CANADIAN THESES ON MICROFICHE

THÈSES CANADIENNES SUR MICROFICHE

National Library of Canada
Collections Development Branch

Canadian Theses on
Microfiche Service

Ottawa, Canada
K1A 0N4

Bibliothèque nationale du Canada
Direction du développement des collections

Service des thèses canadiennes
sur microfiche

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED

LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE

Canada

NL 339 (r. 86/09)

DESIGNING AND TESTING OF A MICROCOMPUTER CONTROL SYSTEM

by

Paulo F. Springmann

A thesis
presented to the University of Ottawa
in partial fulfillment of the
requirements for the degree of
Master of Applied Science
in
Chemical Engineering

OTTAWA, Ontario, 1984

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

ABSTRACT

In this work, an Apple II microcomputer together with a data acquisition and control hardware / software package called ISAAC (Integrated System for Automated Acquisition and Control) was used to implement Direct Digital Control (DDC) on a pilot scale process. This process consisted of a tank in which hot and cold water streams were mixed, the controlled variables being the level and the temperature of the water in the tank.

Three control programs were written, two for level-only control and one for multivariable level / temperature control. A position PID control algorithm was incorporated in the first level control program, whereas a velocity PID control algorithm was incorporated in the second one. A number of differences in controller performance were observed between the position and the velocity PID algorithms.

The third program, the multivariable level and temperature control, showed one-directional coupling with the level loop affected by the temperature loop. Coupling was only noticeable when the level loop was loosely tuned in comparison with the temperature loop. A dynamic decoupler was designed and implemented which significantly reduced coupling. Best multivariable control was achieved by tuning the level tightly which almost eliminated coupling.

ACKNOWLEDGEMENTS

The author wishes to express gratitude to Dr David McLean, who directed this research, for his guidance, inspiration, co-operation and his help in the preparation of this thesis.

The author is also indebted to Mr G. Gasperetti, Mr D. Lefebvre and Mr A. Bonaldo for their suggestions and technical help at all times.

NOMENCLATURE

D = deadtime (minutes),

$D(s)$ = Laplace transformed disturbance,

$\underline{d}$ = Laplace transformed vector of disturbances,

$\bar{D}12(s)$ = Laplace transformed output from decoupler (VDC),

DDC = Direct Digital Control,

$E(s)$ = Laplace transformed error signal,

$\underline{e}$ = Laplace transformed vector of error signals,

$F(s)$ = Laplace transformed feedforward control transfer function,

F = outlet flowrate (L/min),

F_c = cold water inlet flowrate (L/min),

F_h = hot water inlet flowrate (L/min),

$G(s)$ = Laplace transformed process transfer function,

$\underline{G}(s)$ = Laplace transformed process transfer function matrix,

$G_c(s)$ = Laplace transformed controller algorithm,

$\underline{G}_c(s)$ = Laplace transformed controller algorithm matrix,

$Gc_m(s)$ = Laplace transformed master controller algorithm,

$Gc_s(s)$ = Laplace transformed slave controller algorithm,

$G_d(s)$ = Laplace transformed disturbance transfer function,

$\underline{G}_d(s)$ = Laplace transformed disturbance transfer function matrix,

$\underline{G}_I(s)$ = Laplace transformed dynamic decoupler matrix,

$\underline{G}_{I_{s.s.}}(s)$ = Laplace transformed steady-state decoupler matrix,

I.S.E. = Integral of the Squared Error,

K_c = controller gain,

K_p = open-loop process gain,

Λ = Bristol array,

L = Vertical distance between the water surface in the tank and valve A (dm),

L_o = Vertical distance between the bottom of the tank and valve A (dm),

M(s) = Laplace transformed controller output (VDC),

$\underline{m}$ = Laplace transformed vector of controller outputs,

M, Ml = Digital output from level controller, which after D/A conversion is represented by V_a (VDC),

MR = level controller output at steady-state,

N, Nl = Digital output from temperature controller, which after D/A conversion is represented by V_b (VDC),

NR = Temperature controller output at steady-state,

PID = Proportional-Integral-Derivative,

S_t = sampling time or sampling period (seconds),

τ_d = rate or derivative time (seconds),

τ_I = reset or integral time (seconds),

τ_p = process time constant (minutes),

T = tank water temperature (^oC),

T_c = cold water inlet temperature (^oC),

T_h = hot water inlet temperature (^oC),

V_a = level controller output after D/A conversion (VDC),

V_b = temp. controller output after D/A conversion (VDC),

VDC = Volts Direct Current,

X(s) = generic Laplace transformed controlled variable,

Y(s) = generic Laplace transformed controlled variable,

$\underline{y}$ = Laplace transformed vector of controlled variables,

$Y^{set}(s)$ = Laplace transformed setpoint and

$\underline{y}^{set}$ = Laplace transformed vector of setpoints.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter I

INTRODUCTION

Until about 20 years ago, energy was cheap and environmental regulations, as well as product specifications, were loose. Satisfactory control could generally be achieved through application of single-input single-output schemes based on linear control theory. For these reasons, traditional mechanical, pneumatic or electric analog controllers were adequate for the majority of cases[1]. However, the situation has changed drastically in recent years. Energy is no longer cheap, environmental emission regulations are strict and product specifications are tight.

Major advances in process control, namely through application of digital computers and the simultaneous development and implementation of more sophisticated computer control theory, have fortunately provided ways to meet these new, more demanding requirements. According to Ginn[2], computer controlling a process typically increases its production capacity by 5%, the utility consumption per unit product is brought down by 10% and the raw material loss is reduced by 20%.

The first applications of digital computers for chemical process control go back to almost 30 years ago[3].

The system was typically constituted of a large centralized main frame computer whose tasks were mainly process data logging, yield calculations and alarm activation. In these early stages, huge and expensive machines with slow execution speed were common. Interfacing with the process was very difficult, and little stress was put on system analysis. As a result, only limited success was achieved.

With the revolution in the microelectronics industry that began in the late sixties, components for computer control systems have become more varied, more powerful, miniaturized and less expensive. These advances have made computer control the standard in the chemical processing industries.

Digital computers can be classified in a descendent order of complexity as: large main-frames, minicomputers, microcomputers and finally microprocessors. At the bottom of the scale, the microprocessor is essentially a Central Processing Unit (CPU). Microprocessors are programmed using very low level languages, normally loaded in non-volatile memory space, and have an enormous potential for dedicated tasks. A microcomputer is a microprocessor to which has been added input/output interfaces and additional memory. Microcomputer-based control systems are used in small systems or as satellites in distributed systems. They seldom handle more than 40 control loops. Minicomputers have typically 10 times the computing power of microcomputers, and are there-

fore used mainly to supervise several microcomputers, microprocessors or analog controllers. Large main-frame systems are now inconvenient and uneconomical for control purposes.

Because of its ability for accepting high-level languages, system expandability and low price, the microcomputer is especially useful for a variety of process control applications. It is being extensively used in both industrial and academic environments, for industrial process control and training purposes [4,6]. As an example of industrial process control, it is worth mentioning the micro- and minicomputer control of the water distribution and wastewater treatment systems of the city of Philadelphia[28]. For training purposes, a microcomputer controlled pilot scale process involving the heating of a continuous stream of water using steam was used by Exxon Oil Company to train their instrument and process personnel[3]. The setup was used to demonstrate feedback control concepts. A similar setup was used by the University of Louisville to demonstrate more complex control concepts[3].

The primary objective of this project was to design and implement a microcomputer-based control system which could be used in an undergraduate process control lab. Our set-up consisted of a tank in which hot and cold water streams were mixed. The controlled variables were the level and temperature of the water in the tank. The manipulated variables were the cold water inlet flowrate and the mixture

outlet flowrate. Control was carried out using an Apple II Plus microcomputer together with an ISAAC (Integrated System for Automated Acquisiton and Control) model 91A data acquisition and control unit.

Our secondary objective was to evaluate various strategies for dealing with interacting control loops. The traditional procedure of tuning one loop loosely and the other tightly in order to minimize interaction was compared to a decoupled control scheme. Decoupling is one of the more complex or advanced control schemes whose implementation became possible only with the advent of computer control.

To accomplish the objectives above, we first designed and implemented a single-input single-output (SISO) level control. As a second step we designed and implemented simultaneous level and temperature control, and as a last step, dynamic decoupling was added to the multivariable level / temperature control system.

Chapter II

DIGITAL COMPUTER CONTROL

Digital computing applications can be divided into two major areas, namely batch computing and on-line computing [3,7]. In batch computing, the program and the data are collected and grouped together before processing is carried out. During processing, only internal interrupts generated by the Central Processing Unit (CPU) take place. Batch computing appeared with the very first computers, and is still being extensively used especially for business applications (e.g. payroll processing). On-line computing involves internal (from the CPU) as well as external interrupts that arrive from the "process" that is being controlled. On-line computing constitutes most of the current computer applications (e.g. air traffic control, industrial process control, interactive computing).

On-line computing can be further broken down into 3 distinct areas:

1. Clock-based computing: Through a built-in clock the computer links its internal operations to real time (seconds, minutes, etc.).
2. Sensor-based computing: The computer links its internal operations to the condition of the process through analog or digital sensors.

3. Interactive computing: The computer communicates with other computers, users, processes, etc.

The great majority of on-line computer systems can be classified as belonging simultaneously to two or three of the areas above. The control of chemical processes involves both clock and sensor-based on-line computing in what is commonly called real-time computing[3,7].

2.1 REAL-TIME COMPUTER CONTROL

Real-time computing deals with the physical connection of a computer to a process using analog-to-digital (A/D), digital-to-analog (D/A) and digital data interfaces so that communication is established. This communication can be unidirectional as in data logging or bidirectional as in process control. These operations are synchronized with the passage of real physical time.

In the early stages of computer applications to process control, the trend was towards a centralized large main-frame computer, normally installed in the control room. With the new developments in electronics, and the popularity of computer networks in the data processing industry, distributed control systems are replacing the centralized system[2]. The advantages of distributed control are obvious. Less wiring is required, since the controller is closer to the process. The system is safer, since failure in one unit does not affect the others. Because of its modular struc-

ture, the task of computerizing a plant can be carried out in stages.

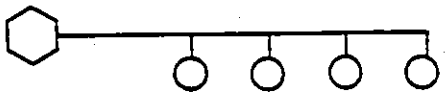
Computers or microprocessors of a distributed system (also known as satellites) are interconnected and form a network. There is normally a computer (Host) which is assigned the task of controlling the flow of information among the satellites, as well as the overall supervision of the system. Computer networks can have several configurations [3], which can be seen in Figure 2.1 .

The present work involves a single satellite, which may or may not be part of a larger system.

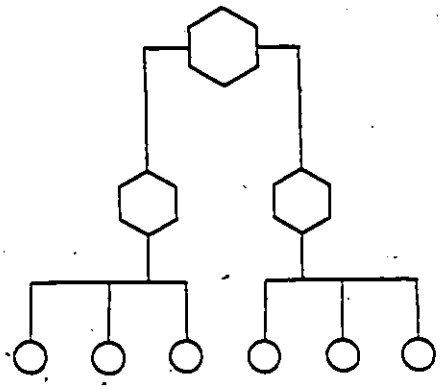
There are two basic methods of implementing digital computer control[3,4] :

1. DIRECT DIGITAL CONTROL (DDC), in which the computer is used as a direct substitute for an analog controller within a loop, performing the same tasks as its analog counterpart. DDC is normally implemented with microprocessors, one for each loop.
2. SUPERVISORY CONTROL, in which the computer supervises several controllers which can be analog or microprocessor-based, by controlling their set-points based on some predetermined control strategy. These various loops can constitute a complete unit operation. In this work the microcomputer was implemented primarily in a DDC mode.

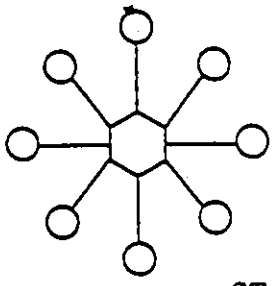
POINT-TO-POINT



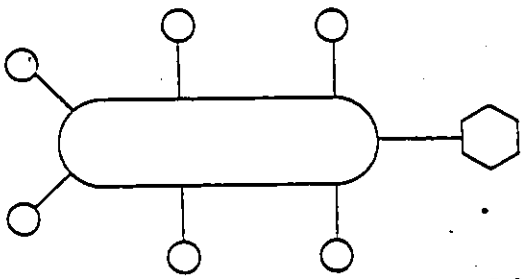
DATA HIGHWAY



HIERARCHICAL



STAR



LOOP

FIGURE 2.1: Computer Network Configurations.

⬡ - Host

○ - Satellite

2.2 COMPONENTS OF A REAL-TIME SYSTEM

Real-time systems are constituted of a process, a real-time computer, and computer- and process-related peripherals. The process is very general. It can be a chemical reactor, the traffic on a railway system or even the launching of a rocket. As far as the real-time computer is concerned, no major restrictions apply. The only requirement is that the computer be able of being effectively interfaced with the process to be controlled. Computer-related peripherals allow interaction of the user with the system. Among them are teletypes, CRT's, on-line printers, etc. Process-related peripherals are specialized devices employed to connect the computer to the process. The most often used are analog-to-digital (A/D) and digital-to-analog (D/A) converters, digital interfaces and real-time clocks. Of particular importance for us are the D/A converter, the A/D converter and the real-time clock.

2.2.1 Real-time Clock

The real-time clock is actually a simple interval timer[7]. Its circuitry uses as time reference the frequency of the AC line (60 Hz). By adding a counter to the interval timer, long time intervals can be measured with precision. The timer can be also converted into a clock if the time is reset to zero each 24 hours. Other real-time systems have both a timer and a real-time clock in independent circuitries[10].

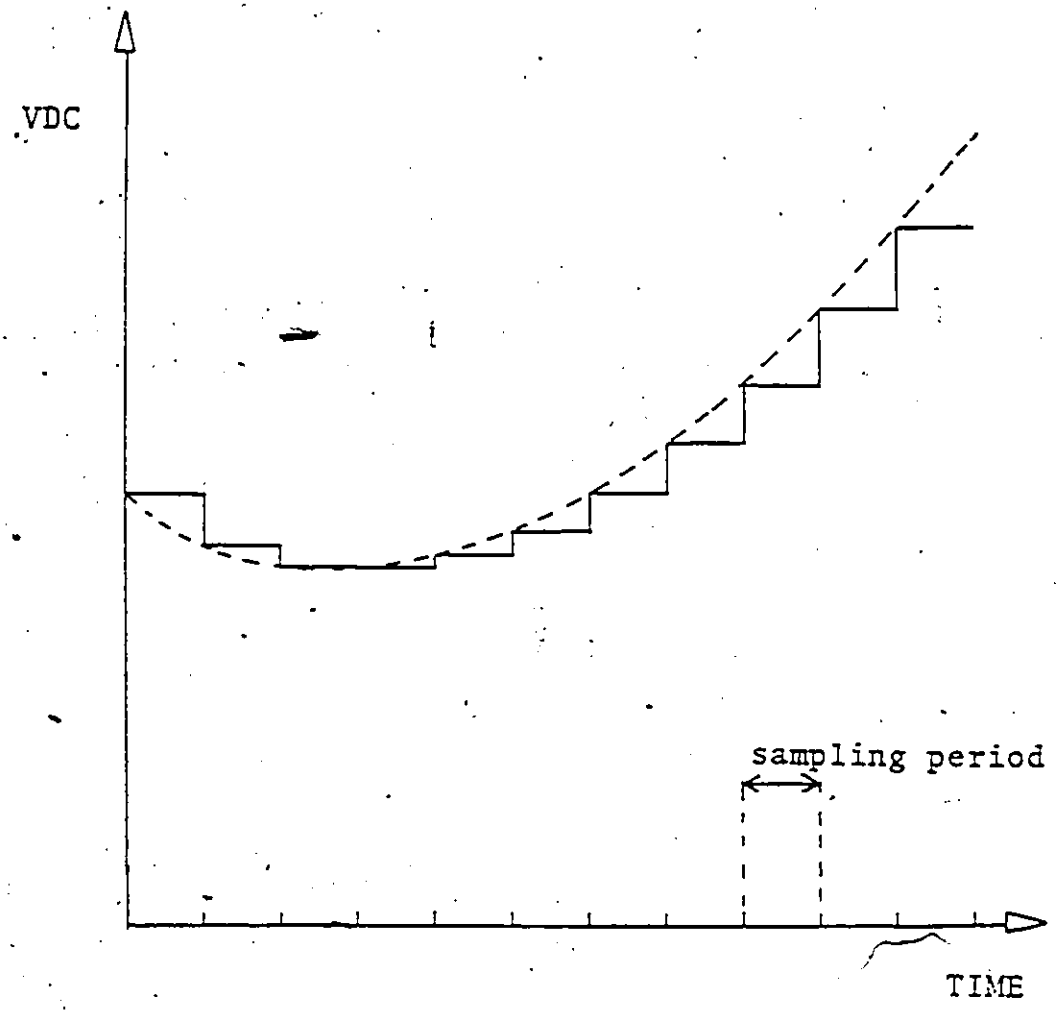
In this case the clock has hardware that resembles common digital watches, where the clock signal is derived from a continuously running pulse generator. A backup battery allows the clock to keep time even if the system power goes off.

2.2.2 A/D converters

Analog continuous signals are not directly readable by computers. A/D converters transform analog quantities (particularly Volts D.C.) into binary quantities, readable by the computer, whose magnitudes are proportional to those of the original analog signals.

Analog signals are converted only at discrete time intervals. Between two consecutive measures, the SAMPLE AND HOLD unit keeps the voltage signal equal to the last measure. This originates a step-shaped curve which approximates the continuous function and is equal to it only at the sampling times. Figure 2.2 shows the step-shaped approximation of a discretized continuous signal.

After the sample and hold unit, the signal is converted into digital quantities. The resolution of an A/D converter is given as "number of bits". For example, a 12-bit conversion resolution means that the full analog voltage range can be approximated by $2^{12} = 4096$ discrete numerical values. Our A/D converter has a 12-bit resolution[10]. Conversion times vary from a few milliseconds



------ original continuous signal and

——— discretized signal.

Figure 2.2 : Discretization of a continuous signal.

down to a few microseconds[7] . In our system, A/D conversion takes 25 microseconds.

A/D converters are normally very expensive, the price depending on conversion time and resolution. Often, for the sake of economy, many analog channels may share a single A/D converter, so that the channels are accessed sequentially. This sequencing is performed by a device called a MULTIPLEXER or MUX for short. For example, our equipment has only one A/D converter [10] , but a built-in MUX allows access to 16 different analog input channels.

2.2.3 D/A Converters

D/A converters perform the inverse operation of the A/D converters. They convert digital quantities into voltage signals. The output of a D/A converter is step-shaped like the output of a sample and hold unit. Compared to the A/D, the D/A converter's circuitry is much simpler and therefore cheaper. Multiplexing is not economically justified, and each analog output channel has its own dedicated D/A converter. This is the case for our system, in which the four analog output channels have their own converters. Resolution is evaluated according to the same criterion used for A/D converters, based on number of bits, and again our system has a 12-bit resolution. Conversion times can be 10 to 100 times faster than for the A/D conversion [7] .

2.3 CONTROL-ORIENTED COMPUTER LANGUAGES

So far, our attention has been directed towards real-time system hardware. It must be pointed out, however, that system software development is equally important, and its price can exceed the price of the hardware by up to a factor of 3 [9]. The development of application software for specific real-time computer control systems is expensive and time-consuming, and also requires a skilled programmer. To circumvent these problems, control-oriented languages have been developed and are now commonly offered along with the hardware and its standard support software. Control-oriented languages handle the most common real-time computer control operations like data input and output, report generation, etc. Some of these languages are so specific for process control that they are called FILL-IN-THE-FORM PACKAGES, because the process engineer is only left the task of adding the necessary process parameters to the already written control program.

Control-oriented languages tend to decrease the cost of the application software, because the basic set of computer instructions can be sold to several different customers which therefore share the cost of the software development. Also, fewer programming skills are required from the process engineer.

The system used for our experiments utilizes a DATA ACQUISITION AND CONTROL software package [11] called Lab-

Soft, which extremely simplifies the programming. Control - oriented languages leave to the process engineer only the task of writing down the suitable control algorithms.

2.4 DIGITAL COMPUTER CONTROL ALGORITHMS

Feedback control is the simplest automatic process control technique that can be implemented. It is typical of the vast majority of control technology used in industrial application today, its block diagram being shown in Figure 2.3 . Feedback control consists in comparing the measured value of the controlled variable with its target or set-point. If a deviation or error exists between the measured and the desired values, an error signal is generated and sent to the controller. The controller, based on a control algorithm, calculates the necessary corrective action to be applied to the final control element, which is usually a control valve.

The most traditional and well known control algorithm is the conventional three-mode PROPORTIONAL - INTEGRAL - DERIVATIVE (PID) algorithm, or one of its simpler versions, PROPORTIONAL - INTEGRAL (PI) or PROPORTIONAL (P) only. PID-based algorithms are easily implemented in analog controllers, yielding satisfactory performance for single feedback loops[8] .

Non-conventional control strategies (e.g. computed variables, nonlinear control, selective control loops, anti-

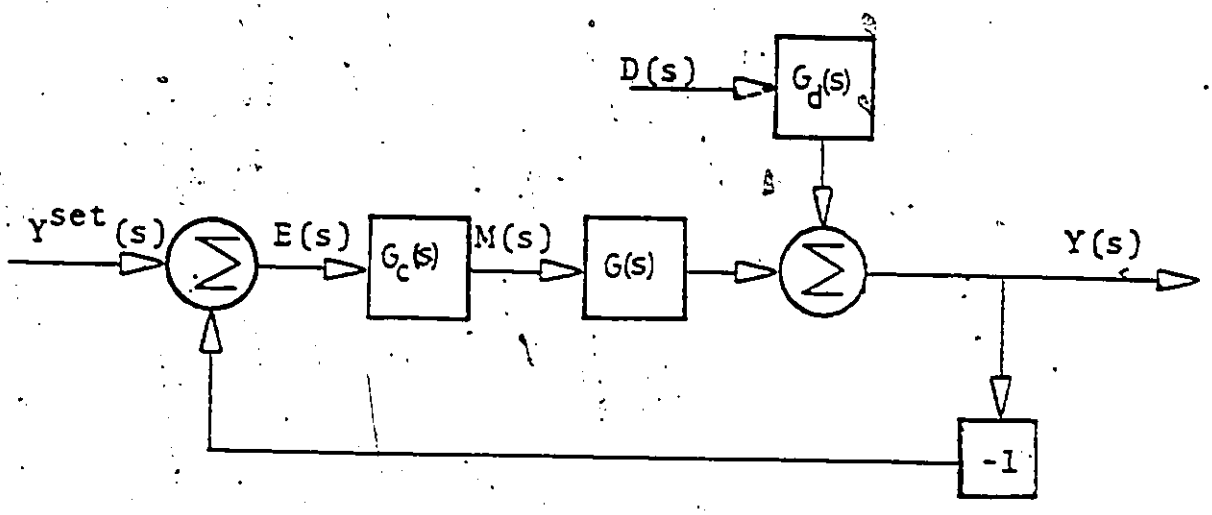


Figure 2.3: Feedback Control Loop.

$Y(s)$ = Laplace transformed controlled variable,

$Y^{set}(s)$ = Laplace transformed setpoint,

$E(s)$ = Laplace transformed error signal,

$G_c(s)$ = Laplace transformed controller algorithm,

$M(s)$ = Laplace transformed controller output,

$G(s)$ = Laplace transformed process transfer function,

$D(s)$ = Laplace transformed disturbance and

$G_d(s)$ = Laplace transformed disturbance transfer function.

reset windup, ratio control and advanced control strategies) have been developed in the last 30 years. Among the advanced or more complex control strategies it is worth mentioning feedforward control, cascade control, dead-time compensation, adaptive control and multivariable control. Most of them could be handled with analog devices, but required special hardware purchases (i.e. lead/lag elements, pure dead-time elements, etc). With the advent of digital computer control, it became possible to easily implement all these non-conventional control strategies.

2.4.1 PID Feedback Control

The ideal continuous PID algorithm is described in the time domain by the equation:

$$m(t) = m_o + K_c\, e(t) + \frac{K_c}{\tau_I} \int_0^t e(t)dt + K_c\, \tau_d \frac{de(t)}{dt} . \quad (2.1)$$

where $m(t)$ = controller output,

m_o = controller output when process is at steady-state at its setpoint.

$e(t)$ = controlled variable deviation from set-point,

K_c = controller gain,

τ_I = integral or reset time (minutes) and

τ_d = derivative time or rate (minutes).

The digital computer, because of its inherent structure, cannot handle continuous algorithms; discrete approximations are necessary. By replacing the integral term by rectangular integration and the derivative by finite difference, we obtain the discrete equivalent of the equation above[4,5]. At time t,

$$m_t = m_o + K_c e_t + \frac{K_c S_t}{\tau_I} \sum_{0}^{t} e_t + \frac{K_c \tau_d}{S_t} (e_t - e_{t-1}) \quad . \quad (2.2)$$

where m_t = output from controller at the t'th sampling time,

S_t = sampling time,

e_t = error at the t'th sampling time and

e_{t-1} = error at the (t-1)'th sampling time.

The equation above is called the POSITION form of the PID control algorithm, because the actual control output is computed at each sampling time. Similarly at time t-1,

$$m_{t-1} = m_o + K_c e_{t-1} + \frac{K_c S_t}{\tau_I} \sum_{0}^{t-1} e_t + \frac{K_c \tau_d}{S_t} (e_{t-1} - e_{t-2}) \quad . \quad (2.3)$$

Subtracting Eqn. (2.3) from Eqn. (2.2) yields

$$\Delta m_t = m_t - m_{t-1} = K_c (e_t - e_{t-1}) + \frac{K_c S_t}{\tau_I} e_t + \frac{K_c \tau_d}{S_t} (e_t - 2e_{t-1} + e_{t-2}) . \quad (2.4)$$

Equation (2.4) is known as the VELOCITY form of the PID algorithm because, instead of the actual output, the incremental output, Δm , is computed. In the computer control program, the PID velocity equation is implemented as

$$m_t = m_{t-1} + \Delta m_t .$$

The performances of the position and velocity PID algorithms are not equivalent[18,24] . To initialize the position algorithm (switch from manual to automatic control), in order to ensure a smooth transition, the operator has to set the summation term in eqn. (2.2) equal to zero and the bias term m_0 equal to the current value of the output. For the velocity algorithm, initialization is carried out simply by setting m_{t-1} equal to the current value of the manipulated variable at the time the controller is switched to automatic. This is performed automatically by the algorithm, as can be seen in eqn. (2.4) .

The other advantage of the velocity algorithm is protection against reset wind-up. Reset wind-up occurs when the manipulated variable, m_t, saturates (i.e. reaches its upper or lower limit) and the error is still nonzero. The reset term (i.e. the summation in the position algorithm) will wind-up (i.e. keep on growing beyond an appropriate limit). When the controlled variable finally reaches its setpoint, a large overshoot in the direction opposite to that which caused the saturation occurs.

Reset wind-up is avoided for the position algorithm by detecting when the control element saturates and terminating the summation until the manipulated variable has returned to within the control range. As can be seen from eqn. (2.4), the velocity PID algorithm does not have the summation term, and reset wind-up will not occur, as long as the manipulated variable m has upper and lower limits.

The major drawback of the PID velocity algorithm is that the reset term is absolutely required in the equation. This is because the setpoints are actually canceled out in computing the proportional and derivative terms[7]. Recall eqn. (2.4), but now making $e = X^{set} - X_t$. At time t ,

$$m_t = m_{t-1} + K_c\{(X^{set} - X_t) - (X^{set} - X_{t-1})\} + \frac{K_c S_t}{\tau_I}(X^{set} - X_t) +$$

$$\frac{K_c \tau_d}{S_t}\{(X^{set} - X_t) - 2(X^{set} - X_{t-1}) + (X^{set} - X_{t-2})\}$$

Therefore, P and PD modes result in very poor control[7], because the value of the setpoint vanishes from the equation. Despite this limitation, the velocity algorithm is the most widely used in digital process control applications[24].

It is important to note, at this point, that the velocity form is the same as the equation obtained by discretizing the PID algorithm using sampled-data control theory, since both are based on finite-difference approximations.

This equality is shown in Appendix A. Both the velocity and the position algorithms were implemented in our system.

