEC2  STA   TEST
1080 02001   0 04 03523         STA   WAS
1081 02002   0 04 03170         STA   XCAT
1082 02003   0 04 02566         STA   PAS1
1083 02004   0 04 02146         STA   EFLG
1084 02005   0 04 02147         STA   ERRM
1085 02006   0 04 02150         STA   MCOD
1086 02007   0 04 02567         STA   PAS2
1087 02010   0 04 02570         STA   PAS3
1088 02011   0 04 02571         STA   PAS4
1089 02012   0 04 02366         STA   CFLG
1090 02013   0 04 02364         STA   ABIM
1091 02014   0 04 01636         STA   TZER
1092 02015   0 02 04324         LDA   MACS
1093 02016   0 04 02572         STA   ABF0
1094                        *
1095 02017   0 02 04312   M0    LDA   MCTR
1096 02020   0 03 02712         ANA   =1
1097 02021   100040             SZE
1098 02022   0 01 02052         JMP   M1
1099 02023   0 02 02712         LDA   =1
1100 02024   0 04 02366         STA   CFLG
1101 02025   0 02 04312         LDA   MCTR
1102 02026   0 03 02711         ANA   =2
1103 02027   101040             SNZ
1104 02030   0 01 02034         JMP   *+4
1105 02031   0 02 02710         LDA   =-3
1106 02032   0 04 02364         STA   ABIM
1107 02033   0 01 02056         JMP   M1+4
1108 02034   0 02 04312   M0X   LDA   MCTR
1109 02035   0 03 02707         ANA   =4
1110 02036   101040             SNZ
1111 02037   0 01 02043         JMP   *+4
1112 02040   0 02 02706         LDA   =-2
1113 02041   0 04 02364         STA   ABIM
1114 02042   0 01 02075         JMP   M2+4
1115 02043   0 02 04312         LDA   MCTR
1116 02044   0 03 02705         ANA   =8
1117 02045   101040             SNZ
1118 02046   000000             HLT
1119 02047   0 02 02704         LDA   =-1
```

```
                                *
1120 02050    0 04 02364        STA   ABIM
1121 02051    0 01 02114        JMP   M3+4
1122 02052    0 02 04312 M1     LDA   MCTR
1123 02053    0 03 02711        ANA   =2
1124 02054    101040            SNZ
1125 02055    0 01 02071        JMP   M2
1126 02056    0 10 02130        JST   CHEC
1127 02057    000002            DEC   2
1128 02060    0 02 02146        LDA   EFLG
1129 02061    100040            SZE
1130 02062    0 01 02071        JMP   M2
1131 02063    0 12 01622        IRS   MC1
1132 02064    0 10 00506        JST   CT1
1133 02065    0 004317          DAC   MCTR+5
1134 02066    0 004316          DAC   MCTR+4
1135 02067    0 10 02152        JST   ML61
1136 02070    0 001026          DAC   TIM1
1137                            *
1138 02071    0 02 04312 M2     LDA   MCTR
1139 02072    0 03 02707        ANA   =4
1140 02073    101040            SNZ
1141 02074    0 01 02110        JMP   M3
1142 02075    0 10 02130        JST   CHEC
1143 02076    000004            DEC   4
1144 02077    0 02 02146        LDA   EFLG
1145 02100    100040            SZE
1146 02101    0 01 02110        JMP   M3
1147 02102    0 12 01626        IRS   MC2
1148 02103    0 10 00526        JST   CT2
1149 02104    0 004321          DAC   MCTR+7
1150 02105    0 004320          DAC   MCTR+6
1151 02106    0 10 02152        JST   ML61
1152 02107    0 001057          DAC   TIM2
1153 02110    0 02 04312 M3     LDA   MCTR
1154 02111    0 03 02705        ANA   =8
1155 02112    101040            SNZ
1156 02113    0 01 02635        JMP   CB1
1157 02114    0 10 02130        JST   CHEC
1158 02115    000010            DEC   8
1159 02116    0 02 02146        LDA   EFLG
1160 02117    100040            SZE
1161 02120    0 01 02635        JMP   CB1
1162 02121    0 12 01632        IRS   MC3
1163 02122    0 10 00546        JST   CT3
1164 02123    0 004323          DAC   MCTR+9
1165 02124    0 004322          DAC   MCTR+8
1166 02125    0 10 02152        JST   ML61
1167 02126    0 001110          DAC   TIM3
1168 02127    0 01 02635        JMP   CB1
1169                        *
1170                        * Check for doubly specified counters
1171                        *
1172 02130    0 000000    CHEC  DAC   **
1173 02131    0 02 00632        LDA   CAL0
1174 02132    101040            SNZ
1175 02133    0 01 02144        JMP   *+9
1176 02134   -0 02 02130        LDA*  CHEC
```

```
                                *                                PAGE       22
1177 02135    0 03 04325        ANA    CTRS
1178 02136    101040            SNZ
1179 02137    0 01 02144        JMP    *+5
1180 02140    0 02 02712        LDA    =1
1181 02141    0 04 02146        STA    EFLG
1182 02142    0 04 02147        STA    ERRM
1183 02143    0 04 02150        STA    MCOD
1184 02144    0 12 02130        IRS    CHEC
1185 02145   -0 01 02130        JMP*   CHEC
1186 02146    000000      EFLG  BSZ    1
1187 02147    000000      ERRM  BSZ    1
1188 02150    000000      MCOD  BSZ    1
1189 02151    000000      CALM  BSZ    1
1190                      *
1191 02152    0 000000    ML61  DAC    **
1192 02153    0 02 00061        LDA    '61
1193 02154   -0 35 02152        LDX*   ML61
1194 02155    1 04 00000        STA    0,1
1195 02156    0 12 02152        IRS    ML61
1196 02157   -0 01 02152        JMP*   ML61
1197                      *
1198                      * CA2 Interrupt response:in a M6800 process
1199                      * scan,only first CA2 accesses Analogue
1200                      * Inputs and Counter(s)..the rest just
1201                      * read the bytes..Counter inputs are scan
1202                      * first if any ..
1203                      *
1204 02160    000401      CA2   ENB
1205 02161    0 02 04221        LDA    PAS
1206 02162    100040            SZE
1207 02163    0 01 03576        JMP    CA2C
1208 02164    0 02 02365        LDA    XXX
1209 02165    101040            SNZ
1210 02166    0 01 02646        JMP    CA2R
1211 02167    0 10 03076        JST    GRAF
1212 02170    0 02 02366        LDA    CFLG
1213 02171    0 11 02703        CAS    =0
1214 02172    0 01 02446        JMP    Y123
1215 02173    0 10 03246        JST    A15
1216 02174    101000            NOP
1217 02175    140040      CA2S  CRA
1218 02176    0 35 02702        LDX    =-48
1219 02177   -0 04 02276        STA*   ABM
1220 02200    0 12 00000        IRS    0
1221 02201    0 01 02177        JMP    *-2
1222 02202    0 02 04313        LDA    MCTR+1
1223 02203    140407            TCA
1224 02204    0 04 02275        STA    MCTS
1225 02205    0 02 04315        LDA    MCTR+3
1226 02206    0 06 02277        ADD    ABM+1
1227 02207    0 04 02300        STA    ABM+2
1228 02210    0 02 04315        LDA    MCTR+3
1229 02211    0 07 04314        SUB    MCTR+2
1230 02212    141206            AOA
1231 02213    140407            TCA
1232                      *
1233                      * Sampling spaced over subintervals of 20ms.
```

```
                                *                                PAGE       23
1234                            * Effect of a simple filter.Maximum ensemble
1235                            * number is 32.Dynamic value returned in
1236                            * A(5).Upper limit in 16-bit integer buffer
1237                            * is 32*1023 ... similar for H316
1238                            *
1239 02214   001001                    INH
1240 02215   0 04 02574 MPC1  STA     CHXM
1241 02216   140040                    CRA
1242 02217   0 04 02361               STA     ENSM
1243 02220   0 02 00061               LDA     '61
1244 02221   0 04 02273               STA     MREG
1245 02222   0 02 00061               LDA     '61
1246 02223   0 04 02274               STA     MREG+1
1247 02224   0 05 02273               ERA     MREG
1248 02225   101040                   SNZ
1249 02226   0 01 02222               JMP     *-4
1250 02227   0 35 02574 ESBM  LDX     CHXM
1251 02230   0 02 04314               LDA     MCTR+2
1252 02231   0 04 02362               STA     SETW
1253 02232   0 02 02362 MUXM  LDA     SETW
1254 02233   74 0170                  OTA     ANAG
1255 02234   0 01 02233               JMP     *-1
1256 02235   14 0070                  OCP     DATA
1257 02236   140040                   CRA
1258 02237   54 0070                  INA     DATA
1259 02240   0 01 02237               JMP     *-1
1260 02241   0404 72                  LGR     6
1261 02242  -0 06 02300               ADD*    ABM+2
1262 02243  -0 04 02300               STA*    ABM+2
1263 02244   0 12 02362               IRS     SETW
1264 02245   0 12 00000               IRS     0
1265 02246   0 01 02232               JMP     MUXM
1266 02247   0 12 02361 MPC2  IRS     ENSM
1267 02250   0 02 02701               LDA     =31
1268 02251   0 07 02361               SUB     ENSM
1269 02252   101400                   SMI
1270 02253   0 01 02255               JMP     *+2
1271 02254   0 01 02261               JMP     *+5
1272 02255   0 02 00061               LDA     '61
1273 02256   0 05 02274               ERA     MREG+1
1274 02257   101040                   SNZ
1275 02260   0 01 02266               JMP     DLYM
1276 02261   0 02 02361               LDA     ENSM
1277 02262  -0 10 04013               JST*    MC12
1278 02263  -0 10 04016               JST*    MH22
1279 02264  -0 002415                DAC*    REP1+3
1280 02265   0 01 02374               JMP     A9
1281 02266   0 02 02275 DLYM  LDA     MCTS
1282 02267   0 04 02273               STA     MREG
1283 02270   0 12 02273               IRS     MREG
1284 02271   0 01 02270               JMP     *-1
1285 02272   0 01 02227               JMP     ESBM
1286 02273   000000      MREG  BSZ     2
1287 02275   000000      MCTS  BSZ     1
1288 02276   1 002361    ABM   DAC     MBUF+48,1
1289 02277   1 002302          DAC     MBUF+1,1
1290 02300   000000            BSZ     1
```

```
                                *                                PAGE      24
1291 02301   000000      MBUF BSZ   48
1292 02361   000000      ENSM BSZ   1
1293 02362   000000      SETW BSZ   1
1294 02363   000000      XCTR BSZ   1
1295 02364   000000      ABIM BSZ   1
1296 02365   000000      XXX  BSZ   1
1297 02366   000000      CFLG BSZ   2
1298 02370   000000      REAL BSZ   2
1299 02372   0 002370         DAC   REAL
1300 02373   0 002361         DAC   ENSM
1301                     *
1302                     * If A(12)=1 variables stored in BASIC
1303                     * B-array as well..Index register can't be
1304                     * used as a counter here .. used in Maths
1305                     * routines ..
1306                     *
1307 02374   000401      A9   ENB
1308 02375   0 02 04340       LDA   CTRS+11
1309 02376   101040           SNZ
1310 02377   0 01 02404       JMP   *+5
1311 02400   0 02 00344       LDA   PARS+1
1312 02401   0 06 04314       ADD   MCTR+2
1313 02402   0 06 04314       ADD   MCTR+2
1314 02403   0 04 02455       STA   ABF4
1315 02404   0 02 04314       LDA   MCTR+2
1316 02405   0 06 02453       ADD   BFM
1317 02406   0 04 02454       STA   ABF2
1318 02407   0 35 02574       LDX   CHXM
1319 02410   0 15 02452       STX   CHNO
1320 02411   0 15 02363       STX   XCTR
1321 02412  -0 02 02454 REP1  LDA*  ABF2
1322 02413  -0 10 04013       JST*  MC12
1323 02414  -0 10 04017       JST*  MD22
1324 02415   0 000000         DAC   **
1325 02416   0 04 02456       STA   AAA
1326 02417   000201           IAB
1327 02420   0 04 02457       STA   BBB
1328 02421   0 02 04340       LDA   CTRS+11
1329 02422   101040           SNZ
1330 02423   0 01 02433       JMP   OSAI
1331 02424   0 02 02457       LDA   BBB
1332 02425   000201           IAB
1333 02426   0 02 02456       LDA   AAA
1334 02427  -0 10 04016       JST*  MH22
1335 02430  -0 002455         DAC*  ABF4
1336 02431   0 12 02455       IRS   ABF4
1337 02432   0 12 02455       IRS   ABF4
1338 02433   0 02 02457 OSAI  LDA   BBB
1339 02434   000201           IAB
1340 02435   0 02 02456       LDA   AAA
1341 02436  -0 10 04014       JST*  MC21
1342 02437  -0 04 02454       STA*  ABF2
1343 02440   0 12 02454       IRS   ABF2
1344 02441   0 12 02363       IRS   XCTR
1345 02442   0 01 02412       JMP   REP1
1346 02443   0 02 04314       LDA   MCTR+2
1347 02444   0 06 02453       ADD   BFM
```

```
                                *                                PAGE        25
1348 02445   0 04 02454        STA   ABF2
1349 02446   140040      Y123  CRA
1350 02447   0 04 02365        STA   XXX
1351 02450   0 12 01636        IRS   TZER
1352 02451   0 01 02646        JMP   CA2R
1353 02452   000000      CHN0  BSZ   1
1354 02453   0 002301    BFM   DAC   MBUF
1355 02454   000000      ABF2  BSZ   1
1356 02455   000000      ABF4  BSZ   1
1357 02456   000000      AAA   BSZ   1
1358 02457   000000      BBB   BSZ   1
1359                     *
1360                     * M6800 control output routine .. also used
1361                     * in M6800 to H316 data transfer except
1362                     * that F254=1 in the latter case ..
1363                     *
1364 02460   54 1070     CB22  INA   DGIB
1365 02461   0 01 02460       JMP   *-1
1366 02462   141050           CAL
1367 02463   0 04 02367       STA   CFLG+1
1368 02464   0 04 00000       STA   0
1369 02465   000201           IAB
1370 02466   0 10 04021       JST   NCEK
1371 02467   0 02 02566       LDA   PAS1
1372 02470   100040           SZE
1373 02471   0 01 02540       JMP   PROC
1374 02472   0 02 04043       LDA   F254
1375 02473   101040           SNZ
1376 02474   0 01 02502       JMP   *+6
1377 02475   000201           IAB
1378 02476   140407           TCA
1379 02477   0 04 04042       STA   N254
1380 02500   0 04 02565       STA   CHAN
1381 02501   0 01 02504       JMP   *+3
1382 02502   0 02 00000       LDA   0
1383 02503   0 04 02565       STA   CHAN
1384 02504   0 12 02566       IRS   PAS1
1385 02505   0 01 02635       JMP   CB1
1386                    *
1387                    * M6800 scans over or any other known
1388                    * error condition in M6800 Executive ...
1389                    * MFIN sets all relevant flags for
1390                    * subsequent M6800 communication ...
1391                    *
1392 02506   0 000000   MFIN  DAC   **
1393 02507   001001           INH
1394 02510   0 02 01622       LDA   MC1
1395 02511   100040           SZE
1396 02512   14 0570          OCP   SET1
1397 02513   0 02 01626       LDA   MC2
1398 02514   100040           SZE
1399 02515   14 0770          OCP   SET2
1400 02516   0 02 01632       LDA   MC3
1401 02517   100040           SZE
1402 02520   14 0370          OCP   SET3
1403 02521   140040           CRA
1404 02522   0 04 01622       STA   MC1
```

```
                                *
1405 02523    0 04 01626            STA    MC2
1406 02524    0 04 01632            STA    MC3
1407 02525    0 04 04043            STA    F254
1408 02526    0 04 04221            STA    PAS
1409 02527    0 04 04222            STA    UMNO
1410 02530    0 04 03171            STA    XXX2
1411 02531    0 04 04217            STA    MIC
1412 02532    0 04 02151            STA    CALM
1413 02533    0 04 02366            STA    CFLG
1414 02534    0 04 00001            STA    M680
1415 02535    0 04 02147            STA    ERRM
1416 02536    000401                ENB
1417 02537   -0 01 02506            JMP*   MFIN
1418                               *
1419 02540    0 02 02567 PROC       LDA    PAS2
1420 02541    100040                SZE
1421 02542    0 01 02547            JMP    LSBY
1422 02543    000201     MSBY       IAB
1423 02544    0 04 02556            STA    UNEW
1424 02545    0 12 02567            IRS    PAS2
1425 02546    0 01 02635            JMP    CB1
1426 02547    000201     LSBY       IAB
1427 02550    141340                ICA
1428 02551    000201                IAB
1429 02552    0 02 02556            LDA    UNEW
1430 02553    0410 70               LLL    8
1431 02554    0 04 02556            STA    UNEW
1432 02555    0 10 02575            JST    UOUT
1433 02556    000000     UNEW       BSZ    1
1434 02557    0 02 04043            LDA    F254
1435 02560    101040                SNZ
1436 02561    0 04 02566            STA    PAS1
1437 02562    140040                CRA
1438 02563    0 04 02567            STA    PAS2
1439 02564    0 01 02635            JMP    CB1
1440 02565    000000     CHAN       BSZ    1
1441 02566    000000     PAS1       BSZ    1
1442 02567    000000     PAS2       BSZ    1
1443 02570    000000     PAS3       BSZ    1
1444 02571    000000     PAS4       BSZ    1
1445 02572    000000     ABF0       BSZ    1
1446 02573    000000     ABF3       BSZ    1
1447 02574    000000     CHXM       BSZ    1
1448                               *
1449 02575    0 000000   UOUT       DAC    **
1450 02576    0 02 04043            LDA    F254
1451 02577    100040                SZE
1452 02600    0 01 02615            JMP    UDAT
1453 02601   -0 02 02575            LDA*   UOUT
1454 02602    0415 77               ALS    1
1455 02603    000201                IAB
1456 02604    0 02 02565            LDA    CHAN
1457 02605    001001                INH
1458 02606    0400 74               LRL    4
1459 02607    000201                IAB
1460 02610    74 1370               OTA    DGOA
1461 02611    0 01 02610            JMP    *-1
```

```
                                *                                PAGE      27
1462 02612   000401                  ENB
1463 02613   0 12 02575              IRS    UOUT
1464 02614  -0 01 02575              JMP*   UOUT
1465                               *
1466                               * If M6800 data transfer,come here.
1467                               *
1468 02615  -0 02 02575 UDAT       LDA*   UOUT
1469 02616  -0 04 04044            STA*   BUFA
1470 02617   0 12 04044            IRS    BUFA
1471 02620   0 12 02575            IRS    UOUT
1472 02621   0 12 02565            IRS    CHAN
1473 02622  -0 01 02575            JMP*   UOUT
1474 02623   140040                CRA
1475 02624   0 04 04043            STA    F254
1476 02625   0 04 02566            STA    PAS1
1477 02626   0 04 02567            STA    PAS2
1478 02627   0 02 02634            LDA    VEC1
1479 02630   0 04 00306            STA    GOAD
1480 02631   0 02 04400            LDA    MDAC
1481 02632   0 04 04044            STA    BUFA
1482 02633   0 01 02635            JMP    CB1
1483 02634   0 004045     VEC1     DAC    B254
1484                               *
1485                               * Output CB1 low ... the CB2 acknowledge
1486                               *
1487 02635   140040       CB1      CRA
1488 02636   140500                SSM
1489 02637   74 1270               OTA    DGOB
1490 02640   0 01 02637            JMP    *-1
1491 02641   140100                SSP
1492 02642   74 1270               OTA    DGOB
1493 02643   0 01 02642            JMP    *-1
1494 02644   0 10 00160            JST    LNRE
1495 02645   0 01 01637            JMP    CB2
1496                               *
1497                               * Send DATA bytes to M6800 now
1498                               *
1499 02646   0 02 02364 CA2R       LDA    ABIM
1500 02647   101040                SNZ
1501 02650   0 01 03126            JMP    CA20
1502 02651   0 11 02706            CAS    =-2
1503 02652   0 01 03050            JMP    CA23
1504 02653   0 01 03031            JMP    CA22
1505 02654   0 02 02570 CA21       LDA    PAS3
1506 02655   100040                SZE
1507 02656   0 01 02664            JMP    *+6
1508 02657   0 02 03526            LDA    MIN3
1509 02660   0 04 03524            STA    WAS+1
1510 02661   0 12 02570            IRS    PAS3
1511 02662   0 02 03527            LDA    C1
1512 02663   0 04 02573            STA    ABF3
1513 02664   0 10 03460            JST    CDAT
1514 02665   0 02 01626            LDA    MC2
1515 02666   101040                SNZ
1516 02667   0 01 02672            JMP    *+3
1517 02670   0 12 02364            IRS    ABIM
1518 02671   0 01 03532            JMP    CA1
```

```
                                *
1519 02672   0 02 01632          LDA   MC3
1520 02673   101040               SNZ
1521 02674   0 01 02700          JMP   *+4
1522 02675   0 12 02364          IRS   ABIM
1523 02676   0 12 02364          IRS   ABIM
1524 02677   0 01 03532          JMP   CA1
1525 02700   0 01 03000          JMP   YYY
1526 02701   000037             FIN
     02702   177720
     02703   000000
     02704   177777
     02705   000010
     02706   177776
     02707   000004
     02710   177775
     02711   000002
     02712   000001

1527         002713       BAS2  EQU   *
1528                              ORG   '3000
1529                              SETB  BAS3
1530                         *
1531 03000   0 02 02366    YYY   LDA   CFLG
1532 03001   101040               SNZ
1533 03002   0 01 03026          JMP   VIAX
1534 03003   0 02 01622          LDA   MC1
1535 03004   100040               SZE
1536 03005   0 01 03015          JMP   MCP1
1537 03006   0 02 01626          LDA   MC2
1538 03007   100040               SZE
1539 03010   0 01 03020          JMP   MCP2
1540 03011   0 02 01632          LDA   MC3
1541 03012   100040               SZE
1542 03013   0 01 03023          JMP   MCP3
1543 03014   000000       HALT  HLT
1544 03015   0 02 03667  MCP1   LDA   =-3
1545 03016   0 04 03170          STA   XCAT
1546 03017   0 10 04235          JST   XXX1
1547 03020   0 02 03666  MCP2   LDA   =-2
1548 03021   0 04 03170          STA   XCAT
1549 03022   0 10 04235          JST   XXX1
1550 03023   0 02 03665  MCP3   LDA   =-1
1551 03024   0 04 03170          STA   XCAT
1552 03025   0 10 04235          JST   XXX1
1553 03026   140040      VIAX   CRA
1554 03027   0 04 03170          STA   XCAT
1555 03030   0 10 04235          JST   XXX1
1556                         *
1557 03031   0 02 02570  CA22   LDA   PAS3
1558 03032   100040               SZE
1559 03033   0 01 03041          JMP   *+6
1560 03034   0 02 03526          LDA   MIN3
1561 03035   0 04 03524          STA   WAS+1
1562 03036   0 12 02570          IRS   PAS3
1563 03037   0 02 03530          LDA   C2
1564 03040   0 04 02573          STA   ABF3
1565 03041   0 10 03460          JST   CDAT
```

```
                                *
1566 03042    0 02 01632        LDA   MC3
1567 03043    101040            SNZ
1568 03044    0 01 03047        JMP   *+3
1569 03045    0 12 02364        IRS   ABIM
1570 03046    0 01 03532        JMP   CA1
1571 03047    0 01 03000        JMP   YYY
1572                            *
1573 03050    0 02 02570  CA23  LDA   PAS3
1574 03051    100040            SZE
1575 03052    0 01 03060        JMP   *+6
1576 03053    0 02 03526        LDA   MIN3
1577 03054    0 04 03524        STA   WAS+1
1578 03055    0 12 02570        IRS   PAS3
1579 03056    0 02 03531        LDA   C3
1580 03057    0 04 02573        STA   ABF3
1581 03060    0 10 03460        JST   CDAT
1582 03061    0 01 03000        JMP   YYY
1583                            *
1584                            * MERR routine checks for error condition
1585                            * If found,modify current BYTE and next
1586                            * for M6800 decoding .. 3 errors detectable
1587                            * FORTRAN library,counter mispecificatinn
1588                            * and DATA byte > 1023 for analogue i/p
1589                            *
1590 03062    0 000000    MERR  DAC   **
1591 03063    0 02 03171        LDA   XXX2
1592 03064    100040            SZE
1593 03065    0 01 03074        JMP   *+7
1594 03066    0 02 02147        LDA   ERRM
1595 03067    101040            SNZ
1596 03070    0 01 03074        JMP   *+4
1597 03071   -0 35 03062        LDX*  MERR
1598 03072    1 02 00000        LDA   0,1
1599 03073    0 04 03521        STA   BYTE
1600 03074    0 12 03062        IRS   MERR
1601 03075   -0 01 03062        JMP*  MERR
1602                            *
1603                            * Activate 'M6800 Graphics' if required ..
1604                            * ie. when A(12)=1.
1605                            *
1606 03076    0 000000    GRAF  DAC   **
1607 03077    0 02 04340        LDA   CTRS+11
1608 03100    101040            SNZ
1609 03101   -0 01 03076        JMP*  GRAF
1610 03102    0 12 00633        IRS   ARR0
1611 03103   -0 01 03076        JMP*  GRAF
1612 03104    0 02 00265        LDA   IFLG
1613 03105    101040            SNZ
1614 03106    0 01 03112        JMP   PLOT
1615 03107    0 02 03122        LDA   VEC4
1616 03110    0 04 00311        STA   GOAD+3
1617 03111   -0 01 03076        JMP*  GRAF
1618 03112    0 02 03121  PLOT  LDA   VEC3
1619 03113    0 04 00310        STA   GOAD+2
1620 03114    0 12 00265        IRS   IFLG
1621 03115    0 02 04325        LDA   CTRS
1622 03116    140407            TCA
```

```
                                *
1623 03117    0 04 00633            STA    ARR0
1624 03120   -0 01 03076            JMP*   GRAF
1625 03121    0 003123      VEC3    DAC    GRF1
1626 03122    0 003124      VEC4    DAC    GRF2
1627                            *
1628 03123    0 01 01553    GRF1    JMP    A7
1629                            *
1630 03124    0 10 03603    GRF2    JST    ERCL
1631 03125    143705                BCI    1,GE
1632                            *
1633                            * Following routine used in M6800 scan and
1634                            * in H316 to M6800 data transfer .. PAS=1
1635                            * or 2 in M6800 off-line mode !
1636                            *
1637 03126    0 02 03525    CA20    LDA    TEST
1638 03127    100040                SZE
1639 03130    0 01 03141            JMP    LSBE
1640 03131   -0 02 02454            LDA*   ABF2
1641 03132    141140                ICL
1642 03133    0 04 03521            STA    BYTE
1643 03134    0 10 03062            JST    MERR
1644 03135    0 003236              DAC    C200
1645 03136    140040                CRA
1646 03137    0 12 03525            IRS    TEST
1647 03140    0 01 03532            JMP    CA1
1648 03141   -0 02 02454    LSBE    LDA*   ABF2
1649 03142    141050                CAL
1650 03143    0 04 03521            STA    BYTE
1651 03144    0 10 03062            JST    MERR
1652 03145    0 002150              DAC    MCOD
1653 03146    0 02 02147            LDA    ERRM
1654 03147    100040                SZE
1655 03150    0 10 02506            JST    MFIN
1656 03151    140040                CRA
1657 03152    0 04 03525            STA    TEST
1658 03153    0 12 02454            IRS    ABF2
1659 03154    0 12 02452            IRS    CHN0
1660 03155    0 01 03532            JMP    CA1
1661 03156    0 02 04221            LDA    PAS
1662 03157    101040                SNZ
1663 03160    0 01 03172            JMP    PNIM
1664 03161    0 12 04221            IRS    PAS
1665 03162    0 01 03532            JMP    CA1
1666 03163    140040        PSRM    CRA
1667 03164    0 04 04217            STA    MIC
1668 03165    0 04 04221            STA    PAS
1669 03166    0 04 04220            STA    MCA
1670 03167   -0 01 01572            JMP*   CL06
1671 03170    000000        XCAT    BSZ    1
1672 03171    000000        XXX2    BSZ    1
1673                            *
1674                            * Return after CALL(4,I,M,D(0))
1675                            *
1676 03172    0 02 03171    PNIM    LDA    XXX2          Send Ensemble
1677 03173    100040                SZE                  number to
1678 03174    0 10 04235            JST    XXX1          micro.
1679 03175    0 02 02571            LDA    PAS4
```

```
                                *
1680 03176   100040             SZE
1681 03177   0 01 03206          JMP   *+7
1682 03200   0 02 03665          LDA   =-1
1683 03201   0 04 02452          STA   CHN0
1684 03202   0 12 02571          IRS   PAS4
1685 03203   0 02 02373          LDA   REAL+3
1686 03204   0 04 02454          STA   ABF2
1687 03205   0 01 03532          JMP   CA1
1688 03206   140040             CRA
1689 03207   0 04 02571          STA   PAS4
1690                            *
1691 03210   0 02 04312          LDA   MCTR
1692 03211   0 03 03664          ANA   =2
1693 03212   101040             SNZ
1694 03213   0 01 03217          JMP   *+4
1695 03214   0 02 03667          LDA   =-3
1696 03215   0 04 02364          STA   ABIM
1697 03216   0 01 03532          JMP   CA1
1698                            *
1699 03217   0 02 04312          LDA   MCTR
1700 03220   0 03 03663          ANA   =4
1701 03221   101040             SNZ
1702 03222   0 01 03226          JMP   *+4
1703 03223   0 02 03666          LDA   =-2
1704 03224   0 04 02364          STA   ABIM
1705 03225   0 01 03532          JMP   CA1
1706                            *
1707 03226   0 02 04312          LDA   MCTR
1708 03227   0 03 03662          ANA   =8
1709 03230   101040             SNZ
1710 03231   0 01 03235          JMP   ZZZ
1711 03232   0 02 03665          LDA   =-1
1712 03233   0 04 02364          STA   ABIM
1713 03234   0 01 03532          JMP   CA1
1714 03235   0 10 04235 ZZZ      JST   XXX1
1715 03236   000200     C200     OCT   200
1716 03237   0 003240   CADR     DAC   C123
1717 03240   000000     C123     BSZ   6
1718                            *
1719                            * M6800 Counter(s) routines .. values
1720                            * also stored in BASIC B-array if
1721                            * A(12)=1.
1722                            *
1723 03246   0 000000   A15      DAC   **
1724 03247   0 02 02146          LDA   EFLG
1725 03250   100040             SZE
1726 03251   -0 01 03246        JMP*  A15
1727 03252   0 02 01622          LDA   MC1
1728 03253   101040             SNZ
1729 03254   0 01 03326          JMP   A16
1730 03255   001001             INH
1731 03256   140040             CRA
1732 03257   54 0470            INA   CTR1
1733 03260   0 01 03257          JMP   *-1
1734 03261   0 04 01624          STA   MC1+2
1735 03262   000201             IAB
1736 03263   0 02 04317          LDA   MCTR+5
```

```
                          *
1737 03264   74 0570            OTA   SET1
1738 03265   0 01 03264         JMP   *-1
1739 03266   000401             ENB
1740 03267   0 02 04340         LDA   CTRS+11
1741 03270   101040             SNZ
1742 03271   0 01 03275         JMP   *+4
1743 03272   0 02 00344         LDA   PARS+1
1744 03273   0 06 03661         ADD   =96
1745 03274   0 01 03276         JMP   *+2
1746 03275   0 02 03237         LDA   CADR
1747 03276   0 04 03316         STA   MB48
1748 03277   140442             OCT   140442
1749 03300   0 04 03306         STA   MB49
1750 03301   140442             OCT   140442
1751 03302   0 04 03323         STA   MB50
1752 03303   000201             IAB
1753 03304  -0 10 04013         JST*  MC12
1754 03305  -0 10 04016         JST*  MH22
1755 03306   0 000000    MB49   DAC   **
1756 03307   0 02 01027         LDA   ITM1
1757 03310   0 10 03513         JST   CZER
1758 03311   0 04 01623         STA   MC1+1
1759 03312  -0 10 04013         JST*  MC12
1760 03313  -0 10 04017         JST*  MD22
1761 03314   0 000646           DAC   F50
1762 03315  -0 10 04016         JST*  MH22
1763 03316   0 000000    MB48   DAC   **
1764 03317   0 02 01030         LDA   FLG1
1765 03320   0 04 01625         STA   MC1+3
1766 03321  -0 10 04013         JST*  MC12
1767 03322  -0 10 04016         JST*  MH22
1768 03323   0 000000    MB50   DAC   **
1769 03324   140040             CRA
1770 03325   0 04 01030         STA   FLG1
1771                     *
1772 03326   0 02 01626  A16    LDA   MC2
1773 03327   101040             SNZ
1774 03330   0 01 03402         JMP   A17
1775 03331   001001             INH
1776 03332   140040             CRA
1777 03333   54 0670            INA   CTR2
1778 03334   0 01 03333         JMP   *-1
1779 03335   0 04 01630         STA   MC2+2
1780 03336   000201             IAB
1781 03337   0 02 04321         LDA   MCTR+7
1782 03340   74 0770            OTA   SET2
1783 03341   0 01 03340         JMP   *-1
1784 03342   000401             ENB
1785 03343   0 02 04340         LDA   CTRS+11
1786 03344   101040             SNZ
1787 03345   0 01 03351         JMP   *+4
1788 03346   0 02 00344         LDA   PARS+1
1789 03347   0 06 03660         ADD   =102
1790 03350   0 01 03352         JMP   *+2
1791 03351   0 02 03237         LDA   CADR
1792 03352   0 04 03372         STA   MB51
1793 03353   140442             OCT   140442
```

```
                                *
1794 03354    0 04 03362        STA   MB52
1795 03355    140442            OCT   140442
1796 03356    0 04 03377        STA   MB53
1797 03357    000201            IAB
1798 03360   -0 10 04013        JST*  MC12
1799 03361   -0 10 04016        JST*  MH22
1800 03362    0 000000    MB52  DAC   **
1801 03363    0 02 01060        LDA   ITM2
1802 03364    0 10 03513        JST   CZER
1803 03365    0 04 01627        STA   MC2+1
1804 03366   -0 10 04013        JST*  MC12
1805 03367   -0 10 04017        JST*  MD22
1806 03370    0 000646          DAC   F50
1807 03371   -0 10 04016        JST*  MH22
1808 03372    0 000000    MB51  DAC   **
1809 03373    0 02 01061        LDA   FLG2
1810 03374    0 04 01631        STA   MC2+3
1811 03375   -0 10 04013        JST*  MC12
1812 03376   -0 10 04016        JST*  MH22
1813 03377    0 000000    MB53  DAC   **
1814 03400    140040            CRA
1815 03401    0 04 01061        STA   FLG2
1816                            *
1817 03402    000401      A17   ENB
1818 03403    0 02 01632        LDA   MC3
1819 03404    101040            SNZ
1820 03405   -0 01 03246        JMP*  A15
1821 03406    001001            INH
1822 03407    140040            CRA
1823 03410    54 0270           INA   CTR3
1824 03411    0 01 03410        JMP   *-1
1825 03412    0 04 01634        STA   MC3+2
1826 03413    000201            IAB
1827 03414    0 02 04323        LDA   MCTR+9
1828 03415    74 0370           OTA   SET3
1829 03416    0 01 03415        JMP   *-1
1830 03417    000401            ENB
1831 03420    0 02 04340        LDA   CTRS+11
1832 03421    101040            SNZ
1833 03422    0 01 03426        JMP   *+4
1834 03423    0 02 00344        LDA   PARS+1
1835 03424    0 06 03657        ADD   =108
1836 03425    0 01 03427        JMP   *+2
1837 03426    0 02 03237        LDA   CADR
1838 03427    0 04 03447        STA   MB54
1839 03430    140442            OCT   140442
1840 03431    0 04 03437        STA   MB55
1841 03432    140442            OCT   140442
1842 03433    0 04 03454        STA   MB56
1843 03434    000201            IAB
1844 03435   -0 10 04013        JST*  MC12
1845 03436   -0 10 04016        JST*  MH22
1846 03437    0 000000    MB55  DAC   **
1847 03440    0 02 01111        LDA   ITM3
1848 03441    0 10 03513        JST   CZER
1849 03442    0 04 01633        STA   MC3+1
1850 03443   -0 10 04013        JST*  MC12
```

```
                                *
1851 03444  -0 10 04017         JST*  MD22
1852 03445   0 000646           DAC   F50
1853 03446  -0 10 04016         JST*  MH22
1854 03447   0 000000    MB54   DAC   **
1855 03450   0 02 01112         LDA   FLG3
1856 03451   0 04 01635         STA   MC3+3
1857 03452  -0 10 04013         JST*  MC12
1858 03453  -0 10 04016         JST*  MH22
1859 03454   0 000000    MB56   DAC   **
1860 03455   140040             CRA
1861 03456   0 04 01112         STA   FLG3
1862 03457  -0 01 03246         JMP*  A15
1863                            *
1864 03460   0 000000    CDAT   DAC   **
1865 03461   0 02 03523         LDA   WAS
1866 03462   100040             SZE
1867 03463   0 01 03473         JMP   LSB
1868 03464  -0 02 02573         LDA*  ABF3
1869 03465   141140             ICL
1870 03466   0 04 03521         STA   BYTE
1871 03467   0 10 03062         JST   MERR
1872 03470   0 003236           DAC   C200
1873 03471   0 12 03523         IRS   WAS
1874 03472   0 01 03532         JMP   CA1
1875 03473  -0 02 02573 LSB    LDA*  ABF3
1876 03474   141050             CAL
1877 03475   0 04 03521         STA   BYTE
1878 03476   0 10 03062         JST   MERR
1879 03477   0 002150           DAC   MCOD
1880 03500   0 02 02147         LDA   ERRM
1881 03501   100040             SZE
1882 03502   0 10 02506         JST   MFIN
1883 03503   140040             CRA
1884 03504   0 04 03523         STA   WAS
1885 03505   0 12 02573         IRS   ABF3
1886 03506   0 12 03524         IRS   WAS+1
1887 03507   0 01 03532         JMP   CA1
1888 03510   140040             CRA
1889 03511   0 04 02570         STA   PAS3
1890 03512  -0 01 03460         JMP*  CDAT
1891                            *
1892 03513   0 000000    CZER   DAC   **
1893 03514   0 04 00000         STA   0
1894 03515   0 02 01636         LDA   TZER
1895 03516   100040             SZE
1896 03517   0 02 00000         LDA   0
1897 03520  -0 01 03513         JMP*  CZER
1898                            *
1899 03521   000000     BYTE   BSZ   2
1900 03523   000000     WAS    BSZ   2
1901 03525   000000     TEST   BSZ   1
1902 03526   177775     MIN3   DEC   -3
1903 03527   0 001623   C1     DAC   MC1+1
1904 03530   0 001627   C2     DAC   MC2+1
1905 03531   0 001633   C3     DAC   MC3+1
1906                            *
1907                            * Output CA1 low --- the CA2 acknowledge
```