2.4.2 Cascade Control

Cascade control is a particular application of PID feedback control. Assume we have a process that can be represented by 2 lags in series as in Figure 2.4 (a).
If process A is faster than process B but subject to frequent disturbances, the overall control performance can be improved with the cascade arrangement shown in Figure 2.4 (b). The output of the master controller, $Gc_m(s)$, is the setpoint to the slave controller, $Gc_s(s)$. The output of the slave controller positions the final control element. With the cascade configuration, minor disturbances that affect the faster process will be detected and compensated for before they affect the outer loop. Cascade control is not implemented in this project.

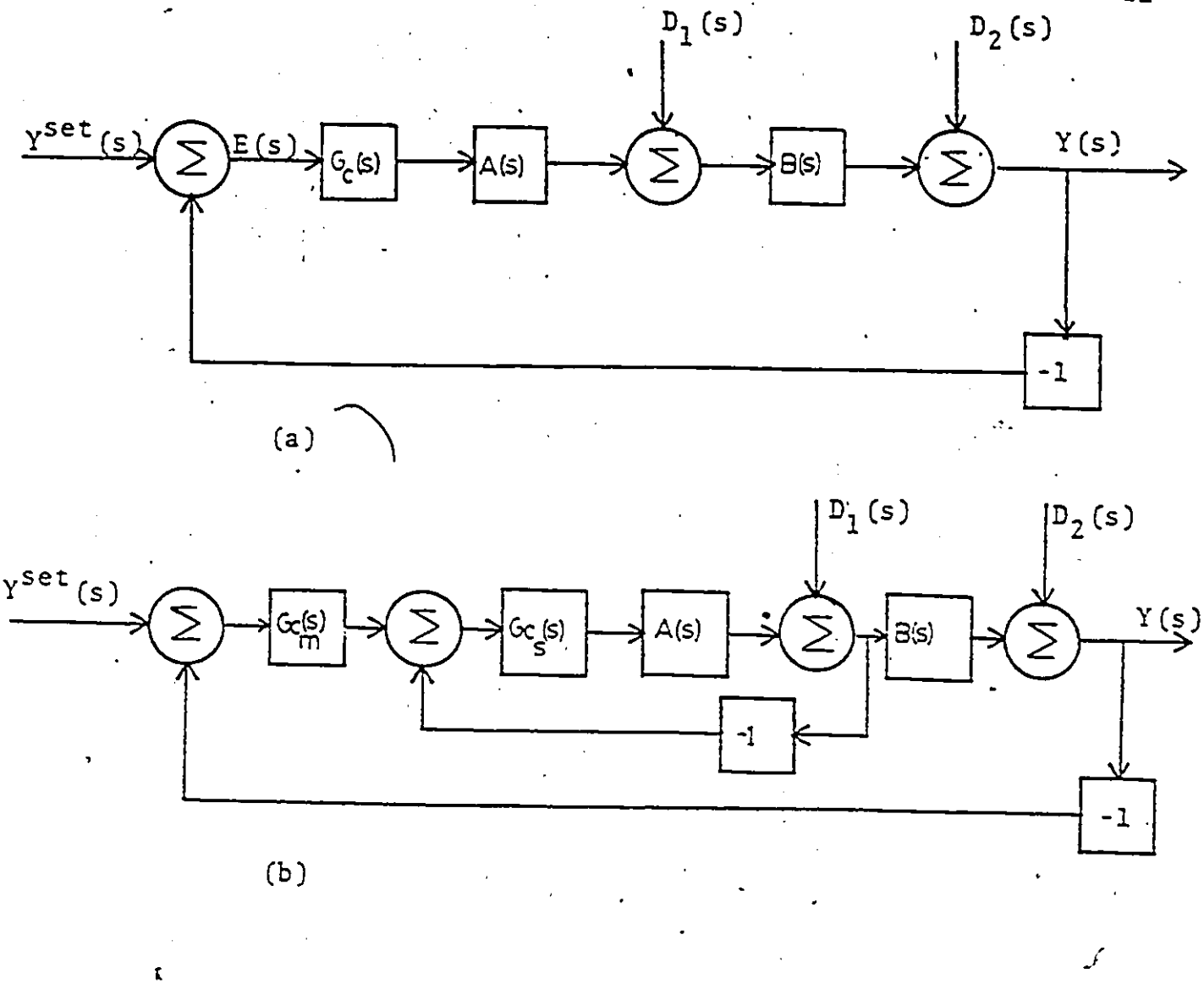


Figure 2.4 :Classical Feedback Control (a) and Cascade Control (b).

$A(s)$, $B(s)$ = Laplace transformed process transfer functions,

$D_1(s)$, $D_2(s)$ = Laplace transformed disturbances,

$G_c(s)$ = Laplace transformed controller algorithm,

$Y(s)$ = Laplace transformed controlled variable,

$Y^{set}(s)$ = Laplace transformed setpoint.

$Gc_m(s)$ = Laplace transformed master controller algorithm and

$Gc_s(s)$ = Laplace transformed slave controller algorithm.

2.4.3 Feedforward Control

Feedforward control is based on measuring disturbances as they enter the process, and adjusting the manipulated variables to compensate for these disturbances, so that the controlled variables are kept at their setpoints (^{s}, ^{is}) . As opposed to feedback control, feedforward control does not wait until the disturbance has propagated throughout the process to generate an error signal, rather it acts immediately as the disturbance enters the process. A block diagram for a feedforward control application is shown in Figure 2.5 (a).

To achieve perfect feedforward control all disturbances must be detected and the exact relationships between both the disturbances and manipulated variable with the controlled variable must be known. As a result, feedforward control is seldom used alone. To overcome these weaknesses, feedforward control is usually implemented together with feedback control, where each strategy covers the other's weaknesses. Figure 2.5 (b) shows combined feedforward / feedback control. Feedforward control was used in this study in connection with the decoupling algorithms.

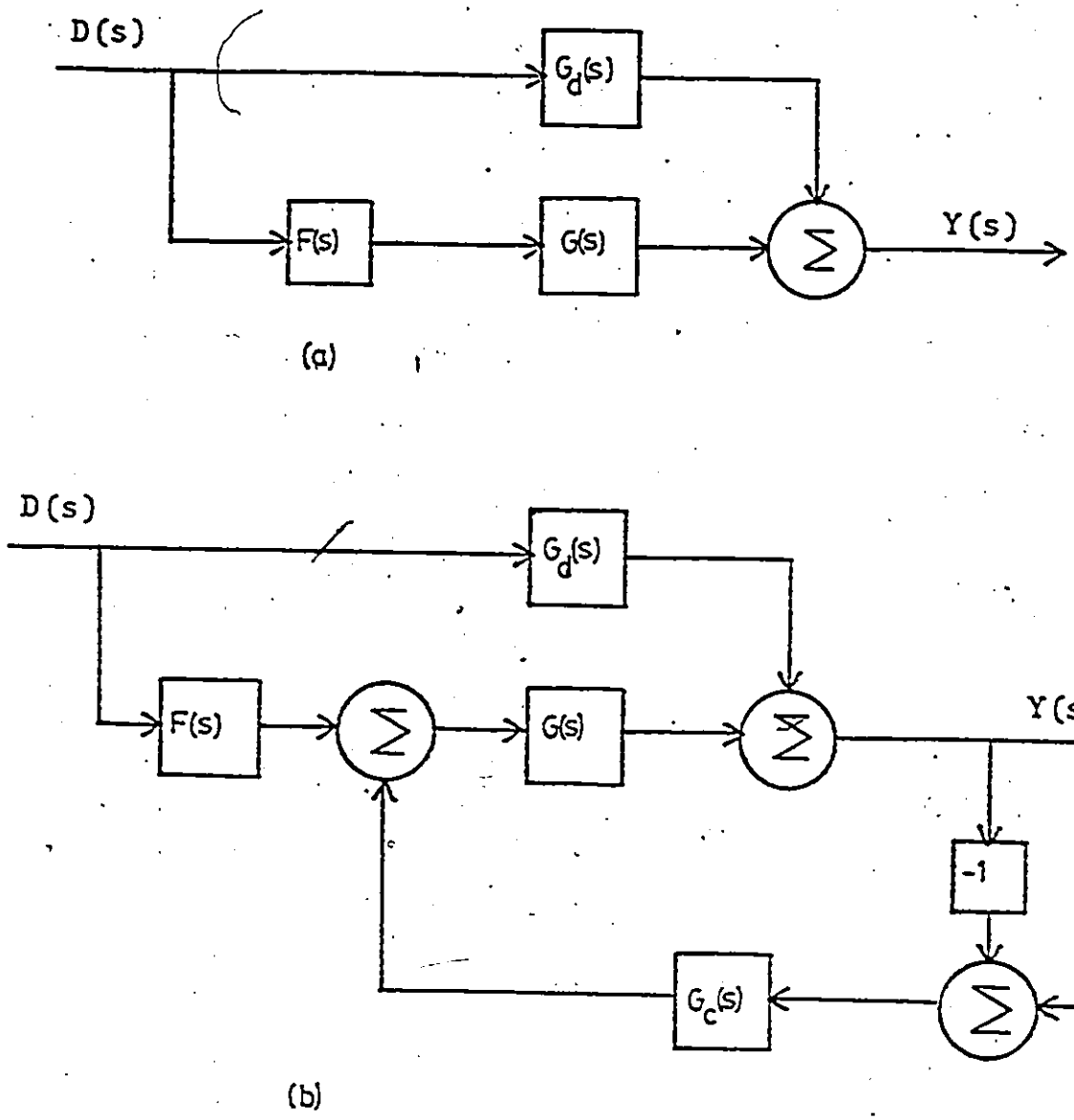


Figure 2.5 : Feedforward Control (a) and combined Feedforward-Feedback Control (b).

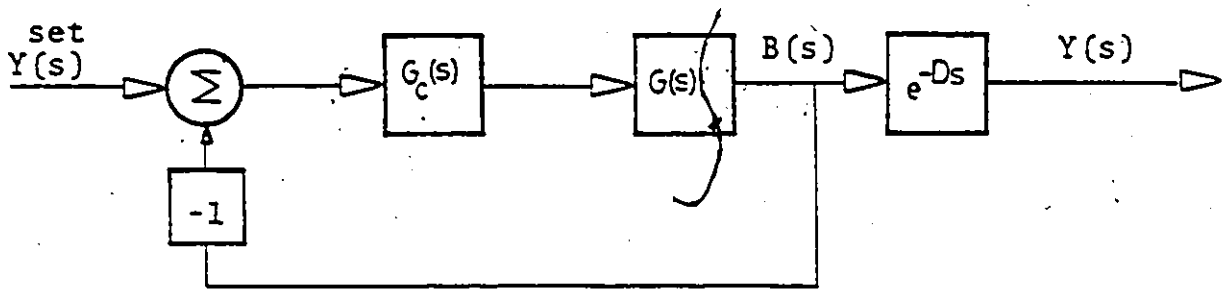
$D(s)$ = disturbance,
$G_d(s)$ = disturbance transfer function,
$G(s)$ = process transfer function,
$F(s)$ = feedforward control transfer function and
$Y(s)$ = controlled variable.
$G_c(s)$ = controller algorithm and
$Y^{set}(s)$ = setpoint.

2.4.4 Dead-time compensation

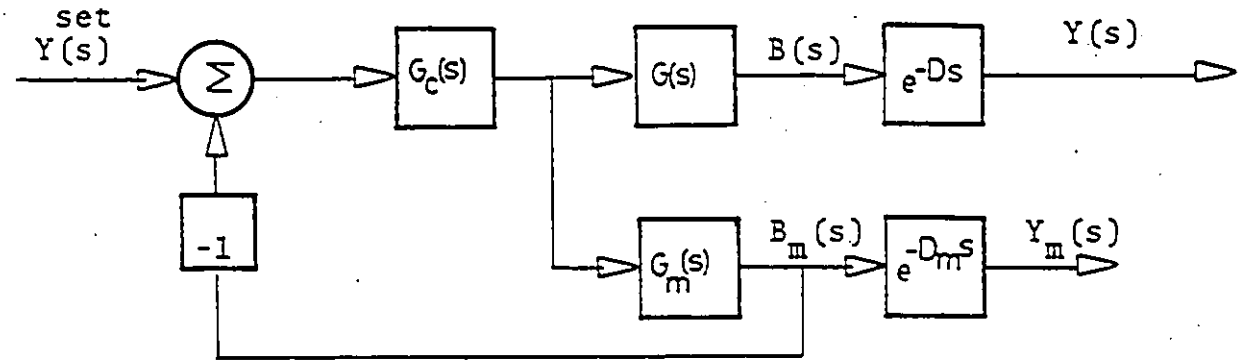
The effects of deadtime on a control loop can sometimes be considerably reduced with the implementation of cascade control or feedforward control. Specific dead-time compensation techniques usually involve a prediction device in the control loop to compensate for the time delay. Here the digital computer becomes extremely important because of its capability of "predicting" values of certain variables in the future, based on mathematical models.

The best known of the dead-time compensation techniques in use today is the Smith predictor. Smith developed this technique in 1957 aiming to use it for analog controllers [11], but the difficulty of implementing the Smith predictor with analog circuitry prevented it from gaining wide acceptance. With digital computer control, the Smith predictor has been used in many applications[9]. This compensator works by eliminating the deadtime from the feedback loop, therefore allowing higher controller gains.

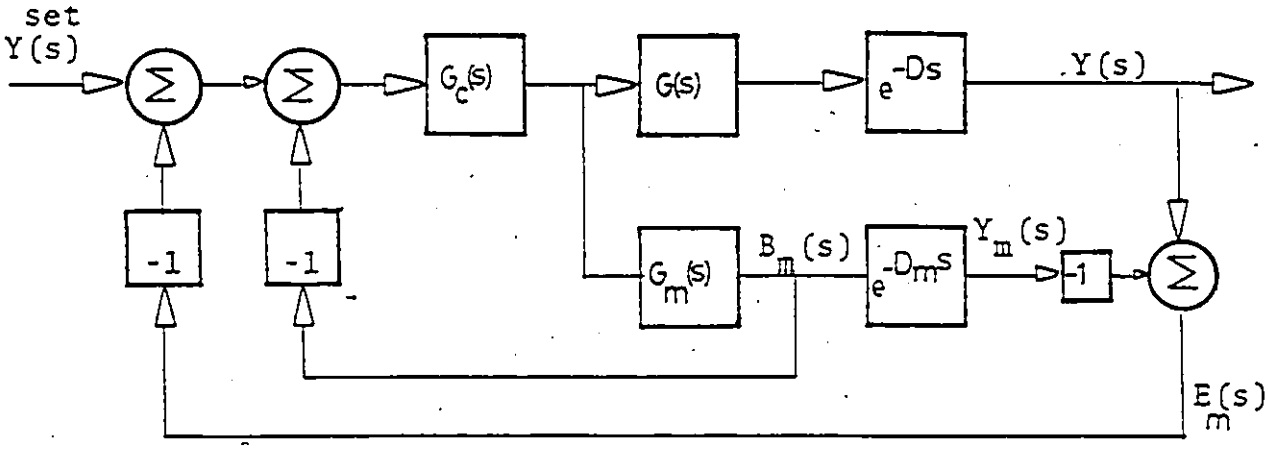
Suppose that a process is composed of a lag-plus-deadtime $G(s)e^{-Ds}$. If it were possible to separate the lag from the deadtime, the ficticious signal B(s) could be connected to the feedback controller and the deadtime would be moved out of the control loop, as can be seen in Figure 2.6 (a). The controlled variable Y(s) would repeat whatever B(s) did, D units of time later.



(a) Desired configuration.



(b) Incorporating the process model.



(c) Final Smith Predictor Scheme.

Figure 2.6 : Feedback loop with Smith Predictor.

The problem with this concept is that the lag and the dead-time do not occur as distinct elements in the plant, and cannot be separated. If, however, a model of the process , $G_m(s)e^{-D_m s}$, where available, this model could be split into a pure lag $G_m(s)$ and a pure deadtime $e^{-D_m s}$, the signal $B_m(s)$ could be computed and connected to the controller, as shown in Figure 2.6 (b). If the process model were perfect and no load upsets occurred, the output from the model, $Y_m(s)$, would be equal to the controlled variable, Y(s), and such a control scheme would work well. In practice, model imperfections and load upsets are always present, and an error $E_m(s)$ between $Y_m(s)$ and Y(s) will be generated. To compensate for this error, a second feedback loop using $E_m(s)$ is implemented as shown in Figure 2.6 (c). This control scheme is known as SMITH PREDICTOR or DEAD TIME COMPENSATOR.

Another technique for dead-time compensation is known as the ANALYTIC PREDICTOR. The analytic predictor has been developed primarily for sampled-data systems. It includes in its structure corrections for the effect of sampling. According to Ray[1] ,page 212, it can be shown that the Smith predictor and the analytic predictor are equivalent. Due to the lack of significant dead-time in our process, none of the techniques above were implemented.

2.4.5 Adaptive Control

Adaptive Control[17,18,19] is an on-line strategy that consists of simultaneously identifying and controlling a process. This strategy is particularly useful for controlling nonstationary processes, i.e. processes whose characteristics change with time. The interest in adaptive control began in 1959 with the aerospace industry[19] . Time varying weight of spacecrafts due to fuel consumption, or changes in altitude, were causing poor control when classical controller concepts were applied.

As can be seen in Figure 2.7 , adaptive control consists in periodically testing the process G(s) (for example, by pulse testing), and allowing the process parameters to be identified by means of the identification package. Having obtained the process parameters, the computer calculates the tuning parameters to be used by the controller.

The major drawback of the adaptive control strategy is that the process has to be tested on-line, which causes off-sets in the controlled variable. The overall improvement of the adaptive control has to overcome these periodic off-sets to justify the implementation of this advanced control strategy. Another limitation is that the additional testing, identification and tuning computations require a relatively large computer system. Therefore adaptive control is not usually implemented with microcomputers, and was not considered in this work.

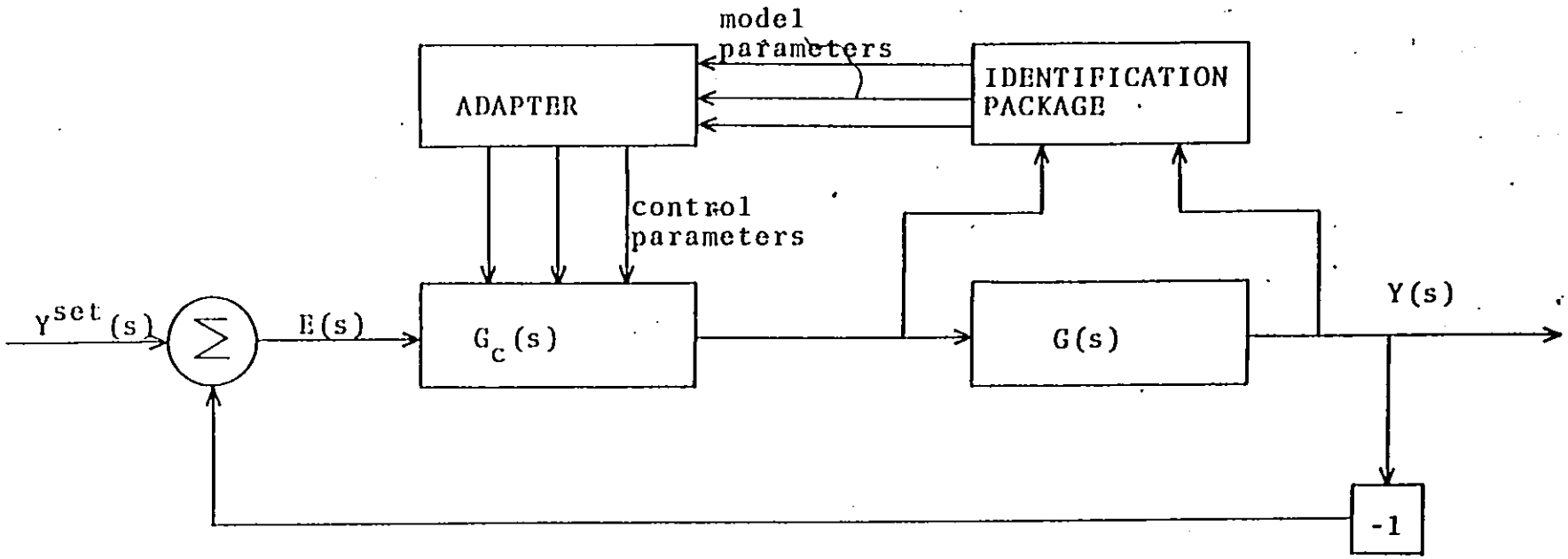
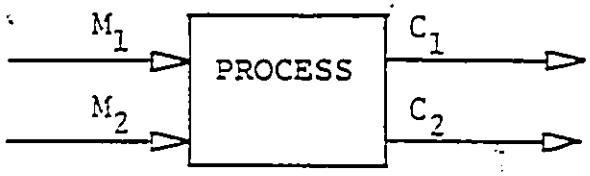


Figure 2.7 : Adaptive Control System.

2.4.6 Multivariable Control

Classical control theory is based on single-input/single-output (SISO) processes [14]. Most large industrial processes, however, contain several controlled variables and several manipulated variables in the same process (multiple inputs/outputs). In these multivariable systems, each manipulated variable affects its own controlled variable and often other controlled variables. In a 2-loop system, for instance, if loop 1 affects loop 2, coupling is said to exist. If loop 2 also affects loop 1, cross-coupling or interaction results[9]. Coupling is not a big problem, since the loops can be tuned one at a time. Cross-coupling, however, causes the loops to fight each other to eliminate offset, and as a result the process can become unstable. In such cases the classical controller design techniques are not effective. One of the main applications of multivariable control is to minimize the interaction among loops so as to obtain a good overall control performance.

A measure of steady-state interaction is given by the Bristol Array [14]. One way of explaining the Bristol Array is to consider a 2x2 multivariable system of the form:



The elements of the Bristol Array (λ_{ij}) are the ratio of the steady-state open loop response and the steady-state closed-loop response when a particular manipulated variable is adjusted:

$$\lambda_{ij} = \frac{\left[\frac{\Delta C_i}{\Delta M_j}\right]_{\text{all loops open}}}{\left[\frac{\Delta C_i}{\Delta M_j}\right]_{\text{all loops closed except for } M_j}} \qquad i,j = 1,2.$$

For a 2x2 system, the Bristol Array can alternatively be given by the expression below, where the K_{ij}'s are the steady-state process gains:

$$\Lambda = \frac{\begin{bmatrix} K_{11}K_{22} & -K_{21}K_{12} \\ -K_{21}K_{12} & K_{11}K_{22} \end{bmatrix}}{K_{11}K_{22} - K_{12}K_{21}}$$

Both ways of calculating it are equivalent (') . In the relative gain matrix, the sum of any row or column is unity. When the process transfer function matrix is diagonal or triangular, the relative gain matrix is the identity matrix. The greater the off-diagonal terms, the greater the interaction among loops. By examining the λ_{ij} , we can choose the

pairings of C's and M's so that the diagonal elements are largest relative to the off-diagonal terms. Therefore, the Bristol Array is a guide to choosing the best pairing of manipulated and controlled variables so as to minimize interaction. The only limitation of the Bristol Array method is that it does not take into account the dynamics of the loops, which can be appreciable in some cases [1].

For control loops still in the design stage, the following procedures[9,14] should be followed:

1. From the interpretation of the Bristol Array (relative gain matrix, or RGM), we can infer if a multivariable control system is likely to be a problem. If the relative gain matrix is the identity matrix, or at least if the diagonal terms are dominant with respect to the off-diagonal terms, single-loop design is feasible[14].
2. If diagonal dominance has not been achieved, a different manipulative/controlled variable pairing should be tried.
3. If re-structuring the system does not improve diagonal dominance in the R.G.M., decoupling will be necessary.

For a multivariable control system already in operation, interaction between loops can be reduced by using feedforward control or alternatiyely, for a 2x2 system, by tuning one loop very tightly and the other loosely (high and low controller gains respectively).

A multivariable system can be represented by the block diagram in Figure 2.8 (a). Multivariable systems are more easily represented using matrix notation, because of its more compact form. Figure 2.8 (b) shows the same multivariable system using matrix representation.

The closed loop transfer function between $\underline{y}$ and $\underline{y}^{set}$ is given by

$$\underline{T}(s) = \underline{y} / \underline{y}^{set} = (\underline{I} + \underline{G}(s)\ \underline{G}_c(s))^{-1}\ \underline{G}(s)\ \underline{G}_c(s)$$

In interacting systems, $\underline{G}(s)$ will have off-diagonal terms, and so will the closed-loop transfer function $\underline{T}(s)$, resulting in strong steady-state and dynamic interactions in the system.

Decoupling consists of inserting a decoupling matrix $\underline{G}_I(s)$ between $\underline{G}_c(s)$ and $\underline{G}(s)$ in the closed-loop dynamic path (Figure 2.9) so that $\underline{T}(s)$ becomes diagonal. In this case

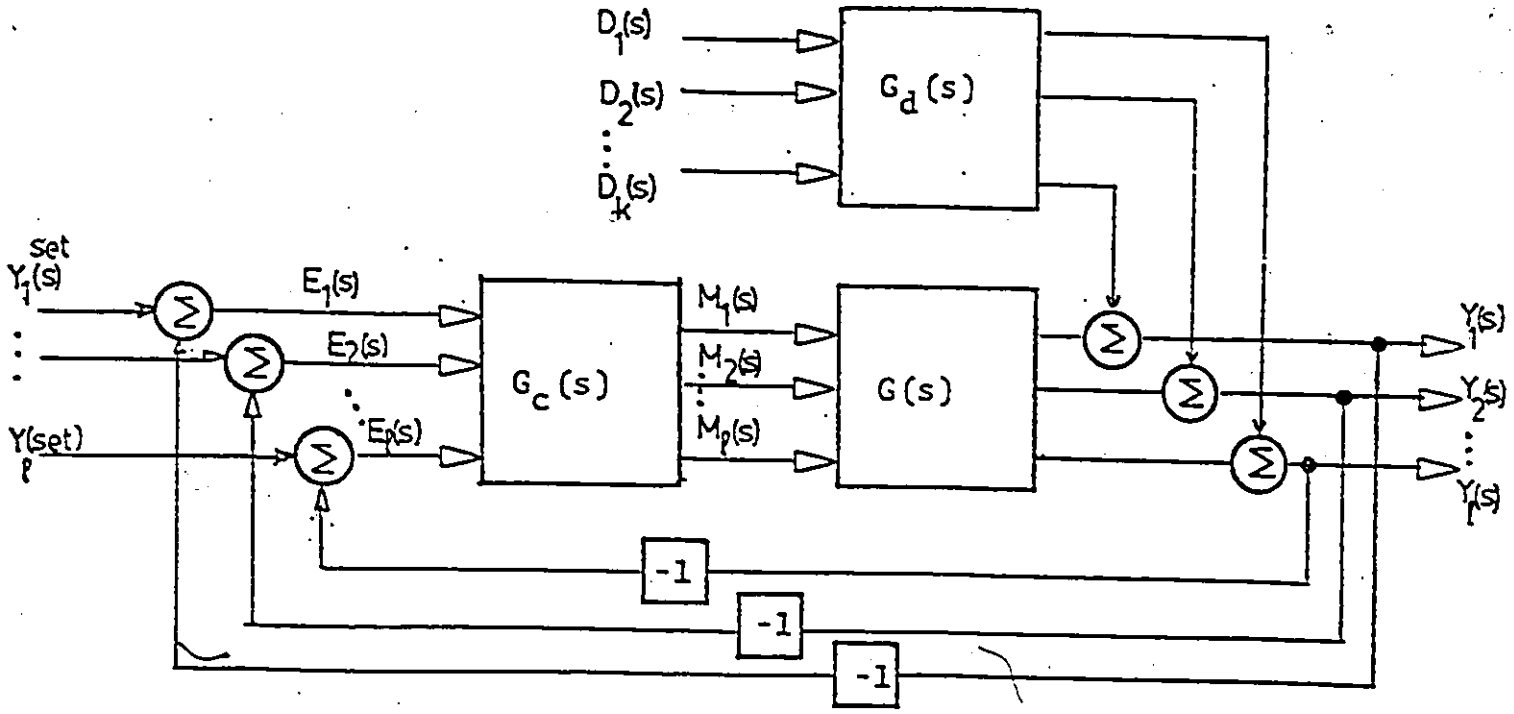
$$\underline{T}(s) = \underline{y} / \underline{y}^{set} = (\underline{I} + \underline{G}(s)\underline{G}_I(s)\underline{G}_c(s))^{-1}\ \underline{G}(s)\underline{G}_I(s)\underline{G}_c(s)$$

Since $\underline{G}_c(s)$ is diagonal (one single controller for each control loop), a sufficient condition for $\underline{T}(s)$ to become diagonal is that

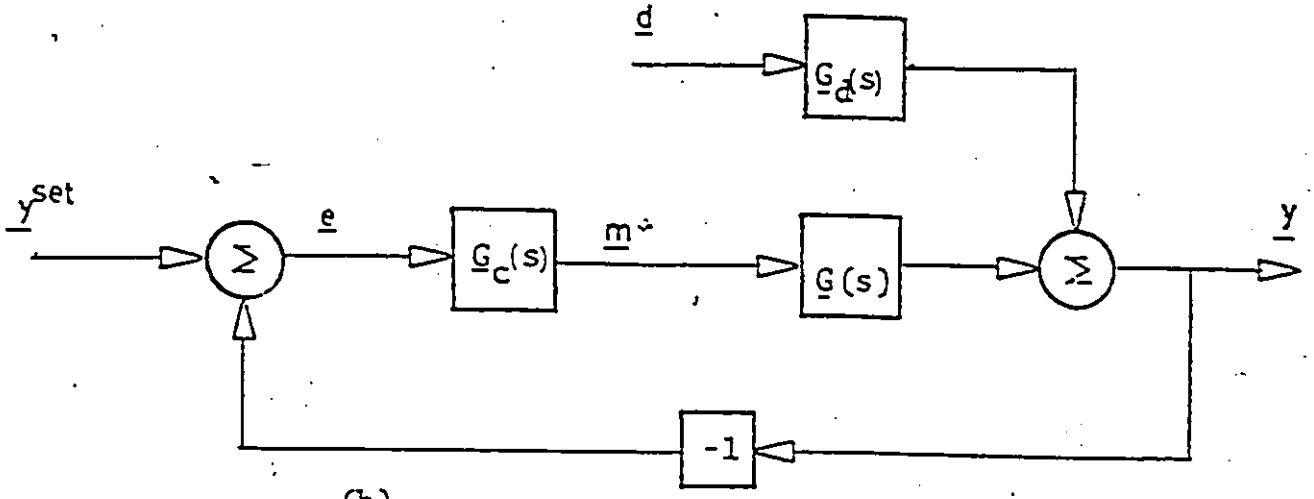
$$\underline{G}(s)\ \underline{G}_I(s) = \text{diag}\ \underline{G}(s)$$

and the compensator is defined as:

$$\underline{G}_I(s) = \underline{G}(s)^{-1}\ \text{diag}\ \underline{G}(s)$$



(a)



(b)

Figure 2.8: Scalar (a) and Matrix (b) multivariable system Representation.

where $\underline{y}$ = vector of controlled variables,

$\underline{y}^{set}$ = vector of setpoints,

$\underline{G}(s)$ = process transfer function matrix,

$\underline{G}_c(s)$ = controller matrix, and

$\underline{G}_d(s)$ = disturbance transfer function matrix.

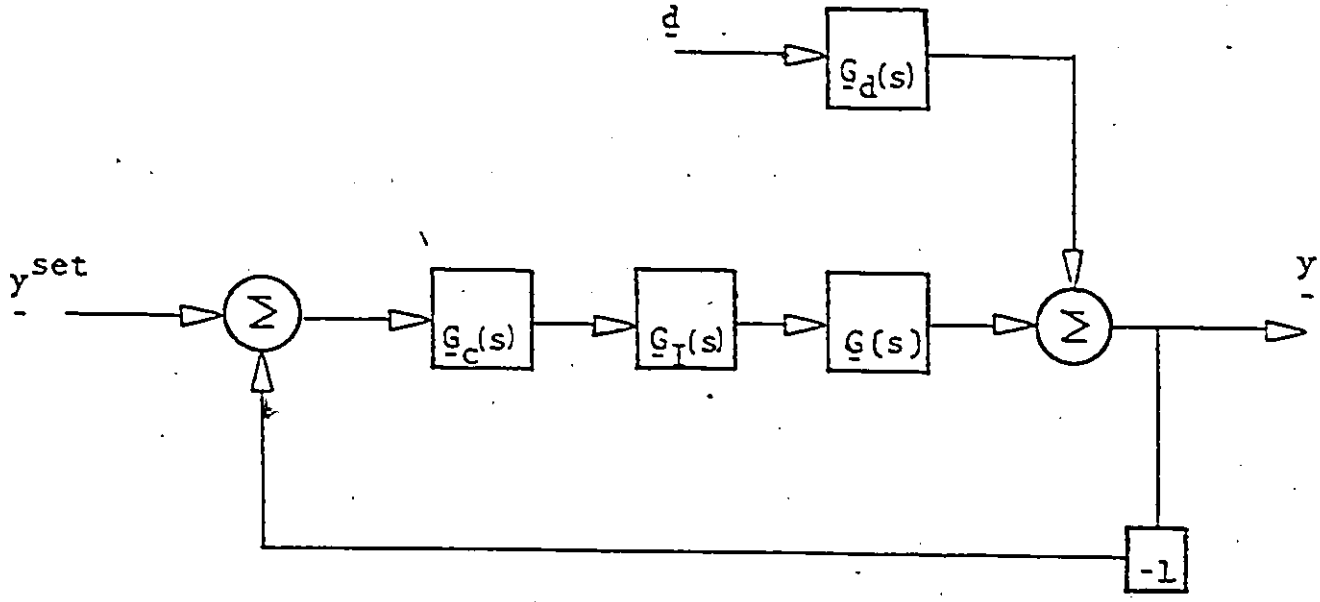


Figure 2.9: Multivariable system with decoupler.

$\underline{y}$ - vector of controlled variables,

$\underline{y}^{set}$ - vector of setpoints,

$\underline{G}(s)$ - process transfer function matrix,

$\underline{G}_d(s)$- disturbance transfer function matrix,

$\underline{G}_c(s)$- controller matrix,

$\underline{d}$ - vector of disturbances and

$\underline{G}_I(s)$- decoupler matrix.

$\underline{G}_I(s)$ is called a SIMPLIFIED DYNAMIC DECOUPLER, because the product $\underline{G}_I(s)\ \underline{G}(s)$ is simply a diagonal matrix. The IDEAL DYNAMIC DECOUPLER is defined as[1] :

$$\underline{G}_I(s) = \underline{G}(s)^{-1}$$

so that $$\underline{G}_I(s)\ \underline{G}(s) = \underline{I}$$

For the compensator $\underline{G}_I(s)$ to be implemented in a digital computer control program, its Laplace domain equations, usually lead-lag elements, must be converted to the time domain. The resulting integrals and derivatives are approximated by summations and finite differences respectively.

Both the process matrix $\underline{G}(s)$ and the controller matrix $\underline{G}_c(s)$ must be square matrices for the compensator to be calculated. This is actually the case if the number of manipulated variables equals the number of controlled variables.

Sometimes dynamic decoupling is difficult or even impossible to implement. For example, the dynamic model may not be well known, or the dynamic decoupler algorithm may not be physically realizable. In this case we can neglect the dynamics and evaluate the steady-state decoupler, defined as

$$\underline{G}_{I_{s.s.}}(s) = \underline{G}_{s.s.}(s)^{-1}\ \text{diag}\ \underline{G}_{s.s.}(s)$$

where $\underline{G}_{ss}(s)$ - steady state process transfer function matrix,

$\underline{G}_{I_{ss}}(s)$ - steady state decoupler.

Of course there will be a period of dynamic interaction, but the steady-state effects will be decoupled. Steady-state decoupling is extremely easy to implement [1] . Multivariable control has been successfully employed in a number of industries, including the chemical, petroleum, steel, paper and cement industries[21] . In this work we will examine its implementation in a microcomputer based control system.

Chapter III

DESCRIPTION OF THE MICROCOMPUTER-BASED CONTROL SYSTEM

Several things were borne in mind when designing the process control system:

1. Since the system will be used for instructional purposes, the process should have fast dynamics (i.e. time constants in the order of a few minutes), so that several experiments could be carried out during a typical lab period.
2. The process should present sufficient complexity so that some advanced control strategies could be implemented.
3. The process should be open-loop stable, so that model identification could be carried out using classical control theory.
4. The control loops (manipulated / controlled variables) should be chosen so as to be representative of those found in a chemical process.
5. Constraints related to equipment availability also had to be taken into account.

Considering the constraints above, we decided to implement a multivariable (2x2) level / temperature control.

A sketch of the set-up is shown in Figure 3.1 .

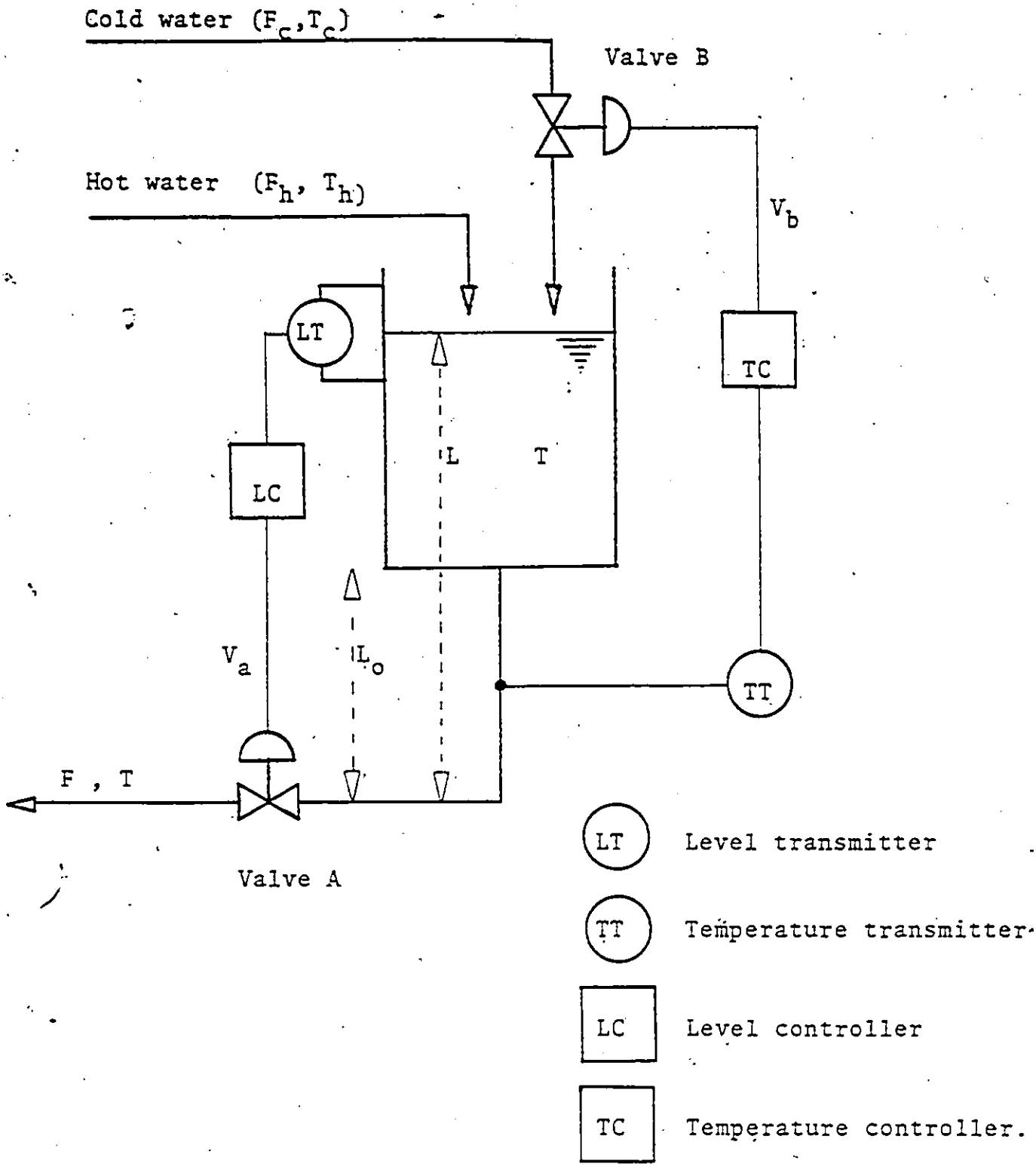


Figure 3.1: Simplified process flowsheet.

The process consists of mixing two streams of water (one hot and one cold) in a tank. The level is controlled by adjusting the outlet water flowrate, F. The temperature of the outlet stream, T, is controlled by adjusting the cold water inlet flowrate, F_c. The hot water flowrate, F_h, is manually adjusted and is heated by passing building service water through a heat exchanger before the tank. The hot water temperature is controlled by a separate satellite microprocessor, which adjusts the amount of steam that enters the heat exchanger. It is important to note at this point that the microprocessor, despite receiving its setpoint from the computer, is not incorporated in the multivariable system considered in this work.

In Figure 3.2 details of the complete set-up, including the controllers, transducers and valves are shown.

Each element of the system will now be described. In the sequence, a theoretical model for the process will be developed, including the open-loop transfer functions. This will give us an idea about the extent of nonlinearities in the process, its time scale and its stability. And finally, the software required to implement digital control will be described.

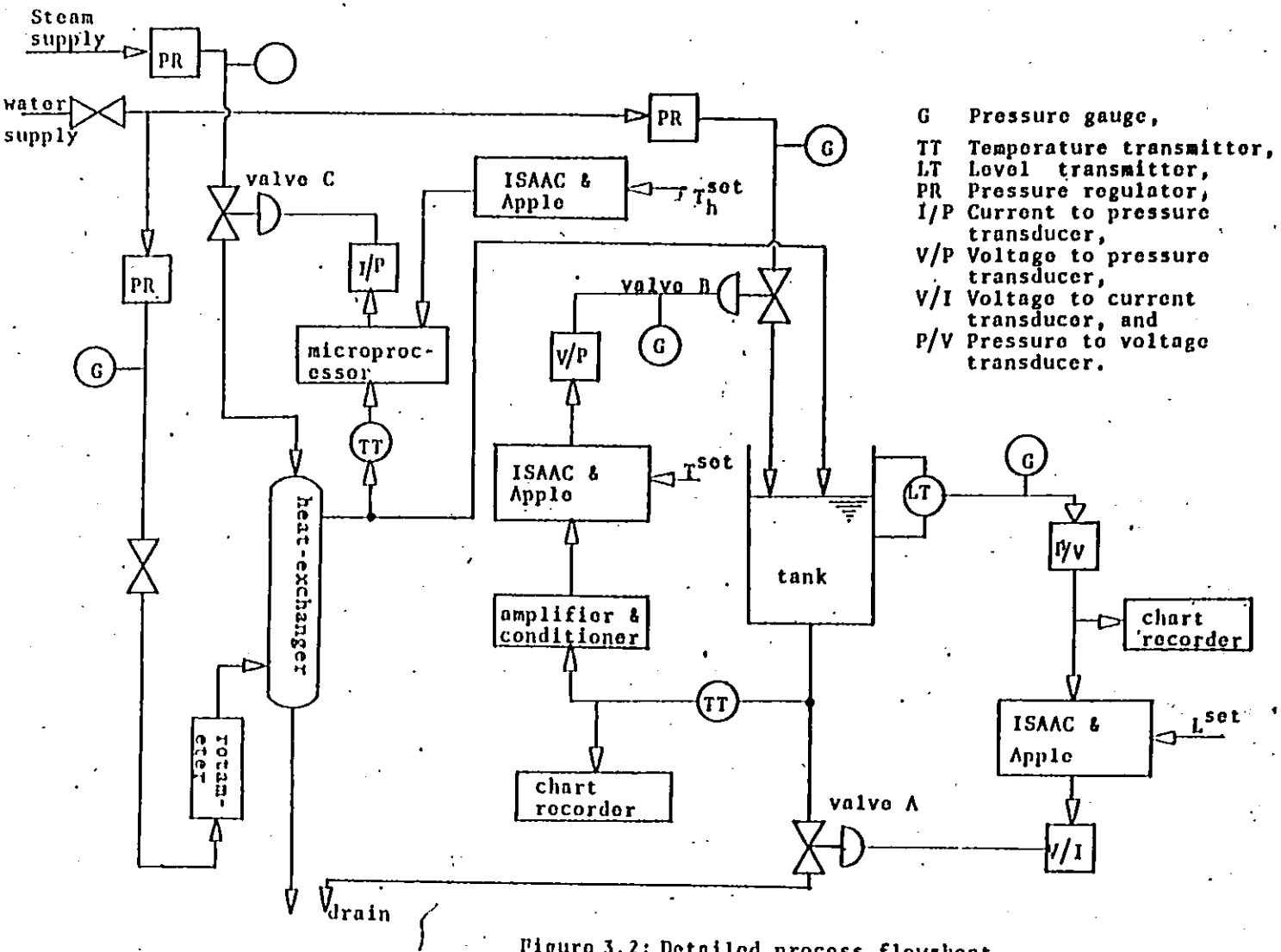


Figure 3.2: Detailed process flowsheet.

3.1 DETAILS OF THE PROCESS EQUIPMENT

3.1.1 The Water Tank

The cylindrical tank in which the hot and cold water streams are mixed is made of clear acrylic plastic, to allow visualization of the liquid level. It sits on a platform situated 1 m above the floor; the distance between the bottom of the tank and valve A being 0.69 m. The inside diameter of the tank is 0.29 m and its height, 0.45 m.

3.1.2 Valve A

Valve A controls the tank outlet flowrate. Its equation, where L is expressed in decimeters, is

$$F = -7.77 + 2.80\ V_a\sqrt{L} \qquad (B.4)$$

The development of equation (B.4) is presented in Appendix B.

Characteristics:

Manufacturer : Honeywell,
Model : 4805 3/4" single seated valve,
Material : cast iron body, bronze plug & seat,
Valve size coefficient : $C_v = 6.3$,
Direct acting actuator,
Air to open ,
Linear trim and
Flowrate = 0 to 25 L/min (ΔP = 1.1 psig).

3.1.3 Valve B

Valve B controls the cold water inlet flowrate. Its equation is

$$F_c = -3.33 + 5.33\ V_b \qquad (C.3)$$

The development of equation (C.3) is presented in Appendix C.

Characteristics:

Manufacturer : Fisher Governor Co. ,
Model : 667 A 1/2" single seated valve,
Material : cast iron body, stainless steel trim,
Valve size coefficient : not given,
Direct acting actuator,
Air to open ,
Trim : not given and
Flowrate = 0 to 28 L/min (ΔP = 30 psig).

3.1.4 Valve C

Valve C controls the steam flowrate that enters the heat exchanger. It is the final control element for the UDC 500 microprocessor controlled temperature loop.

Characteristics:

Manufacturer : Honeywell,
Model : 1403 1/2" ,
Material : stainless steel trim,

Valve size coefficient : C_v = 1.6 ,

Direct acting actuator,

Air to open and

Linear trim .

3.1.5 Hot Water Supply

The heat exchanger heats up building service water, before this water is admitted into the tank. The service water temperature can vary between 4 and 20 ^{o}C, depending on the season. All the piping is copper with 1/2" I.D. and 5/8" O.D. , and the pressure in the line is kept constant at 30 psig by means of a pressure regulator. This hot water flowrate is limited to 14 L/min. Before the heat exchanger, there is an on-line, manual valve and a rotameter so that the hot water flowrate can be manually adjusted. The rotameter calibration curve is shown in Appendix D. The hot water that leaves the heat exchanger has its temperature sensed by a type T thermocouple and controlled by a Honeywell UDC 500 Universal Digital Controller, which manipulates valve C. The UDC 500 implements a separate temperature loop, and receives its set-point from the supervisory computer (ISAAC-Apple unit). The hot water temperature is stable.

3.1.6 Cold Water Supply

Building service water is admitted to the tank through a 1/2" I.D, 5/8" O.D. copper pipe through valve B. The gauge pressure on the pipe upstream to the valve is held constant and equal to 30 psig by means of a pressure regulator.

3.1.7 Temperature sensor/transmitter

The temperature sensor-transmitter is a type T (copper-constantan) thermocouple model GG-28-T manufactured by Thermo Electric Canada Ltd. It is located between the tank and valve A, approximately 0.6 m away from the tank. Its output signal is typically in the range 1.0 to 2.5 millivolts. Appendix E describes the conversion to degrees Celsius for a type T thermocouple.

3.1.8 Level Transmitter

The water level inside the tank is sensed by a float which is connected to the level transmitter. The level transmitter (manufactured by Taylor, model 113 RF) outputs an air pressure signal in the range 3 to 15 psig which is proportional to the position of the float. Adjustable levers allow us to choose the operating region for the sensor (float), as well as its total amplitude of movement (span). Appendix F shows the voltage signal (VDC) vs. float position curve.

3.1.9 Transducers

Transducers transform signals into different physical quantities. This is necessary in our set-up, since we work with several different types of signals which very often are not compatible with the equipment involved. Our system has the following transducers:

1) psig to VDC-

Manufacturer : Viatran,
Model : 218 Pressure Transducer ,
Input : 3 to 15 psig ,
Output : 0 to 5 VDC ,
Power supply : Hammond HPFS 28 VDC and
Response time : < 1 millisecond.

2) VDC to psig-

Manufacturer : Acromag ,
Model : FPM Voltage to Pneumatic Converter ,
Input : 0 to 5 VDC ,
Output : 3 to 15 psig ,
Air supply : 30 psig and
Power supply : 115 VAC .

3) Millivolts DC to VDC-

This operation is performed by the I-140 board, which will be described in the next section.

4) VDC to milliamps-

Manufacturer : Rochester Instrument Systems ,

Model : SC-1300 Signal Transmitter ,
Input : 0 to 5 VDC ,
Output : 4 to 20 milliamps ,
Power supply : 115 VAC ,
Response time : < 50 milliseconds and
Linearity : typically +/- 0.1% of full span.

Transformations performed by the sensors, transmitters and transducers described above are linear and extremely fast if compared with the process time constants. They are therefore treated as simple gains.

3.1.10 Chart-recorder

The chart-recorder (manufactured by Watanabe, model Linear Corder Mark VII) connected to our process is able to plot simultaneously 2 continuous analog signals. We use the chart recorder to record the signal from the level transmitter after it is transduced into VDC, and also the millivolt signal from the thermocouple that measures the outlet water temperature.

3.2 THE ISAAC-APPLE COMPUTER SYSTEM

The ISAAC (acronym for Integrated System for Automated Acquisiton and Control) -Apple system is used to implement Direct Digital Control (DDC) in our process. It also works as supervisory control for the UDC 500 microprocessor. The DDC

system is comprised of 5 parts: the Apple microcomputer, ISAAC, the Labsoft software, the I-130 interface card and the I-140 pre-amplifier.

3.2.1 APPLE Computer

The computer used for our experiments is an Apple II Plus microcomputer with 48K of RAM (Random Access Memory). Some computer-related peripherals are used as well: a Panasonic model CT-1350MGC color video monitor, an Epson model MX-70 dot matrix printer, and 2 Apple model Disk II disc-drives.

3.2.2 ISAAC

ISAAC is manufactured by Cyborg Corporation; we use the model 91A. This device is a collection of hardware that translates signals originated from the process into digital values capable of being interpreted by the computer, and vice versa, translating digital signals originated from the computer into electrical signals that can be sent to the process for control purposes. This machine presents other capabilities, but for our purposes, A/D and D/A conversion are the most extensively used. The ISAAC itself is divided into smaller units.

Interface Card:

This card is plugged into one of the Apple's expansion slots. All the communication between the ISAAC and the Apple is handled through the interface card, which contains ad-

dress decoding and other communication logic. The card also contains a 16-bit count-up timer and an asynchronous real-time clock which is backed-up by a battery. The power drawn by the ISAAC Unit is also received from the Apple through the card. The interface contains 2 directional ports, each 1 word (2 bytes) wide, through which the Apple sends data and commands to the ISAAC, and the ISAAC sends status information and data back to the Apple.

Main Board:

This PC (Printed Circuit) board contains the power source (power drawn from the Apple), Input/Output (I/O) systems, the expansion slots, the reference voltage and many DIP (Dual Inline Package) switches. These switches allow the system to be reconfigured (e.g. changing operating voltages). As for the analog inputs, the on-board analog device contains a 12-bit A/D converter and a multiplexer to handle 16 analog inputs. As for analog outputs, there are four 12-bit D/A converters.

Input/Output (I/O) Distribution Panel:

All the external wiring originated from the process (analog inputs), as well as the outputs from the ISAAC-Apple to the system (analog outputs), are connected to this panel. The distribution panel is a componentless PC board that contains 96 screw-clamp terminals and is plugged into the ISAAC main board.

3.2.3 Labsoft Software

Labsoft is a data acquisiton and control oriented language, which is provided as part of the ISAAC system. In spite of not being a physical entity, the Labsoft software package is one of the most important items of the control system, being responsible for the full integration of the ISAAC unit with the Apple computer at both the hardware and software levels. The Labsoft Object File is automatically loaded in the Apple program memory when the Disc Operating System (DOS) is booted. The Apple resident high-level language is Applesoft, which is an extension of BASIC. The Applesoft-BASIC language is quite cumbersome and limited for real-time programming. Its capabilities for color graphics and for data acquisition and control are quite limited. However, the Applesoft-BASIC is provided with an expansion command, the ampersand (&), which preceeds all Labsoft commands. Program instructions that begin with an "&" cause the program to jump to a specific memory location where the Labsoft Object File resides. The Labsoft instruction is executed and the program pointer returns to the next program instruction (see section 3.4.5).

3.2.4 I-130 Board

The I-130 board is an interface card manufactured by Cyborg Corporation that is plugged into one of the ISAAC expansion slots. It is an I/O device that provides channel selection

logic, 12-bit A/D conversion and on-board programmable cold junction compensation for various types of thermocouple inputs. The I-130 is completely accessible from the command level so that we can program the channel we want to access, set the desired type of cold junction compensation and thermocouple gain (both depend on the type of thermocouple being used), etc. The I-130 works together with the I-140 pre-amplifier board.

3.2.5 I-140 board

The I-140 board is a peripheral signal conditioner also manufactured by Cyborg Corporation. Since it is not an I/O module, it cannot be accessed directly at the program level, but is accessed indirectly through the I-130 board. The job of the I-140 preamplifier board is to provide input protection, isolation, multiplexing and amplification for the thermocouple signal. This is necessary because the thermocouple signals, in the millivolt range, are easily corrupted by electrical noise from the surroundings, and also because these signals are not strong enough to be read directly by the ISAAC. The I-140 is installed on a separate housing, being connected to the I-130 board through a 25 pin I/O cable, from which it also gets its power. The thermocouple wires are connected directly to the I-140 board. The I-130 interface card is used to control the I-140 logic functions and to convert its analog amplified signals into digital form. For an overview of the whole DDC system, see Figure 3.3 .

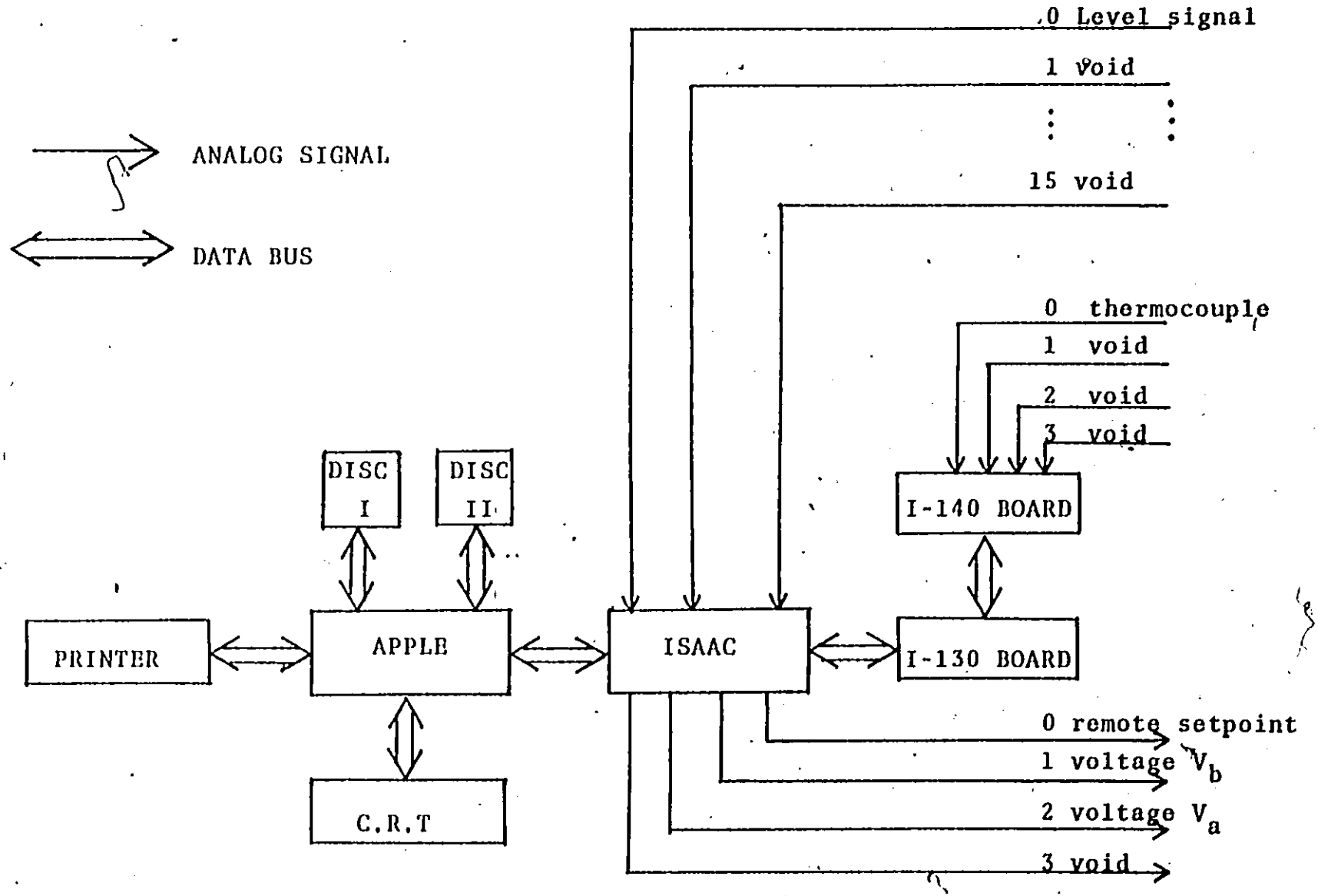


Figure 3.3:ISAAC - APPLE System.