```
1908                        *
1909                        * If M6800 off-line and no longer in
1910                        * communication .. go back to BASIC !
1911 03532   0 02 04312 CA1  LDA   MCTR
1912 03533   0 03 03670      ANA   =16
1913 03534   100040         SZE
1914 03535   0 10 03571      JST   XCA1
1915 03536   140040         CRA
1916 03537   140500         SSM
1917 03540   0405 77        ARS   1
1918 03541   140100         SSP
1919 03542   0 05 03521      ERA   BYTE
1920 03543   74 1270         OTA   DGOB
1921 03544   0 01 03543      JMP   *-1
1922 03545   141050         CAL
1923 03546   74 1270         OTA   DGOB
1924 03547   0 01 03546      JMP   *-1
1925 03550   0 02 04221      LDA   PAS
1926 03551   101040         SNZ
1927 03552   0 01 03567      JMP   IRC
1928 03553   0 02 04222      LDA   UMNO
1929 03554   100040         SZE
1930 03555   0 01 03553      JMP   *-2
1931 03556   0 02 04221      LDA   PAS
1932 03557   0 11 03656      CAS   =1
1933 03560   0 01 03163      JMP   PSRM
1934 03561   101000         NOP
1935 03562   0 12 04222      IRS   UMNO
1936 03563   0 02 03525      LDA   TEST
1937 03564   101040         SNZ
1938 03565   0 01 03126      JMP   CA20
1939 03566   0 01 03141      JMP   LSBE
1940 03567   0 10 00173 IRC  JST   LNRF
1941 03570   0 01 02160      JMP   CA2
1942                        *
1943 03571   0 000000   XCA1 DAC   **
1944 03572   0 02 02151      LDA   CALM
1945 03573   100040         SZE
1946 03574  -0 01 03571      JMP*  XCA1
1947 03575  -0 01 01572      JMP*  CL06
1948                        *
1949 03576   140040     CA2C CRA
1950 03577   0 04 04222      STA   UMNO
1951 03600   0 01 03567      JMP   IRC
1952                        *
1953 03601   0 10 03603 BRKC JST   ERCL
1954 03602   141313         BCI   1,BK
1955                        *
1956                        * HADIOS Executive ERROR handling routine
1957                        * Adjusts location '61 where requires ..
1958                        *
1959 03603   0 000000   ERCL DAC   **
1960 03604   140040         CRA
1961 03605   0 04 04340      STA   CTRS+11
1962 03606  -0 02 03603      LDA*  ERCL
1963 03607   0 04 04011      STA   EDAC
1964 03610   0 02 04341      LDA   CTRS+12
```

```
                                *
1965 03611   100040             SZE
1966 03612   0 01 04006          JMP   MODE
1967 03613   0 02 00632          LDA   CAL0
1968 03614   100040             SZE
1969 03615   0 01 03617          JMP   *+2
1970 03616   0 01 04006          JMP   MODE
1971 03617   001001             INH
1972 03620   0 02 04325 HFIN     LDA   CTRS
1973 03621   0 03 03664          ANA   =2
1974 03622   100040             SZE
1975 03623   14 0570            OCP   SET1
1976 03624   0 02 04325          LDA   CTRS
1977 03625   0 03 03663          ANA   =4
1978 03626   100040             SZE
1979 03627   14 0770            OCP   SET2
1980 03630   0 02 04325          LDA   CTRS
1981 03631   0 03 03662          ANA   =8
1982 03632   100040             SZE
1983 03633   14 0370            OCP   SET3
1984 03634   140040             CRA
1985 03635   0 04 00632          STA   CAL0
1986 03636   0 02 00061          LDA   '61
1987 03637   0 04 00630          STA   L61
1988 03640   0 10 00706          JST   MTES
1989 03641   14 0220            OCP   SCLK
1990 03642   0 02 00102          LDA   LNRA-2
1991 03643   140401             CMA
1992 03644   0 03 00264          ANA   MASK
1993 03645   0 04 00264          STA   MASK
1994 03646   74 0020            SMK   '20
1995 03647   0 02 00635          LDA   NMAX
1996 03650   0 04 00061          STA   '61
1997 03651   14 0020            OCP   CLK
1998 03652   000401             ENB
1999 03653   0 02 00631          LDA   HEND
2000 03654   100040             SZE
2001 03655   0 01 04006          JMP   MODE
2002                            *
2003 03656   000001             FIN
     03657   000154
     03660   000146
     03661   000140
     03662   000010
     03663   000004
     03664   000002
     03665   177777
     03666   177776
     03667   177775
     03670   000020

2004         003671        BAS3 EQU   *
2005                             ORG   '4000
2006                             SETB  BAS4
2007                            *
2008 04000   000401        GFIN ENB
2009 04001   0 02 00643          LDA   SIP2
2010 04002   0 04 00034          STA   SIP
```

```
                                *
2011 04003    0 02 00644        LDA    SBP2
2012 04004    0 04 00037        STA    SBP
2013 04005   -0 01 01572        JMP*   CL06
2014                            *
2015                            * Return to BASIC command mode via '5243
2016                            *
2017 04006    140040      MODE  CRA
2018 04007    0 04 00265        STA    IFLG
2019 04010   -0 10 04012        JST*   ERR
2020 04011    0 000000    EDAC  DAC    **
2021 04012    005243      ERR   OCT    5243
2022                            *
2023                            * M6800 FORTRAN library pointers here
2024                            *
2025 04013    0 000000    MC12  XAC    C#12
2026 04014    0 000000    MC21  XAC    C#21
2027 04015    0 000000    ML22  XAC    L#22
2028 04016    0 000000    MH22  XAC    H#22
2029 04017    0 000000    MD22  XAC    D#22
2030 04020    0 000000    MM22  XAC    M#22
2031                            *
2032                            * All M6800 CB2 bytes checked for error
2033                            * codes here .. 255 = Error or scans over
2034                            *                254 = M6800 data transfer
2035                            *
2036 04021    0 000000    NCEK  DAC    **
2037 04022    0 02 02367        LDA    CFLG+1
2038 04023    0 11 04406        CAS    =254
2039 04024    0 01 04040        JMP    M255
2040 04025    0 01 04027        JMP    M254
2041 04026   -0 01 04021        JMP*   NCEK
2042 04027    0 12 04043  M254  IRS    F254
2043 04030    0 02 04400        LDA    MDAC
2044 04031    0 04 04044        STA    BUFA
2045 04032    140040            CRA
2046 04033    0 35 04405        LDX    =-30
2047 04034   -0 04 04401        STA*   MDAC+1
2048 04035    0 12 00000        IRS    0
2049 04036    0 01 04034        JMP    *-2
2050 04037    0 01 02635        JMP    CB1
2051 04040    0 10 02506  M255  JST    MFIN
2052 04041    0 01 02635        JMP    CB1
2053 04042    000000      N254  BSZ    1
2054 04043    000000      F254  BSZ    1
2055 04044    000000      BUFA  BSZ    1
2056                            *
2057 04045    001001      B254  INH
2058 04046    0 02 00344        LDA    PARS+1
2059 04047    0 06 04404        ADD    =114
2060 04050    0 04 04054        STA    BDAC
2061 04051   -0 02 04044  BREP  LDA*   BUFA
2062 04052   -0 10 00675        JST*   C12
2063 04053   -0 10 00654        JST*   H22
2064 04054    0 000000    BDAC  DAC    **
2065 04055    0 12 04054        IRS    BDAC
2066 04056    0 12 04054        IRS    BDAC
2067 04057    0 12 04044        IRS    BUFA
```

```
                                    *
04060   0 12 04042         IRS   N254
04061   0 01 04051         JMP   BREP
04062   000401             ENB
04063   101000             NOP
04064   0 02 04341         LDA   CTRS+12
04065   100040             SZE
04066   0 01 01553         JMP   A7
04067   0 01 00266         JMP   DISP
                    *
                    * BASIC statement --- CALL(4,I,M,D(0))
                    * I=0 if M6800 on-line , I=1 if off-line
                    * M=nuber of H316 words to be transferred
                    * M = 1 to 30 only
                    *
04070   0 02 00636  SUB4   LDA   NEXT
04071   0 04 04073         STA   *+2
04072   0 01 04074         JMP   *+2
04073   0 000000           DAC   **
04074  -0 10 00640         JST*  FAT
04075   000003             DEC   3
04076   000000      I      BSZ   1
04077   000000      M      BSZ   1
04100   000000      D      BSZ   1
04101   0 02 02151         LDA   CALM
04102   101040             SNZ
04103  -0 01 01572         JMP*  CL06
04104   0 35 04403         LDX   =-4
04105   140040             CRA
04106   1 04 04223         STA   MIC+4,1
04107   0 12 00000         IRS   0
04110   0 01 04106         JMP   *-2
04111  -0 10 00653         JST*  L22
04112  -0 004076           DAC*  I
04113  -0 10 00676         JST*  C21
04114   0 01 00570         JMP   ERI
04115   100040             SZE
04116   0 12 04220         IRS   MCA
04117  -0 10 00653         JST*  L22
04120  -0 004077           DAC*  M
04121  -0 10 00676         JST*  C21
04122   0 01 00570         JMP   ERI
04123   141206             AOA
04124   0 04 04156         STA   BUF1
04125   140407             TCA
04126   0 04 04155         STA   CHN4
04127   0 04 00000         STA   0
04130   0 12 00000         IRS   0
04131   0 02 04215         LDA   BUF2
04132   0 04 04216         STA   BUF3
04133   0 12 04216         IRS   BUF3
04134  -0 10 00653  CA2X   JST*  L22
04135  -0 004100           DAC*  D
04136  -0 10 00676         JST*  C21
04137   0 01 00570         JMP   ERI
04140  -0 04 04216         STA*  BUF3
04141   0 12 04100         IRS   D
04142   0 12 04100         IRS   D
```

```
                                *
2125 04143    0 12 04216          IRS    BUF3
2126 04144    0 12 00000          IRS    0
2127 04145    0 01 04134          JMP    CA2X
2128 04146    0 02 04215          LDA    BUF2
2129 04147    0 04 04216          STA    BUF3
2130 04150    0 12 04217          IRS    MIC
2131 04151    0 02 04220          LDA    MCA
2132 04152    100040              SZE
2133 04153    0 01 04223          JMP    DAP
2134 04154   -0 01 01572          JMP*   CL06
2135 04155    000000       CHN4   BSZ    1
2136 04156    000000       BUF1   BSZ    1
2137 04157    000000              BSZ    30
2138 04215    0 004156     BUF2   DAC    BUF1
2139 04216    000000       BUF3   BSZ    1
2140 04217    000000       MIC    BSZ    1
2141 04220    000000       MCA    BSZ    1
2142 04221    000000       PAS    BSZ    1
2143 04222    000000       UMNO   BSZ    1
2144                        *
2145 04223   -0 02 04215   DAP    LDA*   BUF2
2146 04224    0 06 04407          ADD    =256
2147 04225   -0 04 04215          STA*   BUF2
2148 04226    0 02 04215          LDA    BUF2
2149 04227    0 04 02454          STA    ABF2
2150 04230    0 12 04221          IRS    PAS
2151 04231    0 12 04222          IRS    UMNO
2152 04232    0 02 04155          LDA    CHN4
2153 04233    0 04 02452          STA    CHNO
2154 04234    0 01 03126          JMP    CA20
2155                        *
2156 04235    0 000000     XXX1   DAC    **
2157 04236    0 02 03171          LDA    XXX2
2158 04237    100040              SZE
2159 04240    0 01 04257          JMP    LCA1
2160 04241    0 02 04215          LDA    BUF2
2161 04242    0 04 02454          STA    ABF2
2162 04243    0 02 04217          LDA    MIC
2163 04244    101040              SNZ
2164 04245    0 01 04250          JMP    *+3
2165 04246    0 02 04155          LDA    CHN4
2166 04247    0 01 04252          JMP    *+3
2167 04250    0 04 04156          STA    BUF1
2168 04251    0 02 04402          LDA    =-1
2169 04252    0 04 02452          STA    CHNO
2170 04253    140040              CRA
2171 04254    0 04 02364          STA    ABIM
2172 04255    0 12 03171          IRS    XXX2
2173 04256    0 01 03532          JMP    CA1
2174 04257    140040       LCA1   CRA
2175 04260    0 04 03171          STA    XXX2
2176 04261    0 04 04217          STA    MIC
2177 04262    0 02 03170          LDA    XCAT
2178 04263    0 04 02364          STA    ABIM
2179 04264    0 12 02365          IRS    XXX
2180 04265    0 01 03532          JMP    CA1
2181                        *
```

```
                                    *                                PAGE
2182                                * Time adjustments for counter code ..
2183                                *
2184 04266    0 000000    TIME DAC   **
2185 04267    0 02 00630         LDA   L61
2186 04270   -1 07 04307         SUB*  TIM+3,1
2187 04271   -1 06 04312         ADD*  ADT+3,1
2188 04272   -1 04 04312         STA*  ADT+3,1
2189 04273    0 02 00632         LDA   CAL0
2190 04274    101040             SNZ
2191 04275    0 01 04301         JMP   **+4
2192 04276    0 02 00637         LDA   CINT
2193 04277   -1 04 04307         STA*  TIM+3,1
2194 04300   -0 01 04266         JMP*  TIME
2195 04301    0 02 00635         LDA   NMAX
2196 04302   -1 04 04307         STA*  TIM+3,1
2197 04303   -0 01 04266         JMP*  TIME
2198 04304    0 001026    TIM    DAC   TIM1
2199 04305    0 001057           DAC   TIM2
2200 04306    0 001110           DAC   TIM3
2201 04307    0 001025    ADT    DAC   ADT1
2202 04310    0 001056           DAC   ADT2
2203 04311    0 001107           DAC   ADT3
2204 04312    000000      MCTR   BSZ   10
2205 04324    0 004312    MACS   DAC   MCTR
2206 04325    000000      CTRS   BSZ   13
2207 04342    000000      BUFM   BSZ   30
2208 04400    0 004342    MDAC   DAC   BUFM
2209 04401    1 004400           DAC   BUFM+30,1
2210 04402    177777             FIN
     04403    177774
     04404    000162
     04405    177742
     04406    000376
     04407    000400

2211          004410      BAS4   EQU   *
2212                              *
2213                              * Patch in BASIC 'CALL' sequence
2214                              * '4012 originally contained JMP IBUF
2215                              * '551 originally contained '5243
2216                              * BASIC 'BREAK' instruction in '3267
2217                              *
2218                                    ABS
2219                                    ORG   '63
2220 00063    0 000021              DAC   SKST
2221                                    ORG   '551
2222 00551    0 003603              DAC   ERCL
2223                                    ORG   '716
2224 00716    0 000313              DAC   CALL
2225                                    ORG   '717
2226 00717    0 003601              DAC   BRKC
2227                                    ORG   '4012
2228 04012   -0 01 00716            JMP*  '716
2229                                    END

A1      000455    A15     003246    A16     003326    A17     003402
A2      000471    A20     001552    A3      001142    A4      001207
```

A5	001255	A5A	001516	A6	001323	A7	001553
A9	002374	AAA	002456	AB48	001431	ABF0	002572
ABF2	002454	ABF3	002573	ABF4	002455	ABIM	002364
ABM	002276	ABUF	001434	ACLK	001113	ADBF	001545
ADT	004307	ADT1	001025	ADT2	001056	ADT3	001107
AINT	000071	ALRI	001470A	ANAG	000170A	ARR0	000633
B254	004045	B48	001200	B49	001172	B50	001204
B51	001246	B52	001240	B53	001252	B54	001314
B55	001306	B56	001320	BAS0	000746	BAS1	001700
BAS2	002713	BAS3	003671	BAS4	004410	BBB	002457
BDAC	004054	BF0	001544	BFM	002453	BREP	004051
BRKC	003601	BSTP	001547	BUF1	004156	BUF2	004215
BUF3	004216	BUFA	004044	BUFM	004342	BYTE	003521
C1	003527	C12	000675A	C123	003240	C2	003530
C200	003236	C21	000676A	C3	003531	CA1	003532
CA2	002160	CA20	003126	CA21	002654	CA22	003031
CA23	003050	CA2C	003576	CA2R	002646	CA2S	002175
CA2X	004134	CADR	003237	CAL0	000632	CALL	000313
CALM	002151	CB1	002635	CB2	001637	CB22	002460
CDAT	003460	CERR	000572	CFLG	002366	CH3	000705
CHAN	002565	CHEC	002130	CHN0	002452	CHN4	004155
CHXM	002574	CHXR	001550	CIH	000176	CIHA	000651
CII	000247	CINT	000637	CIR	000222	CIRA	000245
CIRX	000246	CJST	000515A	CL06	001572	CLK	000020A
CONT	000030	CT1	000506	CT2	000526	CT3	000546
CTR1	000470A	CTR2	000670A	CTR3	000270A	CTRS	004325
CZER	003513	D	004100	D22	000670A	DAP	004223
DATA	000070A	DBF	001546	DGIA	001170A	DGIB	001070A
DGOA	001370A	DGOB	001270A	DISP	000266	DISQ	000301
DISS	000305	DLY	001421	DLYM	002266	EDAC	004011
EFLG	002146	ENS	001514	ENSM	002361	ERCL	003603
ERI	000570	ERR	004012	ERRM	002147	ESBL	001362
ESBM	002227	F254	004043	F50	000646	FAT	000640
FLG1	001030	FLG2	001061	FLG3	001112	FRST	000634
GFIN	004000	GOAD	000306	GPAR	000422	GRAF	003076
GRF1	003123	GRF2	003124	H22	000654A	HAD	001770A
HALT	003014	HCTS	001430	HEND	000631	HFIN	003620
HPC1	001350	HPC2	001402	HREG	001426	HTES	001576
I	004076	IBUF	000230A	ICT1	001000	ICT2	001031
ICT3	001062	IFLG	000265	INT	000566	IRC	003567
ITM1	001027	ITM2	001060	ITM3	001111	KT	000645
L22	000653A	L61	000630	LCA1	004257	LNKA	000074
LNKB	000107	LNKC	000122	LNKD	000135	LNKE	000150
LNKF	000163	LNRA	000104	LNRB	000117	LNRC	000132
LNRD	000145	LNRE	000160	LNRF	000173	LSB	003473
LSBE	003141	LSBY	002547	M	004077	M0	002017
M0X	002034	M1	002052	M2	002071	M22	000674A
M254	004027	M255	004040	M3	002110	M680	000001
MACS	004324	MASK	000264	MB48	003316	MB49	003306
MB50	003323	MB51	003372	MB52	003362	MB53	003377
MB54	003447	MB55	003437	MB56	003454	MBUF	002301
MC1	001622	MC12	004013	MC2	001626	MC21	004014
MC3	001632	MC68	000042	MCA	004220	MCKK	000650
MCOD	002150	MCP1	003015	MCP2	003020	MCP3	003023
MCTR	004312	MCTS	002275	MCX	000727	MD22	004017
MDAC	004400	MERR	003062	MFIN	002506	MH22	004016
MIC	004217	MIKR	000073	MIN3	003526	ML22	004015

```
                                *                                  PAGE      42
ML61    002152    MM22    004020    MODE    004006    MPC1    002215
MPC2    002247    MREG    002273    MSBY    002543    MTES    000706
MUX     001365    MUXM    002232    N       000661    N254    004042
NCEK    004021    NEXT    000636    NMAX    000635    OSAI    002433
PARS    000343    PAS     004221    PAS1    002566    PAS2    002567
PAS3    002570    PAS4    002571    PDCS    000000    PLOT    003112
PNIM    003172    PROC    002540    PSRM    003163    REAL    002370
REP     001530    REP1    002412    SAVK    000072    SBP     000037A
SBP1    000642    SBP2    000644    SCAN    001573    SCLK    000220A
SEC2    002000    SET1    000570A   SET2    000770A   SET3    000370A
SETW    002362    SIP     000034A   SIP1    000641    SIP2    000643
SKST    000021    SS      004064A   STOP    000277    STP     001574
STUP    001515    SUB1    000331    SUB2    000601    SUB3    000652
SUB4    004070    TABL    000317    TEST    003525    TIM     004304
TIM1    001026    TIM2    001057    TIM3    001110    TIME    004266
TZER    001636    U       000662    UDAT    002615    UMNO    004222
UNEW    002556    UOUT    002575    VEC1    002634    VEC3    003121
VEC4    003122    VIAX    003026    WAS     003523    X1      001551
XCA1    003571    XCAT    003170    XCTR    002363    XXX     002365
XXX1    004235    XXX2    003171    Y123    002446    YYY     003000
ZZZ     003235
0000 WARNING OR ERROR FLAGS
DAP-16 MOD 2     REV. C     01-26-71
AC
```

Table A4.2 Source listing of the M6800 Executive

```
*-------------------------------------------------*
*    M6800 EXECUTIVE --- VERSION DATE 26.01.1983  *
*    THE HADIOS EXECUTIVE IN THE H316 SHOULD  BE  *
*    INITIALISED BEFORE NORMAL EXECUTION OF THIS  *
*    PROGRAM IS POSSIBLE. THE NMI-SWITCH  SHOULD  *
*    BE USED TO RETURN TO MIKBUG MONITOR IN  ROM  *
*              PREPARED BY A.F.SHAFII             *
*-------------------------------------------------*
*
ORIGIN EQU £4000
*
* MIKBUG ROUTINES
*
INHEX  EQU £E089        ACCEPT 1 HEX DIGIT FROM VDU
INEEE  EQU £E1AC        PLACE CHARACTER IN VDU IN A-REG.
OUTCH  EQU £E075        PRINT CHAR. FROM ADDRESS £F018
PDATA1 EQU £E07E        PRINT BYTES FROM MEMORY
OUT4HS EQU £E0C8        PRINT 4 HEX DIGITS AND SPACE
CRLF   EQU £E0B1        RETURN AND LINE FEED
*
IORA0  EQU £F528        PIA A-SIDE I/O REGISTER
DDRA0  EQU IORA0        SAME FOR ITS DATA DIRECTION REG
CRA0   EQU £F529        A-SIDE CONTROL REGISTER
IORB0  EQU £F52A        PIA B-SIDE I/O REGISTER
DDRB0  EQU IORB0        SAME FOR ITS DATA DIRECTION REG
CRB0   EQU £F52B        B-SIDE CONTROL REGISTER
*
IRQVEC EQU £F000        IRQ INTERRUPT VECTOR
NMIVEC EQU £F006        NMI INTERRUPT VECTOR
SWIVEC EQU £F014        SWI INTERRUPT VECTOR
MONSWI EQU £E133        MONITOR SWI RESPONSE
MON    EQU £E0D0        MIKBUG MONITOR ENTRY POINT
*
* A AND B ADDRESS POINTERS
*
AVEC0  RMB 2            A(0)
AVEC2  RMB 2            A(2)
AVEC5  RMB 2            A(5)
BVEC0  RMB 2            B(0)
C1B48  RMB 2            B(48)
C2B51  RMB 2            B(51)
C3B54  RMB 2            B(54)
B57    RMB 2            B(57)
*
* £F000-£F072 IS SCRATCHPAD RAM FOR MONITOR
* £67F5-£6921 IS RUN-TIME PACKAGE STACK
*
*SD BASIC STATEMENT --- CALL SUB0
*
SUB0   SEI
       LDAA #£FF
```

```
        STAA CHAN7
        LDX #NMIRES     SET INTERRUPT VECTORS
        STX NMIVEC
        LDX #POLL
        STX IRQVEC
        LDX #MYSWI
        STX SWIVEC
        CLI
        LDX #VDU
        JSR PDATA1
        LDX #VDU5
        JSR PDATA1
        LDX #SUB0-1     HIGH SD BASIC ADDRESS
        STX A0
        LDX #A0
        JSR OUT4HS
        LDX #VDU0
        JSR PDATA1
        LDX #END        HIGH PROGRAM ADDRESS
        STX A0
        LDX #A0
        JSR OUT4HS
        JSR CRLF
TYPMOD  JSR CRLF
        LDX #VDU10
        JSR PDATA1
        JSR INEEE       INPUT 'Y' FOR ON-LINE : MODE=0
        CMPA #'Y              'N' FOR OFF-LINE: MODE=1
        BEQ GOYES
        CMPA #'N
        BEQ GONO
        BRA TYPMOD
GOYES   CLRB
        STAB MODE
        BRA TYPECR
GONO    LDAB #1
        STAB MODE
TYPECR  JSR INEEE
        CMPA #£0D
        BEQ GOCRLF
        BRA TYPECR
GOCRLF  JSR CRLF
        JSR CRLF
        SEI
        JSR PIA0AB      CONFIGURE PIA --- BASE ADDRESS £F528
        CLI             IF OFF-LINE THEN A-SIDE CA1
        JSR INIT2       INTERRUPTIBLE ELSE BOTH NORMAL
        RTS             HANDSHAKE MODE
*
*RETURN TO SD BASIC
```

```
MODE    RMB 1
*
*SD BASIC STATEMENT --- CALL SUB1(A(0),B(0))
*
SUB1    STS SPSAVE      SAVE STACK POINTER
        LDAA START
        BNE INIT1
        JMP DISP        JUMP TO DISPATCHER
START   RMB 1
INIT1   CLR START
*
        STX XSAVER      SAVE ADDRESSES A(0) AND B(0)
        LDX 4,X         THROUGH CALL STATEMENT
        DEX
        DEX
        STX TEMP
        JSR XPLUS6
        STX BVEC0       SAVE ADDRESS B(0)
        LDX XSAVER
        LDX 10,X
        JSR XPLUS4
        STX AVEC0       SAVE ADDRESS A(0)
*
        LDAA #3
        TAB
        DECB
        BSR BMULTX
        STX AVEC2       SAVE ADDRESS A(2)
*
        BSR UPDATE
        STX AVEC5       SAVE ADDRESS A(5)
*
        LDX BVEC0
        LDAB #48
        BSR BMULTX
        STX C1B48       SAVE ADDRESS B(48)
        BSR UPDATE
        STX C2B51       SAVE ADDRESS B(51)
        BSR UPDATE
        STX C3B54       SAVE ADDRESS B(54)
        BSR UPDATE
        STX B57         SAVE ADDRESS B(57)
        LDX AVEC2
        LDX 0,X
        STX AVEC2N      STORE VALUE OF A(2) IN AVEC2N
        JMP XMODE
AVEC2N  RMB 2
*
UPDATE  TAB             UPDATE ADDRESS IN X BY 6*(B) TIMES
        BSR BMULTX
```

```
        RTS
*
BMULTX  JSR XPLUS6      BUMP X SIX TIMES
        DECB
        BEQ *+4
        BRA BMULTX
        RTS
*
INIT2   LDAB #6         INITIALISE ALL PROGRAM FLAGS
        STAB MULT1
        STAB SCANO
        STAB FINISH
        STAB START
        LDX #1
        STX TF          COUNTER FOR NUMBER OF SCANS
        LDX #0
        STX GOAD        CLEAR DISPATCHER TABLE - 6 BYTES
        STX GOAD+2
        STX GOAD+4
        STX BYTEO       CLEAR BYTEO AND CFLG
        STX OUTPUT      CLEAR OUTPUT BUFFER ( 2 BYTES )
        STX ADDBUF      CLEAR TEMP. ADDRESS BUFFER
        STX INPUT       CLEAR INPUT BUFFER ( 2 BYTES )
        STX AFLAG       CLEAR AFLAG AND BFLAG
        STX ERR316      CLEAR ERR316 AND PAS5
        STX AO
        STX CB2CTR
        STX CA2CTR
        STX IFLG        CLEAR IFLG AND CHAN
        STX MUXM        CLEAR MUXM AND CHAN7
        STX CHAN9       CLEAR CHAN9 AND LABYTE
        CLRB
        STAB CB1CTR
        LDAA #10
        LDX #MCTR
CLEAR   STAB X          CLEAR MCTR ( HADIOS PARAMETERS )
        DECA            BUFFER
        BEQ CLRFIN
        DEX
        BRA CLEAR
CLRFIN  STAB FREQ
        RTS
*
XMODE   LDAA MODE
        BEQ ONLINE
        LDAA #16
        STAA MCTR
        JSR INITCB      SEND 10 DUMMY PARAMETERS TO H316
        JMP DISP
*
```

```
ONLINE LDX AVECO
       STX AO
       LDAA TEMP+1
       STAA ADD1
       LDAA TEMP
       STAA ADD1+1
       LDX ADD1
       STX ADD10          SAVE ADDRESS A(0)
*
*DEVICE TEST --- CFLG=1 IF NO ANALOGUE INPUTS REQUIRED
*
       STS STACK          SAVE STACK POINTER.LOAD A-ARRAY
       LDS #MCTR          AND STORE INTO MCTR.CHECK VALUES
       LDX AVECO
       LDAA 7,X           GET A(1) - HADIOS DEVICE CODE
       PSHA
       ANDA #1            ANALOGUE INPUTS ONLY ? IF YES,SKIP
       BNE VIAO           COUNTER TESTS
       LDAA #1
       STAA CFLG          SET 'COUNTER(S) ONLY' FLAG
       LDAA MCTR
       ANDA #2            COUNTER 1 ONLY ?
       BNE VIA3
       LDAA MCTR
       ANDA #4            COUNTER 2 ONLY ?
       BNE VIA3
       LDAA MCTR
       ANDA #8            COUNTER 3 ONLY ?
       BNE VIA3
       JMP ERRO           HADIOS DEVICE NOT CORRECTLY
AO     RMB 2              SPECIFIED !
CHANCK SUBA #47           ANALOGUE CHANNELS REQUIRED
       BLS RTCHEK         IN RANGE ?
       JMP ERRO
RTCHEK RTS
FINISH RMB 1
*
VIAO   LDAA 31,X          GET A(5) - DELAY FACTOR IN
       BNE VIA3           ANALOGUE INPUTS MULTIPLEXING
       JMP ERR1           OPERATION.ZERO NOT RECOMMENDED !
*
*
VIA3   PSHA
       LDAA 19,X          GET A(3) - FIRST ANALOGUE CHANNEL
       PSHA
       BSR CHANCK         CHECK RANGE : 0 TO 47 ONLY
       LDAA 25,X          GET A(4) - LAST ANALOGUE CHANNEL
       PSHA
       BSR CHANCK         CHECK RANGE : 0 TO 47 ONLY
       LDAA 37,X          GET A(6) - COUNTER 1 SCAN TYPE
```

```
        PSHA            0 = NON-INTERRUPT MODE
        LDAA 43,X       GET A(7) - COUNTER 1 PRESET VALUE
        PSHA            0 TO 255
        LDAA 49,X       GET A(8) - COUNTER 2 SCAN TYPE
        PSHA            1 = COUNTERS INTERRUPTS ENABLED
        LDAA 55,X       GET A(9) - COUNTER 2 PRESET VALUE
        PSHA
        LDAA 61,X       GET A(10) - COUNTER 3 SCAN TYPE
        PSHA
        LDAA 67,X       GET A(11) - COUNTER 3 PRESET VALUE
        PSHA
        LDS STACK       RESTORE STACK POINTER
*CALCULATE BEGINNING CHANNEL  ADDRESS IN PREADC
*
        JSR PREADC
        JSR INITCB      SEND 10 HCDIOS PARAMETERS TO H316
        BRA CLOCK
STACK   RMB 2
        RMB 9
MCTR    RMB 1
*CB2 IS LOW AND CA2 IS HIGH ON HARDWARE RESET
*SO ENSURE BOTH ARE HIGH INITIALLY
*CLEAR IRQ FLAGS FROM SPURIOUS CB1'S,CA1'S
*
PIAOAB  LDAA #%111000
        STAA CRAO       A-SIDE CONTROL REGISTER
        CLRB
        STAB DDRAO      A-SIDE DATA DIRECTION REGISTER
        STAB CRBO       B-SIDE CONTROL REGISTER
        COMB
        STAB DDRBO      B-SIDE DATA DIRECTION REGISTER
        STAA CRBO
        LDAB #%100100   NORMAL HANDSHAKE-MODE
        STAB CRBO
        STAB CRAO
        JSR CA1DAT      CLEAR SPURIOUS IRQA1 FLAG
        STAB CRAO
        JSR SETCRA      SET CRA TO %100101 IF OFF-LINE
        LDAB IORBO      CLEAR SPURIOUS IRQB1 FLAG
        RTS
*
*SEND HADIOS OR DUMMY PARAMETERS NOW --- 10 CB2 INTERRUPTS
*
INITCB  LDX #MCTR
        LDAB #10
VIA2    LDAA X
        JSR SENCB2
        DEX
        DECB
        BEQ INITOV
```

```
        BRA VIA2
INITOV  RTS
BYTEO   RMB 1
CFLG    RMB 1
XSAVEO  RMB 2
MUXM    RMB 1
CHAN7   RMB 1
*
*CONFIGURE MP-T INTERRUPT TIMER
*
CLOCK   SEI
        LDAA :+TIMER+5  SPECIFY TIMER IN SD BASIC
        STAA FREQ
        LDX #£F530
        LDAB #£FF
        STAB 2,X
        LDAA #£3C       SET CONTROL REGISTER FOR FIRST DUMMY
        STAA 3,X        INTERRUPT REQUEST
        LDAA FREQ
        STAA 2,X
        NOP
        CLI
WAICLK  LDAA 3,X        WAIT FOR FIRST MP-T 'INTERRUPT'
        BMI OUT1
        BRA WAICLK
OUT1    LDAA 2,X        CLEAR IRQB1 FLAG
        LDAA #£3D
        STAA 3,X
        JMP DISP
*
FREQ    RMB 1
SCANO   RMB 1
*
* IRQ INTERRUPT RESPONSE CODE
*
POLL    LDX #£F530
        LDAA 3,X
        BMI CLKDAC      CLOCK INTERRUPT
        LDAA CRA0
        BMI CA1RES      CA1 INTERRUPT
        JMP ERRIRQ      ERROR IRQ !
        RTI
*
CLKDAC  JMP CLKRES
*
ERRIRQ  LDAA CHAN7      IRQ INTERRUPT NOT RECOGNISED
        BEQ SYS1        CHAN7=0 INTERRUPT WAS THERE FROM
        BSR PRINT       BEGINNING.
        BRA SYS2
SYS1    JSR SYSOFF
```

```
SYS2    LDX #VDU12
        JMP GGG-4           BACK TO MIKBUG  EVENTUALLY
ERRCA1  LDAA CHAN7          CA1 INTERRUPT NOT RECOGNISED
        BEQ SYS3
        BSR PRINT
        BRA SYS4
SYS3    JSR SYSOFF
SYS4    LDX #VDU13
        JMP GGG-4
*
PRINT   LDX #VDU14
        JSR PDATA1
        RTS
*
CA1RES  LDAA #%100100       TURN OFF INTERRUPT MODE
        STAA CRA0
        JSR DATA            ACKNOWLEDGE WITH CA2 INTERRUPT !
        ANDA #1             FIRST H316 CA1 INTERRUPT MUST
        BEQ HWTOMC          ACCOMPANY A " 1 " !
        JMP ERRCA1
        RTI
*
HWTOMC  JSR CA1WAI
        JSR DATA
        TAB
        DECB
        STAB NWORDS         NUMBER OF H316 WORDS (2*N BYTES)
        LDX #ADBUF2         LOAD X WITH BUFFER ADDRESS
REPEAT  JSR CA1WAI          MAXIMUM - 30 WORDS ( 60 BYTES ) ONLY
        JSR DATA            GET H316 BYTE AND ACKNOWLEDGE WITH
        STAA 0,X            CA2 INTERRUPT !
        JSR CA1WAI
        JSR DATA
        STAA 1,X
        INX
        INX
        DECB
        BNE REPEAT
        JSR SETCRA          TURN ON INTERRUPT-MODE
        LDX #TASK2          PLACE JOB POINTER IN DISPATCHER
        STX GOAD
        RTI                 EXIT CA1 INTERRUPT RESPONSE
*
CLKRES  LDX #£F530          CLOCK PIA BASE ADDRESS
        LDAA 2,X            CLEAR IRQB FLAG
        LDAA SCAN0          FIRST SCANNING INTERRUPT FLAG
        BNE PASS            M6800 SCANS AT THE FIRST
        LDX A0              CLOCK INTERRUPT
        DEX
        BEQ SCAN            IF X=0 THEN SCANNING INTERRUPT
```

```
        STX AO              ELSE EXIT AND TRY NEXT TIME !
        RTI
SCAN    LDAA IFLG           CHECK INTERRUPT FLAG.
        BEQ PASS
        JMP ERR4            SCANNING INTERVAL TOO SHORT !
PASS    LDX AVECO           RESTORE A(O) COUNTER IN AO
        LDX 0,X
        STX AO
        CLRA
        STAA SCANO
        LDX #TASK1          PLACE JOB POINTER IN DISPATCHER
        STX GOAD+2
        RTI                 EXIT SCANNING INTERRUPT RESPONSE
ADBUF2  RMB 60
*
* NMI INTERRUPT RESPONSE CODE - BOUNCY SWITCH
* NMI MUST BE USED FOR USER-INTERFERENCE
*
NMIRES  LDX #VDU9
        JSR PDATA1
        JSR BEEP1           BEEP ONCE
RTPERR  JSR SYSOFF          TURN-FF CLOCK IF ON-LINE
        INC START
        JMP MON             BACK TO MIKBUG MONITOR
*
MYSWI   LDX #VDU16          MYSWI IS USER SWI INTERRUPT
        JSR PDATA1          RESPONSE CODE
        JSR BEEP1
        JSR SYSOFF          TURN OFF CLOCK IF ON-LINE
        JSR ERRCB2          SEND A CB2 TO RESET HADIOS EXEC
        LDX #MONSWI         RESTORE MONITOR SWI VECTOR
        STX SWIVEC
        RTI                 RETURN TO INTERRUPTED PROGRAM
*
JERRO   JMP ERRO
*
PREADC  LDX #MCTR-3         ROUTINE TO CALCULATE FIRST CHANNEL
        LDAA 0,X            ADDRESS FOR B-ARRAY
        LDAB 1,X            SAVE PERMANENT COPY IN ADD10
        SBA
        BMI JERRO           CHANNELS INCORRECTLY SPECIFIED
        INCA
        STAA CHANX+1        CHANX+1,MUXM = A(4)-A(3)+1
        STAA MUXM
        LDAA 1,X
        LDAB #0
        SBA
        INCA
        STAA CHANX
        BSR MULT
```