3.3 THEORETICAL ANALYSIS OF THE PROCESS

Our two-loop process (recall Figure 3.1) can be represented by the closed-loop block diagram in Figure 3.4 (disregarding disturbances), where the process transfer functions are:

	Output	Input
$g_{11}(s)$	L(s)	$V_a(s)$
$g_{12}(s)$	L(s)	$V_b(s)$
$g_{21}(s)$	T(s)	$V_a(s)$
$g_{22}(s)$	T(s)	$V_b(s)$

and can be summarized in a more condensed form, using matrix notation as $\underline{G}(s)$ where

$$\underline{G}(s) = \begin{bmatrix} g_{11}(s) & g_{12}(s) \\ g_{21}(s) & g_{22}(s) \end{bmatrix}$$

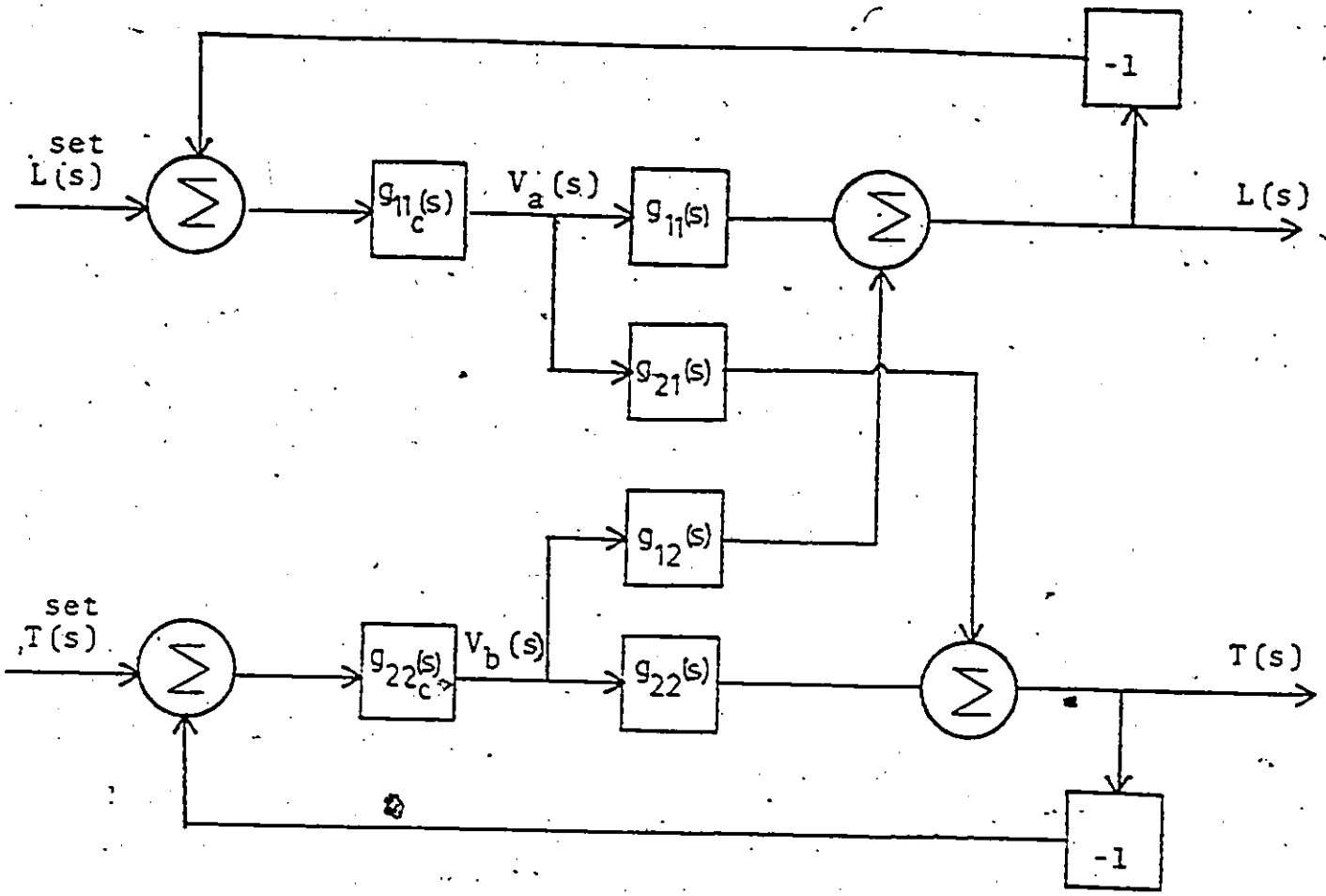


Figure 3.4: Theoretical Level/Temperature Block Diagram.

$g_{11_c}(s)$ - level controller

$g_{22_c}(s)$ - temperature controller,

$V_a(s)$ - output from level controller and

$V_b(s)$ - output from temperature controller.

Regarding the process presented in Figure 3.1, the following assumptions were considered for the mathematical modelling of the process:

1. The tank is perfectly mixed, so that there are no spatial gradients in temperature within the tank.
2. The cold and hot inlet water temperatures are constant.
3. Valves A and B have linear installed characteristics.
4. The dynamics of all sensors and transmitters are negligible.
5. Dead-time in the temperature loop is so small that it can be ignored.
6. Density and heat capacity are constant throughout the process.
7. Tank heat losses are negligible.
8. Linearized process equations adequately describe the process around its steady-state region.

3.3.1 System Equations

The steady-state values are as follows:

$\bar{F}_h$ = 7.6 L/min $\bar{T}_h$ = 60.0 °C

$\bar{F}_c$ = 7.3 L/min $\bar{T}_c$ = 10.0 °C

$\bar{F}$ = 14.9 L/min $\bar{T}$ = 35.5 °C

$\bar{V}_a$ = 2.63 VDC $\bar{V}_b$ = 2.00 VDC

K_a = 2.80 $dm^{5/2}$/VDC min K_b = 5.33 dm^3/VDC min

$\bar{L}$ = 9.46 dm A = 6.61 dm^2

L_o = 6.90 dm

where A = constant = cross-sectional area of the tank,

L_o = constant = vertical distance between the bottom of the tank and valve A.

The inputs are F_h, T_h, V_a, V_b while the process responses are F, T, L and F_c. Thus four equations are required to describe the process. These are:

Valve A equation (from Appendix B):

$$F = -7.77 + K_a V_a \sqrt{L} \qquad (B.3)$$

Valve B equation (from Appendix C):

$$F_c = -3.33 + K_b V_b \qquad (C.3)$$

Material balance:

$$A\, d(L-L_o)/dt = F_h + F_c - F \qquad (3.1)$$

Energy balance:

$$A\, d((L-L_o)T)/dt = F_h T_h + F_c T_c - F T \qquad (3.2)$$

Linearizing the equations above about the steady-state conditions,

$$F = -7.77 + K_a \bar{V}_a \sqrt{\bar{L}} + \frac{K_a \bar{V}_a}{2\sqrt{\bar{L}}}(L - \bar{L}) + K_a \sqrt{\bar{L}}\,(V_a - \bar{V}_a)$$

$$F_c = -3.33 + K_b V_b$$

$$A\frac{dL}{dt} = F_h + F_c - F$$

$$A(\bar{L}-L_o)\frac{dT}{dt} + A\bar{T}\frac{dL}{dt} = \bar{F}_h\bar{T}_h + \bar{F}_c\bar{T}_c - \bar{F}\,\bar{T} + \bar{F}_h(T_h-\bar{T}_h) + \bar{T}_h(F_h-\bar{F}_h) + \bar{T}_c F_c - \bar{F}(T - \bar{T}) - \bar{T}(F - \bar{F})$$

In terms of the perturbation variables $L^p = L-\bar{L}$, $V_a^p = V_a-\bar{V}_a$, etc. these equations become

$$F^p = \frac{K_a \bar{V}_a}{2\sqrt{\bar{L}}} L^p + K_a \sqrt{\bar{L}}\, V_a^p$$

$$F_c^p = K_b V_b^p$$

$$A\frac{dL^p}{dt} = F_h^p + F_c^p - F^p$$

$$A(\bar{L}-L_o)\frac{dT^p}{dt} + A\bar{T}\frac{dL^p}{dt} = \bar{F}_h T_h^p + \bar{T}_h F_h^p + \bar{T}_c F_c^p - \bar{F}\,T^p - \bar{T}\,F^p$$

For convenience the superscript 'p' will be dropped and the use of perturbation variables is implied in further development unless otherwise indicated. Thus, dropping superscripts and Laplace transforming yields

$$F(s) = \frac{K_a \bar{V}_a}{2\sqrt{\bar{L}}} L(s) + K_a \sqrt{\bar{L}}\, V_a(s)$$

$$F_c(s) = K_b\, V_b(s)$$

$$A\,s\,L(s) = F_h(s) + F_c(s) - F(s)$$

$$A(\bar{L}-L_o)sT(s) + A\bar{T}sL(s) = \bar{F}_h T_h(s) + \bar{T}_h F_h(s) + \bar{T}_c F_c(s) - \bar{F}\,T(s) - \bar{T}\,F(s)$$

The required four system equations can thus be summarized using matrix notation as:

$$\begin{bmatrix} 1 & 0 & -\frac{K_a \bar{V}_a}{2\sqrt{\bar{L}}} & 0 \\ 0 & 1 & 0 & 0 \\ 1 & -1 & A\,s & 0 \\ \bar{T} & -\bar{T}_c & A\bar{T}s & A(\bar{L}-L_o)s+\bar{F} \end{bmatrix} \begin{bmatrix} F(s) \\ F_c(s) \\ L(s) \\ T(s) \end{bmatrix} = \begin{bmatrix} K_a \sqrt{\bar{L}}\, V_a(s) \\ K_b\, V_b(s) \\ F_h(s) \\ \bar{F}_h T_h(s) + \bar{T}_h F_h(s) \end{bmatrix} \tag{3.3}$$

3.3.2 Open Loop Transfer Functions

The system equations above will allow us to obtain the transfer funcions (i.e. the dynamic relationships) between the controlled variables, L and T, and the manipulated variables, V_a and V_b. Solving eqn. (3.3) for F(s), F_c(s), L(s) and T(s) yields

$$F(s) = \frac{K_a \bar{V}_a}{2\sqrt{\bar{L}}} L(s) + K_a \sqrt{\bar{L}}\, V_a(s)$$

$$F_c(s) = K_b\, V_b(s)$$

$$L(s) = \frac{F_h(s)}{As + \frac{K_a \bar{V}_a}{2\sqrt{\bar{L}}}} + \frac{K_b}{As + \frac{K_a \bar{V}_a}{2\sqrt{\bar{L}}}} V_b(s) - \frac{K_a\sqrt{\bar{L}}}{As + \frac{K_a \bar{V}_a}{2\sqrt{\bar{L}}}} V_a(s)$$

$$T(s) = \frac{(\bar{T}_h - \bar{T})}{A(\bar{L}-L_o)s+\bar{F}} F_h(s) + \frac{K_b(\bar{T}_c-\bar{T})}{A(\bar{L}-L_o)s+\bar{F}} V_b(s) + \frac{\bar{F}_h}{A(\bar{L}-L_o)s+\bar{F}} T_h(s)$$

from which the required transfer functions are obtained as:

$$g_{11}(s) = \frac{L(s)}{V_a(s)} = -\frac{\bar{K}_a\sqrt{\bar{L}}}{As + \frac{K_a \bar{V}_a}{2\sqrt{\bar{L}}}} = -\frac{2\bar{L} \,/\, \bar{V}_a}{\frac{2A\sqrt{\bar{L}}}{K_a \bar{V}_a} s + 1} \qquad (3.4)$$

$$g_{12}(s) = \frac{L(s)}{V_b(s)} = \frac{K_b}{As + \frac{K_a \bar{V}_a}{2\sqrt{\bar{L}}}} = \frac{2\, K_b\sqrt{\bar{L}}/K_a \bar{V}_a}{\frac{2A\sqrt{\bar{L}}}{K_a \bar{V}_a} s + 1} \qquad (3.5)$$

$$g_{21}(s) = \frac{T(s)}{V_a(s)} = 0 \qquad (3.6)$$

$$g_{22}(s) = \frac{T(s)}{V_b(s)} = \frac{K_b(\bar{T}_c - \bar{T})}{A(\bar{L}-L_o)s+\bar{F}} = \frac{K_b(\bar{T}_c-\bar{T})/\bar{F}}{\frac{A(\bar{L}-L_o)}{\bar{F}}s+1} \qquad (3.7)$$

Substituting numerical values in the transfer functions above yields:

$$g_{11}(s) = \frac{-7.19}{5.52\ s + 1} \quad (dm/VDC)$$

$$g_{12}(s) = \frac{4.45}{5.52\ s + 1} \quad (dm/VDC)$$

$$g_{21}(s) = 0 \quad (^oC/VDC)$$

$$g_{22}(s) = \frac{-9.12}{1.14\ s + 1} \quad (^oC/VDC)$$

Some conclusions can be drawn from these results:

1. The design requirement for the process to be open-loop stable was achieved. All the poles of the transfer functions lie on the left-hand side of the imaginary plane.
2. Time constants in the process are in the order of 1 to 5 minutes, thus the process responds fast enough for lab demonstration purposes.

3. The system equations used to develop the transfer functions were linearized about the steady-state conditions. Therefore, the transfer function models are, strictly speaking, valid only for the process operating around the design region. We can expect the models to be less adequate, the more the process is moved away from its design region. For example, for $g_{11}(s)$ and $g_{12}(s)$, both the process gain K_p and the time constant τ_p depend on L and V_a, as can be seen from eqns. (3.4) and (3.5). For $g_{22}(s)$, according to eqn. (3.7), K_p depends on T_c, T and F while τ_p depends on L and F.

   Some numerical examples of these nonlinearities are shown in Table 3.1. These results show that the nonlinearities can be very significant. The transfer function $g_{22}(s)$ is more sensitive to changes in the operating region, especially changes in T_c and T.

4. Since $g_{21}(s) = 0$, there is no relationship between the temperature in the tank, T, and the voltage signal to valve A, V_a. Coupling is only in one direction and conventional tuning can be carried out.

TABLE 3.1

Numerical Examples of Nonlinearities.

Transfer Function	At Design Region		Away from Design Region	
$g_{11}(s) = \frac{L(s)}{V_a(s)}$ $(\frac{dm}{VDC})$	$\frac{-7.19}{5.52\ s + 1}$	$\bar{L} = 9.46$ dm $\bar{V}_a = 2.63$ VDC	$\frac{-10.46}{7.64\ s + 1}$	$\bar{L} = 10.46$ dm $\bar{V}_a = 2.00$ VDC
$g_{12}(s) = \frac{L(s)}{V_b(s)}$ $(\frac{dm}{VDC})$	$\frac{4.45}{5.52\ s + 1}$	$\bar{L} = 9.46$ dm $\bar{V}_a = 2.63$ VDC	$\frac{6.16}{7.64\ s + 1}$	$\bar{L} = 10.46$ dm $\bar{V}_a = 2.00$ VDC
$g_{22}(s) = \frac{T(s)}{V_b(s)}$ $(\frac{^oC}{VDC})$	$\frac{-9.12}{1.14\ s + 1}$	$\bar{T}_c = 10\ ^oC$ $\bar{T} = 35.5^oC$ $\bar{F} = 14.9$ L/min $\bar{L} = 9.46$ dm	$\frac{-19.19}{2.35\ s + 1}$	$\bar{T}_c = 4^oC$ $\bar{T} = 40^oC$ $\bar{F} = 10$ L/min $\bar{L} = 10.46$ dm

3.4 CONTROL SOFTWARE

The control software developed for this system comprises three control programs. The first two implement level-only control, whereas the third one implements simultaneous level/temperature control.

All programs were written in Applesoft-Basic expanded with Labsoft commands, which are recognized by the ampersand (&) that precedes them. The Labsoft commands most frequently used in our control software are presented on Table 3.2.

Sampling time for all the programs is controlled by the ISAAC-Apple hardware timer, which can measure intervals of time up to 16,383 milliseconds. Sampling time must be greater than the time spent by the computer in interpreting and executing the instructions (i.e. program overhead) plus the hardware delays (i.e. D/A and A/D conversion). For example, the graphing routine takes an average time of 10 milliseconds to be executed, D/A conversion takes 5 microseconds and A/D conversion takes 25 microseconds. For the thermocouple reading, there is a 2.5 millisecond hardware delay. Our slowest program is executed in 0.98 seconds, including both the software and the hardware delay. For the reasons above, sampling time was limited to the range 1.0 to 16.0 seconds. It is possible to obtain sampling times greater than 16 seconds by restarting the timer, but this has not been attempted in this project.

TABLE 3.2

Most often used Labsoft commands.

LABSOFT COMMAND	LABSOFT SYNTAX	PARAMETERS	TASK
ANALOG INPUT	&AIN	TV-Target Variable C#-Channel number	Inputs an anlog value into the target variable through A/D channel specified.
ANALOG OUTPUT	&AOUT	DV-Data Value C#-Channel number	Outputs an anlog value equal to the data value through D/A channel specified.
CLEAR TIMER	&CLRTIMER	None	Resets the timer and begins counting in milliseconds.
LOOK FOR TIMER	&LOOK FOR TIMERIN	TV-Target Variable TH-Threshold	Continuously checks whether the elapsed time (target variable) has reached the threshold (sampling time).
SCROLLING GRAPH	&SCROLLSET	None	Initializes the scrolling graph format.
PLOT FORMAT	&PLTFMT	Color codes	Specifies the sequence of colors to be used when plotting multivariable inputs on an analog graph.
GRAPH OUTLINE	&OUTLINE	None	Outlines the currently initialized analog graph.
NEXT PLOT	&NXTPLT	Arithmetic expression	Plots the value of the arithmetic expression on the currently defined analog graph.
WRITE DEVICE	&WRDEV	DV-Expansion slot # D#-Cold Junction compensation code	Write Device is an expansion command used to specify the type of cold junction compensation to be used.

It was not our objective to study the effect of sampling time on the control system, but rather approximate a continuous system. For our experiments, sampling time was usually set equal to 2 seconds. Since the process time constants are in the order of 1 to 5 minutes, such a sampling time approximates a continuous system.

3.4.1 PID Level Control with Position Algorithm

This control program implements single-input single-output (SISO) control. Execution time for the main program loop is 0.34 seconds, and the essential program length comprises 42 statements. Its flow chart, together with a heavily remarked program listing, is shown in Appendix G.

The control algorithm is

```
560 M = MR + KC * (E1 + IERR/ITAU + DTAU*DERR) * (-1651)
```

This is a POSITION PID algorithm, where

560 = the statement number,

M = controller output between 0 and 4095, which corresponds to a 0 to 5 VDC analog range,

MR = 2048 = bias term, corresponding to 2.5 VDC,

KC = controller gain,

E1 = Error, $L^{set} - L$,

IERR = rectangular integration approximation to the integral of the error,

ITAU = reset time,

DERR = finite difference approximation to the deri-

vative of the error,

DTAU = derivative time and

1651 = scaling factor, ratio between the full digital range (0 to 4095) and the full analog level range (0 to 2.48 dm). Its negative sign ensures negative feedback in the control loop.

This program allows the user to implement P, PI or PID control. Individual effects of controller gain K_c, integral time and derivative time can be shown. Determination of the ultimate gain, Ku, and ultimate period of oscillation, Pu, can also be obtained. Anti-reset wind-up can be put in effect by the user, who can also write his own control algorithm.

3.4.2 PID Level Control with Velocity Algorithm

This program is very similar to the PID Level Control with Position Algorithm, except for the control algorithm. There are 34 essential program statements and the main program loop has an execution time of 0.33 seconds. The program flow chart and listing are shown in Appendix H. The velocity algorithm is not as versatile as the position algorithm, in the sense that the former cannot work with P-only control, which is undesirable for controller tuning purposes. As an advantage, anti-reset windup is incorporated in the control algorithm itself. The control algorithm command is

```
540  M1 = M0 + KC * ((E2-E1) + ST / ITTAU * E2 + DTAU / ST
* (E2 - 2 * E1 + E0)) * (-1651)
```

This is the VELOCITY PID algorithm, where

M1 = controller output at present time,

M0 = controller output at previous sampling time,

E2 = present error ($L^{set}-L$),

E1 = error 1 sampling time ago and

E0 = error 2 sampling times ago.

3.4.3 PID Level and Temperature Control

This is the longest program, with 106 essential statements and 0.98 seconds execution time for the main program loop. It implements simultaneous control of level and temperature in the tank. The program allows the choice of velocity or position PID algorithms for both the level and temperature. The flow chart of the program, together with its listing, is shown in Appendix I. The level POSITION and VELOCITY PID algorithms are the same as the ones presented for level-only control. The temperature control POSITION PID algorithm is

```
1080  N1 = NR + KT * (T2ERR + TIERR / ITTAU + TDERR *
DTTAU) *(-409.5) * 0.67
```

where

N1 = output from temperature controller, in the range 0 to 4095, which is equivalent to a 0 to 5 VDC analog range,

NR = bias term (= 1638), which is equivalent to

2 VDC,

KT = controller gain,

T2ERR = error at present time (T^{set} - T),

T1ERR = rectangular integration approximation to the integral of the error,

ITTAU = reset time,

TDERR = finite difference approximation to the derivative of the error,

DTTAU = derivative time,

409.5 = scaling factor, is the ratio of the digital range (0 to 4095) over the temperature workable range (25 to 35°C). Its negative sign ensures negative feedback in the control loop and

0.67 = constant, limits the valve opening at 2/3 of its maximum.

The VELOCITY PID algorithm is executed by the command

```
1100   N1 = N0 + KT * ((T2ERR - T1ERR)  + ST / ITTAU * T2ERR
+ DTTAU / ST * (T2ERR - 2 * T1ERR + T0ERR)) *(-409.5) * 0.67
```

where

N1 = present output from temperature controller,

N0 = previous output from temperature controller,

T2ERR = error at present sampling time,

T1ERR = error one sampling time ago and

T0ERR = error two sampling times ago.

3.4.4 Dynamic Decoupling

It will be shown in Section 4.1 that our experimental open-loop transfer function is:

$$\underline{G}(s) = \begin{bmatrix} \frac{-14.25}{9.55\,s + 1} & \frac{3.23}{7.87\,s + 1} \\ 0 & \frac{-7.68}{1.24\,s + 1} \end{bmatrix}$$

Coupling arises in the system because of the off-diagonal term in $\underline{G}(s)$. Since $K_{21} = 0$ (see section 2.4.6), the Bristol array yields:

$$\Lambda = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

which is expected from a triangular matrix. This means that there is no serious coupling and that the loops can be tuned using classical control theory. However, even such a system can have its performance improved with a suitable decoupler[1]. From Chapter 2, the dynamic decoupler is defined as:

$$\underline{G}_I(s) = \underline{G}(s)^{-1} \text{ diag } \underline{G}(s)$$

Thus,

$$\underline{G}_I(s) = \begin{bmatrix} \frac{9.55s+1}{14.25} & \frac{0.03(9.55s+1)(1.24s+1)}{(7.87s+1)} \\ 0 & \frac{1.24s+1}{7.68} \end{bmatrix} \begin{bmatrix} \frac{14.25}{9.55s+1} & 0 \\ 0 & \frac{7.68}{1.24s+1} \end{bmatrix}$$

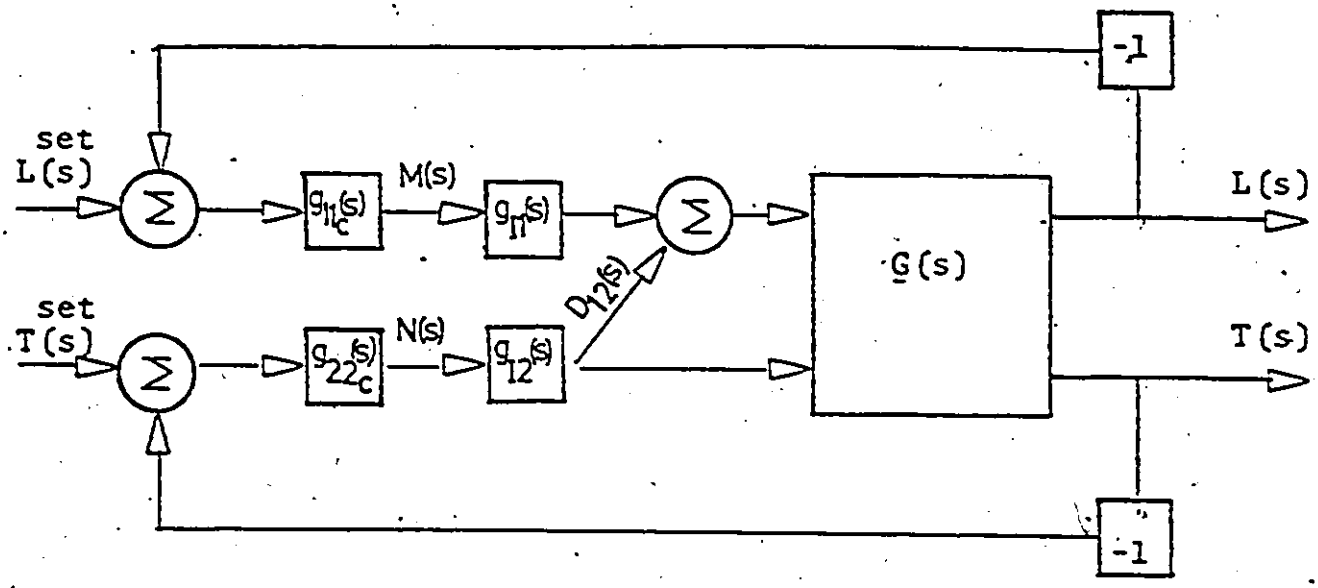
$$\underline{G}_I(s) = \begin{bmatrix} 1 & \frac{0.23(9.55s+1)}{(7.87s+1)} \\ 0 & 1 \end{bmatrix}$$

The dynamic decoupler matrix $\underline{G}_I(s)$ contains a lead/lag element. Ignoring the dynamic part, we obtain the steady-state decoupler matrix $\underline{G}_I(s)_{s.s.}$.

$$\underline{G}_I(s)_{s.s.} = \begin{bmatrix} 1 & 0.23 \\ 0 & 1 \end{bmatrix}$$

The system block diagram with the dynamic decoupler is shown in Figure 3.5 . It is important to notice at this point that the block diagrams shown in Figures 3.4 and 3.5 are equivalent, except for the insertion of the decoupler in the latter.

All the decoupling is carried out by $g_{I2}(s)$, since $g_{I1}(s)$ does not change the closed-loop dynamic path. The decoupler can be written as:



$$g_{I1}(s) = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad g_{I2}(s) = \begin{bmatrix} 0.23\,\dfrac{9.55\,s + 1}{7.87\,s + 1} \\ 1 \end{bmatrix}$$

$D_{12}(s)$ = output from decoupler.