```
        BSR ADDW
        LDAA ADD1+1
        STAA ADDBUF
        STAA ADD10
        LDAA ADD1
        STAA ADDBUF+1
        STAA ADD10+1
        RTS
*
*ADDBUF CONTAINS FIRST B CHANNEL ADDRESS --- DEFAULT B(0)
*
MULT    CLRA            AN 8 BIT BY 8-BIT MULTIPLY ROUTINE
        CLRB            NOTE: COULD HAVE JUST UPDATED ADDRESS
        LDX #8          USING X REG. AS IN PREVIOUS METHOD
SHIFT   ASLB
        ROLA
        ASL MULT1
        BCC DECR


        ADDB CHANX
        ADCA #0
DECR    DEX
        BNE SHIFT
        STAA ADD2+1
        STAB ADD2
        RTS
MULT1   RMB 1
TEMP    RMB 2
ADDBUF  RMB 2
*
ADDW    LDAB BYTENO
        CLC
        LDX #ADD1
MCADD   LDAA X
        ADCA 2,X
        STAA X
        INX
        DECB
        BNE MCADD
        RTS
BYTENO  FCB 2
ADD1    RMB 2
ADD2    RMB 2
ADD10   RMB 2
*
* THE M6800 DISPATCHER TABLE
*
DISP    LDX GOAD        ANY NEW DATA FROM H316 ?
        BNE HWCA1       IF YES,DATA IN ADBUF2 BUFFER
        LDX GOAD+2      CLOCK INTERRUPT REQUEST ?
        BNE CLKVEC      IF YES,BEGIN A HADIOS SCAN
```

```
        LDAA GOAD+4       DATA TRANSFER IN OFF-LINE MODE ?
        BEQ DISP          NO..CONTINUE POLLING
        CLR GOAD+4        CLEAR FLAG
        LDS SPSAVE        RESTORE STACK POINTER
        RTS               RETURN TO SD BASIC
*
* IE. IF MODE=1 RETURN AFTER CALL SUB1(A(0),B(0))
*
HWCA1   SEI               NO IRQ INTERRUPTS DURING
        JSR 0,X           H316 DATA TRANSFER TO B-ARRAY
        CLI
        JMP DISP          RETURN TO DISPATCHER LOOP
*
CLKVEC  JMP 0,X           GOTO CLOCK INTERRUPT RESPONSE
*
TASK2   STS STACK1        SAVE STACK POINTER
        LDX #0
        STX GOAD          CLEAR H316 DATA TRANSFER POINTER
        LDAB NWORDS       B = NO. OF H316 WORDS
        LDX B57           B(57) ADDRESS IN X
LOAD1   LDS #ADBUF2-1     LOAD SP WITH TOP OF ADBUF2
RELOAD  PULA              PULL MSBYTE OFF STACK
        STAA 0,X          STORE IN B-ARRAY
        PULA              PULL LSBYTE OFF STACK
        STAA 1,X          STORE IN B-ARRAY
        DECB
        BEQ TASFIN        FINISH ?
        INX               BUMP X SIX TIMES
        INX
        INX
        INX
        INX
        INX
        BRA RELOAD
TASFIN  INC GOAD+4        BUMP OFF-LINE FLAG IN DISPATCHER
        LDS STACK1        RESTORE ( RTP ) STACK POINTER
        RTS               RETURN TO DISPATCHER
STACK1  RMB 2
*
NWORDS  RMB 1
*
TASK1   LDX #0            BEGIN SCANNING INTERRUPT ROUTINE
        STX GOAD+2        CLEAR ITS POINTER FIRST
        INC IFLG          BUMP INTERRUPT FLAG
        JSR A2CHEK        IS NUMBER OF SCANS = REQUIRED  ?
        LDAA MUXM         YES,THEN THIS IS LAST SCAN.
        STAA CHANX+1      = A(4)-A(3)+1
        BRA NOANAG
CHANX   RMB 2
*
```

```
*READ H316 DATA BYTES NOW 2 * ( CHANX+1 ) BYTES
*ONE CA2/CA1 HANDSHAKE PER BYTE TRANSFERRED
*
NOANAG LDAA CFLG          COUNTER(S) ONLY ?
       BNE NONE
ANAG   JSR DATAIN         NO,GO & READ ANALOGUE INPUT BYTES
*
ENSM   LDAA #1            2 BYTES FOR ENSEMBLE NUMBER
       STAA CHANX+1
       LDX AVEC5          LOAD ADDRESS OF A(5)
       STX ADDBUF
       JSR DATAIN         GO AND READ BYTES VIA CA2/CA1
*
NONE   LDAA MCTR
       ANDA #2            COUNTER 1 ?
       BEQ VIA5
       LDX C1B48          LOAD ADDRESS OF B(48)
       BSR CTRDAT         GO TO COUNTER DATA ROUTINE
*
VIA5   LDAA MCTR
       ANDA #4            COUNTER 2 ?
       BEQ VIA6
       LDX C2B51          LOAD ADDRESS OF B(51)
       BSR CTRDAT         GO TO COUNTER DATA ROUTINE
*
VIA6   LDAA MCTR
       ANDA #8            COUNTER 3 ?
       BEQ VIA7
       LDX C3B54          LOAD ADDRESS OF B(54)
       BSR CTRDAT         GO TO COUNTER DATA ROUTINE
*
VIA7   LDAA #1            SET TEMPORARY FLAG
       STAA CHAN9
       LDX #MWORDS        GET TOP WORD OF BUFFER BUF1 BSZ 31
       JSR XXCA1          OF HADIOS EXECUTIVE
*
VIA8   LDX MWORDS
       BEQ VIA8X           IF ZERO THEN NO H316 DATA TO
       STX SWORDS         BE RETRIEVED..IF NON-ZERO THEN
       LDAA MWORDS+1      READ M WORDS ( (A)-1 ) INTO B(57) ONWARDS.
       DECA
       LDX B57
       JSR XXCA1
*
VIA8X  CLR AFLAG          CLEAR PIA A & B-SIDE PROGRAM
       CLR BFLAG          FLAGS
       CLR CHAN9
       JSR SETCRA
       LDS SPSAVE
       RTS
```

```
MWORDS RMB 2
SWORDS RMB 2
*
XXCA1  STAA CHANX+1
       STX ADDBUF
       JSR DATAIN
       RTS
*
CTRDAT LDAA #3          6 BYTES FOR EACH COUNTER
       STAA CHANX+1     SET UP CA2-CA1 COUNTER
       STX ADDBUF       SET UP COUNTER ADDRRESS IN B-ARRAY
       INC CHAN9
NFIRST JSR DATAIN       GO READ DATA VIA CA2-CA1 HANDSHAKE
       CLR CHAN9
       RTS
*
SETCRA LDAA MODE        IF ON-LINE THEN PIA A-SIDE
       BEQ RSTACK       REMAIN IN HANDSHAKE MODE
       LDAA #%100101    ELSE IN CA1 INTERRUPT MODE.
ACR    STAA CRA0
       RTS
RSTACK LDAA #%100100
       BRA ACR
*
*RETURN TO SD BASIC AFTER CALL SUB1(A(0),B(0))
XSAVE2 RMB 2
*
* SD BASIC STATEMENT --- CALL SUB3(N,U)
* N = DGOA CHANNEL NUMBER ( 0 < N < 15 )
* U = 16-BIT INTEGER CONTROL OUTPUT ( 0 < U < 32767 )
*
SUB3   STX XSAVER       SAVE INDEX REGISTER
       JSR SPBYTE       GET FIRST ARGUMENT ON STACK
       STAA NCHAN
       LDAB #15
       JSR NMTEST       IS N WITHIN RANGE ?
       JSR SENCB2       YES,SEND IT TO H316
       JSR SPDATA       GET 2 BYTE CONTROL OUTPUT
       LDAB PLUS1
       JSR CBDATA       SEND THEM TO H316
       RTS
*
NMTEST PSHA             SAVE (A) ON STACK
       SBA              (A) = (A) - (B)  AS BINARY NUMBERS
       BLS *+5          IF < 15 THEN RESTORE (A) AND RETURN
       JMP ERR9         ERROR IF N > 15 !
       PULA
       RTS
*
SPBYTE LDX 10,X         LOAD SECOND ARGUMENT ON STACK
```

```
        LDAA 5,X        NORMALLY 'N' OR 'M' FROM SUB3 OR SUB4
        STAA CHAN
        RTS
*
SENCB2  STAA BYTEO      SAVE CB2 OUTPUT BYTE
        JSR CB2OPO      SEND BYTE TO H316 VIA CB2/CB1 HANDSHAKE
WAITCB  LDAA CRBO       IRQB1 FLAG SET BY CB1 ACKNOWLEDGE ?
        BMI OUTCB       BRANCH OUT IF SET
        BRA WAITCB      ELSE WAIT FOR CB1.
OUTCB   LDAA IORBO      CLEAR IRQB1 FLAG
        LDAA CRAO       IS IRQA1 FLAG SPURIOUSLY SET ?
        BMI XCA1
        RTS             RETURN IF NOT.
XCA1    CLR BFLAG       IF YES THEN OUTPUT MESSAGE VIA SWI
        SWI             AND RETURN TO MIKBUG.
        SWI
*
SPDATA  LDX XSAVER      THIS ROUTINE IS USUALLY USED TO
        LDX 4,X         CONSECUTIVE WORDS FROM ADDRESS
        STX XSAVER      IN STACK.....SEE SUB4(M,D(O))
ADDX4   LDX 4,X
        STX OUTPUT
        RTS
*
XPLUS6  INX             BUMP X SIX TIMES
        INX
XPLUS4  INX             BUMP X FOUR TIMES
        INX
        INX
        INX
        INX
        RTS
*
CBDATA  LDAA OUTPUT     GET MSBYTE OF OUTPUT WORD.
        BRA VIA9        BRANCH TO CB2 O/P ROUTINE
LSB     LDAA OUTPUT+1   GET LSBYTE OF OUTPUT WORD.
        BSR IN253       CHECK FOR RANGE : 0 <= LSBYTE < 254
VIA9    JSR SENCB2      SEND MSBYTE VIA CB2/CB1 HANDSHAKE
        LDAA BFLAG      BFLAG SHOULD BE ZERO FOR MSBYTE
        BNE VIA9X       TRANSFER...BUMP IT FOR LSBYTE TRANSFER.
        INC BFLAG
        JMP LSB
VIA9X   CLR BFLAG       CLEAR BFLAG AFTER EVERY WORD TRANSFER
        DECB            DECREMENT WORD COUNTER
        BEQ CB2FIN      CB2 TRANSFERS OVER IF (B) ZERO.
        LDX XSAVER      FETCH MORE CONSECUTIVE WORDS FROM
        JSR XPLUS6      ADDRESS IN STACK...BUMP SIX TIMES
        STX XSAVER      AS ADDRESS VARIABLE IS 6-BYTES LONG.
        LDX 4,X         LOAD X WITH VALUE OF LOCATION IN X !
        STX OUTPUT      STORE IN 2 BYTE OUTPUT BUFFER
        JMP CBDATA      JUMP TO BEGIN CB2 TRANSFER
```

```
CB2FIN RTS
*
IN253   CMPA D253       COMPARE (A) WITH 253
        BLS OUT253      BRANCH IF LESS OR EQUAL TO 253 ( BINARY )
        DECA            MUST BE > 253 , SO DECREASE IT BY ONE
        BRA IN253       GO AND CHECK AGAIN.
OUT253 RTS
*
XSAVER FDB 0
D253    FCB 253
D254    FCB 254
D255    FCB 255
PLUS1   FCB +1
*
* SD BASIC STATEMENT --- CALL SUB4(M,D(0))
* M=NUMBER OF DATA WORDS INCLUDING D(0) : M <= 30
* USER MUST ENSURE AT SD BASIC LEVEL THAT D(0) IS
* INTEGER AND LESS OR EQUAL THAN 32767.
*
SUB4    STX XSAVER      SAVE X AS IT POINTS TO LAST ARGUMENT
        JSR SPBYTE      ADDRESS...GET FIRST ARGUMENT 'M'
        STAA MDATA      SAVE FOR DEBUGGING PURPOSES
        BEQ JERR9       YOU CAN'T SEND NOTHING !
        LDAB #30        IS M WITHIN RANGE ?
        JSR NMTEST      GO AND ASCERTAIN
        LDAA D254       PREPARE TO SEND A '254' CONTROL BYTE
        JSR SENCB2      TO H316 FOR M6800 DATA TRANSFER
        LDAA CHAN       FETCH 'M' AND SEND IT TO
        JSR SENCB2      H316 VIA A CB2/CB1 HANDSHAKE
        JSR SPDATA      GET D-ARRAY WORDS
        LDAB CHAN       (B) IS WORD COUNTER : (B)=M
        JSR CBDATA      SEND DATA WORDS VIA CB2/CB1 ROUTINE
        RTS             RETURN TO SD BASIC
JERR9   JMP ERR9
*
NCHAN   RMB 1
MDATA   RMB 1
SPSAVE  RMB 2
*
*
*SD BASIC STATEMENT --- CALL SUB2
*
SUB2    LDAA MODE       ON-LINE MODE ?
        BEQ NORM        BRANCH IF YES
        JSR A2CHEK      ELSE IS NUMBER OF SCANS = REQUIRED ?
NORM    CLR IFLG        CLEAR INTERRUPT FLAG
        LDAA FINISH     FINISH = 0 IF LAST SCAN IS DONE
        BEQ VIA10
        RTS
```

```
        RTS
*RETURN TO SD BASIC AFTER CALL SUB2
A2CHEK  LDX TF          TF IS SCAN COUNTER
        CPX AVEC2N      COMPARE WITH REQUIRED IE. A(2)
        BNE NORMAL
        CLR FINISH      CLEAR FINISH FLAG WHEN SCANS OVER
        RTS
NORMAL  LDX TF          BUMP SCAN COUNTER BY ONE
        INX
        STX TF
        RTS
*
VIA10   JSR SYSOFF      IF SCANS OVER,GOTO HOUSEKEEPING ROUTINE
        LDX #VDU4       AND EXIT.
        JMP GGG
*
TF      RMB 2
CB2OPO  LDAA #%100100   SET B-SIDE TO HANDSHAKE MODE
        STAA CRBO
        LDAA BYTEO      FETCH BYTE TO BE OUTPUT
CB2     STAA IORBO      STORE IN B-SIDE I/O REGISTER
        INC CB2CTR
        RTS             RETURN TO WAIT FOR CB1 ACKNOWLEDGE
INPUT   RMB 2
GOAD    RMB 6
ERR316  RMB 1
PAS5    RMB 1
*
DATAIN  LDAA #%100100   SET A-SIDE TO HANDSHAKE MODE
        STAA CRAO
CA2     LDAA IORAO      READ INPUT BYTE ON I/O REGISTER..THIS
        INC CA2CTR      SENDS OUT A LOW-ACTIVE CA2 !!
        JSR CA1WAI      GOTO 'WAIT FOR CA1' ROUTINE
        JSR CA1DAT      GOTO 'GET THE INPUT BYTE' ROUTINE
        BRA GETDAT
        END
#

        BRA GETDAT      BEGIN CA2/CA1 ROUTINE
*
CA1WAI  LDAA CRAO       IS IRQA1 FLAG SET ?
        BMI CA1IN       IF MINUS,A LOW-ACTIVE CA1 HAS OCCURRED
        BRA CA1WAI      ELSE WAIT FOR A CA1.
CA1IN   LDAA CRBO       IS IRQB1 FLAG SPURIOUSLY SET ?
        BMI XCB1
        RTS             RETURN IF NOT.
XCB1    CLR AFLAG       IF YES THEN OUTPUT MESSAGE VIA SWI
        SWI             AND RETURN TO MIKBUG
        SWI
CB1CTR  FCB 0
*
CA1DAT  LDAA #%100000   'EXOR'ED CRA TO DISENABLE CA2 OUTPUT
        EORA CRAO       WHEN READING INPUT BYTE IN ON-LINE
        STAA CRAO       CASE ONLY...IF OFF-LINE THEN CA2 IS
DATA    LDAA IORAO      OUTPUT ON THIS READ !!
        STAA LABYTE     SAVE FOR POSSIBLE DEBUGGING
        RTS
CHAN9   RMB 1
```

```
LABYTE RMB 1
*
GETDAT INC CA2CTR+1
       LDAB AFLAG        AFLAG = 0 WHEN MSBYTE IS BEING READ
       BNE LSBYTE        AFLAG = 1 ....READING LSBYTE
MSBYTE STAA INPUT        SAVE MSBYTE
       LDAA CHAN9        NO DATA CHECKS IF H316 DATA TRANSFER !
       BNE NOERR         IE..WHEN CHAN9 NON-ZERO.
       LDAA INPUT        RESTORE MSBYTE
       ANDA #%11111100   MSBYTE > %11 ? MAXIMUM ANALOGUE
       BEQ NOERR         INPUT IS 1023 FOR 10-BIT ADC.
       ANDA #%10000000   MSBYTE = £80 IS 'DATA > 1023' ERROR CODE
       BEQ LE1023        ELSE MSBYTE WAS < 1023 , MUST BE OTHER ERRO
       INC PAS5          BUMP MSBYTE ERROR FLAG
LE1023 LDAA #1           SET AN ERROR FLAG FOR LATER USE
       STAA ERR316
NOERR  INC AFLAG         BUMP AFLAG TO READ LSBYTE
       BRA VIA4
LSBYTE STAA INPUT+1      SAVE LSBYTE
       LDAA ERR316       IS ERROR FLAG SET ?
       BEQ VIA13         IF NOT , CONTINUE READING NEXT 2 BYTES IF A
       LDAA INPUT+1      IF YES , DECODE THE ERROR...PREVIOUS LSBYTE
       ANDA #1           DID YOU SPECIFY A COUNTER BEING USED BY
       BEQ FORTRN        THE H316 USER ?
       JMP ERR7          REPORT COUNTER ERROR
FORTRN LDAA INPUT+1      DECODE ERROR BYTE AGAIN
       ANDA #2           IS IT A M6800 FORTRAN LIBRARY ERROR ?
       BEQ G1023
       JMP ERR8          REPORT A FORTRAN LIBRARY ERROR
G1023  LDAA PAS5         WAS LAST ANALOGUE INPUT DATA > 1023 ?
       BEQ VIA13
       JMP ERR2          YES....
VIA13  CLR AFLAG         IF NO ERRORS THEN READ MORE BYTES
       LDX ADDBUF        LOAD VARIABLE ADDRESS FOR INPUT WORD
       LDAA INPUT        STORE IN SD BASIC VARIABLE IN X
       STAA 0,X
       LDAA INPUT+1
       STAA 1,X
       LDAB CHANX+1
       DECB              DECREMENT WORD COUNTER
       BEQ RTS           EXIT CA2/CA1 ROUTINE IF OVER
       STAB CHANX+1
       JSR XPLUS6        IF MORE , BUMP X TO NEXT VARIABLE
       STX ADDBUF
VIA4   JMP DATAIN        CONTINUE READING IN BYTES
RTS    LDX ADD10         RESTORE FIRST ANALOGUE CHANNEL ADDRESS
       STX ADDBUF        IN ADDBUF
       RTS
AFLAG  RMB 1
BFLAG  RMB 1
```

```
*
OUTPUT RMB 2
IFLG   RMB 1
CHAN   RMB 1
CB2CTR RMB 1
CTR255 RMB 1
CA2CTR RMB 2
*
* SD BASIC STATEMENT CALL SUB99(EN,LN)
* EN = ERROR NUMBER FROM SD BASIC 'ERR' FUNCTION
* LN = LINE  NUMBER FROM SD BASIC 'ELN' FUNCTION
*
SUB99  STX XSAVER       THIS ROUTINE PRINTS RUN-TIME ERROR
       JSR SYSOFF       NUMBER AND LAST SD BASIC LINE NUMBER
       LDX #VDU99       IN HEXADECIMAL
       JSR PDATA1
       LDX XSAVER
       LDX 10,X         LOAD FIRST ARGUMENT ADDRESS FROM STACK
       JSR XPLUS4       BUMP X FOUR TIMES TO GET CORRECT
       JSR OUT4HS       BYTE ADDRESS.
       LDX #VDU100
       JSR PDATA1
       LDX XSAVER
       LDX 4,X          LOAD SECOND ARGUMENT ADDRESS FROM STACK
       JSR XPLUS4       POINT TO CORRECT BYTES
       JSR OUT4HS
       JSR BEEP1
       JMP RTPERR
*
CLKOFF LDAA MODE        IF ON-LINE THEN TURN OFF CLOCK
       BNE RETURN       ELSE RETURN
       SEI
       LDX #£F530
       LDAA 2,X         CLEAR CLOCK PIA IRQB1 FLAG
       LDAA #£80        OUTPUT £80 TO MP-T TIMER TO
       STAA 2,X         DISENABLE IT.
       CLRA
       STAA 3,X
       NOP
       CLI
RETURN RTS
*
BEEP1  LDAA #7          SOUND A BEEP !
       JSR OUTCH
       RTS
*
ERRCB2 LDAA #255        THIS CB2/CB1 HANDSHAKE IS USED ONLY
       STAA BYTE0       TO RE-INITIALISE THA HADIOS EXECUTIVE
       JSR CB2OP0       FOR A SUBSEQUENT M6800 COMMUNICATION.
WAI255 LDAA CRB0        NOTE THE '255' ERROR CODE.
```

```
        BMI OUT255
        BRA WAI255
OUT255  LDAA IORB0
        INC CTR255
        RTS
*
* FOLLOWING ARE ERROR MESSAGES REPORTED BY THE THE
* M6800 EXECUTIVE.THEY ARE SELF-EXPLANATORY.
*
ERR0    LDX #VDU1
        JMP GGG
ERR1    LDX #VDU2
        JMP GGG
*
ERR2    JSR SYSOFF
        JSR ERRCB2
        LDX #VDU3
        JMP GGG
*
ERR4    JSR SYSOFF
        LDX #VDU6
        JMP GGG
ERR7    JSR SYSOFF
        LDX #VDU7
        JMP GGG
ERR8    JSR SYSOFF
        LDX #VDU8
        JMP GGG
ERR9    JSR SYSOFF
        LDX #VDU15
        JMP GGG
ERR11   JSR CLKOFF
        LDX #VDU11
        JMP GGG
*
        JSR ERRCB2
        CLI
GGG     JSR PDATA1
        JSR BEEP1
        INC START
        JMP ORIGIN
*
SYSOFF  JSR CLKOFF
        JSR ERRCB2
        RTS
*
VDU     FCB £0D,£0A
        FCC /MC6800 EXECUTIVE 1981/
        FCB £0D,£0A,£04
VDU0    FCB £0D,£0A
```

```
        FCC /HIGH HEXADECIMAL ADRRESS IS /
        FCB £04
VDU1    FCB £0D,£0A
        FCC /HADIOS DEVICE NOT SPECIFIED ! /
        FCB £0D,£0A,£04
VDU3    FCB £0D,£0A
        FCC /H316 DATA >1023 !/
        FCB £0D,£04
VDU2    FCB £0D,£0A
        FCC /ENSEMBLE NO. ZERO ! /
        FCB £0D,£0A,£04
VDU4    FCB £0D,£0A
        FCC /ALL MC6800 SCANS DONE ! /
        FCB £0D,£0A,£04
VDU5    FCB £0D,£0A
        FCC /HIGH SD BASIC ADDRESS IS /
        FCB £04
VDU6    FCB £0D,£0A
        FCC /SAMPLING TIME TOO FAST ! /
        FCB £04
VDU7    FCB £0D,£0A
        FCC /COUNTER CODE ERROR ! /
        FCB £04
VDU8    FCB £0D,£0A
        FCC /FORTRAN ERROR IN THE H316 !/
        FCB £04
VDU9    FCB £0D,£0A
        FCC /USER NMI INTERRUPT --- PRESS RESET NOW !/
        FCB £04
VDU10   FCB £0D,£0A
        FCC /ON-LINE MODE ?  ANSWER Y OR N /
        FCB £04
VDU11   FCB £0D,£0A
        FCC /ERROR IN H316-M6800 DATA TRANSFER !/
        FCB £04
VDU99   FCB £0D,£0A
        FCC /RTP ERROR  /
        FCB £04
VDU12   FCB £0D,£0A
        FCC /UNIDENTIFIED IRQ INTERRUPT ! /
        FCB £04
VDU13   FCB £0D,£0A
        FCC /UNIDENTIFIED H316 INTERRUPT !/
        FCB £04
VDU14   FCB £0D,£0A
        FCC /HADIOS EXEC. NOT INITIIALISED - PRESS RESET !/
        FCB £04
VDU15   FCB £0D,£0A
        FCC /N OR M IN SUB3 OR SUB4 OUT OF RANGE !/
        FCB £04
```

```
VDU16   FCB £0D,£0A
        FCC /SPURIOUS CA1,CB1 - CHECK INTERFACE !/
        FCB £0D,£0A,£04
VDU100  FCC /AT ABOUT LINE  /
        FCB £04
END     RMB 1
        END
#
*
```

Table A4.3 The ADT1-8 program

```
                                  *  ADTI-8 RELOCATION
0001                              *  ADTI-8 RELOCATION
0002                              *
0003                                    ORG   '4757
0004 04757    0 04 00770                STA   '770
0005                                    ORG   '4761
0006 04761    0 04 00771                STA   '771
0007                                    ORG   '4763
0008 04763    0 04 00773                STA   '773
0009 04764    0 15 00772                STX   '772
0010                                    ORG   '4766
0011 04766    0 04 00775                STA   '775
0012                                    ORG   '5006
0013 05006    000773                    OCT   '773
0014 05007    0 04 00773                STA   '773
0015 05010    0 12 00771                IRS   '771
0016                                    ORG   '5023
0017 05023    0 02 00773                LDA   '773
0018                                    ORG   '5025
0019 05025    0 06 00771                ADD   '771
0020                                    ORG   '5027
0021 05027    0 04 00774                STA   '774
0022                                    ORG   '5034
0023 05034    0 04 00766                STA   '766
0024 05035    0 07 00774                SUB   '774
0025                                    ORG   '5037
0026 05037    0 04 00767                STA   '767
0027 05040   -0 02 00766                LDA*  '766
0028 05041   -0 04 00767                STA*  '767
0029 05042    0 12 00766                IRS   '766
0030 05043    0 12 00767                IRS   '767
0031                                    ORG   '5045
0032 05045    0 07 00766                SUB   '766
0033                                    ORG   '5050
0034 05050    0 02 00767                LDA   '767
0035                                    ORG   '5061
0036 05061    0 07 00774                SUB   '774
0037                                    ORG   '5065
0038 05065    0 02 00773                LDA   '773
0039                                    ORG   '5070
0040 05070    0 04 00766                STA   '766
0041                                    ORG   '5074
0042 05074    0 12 00766                IRS   '766
0043                                    ORG   '5076
0044 05076    0 02 00771                LDA   '771
0045                                    ORG   '5100
0046 05100    0 04 00766                STA   '766
0047                                    ORG   '5104
0048 05104    0 12 00766                IRS   '766
0049                                    ORG   '5112
0050 05112    0 02 00771                LDA   '771
0051                                    ORG   '5114
0052 05114    0 02 00770                LDA   '770
0053                                    ORG   '5123
0054 05123    0 02 00775                LDA   '775
0055                                    ORG   '5126
0056 05126    0 35 00772                LDX   '772
0057                                    END
```

Table A4.4 The BASIC I/O MOD program

```
                                *                                PAGE          1
0001                            *
0002                            * I/O MOD for BASIC . If sense switch 3 is
0003                            * set,then i/p via paper tape else VDU.
0004                            * If sense switch 4 is set then o/p
0005                            * via paper-tape punch else VDU.
0006                            *
0007                                 REL
0008 00000   101004         AA      SS3
0009 00001   0 01 00004              JMP    *+3
0010 00002   140040                 CRA
0011 00003   0 04 00105              STA    '105
0012 00004   0 02 00406              LDA    '406
0013 00005  -0 01 00006              JMP*   *+1
0014 00006   004143                  OCT    4143
0015 00007   0 12 00105 BB           IRS    '105
0016 00010   140040                 CRA
0017 00011   0 04 00106              STA    '106
0018 00012  -0 10 00014              JST*   *+2
0019 00013  -0 01 00015              JMP*   *+2
0020 00014   003065                 OCT    3065
0021 00015   004575                 OCT    4575
0022 00016   101002         CC      SS4
0023 00017   0 01 00022              JMP    *+3
0024 00020   14 0002                OCP    '2
0025 00021   0 12 00106              IRS    '106
0026 00022  -0 10 00024              JST*   *+2
0027 00023  -0 01 00025              JMP*   *+2
0028 00024   003047                 OCT    3047
0029 00025   004212                 OCT    4212
0030 00026   0 12 00105 DD           IRS    '105
0031 00027   140040                 CRA
0032 00030   0 04 00106              STA    '106
0033 00031  -0 01 00032              JMP*   *+1
0034 00032   005245                 OCT    5245
0035                            *
0036                                 ABS
0037                                 ORG    '4142
0038 04142   0 01 00000              JMP    AA
0039                                 ORG    '4211
0040 04211   0 01 00016              JMP    CC
0041                                 ORG    '4574
0042 04574   0 01 00007              JMP    BB
0043                                 ORG    '5244
0044 05244   0 01 00026              JMP    DD
0045                                 END

AA      000000     BB      000007     CC      000016     DD      000026

0000 WARNING OR ERROR FLAGS
DAP-16 MOD 2     REV. C     01-26-71
```

AC

Table A4.5 Subroutine GRAPH

```
      SUBROUTINE GRAPH(XN,A)
C-----Subroutine Graph for use with Tektronix Graphics Routines
C     This is a steering routine ... N and C-array elements
C     must be defined at the BASIC level.Note that C(0) in
C     BASIC corresponds to A(1) in this routine.
C
      DIMENSION A(8)
      IX=IFIX(XN+.2)
      GOTO(10,15,20,30,40,50),IX
10    CALL INITT(0)
      RETURN
15    CALL VWINDO(A(2),A(3),A(4),A(5))
      RETURN
20    IF=IFIX(A(1)+.1)
      GOTO (1,2,3,4,5,6),IF
    1 CALL MOVEA(A(6),A(7))
      RETURN
    2 CALL POINTA(A(6),A(7))
      RETURN
    3 CALL DRAWA(A(6),A(7))
      RETURN
    4 CALL MOVER(A(6),A(7))
      RETURN
    5 CALL POINTR(A(6),A(7))
      RETURN
    6 CALL DRAWR(A(6),A(7))
      RETURN
30    IC=IFIX(A(8)+.2)
      CALL VCURSR(IC,A(6),A(7))
      A(8)=FLOAT(IC)
      RETURN
40    IC=IFIX(A(8)+.2)
      CALL ANCHO(IC)
      RETURN
50    IX=A(6)
      IY=A(7)
      CALL FINITT(IX,IY)
      RETURN
      END
```

Table A4.6 Construction of the Graphics Package

Loader COMMON base is set to '37777.

(See also Table A6.2 of Appendix 6). G1,G2,G3,G4,G6,G7 are groups of library routines and addresses are in octal.

	P	A	B	Subroutines
1.	16000	20000 (or 20006)	20720	G1
2.	16003	21000	21725	G2
3.	16003	22000	22720	G3
4.	16003	23000	23724	G4
5.	16003	24000	24750	G5
6.	16003	25000	25720	G6
7.	160003	26000	26730	G7 MATHS

Table A4.7 Memory Map of the Graphics Package

*LOW	05470	EXP	06563	VECMOD	22000	D£11	26534
*BASE	24761	SQRT	06677	V2ST	22052	F£ER	26604
*HIGH	26756	SIN	06757	PNTMOD	22260	F£HT	26616
*BASE	23755	ATAN	07065	MODCHK	22316	ERCL	26653
*COMN	37505	GRAPH	20006	REL2AB	22344	F£AT	26654
*NAMES	11506	INITT	20410	REVCOT	23610	ARG£	26736
*BASE	26774	FINITT	20560	TKDASH	23000	SAVE	37505
*START	20005	VWINDO	20574	XYCNVT	22424	TKTRNX	37675
*BASE	21761	MOVEA	20634	CLIPT	24000		
*BASE	25751	POINTA	20654	DRAWA	25000		
*BASE	22747	MOVER	21000	LVLCHT	25044		
*BASE	20763	POINTR	21026	IOWAIT	25100		
ABS	05470	DRAWR	21054	TINPUT	25144		
L£22	05554	MOVABS	21102	TOUTPT	25162		
H£22	05560	VCURSR	21122	WINCOT	25200		
N£22	05576	DCURSR	21150	PCLIPT	25316		
S£22	05753	ANCHO	21246	PARCLT	25376		
A£22	05760	ANMODE	21302	C£21	26460		
D£22	06035	NEWLIN	21324	MOD	26466		
M£11	06163	CARTN	21332	FLOAT	26514		
M£22	06204	LINEF	21372	INT	26524		
C£12	06247	NEWPAG	21416	IDINT	26524		
E£22	06370	SVSTAT	21462	IFIX	26524		
ALOG	06477	RESTAT	21560	D£11X	26534		

Table A4.8 Special FORTRAN routines: FHT, FER

```
                                                          PAGE    1
0001                          *
0002                          * Modified FORTRAN error routines F£ER,F£HT
0003                          * Executive .. Patch ERCL of Hadios Executi
0004                          * in location ERCL in this routine..
0005                          *
0006                               SUBR  F£HT,HALT
0007                               SUBR  F£ER,ERR
0008                               ENT   ERCL
0009                               REL
0010 00000   0 000000    ERR  DAC   **
0011 00001  -0 02 00000        LDA*  ERR
0012 00002   0 04 00044        STA   TEMP
0013 00003  -0 02 00044        LDA*  TEMP
0014 00004   0 04 00011        STA   ERR1+1
0015 00005   0 02 00045        LDA   FT
0016 00006   0 04 00046        STA   OP
0017 00007   0 10 00022        JST   CONT
0018 00010  -0 10 00047  ERR1  JST*  ERCL
0019 00011   000000            OCT   0
0020                          *
0021 00012   0 000000    HALT DAC   **
0022 00013  -0 02 00012        LDA*  HALT
0023 00014   0 04 00044        STA   TEMP
0024 00015  -0 02 00044        LDA*  TEMP
0025 00016   0 04 00046        STA   OP
0026 00017   0 10 00022        JST   CONT
0027 00020  -0 10 00047        JST*  ERCL
0028 00021   144324            BCI   1,HT
0029                          *
0030 00022   0 000000    CONT DAC   **
0031 00023   34 0104           SKS   '104
0032 00024   0 01 00023        JMP   *-1
0033 00025   14 0104           OCP   '104
0034 00026   0 02 00043        LDA   CRLF
0035 00027   0 10 00033        JST   OUT
0036 00030   0 02 00046        LDA   OP
0037 00031   0 10 00033        JST   OUT
0038 00032  -0 01 00022        JMP*  CONT
0039                          *
0040 00033   0 000000    OUT  DAC   **
0041 00034   0416 70          ALR   8
0042 00035   74 0004          OTA   4
0043 00036   0 01 00035        JMP   *-1
0044 00037   0416 70          ALR   8
0045 00040   74 0004          OTA   4
0046 00041   0 01 00040        JMP   *-1
0047 00042  -0 01 00033        JMP*  OUT
0048 00043   106612      CRLF OCT   106612
0049 00044   000000      TEMP OCT   0
0050 00045   143324      FT   OCT   143324
0051 00046   000000      OP   OCT   0
0052 00047   0 000000    ERCL DAC   **
0053                           FIN
0054                           END

CONT   000022    CRLF   000043    ERCL   000047    ERR    000000
ERR1   000010    FT     000045    HALT   000012    OP     000046
```

Table A4.9 Modified FORTRAN FHT, FER for use with M6800 interrupt response code

```
                                                     PAGE      1
                              *
                              *
                              * Modified FORTRAN F≠ER,F≠ET routines for
                              * M6800 interrupt response code in HADIOS
                              * Executive ... ERRM = 1 is first byte sent
                              * to micro .. followed by MCOD = 2
                              *
                                   SUBR  F≠HT,HALT
                                   SUBR  F≠ER,ERR
                                   EXT   ERRM
                                   EXT   MCOD
                              *
                                   REL
00000    0 000000             ERR  DAC   **
00001    0 02 00021                LDA   =1
00002    0 04 00000                STA   ERRM
00003    0 02 00020                LDA   =2
00004    0 04 00000                STA   MCOD
00005    0 12 00000                IRS   ERR
00006   -0 01 00000                JMP*  ERR
                              *
00007    0 000000             HALT DAC   **
00010    000201                    IAB
00011    0 02 00021                LDA   =1
00012    0 04 00000                STA   ERRM
00013    0 02 00020                LDA   =2
00014    0 04 00000                STA   MCOD
00015    000201                    IAB
00016    0 12 00007                IRS   HALT
00017   -0 01 00007                JMP*  HALT
00020    000002                    FIN
00021    000001

                                   END

ERR    000000    ERRM   000000E   HALT   000007    MCOD   000000E

0000 WARNING OR ERROR FLAGS
DAP-16 MOD 2    REV. C    01-26-71
AC
```

Table A4.10 BASIC MTH-PAK Pointers

```
                                                                    PAGE
0001                        *
0002                        * BASIC Maths Package routines
0003                        * C‡21 is a special case
0004                        *
0005                             ENT    ABS,ABSF
0006                             ENT    L‡22,L22
0007                             ENT    H‡22,H22
0008                             ENT    N‡22,N22
0009                             ENT    S‡22,S22
0010                             ENT    A‡22,A22
0011                             ENT    D‡22,D22
0012                             ENT    M‡11,M11
0013                             ENT    M‡22,M22
0014                             ENT    C‡12,C12
0015                             ENT    C‡21
0016                             ENT    E‡22,E22
0017                             ENT    ALOG,LOGF
0018                             ENT    EXP,EXPF
0019                             ENT    SQRT,SQRF
0020                             ENT    SIN,SINF
0021                             ENT    ATAN,ATNF
0022                        *
0023          005470        ABSF EQU    '5470
0024          005554        L22  EQU    '5554
0025          005560        H22  EQU    '5560
0026          005576        N22  EQU    '5576
0027          005753        S22  EQU    '5753
0028          005760        A22  EQU    '5760
0029          006035        D22  EQU    '6035
0030          006163        M11  EQU    '6163
0031          006204        M22  EQU    '6204
0032          006247        C12  EQU    '6247
0033          006370        E22  EQU    '6370
0034          006477        LOGF EQU    '6477
0035          006563        EXPF EQU    '6563
0036          006677        SQRF EQU    '6677
0037          006757        SINF EQU    '6757
0038          007065        ATNF EQU    '7065
0039          006260        C21  EQU    '6260
0040                        *
0041                             REL
0042 00000     0 000000     C‡21 DAC    **
0043 00001     0 10 06260        JST    C21
0044 00002     0 01 00004        JMP    ERR
0045 00003    -0 01 00000        JMP*   C‡21
0046 00004    -0 10 00551  ERR   JST*   '551
0047 00005     151311             BCI    1,RI
0048                             END

A22     005760A   ABSF    005470A   ATNF    007065A   C‡21    000000
C12     006247A   C21     006260A   D22     006035A   E22     006370
ERR     000004    EXPF    006563A   H22     005560A   L22     005554
LOGF    006477A   M11     006163A   M22     006204A   N22     005576
S22     005753A   SINF    006757A   SQRF    006677A

0000 WARNING OR ERROR FLAGS
DAP-16 MOD 2     REV. C     01-26-71
```

AC

Table A4.11 Utility subroutine SU10

```
                                *                                PAGE        1
0001                            *
0002                            *Subroutine 10 --- Timer and PUNCH Tape lead
0003                            *BASIC statement --- CALL(10,X1,X2,X3)
0004                            *Following are BASIC MATHPAK Pointers
0005                            *
0006          000675      C£12  EQU   '675
0007          000654      H£22  EQU   '654
0008          000670      D£22  EQU   '670
0009                            ENT   SU10
0010                            *
0011                            REL
0012                            SETB  BAS9
0013 00000    0 000000    SU10  DAC   **
0014 00001    0 10 00036        JST   ARG              Get the first
0015 00002    100040            SZE                    argument
0016 00003    0 01 00012        JMP   CLK
0017                            *
0018 00004   -0 10 00043 TAPE  JST*  LEAD             Output Leader
0019 00005    0 12 00000        IRS   SU10
0020 00006    0 12 00000        IRS   SU10
0021 00007    0 12 00000        IRS   SU10
0022 00010    0 12 00000        IRS   SU10
0023 00011   -0 01 00000        JMP*  SU10             Return
0024                            *
0025 00012    0 12 00000  CLK   IRS   SU10             Get second
0026 00013    0 10 00036        JST   ARG                argument
0027 00014    100040            SZE
0028 00015    0 01 00022        JMP   STOP
0029 00016    0 02 00046  MULA  LDA   =-32768          Start CLOCK
0030 00017    0 04 00061        STA   '61
0031 00020    14 0020           OCP   '20
0032 00021    0 01 00006        JMP   TAPE+2
0033                            *
0034 00022    0 12 00000  STOP  IRS   SU10             Get address
0035 00023    0 10 00036        JST   ARG                of X3..
0036 00024    0 15 00034        STX   X3
0037 00025    0 02 00061        LDA   '61
0038 00026    0 07 00046        SUB   =-32768          Time elapsed
0039 00027    14 0220           OCP   '220               in 20 ms.
0040 00030   -0 10 00675        JST*  C£12             Convert to
0041 00031   -0 10 00670        JST*  D£22             REAL in sec.
0042 00032    0 000044          DAC   F50
0043 00033   -0 10 00654        JST*  H£22
0044 00034    0 000000    X3    DAC   **
0045 00035    0 01 00007        JMP   TAPE+3           Return
0046                            *
0047 00036    0 000000    ARG   DAC   **
0048 00037   -0 02 00000        LDA*  SU10
0049 00040    0 04 00000        STA   0
0050 00041   -0 02 00000        LDA*  0
0051 00042   -0 01 00036        JMP*  ARG
0052                            *
0053 00043    0 005432    LEAD  DAC   '5432
0054 00044    041544      F50   DEC   50.0
     00045    000000
0055 00046    100000            FIN
                                *
0056          000047      BAS9  EQU   *
0057                            END
```

Table A4.12 H316 BASIC Benchmark program

```
10  DIM A(100),B(100),C(100),D(100)
20  INPUT N,V
25  X1=1:X2=0
30  CALL (10,X1,X2,X3)
40  FOR J=0,N
50  FOR I=0,100:A(I)=V:B(I)=V:C(I)=V:D(I)=V: NEXT I
60  FOR I=0,100
70  IF I<50 THEN 85
80  A(I)=V:B(I)=V:C(I)=V:D(I)=V: NEXT I
82  GOTO 87
85  GOSUB 100: NEXT I
87  REM
88  FOR I=0,100: GOSUB 200: NEXT I
90  REM
92  FOR I=0,100:C(I)=A(I)*B(I):D(I)=C(I)*D(I)
93  D(I)=(D(I)*10/1023+23.303)
94  NEXT I
95  NEXT J
96  X2=1
97  CALL (10,X1,X2,X3)
98  PRINT "TIME=";X3
99  END
100  A(I)=V+V:B(I)=V+V:C(I)=V+V:D(I)=V+V: RETURN
200  RETURN
```

```
10  DIM A(100),B(100),C(100),D(100)
20  INPUT N,V
25  X3=0
30  X1=1:X2=0: CALL (10,X1,X2,X3)
40  CALL (9,A(0),B(0),C(0),D(0),N,V)
50  X2=1: CALL (10,X1,X2,X3)
60  PRINT "TIME TAKEN =";X3
100  END
```

Table A4.13 H316 FORTRAN Benchmark program

```
      SUBROUTINE FTIME(A,B,C,D,RN,VALUE)
      DIMENSION A(1),B(1),C(1),D(1)
C------Benchmark program for H316
C------
      N=IFIX(RN)
      DO 10 J=1,N
      DO 20 I=1,101
      IF ( I .LT. 50 ) GOTO 100
      A(I)=VALUE
      B(I)=VALUE
      C(I)=VALUE
      D(I)=VALUE
      GOTO 20
  100 A(I)=VALUE+VALUE
      B(I)=VALUE+VALUE
      C(I)=VALUE+VALUE
      D(I)=VALUE+VALUE
   20 CONTINUE
C-----
      DO 30 I=1,101
      GOTO 200
  201 CONTINUE
   30 CONTINUE
C-----
      DO 40 I=1,101
      C(I)=A(I)*B(I)
      D(I)=C(I)*B(I)
      D(I)=( D(I)*10./1023. + 23.303 )
   40 CONTINUE
   10 CONTINUE
      RETURN
  200 GOTO 201
      END
```

Table A4.14 SD BASIC Benchmark program

```
      PROGRAM ORIGIN :4000
      DATA ORIGIN    :4500
      DIM TIME/0/,F1/0/,F2/:80/,I,J,N,A(100),B(100),C(100),D(100)
      DIM VALUE
      REM
      REM - PROGRAM ENDS WITH A SWI .. TIME ELAPSED IN
      REM   VARIABLE 'TIME' IN LOCATION :4500-:4505
      REM   CONVERT TO APPROPRIATE UNITS DEPENDING ON
      REM   TIMER RESOLUTION 'F1'..
      REM
      INPUT "INPUT N,F1,VALUE " N,F1,VALUE
      CALL CLOCK(F1)
      FOR J=0 TO N
      FOR I=0 TO 100 \ A(I)=VALUE \ B(I)=VALUE
                       C(I)=VALUE \ D(I)=VALUE
      NEXT I
      FOR I=0 TO 100
      IF I<50 THEN GOSUB 100 \ NEXT I
      ELSE A(I)=VALUE \ B(I)=VALUE \ C(I)=VALUE \ D(I)=VALUE
      NEXT I
      REM
      FOR I=0 TO 100
      GOSUB 200
      NEXT I
      FOR I=0 TO 100
      C(I)=A(I)*B(I)
      D(I)=C(I)*B(I)
      D(I)=( D(I)*10/1023 + 23.303 )
      NEXT I
      NEXT J
      CALL CLOCK(F2)
  100 A(I)=VALUE+VALUE \ B(I)=VALUE+VALUE
      C(I)=VALUE+VALUE \ D(I)=VALUE+VALUE
      RETURN
  200 RETURN
      END
```

Table A4.15 Subroutine CLOCK for use with SD BASIC benchmark program

```
CLOCK   SEI
        LDAB FLAG
        BNE OFF
        LDX 4,X
        LDAA 5,X
        STAA FREQ
        LDX #IRQRES
        STX £F000
        LDX #£F530
        LDAB #£FF
        STAB 2,X
        LDAA #£3C
        STAA 3,X
        LDAA FREQ
        STAA 2,X
        NOP
        CLI
WAICLK  LDAA 3,X
        BMI OUTCLK
        BRA WAICLK
OUTCLK  LDAA 2,X
        LDAA #£3D
        STAA 3,X
        INC FLAG
        RTS
*
OFF     LDX 4,X
        LDAA 5,X
        LDX #£F530
        STAA 2,X
        NOP
        CLI
        SWI
*
IRQRES  LDX :+TIME+4
        INX
        BMI SWI
        STX :+TIME+4
        LDX #£F530
        LDAA 2,X
        RTI
SWI     SWI
FLAG    FCB 0
FREQ    RMB 1
        END
```

Table A4.16 Construction of the HADIOS Executive Rev. 03

1. Clear the memory.

2. Load the SLST of LDR-APM Rev. E (constructed in Appendix 3).

3. Set P = '16000, A = '27000 and load the object code of the HADIOS Executive Addresses are in actal.

4.

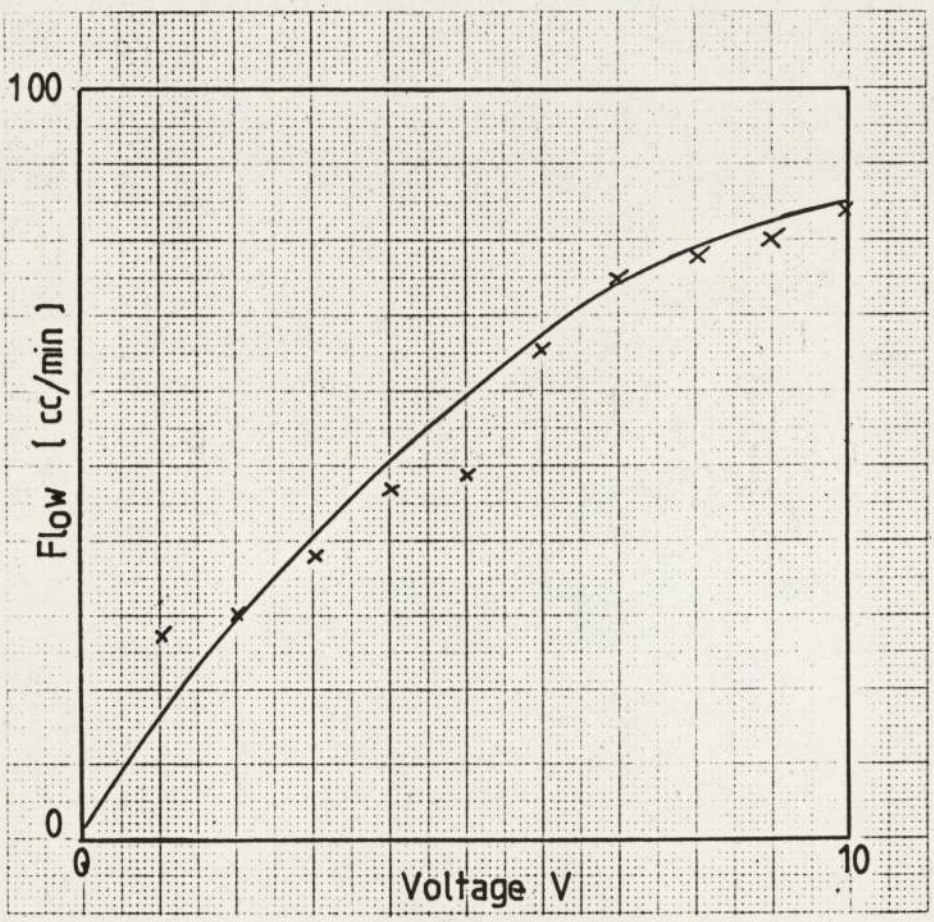
P	A	B	Subroutine
16003	34000	34760	M$22
			D$22
			C$12
			C$21
			S$22
			A$22
			ARG$
16003	33450	33760	N$22
			REAL
			L$22
			H$22
			F$AT
			F$ER - Modified
			F$HT - Modified

5. The utility PAL-AP is then used to punch out an SLST of the Executive occupying location '27000 to '34762.

Table A4.17 Memory map of the HADIOS Executive Rev. 03

*LOW	00063	SUB3	27652	REAL	33462
*START	00063	ICT1	30000	L$22	33462
*HIGH	34764	ABUF	30434	H$22	33472
*NAMES	12443	CB2	30637	F$AT	33510
*CORE	37777	M0	31017	F$ER	33572
*BASE	34762	ERRM	31147	F$HT	33601
*BASE	33445	MCOD	31150	M$22X	34000
*BASE	32773	CALM	31151	M$22	34000
*BASE	31774	CA2	31160	D$22X	34161
*BASE	30750	MBUF	31301	D$22	34161
*BASE	27772	CA22	32031	C$12	34420
*HADIOS	27000	C123	32240	C$21	34452
SKSI	27021	ERCL	32603	S$22	34504
IFLG	27265	SUB4	33070	A$22	34512
TABL	27317	TIME	33266	ARG$	34734
SUB1	27331	MCTR	33312		
SUB2	27601	CTRS	33325		
CAL0	27632	N$22	33450		

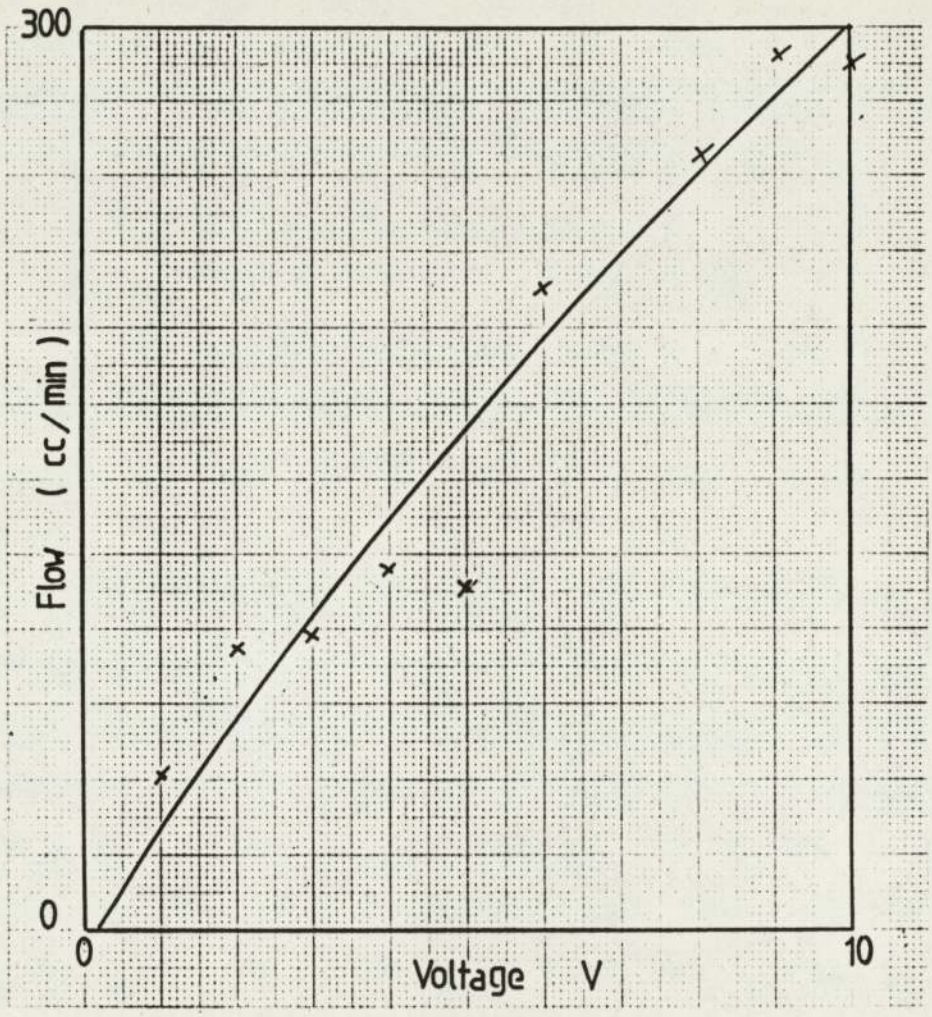
Figure A5.1 Calibration of Reflux valve, A1



$$Q = -.00319 + .01414V - .000586V^2 + .106 \times 10^{-4}V^3$$

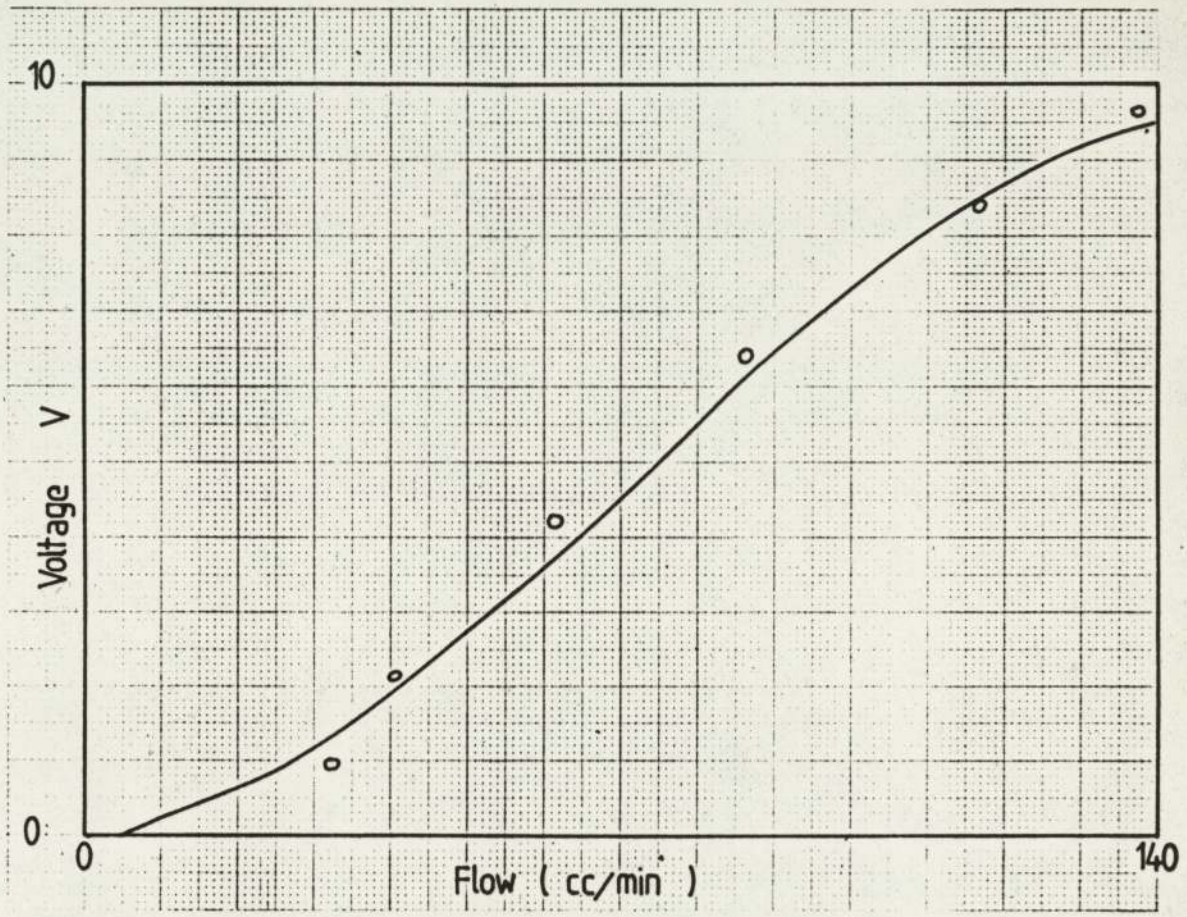
Note: Valve and Level calibrations are done using water at room temperature.

Figure A5.2 Calibration of Bottom product valve, A2



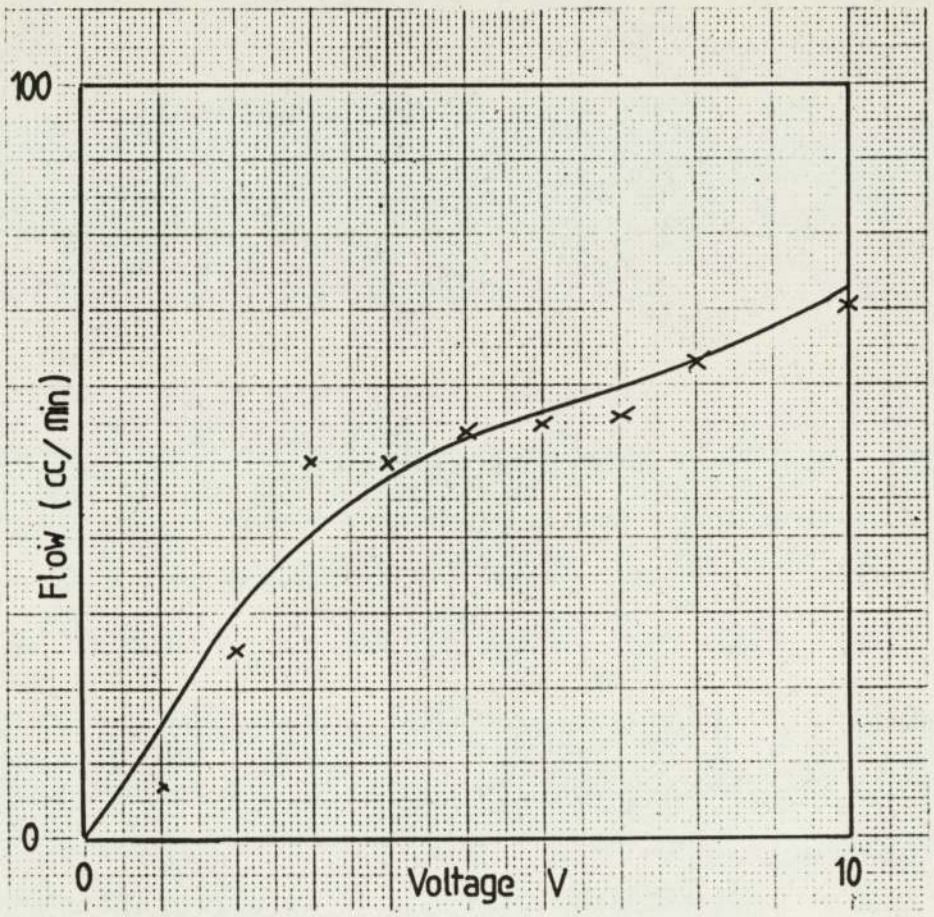
$$Q = 37.26 + 30.04V + .9524V^2 - .1548V^3$$

Figure A5.3 Calibration of Feed Valve, A3



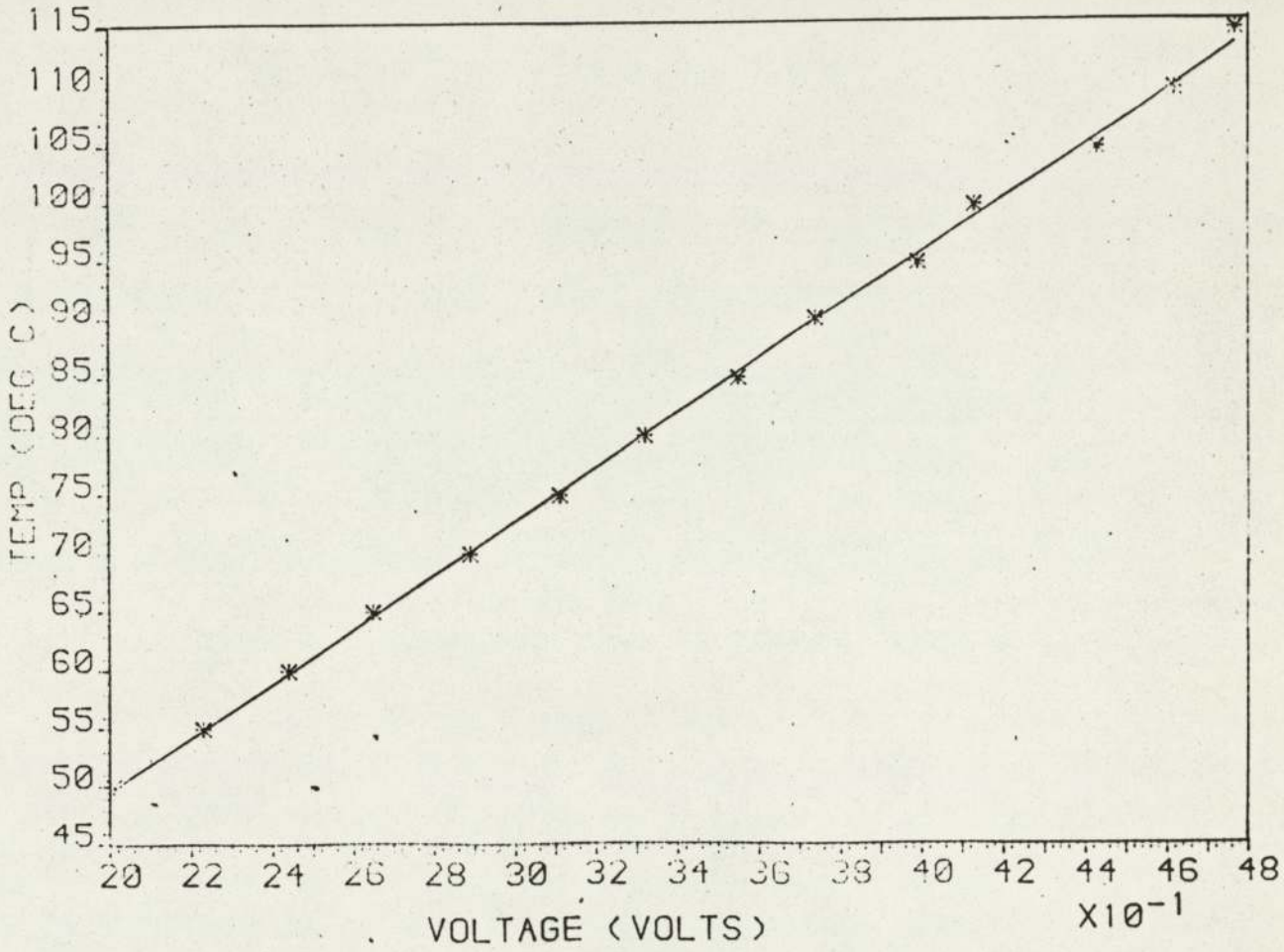
$$V = -.123 + 2.453Q + 9.195Q^2 - 4.266Q^3$$

Figure A5.4 Calibration of Distillate Valve, A5



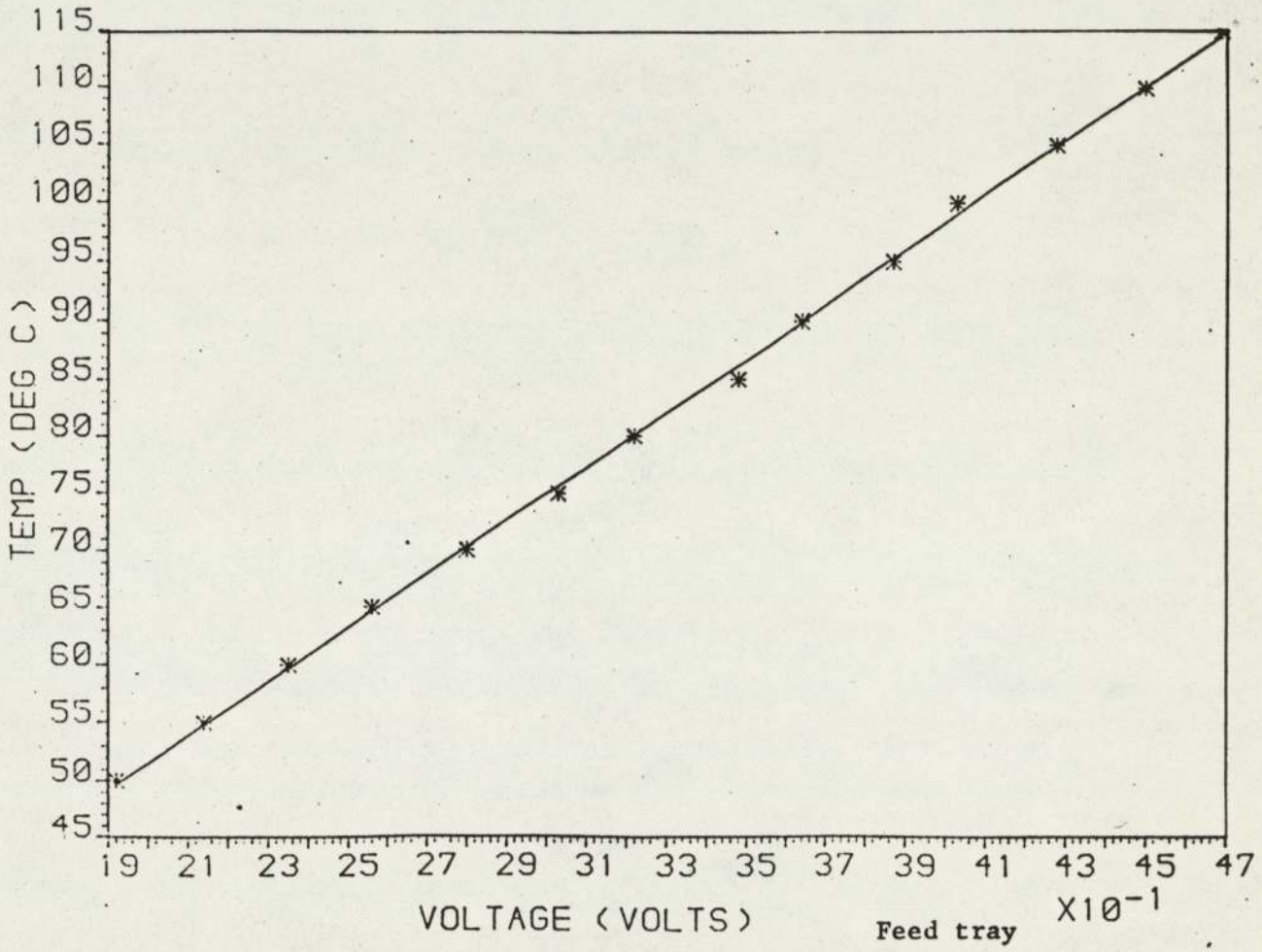
$$Q = .0365 + .0463V - .0075V^2 + .0004V^3$$

Figure A5.5 Calibration of Topmost tray thermocouple



$T = 23.181V + 3.2204$

Figure A5.6 Calibration of Feed tray thermocouple



$$T = 23.422V + 4.688$$

Figure A5.7 Calibration of Lowermost tray thermocouple

T = 23.181V + 3.2204

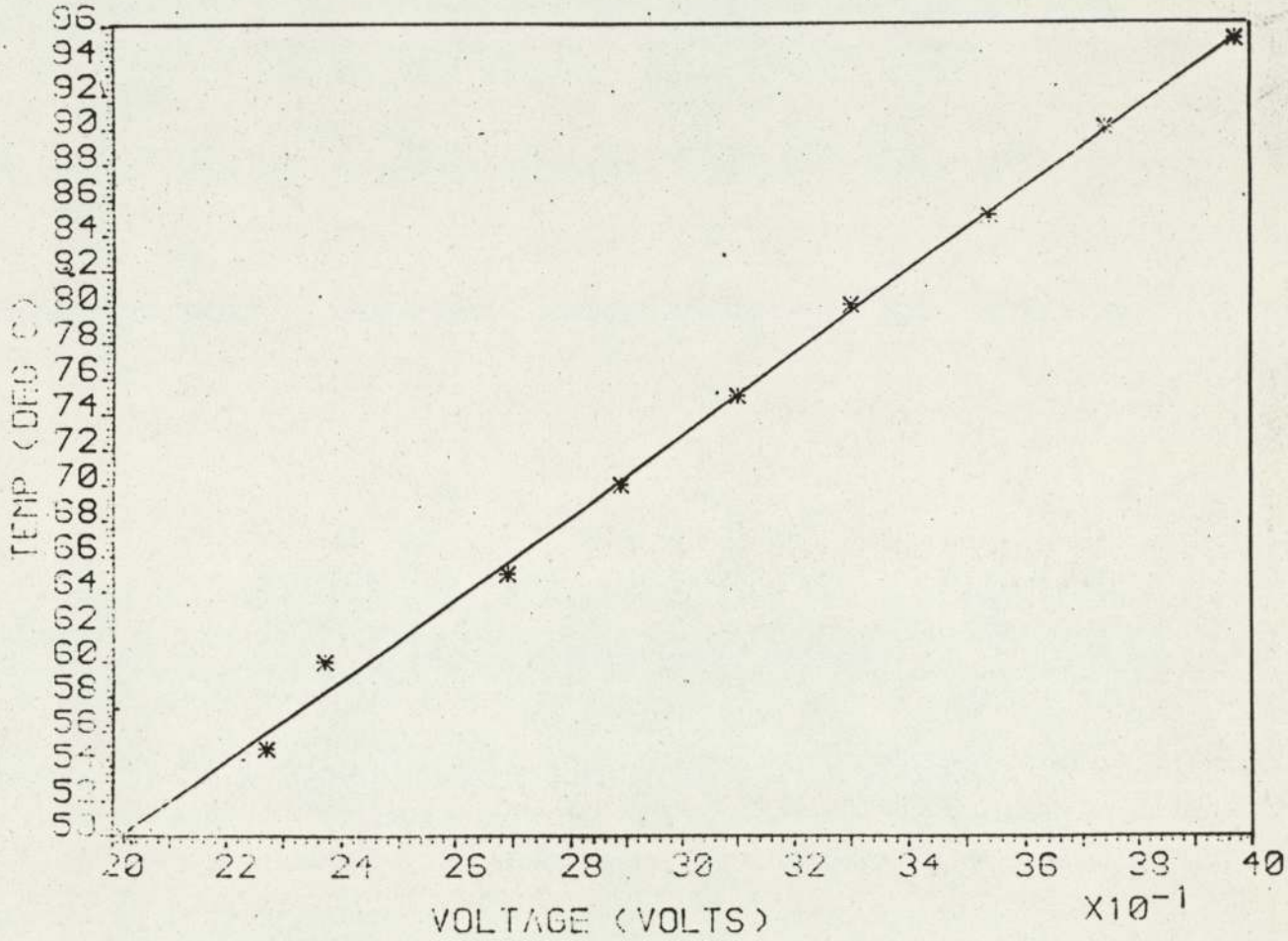
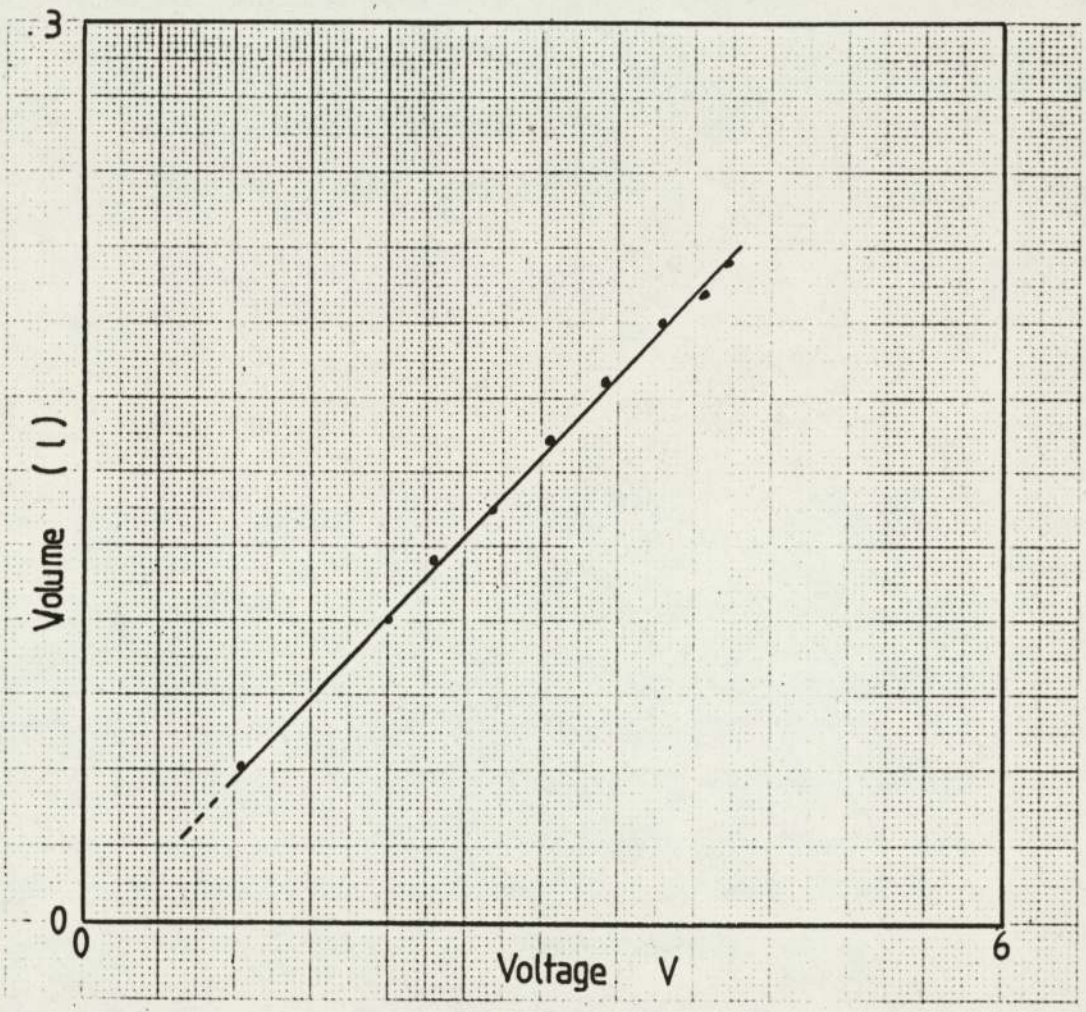


Figure A5.8 Characteristic of Reflux Stream thermocouple
T = 20.5V + 4.16

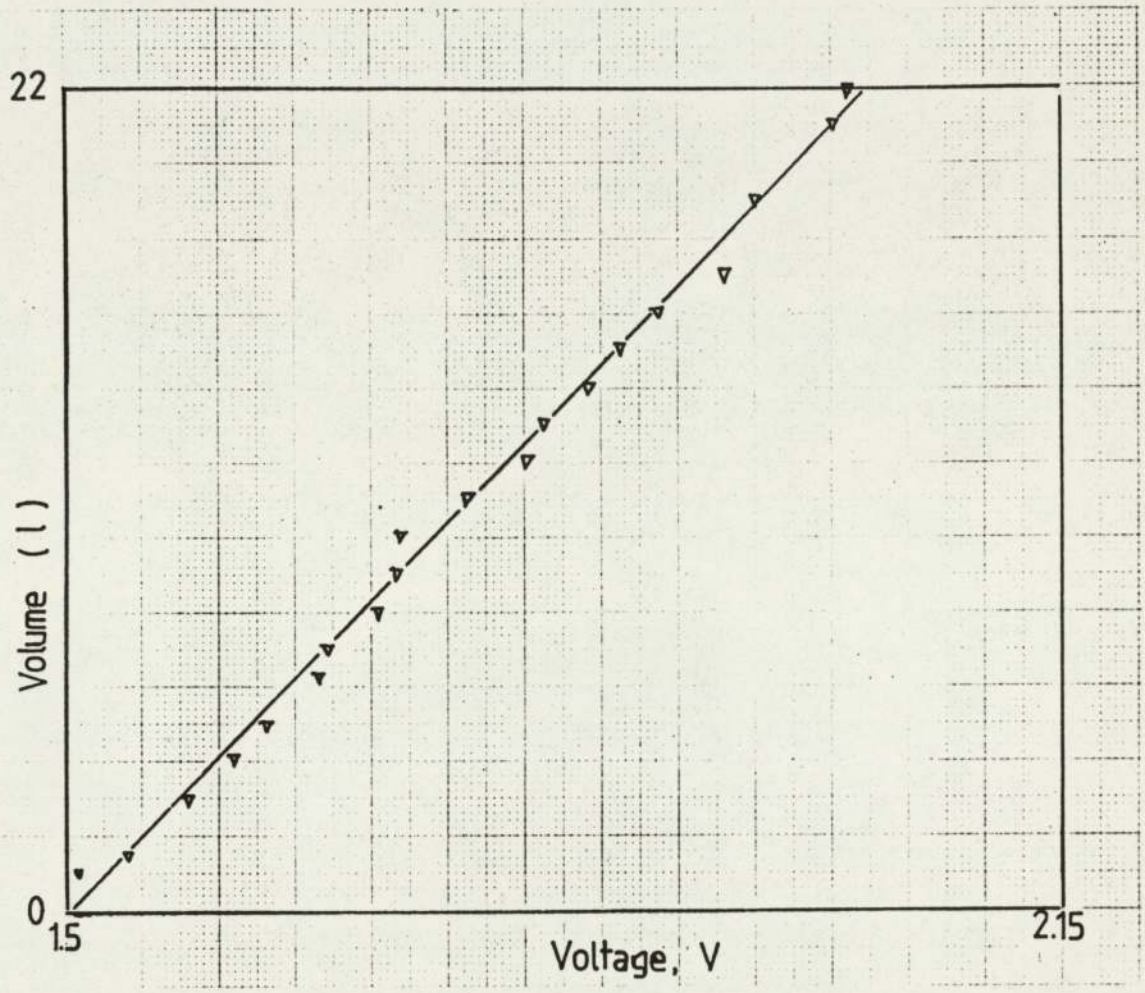
Figure A5.9 Characteristic of Feed Stream thermocouple
T = 22.1V - .307

Figure A5.10 Calibration of Reflux Drum Level



$$L_1 = -.05864 + .541V$$

Figure A5.11 Calibration of Reboiler Level



$$L_2 = .60.0 + 40.1V$$

Table A6.1 Construction of the Disk-Overlay Utility Program

The three main units of the program are subroutines DTOC, RWSC and A$DO3. The first two are listed here for the sake of completeness. Note that for more reliable disk operation, the instruction ANA CMD in A$DO3 has been changed to ERA CMD. Also, the RWSC routine has been made interrupt inhibited by the use of INH and ENB instructions. This means the H316 cannot be interrupted while it is reading a sector from disk storage - a situation which is considered desirable in an on-line environment since the detailed operation of the disk unit is not fully understood.