Figure 3.5: System Block Diagram with Dynamic Decoupler.

$$\frac{D12}{N} = 0.23\ \frac{(9.55\ s + 1)}{(7.87\ s + 1)} \qquad (3.8)$$

where D12 = digital decoupler output signal

N = digital temperature controller output signal.

This continuous lead-lag element can be approximated by a finite-difference equation(*) in the time domain. First, it has to be written in terms of s^{-1} :

$$\frac{D12}{N} = 0.23\ \frac{(9.55 + s^{-1})}{(7.87 + s^{-1})} \qquad (3.9)$$

Then we substitute the trapezoidal integration approximation for s^{-1} :

$$s^{-1} = \frac{S_t}{2}\ \frac{1 + z^{-1}}{1 - z^{-1}} \qquad (3.10)$$

where S_t = sampling time in seconds and

z = complex variable such that $z = e^{S_t s}$.

Combining eqns. (3.9) and (3.10) yields:

$$\frac{D12}{N} = 0.23\ \frac{\{9.55(1-z^{-1}) + (S_t/2)(1+z^{-1})\}}{\{7.87(1-z^{-1}) + (S_t/2)(1+z^{-1})\}}$$

Since z^{-1} implies a backward shift operation then

$$z^{-1} D12_t = D12_{t-1}$$

and $z^{-1} N_t = N_{t-1}$.

$$D12_t = \frac{N_t(2.2+0.115\ S_t)-N_{t-1}(2.2-0.115\ S_t)+D12_{t-1}(7.87-S_t/2)}{7.87 + S_t/2} \qquad (3.11)$$

Equation (3.11) is the discrete approximation of the dynamic decoupler. It was incorporated to the level and temperature control program that has been presented in subsection 3.4.3. The command is:

```
1160  D1 = (N1 * (2.2 + 0.115 * ST)  - N0 * (2.2 - 0.115 *
ST) + D0 * (7.87 - ST/2)) / (7.87+ST/2)
```

where D1 = present decoupler output,
D0 = previous decoupler output,
N1 = present temperature controller output,
N0 = previous temperature controller output and
ST = sampling time in seconds.

When the program starts (at t = 0), N0 is made equal to its steady-state value (= 1638), which corresponds to a 2 VDC output. D0 is made equal to zero (no decoupling).

When dynamic decoupling is not in effect, the bias term MR in the PID position algorithm is equal to 2048, which corresponds to 2.5 VDC. Once the dynamic decoupler is in effect, it will output a steady-state signal equal to 0.23 * N1 = 376.74, which is equivalent to 0.46 VDC. This will cause an

offset in the level if not corrected. Therefore, whenever decoupling is in effect, MR is set equal to MR = 2048 - 376.74 = 1671.

3.4.5 Computer Memory Usage

Our Apple microcomputer has a 48K byte Random Access Memory (RAM), represented by a stack as in Figure 3.6 . The lowest 2K bytes are reserved for BASIC system use (i.e. low resolution graphics, text screen, etc). The following 6K bytes are reserved for BASIC user's programs. This is the memory area in which our control programs are loaded. The memory addresses between 8K and 16K are reserved for High Resolution Graphics (HGR) page 1, and the area between 16K and 24K is reserved for High Resolution Graphics (HGR) page 2. The Disk Operating System (DOS) is loaded from the top of the stack down, which corresponds to memory addresses between 48K and 38K. Labsoft is loaded just under DOS, taking also 10K bytes (i.e. from 38K to 28K). Our longest program takes, without the remarks, 5.6K bytes out of the 6K bytes of memory available for programs. As a consequence, only one or two more control loops or algorithms may be added to this program before the computer runs out of memory.

To circumvent this problem, the best solution is to reset to 16K the highest memory address available to the user's program, which is currently set at 8K. This is carried out by issuing the Applesoft command "HIMEM". Resetting

HIMEM to 16K would eliminate HGR page 1, but HGR page 2 could be used instead. If space is still a problem, HIMEM could be reset to 28K, and therefore high resolution graphics would no longer be available. On the other hand, the additional 20K of memory would allow the implementation of many more loops and/or algorithms.

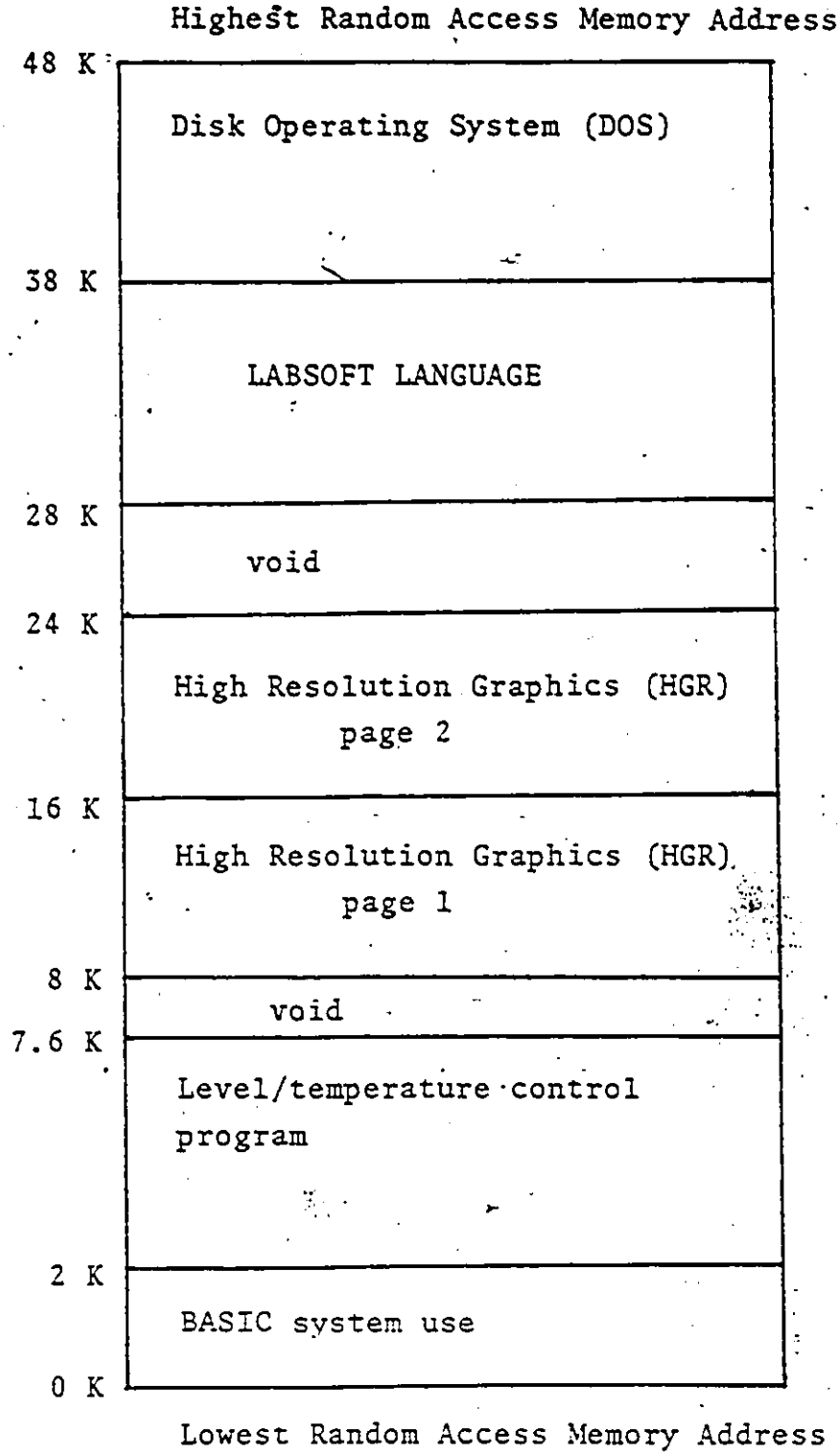


Figure 3.6:Computer memory usage map.

Chapter IV

EXPERIMENTAL EVALUATION OF THE CONTROL SYSTEM

In this chapter we will experimentally determine the open-loop process transfer functions, which will be compared with the theoretical models obtained in Chapter 3. The performance of the control software developed previously will also be tested. Finally, the dynamic decoupler will be tested and its performance compared with other decoupling techniques.

4.1 DETERMINATION OF OPEN LOOP TRANSFER FUNCTIONS

Experimental process identification is necessary for process control purposes when a mechanistic model of the process is not available. It is also a way of checking the adequacy of the mechanistic model if it is available.

In our case, experimental modelling was carried out to give us a more realistic model which could be used for designing the decoupler. It also provided a way of verifying the theoretical model and the extent of the nonlinearities, since the theoretical model was based on a number of simplifying assumptions and linearized equations. It was also necessary to confirm the suitability of the system for lab experiments.

Luyben[8] presents some off-line identification techniques. The most common ones, in increasing order of complexity, are: step-testing, direct sine-wave testing and pulse testing.

On-line identification techniques are also available[21,22], in which the process is periodically identified while running in closed-loop. The major application of on-line identification is in Adaptive Control. On-line identification requires the process to be running under digital computer control. Off-line techniques can be carried out with standard analog hardware, as well as digital.

From all the techniques above, step-testing is by far the simplest. It does not take much time, nor does it require special hardware, and process parameters can be directly obtained from the process reaction curve. Our process was identified using step-testing, and the parameters were evaluated using a two-point method, which will be explained in the following subsection.

4.1.1 Step Testing

The procedure consisted of opening the loop we were interested in, waiting for the process to reach steady-state, and then making a step change in the manipulated variable (V_a or V_b). If any other loop affected the output variable of interest, it was also opened. The response of the output variable to a step change in the input (called the process reac-

tion curve) was then recorded for further analysis using the Watanabe strip chart recorder described in section 3.1.10 . Care was taken that no load disturbances occurred during the duration of the test. The proposed time domain model was then fitted to the reaction curve data using a two-point method. For a first-order-plus-deadtime model, the process parameters can be determined directly using this method.

4.1.2 Two-point method

The response curve of a first-order process to a step change in input is given (using perturbation variables) by:

$$x(t) = \bar{Q} K_p (1 - e^{-t/\tau_p})$$

where

$x(t)$ = change in output,

$\bar{Q}$ = magnitude of the stepchange in input,

K_p = process gain and

τ_p = process time constant.

The process gain K_p is found by calculating the ratio of the steady-state change in the value of output over input:

$$K_p = x(\infty)/\bar{Q}$$

Our equation can therefore be written as:

$$x(t) = x(\infty) (1 - e^{-t/\tau_p})$$

When $t = 0$ $\qquad x(t) = 0$

When $t = \tau_p/3$ $\quad x(t) = 0.283\ x(\infty)$

When $t = \tau_p$ $\quad x(t) = 0.632\ x(\infty)$

Therefore, for a first-order process;

$$t_{0.283} = \tau_p/3 \qquad (4.1)$$

$$t_{0.632} = \tau_p \qquad (4.2)$$

where $t_{0.283}$= time for the output variable to reach 28.3% of its steady-state change.

$t_{0.632}$= time for the output variable to reach 63.2% of its steady-state change.

If dead-time is present, the reaction curve is delayed D units of time and equations (4.1) and (4.2) can be written:

$$t_{0.283} = \tau_p/3 + D \qquad (4.3)$$

$$t_{0.632} = \tau_p + D \qquad (4.4)$$

By simultaneously solving (4.3) and (4.4) we can determine D and τ_p.

The two-point fitting method has some limitations:

1. We are assuming that our process can be described by a first order plus deadtime system, which may not be strictly true.

2. Only 2 points of the reaction curve are actually used to estimate the model parameters and this can lead to error.
3. We are not taking into account the noise in the response variable signal, i.e., we are using a deterministic approach.

For our process, the theoretical model predicted first-order responses. Also, our experimental reaction curves resulted in well-behaved 1st-order plus deadtime exponentials. The other two limitations were dealt with by repeating the step-changes many times to check consistency of results.

4.1.3 Experimental Transfer Functions

Process testing was carried out using 3 computer programs which are shown in Appendix J. The first identification program, called "G11 RESPONSE CURVE", was written to identify $g_{11}(s) = L(s)/V_a(s)$. The dynamics of the level sensor/transmitter, air-pressure-to-voltage transducer and valve A were lumped into $g_{11}(s)$. The temperature loop was not implemented for the $g_{11}(s)$ testing, since we were interested only in the level response. The cold water stream was shut off ($F_c = 0$), whereas the hot water flowrate F_h was increased in order to keep the outlet flowrate F close to its design region. The process reaction curve was generated by positioning the level either very low or very high in the tank, waiting for the voltage signal V_a to valve A to become con-

stant, and introducing a positive stepchange in V_a if the level were initially high (to decrease it), or a negative stepchange if the level were initially low. Because of the high process gain, K_p, the stepchanges had to be very small (around 0.1 VDC), in order to keep the level response within a measurable range. This is the reason why it was necessary to begin the testing with the level at either extreme of its range. In the theoretical modelling in Chapter 3, it was shown that $g_{11}(s)$ is nonlinear and that its parameters depend on both L and V_a. Because of the limitations above, only $V_a(s)$ could be actually tested at different regions to determine nonlinearities. The model parameters τ_p, K_p and D were fitted to the process reaction curve using the 2-point method described above. A sample calculation is shown in Appendix K, the experimental results being condensed in Table 4.1 .

From equation (3.4), the absolute magnitudes of K_p and τ_p were expected to decrease for an increase in V_a. The experimental results in Table 4.1 , however, do not show any visible trend in this direction. Nonlinearities in K_p and τ_p may have been masked by the scatter in the data. Since the experimental model is needed for the designing of the decoupler, and since the decoupler is aimed to be used within a broad range of values of V_a, we decided to average both the process gain and the time constant.

TABLE 4.1

$g_{11}(s)$ Experimental Identification.

F_h (L/min)	INITIAL V_a (VDC)	STEPCHANGE IN V_a (VDC)	$t_{28.3\%}$ (min)	$t_{63.2\%}$ (min)	K_p (dm/VDC)	τ_p (min)	D (min)
7.6	1.66	0.12	5.50	13.17	-17.17	11.50	1.67
7.6	1.75	-0.09	4.75	11.58	-16.50	10.25	1.33
14.0	1.90	0.10	5.10	13.50	-13.85	12.60	0.90
11.7	2.17	0.11	3.58	9.58	-15.23	9.00	0.58
11.7	2.27	-0.12	4.00	9.25	-12.50	7.88	1.38
11.7	2.30	-0.10	3.67	10.17	-16.05	9.75	0.42
14.0	2.43	0.07	4.00	10.60	-13.86	9.90	0.70
14.0	2.44	0.08	3.38	8.88	-13.23	8.26	0.63
14.0	2.47	-0.05	3.84	9.47	-17.80	8.44	1.03
14.0	2.49	0.09	3.91	10.05	- 9.78	9.21	0.84
14.0	2.51	0.06	3.82	10.10	-13.90	9.42	0.68
14.0	2.51	-0.05	3.60	9.47	-12.40	8.80	0.67
14.0	2.52	0.09	3.54	8.64	-12.23	7.65	0.99
14.0	2.57	-0.13	3.83	11.17	-15.00	11.01	0.16

Although deadtimes in the range 0.16 to 1.67 minutes were observed in these open loop tests, under closed loop conditions no deadtime was observed (see section 4.2). This may have been caused by the small magnitude of voltage stepchange we were forced to work with (=< 0.1 VDC). Under closed loop conditions, this signal was typically 10 to 50 times higher, and the changes in L were more quickly detected. For small magnitude stepchanges, some time elapsed before any change in level could be detected by the level sensor-transmitter, resulting in an overestimated deadtime. Since it was our intention to operate the process under closed loop conditions, we decided not to include the deadtime in our model. Averaging the parameters K_p and τ_p and excluding the deadtime yields

$$g_{11}(s) = \frac{L(s)}{V_a(s)} = \frac{-(14.25 \pm 1.27)}{(9.55 \pm 0.82)s+1} \quad (dm/VDC)$$

where K_p and τ_p are shown in terms of a 95% confidence interval for their true means. Figure 4.1 shows a comparison between the theoretical and the full experimental band of $g_{11}(s)$ reaction curves.

Recall that our theoretical $g_{11}(s)$ was given by

$$g_{11}(s) = \frac{L(s)}{V_a(s)} = \frac{-7.19}{5.52\ s + 1} \quad (dm/VDC)$$

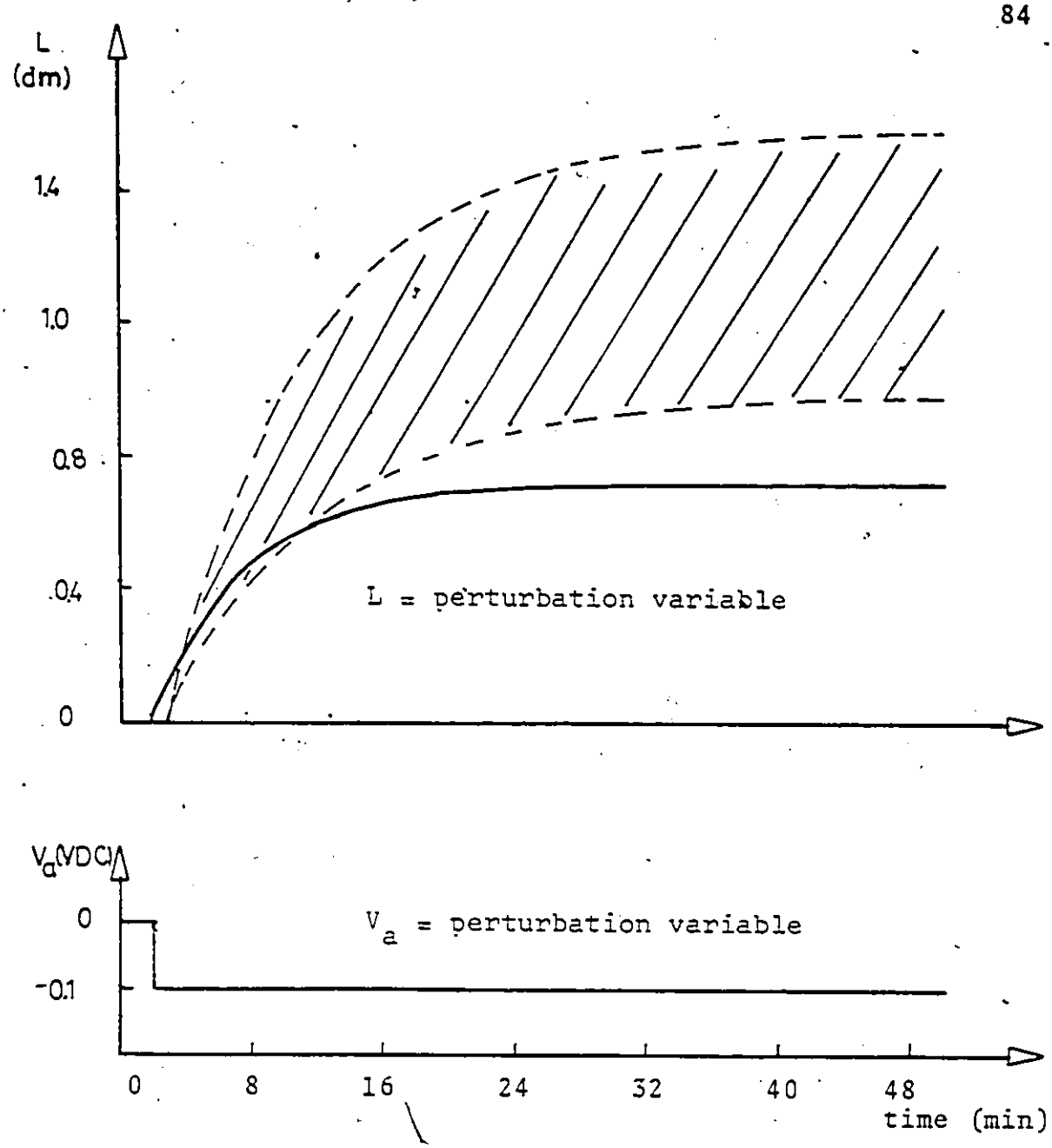


Figure 4.1: Theoretical and Experimental $g_{11}(s)$ reaction curves.

——— theoretical

band of experimental results.

The process gain K_p is approximately twice as big as predicted. No exact explanation for this has been found. It is suspected that it has been caused by the small magnitudes of voltage stepchanges introduced, associated with process nonlinearities. The time constant was also greater than predicted, but to a lesser extent. The difference has likely been caused by the combined effects of the dynamics of valve A, sensor, transmitter and transducer, which were all considered simple gains for the theoretical modelling.

The second identification program, called "G12 RESPONSE CURVE", identifies $g_{12}(s) = L(s)/V_b(s)$. The dynamics of the voltage-to-air-pressure transducer, valve B, the level sensor/transmitter and the air-pressure-to-voltage transducer were lumped in $g_{12}(s)$. For this test both hot and cold streams were used but again, since temperature was not recorded, the steam supply was kept shut off. The process reaction curve was generated by lining up the level at a very low or very high position with the level loop closed, while holding the voltage signal V_b to valve B constant. When V_a also became constant, the level loop was opened and a stepchange in V_b was introduced (positive, if the level were initially low and negative, if the level were initially high). Again, because of the high process gain, stepchanges had to be small (<0.5 VDC) to keep the level within a measurable range. From equation (3.5), both the process gain and time constant depend on $\bar{L}$ and $\bar{V}_a$. The level, however,

could not be tested at different regions, for the same reason it could not for $g_{11}(s)$. Thus, only V_a was tested at different regions, where an increase in V_a was expected to decrease both K_p and τ_p. As can be seen from Table 4.2, such a trend was not observed, probably being again masked by the scatter in the data.

Since nonlinearities did not become evident, we decided to average K_p and τ_p, computing also a 95% confidence interval for their true means.

Deadtimes in the range 0.17 to 0.96 minutes were observed in the open loop tests. Under closed loop conditions, however, no deadtime was observed (see section 4.3). The cause seems to be the same one that caused deadtime to appear in the $g_{11}(s)$ open-loop testing, i.e., the small magnitude voltage stepchanges. Small magnitude voltage stepchanges cause the level to rise or drop rather slowly, some time having elapsed before any change in level is detected by the level sensor-transmitter. Since process operation was intended to be conducted under closed loop conditions, deadtime was not included in the experimental $g_{12}(s)$ model, which was reduced to

$$g_{12}(s) = \frac{L(s)}{V_b(s)} = \frac{(3.23 \pm 0.44)}{(7.87 \pm 0.58)s + 1} \quad \text{(dm/VDC)}$$

The theoretical model was given by

TABLE 4.2

$g_{12}(s)$ Experimental Identification.

F(L/min)	V_a (VDC)	INITIAL V_b (VDC)	STEP - CHANGE IN V_b (VDC)	$t_{28.3\%}$ (min)	$t_{63.2\%}$ (min)	K_p (dm/VDC)	τ_p (min)	D (min)
11.30	2.27	1.2	0.6	3.33	8.08	3.34	7.13	0.96
12.20	2.37	1.4	0.6	3.25	8.83	3.50	8.38	0.46
13.00	2.43	1.6	0.4	3.17	8.25	1.72	7.63	0.63
13.80	2.50	1.3	0.5	2.92	8.00	4.52	7.63	0.38
14.20	2.55	1.4	0.5	3.33	8.58	2.90	7.88	0.71
14.30	2.57	1.9	-0.5	3.58	9.75	3.71	9.25	0.50
14.90	2.66	2.0	-0.5	3.33	9.33	3.60	9.00	0.33
15.40	2.70	1.7	-0.6	3.58	9.17	3.64	8.38	0.79
15.30	2.70	2.1	-0.5	3.50	9.33	3.35	8.75	0.58
17.80	3.00	1.2	0.7	2.58	7.17	3.15	6.88	0.29
21.20	3.66	2.0	-0.6	2.67	6.83	2.77	6.25	0.58
21.70	3.83	2.1	-0.6	2.58	7.42	2.59	7.25	0.17

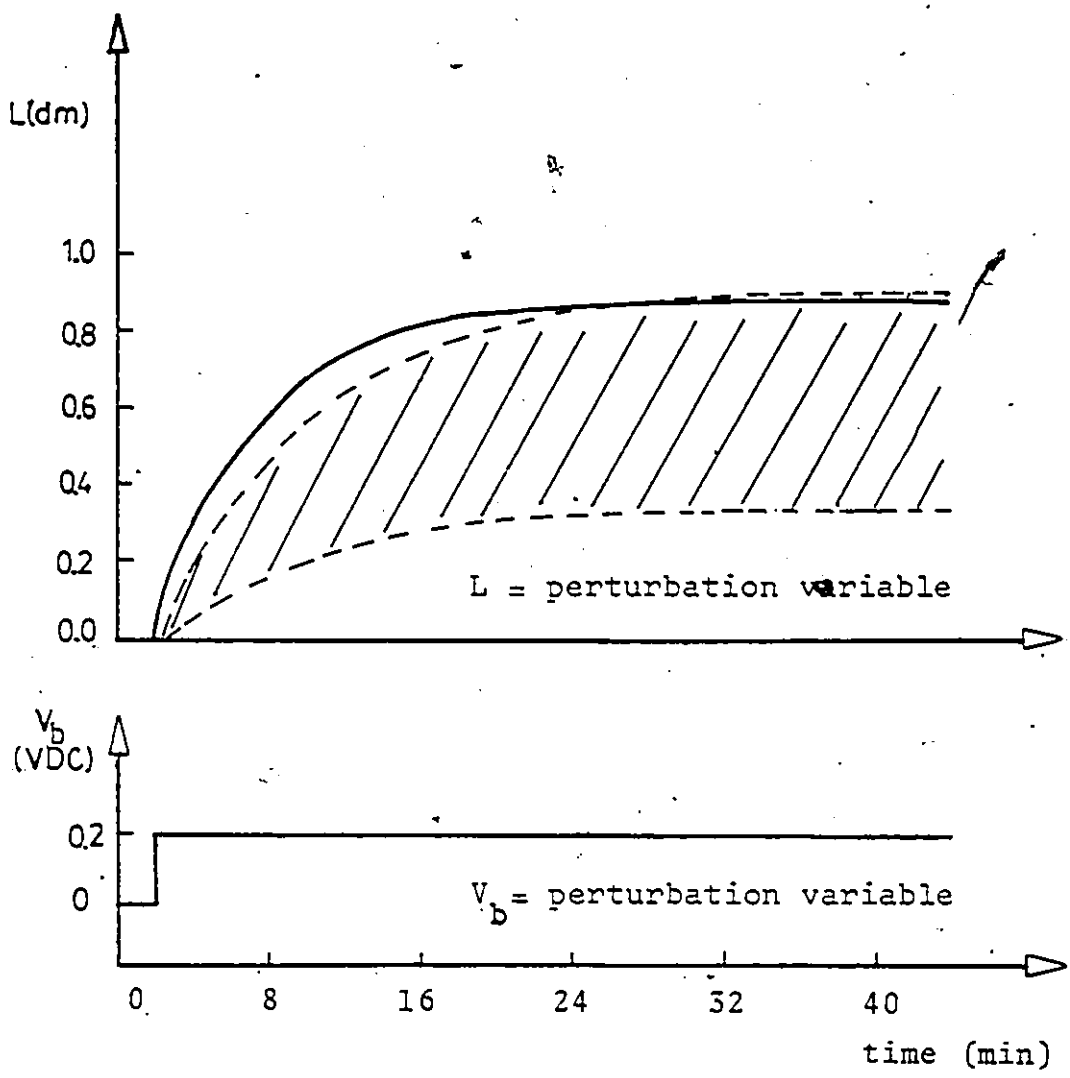


Figure 4.2 : Theoretical and Experimental $g_{12}(s)$ reaction curves.

——— theoretical

band of experimental results.

$$g_{12}(s) = \frac{L(s)}{V_b(s)} = \frac{4.45}{5.52\ s + 1} \quad \text{(dm/VDC)}$$

The process gain is close to the value expected theoretically, the small difference likely to have been caused by the small magnitude of stepchange associated with process nonlinearities. Neglected valve, sensor, transmitter and transducer dynamics for the theoretical model may explain the small discrepancy in the time constant. Figure 4.2 shows a comparison between the theoretical and full experimental band of $g_{12}(s)$ reaction curves.

For $g_{21}(s)$, results were as expected. As will be shown in section 4.3, temperature is not upset by changes in level setpoint.