```
      SUBROUTINE DTOC(UNIT,TRAK,SECTOR,FADDR,LADDR,ERR)
C
C-----This subroutine,in conjunction with RWSC,overlays
C     program segment on disk into core ..
C
      REAL LADDR
      IUNIT=UNIT
      ITRAK=TRAK
      ISECT=SECTOR
      IBUF1=FADDR
      IBUF2=LADDR
   70 K=IBUF1-IBUF2+1
      IF(K.GT.64) K=64
C
C-----Call the Read a sector routine
C     If unsuccessful after 10 searches then call A$DO3
C     to turn off motor by  Rezero seek and
C     outputting a Type 0 command with bit 8 set..
C     Repeat another 10 times..
C
      DO 30 I=1,20
      CALL RWSC(IUNIT,ITRAK,ISECT,IBUF1,K,IER)
      IF(IER.EQ.0) GOTO 50
      IF(I.EQ.10)  CALL A$DO3(IUNIT)
   30 CONTINUE
   50 IF(K.LT.64.OR.IER.NE.0) GOTO 90
      ISECT=ISECT+8
      IF(ISECT.LE.26) GOTO 80
      ISECT=ISECT-26
      IF(ISECT.EQ.1) I=I+1
   80 IBUF1=IBUF1+64
      GOTO 70
   90 ERR=IER
      CALL A$DO3(IUNIT)
      RETURN
      END
```

Memory map of DTOC

*LOW	05470	RWSC	30222
*START	27777	A≠D03	30336
*HIGH	30460	L≠33	30354
*NAMES	12636	C≠21	30370
*COMN	37777	F≠AT	30376
*BASE	30464		
ABS	05470		
L≠22	05554		
H≠22	05560		
N≠22	05576		
S≠22	05753		
A≠22	05760		
D≠22	06035		
M≠11	06163		
M≠22	06204		
C≠12	06247		
E≠22	06370		
ALOG	06477		
EXP	06563		
SQRT	06677		
SIN	06757		
ATAN	07065		
DTOC	30000		

```
                                        SUBR  RWSC                    PAGE
0001                                    SUBR  RWSC
0002                                    REL
0003          000130        OPMD      EQU   '130
0004          001030        DSKO      EQU   '1030
0005          000230        OPCD      EQU   '230
0006          000530        DEMK      EQU   '530
0007          000030        IPMD      EQU   '030
0008          000130        DFMK      EQU   '130
0009          000530        IPDA      EQU   '530
0010          001030        DSKI      EQU   '1030
0011          000330        OPDA      EQU   '330
0012          000430        STAT      EQU   '430
0013 00000    0 00 00000    COUT      ***
0014 00001    077777        STT       OCT   77777
0015 00002    0 00 00000    DERR      ***
0016 00003    001000        SCCD      OCT   '1000
0017 00004    0 00 00000    TEMP      ***
0018 00005    0 00 00000    SBUF      ***
0019 00006    0 000000      RWSC      DAC   **
0020 00007    0 10 00000              CALL  F±AT
0021 00010    000006                  DEC   6
0022 00011    0 000000      UNIT      DAC   **
0023 00012    0 000000      TRAK      DAC   **
0024 00013    0 000000      SECT      DAC   **
0025 00014    0 000000      BUFF      DAC   **
0026 00015    0 000000      SECL      DAC   **
0027 00016    0 000000      ERR       DAC   **
0028 00017    001001                  INH
0029 00020    14 0130                 OCP   OPMD
0030 00021   -0 02 00015              LDA*  SECL
0031 00022    140407                  TCA
0032 00023    0 04 00000              STA   COUT
0033 00024   -0 02 00014 L            LDA*  BUFF
0034 00025    0 04 00005 S0           STA   SBUF
0035 00026    0 02 00120              LDA   ='37777
0036 00027    0 07 00005              SUB   SBUF
0037 00030    140407                  TCA
0038 00031    0 04 00005              STA   SBUF
0039 00032    140040                  CRA
0040 00033    0 04 00002              STA   DERR
0041 00034   -0 02 00011 S1           LDA*  UNIT
0042 00035    0406 76                 ARR   2
0043 00036   -0 06 00013              ADD*  SECT
0044 00037    0 06 00003              ADD   SCCD
0045 00040    74 1030                 OTA   DSKO
0046 00041    0 01 00040              JMP   *-1
0047 00042    14 0230                 OCP   OPCD
0048 00043   -0 02 00011 S2           LDA*  UNIT
0049 00044    0406 76                 ARR   2
0050 00045   -0 06 00012              ADD*  TRAK
0051 00046    0 04 00004              STA   TEMP
0052 00047    0 02 00117              LDA   =1
0053 00050    0406 73                 ARR   5
0054 00051    0 06 00004              ADD   TEMP
0055 00052    74 1030                 OTA   DSKO
0056 00053    0 01 00052              JMP   *-1
0057 00054    14 0230                 OCP   OPCD
```

```
                                        SUBR  RWSC                PAGE        2
0058 00055    0 35 00005          LDX   SBUF
0059 00056    14 0030             OCP   IPMD
0060 00057    140040              CRA
0061 00060    34 0130       S4    SKS   DFMK
0062 00061    0 01 00077          JMP   DEST
0063 00062    14 0530             OCP   IPDA
0064 00063    54 1030             INA   DSKI
0065 00064    0 01 00063          JMP   *-1
0066 00065   -0 04 00001          STA*  STT
0067 00066    140040              CRA
0068 00067    0 12 00000    S6    IRS   0
0069 00070    101000              NOP
0070 00071    0 12 00000          IRS   COUT
0071 00072    0 01 00060          JMP   S4
0072 00073    34 0530             SKS   DEMK
0073 00074    0 01 00073          JMP   *-1
0074 00075    14 0030             OCP   IPMD
0075 00076    0 01 00103          JMP   D2
0076 00077    34 0530       DEST  SKS   DEMK
0077 00100    0 01 00060          JMP   S4
0078 00101    0 02 00116    D1    LDA   ='100
0079 00102    0 04 00002          STA   DERR
0080 00103    14 0030       D2    OCP   IPMD
0081 00104    14 0430             OCP   STAT
0082 00105    54 1030             INA   DSKI
0083 00106    0 01 00105          JMP   *-1
0084 00107    0 03 00115          ANA   ='37000
0085 00110    0 06 00002          ADD   DERR
0086 00111    0404 72             LGR   6
0087 00112   -0 04 00016          STA*  ERR
0088 00113    000401              ENB
0089 00114   -0 01 00006          JMP*  RWSC
0090 00115    037000              END
     00116    000100
     00117    000001
     00120    037777

BUFF    000014    COUT    000000    D1      000101    D2      000103
DEMK    000530A   DERR    000002    DEST    000077    DFMK    000130A
DSKI    001030A   DSKO    001030A   ERR     000016    IPDA    000530A
IPMD    000030A   L       000024    OPCD    000230A   OPDA    000330A
OPMD    000130A   RWSC    000006    S0      000025    S1      000034
S2      000043    S4      000060    S6      000067    SBUF    000005
SCCD    000003    SECL    000015    SECT    000013    STAT    000430A
STT     000001    TEMP    000004    TRAK    000012    UNIT    000011

0000 WARNING OR ERROR FLAGS
DAP-16 MOD 2    REV. C    01-26-71
```

AC

```
                                        SUBR   A≠DO3,ADO3          PAGE          1
0001                                    SUBR   A≠DO3,ADO3
0002                                    REL
0003 00000    0 000000    ADO3  DAC    **
0004 00001    0 10 00000        CALL   F≠AT
0005 00002    000001            DEC    1
0006 00003    0 000000    LD    DAC    **
0007 00004   -0 02 00003        LDA*   LD
0008 00005    0406 76           ARR    2
0009 00006    0 05 00014        ERA    CMD
0010 00007    14 0130           OCP    OPMD
0011 00010    74 1030           OTA    DSKO
0012 00011    0 01 00010        JMP    *-1
0013 00012    14 0230           OCP    OPCD
0014 00013   -0 01 00000        JMP*   ADO3
0015          000130      OPMD  EQU    '130
0016          001030      DSKO  EQU    '1030
0017 00014    034400      CMD   VFD    2,0,3,7,2,0,1,1,1,0,7,0
0018          000230      OPCD  EQU    '230
0019                              END

ADO3   000000     CMD    000014     DSKO   001030A   LD     000003
OPCD   000230A    OPMD   000130A

0000 WARNING OR ERROR FLAGS
DAP-16 MOD 2    REV. C    01-26-71
AC
```

Table A6.2 Construction of EKF1 - Segment 1

Note:

P, A, B are the settings (in octal) of the P, A and B registers respectively. The COMMON base is changed to '37504 before the first loading. Due to an error in the loader program, subroutines cannot be force-loaded (P = '16004) with a new origin and base address (for the intersector links) as is possible with a loader program break (P = '16003). This difficulty is solved by following a modified procedure denoted as FORCE.

	P	A	B	Subroutine
1.	16000 16004	20000	20750	KOMMON INIT
2.	16003	21000	21740	INIT1
3.	16003	22000	22750	INIT2
4.	16003	23000	23750	INIT3
5.	FORCE	24000	24750	P1OF1O
6.	16003	25000	25750	TRANS
7.	16003	26000	26740	MATMUL
8.	FORCE -	27000	27750	COMN MATADD
9.	FORCE	30470	30760	FDTX
10.	16003 -	31000	31750	SDBUB DIAADD
11.	FORCE 16004 16004	32000	32750	KALMA1 SDINT MATTPS MATHS
13.	FORCE 16004	33000	33120	INTPAS INTJS

Memory map for Segment 1

*LOW	05470
*START	17777
*HIGH	33110
*NAMES	11547
*COMN	33143
*BASE	30762
*BASE	33121
*BASE	32762
*BASE	31765
*BASE	27760
*BASE	26746
*BASE	25761
*BASE	24760
*BASE	23757
*BASE	22761
*BASE	21760
*BASE	20763
ABS	05470
L‡22	05554
H‡22	05560
N‡22	05576
S‡22	05753
A‡22	05760
D‡22	06035
M‡11	06163
M‡22	06204
C‡12	06247
L‡22	06370
ALOG	06477
EXP	06563
SQRT	06677
SIN	06757
ATAN	07065
KOMMON	20000
INIT	20006
INIT1	21000
INIT2	22000
INIT3	23000
P10F10	24000
TRANS	25000
MATMUL	26000
MATTPX	26500
COMN	27000
MATADD	27570
SDBUB	31000
DIAADD	31504
KALMA1	32000
SDINT	32240
MATTPS	32416
L‡33	32510
IFIX	32524
INT	32524
IDINT	32524
C‡21	32534
F‡AT	32542
INTPAS	33000
INTJS	33052
PRIN	33143
DATA	33217
HOUSE	33257
BUB	33301
CONVEC	33305
INTEG	33361
INVERS	33365
WORK	33671
COLUMN	34021
MODEL	34427
FILTER	34757
KOLPAR	37477

Table A6.3 Construction of EKF1 - Segment 2

	P	A	B	Subroutine
1.	16000 16004	20000	20745	KOMMON DCOL
2.	16003	21000	21760	DCOL1
3.	16003	22000	22760	DCOL2
4.	16003	23000	23760	DCOL3
5.	16003 - 16004	24000 (if required)	24760	DCOL5 DCOL6 SWAVE
6.	FORCE -	25000 (if required)	25760	PERTUB RAMP STEP
7.	16003	26000	26760	DCOL4
8.	16003 16004	31000	31750	SDBUB DIAADD
9.	FORCE - 16004 -	32000	32750	KALMA1 SDINT MATTPS MATHS
10.	16003 -	33000	33120	INTPAS INTJS

Memory map of Segment 2

*LOW	05470
*START	17777
*HIGH	33110
*NAMES	11514
*COMN	33143
*BASE	33121
*BASE	32762
*BASE	31765
*BASE	26771
*BASE	25770
*BASE	24773
*BASE	23774
*BASE	22775
*BASE	21774
*BASE	20772
ABS	05470
L#22	05554
H#22	05560
N#22	05576
S#22	05753
A#22	05760
D#22	06035
M#11	06163
M#22	06204
C#12	06247
E#22	06370
ALOG	06477
EXP	06563
SQRT	06677
SIN	06757
ATAN	07065
KOMMON	20000
DCOL	20006
DCOL1	21000
DCOL2	22000
DCOL3	23000
DCOL5	24000
DCOL6	24444
PERTUB	25000
RAMP	25337
STEP	25476
DCOL4	26000
SDBUB	31000
DIAADD	31504
KALMA1	32000
SDINT	32240
MATTPS	32416
L#33	32510
IFIX	32524
INT	32524
IDINT	32524
C#21	32534
F#AT	32542
INTPAS	33000
INTJS	33052
PRIN	33143
DATA	33217
HOUSE	33257
BUB	33301
CONVEC	33305
INTEG	33361
INVERS	33365
WORK	33671
COLUMN	34021
MODEL	34427
FILTER	34757
KOLPAR	37477

Table A6.4 Construction of EKF1 - Segment 3

	P	A	B	Subroutine
1.	16000	20000	20750	KOMMON SIMUL
2.	16003 16004	21000	21762	SIMULX VECTOR
3.	FORCE	22000	22760	KALMAN
4.	16003	23000	23760	MATINV
5.	FORCE	24000	24760	P1OF1O MATMUL
6.	16003	25000	25740	TRANS
7.	16003	26000	26760	PKK DIAMUL DIASUB MATTPX
8.	16003	30470	30760	FDTX
9.	FORCE	27000	27750	COMN MATADD
10.	16003	3100	31750	SDBUB DIAADD
11.	16003	32000	32750	KALMA1 SDINT MATTPS MATHS
12.	16003	33000	33120	

Memory map of Segment 3

*LOW	05470	M#11	06163	SDBUB	31000
*START	17777	M#22	06204	DIAADD	31504
*HIGH	33110	E#22	06370	KALMA1	32000
*NAMES	11420	ALOG	06477	SDINT	32340
*COMN	33143	EXP	06563	MATTPS	32416
*BASE	27760	SQRT	06677	L#33	32510
*BASE	31765	SIN	06757	IFIX	32524
*BASE	24761	ATAN	07065	INT	32534
*BASE	23771	KOMMON	20000	IDINT	32524
*BASE	32762	SIMUL	20006	C#21	32534
*BASE	33121	SIMULX	21000	F#AT	32542
*BASE	25761	VECTOR	21636	INTPAS	33000
*BASE	26763	KALMAN	22000	INTJS	33052
*BASE	30762	MATINV	23000	PRIN	33143
*BASE	22773	P10F10	24000	DATA	33217
*BASE	21775	MATMUL	24172	HOUSE	33257
*BASE	20774	TRANS	25000	BUB	33301
ABS	05470	PKK	26000	CONVEC	33305
L#22	05554	DIAMUL	26124	INTEG	33361
H#22	05560	DIASUB	26310	INVERS	33365
N#22	05576	MATVEC	26440	WORK	33671
S#22	05753	MATTPX	26556	COLUMN	34021
A#22	05760	COMMON	27000	MODEL	34427
D#22	06035	MATADD	27570	FILTER	34757
				KOLPAR	37477

Table A6.5 Construction of EKF1 - Segment 4

This segment is identical to the Graphics package constructed in Table A4.6 of Appendix 4. Note that in EKF1, subroutine GRAPH must be loaded at location '20006.

Table A6.6 Construction of EKF1 - Segment 5

	P	A	B	Subroutine
1.	16000	20000	20700	KOMMON
	16004			KOMN
2.	FORCE	32000	32750	KALMA1
	16004			SDINT
	16004			MATTPS
	-	(L$33 force-loaded)		MATHS

Memory map of Segment 5

*LOW	05470	KOMMON	20000
*START	17777	KOMN	20006
*HIGH	33110	KALMA1	32000
*NAMES	12446	SDINT	32240
*COMN	33143	MATTPS	32416
*BASE	30762	L$33	32510
*BASE	33121	IFIX	32524
*BASE	20707	INT	32524
ABS	05470	IDINT	32524
L$22	05554	C$21	32534
H$22	05560	F$AT	32542
N$22	05576	INTPAS	33000
S$22	05753	INTJS	33052
A$22	05760	PRIN	33143
D$22	06035	DATA	33217
M$11	06163	HOUSE	33257
M$22	06204	BUB	33301
C$12	06247	CONVEC	33305
E$22	06370	INTEG	33361
ALOG	06477	INVERS	33365
EXP	06563	WORK	33671
SQRT	06677	COLUMN	34021
SIN	06757	MODEL	34427
ATAN	07065	FILTER	34757
		KOLPAR	37477

Table A6.7 BASIC program to calculate Antoine constants

```
10  REM  --- PROGRAM TO CALCULATE TIME-INVARIANT ANTOINE COEFFICIENTS
20  DIM U(3),V(3),S(3),T(3),L(4)
30  FOR I=1 TO 3: READ U(I),V(I),S(I),T(I),L(I),M(I),N(I): NEXT I
40  DATA 40,40,11.9,40.1,.175,1.07,.98
50  DATA 100,400,31.4,100,.348,1.05,.98
60  DATA 760,760,86.7,120.8,.487,1.05,.98
70  L(4)=.434:M(4)=1.05:N(4)=.98
80  N0=2
90  REM   -----
100  FOR I=1 TO N0: GOSUB 1000: NEXT I
110  GOSUB 1500
112  PRINT "VA=";C1;: PRINT "VB=";C2
120  PRINT "A(1)=";A(1);: PRINT "B(1)=";B(1);: PRINT "C(1)=";C(1)
125  PRINT "A(2)=";A(2): PRINT "B(2)=";B(2): PRINT "C(2)";C(2)
130  END
132  RETURN
1000  IF I>1 THEN 1200
1010  P1=(LOG(U(1)))/2.303:X1=S(1)
1020  P2=(LOG(U(2)))/2.303:X2=S(2)
1030  P3=(LOG(U(3)))/2.303:X3=S(3): GOTO 1250
1040  REM   ------
1200  P1=(LOG(V(1)))/2.303:X1=T(1)
1210  P2=(LOG(V(2)))/2.303:X2=T(2)
1220  P3=(LOG(V(3)))/2.303:X3=T(3)
1250  X4=(X2-X3)*(P1-P2)-(X2-X1)*(P3-P2)
1260  X5=(X2-X3)*(P1-P2)*X1+(X2-X3)*(P1-P2)*X2
1270  X5=X5-(X2-X1)*(P3-P2)*X3-(X2-X1)*(P3-P2)*X2
1280  X6=(X2-X3)*(P1-P2)*X1*X2-(X2-X1)*(P3-P2)*X2*X3
1290  C(I)=((-X5)+SQR(X5^2-4*X4*X6))/(2*X4)
1300  B(I)=(P1-P2)*(X1+C(I))*(X2+C(I))/(X1-X2)
1310  A(I)=P1+B(I)/(X1+C(I))
1320  RETURN
1500  P=0:Q=0
1510  FOR J=1 TO 4
1520  GOSUB 1600: GOSUB 1700
1530  P=P+P0:Q=Q+Q0
1540  NEXT J
1550  C1=P/4:C2=Q/4: RETURN
1600  X1=L(I):X2=1-X1
1610  P0=(X1-X2)*LOG(N(I))/((X1^2)*2.303)+2*LOG(M(I))/(X2*2.303)
1620  RETURN
1700  X1=L(I):X2=1-X1
1710  Q0=(X2-X1)*LOG(M(I))/((X2^2)*2.303)+2*LOG(N(I))/(2.303*X1)
1720  RETURN
```

Table A6.8 BASIC program to perform a Mcabe-Thiele analysis of the Distillation Column

```
10  DIM E(11),X(22,2),Y(22,2),Z(11,2),T(11),M(11)
20  DIM P(2),G(2),A(70),B(20),C(20),H(16),K(10),F(16)
25  DIM Q(2),O(6),S(6),L(11),U(110),W(20)
30  REM  --- ENTER EQUILIBRIUM COEFFICIENTS NOW
35  READ C1,C2,A(1),A(2),B(1),B(2),C(1),C(2)
40  DATA .42E-02,-.4E-03,7.4266,8.08374
45  DATA 1549.3,2128.93,254.082,288.34
46  REM  --- INPUT FEED,FEED COMP.,BOTTOM COMP.,TOP COMP.
47  REM  --- EFFICIENCY FLAG AND REFLUX RATIO GUESS..
48  PRINT "INPUT F,XF,XB,XD,EFFLAG,RG"
50  INPUT F2,F1,B1,D1,E0,R1
55  B2=.2:B3=.6E-02:V8=31.5074:V9=34.777:D3=.10089E05
57  B2=.25
60  D4=8686.75:R2=.381E-01:F7=7:N=10:G9=.1
65  T9=1*760
68  Z1=.762E-01:Z2=.105E-01
70  IF E0=0 THEN 85
80  E1=.94: GOTO 200
85  Z1=.762E-01:Z2=.105E-01:Z3=.3E-02:Z4=.435E-01:Z5=.2335E-01
90  Z6=.381E-01:Z7=0:Z8=0:Z9=0
95  F(1)=359.9:F(2)=391.8:F(3)=560.94:F(4)=579.8:F(5)=45.445
100  F(6)=40.72:F(7)=.265:F(8)=.268:F(9)=.286E-03:F(10)=.344E-03
105  F(11)=.295E-01:F(12)=.321E-01:F(13)=.345:F(14)=.436
110  F(15)=131.5:F(16)=166
200  J=1:N9=N+1: GOSUB 1000: REM          ----CALL SUBROUTINE SDWEG: PRI
210  GOSUB 3000:M(N9)=60
212  PRINT
215  PRINT "-------------------------------------------------------"
220  PRINT "FEED COMPOSITION=";: PRINT F1;
230  PRINT TAB(30);"FEED RATE=";: PRINT F2;: PRINT "MOL/HR"
240  PRINT
250  PRINT "DISTILLATE COMP=";: PRINT D1;
260  PRINT TAB(30);"DISTILLATE=";: PRINT D: PRINT
270  PRINT "BOTTOMS COMP=";: PRINT B1;
280  PRINT TAB(30);"BOTTOMS=";: PRINT B0: PRINT
282  PRINT "VTOP=";: PRINT V4;
284  PRINT TAB(30);"VBOT=";: PRINT V5: PRINT
286  PRINT "REFLUX R=";: PRINT (V4-D)/D;
288  PRINT TAB(30);"HEAT Q=";: PRINT Q9: PRINT
290  PRINT "-------------------------------------------------------"
300  PRINT
310  PRINT "    I                L(I)        M(I)              T(I)"
320  PRINT
330  FOR I=1 TO 11: PRINT I,L(I),M(I),T(I): NEXT I
340  PRINT : PRINT
350  PRINT "    I                XS          YS                EFF"
360  PRINT : FOR I=1 TO 11
370  PRINT I,X(I,J),Z(I,J),E(I)
380  NEXT I: PRINT "SET SS 4 NOW AND INPUT 1 OR 0 ";: INPUT W9
385  IF W9=0 THEN  STOP
390  FOR I=1,11: PRINT X(I,J);: PRINT ",";: PRINT Z(I,J);
392  PRINT ",";: PRINT M(I): NEXT I
394  FOR I=1,11: PRINT L(I);: PRINT ",";: PRINT T(I);
396  PRINT ",";: PRINT E(I): NEXT I
```

```
2255  REM              ----
2260  L(I)=R0+F2
2270  V=L(I)-B0
2280  I2=I-1:Y(I,J)=(L(I)*X(I2,J)-B0*B1)/V
2290  GOSUB 4000:T(I)=T8:X(I,J)=X9
2300  IF E9=1 THEN  RETURN
2320  IF I<>N9 THEN 2350
2340  E(I)=1: GOTO 2360
2350  IF E0=1 THEN 2360
2355  GOSUB 7000
2360  NEXT I
2370  F9=X(N9,J)
2380  RETURN
2997  REM
2998  REM  ---------- MCCABE THIELE STEADY STATE MODULE ----------
2999  REM
3000  IF E0=0 THEN 3030
3010  FOR I3=1 TO N9: IF I3=N9 THEN 3020
3015  E(I3)=E1: GOTO 3025
3020  E(I3)=1
3025  NEXT I3
3030  D=(F1-B1)*F2/(D1-B1):B0=F2-D:R0=D*R
3040  V4=R0+D
3050  FOR I=1 TO N9: IF I>1 THEN 3130
3060  L(I)=R0:Y(I,J)=D1
3070  GOSUB 4000
3080  T(I)=T8:X(I,J)=X9
3090  IF E9=1 THEN  RETURN
3100  IF E0=1 THEN 3120
3110  GOSUB 7000
3120  GOTO 3300
3130  IF I>=F7 THEN 3210
3140  L(I)=R0:Y(I,J)=L(I)*X(I-1,J)/V4+D*D1/V4
3150  GOSUB 4000
3160  T(I)=T8:X(I,J)=X9
3170  IF E9=1 THEN  RETURN
3180  IF E0=1 THEN 3200
3190  GOSUB 7000
3200  GOTO 3300
3205  REM              ----
3210  L(I)=R0+F2
3220  V5=L(I)-B0
3230  Y(I,J)=(L(I)*X(I-1,J)-B0*B1)/V5
3240  GOSUB 4000:T(I)=T8:X(I,J)=X9
3250  IF E9=1 THEN  RETURN
3260  IF I<>N9 THEN 3290
3270  E(I)=1: GOTO 3300
3290  IF E0=1 THEN 3300
3295  GOSUB 7000
3300  NEXT I
3305  REM              -----
3310  Z(N9,J)=Y(N9,J)*E(N9)
3320  I=N
3330  I3=I+1
3340  Z(I,J)=E(I)*Y(I,J)+(1-E(I))*Z(I3,J)
3350  I=I-1
```

```
3360  IF I<1 THEN 3380
3370  GOTO 3330
3380  D1=Z(1,J)
3390  B1=(F2*F1-D*D1)/B0
3400  X(N9,J)=B1
3410  V6=L(N)-B0-V5
3420  V7=B1*V8+(1-B1)*V9
3430  Q9=V7*V5
3440  Q7=22/7:Q8=2*R2
3450  FOR I=1 TO N
3460  D2=X(I,J)*D3+(1-X(I,J))*D4
3470  D6=L(I)*10/(D2*Q7*Z2)
3480  D7=B2+B3*D6
3490  D8=(Q7*.1E05/4)*(Z1^2-2*Z2^2)*D7
3500  M(I)=D8*.1E-05*D2
3510  NEXT I
3515  L(11)=B0
3520  RETURN
3525  REM              -----
3997  REM
3998  REM  ---------- DEW POINT CALCULATIONS ----------
3999  REM
4000  E9=0
4010  IF Y(I,J)>1 THEN  PRINT "ERRORTB": GOTO 4040
4020  IF Y(I,J)<0 THEN  PRINT "ERNEG": GOTO 4040
4030  GOTO 4050
4040  E9=1: RETURN
4050  J2=J
4060  Y(I+11,J)=Y(I,J)
4070  F0=LOG(10):Y(I+11,2)=1-Y(I+11,1)
4080  FOR I1=1 TO 2:G(I1)=1: NEXT I1
4090  T0=80
4100  REM              ----
4105  S1=0:S2=0
4110  FOR J1=1 TO 2
4120  P(J1)=EXP(F0*(A(J1)-B(J1)/(T0+C(J1))))
4130  X(I+11,J1)=Y(I+11,J1)*T9/(G(J1)*P(J1))
4140  S1=S1+X(I+11,J1)
4150  S3=F0*B(J1)*X(I+11,J1)/(T0+C(J1)^2)
4160  S2=S2+S3: NEXT J1
4170  E7=1-S1: IF ABS(E7)<=.1E-02 THEN 4800
4190  REM              -----
4210  T0=T0-E7/S2
4220  IF ABS(1-X(I+11,1))<.1E-06 THEN 4500
4230  X(I+11,1)=X(I+11,1)/(X(I+11,1)+X(I+11,2))
4240  X(I+11,2)=X(I+11,2)/(X(I+11,1)+X(I+11,2))
4250  FOR J1=1 TO 2
4260  IF J1>1 THEN 4300
4270  S4=(2*C2-C1)*(1-X(I+11,J1))^2
4280  S5=2*(C1-C2)*(1-X(I+11,J1))^3
4290  G(J1)=EXP(F0*(S4+S5)): GOTO 4320
4295  REM              ----
4300  S4=2*(C1-C2)*(1-X(I+11,J1))^2
4310  S5=2*(C2-C1)*(1-X(I+11,J1))^3
4315  G(J1)=EXP(F0*(S4+S5))
4320  NEXT J1
4330  GOTO 4510
4500  G(1)=1:G(2)=1
```

```
4510  GOTO 4105
4800  T8=T0
4810  FOR J1=1 TO 2
4820  P(J1)=EXP(F0*(A(J1)-B(J1)/(T8+C(J1))))
4830  Q(J1)=Y(I+11,J1)*T9/(G(J1)*P(J1))
4840  NEXT J1
4850  J=J2
4890  X9=Q(J)
4900  RETURN
5999  REM
6990  REM
6991  REM  ---------- PLATE EFFICIENCY CALCULATIONS ----------
6992  REM
7000  L1=L(I):V1=V
7005  DEF FNS(Z0)=1.85335-2*.7327*Z0-3*.32393*Z0^2+4*.16057*Z0^3
7010  T7=T(I)+273
7015  Z8=1:Z9=1
7020  H1=Z1:H2=Z2:H3=Z3:H4=Z4:H5=Z5:H6=Z6:H7=Z7:H8=Z8:H9=Z9
7030  REM              ----
7040  FOR J3=1 TO 2
7050  H(J3)=F(J3):H(J3+2)=F(J3+2):H(J3+4)=F(J3+4)
7055  H(J3+6)=F(J3+6):H(J3+8)=F(J3+8):H(J3+10)=F(J3+10)
7060  H(J3+12)=F(J3+12):K(J3+8)=F(J3+14)
7070  NEXT J3
7080  REM              ----
7090  FOR I7=1 TO 2:H(14+I7)=T7/H(2+I7)
7100  K(I7)=(K(8+I7)*H(4+I7)/H(2+I7))
7110  K(I7)=K(I7)*(.653E-01/(H(6+I7)^.773)-.9E-01*H(14+I7))
7120  K(4+I7)=.324*K(I7)^.5
7130  K(4+I7)=K(I7+4)*3.6
7140  K(I7)=K(I7)*.1E07/K(I7+8)
7150  K(2+I7)=Z8*.101325E06/(8.314*T7)
7160  F3=(1.9*H(14+I7))^(.9*LOG(1.9*H(I7+14))/2.303)
7170  F4=1.058*H(I7+14)^.645-.261/F3
7180  K(I7+6)=33.3*SQR(K(I7+8)*H(I7+2))*F4
7190  K(6+I7)=K(6+I7)/((H(8+I7)*.1E07)^(2/3))
7200  K(6+I7)=K(6+I7)*3.6*.1E-03
7210  NEXT I7
7220  REM              -----
7230  FOR I7=1 TO 2:H(10+I7)=H(10+I7)*.1E07: NEXT I7
7240  W1=1.18*(H(10+1)^(1/3))
7245  W2=1.18*(H(10+2)^(1/3))
7250  W0=(W1+W2)/2
7255  W3=1.15*H(1):W4=1.15*H(2):W5=SQR(W3+W4):W6=T7/W5
7260  W7=1.16145/(W6^.14874)+.52487/EXP(.7732*W6)
7265  W7=W7+2.16178/EXP(2.437*W6):W8=SQR(1/K(9)+1/K(10))
7270  W9=1-2.46*W8*.1E-03
7280  U0=W8*W9*(T7^(3/2)):U1=3600*.1E-03*(U0/(Z8*W7*W0^2))
7380  U2=7.4*.1E-07*K(9)^.5/(H(11)^.6)
7390  U3=7.4*.1E-07*K(10)^.5/(H(12)^.6)
7400  U4=1
7410  IF X(I,J)<0 THEN U4=0
7420  U5=U4*(X(I,J)*U3+(1-X(I,J))*U2)
7430  U6=(X(I,J)*K(5)^(1/3)+(1-X(I,J))*K(6)^(1/3))^3
```

```
7435  U6=U6/3.6:U7=U5*T7/U6
7440  GOSUB 8500: REM              ----CALL CONVRT
7450  M5=22/7:M6=M5*(H1^2-2*H2^2)/4
7455  Y4=X(I,J)*K(1)+(1-X(I,J))*K(2)
7460  Y5=Y(I,J)*K(3)+(1-Y(I,J))*K(4)
7470  Y6=(X(I,J)*K(5)^(1/3)+(1-X(I,J))*K(6)^(1/3))^3
7480  Y7=(Y(I,J)*K(7)*SQR(K(9))+(1-Y(I,J))*K(8)*SQR(K(10)))
7485  Y7=Y7/(Y(I,J)*SQR(K(9))+(1-Y(I,J))*SQR(K(10)))
7490  Y8=Y7/(Y5*U1)
7495  Y9=1.25*(.124E-01+.171E-01*M7+.25E-02*Y1/H4+.15E-01*H3)^2
7500  A1=M7*SQR(Y5): IF A1>2.5 THEN 7520
7505  IF A1<.5 THEN 7530
7508  GOSUB 8000
7510  GOTO 7600
7520  B9=.6: GOTO 7600
7530  B9=1
7600  A2=B9*Y3*M6/(12*M9*3600)
7610  A3=H5^2/(Y9*A2)
7620  A4=SQR(1.065*.1E05*U7)*(.26*M7*SQR(Y5)+.15)
7625  A5=A4*A2*3600
7630  A6=(.77+.116*H3-.29*M7*SQR(Y5)+.217E-01*M8)/(SQR(Y8))
7633  X0=X(I,J)
7635  A7=FNS(X0)
7640  A8=A7*V1/L1:A9=A5*A6/(A5+A8*A6): GOTO 7650
7645  A9=A6
7650  C3=1-EXP(-A9)
7655  C4=(A3/2)*(SQR(1+4*A8*C3/A3)-1)
7660  C5=C4+A3:C6=(EXP(C4)-1)/(C4*(1+C4/C5))
7665  IF C5>10 THEN 7700
7670  C7=1/EXP(-C5): GOTO 7710
7700  C7=0
7710  C8=(1-C7)/(C5*(1+C5/C4))
7720  C9=C8+C6
7730  E(I)=C9*C3
7740  IF E(I)>=.99 THEN E(I)=.99
7750  RETURN
7997  REM
7998  REM  ------------ INTERPOLATION MODULE -------------
7999  REM
8000  FOR I9=1 TO 6: READ O(I9),S(I9): NEXT I9
8010  DATA 0,1,.5,.726,1,.642,1.5,.6,2,.5789,2.5,.56
8020  N3=6
8030  IF (A1-O(1)),8100,8100,8070
8070  IF (A1-O(N3)),8120,8080,8080
8080  B9=S(N3)
8090  RETURN
8100  B9=S(1): RETURN
8120  FOR I9=2 TO N3
8130  IF (A1<O(I9)) THEN 8150
8140  NEXT I9
8150  B9=S(I9-1)+(A1-O(I9-1))*(S(I9)-S(I9-1))/(O(I9)-O(I9-1))
8160  RETURN
```

```
8497  REM
8498  REM  ------------ CONVERT TO COMPATIBLE UNITS -----------
8499  REM
8500  H1=H1/.3048:H2=H2/.3048:H3=H3*12/.3048
8510  H4=H4/.3048:H5=H5/.3048:H6=H6*12/.3048:H7=H7*12/.3048
8520  FOR I8=1 TO 2
8530  K(I8)=K(I8)*K(8+I8)*.1E-05*62.43
8540  K(2+I8)=K(2+I8)*K(8+I8)*.1E-05*62.43
8550  K(I8+4)=K(4+I8)*.672E-03/3.6
8560  K(I8+6)=K(I8+6)*.672E-03/3.6
8570  NEXT I8
8580  U1=U1/(.929E-01*3600):U7=U7*10/2.581
8590  M1=K(9)*Y(I,J)+(1-Y(I,J))*K(10)
8600  M2=K(9)*X(I,J)+(1-X(I,J))*K(10)
8610  M3=K(3)*Y(I,J)+(1-Y(I,J))*K(4)
8620  M4=K(1)*X(I,J)+(1-X(I,J))*K(2)
8630  M5=22/7:M6=(M5/4)*(H1^2-2*H2^2)
8640  M7=V1*(M1*.2205E-02)/(3600*M3*M6)
8650  M8=L1*(M2*.2205E-02)/(H4*60*M4*.13368)
8660  M9=L1*(M2*.2205E-02)/(3600*M4)
8670  Y1=L1*(M2*.2205E-02)/(M4*60*.13368)
8680  Y2=.48*1.25*(Y1/H6)^(2/3)
8690  Y3=H3+Y2+H7
8700  RETURN
```

Sample Output from Program in Table A6.8

```
INPUT P,XF,XP,XD,FFLAG,NG
?10,.4,.09,.98,4,2.09

?SIMBG-2000
?SIMBG-2000
?SIMBG-2000
```

FEED COMPOSITION= .4 FEED RATE= 10 MOL/HR

DISTILLATE COMP= .97975 DISTILLATE= 3.48315

BOTTOMS COMP= .9013378-01 BOTTOMS= 6.51685

VIN= 10.7381 VBOT= 10.7381

REFLUX R= 2.08286 HEAT Q= 370.273

I	L(I)	V(I)	T(I)
1	7.25492	.119418	87.9771
2	7.25492	.109435	89.4365
3	7.25492	.107942	91.8049
4	7.25492	.107942	95.6959
5	7.25492	.104114	98.7879
6	7.25492	.102537	102.131
7	17.2549	.102698	103.573
8	17.2549	.101739	105.852
9	17.2549	.101568	109.016
10	17.2549	.9928671-01	112.655
11	6.51685	10	116.125

I	XS	YS	EFF
1	.943681	.97975	.99
2	.886137	.954973	.99
3	.783091	.911866	.99
4	.659689	.844121	.99
5	.534323	.748635	.99
6	.431816	.643435	.99
7	.340388	.53447	.92546
8	.328694	.501419	.904683
9	.244032	.454428	.878048
10	.163442	.383649	.851711
11	.9013378-01	.218013	1

SET SS & VOL AND INPUT 1 OR 0 ?1

JOB EXIT

Table A6.9 FORTRAN Subroutines for Model I and general BASIC Calls

INIT: CALL(2,C(1),D(1),E(1),F(1),E1,E2,E4,V1)

where
- C array contains the initial (steady state) profiles for Model I
- D array contains operating conditions
- E array contains the initial (steady state) profiles for Model II
- F is array containing assignments to P(0,0), Q and R
- E1,E2,E4 are steering flags
- V1 Vapour boil-up rate

COMN: CALL(4,C(1),D(1),E(1),RO,QO,E2,E3)

where
- C array contains Model I profiles
- D array contains process parameters
- E array contains Model II profiles (dynamic response or estimates)
- RO Reflux rate in Model I
- QO Heat duty in Model I
- E2,E3 are steering flags

DCOL: CALL(2(T1,T2,Z1)

where
- T1 is time
- T2 integration step length
- Z1 integration method 1 = Simple Euler, 2 = Modified Euler

PERTUB: CALL(7,T1,S(1),R(1),W(1),C1,F2)

where
- S array contains STEP disturbance parameters
- R array contains RAMP disturbance parameters
- W array contains SWAVE disturbance parameters
- F1 - index for type of disturbance
- F2 - index for variable to be disturbed

KOMN: CALL(2,C(1),L6,L7,L8,L9)

where the C array is returned containing the specified matrix (rows = L_6, column = L_7)

```
      SUBROUTINE KOMMON
      REAL LR,L0,LR1,L01,LAMDA1,LAMDA2
      COMMON/PRIN/DX(11),DM(11)
      COMMON/DATA/AREA,PAI,HC,THETA
     1,LAMDA1,LAMDA2,DEN1,DEN2,RAD,DDIAM,PP(2),PB(2),HHD(2)
      COMMON/HOUSE/VA,VB,A(2),B(2),C(2),F10
      COMMON/SUB/NPB,NCOMPB,TCPB,EPSIL
      COMMON/CONVEC/FKR(7),FKQ(15)
      COMMON/INTEG/JS4,JS,JE,ND
      COMMON/INVERS/AI(7,14)
      COMMON/WORK/Z(2),ZZ(2),PP(15),QQ(15),S1,S2,S3,S4,S5
     1,S6,S7,S8,S9,S10
      COMMON/COLUMN/SX(11,2),SY(11,2),SYADT(11,2),SSM(11),SL(11)
     1,ST(11),SV(11),EMV(11),L0,D0,F0,XF,XD,XR,FT,RT,LR,Q0
      COMMON/MODEL/SX1(11,2),SY1(11,2),SYADT1(11,2),SSM1(11)
     2,SL1(11),ST1(11),SV1,L01,D1,F1,XF1,XD1,XR1,LP1,Q1
      COMMON/FILTER/P(15,15),FKT(15,15),EX(11),FKK(7,15),FKK(15,7)
     3,ET2,ET4,ET6,ET8,ET10,EFR,ELR,EXF,EV
      COMMON/KOLPAR/NT,NPLATE,NCOMP,IPF,IOO
      IOO=1
      RETURN
      END
```

```
      SUBROUTINE INIT(C,D,E,F,E1,E2,E4,SVB)
C
C-----This subroutine in conjuction with INIT1,INIT2 and INIT3
C     are used in initialising variables..
      REAL LR,L0,L01,LR1
      COMMON/COLUMN/SX(11,2),SY(11,2),SYACT(11,2),SSM(11),SL(11
     1,ST(11),SV(11),EMV(11),L0,D0,F0,XF,XD,XB,FT,RT,LR,Q0
      COMMON/MODEL/SX1(11,2),SY1(11,2),SYACT1(11,2),SSM1(11)
     1,SL1(11),ST1(11),SV1,L01,D01,F1,XF1,XD1,XB1,LR1,Q1
      COMMON/KOLPAR/NT,NPLATE,NCOMP,IPF,ICC
      DIMENSION C(1),D(1),E(1),F(1)
C-----
      ITASK=E1
      IC=IFIX(D(1))
      NPLATE=IFIX(D(2))
      ICC=IC
      NT=NPLATE+1
      GOTO(5,10),ITASK
    5 CONTINUE
      DO 55 IP=1,NT
      SX(IP,IC)=C(IP)
      SY(IP,IC)=C(IP+11)
      SYACT(IP,IC)=C(IP+11)
      SSM(IP)=C(IP+22)
      SL(IP)=C(IP+33)
      ST(IP)=C(IP+44)
      EMV(IP)=C(IP+55)
   55 CONTINUE
      IF (E4.EQ.0.) RETURN
      DO 77 I=1,NT
      SX1(I,1)=E(I)
      SYACT1(I,1)=E(I+11)
      SY1(I,1)=E(I+11)
      SSM1(I)=E(I+22)
      SL1(I)=E(I+33)
      ST1(I)=E(I+44)
   77 CONTINUE
      SV1=SVB
      RETURN
C-----
   10 CONTINUE
      CALL INIT1(D,F,E2,E4)
      RETURN
      END
```

```
      SUBROUTINE INIT1(D,F,E2,E4)
C
C-----This subroutine in conjunction with INIT,INIT2,INIT3
C     are used for initialisation..
C
      COMMON/PRIN/DX(11),DM(11)
      COMMON/COLUMN/SX(11,2),SY(11,2),SYACT(11,2),SSM(11),SL(11)
     ,ST(11),SU(11),EMU(11),L0,D0,F0,XF,XD,XB,FT,RT,LR,Q0
      COMMON/KOLPAR/NT,NPLATE,NCOMP,IPF,ICC
      COMMON/BUB/NPB,NCOMPB,TCPB,EPSIL
      COMMON/INTEG/JS4,JS,JE,ND
      DIMENSION D(1),F(1)
C-----
      IC=ICC
      F0=D(3)
      XF=D(4)
      RT=D(5)
      FT=D(6)
      LR=D(7)
      IPF=IFIX(D(8))
      SSM(NT)=D(10)
      Q0=D(11)
      D0=D(12)
      L0=D(13)
      EPSIL=D(14)
C-----
      ND=2*NT
      CALL INIT2(LAMDA1,LAMDA2)
      XD=SYACT(1,IC)
      XB=SX(NT,IC)
      SX(NT,2)=1.-XB
      CALL SDBUB(NT,IC,SX(NT,IC),SY(NT,IC),ST(NT))
      SYACT(NT,IC)=SY(NT,IC)
      LAMDA=XB*LAMDA1+(1.-XB)*LAMDA2
      VTOP=Q0/LAMDA
C-----
      DO 49 IP=1,NT
      SU(IP)=VTOP
   49 CONTINUE
C-----
      IF(E2.EQ.0.) RETURN
      CALL INIT3(F,VTOP,E4)
      RETURN
      END
```

```
      SUBROUTINE INIT2(LAM1,LAM2)
C
C-----This subroutine in conjunction with INIT,INIT1 and INIT3
C     are used for initialisation ..
C
      REAL LAM1,LAM2,LAMDA1,LAMDA2
      INTEGER ECOUNT
C-----
      COMMON/PRIN/DX(11),DM(11),KFIL,KCOUNT,IE,IEC,ECOUNT
      COMMON/DATA/AREA,PAI,HC,THETA
     1,LAMDA1,LAMDA2,DEN1,DEN2,RAD,DDIAM,HA(2),HB(2),HHC(2)
      COMMON/KOLPAR/NT,NPLATE,NCOMP,IPF,ICC
      COMMON/HOUSE/VA,VB,A(2),B(2),C(2),F10
      COMMON/BUB/NPB,NCOMPB,TCPB
C-----
      HC=.25
      THETA=.006
      LAMDA1=31.507
      LAMDA2=34.777
      LAM1=LAMDA1
      LAM2=LAMDA2
      DEN1=10089.743
      DEN2=8686.75
      RAD=.0381
      XCDIAM=.0762
      DDIAM=.0105
      XDDIAM=DDIAM
      XTCP=1.
C-----
      KCOUNT=0
      NCOMP=2
      HA(1)=31.1766
      HA(2)=41.2685
      HB(1)=.19166
      HB(2)=.20319
      HHC(1)=.0001154
      HHC(2)=.000136
      PAI=3.1416
      AREA=.004387
      NPB=NPLATE
      NCOMPB=NCOMP
      TCPB=XTCP*760.
C-----
      VA=.0042
      VB=-.0004
      A(1)=7.4266
      A(2)=8.08374
      B(1)=1549.3
      B(2)=2128.93
      C(1)=254.082
      C(2)=288.34
      F10=ALOG(10.)
      RETURN
      END
```

```
      SUBROUTINE INIT3(F,VTOP,E4)
C
C-----This subroutine in conjunction with INIT,INIT1 and INIT2
C     are used for initialisation ..
C
      REAL LR,L0,LR1,L01
      COMMON/COLUMN/SX(11,2),SY(11,2),SYACT(11,2),SSM(11),SL(11)
     1,ST(11),SV(11),EMV(11),L0,D0,F0,XF,XD,XB,FT,RT,LR,Q0
      COMMON/MODEL/SX1(11,2),SY1(11,2),SYACT1(11,2),SSM1(11)
     2,SL1(11),ST1(11),SV1,L01,D1,F1,XF1,XD1,XB1,LR1,Q1
      COMMON/FILTER/P(15,15),FKT(15,15),EX(11),FKM(7,15),FKK(15,7)
     3,ET2,ET4,ET6,ET8,ET10,EFR,ELR,EXF,EV
      COMMON/CONVEC/FKR(7),FKQ(15)
      COMMON/KOLPAR/NT,NPLATE,NCOMP,IPF,ICC
      DIMENSION F(1)
C-----
      IC=ICC
      DO 33 I=1,15
      FKQ(I)=F(I)
   33 CONTINUE
C-----
      FKR(1)=F(16)
      FKR(2)=F(17)
      FKR(3)=F(18)
      FKR(4)=F(19)
      FKR(5)=F(20)
      FKR(6)=F(21)
      FKR(7)=F(22)
C-----
      DO 100 I=1,15
      DO  90 J=1,15
      P(I,J)=0.
      IF(I.EQ.J) P(I,J)=F(23)
   90 CONTINUE
  100 CONTINUE
      P(12,12)=F(24)
      P(13,13)=F(25)
      P(14,14)=F(26)
      P(15,15)=F(27)
C-----
      F1=F0
      XF1=XF
      XD1=XD
      XB1=XB
      LR1=LR
      D1=D0
      L01=L0
      Q1=Q0
      EFR=F1
      ELR=LR1
      EXF=XF1
      EV=VTOP
C-----
      IF(E4.NE.0.) GOTO 400
      DO 200 I=1,NT
      SX1(I,IC)=SX(I,IC)
      SY1(I,IC)=SY(I,IC)
      SYACT1(I,IC)=SY(I,IC)
```

```
      SSM1(I)=SSM(I)
      SL1(I)=SL(I)
      ST1(I)=ST(I)
  200 CONTINUE
      SV1=VTOP
C-----
  400 SX1(NT,2)=SX(NT,2)
      SL1(NT)=L0
      RETURN
      END



      SUBROUTINE KOMN(E,R,C,RN,RM)
C
C-----Subroutine used to retrieve COMMON arrays from BASIC ..
C
      COMMON/INVERS/A(7,14)
      COMMON/CONVEC/FKR(7),FKQ(15)
      COMMON/FILTER/P(15,15),FKT(15,15),EX(11),FKM(7,15)
     1,FKK(15,7),ET2,ET4,ET6,ET8,ET10,EFR,ELR,EXF,EV
      DIMENSION E(1)
C-----
      IR=R
      JC=C
      N=IFIX(RN)
      IF(N.EQ.0) GOTO 20
C-----
      DO 10 J=1,JC
      DO 9  I=1,IR
      K=I+(J-1)*IR
      GOTO(1,2,3,4,5),N
    1 E(K)=P(I,J)
      GOTO 9
    2 E(K)=FKT(I,J)
      GOTO 9
    3 E(K)=FKM(I,J)
      GOTO 9
    4 E(K)=FKK(I,J)
      GOTO 9
    5 E(K)=A(I,J)
    9 CONTINUE
   10 CONTINUE
      RETURN
C-----
   20 M=RM
      DO 23 I=1,IR
      GOTO(21,22),M
   21 E(I)=FKR(I)
      GOTO 23
   22 E(I)=FKQ(I)
   23 CONTINUE
      RETURN
      END
```

```
:
      SUBROUTINE DCOL(TIME,DT,RIOD)
C
C-----This subroutine in conjunction with DCOL1,DCOL2,DCOL3,
C     DCOL4,DCOL5,and DCOL6 describes the actual PROCESS dynamics model.
C     It is called from BASIC which supplies the time,integration
C     step interval,etc....
C
      REAL L,M,LR,L0
      COMMON/PRIN/DXT(11),DMT(11)
      COMMON/COLUMN/X(11,2),Y(11,2),YACT(11,2),M(11),L(11)
     1,T(11),V(11),EMV(11),L0,D,F,XF,XD,XB,FT,RT,LR,Q
      COMMON/INTEG/JS4,JS,JE,ND
      COMMON/KOLPAR/NT,NPLATE,NCOMP,IPF,ICC
      DIMENSION ENTHV(11,2),ENTHL(11,2),Z(24),DZ(24)
C-----
      IOD=RIOD
      IC=ICC
      JE=0
      JS=0
      IPASS=0
C-----
    9 CONTINUE
      XD=YACT(1,IC)
      L0=L(NPLATE)-V(NT)
      D=V(1)-LR
C-----
      IZ=1
      CALL DCOL1(IZ)
      DO 80 IP=1,ND
      IF(IP.GT.NT) GOTO 60
      Z(IP)=X(IP,IC)
      DZ(IP)=DXT(IP)
      GOTO 80
   60 J=IP-NT
      Z(IP)=M(J)
      DZ(IP)=DMT(J)
   80 CONTINUE
C-----
      STP=TIME
      DO 90 INTCAL=1,ND
      TIME=STP
      CALL INTJS(JS4,JS,JE,IOD,IPASS)
      CALL SDINT(Z(INTCAL),DZ(INTCAL),TIME,DT,IOD,INTCAL)
   90 CONTINUE
C-----
      CALL DCOL6(DXT,DMT,DZ,X,M,Z)
      DO 200 IP=1,NT
  200 CALL SDBUB(IP,IC,X(IP,IC),Y(IP,IC),T(IP))
C-----
      IZ=2
      CALL DCOL1(IZ)
C-----
      CALL DCOL2(ENTHV,ENTHL)
      CALL DCOL3(ENTHV,ENTHL)
C-----
      CALL DCOL4(L,V,F,LR)
      IPASS=IPASS+1
      CALL INTPAS(IPASS,IOD)
      IF(JS.NE.0) GOTO 9
      RETURN
```

```
      SUBROUTINE DCOL1(IZ)
C
C-----This subroutine in conjunction with DCOL,DCOL2,DCOL3,DCOL4,
C     DCOL5 and DCOL6 describes dynamic PROCESS model ..
C
      REAL L,M,L0,LR,LAMDA1,LAMDA2
      COMMON/PRIN/DXT(11),DMT(11)
      COMMON/COLUMN/X(11,2),Y(11,2),YACT(11,2),M(11),L(11)
     1,T(11),V(11),EMV(11),L0,D,F,XF,XD,XB,FT,RT,LR,Q
      COMMON/DATA/AREA,PAI,HC,THETA,LAMDA1,LAMDA2,DEN1
     1,DEN2,RAD,DDIAM,HA(2),HB(2),HHC(2)
      COMMON/KOLPAR/NT,NPLATE,NCOMP,IPF,IC
C-----
      GOTO(5,55),IZ
    5 CONTINUE
      DO 50 IP=1,NT
      IF(IP.EQ.1) IK=1
      IF(IP.GT.1.AND.IP.LT.IPF) IK=2
      IF(IP.EQ.IPF) IK=3
      IF(IP.GT.IPF.AND.IP.LT.NT) IK=4
      IF(IP.EQ.NT) IK=5
C-----
      GOTO(10,20,30,20,40),IK
C-----
   10 I=IP+1
      DMT(IP)=LR-L(IP)+V(I)-V(IP)
      DXT(IP)=(LR*(XD-X(IP,IC))+V(IP)*(X(IP,IC)-
     1YACT(IP,IC))+V(I)*(YACT(I,IC)-X(IP,IC)))/M(IP)
      GOTO 50
   20 J=IP-1
      I=IP+1
      DMT(IP)=L(J)-L(IP)+V(I)-V(IP)
      DXT(IP)=(L(J)*(X(J,IC)-X(IP,IC))+V(IP)*(X(IP,IC)-
     1YACT(IP,IC))+V(I)*(YACT(I,IC)-X(IP,IC)))/M(IP)
      GOTO 50
   30 CALL DCOL5(IK,IP,DMT,DXT)
      GOTO 50
   40 CALL DCOL5(IK-1,IP,DMT,DXT)
   50 CONTINUE
      RETURN
C-----
   55 CONTINUE
      DO 300 IP=1,NT
      IF(IP.EQ.NT) GOTO 250
      DENM=X(IP,IC)*DEN1+(1.-X(IP,IC))*DEN2
      ZZ=1.E2*M(IP)/(DENM*AREA)
      ZZ=(ZZ-HC)/THETA
      ZZ=ZZ*PAI*DDIAM/10.
      L(IP)=ZZ*DENM
      GOTO 300
  250 L(IP)=L0
  300 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE DCOL3(ENTHV,ENTHL)
      REAL L,L0,LR,NUM,LAMDA1,LAMDA2
      COMMON/COLUMN/X(11,2),Y(11,2),YACT(11,2),SSM(11),L(11)
     1,T(11),V(11),EMV(11),L0,D,F,XF,XD,XB,FT,RT,LR,Q
      COMMON/DATA/AREA,PAI,HC,THETA
     2,LAMDA1,LAMDA2,DEN1,DEN2,RAD,DDIAM,HA(2),HB(2),HHC(2)
      COMMON/KOLPAR/NT,NPLATE,NCOMP,IPF,ICC
      COMMON/WORK/ENTHF(2),ENTHR(2),HL(15),HV(15),ZZZ,FHEAT
     3,RHEAT,ZH,COMPF,COMPR,NUM,DENM,ZZ,S10
      DIMENSION ENTHV(11,2),ENTHL(11,2)
C-----
      ZZZ=FT-25.
      DO 1400 IC=1,NCOMP
 1400 ENTHF(IC)=(HA(IC)+HB(IC)*ZZZ+HHC(IC)*ZZZ**2.0)/1000.
      FHEAT=0.
      DO 1600 IC=1,NCOMP
      IF(IC.EQ.1) COMPF=XF
      IF(IC.EQ.2) COMPF=1.-XF
      FHEAT=FHEAT+ENTHF(IC)*COMPF
 1600 CONTINUE
C-----
      ZZZ=RT-25.
      DO 1800 IC=1,NCOMP
 1800 ENTHR(IC)=(HA(IC)+HB(IC)*ZZZ+HHC(IC)*ZZZ**2.0)/1000.
      RHEAT=0.
      DO 2000 IC=1,NCOMP
      IF(IC.EQ.1) COMPR=XD
      IF(IC.EQ.2) COMPR=1.-XD
      RHEAT=RHEAT+ENTHR(IC)*COMPR
 2000 CONTINUE
C-----
      DO 2400 IP=1,NT
      HL(IP)=0.
      HV(IP)=0.
      DO 2200 IC=1,NCOMP
      HL(IP)=HL(IP)+X(IP,IC)*ENTHL(IP,IC)
      HV(IP)=HV(IP)+YACT(IP,IC)*ENTHV(IP,IC)
 2200 CONTINUE
 2400 CONTINUE
C-----
      V(NT)=(L(NPLATE)*HL(NPLATE)-L0*HL(NT)+Q)/HV(NT)
      RETURN
      END
```

```
      SUBROUTINE DCOL4(L,V,F,LR)
      REAL NUM,LR,L
      COMMON/KOLPAR/NT,NPLATE,NCOMP,IPF,ICC
      COMMON/WORK/ENTHF(2),ENTHR(2),HL(15),HV(15),ZZZ,FHEAT
     2,RHEAT,ZH,COMPF,COMPR,NUM,DENM,ZZ,S10
      DIMENSION L(1),V(1)
C-----
      IP=NPLATE
 2500 CONTINUE
      K=IP+1
      IF(IP.NE.1) GOTO 2510
      J=IP
      GOTO 2520
 2510 J=IP-1
 2520 CONTINUE
      IF(IP.EQ.IPF) NUM=L(J)*HL(J)-L(IP)*HL(IP)+
     1V(K)*HV(K)+F*FHEAT
      IF(IP.EQ.1)   NUM=LR*RHEAT-L(IP)*HL(IP)+V(K)*HV(K)
      IF(IP.NE.1.AND.IP.NE.IPF) NUM=L(J)*HL(J)-
     1L(IP)*HL(IP)+V(K)*HV(K)
      V(IP)=NUM/HV(IP)
      IP=IP-1
      IF(IP.LT.1) RETURN
      GOTO 2500
      END



      SUBROUTINE DCOL5(IK,IP,DMT,DXT)
      REAL L,M,L0,LR
      COMMON/COLUMN/X(11,2),Y(11,2),YACT(11,2),M(11),L(11)
     1,T(11),V(11),EMV(11),L0,D,F,XF,XD,XB,FT,RT,LR,Q
      COMMON/KOLPAR/NT,NPLATE,NCOMP,IPF,IC
      DIMENSION DMT(1),DXT(1)
C-----
      IK=IK-2
      GOTO (30,40),IK
   30 J=IP-1
      I=IP+1
      DMT(IP)=L(J)-L(IP)+V(I)-V(IP)+F
      DXT(IP)=(L(J)*(X(J,IC)-X(IP,IC))+V(IP)*(X(IP,IC)-
     1YACT(IP,IC))+V(I)*(YACT(I,IC)-X(IP,IC))+F*(XF-
     2X(IP,IC)))/M(IP)
      GOTO 50
   40 J=IP-1
      I=IP+1
      DMT(IP)=L(NPLATE)-L0-V(NT)
      DXT(IP)=(L(J)*(X(J,IC)-X(IP,IC))+V(IP)*(X(IP,IC)-
     1Y(IP,IC)))/M(IP)
   50 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE DCOL6(DX,DM,DZ,X,M,Z)
      REAL M
      COMMON/KOLPAR/NT,NPLATE,NCOMP,IPF,IC
      DIMENSION X(11,2),M(1),Z(1),DX(1),DM(1),DZ(1)
C-----
      ND=NT+NT
      DO 100 IP=1,ND
      IF(IP.GT.NT) GOTO 95
      X(IP,IC)=Z(IP)
      DX(IP)=DZ(IP)
      GOTO 100
   95 J=IP-NT
      M(J)=Z(IP)
      DM(J)=DZ(IP)
  100 CONTINUE
      RETURN
      END



      SUBROUTINE DCOL2(ENTHV,ENTHL)
      REAL L,LAMDA1,LAMDA2,L0
      COMMON/COLUMN/X(11,2),Y(11,2),YACT(11,2),SSM(11),L(11)
     1,T(11),V(11),EMV(11),L0,D,F,XF,XD,XB,FT,RT,XLR,Q
      COMMON/DATA/AREA,PAI,HC,THETA
     2,LAMDA1,LAMDA2,DEN1,DEN2,RAD,DDIAM,HA(2),HB(2),HHC(2)
      COMMON/KOLPAR/NT,NPLATE,NCOMP,IPF,ICC
      DIMENSION ENTHV(11,2),ENTHL(11,2)
C-----
      IC=ICC
      YACT(NT,IC)=Y(NT,IC)*EMV(NT)
      IP=NPLATE
  500 CONTINUE
      K=IP+1
      YACT(IP,IC)=EMV(IP)*Y(IP,IC)+(1.-EMV(IP))*YACT(K,IC)
      IP=IP-1
      IF(IP.LT.1) GOTO 600
      GOTO 500
  600 CONTINUE
      DO 900 IP=1,NT
      IF(IC.NE.1) GOTO 800
      X(IP,2)=1.-X(IP,1)
      Y(IP,2)=1.-Y(IP,1)
      YACT(IP,2)=1.-YACT(IP,1)
      GOTO 900
  800 CONTINUE
      X(IP,1)=1.-X(IP,2)
      Y(IP,1)=1.-Y(IP,2)
      YACT(IP,1)=1.-YACT(IP,2)
  900 CONTINUE
C-----
      DO 1200 IP=1,NT
      DO 1100 IC=1,NCOMP
      ZZZ=T(IP)-25.
      ENTHL(IP,IC)=(HA(IC)*ZZZ+.5*HB(IC)*ZZZ**2.0+
     1(1./3.)*HHC(IC)*ZZZ**3.0)/1000.
      IF(IC.EQ.1) ZH=LAMDA1
      IF(IC.EQ.2) ZH=LAMDA2
      ENTHV(IP,IC)=ENTHL(IP,IC)+ZH
 1100 CONTINUE
 1200 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE PERTUB(TIME,S,R,W,F1,F2)
      REAL L0,LR,M,L,L01,LR1
      COMMON/COLUMN/SX(11,2),SY(11,2),SYACT(11,2),SSM(11),SL(11)
     1,ST(11),SV(11),EMV(11),L0,D0,FEED,XFEED,XD,XB,FT,RT,LR,Q
      COMMON/MODEL/X(11,2),Y(11,2),YACT(11,2),M(11),L(11),T(11),V
     1,L01,D1,FEED1,XF1,XD1,XB1,LR1,Q1
      DIMENSION S(1),R(1),W(1)
C-----
      ITYPE=IFIX(F2)
      IF(ITYPE.EQ.0) RETURN
      IVAR=IFIX(F1)
      GOTO (10,20,30),ITYPE
C-----
10    STIME=S(1)
      SMAG=S(2)
      GOTO (11,12,13,14,15),IVAR
C-----
11    XFEED=STEP(TIME,STIME,XFEED,SMAG)
      RETURN
12    FEED=STEP(TIME,STIME,FEED,SMAG)
      RETURN
13    FT=STEP(TIME,STIME,FT,SMAG)
      RETURN
14    LR=STEP(TIME,STIME,LR,SMAG)
      RETURN
15    RT=STEP(TIME,STIME,RT,SMAG)
      RETURN
C-----
20    GOTO (21,22,23,24,25),IVAR
21    XFEED=RAMP(TIME,R(1),R(2),R(3),XFEED)
      RETURN
22    FEED=RAMP(TIME,R(1),R(2),R(3),FEED)
      RETURN
23    FT=RAMP(TIME,R(1),R(2),R(3),FT)
      RETURN
24    LR=RAMP(TIME,R(1),R(2),R(3),LR)
      RETURN
25    RT=RAMP(TIME,R(1),R(2),R(3),RT)
      RETURN
C-----
30    GOTO(31,32,33,34,35),IVAR
31    XFEED=SWAVE(TIME,W(1),W(2))
      RETURN
32    FEED=SWAVE(TIME,W(1),W(2))
      RETURN
33    FT=SWAVE(TIME,W(1),W(2))
      RETURN
34    LR=SWAVE(TIME,W(1),W(2))
      RETURN
35    RT=SWAVE(TIME,W(1),W(2))
      RETURN
      END
```

```
      SUBROUTINE INTJS(JS4,JS,JE,IOD,IPASS)
      IF(IOD.EQ.1) RETURN
      IF(IOD.EQ.2) JS=IPASS
      RETURN
      END



      FUNCTION RAMP(TIME,RTIME,ERAMP,RMAG,FNT)
      SLOPE=RMAG/(ERAMP-RTIME)
      IF(TIME.LT.RTIME.OR.TIME.GT.ERAMP) I=1
      IF(TIME.LT.ERAMP.AND.TIME.GT.RTIME) I=2
      IF(ABS(TIME-ERAMP).LT..00005)       I=3
      GOTO (1,2,3),I
1     RAMP=FNT
      RETURN
2     RAMP=FNT+SLOPE*(TIME-RTIME)
      RETURN
3     RAMP=FNT+RMAG
      RETURN
      END



      FUNCTION STEP(TIME,STIME,FNT,SMAG)
      IF(TIME.LT.STIME.OR.TIME.GT.STIME) I=1
      IF(ABS(TIME-STIME).LT..00005)      I=2
      IF(I.EQ.1) GOTO 1
      IF(I.EQ.2) GOTO 2
    1 STEP=FNT
      RETURN
    2 STEP=FNT+SMAG
      RETURN
      END



      FUNCTION SWAVE(TIME,WMAG,FREQ)
      SWAVE=WMAG*SIN(FREQ*TIME)
      RETURN
      END
```

```
      SUBROUTINE COMN(C,D,E,REFLUX,HEAT,E2,E3)
C-----Subroutine used to retrieve variable values from COMMON areas..
C     It is called from BASIC
C
      REAL LR1,LR,L0,L01
      COMMON/COLUMN/SX(11,2),SY(11,2),SYACT(11,2),SSM(11),SL(11)
     1,ST(11),SV(11),EMV(11),L0,D0,F0,XF,XD,XB,FT,RT,LR,Q
      COMMON/MODEL/SX1(11,2),SY1(11,2),SYACT1(11,2),SSM1(11),SL1(11)
     2,ST1(11),SV1,L01,D1,F1,XF1,XD1,XB1,LR1,Q1
      COMMON/KOLPAR/NT,NPLATE,NCOMP,IPF,IC
      COMMON/FILTER/P(15,15),FKT(15,15),EX(11),FKM(7,15)
     1,FKK(15,7),ET2,ET4,ET6,ET8,ET10,EFR,ELR,EXF,EV
      DIMENSION C(1),D(1),E(1)
C-----
      IE2=E2
      IE3=E3
      IF(IE3.EQ.0) GOTO 50
      LR=REFLUX
      Q=HEAT
      RETURN
C-----
   50 CONTINUE
      DO 80 IP=1,NT
      C(IP)=SX(IP,IC)
      C(IP+11)=SYACT(IP,IC)
      C(IP+22)=SSM(IP)
      C(IP+33)=SL(IP)
      C(IP+44)=ST(IP)
      C(IP+55)=EMV(IP)
      C(IP+66)=SV(IP)
   80 CONTINUE
C-----
      D(3)=F0
      D(4)=XF
      D(5)=RT
      D(6)=FT
      D(7)=LR
      D(11)=Q
C-----
      IF(IE2.EQ.0) RETURN
C-----
      DO 400 IP=1,NT
      E(IP)=SX1(IP,IC)
      E(IP+11)=SYACT1(IP,IC)
      E(IP+22)=SSM1(IP)
      E(IP+33)=SL1(IP)
      E(IP+44)=ST1(IP)
  400 CONTINUE
C-----
      E(56)=EFR
      E(57)=ELR
      E(58)=EXF
      E(59)=EV
      RETURN
      END
```

```
      SUBROUTINE SDINT(XX,DX,TD,DTD,IOD,INTCAL)
C
C-----Integrator subroutine : Simple and Modified Euler methods
c
      COMMON/INTEG/JS4,JS,JE,ND
      DIMENSION DXA(24)
      IF(INTCAL.EQ.1) JN=0
      GOTO(6,5),IOD
6     CONTINUE
      GOTO 7
C-----
5     JS=JS+1
      IF(JS.EQ.3) GOTO 20
      IF(JS.EQ.2) GOTO 10
C-----
7     DT=DTD
3     TD=TD+DT
10    CONTINUE
C-----
      JN=JN+1
      IF(JN.GT.ND) JN=JN-ND
      GOTO(19,18),IOD
C-----
19    XX=XX+DX*DT
      TD=TD-DTD
      RETURN
18    IF(JS.EQ.1) GOTO 11
      IF(JS.EQ.2) GOTO 12
C-----
11    DXA(JN)=DX
      XX=XX+DX*DT
      RETURN
C-----
12    XX=XX+(DX-DXA(JN))*DT/2.
      RETURN
C------
20    CONTINUE
      TD=TD-DTD
      RETURN
      END



      SUBROUTINE INTPAS(IPASS,IOD)
      COMMON/INTEG/JS4,JS,JE,ND
      IF(IOD.EQ.1) RETURN
      IF(JS.GT.2) JS=0
      IF(IOD.EQ.2.AND.IPASS.GT.2) IPASS=0
      RETURN
      END
```

```
:L
      SUBROUTINE SDBUB(IIP,IIC,XX,YYC,TBUB)
      COMMON/BUB/NPLATE,NCOMP,TCP,EPSIL
      COMMON/HOUSE/VA,VB,A(2),B(2),C(2),F
      COMMON/WORK/Z(2),P(2),Y(15),G(15),S1,S2,S3,S4,S5,S6
     1,S7,S8,S9,S10
C-----
      J7=IIC
      Z(J7)=XX
      Z(2)=1.-Z(1)
      DO 20  J=1,NCOMP
      IF(IC.GT.1) GOTO 1
      S1=(2.*VB-VA)*(1.-Z(J))**2.0
      S2=2.*(VA-VB)*(1.-Z(J))**3.00
      G(IC)=EXP(F*(S1+S2))
      GOTO 2
    1 CONTINUE
      S1=(2.*VA-VB)*(1.-Z(J))**2.0
      S2=2.*(VB-VA)*(1.-Z(J))**3.0
      G(J)=EXP(F*(S1+S2))
    2 CONTINUE
   20 CONTINUE
C-----
   25 CONTINUE
      S1=0.
      S2=0.
      DO 30 J=1,NCOMP
      P(J)=EXP(F*(A(J)-B(J)/(TBUB+C(J))))
      Y(J)=G(J)*Z(J)*P(J)/TCP
      S1=S1+Y(J)
      S3=-F*B(J)*Y(J)/((TBUB+C(J))**2.0)
      S2=S2+S3
   30 CONTINUE
      S4=1.-S1
      IF(ABS(S4).LE.EPSIL) GOTO 50
      TBUB=TBUB-S4/S2
      GOTO 25
C-----
   50 CONTINUE
      DO 60 J=1,NCOMP
      P(J)=EXP(F*(A(J)-B(J)/(TBUB+C(J))))
      Y(J)=G(J)*Z(J)*P(J)/TCP
   60 CONTINUE
C-----
      J=IIC
      YYC=Y(J)
      RETURN
      END
```

Table 6.10 FORTRAN Subroutines: for Model II

BASIC Call

SIMUL: CALL(2,T3,T4,Z2,T9)

T3 is time

T4 is integration step length

Z2 is integration method

T9 steering flag (integrate Model II or refine Model II profiles based on latest estimates)