The last identification program, called "G22 RESPONSE CURVE", identifies $g_{22}(s) = T(s)/V_b(s)$. The dynamics of the voltage-to-air-pressure transducer, valve B and the temperature sensor/transmitter were lumped into $g_{22}(s)$. This time both water streams had to be on, as well as the steam supply. The process reaction curve was generated by keeping the level under closed loop control while holding the voltage signal V_b to valve B constant, i.e. opening the temperature loop. Once the temperature lined up, a stepchange in V_b was introduced and the reaction curve recorded. The level loop was kept closed during the test because there is no coupling in the direction level to temperature. Model parameters were then fitted to the resulting reaction

curves. Experimental results are presented in Table 4.3 . From the theoretical modelling in Chapter 3, $g_{22}(s)$ was shown to be nonlinear, its parameters being a function of $\bar{T}, \bar{T}_c$, $\bar{L}$ and $\bar{F}$.

The process was tested for different $\bar{T}$ and $\bar{F}$, with $\bar{L}$ = 9.46 dm. Since the parameters depend on several inputs, the nonlinearities seem to have their effects cancelled out. Deadtime is very close to zero (approx. 3 seconds), and was neglected. Averaging the process parameters and calculating a 95% confidence interval for their true means yields

$$g_{22}(s) = \frac{T(s)}{V_b(s)} = \frac{-(7.68 \pm 2.34)}{(1.24 \pm 0.29)s+1} \quad (^{\circ}C/VDC)$$

The predicted model was

$$g_{22}(s) = \frac{T(s)}{V_b(s)} = \frac{-9.12}{1.14\ s + 1} \quad (^{\circ}C/VDC)$$

One can notice that both the process gain and the time constant are close to the theoretically expected values. Figure 4.3 shows a comparison between both $g_{22}(s)$ models.

The following conclusions can be drawn from the results above:

1. There is a significant discrepancy between the theoretical and experimental $g_{11}(s)$ process gains. The limitation in the magnitude of voltage stepchange

used to identify $g_{11}(s)$, together with process nonlinearities, may explain this difference in K_p.

2. The small magnitude of voltage stepchange may also explain the presence of deadtime in the experimental $g_{11}(s)$ and $g_{12}(s)$, whereas no deadtime was noticed under closed-loop conditions.
3. The discrepancies in the $g_{11}(s)$ and $g_{12}(s)$ time constants may have been originated from the simplifying assumptions used to build the theoretical process model (i.e. linearization of the equations that describe the process, neglected dynamics of sensors, transmitters and transducers, etc.).
4. Contrary to what was expected from the theoretical modelling, no significant evidence of nonlinearities has been found in any of the experimental open-loop transfer functions. Nonlinearities do exist but, due to the limitations in the identification procedure explained above, together with the scatter in the data, they did not become apparent.
5. The suitability of the system for lab experiments has been confirmed by the experimental results. Time scale of the process is in the order of a few minutes, and process is also open-loop stable.

TABLE 4.3

$g_{22}(s)$ Experimental Identification.

F(L/min)	T(^{o}C)	INITIAL V_b (VDC)	STEP-CHANGE IN V_b (VDC)	$t_{28.3\%}$ (min)	$t_{63.2\%}$ (min)	K_p (^{o}C/VDC)	τ_p (min)	D (min)
9.3	39	1.64	-1.01	0.80	2.30	-19.50	2.25	0.05
12.2	29	3.00	-1.60	0.55	1.45	-6.88	1.35	0.10
13.0	29	2.60	-1.00	0.60	1.50	-6.20	1.35	0.15
13.8	30	1.00	0.80	0.29	0.81	-7.16	0.78	0.03
15.5	32	2.00	-0.80	0.42	1.22	-7.56	1.20	0.02
17.3	24	3.10	-1.03	0.45	1.00	-4.10	0.83	0.18
17.6	25	1.64	1.02	0.30	1.12	-7.50	1.23	-0.11
19.0	34	1.00	1.00	0.38	1.07	-6.20	1.03	0.04
19.0	33	1.20	0.80	0.48	1.93	-5.00	2.17	0.10
20.6	29	1.40	1.60	0.40	1.00	-6.94	0.90	0.10
23.3	34	0.54	2.56	0.33	0.97	-10.00	0.95	0.02
23.3	28	1.05	2.04	0.18	0.98	-5.90	1.20	-0.22
23.6	28	1.40	2.00	0.48	1.05	-6.85	0.86	0.18

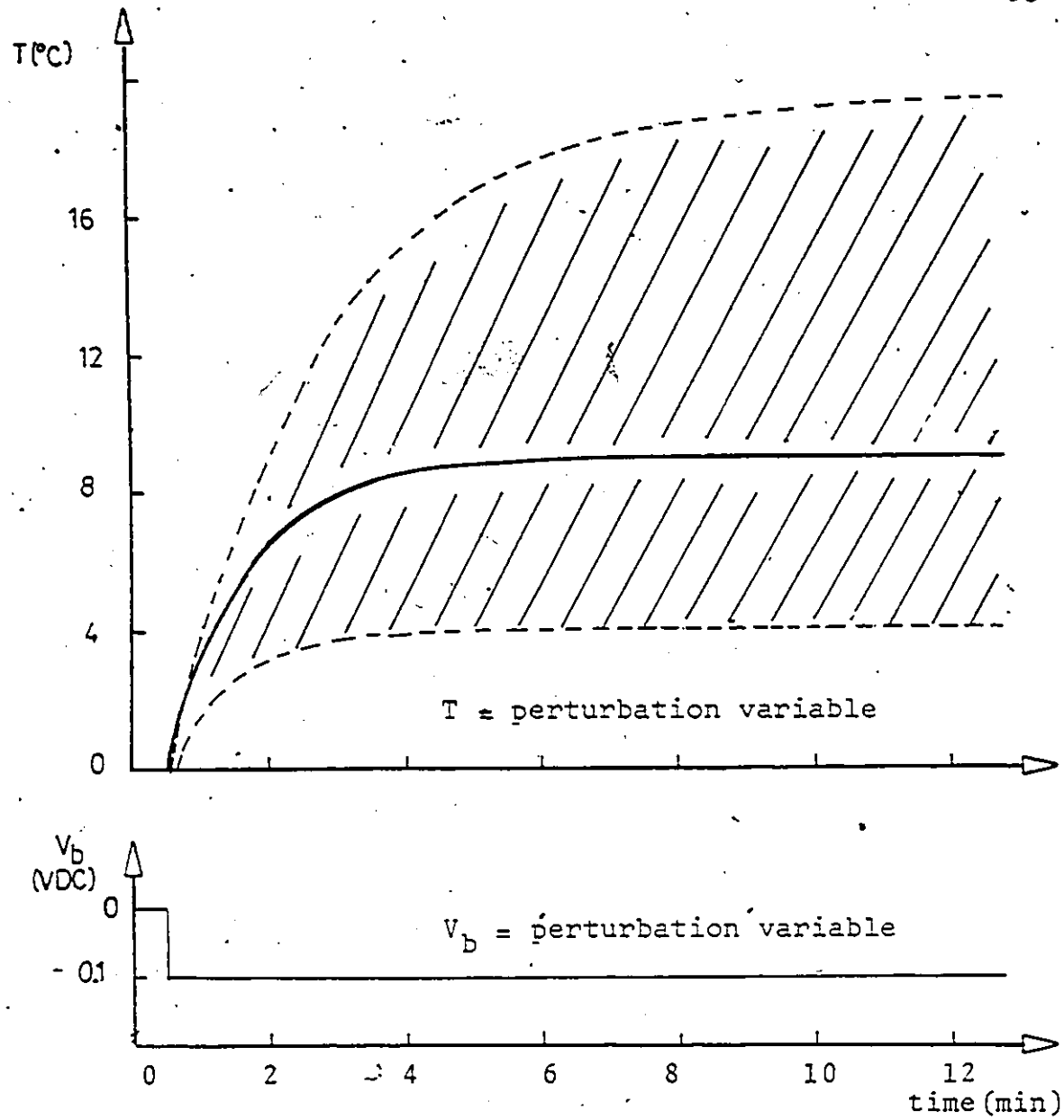


Figure 4.3: Theoretical and Experimental $g_{22}(s)$ reaction curves.

——— theoretical

ZZZ band of experimental results.

4.2 PID LEVEL CONTROL

Control algorithms require tuning i.e. choosing appropriate values of the controller parameters K_c, τ_I, and τ_d. One common way to obtain initial values for these parameters is to use Ziegler-Nichols values[8,20], which are related to the ultimate gain and period of oscillation for the loop as given in Table 4.4 .

TABLE 4.4

Ziegler-Nichols Settings.

	P	PI	PID
K_c	Ku/2	Ku/2.2	Ku/1.7
τ_I (seconds)	-	Pu/1.2	Pu/2
τ_D (seconds)	-	-	Pu/8

To obtain these ultimate values the controller is operated with proportional mode alone at higher and higher controller gains until the controlled variable oscillates continuously at constant amplitude. At this point any further increase in controller gain will cause the system to become unstable. The values of the controller gain and period of oscillation are the desired "ultimate values".

4.2.1 Position Algorithm

For this work the position algorithm program was used to determine the ultimate gain, Ku, and the ultimate period of oscillation, Pu. The values

$$Ku = 30$$

and $Pu = 12$ seconds

were obtained. The details of this procedure are outlined in Appendix L. Further tuning is then usually performed by making additional small adjustments in the parameter values. Level control is almost always used for inventory purposes, in which a small offset is acceptable. Therefore, P-only control is sufficient. For our applications this was the case. PI and PID modes were made available in our control program mainly for demonstration purposes (e.g. to show the individual effects of the controller parameters). To demonstrate the closed-loop performance of the level loop, the responses to a stepchange in level setpoint for both P and PI position algorithms are shown in Figure 4.4 (a).

For the P-only mode, the Ziegler-Nichols setting ($K_c = Ku/2 = 15$) was satisfactory. For PI mode, however, the Ziegler-Nichols settings ($K_c = 14$ and $\tau_I = 10$ seconds) resulted in appreciable overshoot. This overshoot was reduced by reducing the integral action and increasing the controller gain. The chosen controller settings were

$$K_c = 20$$

and $\tau_I = 40$ seconds.

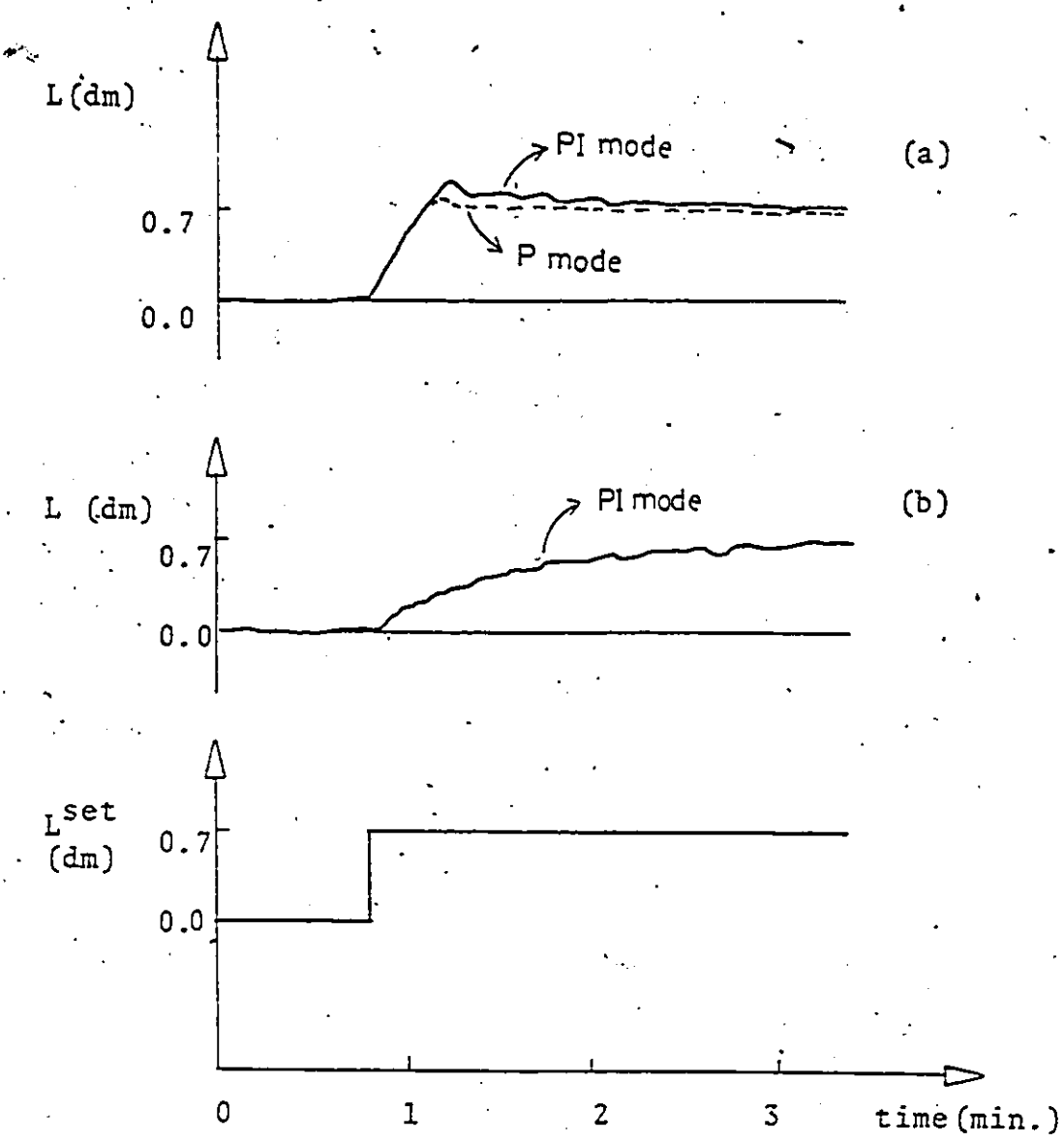


Figure 4.4 : Level response using P and PI position algorithms (a) and a PI velocity algorithm (b).
For P mode : K_c = 15
For PI mode: K_c = 20 and τ_I = 40 seconds.
F_b = 14.10 L/minute.

It is important to notice the lack of deadtime under closed-loop conditions. This fact confirms that the deadtime found in the open loop transfer function $g_{11}(s)$ was overestimated and could be neglected. The new setpoint was reached for the first time within 25 seconds. Overshoot is still noticeable, the level lining up at the new setpoint in approximately 2 minutes.

4.2.2 Velocity Algorithm

Figure 4.4 (b) shows the level response to a setpoint stepchange using the PI velocity algorithm. Settings were the same as the ones used for the position PI (K_c = 20 and τ_I = 40 seconds), and the setpoint stepchange was also the same. The velocity algorithm, as opposed to the position algorithm, reacts one sampling time later (a sampling interval of 2 seconds was used). Response is slower than the previous one, but there is no overshoot. Both the position and the velocity algorithms take approximately the same amount of time (2 minutes) to line up at the new setpoint. Level controller performance under load disturbances will be shown in section 4.5 .

4.3 LEVEL AND TEMPERATURE CONTROL

This program controls simultaneously the level and water temperature in the tank. The level was usually controlled with the position P-only control, for the reasons already mentioned in the beginning of this section. For temperature control, offsets are normally not tolerated, thus integral action should be present in the algorithm so that the offset is driven to zero.

In order to tune the temperature loop the P-only level control loop was kept tightly tuned (K_c = 30). The ultimate gain for the temperature loop was found to be 1.75, and the ultimate period of oscillation, 45 seconds (see Appendix L).

Figure 4.5 shows both the level and temperature responses for 3 successive level setpoint stepchanges. Ziegler-Nichols settings were used for both the level and temperature loops. The P-only level controller gain was made equal to 15, whereas the PI temperature controller had the settings K_c = 0.8 and τ_I = 38 seconds.
Notice that there is no fluctuation in the temperature as expected from the open loop analyses confirming that $g_{21}(s) = T(s)/V_a(s) = 0$.

Figure 4.6 shows the level and temperature responses to 2 successive temperature setpoint stepchanges. Notice that the level is only slightly upset by the temperature stepchanges, thus coupling in that direction is almost neg-

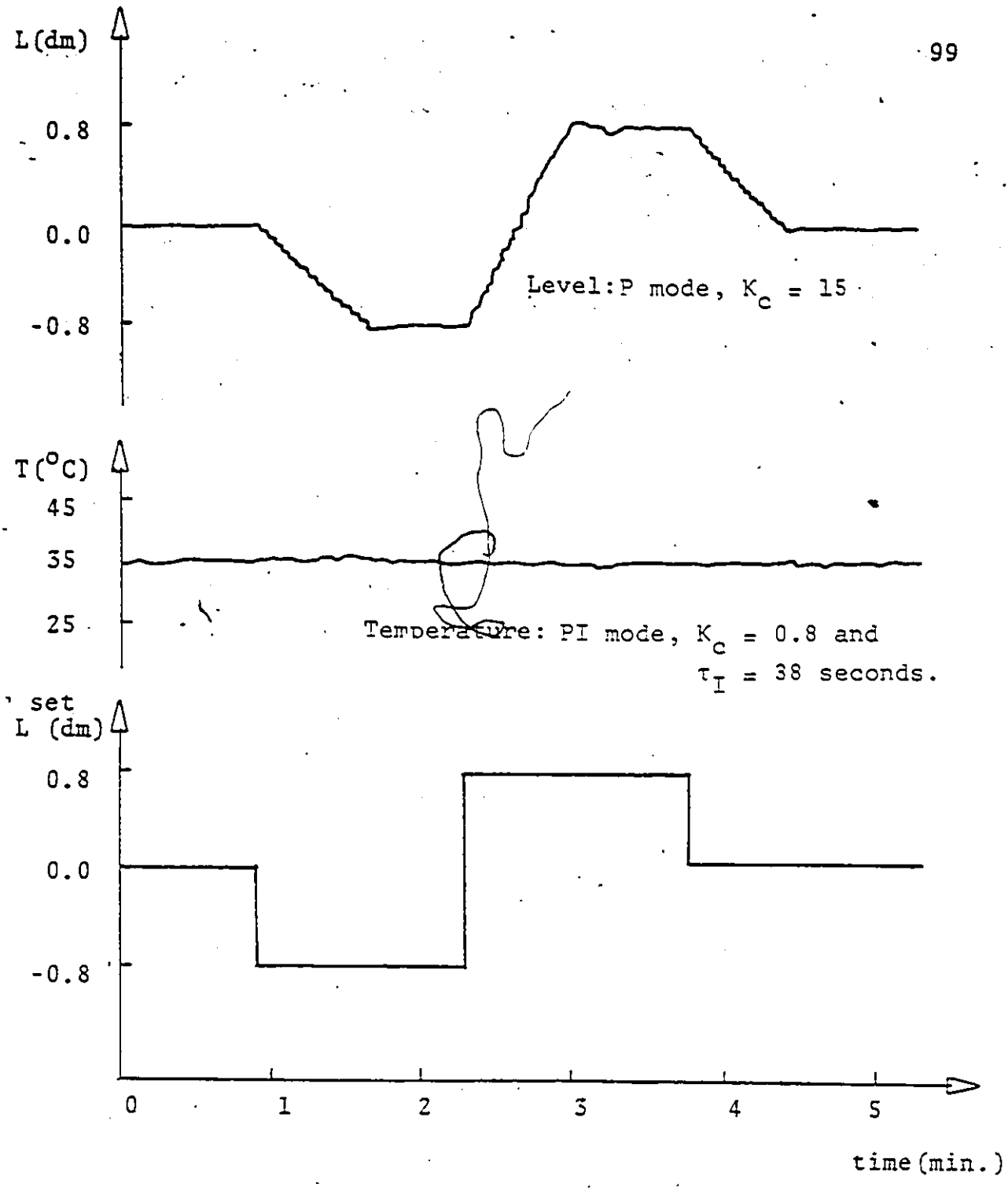


Figure 4.5: Multivariable response to level setpoint stepchanges.
F_h = 7.6 L/minute and T_h = 60 °C

ligible. In fact, coupling can only be seen when the level is loosely tuned, as will be shown in Section 4.4 . It can also be noticed from Figure 4.6 that these little upsets in level occur almost instantly after the temperature setpoint stepchanges were introduced, even before the temperature begins changing. This confirms that there is no time delay in the open-loop transfer function $g_{12}(s) = L(s)/V_b(s)$. System performance under load disturbances will be shown in section 4.5 .

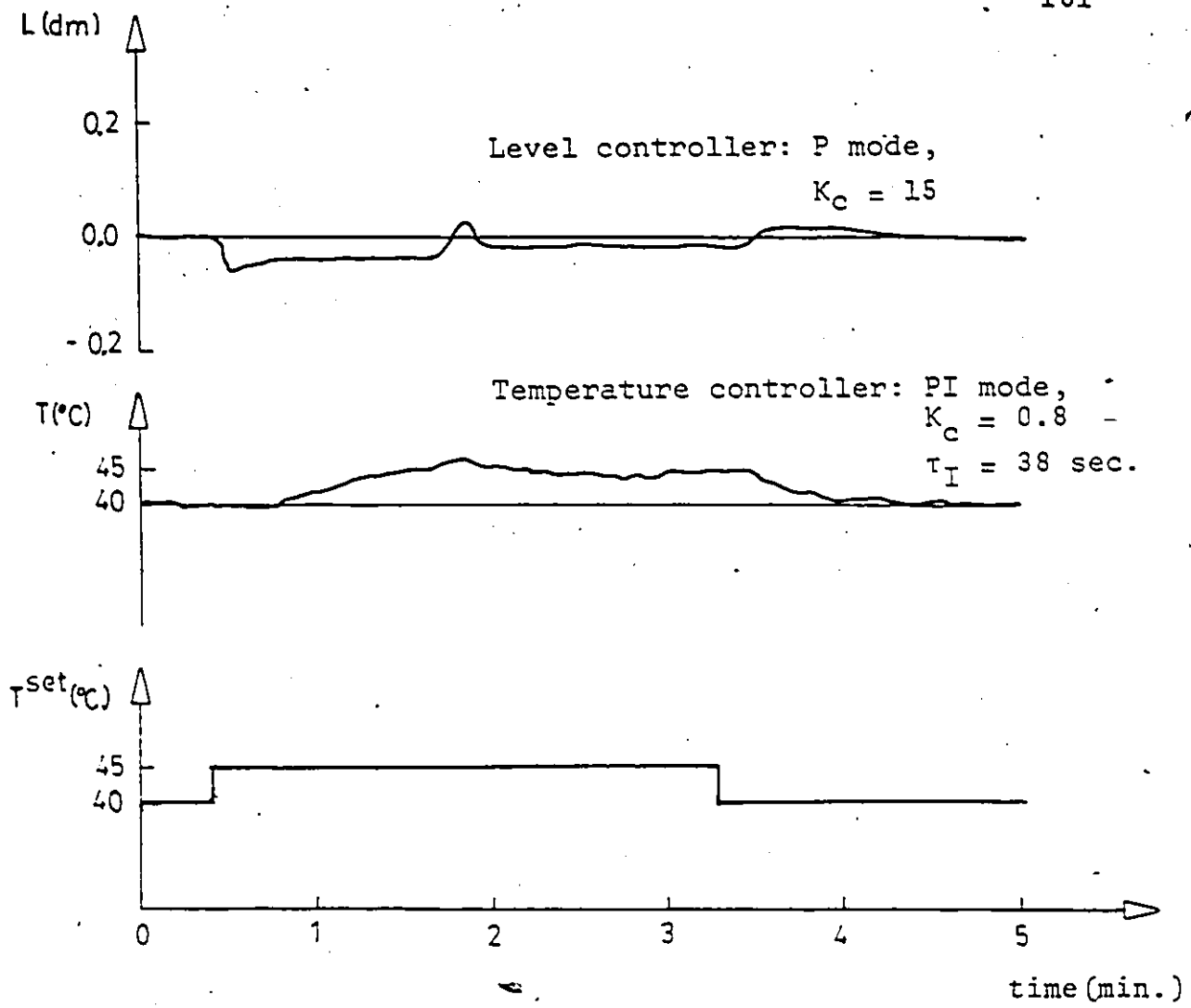


Figure 4.6: Multivariable response to temperature setpoint step changes.
F_h = 7.6 L/minute and T_h = 60 °C

4.4 LEVEL / TEMPERATURE CONTROL USING DYNAMIC DECOUPLING

As was already mentioned in Section 4.3, coupling was not noticed in our system unless the level, which is affected by the coupling, was loosely tuned (K_c <= 1.0). Figure 4.7 shows the level and temperature responses to a temperature setpoint stepchange. The controller settings were as follows:

Level : position P-only algorithm loosely tuned,

K_c = 1.0 and

Temperature : position PI algorithm,

(Ziegler-Nichols settings)

K_c = 0.8,

τ_I = 38 seconds.

The decoupling algorithm developed in section 3.4 is available in the level/temperature control program. The program asks the user if the decoupler is to be implemented or not. If the answer is no, the algorithm is simply skipped during program execution. Figure 4.7 (a) indicates the level response to a temperature setpoint stepchange without decoupling, whereas Figure 4.7 (b) shows the decoupled response. There is a substantial decrease in the level upset when the decoupler is implemented. If we were using a perfect dynamic decoupler, and if the process transfer functions were perfectly known, no upset whatsoever would be observed. Decoupler performance under other controller settings and load disturbances will be shown in the next section.

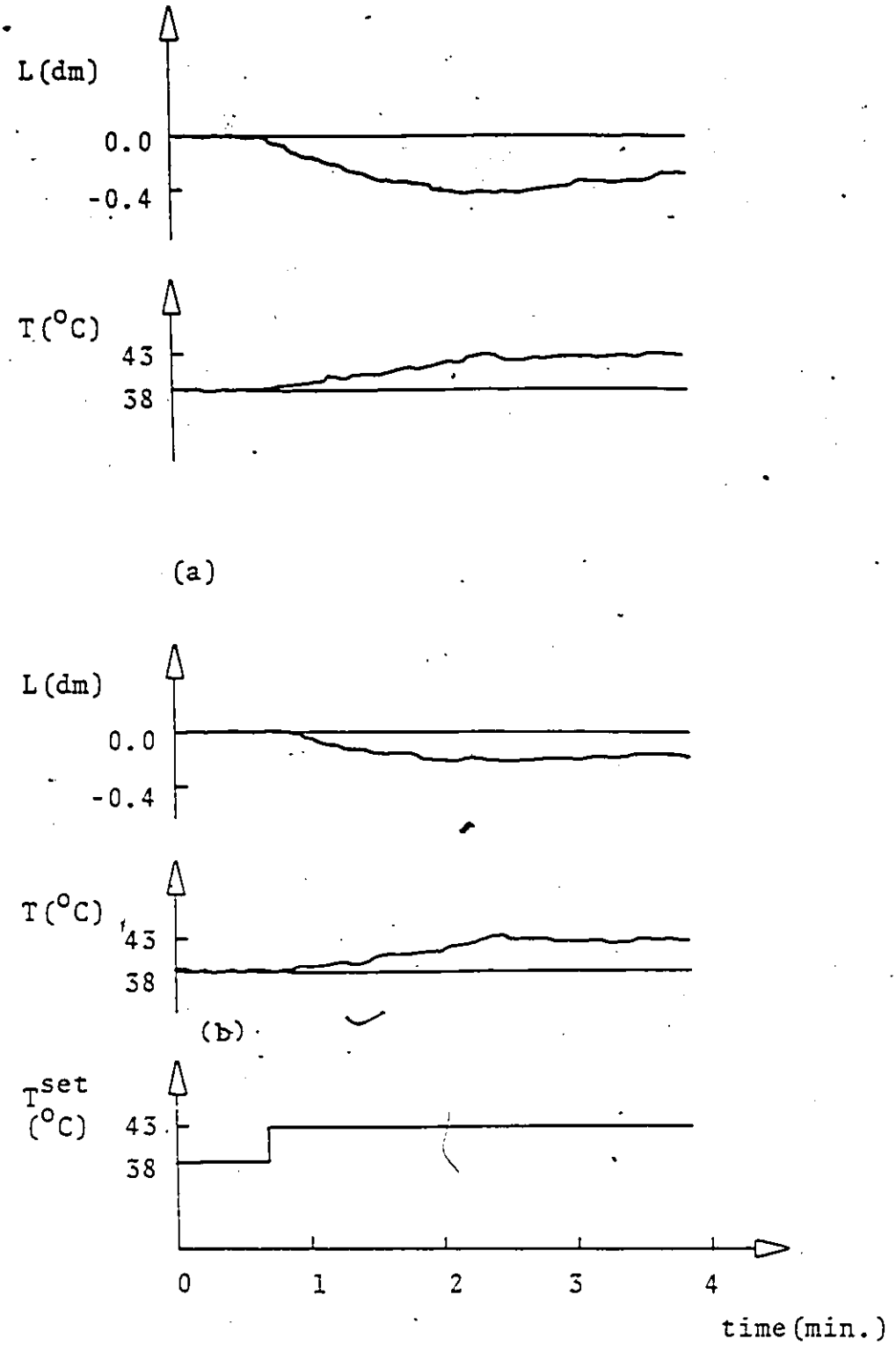


Figure 4.7: Multivariable control without decoupling (a) and with dynamic decoupling (b).

4.5 COMPARISON OF STRATEGIES FOR DECOUPLING

In this section the performance of the dynamic decoupler presented in the previous section will be compared with a steady-state decoupler and with the conventional one-loop-tight one-loop-loose strategy.

The steady-state decoupler developed in Section 3.4 was

$$\underline{G}(s)_{I_{s.s.}} \begin{bmatrix} 1 & 0.23 \\ 0 & 1 \end{bmatrix}$$

To implement the steady-state decoupler in our level/temperature control program, the dynamic decoupler equation in line 1160 was reduced to

$$D1 = 0.23\ N1$$

where D1 = decoupler output(VDC) and

N1 = output from temperature controller(VDC).

Five different controller configurations were tested against both load and setpoint stepchanges. The controller configurations are summarized in Table 4.5 .

Configuration A was chosen to make coupling very visible, i.e., to enhance coupling. The smaller the level controller gain, the greater coupling is expected to become. Configurations B and C were chosen to show the improvement in the level response obtained respectively with steady-state and

TABLE 4.5

LEVEL AND TEMPERATURE CONTROLLER CONFIGURATIONS

CONFIGURATION	LEVEL ALGORITHM	TEMPERATURE ALGORITHM	DECOUPLER
A	Position P only K_c = 1.0 (loosely tuned)	Position PI Ziegler-Nichols settings: K_c = 0.8 τ_I = 38 sec.	NO
B	Same as above	same as above	Steady-state decoupler
C	Same as above	Same as above	Dynamic Decoupler
D	Position P only Ziegler-Nichols setting: K_c = 15	Same as above	NO
E	Position P only K_c = 20 (tightly tuned)	Same as above	NO

dynamic decoupling, as compared with the poor situation in configuration A.

For higher level controller gains, coupling is expected to decrease, becoming less and less visible. Configurations D and E were intended to show this trend. At this point the decouplers were no longer implemented, because their effects would no longer be noticeable.

Tighter temperature control configurations were not considered, because the temperature response had a tendency to become oscillatory, resulting in poorer control. The controller configurations above have been tested against the step disturbances summarized in Table 4.6 .

TABLE 4.6

Setpoint and Load Disturbances.

STEP DISTURBANCE	T^{set} (^{o}C)	F_h (L/min)	T_h (^{o}C)
#1	35 to 30	-	-
#2	30 to 40	-	-
#3	-	7.6 to 9.6	-
#4	-	-	60 to 45

Disturbances #1 and #2 were intended to test the control system with decoupler under temperature setpoint stepchanges, because they affect the level response. Level setpoint stepchanges were not included because they do not affect the temperature loop.

Disturbances #3 and #4 were intended to test not only the decoupling scheme, but also the individual performance of the level and temperature control loops under load disturbances. A visual comparison of the various level responses is shown in Figures 4.8 through 4.11 . A quantitative comparison of the various controller performances was obtained by computing the Integral of the Squared Error ISE defined as

$$ISE = \int_0^t e(t)^2 \, dt$$

and calculated in our computer program by the expression

```
770 LEVISE = LEVISE + ST * (LSET - L)^2
```

for the various level responses. The I.S.E was computed during 3 minutes, time enough for the slowest transients to die out. Results are shown in Table 4.7 . Temperature I.S.E. data were also presented in Table 4.7 . The relative magnitudes of the temperature I.S.E. data, due to scaling in the variables, are greater than those for the level. Temperature I.S.E. data were, however, approximately constant within each one of the four step disturbances. For this reason, only one temperature response has been shown in each figure, even though there was one for each level response.

As can be seen in Table 4.7 , the smallest I.S.E. was obtained, for all four disturbances, with the level tightly tuned (configuration E). The second best was configuration

TABLE 4.7

Level and Temperature I.S.E. for various step disturbances.

STEPCHANGE		CONTROLLER CONFIGURATION A	B	C	D	E
#1	Level	11.39	6.28	6.00	0.08	0.05
	Temp.	498.44	607.32	440.94	470.35	468.43
#2	Level	9.29	2.11	2.59	0.08	0.05
	Temp.	2423.41	2914.77	3267.34	3357.18	3292.14
#3	Level	6.66	6.24	6.08	0.07	0.03
	Temp.	664.71	705.97	681.03	609.92	553.61
#4	Level	6.33	1.37	1.39	0.04	0.02
	Temp.	167.62	230.56	227.72	215.66	216.42

D, in which the level was tuned according to the Ziegler-Nichols settings. For the poorer situations in which the level was loosely tuned (configurations A, B and C), dynamic and steady-state decoupling had approximately the same performance, yielding the smallest I.S.E.. Appendix M shows that dynamic decoupling (configuration C) significantly improved the level response to a temperature setpoint stepchange, in comparison with the no decoupling case (configuration A).

It is also interesting to notice that the level response to load disturbances in F_h (disturbance #3) was not greatly improved with decoupling. Since the decoupling algorithm is contained inside the feedback loop, improvement was expected.

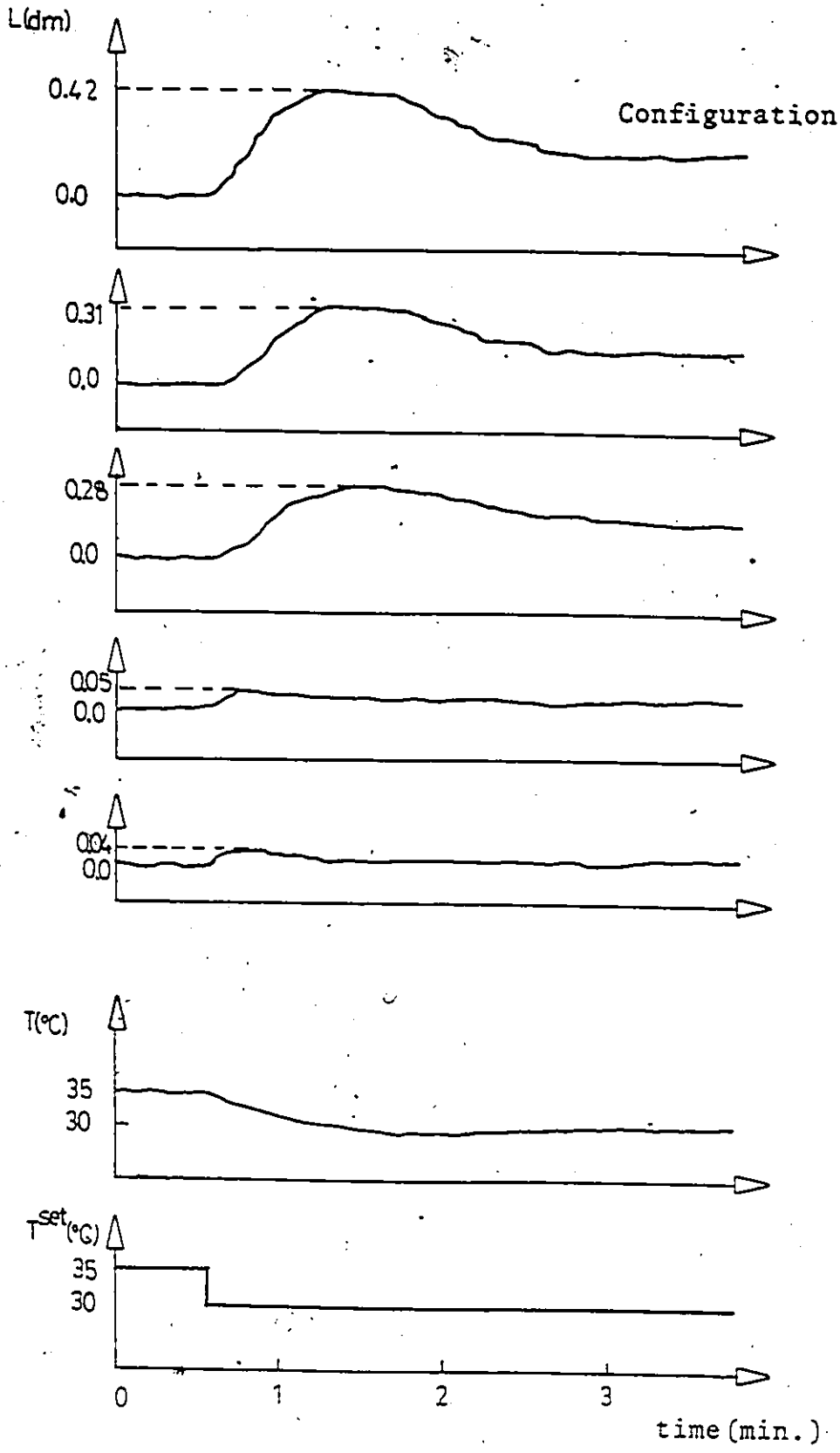


Figure 4.8: Level Response to Step Change #1 .

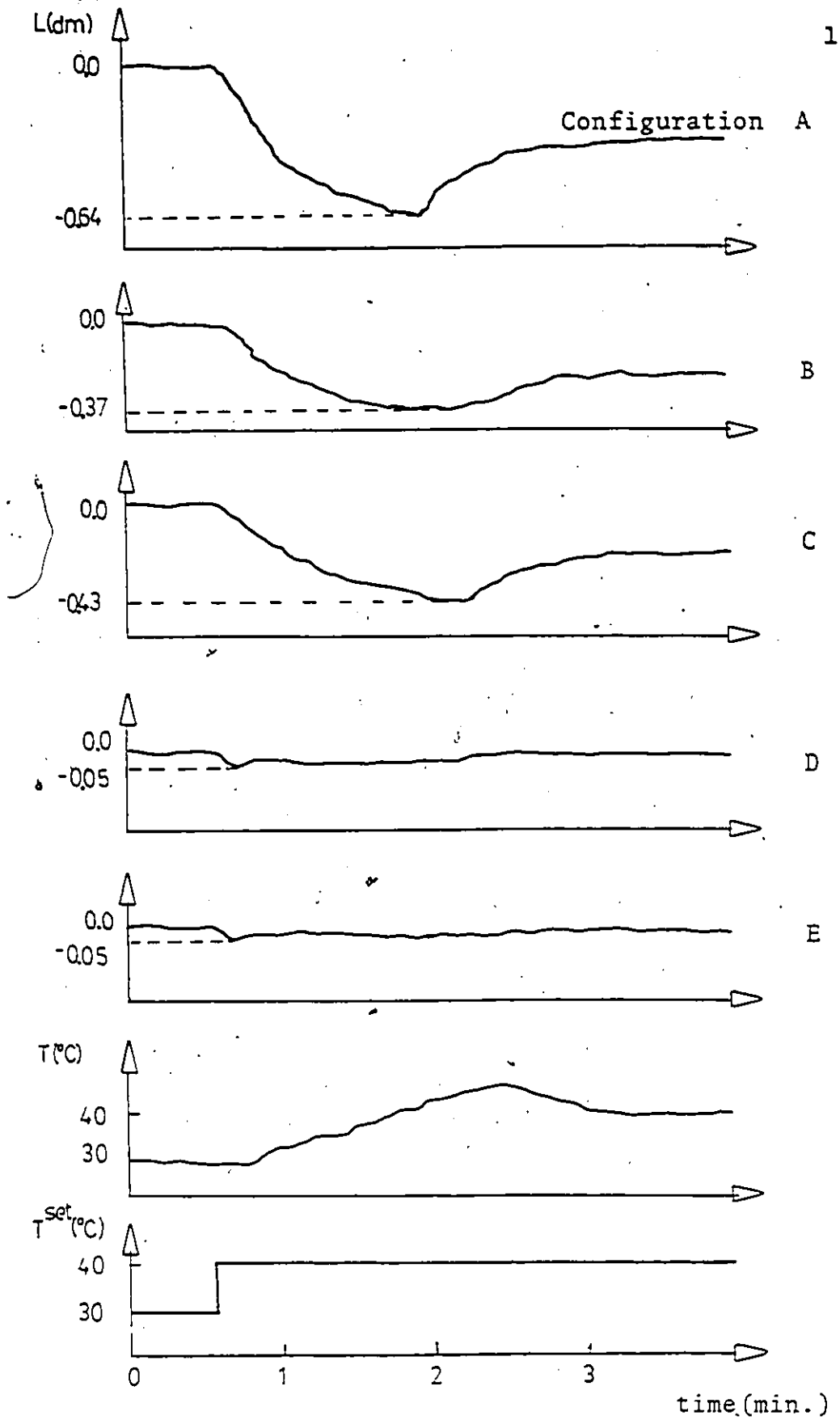


Figure 4.9: Level Response to Step Change #2 .

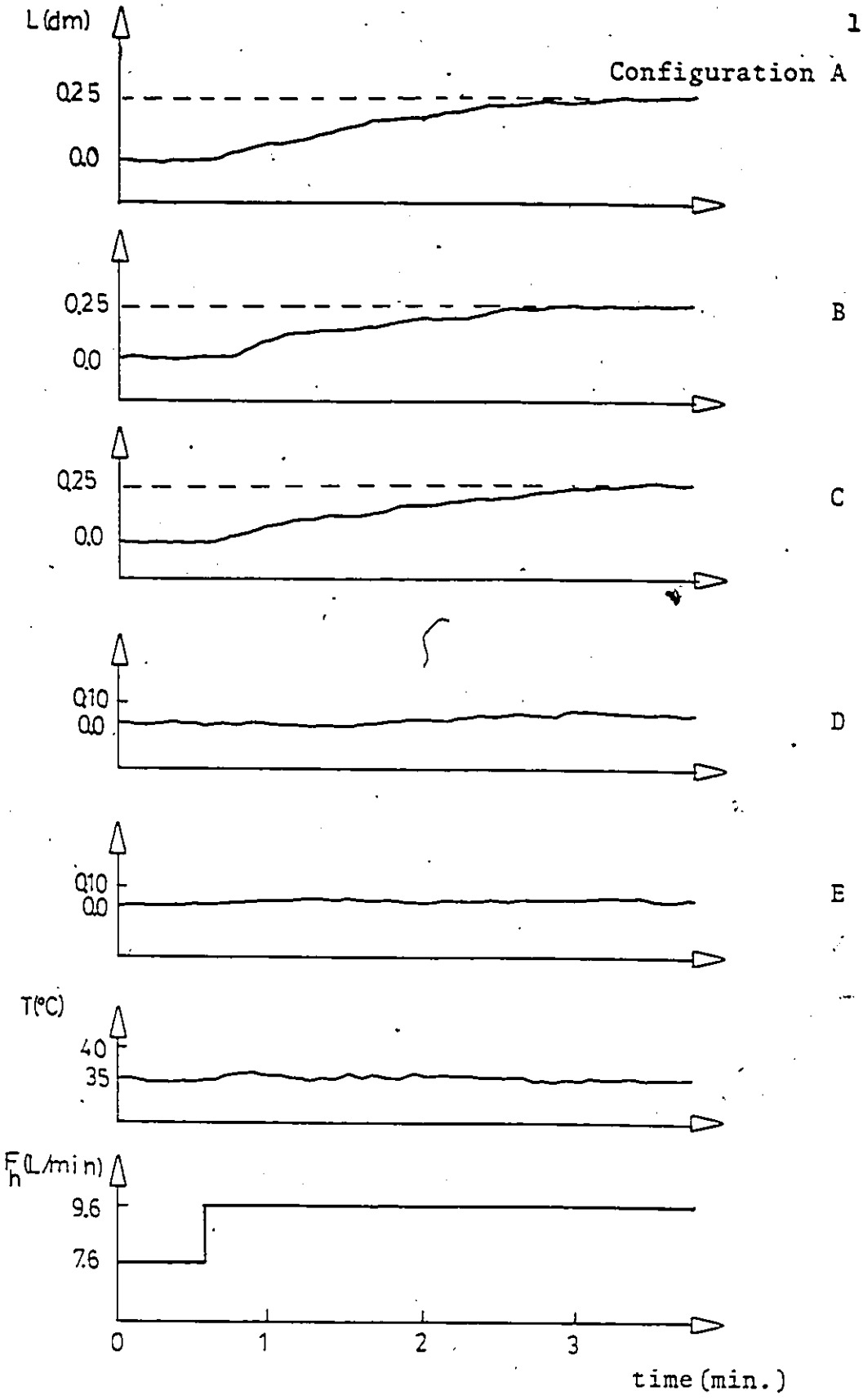


Figure 4.10: Level Response to Step Change #3 .

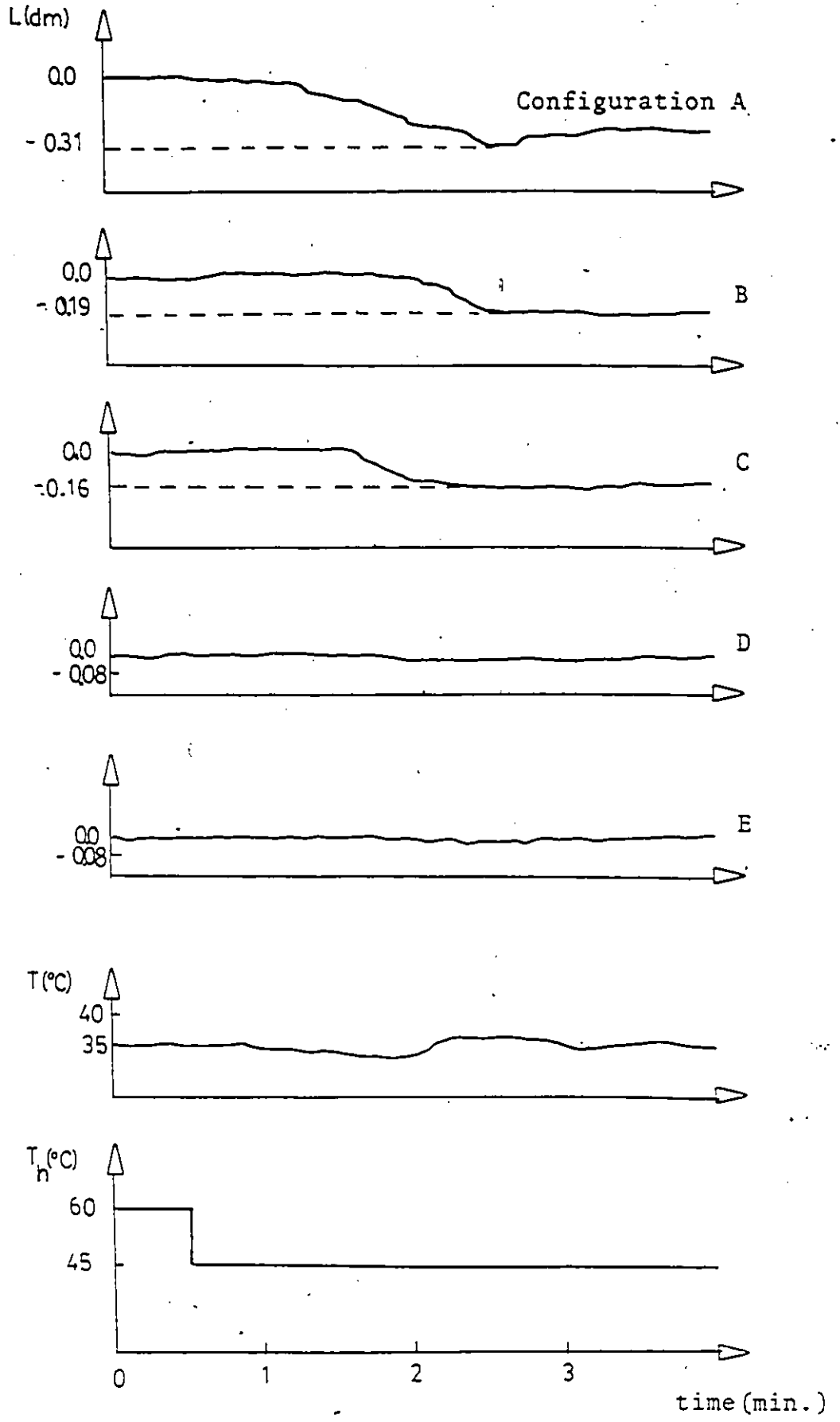


Figure 4.11: Level Response to Step Change #4 .

Chapter V

CONCLUSIONS

Digital computer control was successfully designed and implemented in two loops, namely level and temperature. Both loops presented good performance under load and set-point disturbances. With respect to the implementation of the computer control system, the following conclusions were drawn:

1. The control system is suitable for lab demonstration purposes, because the process time constants are in the order of a few minutes, therefore allowing several experiments to be carried out during a lab period. The process is also open-loop stable, and can be identified using conventional process control theory.
2. Labsoft, the control-oriented language used in this project, greatly simplified the design and implementation of computer control, particularly input and output operations.
3. The position and velocity PID algorithms have different performances, despite being originated from the same equation (i.e. the continuous PID algorithm).

Four strategies for dealing with coupling have been evaluated, namely no decoupling, steady-state decoupling,

dynamic decoupling and the one-loop-tight one-loop-loose strategy. With respect to the strategies above, the following conclusions were drawn:

1. Dynamic decoupling, despite its greater complexity, did not work much better than steady-state decoupling. This arises from the fact that the magnitudes of the two time constants in the lead-lag element are approximately the same. In this case the steady-state response dominates over the transient response.
2. Coupling was best handled by keeping the level loop tightly tuned in comparison with the temperature loop.

Chapter VI

RECOMMENDATIONS

1) If computer memory space becomes a problem for future control schemes, the highest memory address available for the user (HIMEM) should be relocated 8K higher, thus making available for BASIC programs the space originally reserved for high resolution graphics (HGR) page 1. If space continues being a problem, HIMEM should be set 12K even higher, using the memory space originally reserved for high resolution graphics page 2.

2) Valve A could be relocated so that it would control the level in the tank by adjusting the hot water inlet flowrate instead of the outlet flowrate. The outlet stream would exit through a pipe with no restrictions in it. This setup would improve valve A's linear installed characteristics, since a higher and constant pressure drop over the valve would be available. This setup would also create bi-directional coupling (i.e. interaction) between the control loops, because a change in the inlet hot water flowrate would upset both the level and the temperature in the tank.

3) Time delay in the temperature loop could be increased by installing a longer outlet pipe and measuring the temperature farther away from the tank. With the present setup, however, too long an outlet pipe may worsen valve A's installed characteristics, due to the smaller flowrate and smaller pressure drop available over the valve.

LIST OF REFERENCES

1. Ray, W. H., Advanced Process Control, McGraw Hill, New York (1982).

2. Ginn, P. L., An Introduction to Process Control and Digital Minicomputers, Gulf Publishing Company, Houston (1982).

3. Skrokov, M. R., Mini- and Microcomputer Control in Industrial Processes, Van Nostrand, New York (1980).

4. Abd-El-Bary, M. F. and Chari, S., Chemical Engineering Education, Summer 1982, 118 (1982).

5. Deshpande, P. B. et al., Chemical Engineering Education, Winter 1980, 26 (1980).

6. Abd-El-Bary, M. F., Chemical Engineering Education, Winter 1983, 28 (1983).

7. Mellichamp, D.A., editor, CACHE Monograph Series in Real-time Computing, CACHE, Cambridge, MA (1977).

8. Luyben, W. L., Process Modeling, Simulation and Control for Chemical Engineers, McGraw Hill, New York (1973).

9. Deshpande, P. B. and Ash, R. H., Elements of Computer Process Control, Instrument Society of America, Research Triangle Park, N.C. (1981).

10. Cyborg Corporation, ISAAC 91A System Reference Manual.

11. Cyborg Corporation, ISAAC Labsoft Reference Manual.

12. Considine, D. M., Process Instruments and Controls Handbook, McGraw Hill, New York (1974).

13. Box, G. E. P. and Jenkins, G. M., Time Series Analysis: forecasting and control, Holden-Day, Oakland, CA (1976).

14. Bristol, E. H., IEEE Trans. Auto. Control, AC-11, 133 (1966).

15. Luyben, W. L., AIChEJ., 2, 198 (1970).

16. Rizvi, S. F. H., Multivariable Control System Design in the Presence of Interaction, Warren Spring Laboratory, Hertfordshire, U.K. (1978).

17. Landau, Y. D., Adaptive Control, Marcel Dekker, Inc., New York (1979).

18. Smith, C. L., Digital Computer Process Control, Intex Educational Publishers, Scranton, PA, (1972).

19. Harris, C. J. and Billings, S.A., Self-tuning and Adaptive Control, Peter Peregrinus Ltd., Stevenage, U.K. (1981).

20. Murril, P. W., Fundamentals of Process Control Theory, Instrument Society of America, Research Triangle Park, N.C. (1981).

21. Rijnsdorp, J.E. and Seborg, D. E., AIChE Symposium Series, 72, 159, 112 (1976).

22. Box, G. E. P. and McGregor, J. F., Technometrics, 16, 3, 391 (1974).

23. Stire, T. G., Process Control Computer Systems, Guide for Managers, Ann Arbor Science Publishers, Ann Arbor, MI (1983).

24. Apple Computer Inc., Basic Programming Reference Manual.

25. Olsson, G., AIChE Symposium Series, 72, 159, 52 (1976).

26. Edgar, T.F., editor, Design of Sampled Data Computer Control Systems, AIChEMI, Series A, Volume 3, New York (1982).

27. Honeywell Co., Service Manual 59095M.

Appendix A

SAMPLED-DATA DISCRETE APPROXIMATION TO A PID ALGORITHM

The velocity form for the PID controller (Equation 2.4, Chapter 2), can also be obtained using the sampled-data theory approach.

Recall the ideal PID controller equation:

$$G_c(s) = \frac{M(s)}{E(s)} = K_c \left(1 + \frac{1}{\tau_I s} + \tau_d s \right)$$

The finite difference equation that approximates the ideal PID controller is obtained by writting G(s) in terms of s^{-1} (7).

$$\frac{M(s)}{E(s)} = K_c \left(1 + \frac{s^{-1}}{\tau_I} + \frac{\tau_d}{s^{-1}} \right)$$

Using the rectangular approximation for s^{-1} namely,

$$s^{-1} = \frac{S_t}{1 - z^{-1}} \quad \text{gives}$$

$$\frac{M(s)}{E(s)} = K_c \left(1 + \frac{\frac{S_t}{1-z^{-1}}}{\tau_I} + \frac{\tau_d}{\frac{S_t}{1-z^{-1}}} \right)$$

$$\frac{M(s)}{E(s)} = K_c \left(1 + \frac{S_t}{\tau_I(1-z^{-1})} + \frac{\tau_d(1-z^{-1})}{S_t} \right)$$

$$M(s)\ (1-z^{-1}) = K_c \left(1 + \frac{S_t}{\tau_I(1-z^{-1})} + \frac{\tau_d(1-z^{-1})}{S_t}\right) E(s)\ (1-z^{-1})$$

In the time domain this becomes:

$$m_t = m_{t-1} + K_c\ (e_t - e_{t-1}) + \frac{K_c\ S_t}{\tau_I} e_t + \frac{K_c \tau_d}{S_t} (e_t - 2e_{t-1} - e_{t-2})$$

which is identical to Equation 2.4 .

Appendix B

VALVE A EQUATION

The flowrate through control valve A is given by the equation (Luyben[8], p. 313) :

$$F = C_V \, f(x) \sqrt{\Delta P / sp.gr.} \qquad (B.1)$$

where F = flowrate in gpm,

C_V = valve size coefficient,

$f(x)$ = valve flow characteristics curve,

$sp.gr.$ = specific gravity of liquid (=1 for water),

ΔP = pressure drop over valve in psi and

$f(x)$ = x (linear trim) .

The pressure drop across the valve, ΔP, is actually the hydraulic head available, L, measured from the valve seat to the liquid surface in the tank. Also, for a linear trim, the stem position x is proportional to the voltage applied V_a (i.e. $x = k \, V_a$) . Thus, equation (B.1) reduces to:

$$F = C_V \, K \, V_a \sqrt{L} \qquad (B.2)$$

Grouping the constants C_V and K together yields:

$$F = K_a \, V_a \sqrt{L} \qquad (B.3)$$

where F = flowrate (L/min),

K_a = proportionality constant ($dm^{3/2}$ /V min),

V_a = signal from controller (VDC) and

L = hydraulic head (dm).