```
      SUBROUTINE SIMUL(TIME,DT,RIOD,KFLAG)
      REAL LR,L,LAMDA,LAMDA1,LAMDA2,M,L0,KFLAG
C
C-----This subroutine in conjunction with SIMULX form
C     the FILTER model..
C
      COMMON/COLUMN/SX(11,2),SY(11,2),SYACT(11,2),SSM(11),SL(11),
     1ST(11),SVR(11),EM(11),SL0,SD0,SF0,SXF,SXD,SXB,FT,RT,RL0,Q0
      COMMON/MODEL/X(11,2),Y(11,2),YACT(11,2),M(11),L(11),T(11)
     1,SV,L0,D,F,XF,XD,XB,LR,Q
      COMMON/DATA/AREA,PAI,HC,THETA
     2,LAMDA1,LAMDA2,DEN1,DEN2,RAD,DDIAM,HA(2),HB(2),HHC(2)
      COMMON/KOLPAR/NT,NPLATE,NCOMP,IPF,ICC
      COMMON/INTEG/JS4,JS,JE,ND
      COMMON/WORK/G(2),PP(2),DXT(15),Z(15),ZZ,DENM,S3,S4
     3,S5,S6,S7,S8,S9,S10
C-----
      IC=ICC
      IF(KFLAG.NE.0.) GOTO 50000
      IOD=IFIX(RIOD)
C-----
      JS=0
      JE=0
      IPASS=0
C-----
      VBOTT=SV
      VT=VBOTT
    9 CONTINUE
      L0=L(NPLATE)-VBOTT
      D=VT-LR
      XD=YACT(1,IC)
C-----
      CALL SIMULX(VBOTT)
C-----
      STP=TIME
      DO 90 INTCAL=1,NT
      TIME=STP
      CALL INTJS(JS4,JS,JE,IOD,IPASS)
      CALL SDINT(X(INTCAL,IC),DXT(INTCAL),TIME,DT,IOD,INTCAL)
   90 CONTINUE
50000 CONTINUE
C-----
      DO 200 IP=1,NT
  200 CALL SDBUB(IP,IC,X(IP,IC),Y(IP,IC),T(IP))
C-----
      LAMDA=X(NT,IC)*LAMDA1+(1.-X(NT,IC))*LAMDA2
      VBOTT=Q/LAMDA
      VT=VBOTT
C-----
      YACT(NT,IC)=Y(NT,IC)*EM(NT)
      IP=NPLATE
C-----
  350 CONTINUE
      K=IP+1
      YACT(IP,IC)=EM(IP)*Y(IP,IC)+(1.-EM(IP))*YACT(K,IC)
      IP=IP-1
      IF(IP.LT.1) GOTO 360
      GOTO 350
  360 CONTINUE
```

```
C-----
      DO 250 I=1,NT
      IF(I.GE.IPF) GOTO 230
      L(I)=LR
      GOTO 250
  230 IF(I.EQ.NT) GOTO 240
      L(I)=LR+F
      GOTO 250
  240 L(I)=L0
  250 CONTINUE
C-----
      IF(KFLAG.NE.0.) GOTO 400
      IPASS=IPASS+1
      CALL INTPAS(IPASS,IOD)
      IF(JS.NE.0) GOTO 9
  400 SV=VBOTT
      RETURN
      END



      SUBROUTINE SIMULX(VBOTT)
C
C-----Used in conjunction with SIMUL for filter model integration..
C
      REAL L,M,L0,LR
      COMMON/MODEL/X(11,2),Y(11,2),YACT(11,2),M(11),L(11)
     1,T(11),SV,L0,D,F,XF,XD,XB,LR,Q
      COMMON/WORK/G(2),PP(2),DXT(15),Z(15),ZZ,DENM,S3,S4,S5,S6
     2,S7,S8,S9,S10
      COMMON/KOLPAR/NT,NPLATE,NCOMP,IPF,IC
C-----
      VT=VBOTT
      DO 50 IP=1,NT
      IF(IP.EQ.1) IK=1
      IF(IP.GT.1.AND.IP.LT.IPF) IK=2
      IF(IP.EQ.IPF) IK=3
      IF(IP.GT.IPF.AND.IP.LT.NT) IK=4
      IF(IP.EQ.NT) IK=5
      GOTO(10,20,30,20,40),IK
   10 K=IP+1
      DXT(IP)=(LR*(XD-X(IP,IC))+VT*(YACT(K,IC)-YACT(IP,IC)))/M(IP)
      GOTO 50
   20 J=IP-1
      K=IP+1
      DXT(IP)=(L(J)*(X(J,IC)-X(IP,IC))+VT*(YACT(K,IC)-
     1YACT(IP,IC)))/M(IP)
      GOTO 50
   30 J=IP-1
      K=IP+1
      DXT(IP)=(L(J)*X(J,IC)-L(IP)*X(IP,IC)+VBOTT*(YACT(K,IC)-
     +YACT(IP,IC))+F*XF)/M(IP)
      GOTO 50
   40 J=IP-1
      DXT(IP)=(L(J)*X(J,IC)-L(IP)*X(IP,IC)-VBOTT*YACT(IP,IC))/M(IP)
   50 CONTINUE
      RETURN
      END
```

Table 6.11 FORTRAN Subroutines: for Estimation with EKF1

BASIC Calls

KALMAN: CALL(5,Y(1),U(1),K(1),M(1),L(1),P9,D9)

where Y array contains the noisy measurements

U,K,M and L are workspace arrays

P9 is flag set in the matrix inversion routine

D9 is flag in matrix inversion routine

P10F10: CALL(3,S9)

where S9 is sample interval

PKK: CALL(6).

```
      SUBROUTINE KALMAN(Y,DX,XX,DY,YY,FLAG,DET)
C
C-----The Kalman Estimation step is done here.Measurement vector
C     Y is supplied from BASIC.Note reference trajectory is
C     previous estimates .. DET and FLAG flag any numerical
C     difficulty in the Matrix inversion.
C
      REAL LR,L0
      COMMON/MODEL/SX(11,2),SY(11,2),SYACT(11,2),SSM(11),SL(11)
     1,ST(11),SV,L0,D,F,XF,XD,XB,LR,Q
      COMMON/FILTER/P(15,15),FKT(15,15),EX(11),FKM(7,15)
     1,FKK(15,7),ET2,ET4,ET6,ET8,ET10,EFR,ELR,EXF,EV
      COMMON/CONVEC/FKR(7),FKQ(15)
      COMMON/INVERS/A(7,14)
      DIMENSION Y(1),DX(1),XX(1),DY(1),YY(1),SR(7,7)
C-----
      DO 50 I=1,11
      DX(I)=SX(I,1)-EX(I)
   50 CONTINUE
      DX(12)=F-EFR
      DX(13)=XF-EXF
      DX(14)=SV-EV
      DX(15)=LR-ELR
C-----
      DY(1)=Y(1)-ET2
      DY(2)=Y(2)-ET4
      DY(3)=Y(3)-ET6
      DY(4)=Y(4)-ET8
      DY(5)=Y(5)-ET10
      DY(6)=Y(6)-EFR
      DY(7)=Y(7)-ELR
C-----
      FKM(1,1)=FDTX(ET2)
      FKM(2,3)=FDTX(ET4)
      FKM(3,5)=FDTX(ET6)
      FKM(4,7)=FDTX(ET8)
      FKM(5,9)=FDTX(ET10)
      FKM(6,12)=1.
      FKM(7,13)=1.
C-----
C     CALCULATE K(k+1) NOW
C-----
      CALL MATTPS(7,15,FKM,FKK)
      CALL MATMUL(15,15,7,P,FKK,FKK,2)
      CALL MATMUL(7,15,7,FKM,FKK,SR,0)
      CALL DIAADD(7,SR,FKR,SR)
      CALL MATINV(7,SR,SR,FLAG,DET)
      CALL MATMUL(15,7,7,FKK,SR,FKK,1)
C-----
C     CALCULATE DX(k+1,k+1) NOW IE. STATE DEVIATIONS
C-----
      CALL MATVEC(7,15,FKM,DX,YY)
      CALL VECTOR(7,DY,YY,YY,2)
      CALL MATVEC(15,7,FKK,YY,XX)
      CALL VECTOR(15,DX,XX,DX,1)
      CALL KALMA1(DX)
      RETURN
      END
```

```
      SUBROUTINE KALMA1(DX)
C
C-----Used in conjunction with Subroutine KALMAN.
C
      REAL LR,L0
      COMMON/MODEL/SX(11,2),SY(11,2),SYACT(11,2),SSM(11),SL(11)
     1,ST(11),SV,L0,D,F,XF,XD,XB,LR,Q
      COMMON/FILTER/P(15,15),FKT(15,15),EX(11),FKM(7,15)
     1,FKK(15,7),ET2,ET4,ET6,ET8,ET10,EFR,ELR,EXF,EV
      DIMENSION DX(1)
C-----
      DO 100 I=1,11
      SX(I,1)=EX(I)+DX(I)
      IF(SX(I,1).GT.1.) SX(I,1)=.9999
      IF(SX(I,1).LT.0.)  SX(I,1)=.0001
  100 CONTINUE
      F=EFR+DX(12)
      XF=EXF+DX(13)
      SV=EV+DX(14)
      LR=ELR+DX(15)
      EFR=F
      ELR=LR
      EXF=XF
      EV=SV
      RETURN
      END



      SUBROUTINE PKK
C
C-----Calculates P(K,K)
C
      REAL L0,LR
      COMMON/MODEL/SX(11,2),SY(11,2),SYACT(11,2),SSM(11),SL(11)
     2,ST(11),SV,L0,D,F,XF,XD,XB,LR,Q
      COMMON/CONVEC/FKR(7),FKQ(15)
      COMMON/FILTER/P(15,15),FKT(15,15),EX(11),FKM(7,15)
     1,FKK(15,7),ET2,ET4,ET6,ET8,ET10,EFR,ELR,EXF,EV
      FKI=1.
      CALL MATMUL(15,7,15,FKK,FKM,FKT,0)
      CALL DIASUB(15,FKI,FKT,FKT)
      CALL MATMUL(15,15,15,FKT,P,P,2)
      CALL MATTPX(15,FKT,FKT)
      CALL MATMUL(15,15,15,P,FKT,P,1)
      CALL MATTPS(15,7,FKK,FKM)
      CALL DIAMUL(7,7,15,FKR,FKM,FKM)
      CALL MATMUL(15,7,15,FKK,FKM,FKT,0)
      CALL MATADD(15,P,FKT,P)
      RETURN
      END
```

```
      SUBROUTINE P10F10(S)
C
C------Calculates P(k+1,k)
C
      REAL L0,LR
      COMMON/MODEL/SX(11,2),SY(11,2),SYACT(11,2),SSM(11),SL(11)
     2,ST(11),SV,L0,D,F,XF,XD,XB,LR,Q
      COMMON/CONVEC/FKR(7),FK0(15)
      COMMON/FILTER/P(15,15),FKT(15,15),FX(11),FKM(7,15)
     1,FKK(15,7),ET2,ET4,ET6,ET8,ET10,EFR,ELR,EXF,EV
C-----
  100 CALL TRANS(S)
C-----
      DO 50 I=1,11
      SX(I)=SX(I,1)
   50 CONTINUE
      ET2=ST(1)
      ET4=ST(3)
      ET6=ST(5)
      ET8=ST(7)
      ET10=ST(9)
C-----CALCULATE P(K+1,K) NOW IE. BEFORE NEXT SAMPLING INSTANT
      CALL MATMUL(15,15,15,FKT,P,P,2)
      CALL MATTPX(15,FKT,FKT)
      CALL MATMUL(15,15,15,P,FKT,P,1)
      CALL DIAADD(15,P,FKQ,P)
C-----
      DO 80 I=1,7
      DO 70 J=1,15
      FKM(I,J)=0.
70    CONTINUE
80    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE TRANS(S)
C
C-----Calculates the system Transition Matrix by a first
C     order Taylor Series approximation.
C
      REAL LR,L0,K,M,L
      COMMON/MODEL/X(11,2),SY(11,2),Y(11,2),M(11),L(11)
     1,ST(11),SV,L0,D,F,XF,XD,XB,LR,Q
      COMMON/FILTER/P(15,15),FKT(15,15),EX(11),FKM(7,15)
     1,FKK(15,7),ET2,ET4,ET6,ET8,ET10,EFR,ELR,EXF,EV
      DIMENSION K(11)
C-----
      V=EV
      DO 20 I=1,15
      IM1=I-1
      IP1=I+1
      DO 10 J=1,15
      FKT(I,J)=0.
      IF(I.GT.11) GOTO 9
      K(I)=Y(I,1)/X(I,1)
      IF(J.EQ.IM1) FKT(I,J)=S*L(IM1)/M(I)
      IF(J.EQ.I  ) FKT(I,J)=1.-S*(L(I)+V*K(I))/M(I)
      IF(I.EQ.11 ) GOTO 10
      IF(J.EQ.IP1) FKT(I,J)=(S*V*Y(J,1)/X(J,1))/M(I)
      GOTO 10
    9 IF(I.EQ.J) FKT(I,J)=1.
   10 CONTINUE
   20 CONTINUE
C-----
      FKT(11,11)=1.-S*(L0+V*K(11))/M(11)
C-----
      FKT(1,1)=1.+S*(K(1)*(LR-V)-LR)/M(1)
      FKT(1,15)=S*(Y(1,1)-X(1,1))/M(1)
      FKT(7,12)=S*(XF-X(7,1))/M(7)
      FKT(7,13)=S*F/M(7)
      FKT(11,12)=S*X(10,1)/M(11)
      FKT(11,15)=S*X(10,1)/M(11)
C-----
      J=15
      DO 11 I=1,10
      T=S*(X(I-1,1)-X(I,1))/M(I)
      FKT(I,J)=T
      IF(I.LE.7) GOTO 11
      FKT(I,J-1)=T
   11 CONTINUE
C-----
      J=14
      DO 12 I=2,10
      FKT(I,J)=S*(Y(I+1,1)-Y(I,1))/M(I)
   12 CONTINUE
      FKT(11,14)=-S*Y(11,1)/M(11)
      RETURN
      END
```

```
      SUBROUTINE MATMUL(N,M,IP,A,B,C,IFLAG)
      DIMENSION A(N,M),B(M,IP),C(N,IP),T(15)
C
C-----If IFLAG .EQ. 0  then normal
C     If IFLAG .EQ. 1  then first matrix is answer
C     If IFLAG .EQ. 2  then second matrix is answer
C
      IF(IFLAG.NE.0) GOTO 3
      DO 11 I=1,N
      DO 9  J=1,IP
      S1=0.
      DO 8  K=1,M
      S1=S1+A(I,K)*B(K,J)
    8 CONTINUE
      C(I,J)=S1
    9 CONTINUE
   11 CONTINUE
      RETURN
C-----
    3 II=1
      JJ=1
    4 L=0
      I=0
      J=0
    5 CONTINUE
      GOTO(10,20),IFLAG
   10 J=J+1
      I=II
      GOTO 30
   20 I=I+1
      J=JJ
   30 L=L+1
      S1=0.
      DO 1 K=1,M
      S1=S1+A(I,K)*B(K,J)
    1 CONTINUE
      T(L)=S1
      IF(L.EQ.M) GOTO 6
      GOTO 5
    6 CONTINUE
C-----
      DO 50 L=1,M
      IF(IFLAG.EQ.1) C(II,L)=T(L)
      IF(IFLAG.EQ.2) C(L,JJ)=T(L)
   50 CONTINUE
C-----
      GOTO(60,70),IFLAG
   60 II=II+1
      IF(II.GT.N) RETURN
      GOTO 4
   70 JJ=JJ+1
      IF(JJ.GT.IP) RETURN
      GOTO 4
      END
```

```
      SUBROUTINE MATTPX(N,A,AT)
      DIMENSION A(N,N),AT(N,N)
      J=1
      DO 1 I=J,N
      DO 2 K=J,N
      IF(I.EQ. K) GOTO 2
      TEMP=A(I,K)
      AT(I,K)=A(K,I)
      A(K,I)=TEMP
    2 CONTINUE
      J=J+1
      IF(J.EQ.N) RETURN
    1 CONTINUE
      END
```

```
      SUBROUTINE DIAADD(N,A,B,C)
      DIMENSION A(N,N),B(N),C(N,N)
      DO 1 I=1,N
      C(I,I)=A(I,I)+B(I)
    1 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE DIASUB(N,FKI,B,C)
      DIMENSION B(N,N),C(N,N)
      DO 3 I=1,N
      DO 1 J=1,N
      IF(I.EQ.J) GOTO 2
    1 C(I,J)=-B(I,J)
      GOTO 3
    2 C(I,J)=FKI-B(I,J)
    3 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE MATADD(N,A,B,C)
      DIMENSION A(N,N),B(N,N),C(N,N)
      DO 2 I=1,N
      DO 1 J=1,N
    1 C(I,J)=A(I,J)+B(I,J)
    2 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE MATTPS(N,M,A,AT)
      DIMENSION A(N,M),AT(M,N)
      DO 2 I=1,N
      DO 1 J=1,M
    1 AT(J,I)=A(I,J)
    2 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE DIAMUL(N,N,IP,R,B,C)
      DIMENSION R(1),T(10),C(N,IP),B(N,IP)
      DO 3 J=1,IP
      DO 2 I=1,N
      T(I)=R(I)*B(I,J)
    2 CONTINUE
      DO 1 K=1,N
    1 C(K,J)=T(K)
    3 CONTINUE
      RETURN
      END
```

```
      FUNCTION FDTX(ET)
      COMMON/HOUSE/VA,VB,A(2),B(2),C(2),F
      COMMON/WORK/VP(2),PP(2),Y(15),G(15),P,F1,F2,DF1,DF2
     1,S6,S7,S8,S9,S10
      P=760.
      DO 1 I=1,2
    1 VP(I)=EXP(F*(A(I)-B(I)/(C(I)+ET)))
      F1=P-VP(2)
      F2=VP(1)-VP(2)
      DF1=-F*B(2)*VP(2)/((C(2)+ET)**2.)
      DF2=F*B(1)*VP(1)/((C(1)+ET)**2.)-F*B(2)*VP(2)/((C(2)+ET)**2.)
      FDTX=F2**2.0/(DF1*F2-DF2*F1)
      RETURN
      END



      SUBROUTINE VECTOR(N,X,Y,ANS,IFLAG)
      DIMENSION X(1),Y(1),ANS(1)
      DO 3 I=1,N
      GOTO (1,2),IFLAG
    1 ANS(I)=X(I)+Y(I)
      GOTO 3
    2 ANS(I)=X(I)-Y(I)
    3 CONTINUE
      RETURN
      END



      SUBROUTINE MATVEC(N,M,A5,XV,AX)
      DIMENSION A5(N,M),XV(M),AX(N)
      DO 2 I=1,N
      SUM=0.
      DO 1 J=1,M
      SUM=SUM+A5(I,J)*XV(J)
      AX(I)=SUM
      CONTINUE
      RETURN
      END
```

```
      SUBROUTINE MATINV(N,B,BI,IFLAG,DET)
C
C-----IFLAG and DET are error flags.
C
      REAL IFLAG
      COMMON/INVERS/A(7,14)
      DIMENSION B(7,7),BI(7,7)
      DET=1.0
      IFLAG=0.
      DO 200 I=1,N
      DO 100 J=1,N
100   A(I,J)=B(I,J)
200   CONTINUE
      EPSIL=1.E-10
      N1=2*N
      DO 3 I=1,N
      DO 3 J=1,N
      IF(I-J) 1,2,1
1     KK=J+N
      A(I,KK)=0.0
      GOTO 3
2     KK=J+N
      A(I,KK)=1.0
3     CONTINUE
      DO 50 IP=1,N
      IM=IP
      IST=IP+1
      IF(IST.GT.N) GOTO 11000
      DO 10 I=IST,N
      IF(ABS(A(IM,IP))-ABS(A(I,IP))) 9,10,10
9     IM=I
10    CONTINUE
11000 CONTINUE
C-----
      IF(ABS(A(IM,IP))-EPSIL) 11,13,13
11    IFLAG=1.
      IF(A(IM,IP)) 13,70,13
13    IF(IM-IP) 14,20,14
14    DO 15 J=IP,N1
      AL=A(IP,J)
      DET=DET*AL
      A(IP,J)=A(IM,J)
15    A(IM,J)=AL
20    AL=A(IP,IP)
      A(IP,IP)=1.
      DO 25 J=IST,N1
25    A(IP,J)=A(IP,J)/AL
      DO 40 I=1,N
      IF(I-IP) 30,40,30
30    AL=A(I,IP)
      DO 35 J=IP,N1
35    A(I,J)=A(I,J)-AL*A(IP,J)
40    CONTINUE
50    CONTINUE
C-----
      DO 60 I=1,N
      DO 55 J=1,N
      K=J+N
55    BI(I,J)=A(I,K)
60    CONTINUE
      RETURN
70    IFLAG=2.
      DET=0.0
      RETURN
      END
```

```
      SUBROUTINE MATINV(N,B,BI,IFLAG,DET)
C
C-----IFLAG and DET are error flags.
C
      REAL IFLAG
      COMMON/INVERS/A(7,14)
      DIMENSION B(7,7),BI(7,7)
      DET=1.0
      IFLAG=0.
      DO 200 I=1,N
      DO 100 J=1,N
100   A(I,J)=B(I,J)
200   CONTINUE
      EPSIL=1.E-10
      N1=2*N
      DO 3 I=1,N
      DO 3 J=1,N
      IF(I-J) 1,2,1
1     KK=J+N
      A(I,KK)=0.0
      GOTO 3
2     KK=J+N
      A(I,KK)=1.0
3     CONTINUE
      DO 50 IP=1,N
      IM=IP
      IST=IP+1
      IF(IST.GT.N) GOTO 11000
      DO 10 I=IST,N
      IF(ABS(A(IM,IP))-ABS(A(I,IP))) 9,10,10
9     IM=I
10    CONTINUE
11000 CONTINUE
C-----
      IF(ABS(A(IM,IP))-EPSIL) 11,13,13
11    IFLAG=1.
      IF(A(IM,IP)) 13,70,13
13    IF(IM-IP) 14,20,14
14    DO 15 J=IP,N1
      AL=A(IP,J)
      DET=DET*AL
      A(IP,J)=A(IM,J)
15    A(IM,J)=AL
20    AL=A(IP,IP)
      A(IP,IP)=1.
      DO 25 J=IST,N1
25    A(IP,J)=A(IP,J)/AL
      DO 40 I=1,N
      IF(I-IP) 30,40,30
30    AL=A(I,IP)
      DO 35 J=IP,N1
35    A(I,J)=A(I,J)-AL*A(IP,J)
40    CONTINUE
50    CONTINUE
C-----
      DO 60 I=1,N
      DO 55 J=1,N
      K=J+N
55    BI(I,J)=A(I,K)
60    CONTINUE
      RETURN
70    IFLAG=2.
      DET=0.0
      RETURN
      END
```

Table A6.12 BASIC program for estimation with EKF1

```
5  REM  ..... BASIC PROGRAM FOR USE WITH EKF1 .....
6  REM
7  REM   KALMAN FILTER SIMULATION OF A BINARY DISTILLATION PROCESS
8  REM   ESTIMATING 11 PLATE LIQUID COMPOSITIONS AND 4 PARAMETERS- FEE
9  REM   RATE,FEED COMPOSITION,REFLUX RATE AND BOIL-UP RATE..
10 REM    7 PROCESS MEASUREMENTS .. 5 TEMPERATURES AND 2 FLOWRATES
11 REM
12 DIM C(225),D(14),E(70),F(28),F(4),S(2),W(2),Z(7),Y(7),X(11)
13 DIM J(10),U(15),K(15),L(7),M(7)
14 GOSUB 9800
15 T5=0:E9=13:C9=FND(E9)
17 T1,T3,T4,T8,T9,D9=0
20 PRINT "INPUT MEASUREMENT NOISE PARAMETERS ";: INPUT C7,C8,C9
22 PRINT "PRINT INTERVAL ? ";: INPUT G2
27 PRINT "INPUT X0,X1,Y0,Y1,J7,X2";: INPUT X0,X1,Y0,Y1,J7,X2
28 S9=X2: REM  SAMPLING INTERVAL
30 PRINT "INPUT Z1,T2,T6 ";: INPUT Z1,T2,T6
31 PRINT "INPUT Z2,T4,T7 ";: INPUT Z2,T2,T7
32 PRINT "SET SS3 AND SS4 AND INPUT Z3 ";: INPUT Z3
40 F1=2:F2=1:L(1)=1:L(2)=10:E1=1:E4=0:V1=10.34
42 E4=1
45 GOSUB 1000:F1=2:E2=1
50 FOR I=1,6:F(I)=.1E-01: NEXT I: FOR I=7,11:F(I)=.5E-02: NEXT I
55 F(12),F(14),F(15)=1:F(13)=.15
60 FOR I=16,20:F(I)=.1E-01: NEXT I:F(21),F(22)=1
65 F(23)=.1E-02: FOR I=24,27:F(I)=.4E-01: NEXT I:F(25)=.1E-01
70 FOR I=3,13: READ D(I): NEXT I
80 DATA 10,.4,30.44,29.44,7.25,7
82 DATA 10,116.27,390,3.41,6.59
95 L(14)=.1E-02
110 GOSUB 1100
115 FOR I=1,11:X(I)=C(I): NEXT I
125 CALL (3,59)
126 GOSUB 800
127 GOSUB 3000: GOSUB 3030
128 GOSUB 9300: GOSUB 9305
130 GOSUB 8000
180 REM
200 F1=2:S(1)=.1:S(2)=4: GOSUB 999
201 S(1)=.16:S(2)=-4: GOSUB 999
202 S(1)=.22:S(2)=-3: GOSUB 999
204 S(1)=.4:S(2)=4: GOSUB 999
206 F1=1:S(1)=.18:S(2)=.5E-01: GOSUB 999
208 S(1)=.3:S(2)=-.7E-01: GOSUB 999
210 CALL (2,11,12,Z1)
220 T1=T1+12
230 T8=T8+1
233 IF T8=16 THEN 240
235 GOTO 200
240 GOSUB 9700
245 REM
250 CALL (2,13,14,Z2,0)
```

```
260  I3=I3+I4
270  I9=I9+1
280  IF I9=I7 THEN 400
290  GOTO 245
400  I8=0:I9=0:I4=I4+1
402  GOSUB 9650: GOSUB 9900: FOR I=1,11:X(I)=E(I): NEXT I
406  Y(1)=C(45)+C8*J(1):Y(2)=C(47)+C8*J(2):Y(3)=C(49)+C8*J(3)
407  Y(4)=C(51)+C8*J(4):Y(5)=C(53)+C8*J(5)
408  Y(6)=D(3)+C9*J(6):Y(7)=D(7)+C9*J(7)
409  GOSUB 810
410  CALL (2,I3,I4,Z1,1)
411  CALL (6)
412  IF I4=62 THEN 415
413  CALL (3,59): GOTO 430
415  GOSUB 3900:I4=0
420  GOSUB 800: GOSUB 3000: GOSUB 3000: PRINT : CALL (3,.5E-02)
422  GOSUB 3030
425  PRINT : PRINT "----------
430  GOSUB 9650: GOSUB 9300: GOSUB 9500
435  IF I1>10 THEN 465
440  GOTO 130
445  CALL (2,6,Z(0)):Z(7)=24: CALL (2,5,Z(0))
470  END
800  PRINT : GOSUB 9650
802  PRINT "STATE OF COLUMN ": PRINT
803  FOR I=1,11: PRINT C(I),C(I+11),C(I+22): NEXT I: PRINT
804  FOR I=1,11: PRINT C(I+33),C(I+44),C(I+66): NEXT I
805  GOSUB 900: PRINT : PRINT "STATE OF MODEL": PRINT
806  FOR I=1,11: PRINT X(I),E(I),E(I+11),E(I+33),E(I+44): NEXT I
808  PRINT E(56),E(57),E(58),E(59),D9
809  PRINT : PRINT "----------": RETURN
810  CALL (5,Y(1),D(1),K(1),L(1),M(1),F9,I9)
815  IF F9=0 THEN  RETURN
820  IF F9=1 THEN 830
825  IF F9=2 THEN 840
827  STOP
830  PRINT "PIVOTAL ELEMENT TOO SMALL IN MATINV": STOP
840  PRINT "PIVOTAL ELEMENT ZERO ": STOP
900  PRINT D(3),D(4),D(5),D(6),D(7)
901  RETURN
999  CALL (7,11,S(1),R(1),W(1),F1,F2): RETURN
1000  REM
1020  FOR I=1,11: INPUT C(I),C(I+11),C(I+22): NEXT I
1030  FOR I=1 TO 11: INPUT C(I+33),C(I+44),C(I+55): NEXT I
1037  FOR I=1,66:E(I)=C(I): NEXT I
1040  REM
1050  CALL (2,C(1),D(1),E(1),F(1),E1,E2,E4,V1)
1060  RETURN
1100  REM
1110  CALL (2,C(1),D(1),E(1),F(1),E1,E2,E4,V1)
1130  RETURN
2000  L2=25:S1=4:E2=.11771E05: GOSUB 8500: RETURN
3000  L2=36:S1=4:E2=9688: GOSUB 8500: RETURN
3030  CALL (2,C(1),15,15,1,0): GOSUB 3500
3040  CALL (2,C(1),15,15,2,0): GOSUB 3500
3050  IF I3=0 THEN  GOSUB 3000
3055  RETURN
```

```
3060  CALL (2,C(1),15,7,4,0): GOSUB 3600
3070  CALL (2,C(1),7,14,5,0): GOSUB 3620: PRINT
3100  CALL (2,C(1),15,15,1,0): GOSUB 3500: RETURN
3200  N9=7: CALL (2,C(1),N9,1,0,1): GOSUB 3250
3210  N9=15: CALL (2,C(1),N9,1,0,2)
3250  FOR I=1,N9: PRINT C(I): NEXT I: PRINT : RETURN
3500  FOR I=1,15: PRINT C(I),C(I+15),C(I+30),C(I+45),C(I+60): NEXT I
3505  PRINT
3510  FOR I=1,15: PRINT C(I+75),C(I+90),C(I+105),C(I+120),C(I+135)
3515  NEXT I: PRINT
3520  FOR I=1,15: PRINT C(I+150),C(I+165),C(I+180),C(I+195),C(I+210)
3525  NEXT I: PRINT : RETURN
3600  FOR I=1,15: PRINT C(I),C(I+15),C(I+30),C(I+45): NEXT I
3610  PRINT : FOR I=1,15: PRINT C(I+60),C(I+75),C(I+90): NEXT I: RETUR
3620  PRINT
3650  FOR I=1,7: PRINT C(I+49),C(I+56),C(63+I),C(70+I): NEXT I
3660  PRINT : FOR I=1,7: PRINT C(I+77),C(I+84),C(I+91): NEXT I: RETURN
3800  GOSUB 9650: FOR I=1,11: PRINT C(I),X(I),E(I): NEXT I
3810  PRINT : PRINT E(56),E(57),E(58),E(59): RETURN
3900  PRINT "MEASUREMENTS --- TIME =";T1: PRINT Y(1),Y(2),Y(3)
3910  PRINT Y(4),Y(5),Y(6),Y(7): PRINT : RETURN
8500  CALL (1,U1,U2,S1,B1,B2,E7)
8510  IF E7<>0 THEN 9000
8520  RETURN
9000  PRINT "DISK ERROR !": STOP
9300  U2=2:S1=25:B2=.11772E05: GOSUB 8500: RETURN
9305  Z(7)=29: CALL (2,5,Z(0))
9310  CALL (2,1,Z(0)):Z(1)=X0:Z(2)=X1:Z(3)=Y0:Z(4)=Y1: CALL (2,2,Z(0))
9320  Z(5)=X0:Z(6)=Y0:Z(0)=2: GOSUB 9600
9330  Z(5)=X1:Z(0)=3: GOSUB 9600
9340  Z(6)=Y0+Y1: GOSUB 9600:Z(5)=X0: GOSUB 9600
9350  Z(6)=Y0: GOSUB 9600: GOSUB 9650
9360  Z(6)=V1:Z(0)=2: GOSUB 9600
9370  Z(6)=V2: GOSUB 9600:Z(6)=V3: GOSUB 9600
9375  Z(6)=V4: GOSUB 9600:Z(6)=V1: GOSUB 9600
9380  X7=X0:W1=V1:W2=V2:W3=V3:W4=V4: RETURN
9500  T5=T5+X2
9505  Z(0)=3:Z(5)=15:Z(6)=V1: GOSUB 9600:V1=V1
9510  IF J7=1 THEN 9590
9520  Z(0)=2:Z(5)=X7:Z(6)=W2: GOSUB 9600
9530  Z(0)=3:Z(5)=15:Z(6)=V2: GOSUB 9600:W2=V2
9535  IF J7=2 THEN 9590
9537  Z(0)=2:Z(5)=X7:Z(6)=W3: GOSUB 9600
9540  Z(0)=3:Z(5)=15:Z(6)=V3: GOSUB 9600:W3=V3
9545  IF J7=3 THEN 9590
9550  Z(0)=2:Z(5)=X7:Z(6)=W4: GOSUB 9600
9555  Z(0)=3:Z(5)=15:Z(6)=V4: GOSUB 9600:W4=V4
9590  X7=15:Z(0)=2:Z(6)=W1: GOSUB 9600: RETURN
9600  CALL (2,3,Z(0)): RETURN
9650  F3=0: CALL (4,C(1),D(1),E(1),K0,G0,E2,F3)
9660  V1=C(10): V2=E(10): V3=C(4)-.4:V4=E(4)-.4: RETURN
9700  U2=37:S1=22:B2=.11768E05: GOSUB 8500: RETURN
9800  REM
9820  U1=0:U2=42:S1=18:B1=8192:B2=.11756E05: GOSUB 8500: RETURN
9900  FOR I=1,7:K7=0: FOR I2=1,50:K7=K7+INT(C): NEXT I2
9992  J(I)=C7*(K7-25): NEXT I: RETURN
```

Table A6.13 Memory map of Segment 1: EKF2

*LOW	05470
*START	17777
*HIGH	33224
*NAMES	11445
*COMN	34737
*BASE	33253
*BASE	32764
*BASE	30772
*BASE	31765
*BASE	27760
*BASE	26746
*BASE	25761
*BASE	24757
*BASE	23757
*BASE	22761
*BASE	21760
*BASE	20763
ABS	05470
L≠22	05554
H≠22	05560
N≠22	05576
S≠22	05753
A≠22	05760
D≠22	06035
M≠11	06163
M≠22	06204
C≠12	06247
E≠22	06370
ALOG	06477
EXP	06563
SQRT	06677
SIN	06757
ATAN	07065
KOMMON	20000
INIT	20006
INIT1	21000
INIT2	22000
INIT3	23000
P10F10	24000
MATTPX	24162
TRANS	25000
MATMUL	26000
COMN	27000
MATADD	27524
FDTX	30470
SDBUB	31000
DIAADD	31504
KALMA1	32000
SDINT	32164
MATTPS	32502
INTPAS	33000
INTJS	33052
L≠33	33110
INT	33124
IDINT	33124
IFIX	33124
C≠21	33134
F≠AT	33142
PRIN	34737
DATA	35013
HOUSE	35053
BUB	35075
CONVEC	35101
INTEG	35135
INVERS	35141
WORK	35205
COLUMN	35335
MODEL	35743
FILTER	36273
KOLPAR	37477

Table A6.14 Memory map of Segment 2: EKF2

*LOW	05470
*START	17777
*HIGH	33224
*NAMES	11541
*COMN	34737
*BASE	33253
*BASE	32764
*BASE	31765
*BASE	26773
*BASE	25765
*BASE	24773
*BASE	23774
*BASE	22775
*BASE	21774
*BASE	20775
ABS	05470
L£22	05554
H£22	05560
N£22	05576
S£22	05753
A£22	05760
D£22	06035
M£11	06163
M£22	06204
C£12	06247
E£22	06370
ALOG	06477
EXP	06563
SQRT	06677
SIN	06757
ATAN	07065
KOMMON	20000
DCOL	20006
DCOL1	21000
DCOL2	22000
DCOL3	23000
DCOL5	24000
DCOL6	24444
PERTUB	25000
RAMP	25302
DCOL4	26000
STEP	26434
SDBUB	31000
DIAADD	31504
KALMA1	32000
SDINT	32164
MATTPS	32502
INTPAS	33000
INTJS	33052
L£33	33110
INT	33124
IDINT	33124
IFIX	33124
C£21	33134
F£AT	33142
PRIN	34737
DATA	35013
HOUSE	35053
BUB	35075
CONVEC	35101
INTEG	35135
INVERS	35141
WORK	35205
COLUMN	35335
MODEL	35743
FILTER	36273
KOLPAR	37477

Table A6.15 Memory map of Segment 3: EKF2

*LOW	05470	MATINV	23000
*START	17777	P10F10	24000
*HIGH	33224	DIASUB	24162
*NAMES	11321	DIAMUL	24312
*COMN	34737	MATTPX	24476
*BASE	31765	TRANS	25000
*BASE	26773	MATMUL	26000
*BASE	25761	PKK	26500
*BASE	24762	MATVEC	26624
*BASE	23771	COMN	27000
*BASE	32764	MATADD	27524
*BASE	33253	FDTX	30470
*BASE	30772	SDBUB	31000
*BASE	27760	DIAADD	31504
*BASE	22764	KALMA1	32000
*BASE	21775	SDINT	32164
*BASE	20774	MATTPS	32502
ABS	05470	INTPAS	33000
L£22	05554	INTJS	33052
H£22	05560	L£33	33110
N£22	05576	INT	33124
S£22	05753	IDINT	33124
A£22	05760	IFIX	33124
D£22	06035	C£21	33134
M£11	06163	F£AT	33142
M£22	06204	PRIN	34737
C£12	06247	DATA	35013
E£22	06370	HOUSE	35053
ALOG	06477	BUB	35075
EXP	06563	CONVEC	35101
SQRT	06677	INTEG	35135
SIN	06757	INVERS	35141
ATAN	07065	WORK	35205
KOMMON	20000	COLUMN	35335
SIMUL	20006	MODEL	35743
SIMULX	21000	FILTER	36273
VECTOR	21636	KOLPAR	37477
KALMAN	22000		

Table A7.1 On-line BASIC program: Distillation Column

```
10  REM  .. M6800 CONTROL OF REBOILER AND REFLUX HOLD-UPS
15  DIM A(13),B(87),C(7),D(29):N,T,S=0
20  PRINT "INPUT A(1),A(2) AND M6800 A(0) ";: INPUT A(1),A(2),S1
22  A(12)=1:S=S1*A(1)
25  PRINT "INPUT P=1 IF PRINT TEMPERATURES ";: INPUT P
30  PRINT "INPUT P9=1 IF PLOT ";: INPUT P9: IF P9=1 THEN  GOSUB 400
35  REM      ------------------------------------------------------
40  CALL (1,A(0),B(0))
50  REM       GET HOLD-UP VALUES
60  GOSUB 1000
80  REM       GET M6800 CONTROL VALUES
85  GOSUB 2000
87  IF P9=0 THEN 110
90  IF I7=0 THEN  GOSUB 410
100  GOSUB 500
110  CALL (2)
120  CALL (5,6,C(0)):C(7)=24: CALL (5,5,C(0))
132  END
400  PRINT "INPUT X0,X1,Y0,Y1,M6800 A(0) ";: INPUT X0,X1,Y0,Y1,S:I7=0
405  RETURN
410  REM       INITIALISE GRAPHICS
420  C(7)=29: CALL (5,5,C(0))
430  CALL (5,1,C(0)):C(1)=X0:C(2)=X1:C(3)=Y0:C(4)=Y1: CALL (5,2,C(0))
440  C(5)=X0:C(6)=Y0:C(0)=2: GOSUB 9600:C(5)=X1:C(0)=3: GOSUB 9600
450  Y1=Y0+Y1:C(6)=Y1: GOSUB 9600:C(5)=X0: GOSUB 9600:C(6)=Y0: GOSUB 960
460  I7=1
470  C(0)=2:C(6)=L1: GOSUB 9600:C(6)=L2: GOSUB 9600:C(6)=L1: GOSUB 9600
480  X7=X0:W1=L1:W2=L2: RETURN
500  REM       PLOT ROUTINE
510  X7=T:T=T+S
520  C(0)=3:C(5)=T:C(6)=L1: GOSUB 9600:W1=L1
530  C(0)=2:C(5)=X7:C(6)=W2: GOSUB 9600
540  C(0)=3:C(5)=T:C(6)=L2: GOSUB 9600:W2=L2
550  REM
555  IF N=1 THEN 700
560  C(0)=2:C(5)=X7:C(6)=C1: GOSUB 9600
570  C(0)=3:C(5)=T: GOSUB 9600:W3=C1
580  C(0)=2:C(5)=X7:C(6)=C2: GOSUB 9600
590  C(0)=3:C(5)=T: GOSUB 9600:W4=C2
595  N=1
600  C(0)=2:C(6)=W1: GOSUB 9600: RETURN
700  C(0)=2:C(5)=X7:C(6)=W3: GOSUB 9600
710  C(0)=3:C(6)=C1: GOSUB 9600:C(5)=T: GOSUB 9600:W3=C1
720  C(0)=2:C(5)=X7:C(6)=W4: GOSUB 9600
730  C(0)=3:C(6)=C2: GOSUB 9600:C(5)=T: GOSUB 9600:W4=C2
740  GOTO 600
1000  V1=B(45)*5/1023:L1=(-.58684E-01+.541*V1)/1.5
1005 0V2=B(44)*5/1023:L2=(-60+40.1*V2)/1.6
1010  IF P=0 THEN  RETURN
1020  T1=B(37)*5*23.181/1023+3.2204
1030  T7=B(38)*5*23.422/1023+4.688
1040  T0=B(36)*5*23.038/1023+3.6394
1050  T=T+S: PRINT "TIME=";T
1060  PRINT L1;L2;T1;T7;T0;C1;C2: PRINT
1070  RETURN
2000  C1=B(57)/.32767E05*10
2005  C2=B(58)/.32767E05*10
2010  RETURN
9600  CALL (5,3,C(0)): RETURN
```

Table A7.2 On-line SD BASIC program: Distillation Column

```
   PROGRAM ORIGIN :4000
   DATA ORIGIN    :5500
   REM
   REM .. PROGRAM FOR M6800 CONTROL OF REBOILER AND REFLUX
   REM    HOLD-UPS IN THE IBM DISTILLATION COLUMN
   REM    CONTROL O/P'S ARE SENT TO THE H316 EVERY SCAN
   REM    INTERVAL ..
   DIM A(11),B(86),D(29),E1(1),E2(1),V(6),BIAS
   DIM TIMER/:06/,PFLAG/0/,I/0/,EN,LN,N,U,M,T1,T7,T10
   DIM J,VARPI,K1,K2,TI1,TI2,S1,S2,STIME,SMAG,U1,U2,UN,L1,L2
   DIM XFLAG/0/,P1,P2,V1,V2,ISTEP/0/
   REM
   ON ERROR GOTO 9999
   FOR J=0 TO 1 \ E1(J)=0.0 \ E2(J)=0.0 \ NEXT J
   FOR J=0 TO 11 \ A(J)=0 \ NEXT J
   FOR J=0 TO 86 \ B(J)=0 \ NEXT J
   FOR J=0 TO 29 \ D(J)=0 \ NEXT J
   REM
 5 CALL SUB0
   INPUT "A(0),A(2),VARPI,STIME,SMAG " A(0),A(2),VARPI,STIME,SMAG
   INPUT "REF. & REB. SET-POINTS AND PERC1" S1,S2,P1
   INPUT "P+I CONTROL PARAMETERS K1,K2,TI1,TI2 " K1,K2,TI1,TI2
   PRINT "INPUT VALVE SETTINGS V(1),V(2),V(3),V(5),V(6)"
   INPUT V(1),V(2),V(3),V(5),V(6)
   REM
   A(1)=1 \ A(3)=36 \ A(4)=47 \ A(5)=33
10 CALL SUB1(A(0),B(0))
   GOSUB 9500
   IF XFLAG=0 THEN GOSUB 600
   I=I+1 \ IF I=VARPI THEN PFLAG=1
   ISTEP=ISTEP+1 \ IF ISTEP=STIME THEN GOSUB 900
   FOR J=36 TO 47 \ IF B(J)<0 THEN B(J)=0 \ NEXT J
   T1=B(37)*5*23.181/1023+3.2204
   T7=B(38)*5*23.422/1023+4.688
   T10=B(36)*5*23.038/1023+3.6394
   V1=B(45)*5/1023 \ L1=(-.05864+.541*V1)/1.5
   V2=B(44)*5/1023 \ L2=(-60.0+40.1*V2)/1.6
   REM
20 IF L1 > P1*S1 THEN GOTO 25
   N=5 \  GOSUB 9000 \ GOTO 27
25 E1(1)=L1-S1
   U1=V(5) + K1*(E1(1)-E1(0)) + K1*A(0)*E1(1)/TI1
   IF U1<0.0 THEN U1=0.0 \ IF U1>10.0 THEN U1=10.
   N=5 \ UN=U1*32767/10 \ GOSUB 8000
   V(5)=U1 \ GOTO 28
27 V(5)=UN
28 D(0)=U  \ E1(0)=E1(1)
   REM
30 E2(1)=L2-S2
   U2=V(6) + K2*(E2(1)-E2(0)) + K2*A(0)*E2(1)/TI2
```

```
     IF U2<0.0 THEN U2=0.0 \ IF U2>10.0 THEN U2=10.0
     N=2 \ UN=U2*32767/10 \ GOSUB 8000
     D(1)=U
     V(6)=U2 \ E2(0)=E2(1)
     REM
  35 IF PFLAG=1 THEN GOSUB 800
     REM
     M=2
     CALL SUB4(M,D(0))
  40 CALL SUB2
     GOTO 10
 600 FOR J=1 TO 6 \ IF J=4 THEN NEXT J
     ELSE UN=V(J)*32767/10 \ N=J  \ GOSUB 8000 \ NEXT J
     XFLAG=1
     RETURN
     REM
 800 PRINT \ PRINT "----------------------------------------"
     PRINT \ PRINT USING 802,ISTEP,L1,L2,E1(1),U1
 802 FORMAT "SCAN=#### L1=-##.### L2=-##.### E1=-##.### U1=-##.###"
     PRINT \ PRINT USING 804,T1,T7,T10,E2(1),U2
 804 FORMAT "T1=-###.## T7=-###.## T10=-###.## E2=-##.### U2=-##.###"
     PRINT \ I=0 \ PFLAG=0 \ RETURN
 900 N=3 \ UN=SMAG*32767/10 \ GOSUB 8000 \ RETURN
8000 U=INT(UN)
     CALL SUB3(N,U)
     RETURN
9000 UN=.05/10*32767 \ GOSUB 8000 \ RETURN
9500 IF B(57)=0 THEN RETURN
     ELSE S1=B(57) \ S2=B(58) \ RETURN
9999 EN=ERR \ LN=ELN \ CALL SUB99(EN,LN)
     END
```

Table A7.3 Machine Code Patch in BASIC Interpreter for use with the Newbury VDU

Program			Location
TEST	DAC	**	'753
	STA	TEMP	'754
	INA	'1004	'755
	JMP	*-1	'756
	STA	FLAG	'757
	LDA	TEMP	'760
	JMP*	TEST	'761
TEMP	BSZ	1	'762
FLAG	BSZ	1	'763

Table A7.4 On-line BASIC program: Double-Effect Evaporator

```
10  REM  ... PROGRAM FOR ON-LINE M6800 CONTROL OF TEMPERATURES ...
12  REM                  IN A DOUBLE-EFFECT EVAPORATOR
20  REM
25  REM        .   D(0) GOES TO STEAM  VALVE
27  REM            D(1) GOES TO INFLOW  VALVE
35  DIM E(4),Z(4)
37  PRINT " INPUT PROP. CONS. K5 , K7 ";: INPUT K5,K7
39  PRINT " INPUT TIME INTEGRAL T1,T2 ";: INPUT T1,T2
40  PRINT " INPUT SET VALUES  D1 , D3 ";: INPUT D1,D3
42  PRINT " INPUT SET VALUE OF VARIABLES";: INPUT S7,S8
43  PRINT " INPUT CONTROLLED CHANNELS   ";: INPUT M1,M2
44  Z(1)=S7:Z(3)=S8
46  E(1)=0:E(3)=0
50  DIM A(13),B(86),C(7),D(29)
122  PRINT "INPUT X0,X1,Y0,Y1 ";: INPUT X0,X1,Y0,Y1
123  PRINT "INPUT CHANN. NO. TO BE PLOTTED J";: INPUT J
124  PRINT "INPUT CHANN.NO TO BE PLOTTED  K ";: INPUT K
126  PRINT "INPUT CHANN. NO. TO BE PLOTTED L ";: INPUT L
127  PRINT " INPUT CHANN. FOR HISTO. PLOTTING C1&C2 ";: INPUT C1,C2
128  PRINT "INPUT  SCALING FACTOR FOR C1 & C2 ";: INPUT F1
130  PRINT "INPUT SCAN INTERVAL  ";: INPUT S3
140  PRINT "INPUT MULTIPLES OF MICRO SCAN ";: INPUT S2
145  PRINT " NO. OF SCANS ..";: INPUT S1
162  GOSUB 700: CALL (5,1,C(0))
163  CALL (5,2,C(0)): GOSUB 700
164  N=1
165  B(C1)=D1:B(C2)=D3
200  S=S2*S3
265  X=0:A(1)=S2:A(2)=S1:A(12)=1
270  CALL (1,A(0),B(0))
280  FOR I=0 TO 9
290  B(I)=B(I)*128/1024
300  NEXT I
310  B(10)=B(10)*.245E-01/716.8+.105E-01
320  B(11)=B(11)*.5E-01/1024+.222E-01
330  B(12)=B(12)*31.5/1024-2.52
340  B(14)=B(14)*52.636/1024+1.4
370  B(15)=B(15)*19.127/1024
400  B(37)=B(55)/(3270*S3)+.394E-02
405  GOSUB 900
410  REM   TO ENABLE PLOTTING B(75),B(76) IS REMOVED
415  REM    ..D(0) IN M6800 CORRESPONDS TO B(57) IN H316 & SO ON
420  REM     M6800 CONTROL OUTPUTS
425  B(75)=B(57)/.32E05:B(76)=B(58)/.32E05
502  IF N>1 THEN 512
504  C(5)=0:C(6)=B(J):C(0)=2
506  CALL (5,3,C(0))
507  L1=B(C1)*F1:L3=B(C2)*F1
508  Y1=B(J):V1=B(K):W1=B(L)
510  GOTO 513
512  GOSUB 540
513  N=N+1
515  E(1)=B(2):E(3)=B(4)
520  CALL (2)
522  CALL (5,6,C(0))
523  C(7)=24: CALL (5,5,C(0))
524  END
540  X=X+S
```

```
 541  L2=B(C1)*F1:L4=B(C2)*F1
545  Y2=B(J):V2=B(K):W2=B(L)
574  C(5)=X:C(6)=Y2:C(0)=3
546  GOSUB 1000
548  C(5)=X-S:C(6)=V1:C(0)=2
550  GOSUB 1000
552  C(5)=X:C(6)=V2:C(0)=3
554  CALL (5,3,C(0))
556  C(5)=X-S:C(6)=W1:C(0)=2
558  GOSUB 1000
560  C(5)=X:C(6)=W2:C(0)=3
562  GOSUB 1000
564  C(5)=X-S:C(6)=L1:C(0)=2: GOSUB 1000
566  C(5)=X:C(0)=3: GOSUB 1000
567  C(6)=L2: GOSUB 1000
568  C(5)=X-S:C(6)=L3:C(0)=2: GOSUB 1000
570  C(5)=X:C(0)=3: GOSUB 1000
572  C(6)=L4: GOSUB 1000
574  C(5)=X:C(6)=Y2:C(0)=2: GOSUB 1000
575  Y1=Y2:V1=V2:W1=W2
576  L1=L2:L3=L4
580  RETURN
600  C(7)=29: CALL (5,5,C(0)):C(1)=X0:C(3)=Y0
620  C(2)=X1:C(4)=Y1
630  RETURN
700  C(5)=X0:C(6)=Y0:C(0)=2: GOSUB 1000
710  C(5)=X1:C(6)=Y0:C(0)=3: GOSUB 1000
715  Y1=Y1+Y0
720  C(5)=X1:C(6)=Y1: GOSUB 1000
730  C(5)=X0:C(6)=Y1: GOSUB 1000
740  C(5)=X0:C(6)=Y0: GOSUB 1000
750  RETURN
900  E(2)=S7-B(M1):E(4)=S8-B(M2)
915  K6=K5/11:K8=K7/12
917  D2=K5*(E(2)-E(1))+K6*S*E(2)
920  D1=D1+D2
922  D4=K7*(E(4)-E(3))+K8*S*E(4)
924  D3=D3-D4
925  IF D1<=0 THEN D1=0
926  IF D3<=0 THEN D3=0
930  D(0)=D1*.32767E05/.4E-01:D(1)=D3*.32767E05/.8E-01
932  D(2)=100
937  IF ABS(D(0))>.327E05 THEN 950
938  IF ABS(D(1))>.327E05 THEN 950
944  CALL (4,0,3,D(0))
950  RETURN
1000  CALL (5,3,C(0)): RETURN
```

Table A7.5 On-line SD BASIC program: Double-Effect Evaporat

```
    PROGRAM ORIGIN :4000
    DATA ORIGIN    :5000
    REM ... PROGRAM FOR M6800 CONTROL OF TEMPERATURES
    REM           IN A DOUBLE-EFFECT EVAPORATOR
    REM
    REM              2 P+I CONTROL LOOPS
    REM      RPV1,RPV2 ARE INITIAL VALVE SETTINGS
    REM
    DIM A(11),B(86),D(29),E1(2),E2(2),TIMER/06/,PFLAG
    DIM KPS,KPF,KIS,KIF,AVS,AVF,RPV1,RPV2,TIS,TIF,L/0/
    DIM S,SV1,SV2,I,F,L1,S1,EN,LN,U,N,M
    REM
    ON ERROR GOTO 9999
    CALL SUB0
    PRINT "INPUT CONTROLLER PARAMETERS !"
    PRINT \ INPUT "KPS & TIS FOR STEAM " KPS,TIS
    INPUT "KPF & TIF FOR FEED " KPF,TIF
    INPUT "STEAM & FEED VALVE AVERAGE POS. AVS,AVF " AVS,AVF
    INPUT "SET POINTS SV1 & SV2 " SV1,SV2
    INPUT "A(0),PRINT FLAG " S,PFLAG
    FOR I=0 TO 11 \ A(I)=0 \ NEXT I
    INPUT "F,L1,S1,A(1) " F,L1,S1,A(1)
    REM
    FOR I=0 TO 86 \ B(I)=0 \ NEXT I
    FOR I=0 TO 29 \ D(I)=0 \ NEXT I
    REM
    A(0)=S \  A(2)=S1 \ A(3)=F \ A(4)=L1
    A(5)=33
    E1(1)=0.0 \ E2(1)=0.0
    KIS=KPS/TIS \ KIF=KPF/TIF
    REM
 10 CALL SUB1(A(0),B(0))
    IF B(59)=0 THEN 15
    SV1=B(57)*.04/32767 \ SV2=B(58)*.08/32767
 15 REM
    L=L+1 \ IF L<=6 THEN GOSUB 1000
    IF A(1)=1 THEN 18
    ELSE GOSUB 2000
 18 IF L=12 THEN L=0
    E1(1)=E1(2) \ IF A(1)=9 THEN  E2(1)=E2(2)
    REM
    IF PFLAG=0 THEN 20
    PRINT USING 901,SV1,SV2,AVS,AVF
901 FORMAT "-##.###  -##.###  -######.##  -######.##"
    B(35)=B(54)/50.0 \ REM -- CONVERT TO SECS.
    PRINT USING 902,B(57),B(58),B(37),B(55),B(56)
902 FORMAT "-#####.# -#####.# -##.### -#### -###"
    PRINT D(0),D(1)
    PRINT
 20 M=2
```

```
      CALL SUB4(M,D(0))
      CALL SUB2
      GOTO 10
      REM
 1000 B(36)=B(10)*.0245/716.8 + .105E-01
      E1(2)=SV1-B(36) \ IF ABS( E1(2) ) < 3.E-04 THEN RETURN
      RPV1=KPS*(E1(2)-E1(1)) + KIS*S*E1(2)
      AVS=AVS+RPV1 \ IF AVS>32.E3 THEN AVS=32.E3
      IF AVS<0.0 THEN AVS=0.0
      D(0)=INT(AVS)
      U=INT(AVS) \ N=1
   30 CALL SUB3(N,U) \ RETURN
      REM
 2000 B(37)=( B(55)/(3270*S) + .00394 )
      E2(2)=SV2-B(37) \ IF ABS( E2(2) ) < 3.0E-03 THEN RETURN
      RPV2=KPF*(E2(2)-E2(1)) + KIF*S*E2(2)
      AVF=AVF-RPV2 \ IF AVF>32.E3 THEN AVF=32.E3
      IF AVF<0.0 THEN AVF=0.0
      D(1)=INT(AVF)
      U=INT(AVF) \ N=2
   40 CALL SUB3(N,U) \ RETURN
      REM
 9999 EN=ERR \ LN=ELN
      CALL SUB99(EN,LN)
      END
#
```

REFERENCES

1. Knuth, D.E. "Ancient Babylonian Algorithms, Communications of the ACM, Vol. 15, p. 671, July (1971).
2. Wilkes, M.V., Babbage as a computer pioneer, Historia Mathematica, Vol. 14 (1977).
3. Randell, B. (ed.), The Origins of Digital Computers, Selected Papers, Springer-Verlag, New York (1973).
4. Mollenhoff, G.G. "John V. Atanasoff, DP Pioneer" Computer-world, March 13, 20, 27 (1974).
5. Tremblay, J.P. and Bunt, R.B. An Introduction to Computer Science - An Algorithmic Approach, McGraw-Hill Book Co. (1981).
6. Huskey, H.D., and Huskey, V.R., Chronology of Computing Devices, IEEE Trans. on Computers, Vol. C-25, p. 1190, Dec. (1976).
7. Push-Button Era Inaugurated, Oil Gas. J., 57, p. 78, April 6 (1959).
8. Lemay, L.P., On-Line Computer Control of a Chemical Plant, Proc. Computing and Data Process. Soc. Canada, pp. 258-277 (1962).
9. Guisti, A.L., Otto, R.E. and T.J. Williams, Direct Digital Computer Control, Control Eng. 9(6), pp. 104-108 (1962).
10. Morello, V.S., R.H. Foy and K.A. Otto., Computer Applications Symposium, Chicago, Illinois (1962).
11. Savas, E.S., Computer Control of Industrial Processes, McGraw-Hill, Inc. (1965).
12. Microprocessor Applications, Cases and Observations, H.M.S.O. London (1980).

13. Hilburn, J.L. and P.M. Julich, Microcomputers/Microprocessors: Hardware, Software and Applications, Prentice-Hall, Inc., Englewood Cliffs, N.Y. (1976).

14. Callacher, J., Industrial Data Acquisition and Control Systems, Microprocessors and Microsystems, Vol. 3, No. 5, p. 210, June (1970).

15. Depledge, P.G. "A Review of Available Microprocessors, I.J.E.E.E. 16, No. 2, pp. 114-123 (1979).

16. 'What's New in Computing' Magazine, p. 3, April (1982).

17. Cushman, R.H., Eighth Annual Microprocessor/Microcomputer Chip Directory, EDN, Vol. 26, No. 22, pp. 100-200, Nov. (1981).

18. Smith, M.F., Comparative Software Analysis of the MC6809, Microprocessors and Microsystems, Vol. 5, No. 9, Nov. (1981).

19. Depledge, J.P., Recent Developments in the Design of Microcomputer System Components, I.J.E.E.E., Vol. 19, p. 197-122, Manc. U.P. (1982).

20. Office Systems, Sept. (1982).

21. Computer Weekly, p. 12, Dec. (1982).

22. What Micro? No. 1, November/December (1982).

23. Beyers, J.W., et al., A 32-bit VLSI CPU chip, IEEE Journal of Solid State Circuits SC-16, No. 5, pp. 537-545, October (1981).

24. Best, D.W., et al., An Advanced-Architecture CMOS/SOS Microprocessor, J. of I.E.E.E. Micro, p. 11, Aug. (1982).

25. Stanton, B.D., Reduced Problems in New Control System Design, Hydrocarbon Processing, p. 67, Aug. (1982).

26. Kane, L.A. What's wrong with Process Control?, Hydrocarbon Processing, Vol. 8, p. 61, Aug. (1982).

27. Takahashi, Y., et al., Simple Discrete Control of Industrial Processes, Trans. ASME, Series G, J. Dyn. Sys. Meas. and Contr., Vol. 97, No. 4, p. 354, Dec. (1975).

28. Cummings, G.A. and G.S. Miller, Applications of a Bipolar Microprocessor Chip Set to Control Systems, Proc. Joint Automatic Control Conf. p. 27 (1977).

29. Auslander, D.M., et al., Process Control Experience and a Self-Tuning method for a Discrete-Time, finite-time setting controller/observer, Trans. ASME, J.Dyn.Sys.Meas. and Control, Sept. (1977).

30. McMahon, T.K., Distributed Digital Control - Control Technology Breakthrough, Chem. Eng., p. 117, Oct. 19 (1979).

31. Auslander, D.M., et al., Direct Digital Process Control: Practice and Algorithms for Microprocessor Application, Proc. of the IEEE, Vol. 66, No. 2, p. 199, Feb. (1978).

32. Bond, A., Distributed microprocessors replace the computer in total digital control system, Process Engineering, p. 64, Jan. (1976).

33. Taylor, R.A., A Distributed Microprocessor Control Philosophy, ISA Annual Conference 1977, p. 131.

34. Buchner, M.R. and I. Lefkowitz, Distributed Computer Control for Industrial Process Systems: Characteristics, Attributes, and An Experimental Facility, Control Systems Magazine IEEE, p. 8, Mar. (1982).

35. Williams, D.L., Microcomputers enhance computer control of chemical plants, Chemical Engineering, p. 95, July 18 (1977).

36. Process Control systems for the 80s, Processing, p. 33, January (1980).

37. Sheppard, P.F., Functional and Geographical Distributed Control of Industrial Processes, Fourth International Conference on Trends in On-Line Computer Control Systems, IEE, 5-8 April (1982).

38. Steelman, D.M., Distributed Cement Plant Control - The Intelligent Approach, IEEE Trans. on Industry Applications, Vol. IA-18, No. 2, p. 192, Mar/Apr (1982).

39. Fisher, K.J., Distributed Process Control System for an Iron Ore Processing Plant, op. cit. p. 199.

40. Sansom, P., Concepts and Trends in Industrial Schemes, Chemistry and Industry, p. 39, Sept.(1982).

41. Adam Osborne and Associates, In., An Introduction to Microcomputers, (1975).

42. Hilburn, J.L. and P.M. Julich, Microcomputers/Microprocessors: Hardware, Software, and Applications, Prentice-Hall, Inc., Englewood Cliffs, N.J. (1976).

43. Leventhal, L.A., Introduction to Microprocessors: Software, Hardware and Programming - Englewood Cliffs, London, Prentice-Hall (1978).

44. Ward, A.R., LSI Microprocessors and Microcomputers. A bibliography, Computer, p. 35, July (1974).

45. Nichols, A.J., An Overview of Microprocessor Applications, Proc. of the IEEE, p. 951, June (1976).

46. Biewer, M., Don't Confuse Microprocessors with computers, Systems, p. 27, Dec/Jan. (1975).

47. Capece, R.P., and J.G. Posa, Microprocessors and Microcomputers: One-chip Controllers to High-end Systems, McGraw-Hill Publications Co., N.Y., (1981).

48. Randle, W.C., and N. Kerth, Microprocessors in Instrumentation, Proc. of the IEEE, Vol. 66, No. 2, p. 172, Feb. (1978).

49. Leventhal, L.A., Microprocessors in Aerospace Applications, Simulation, p. 111, April (1978).

50. Klig, V., Biomedical Applications of Microprocessors, Proc. of the IEEE, Vol. 66, No. 2, p. 151, Feb. (1978).

51. Weissberger, A., Microprocessors in the Processing Plant, IEEE Trans. on Ind. Electr. and Cont. Inst., Vol. IECI-22, No. 3, p. 354, Aug. (1976).

52. Rees, V.J., Digital Filter Design - Programming a Microprocessor to act as a digital filter, Wireless World, p. 47, Oct. (1976).

53. Ahmed, N. and J.P. Jayapalan, On Digital Filter Implementation via Microprocessors, IEEE Trans. on Ind. Electr. and Contr. Instru., Vol. IECI-23, No. 3, p. 249, Aug. (1976).

54. Allen, J. and A.G.J. Holt, Microprocessor Implementation of a Simple, Low-Pass Filter, Int. J. Elect. Eng. Educ., Vol. 16, pp. 191-209, Manchester U.P., (1979).

55. Edwards, G., Digital Filters can Simplify Signal Processing, Electronic Eng., p. 53, June (1976).

56. Lin, B., Effect of finite wordlength on the Accuracy of Digital Filters - A Review, IEEE Trans. on Circuit Theory, CT-18 No. 6, pp. 670-677, Nov. (1971).

57. Bibbero, R.J., Microprocessors in Instruments and Control, John Wiley and Sons, N.Y. (1977).

58. McKay, C.W. and C. Gross, A Microprocessor-based Flow Monitoring System, J. of ISA AC, p. 821 (1976).

59. Beveridge, G.S.G., Hill, R.G. and M.R. Aguilar, Interactive Computer Flowsheeting on a Microcomputer, Annual Research Meeting, Inst. I. Chem. E., 6th-8th April, London (1982).

60. Mann, R., Stavridis, Maduka, V. and M. Abdul Ahad, Some practical "apple"-ications of a microcomputer, op.cit.

61. Sucksmith, I., Microcomputer Systems for Process Design, Chemical Eng., p. 118, January 10 (1983).

62. Tao, T.F., Yehoshua, D.B. and R. Martinez, Applications of Microprocessors in Control Problems, Proc. 1977 Joint Automatic Control Conf., pp. 8-16.

63. Reed, M. and H.W. Megler, A Microprocessor-based Control System, DEEE Trans. Ind. Electron. Cont. Instrum., Vol. IECI-24, No. 3, pp. 253-257 (1977).

64. Farrar, F.A. and R.S. Aidens, Microprocessor Requirements for Implementing Modern Control Logic, IEEE Trans. Automa. Contr. Vol. AC-25, No. 3, pp. 461-468 (1980).

65. Borisson, U. and R. Syding, Self-tuning of an One-crusher, Automatica, Vol. 12, pp. 1-7 (1976).

66. Sastry, V.A., D.E. Seborg and R.K. Woods, An Application of a Self-tuning Regulator to a Binary Distillation Column, Proc. JACC, pp. 346-354 (1976).

67. Keviczky, L., J. Hetthesy, M. Higler and J. Korbostori, Self-tuning Control of Cement raw material handling, Automatica, Vol. 14, pp. 525-532 (1978).

68. Cegrell, T. and T. Hedqvist, Successful Adaptive Control of Paper Machines, Automatica, Vol. 11, pp. 53-59 (1975).

69. Clarke, D.W., Cope, S.N. and P.J. Gawthrop, Feasibility Study of the Application of Microprocessors to Self-tuning Regulators, OUEL Report, 1137/75, 110p.

70. Astrom, K.J. and B. Wiffenmark, On Self-Tuning Regulators, Automatica, Vol. 9, pp. 185-199 (1973).

71. Clarke, D.W. and P.J. Frost, Control BASIC for Microcomputers, IEE Conf. in Trends in On-line Computer Control, Sheffield (1979).

72. Sheirah, M.A., Malik, O.P. and G.S. Hope, Self-tuning Microprocessor Universal Controller, IEEE Trans. on Ind. Electon., Vol. IE-29, No. 1, Feb. (1982).

73. Baradello, C.S., Design of Adaptive Algorithms for Microcomputer Process Control, Ph.D. Thesis, Carnegie-Mellon University (1978).

74. Jensen, K.M., A Microprocessor-based Adaptive Observer for Control of Distributed Parameter Systems, Ph.D. Thesis, Univ. of Wyoming (1979)

75. Stinson, S.C., Process control equipment ignores recession, C & EN, p. 15, Dec. 7 (1981).

76. Bailey, S.J., Process Controllers 1975: Tradition Holding off New Technology, Control Engineering, p. 36, Oct. (1975).

77. Process Control Survey 1982, Process Engineering, Sept. (1982).

78. Barnes, G.F., Single-loop Microprocessor Controllers, Inst. Tech., p. 47, Dec. (1977).

79. Fraade, D.J., Using a Microprocessor to Solve pH Control Problems, Inst. Tech., p. 61, Nov. (1978).

80. Mitchell, J.W., Microprocessors control chemical addition and Coking unit Cooling, Hydrocarbon Processing, p. 137, Feb. (1979).

81. Langill, A.W. and D.A. Borses, A close look at two microprocessor control applications, Instruments and Control Systems, p. 43, April (1977).

82. Carter, J.W., The Problems of using microprocessors, Measurement and Control, Vol. 11, p. 45, Feb. (1978).

83. Crutchley, W., Microcomputer Control Course, Instruments and Control Systems, Part I, p. 63, Jan.; Part II, p. 43, Feb.; Part III, p. 49, March; Part IV, p. 67, April (1979).

84. Computer Weekly, Dec. 16/23 (1982).

85. Dickey, T.E., Evaluation of Microprocessors for Process Control Applications, Ph.D. Thesis, Carnegie-Mellon University (1981).

86. American Standards Association, FORTRAN-proposed standard, 1969.

87. DAP-16 & DAP-16MOD2 Honeywell Series 16 Assembly Languages Document No. 41286384-000-01, Honeywell Inc. (1971).

88. Motorola, M6800 Programming Reference Manual, Phoenix, Arizona (1976).

89. M6800 Microcomputer System Design Data, Motorola SPD, Inc.

90. Arteaga, P., M.Sc. Research Project, University of Aston-in-Birmingham (1979).

91. Varelas, T., M.Sc. Research Project, University of Aston-in-Birmingham (1981).

92. The SD BASIC Compiler Manual, Software Dynamics Inc. (1976).

93. Gay, B. and A. Jordan, Private Communication.

94. Shafii, A.F.B., User Manual for the Linked H316-M6800 twin-processor system, February (1983).

95. Kalman, R.E., A new approach to linear filtering and prediction problems, Trans. ASME, J. Basic Eng., Vol. 83, pp. 95-107, Dec. (1961).

96. Kalman, R.E. and R.S. Bucy, New Results in linear filtering and prediction theory, Trans. ASME, J. Basic Eng., Vol. 83, p. 95-107, Dec. (1961).

97. Doob, J.L., Stochastic Processrs, John Wiley and Sons Inc., N.Y., (1955).

98. Bryson, A.E. and Y.C. Ho, Applied Optimal Control, Hemisphere Publishing Corp. (1975).

99. Jazwinski, A.H., Stochastic Processes and Filtering Theory, Academic Press, London (1970).

100. Astrom, K.J., Introduction to Stochastic Control Theory, N.Y., Academic Press (1967).

101. Kuo, B.C., Discrete Data Control Systems, Prentice-Hall, Inc. Englewood Cliffs, N.Y. (1970).

102. Mendel, J.M., Computational Requirements for a Discrete Kalman Filter, IEEE Trans. on Auto. Contr., Vol. 4C-16, No. 6, p. 748, Dec. (1971).

103. Noton, A.R.M., Two-level form of the Kalman Filter, IEEE Trans. on Auto. Contr., Vol. AC-16, No. 2, p. 128, April (1971).

104. Aoki, M., Optimisation of Stochastic Systems, Academic Press, N.Y. (1967).

105. Noton, A.R.M. and P. Choquett, Proc. IFAC/IFIP Symp., Toronto (1968).

106. Choquette, P., A.R.M. Noton and C.A.G. Watson, Proc. IEEE, Vol. 58, No. 1, p. 10 (1970).

107. Coggan, G.C. and A.R.M. Noton, Discrete-Time Sequential State and Parameter Estimation in Chemical Engineering, Trans. Inst. Chem. Eng., Vol. 48 (1970).

108. Coggan, G.C. and J.A. Wilson, Symp. on On-Line Computer Methods Relevant to Chemical Engineering, Trans. Inst. Chem. Engrs., Nottingham, England.

109. Goldman, S.F. and R.W.H. Sargent, Chem. Eng. Sci., Vol. 26, pp. 1535-1542 (1971).

110. Coggan, G.C. and J.A. Wilson, Computer Journal, Vol. 14, pp. 61-64 (1971).

111. Webb, R.N., Development of an Adaptive Kalman Filter for Estimation in Chemical Plants, Ph.D. Thesis, University of Aston-in-Birmingham (1977).

112. Fisher, D.G. and D.E. Seborg, Multivariable Computer Control - A Case Study, North-Holland/American-Elsevier, (1976).

113. Payne, S.G., The Application of On-Line Estimation to a Double-Effect Evaporator, Ph.D. Thesis, University of Aston-in-Birmingham (1974).

114. Coleby, J.M., The Application of Digital Filtering Methods to State and Parameter Estimation in Process Plant, Ph.D. Thesis, University of Aston-in-Birmingham, (1974).

115. Dahlqvist, S.A., Control of Top and Bottom Product Compositions in a Pilot Distillation Control, I.Chem.E. Symp. Series, No. 56 (1979).

116. Joseph, B. and C.B. Brosilow, Inferential Control of Processes, AIChE Journal, Vol. 24, No. 3, p. 485, May (1978).

117. Brosilow, C.B. and M. Tong, The Structure and Dynamics of Inferential Control Systems, Ibid., Vol. 24, No. 3, p. 492 (1978).

118. Joseph, B. and C.B. Brosilow, Construction of Optimal and Sub-Optimal Dynamic Estimators, Ibid. Vol. 24, No. 3, p. 500 (1978).

119. Daie, S., Computer Control of Chemical Plants with Special Reference to Distillation, Ph.D. Thesis, University of Aston-in-Birmingham, (1980).

120. Wick, H.J., On-Line Estimation of Centre Temperatures of Ingots in Soaking Pits, Colloquim on Kalman Filters and their Applications, IEE Computing and Control Division, London, 17th Nov. (1980).

121. Brambilla, A., G. Nardini, G.F. Nencetti and S. Zanelli, Hydrodynamic Behaviour of Distillation Columns: Pressure Drop in Plate Distillation Columns. I.Chem.E. Symposium Series, No. 32 (1969).

122. Bubble Tray Design Manual, AIChemE, New York, (1958).

123. Jordan, A. and B. Gay, Private Communication.

124. Mukesh, D., Dynamics of a Continuous Stirred Tank Reactor for Different Reaction Orders, Ph.D. Thesis, University of Aston-in-Birmingham (1980).

125. Dilfanian, S., Simultaneous Chemical Reaction and Distillation of Formaldehyde, Ph.D. Thesis, University of Aston-in-Birmingham, (1978).

126. Yu, T.K. and J.H. Seinfeld, IEEE Trans. Auto Control, AC-16, p. 495 (1971).

127. Gay, B. and A.F. Shafii, The Development and Use of a Linked Micro- and Minicomputer System for Real-Time Data Acquisition and Process Control, Annual Research Meeting, IChemE, 6th-8th April, London (1982).

128. Nawari, M.O.M., Private Communication.

129. Youla, D.C., J.J. Bongiorno and H.A. Jabr, Modern Wiener-Hopf Design of Optimal Controllers, Part I: The Single-Input-Output Case, IEEE Trans. on Auto. Contr., Vol. AC-21, No. 1, p. 3, Feb. (1976).

130. ibid; Modern Wiener-Hopf Design of Optimal Controllers - Part II: The Multivariable Case, IEEE Trans. on Auto. Contr., Vol. AC-21, p. 373, June (1976).

131. Bishop, R., Basic Microprocessors and the 6800, Hayden Book Co., Inc., Rochelle Park, N.J., (1979).