Figure B.1 shows the outlet flowrate vs. V_a for the case in which the level was held at its steady-state value (L = 9.48 dm) . This S-shaped curve can be approximated by a straight line passing through the steady-state voltage V_a = 2.63 V and given by the equation:

$$F = -7.77 + 2.80\ V_a \sqrt{L} \qquad (B.4)$$

It must be pointed out that this approximation is valid only if both the level L and the flowrate F are not moved too far away from their design values. Otherwise, the error may become quite large. Equation (B.4) was used in the theoretical analysis of the process, presented in section 3.3 .

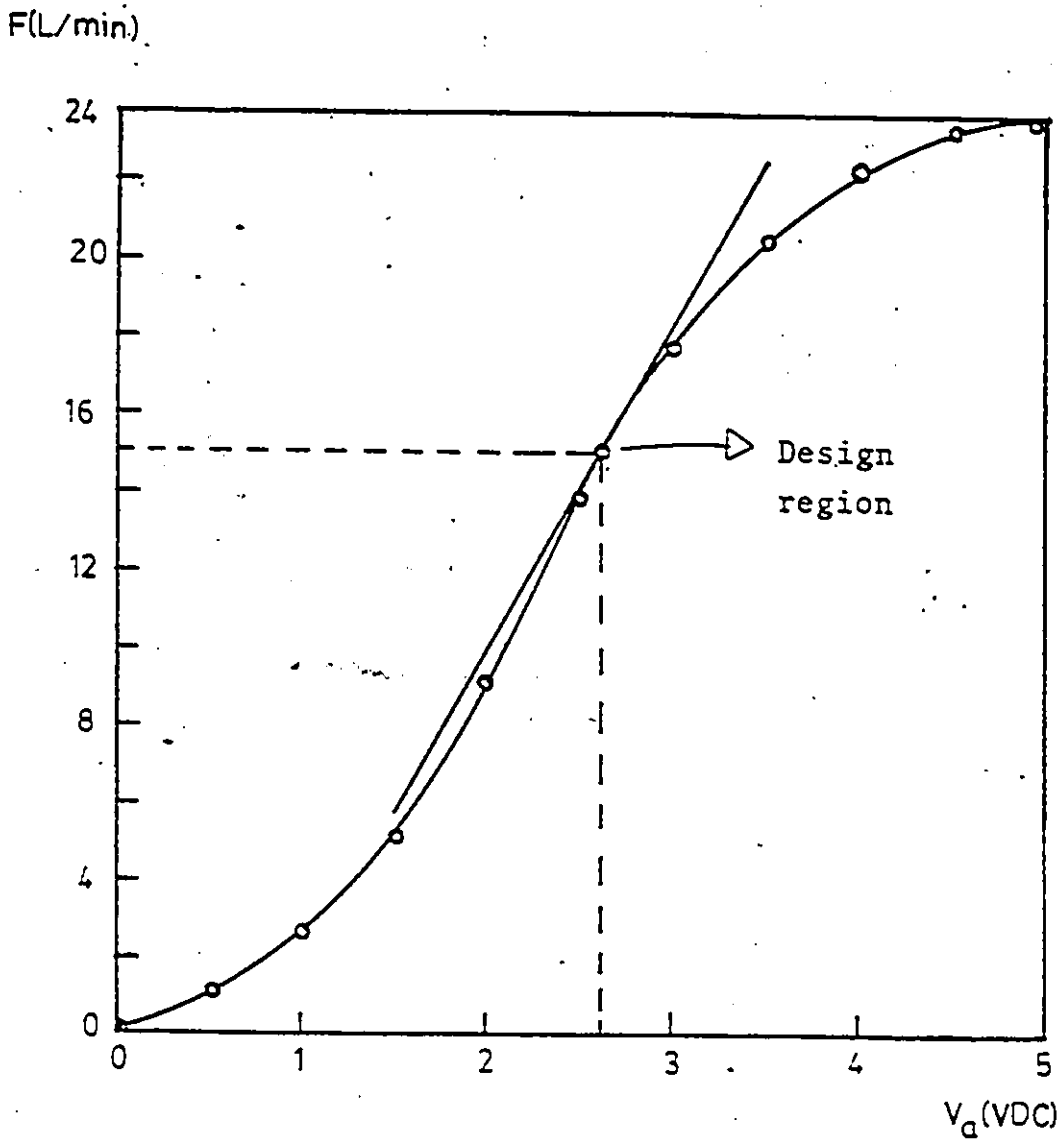


Figure B.1: Valve A - Flowrate F vs. V_a.

L = 9.48 dm

Appendix C

VALVE B EQUATION

As opposed to valve A, the pressure drop across valve B is constant, because the gauge pressure in the line is constant and equal to 30 psi. There is also a flow constraint, which limits the valve's opening to 2/3 of its maximum opening. This is necessary because, due to the higher pressure drop available, the cold and hot water inlet streams may become greater than the maximum attainable outlet flowrate.

Figure C.1 shows the cold water inlet flowrate F_c vs. the voltage signal V_b .Since our operating conditions were usually limited to the range 1 to 3 volts, linear trim was assumed (i.e. $f(x) = x$), where x, the stem opening, is proportional to the voltage V_b applied ($x = k_1 V_b$). Recall eqn. (B.1):

$$F = C_v f(x) \sqrt{\Delta P/\text{sp. gr.}} \qquad (B.1)$$

Since $f(x) = x = K_1 V_b$ and making $\sqrt{\Delta P/\text{sp.gr.}} = K_2$, eqn. (B.1) reduces to:

$$F_c = C_v K_1 V_b K_2 \qquad (C.1)$$

Grouping the constants together,

$$F = K_b V_b \qquad (C.2)$$

The straight line approximation passing through the steady-state voltage V_b = 2 V has equation

$$F_c = -3.33 + 5.33\ V_b \qquad (C.3)$$

Equation (C.3) was used in the theoretical analysis of the process, presented in chapter 3.3 .

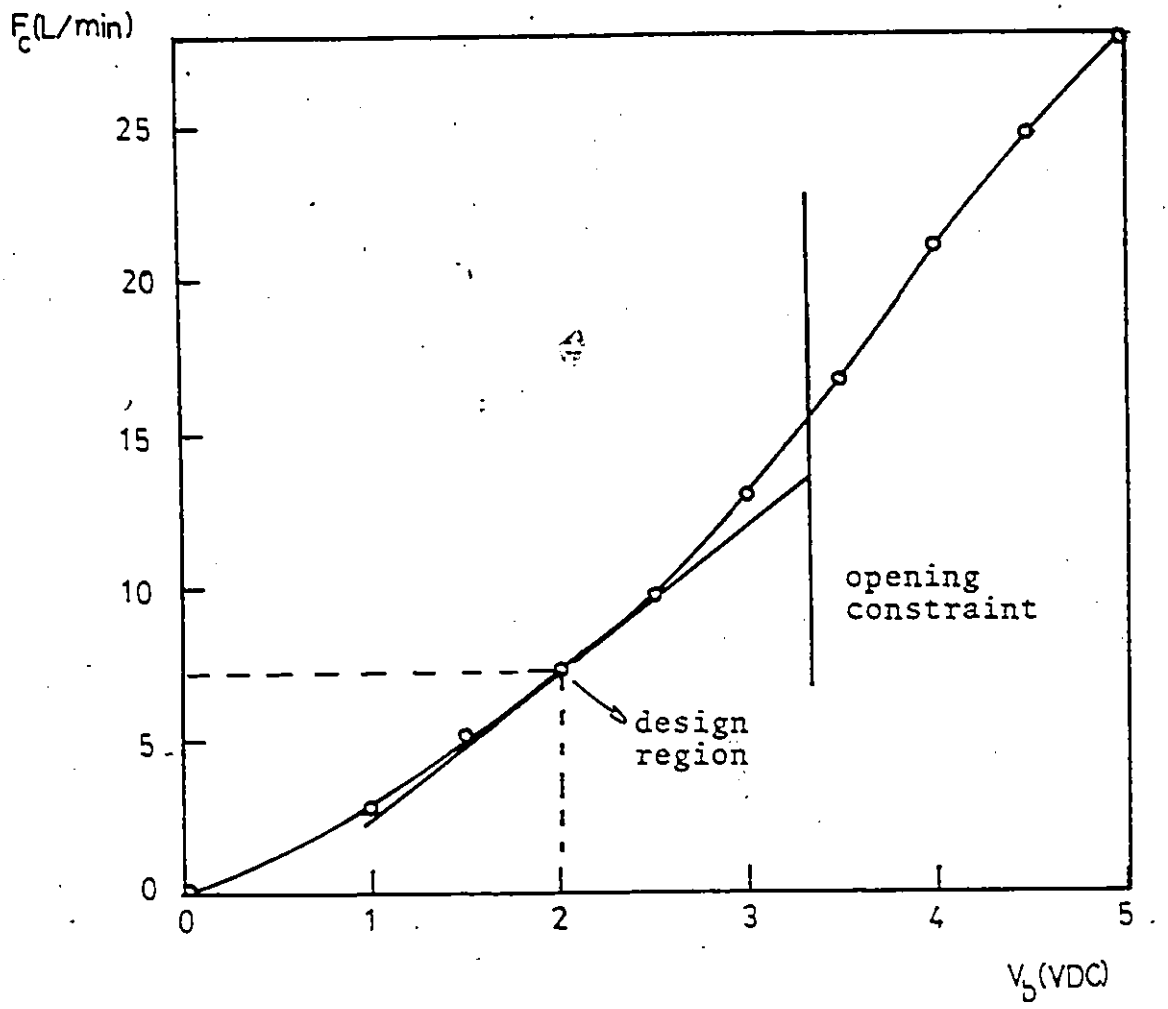


Figure C.1: Valve B - Flowrate F_c vs. V_b.

Appendix D

ROTAMETER CALIBRATION CURVE.

The hot water supply has its flowrate adjusted by a manual valve. The flowrate is read on a Fisher Porter rotameter model FP-1-35-G-10, whose calibration curve is shown in Figure D.1 . The abscissa (reading in %) corresponds to percentage of maximum flowrate, which for our system is limited to the range 10 to 30% (5.2 to 14 L/minute).

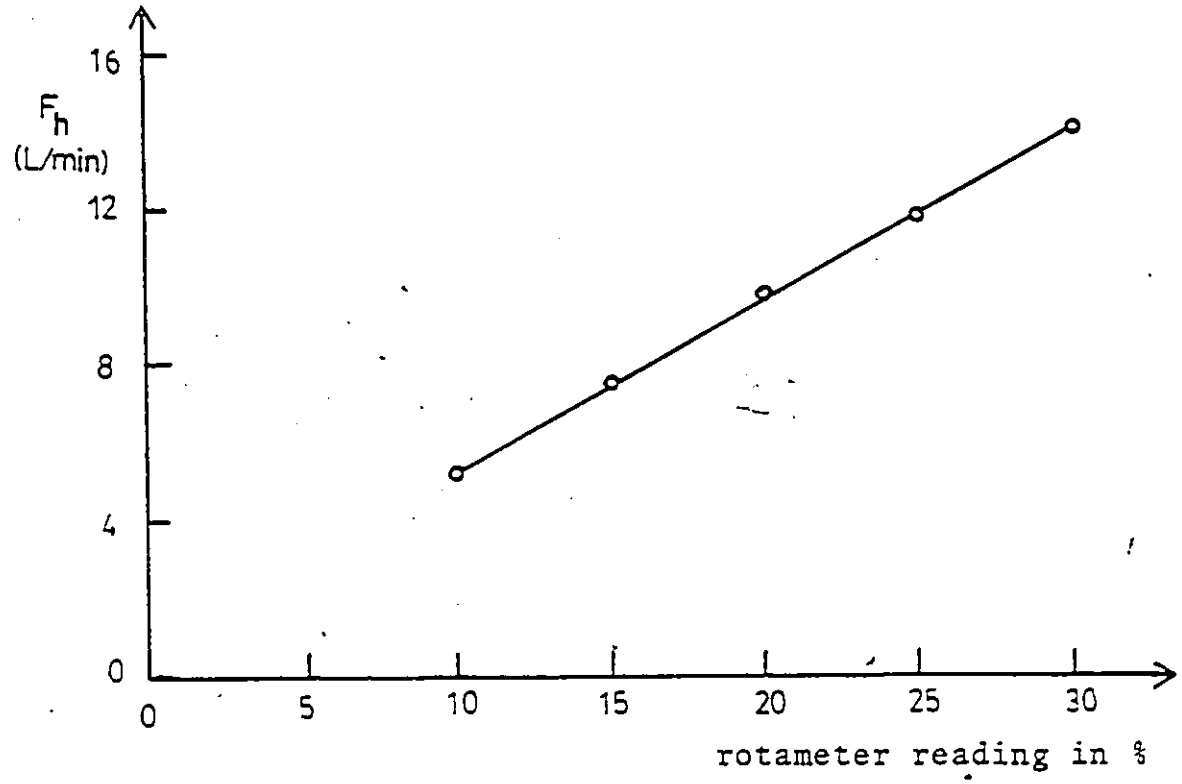


Figure D.1: Rotameter Calibration Curve.

Appendix E

TYPE T THERMOCOUPLE CURVE.

Type T thermocouple signals are nonlinear, as can be seen in Figure E.1 . This curve can be represented by the third order polynomial

$$T = a_0 + a_1 E + a_2 E^2 + a_3 E^3$$

where $T = {}^{o}C$,

E = millivolts,

$a_i (i=0,1,2,3)$ = parameters to be fitted.

Accurate temperature measurements require this model to be fitted to data from a type T thermocouple table. The fitted model is then incorporated to the computer control program. Experimental data from the Honeywell Thermocouple Tables[27] were used to fit the model above and the resulting equation is

$$T = 0.011 + 25.962\ E - 0.755\ E^2 + 0.035\ E^3$$

This equation is incorporated in line 1000 in the level/temperature control program.

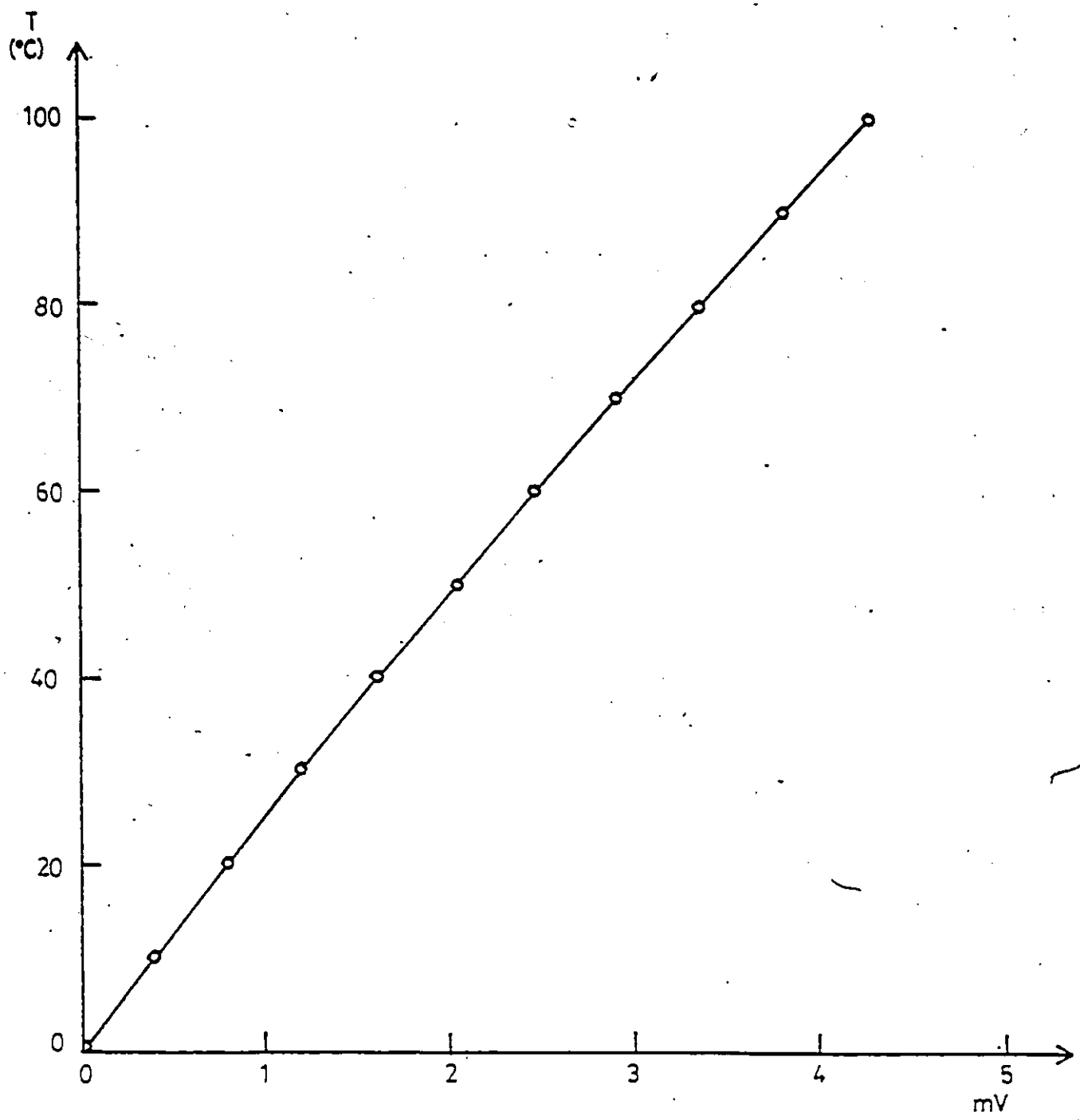


Figure E.1: Type T Thermocouple Curve.

Appendix F

LEVEL TRANSMITTER CALIBRATION CURVE.

Transducer operations are expected to be linear. The purpose of the level transmitter calibration curve is to infer about the linearity of the transformation level to VDC. The level transmitter translates float position into an air pressure signal. This signal is further transduced into VDC before it can be read by the ISAAC-Apple system. The level transmitter calibration curve shown in Figure F.1 was obtained experimentally, since no information from the manufacturer was available. The absolute span of the level transmitter is 0.62 dm. It has been amplified to 2.48 dm with the help of a lever. The solid line in Figure F.1 shows the ideal linear behavior to be expected from the transmitter. One can notice that experimental results are fairly linear over the whole span, and can therefore be described by a simple gain.

The transfer functions of both the level transmitter and the air-pressure-to-voltage transducer were not explicitly identified. They were lumped into $g_{11}(s)$ and $g_{12}(s)$.

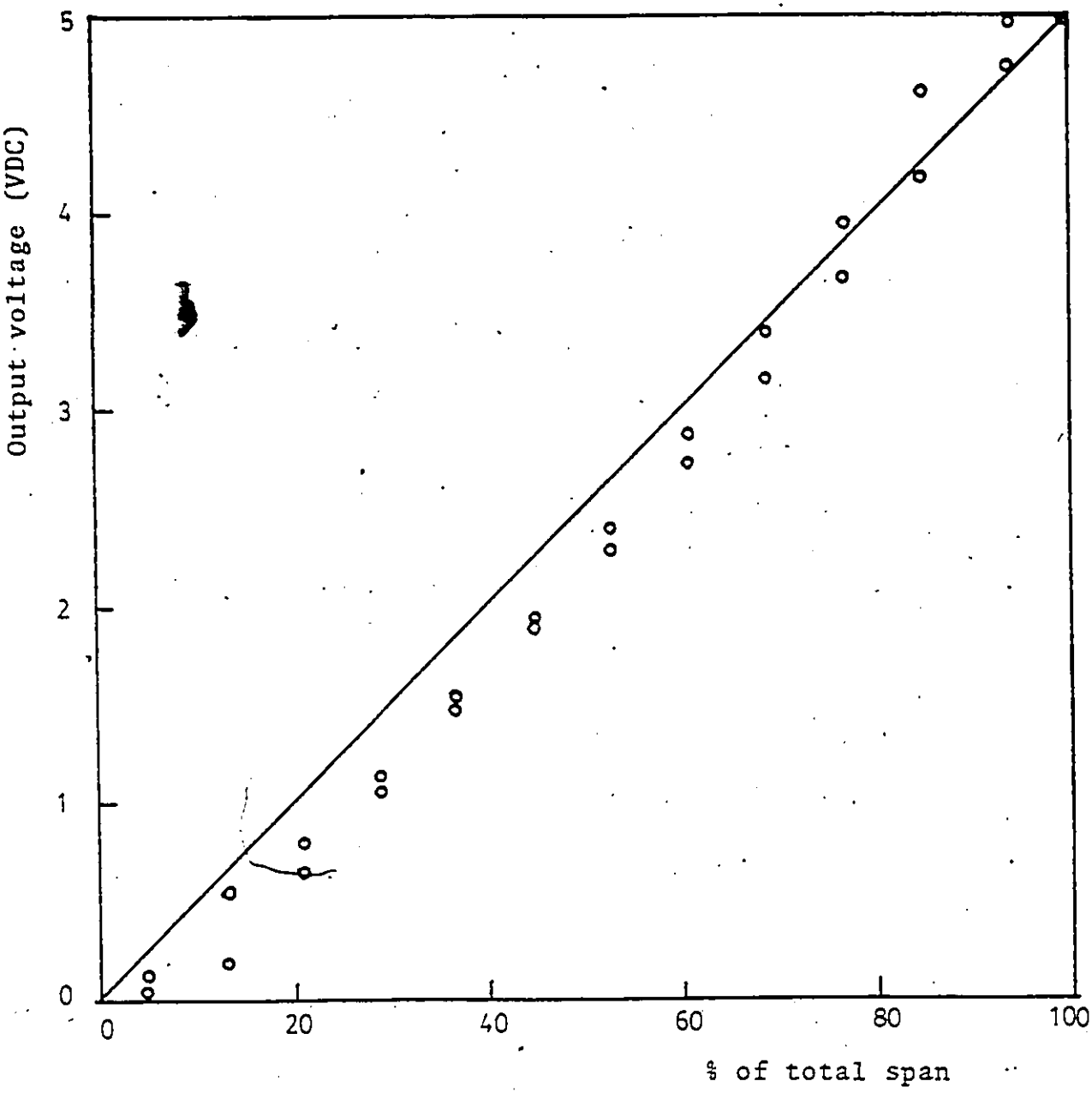


Figure F.1: Level Transmitter Calibration Curve.

Appendix G

POSITION PID LEVEL CONTROL PROGRAM LISTING AND FLOWCHART

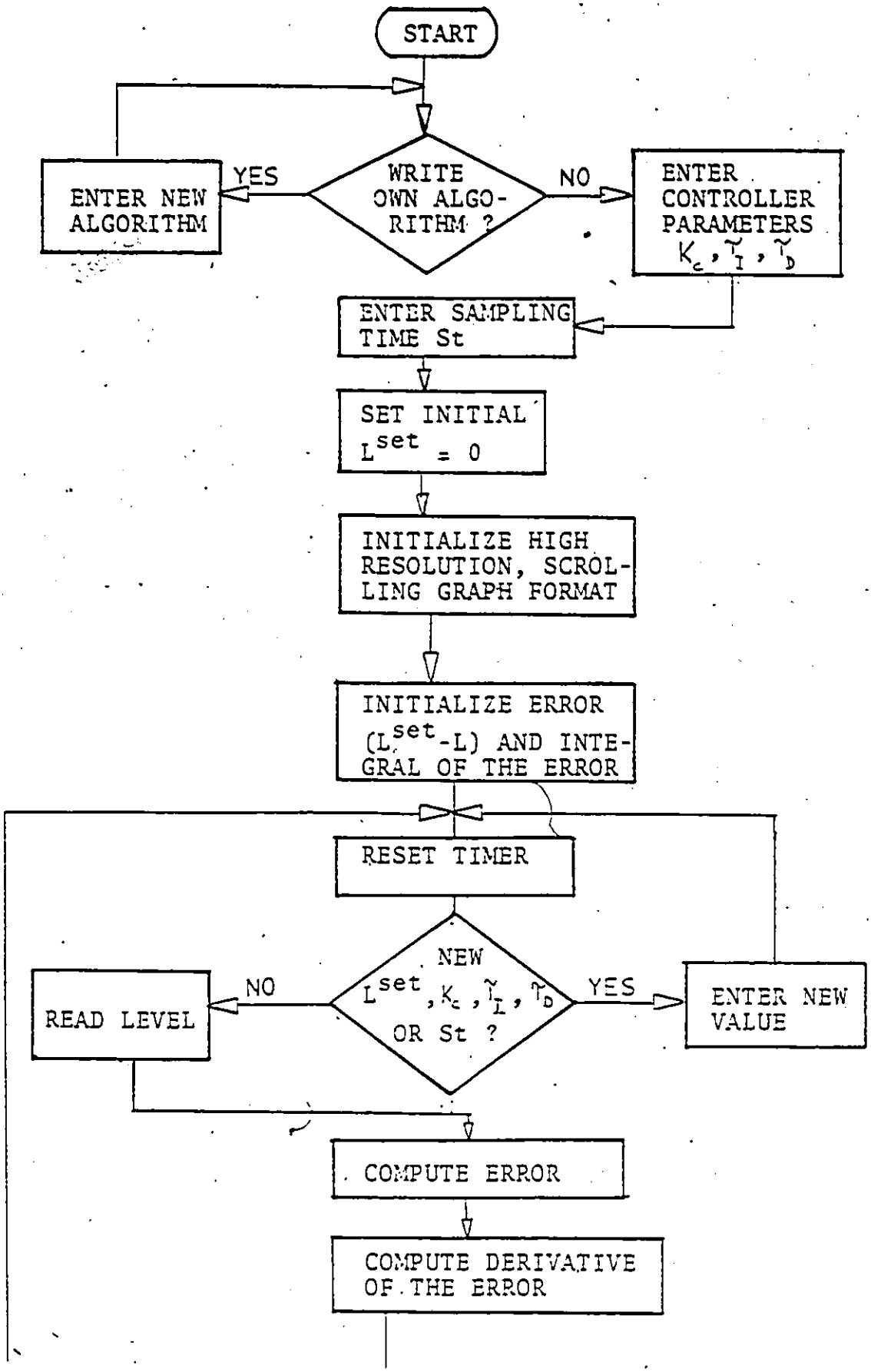
```
10  PRINT "============================"
20  PRINT "DIRECT DIGITAL CONTROL OF A"
30  PRINT "LEVEL LOOP USING THE   PID "
40  PRINT "POSITION ALGORITHM."
50  PRINT "============================"
60  REM  PROGRAM WRITTEN BY PAULO  SPRINGMANN IN JUNE 1984.
70  PRINT : PRINT
80  PRINT "REMARKS:"
90  PRINT "========"
100  PRINT
110  PRINT "THIS PROGRAM ALLOWS YOU TO CHOOSE BETWEEN WRITING YOUR OWN CON
     TROL ALGORITHM, OR CHOOSING THE STANDARD P, PI OR PID ALGORITHM. FOR
     PI AND PID MODES, ANTI-RESET WINDUP IS ALSO AVAILABLE."
120  PRINT "FOR P-ONLY CONTROL, SET INTEGRAL TIME ITAU = 30000 SEC., AND T
     HE DERIVATIVE TIME DTAU = 0 SEC."
130  PRINT "FOR PI MODE, SIMPLY SET DTAU = 0 SEC."
132  PRINT : PRINT
134  INPUT "DO YOU WISH TO WRITE YOUR OWN CONTROL ALGORITHM ? (Y/N)";A$
136  IF  ASC (A$) = 89 THEN  GOTO 5000
140  PRINT : PRINT : PRINT
150  INPUT "ENTER CONTROLLER GAIN, KC = ";KC
155  PRINT
160  INPUT "ENTER RESET TIME IN SEC., ITAU = ";ITAU
170  PRINT
180  INPUT "ENTER DERIVATIVE TIME IN SEC., DTAU = ";DTAU
190  PRINT
200  INPUT "ENTER SAMPLING TIME BETWEEN 1 AND 16 SEC., ST = ";ST
203  IF ST > 16 OR ST < 1 THEN  GOTO 200
204  INPUT "DO YOU WANT ANTI-RESET WINDUP TO BE IN EFFECT? (Y/N)";B$
205 LSET = 0
210  PRINT : PRINT : PRINT
220  PRINT "KC    = ";KC
230  PRINT "ITAU = ";ITAU" SEC"
240  PRINT "DTAU = ";DTAU" SEC"
250  PRINT "ST    = ";ST"   SEC"
260  PRINT "LSET = ";LSET"  DM"
270  PRINT : PRINT
280  PRINT "REMARK► LSET IS GIVEN AN INITIAL VALUE OF 0 DM, WHICH IS ITS S
     TEADY-STATE VALUE."
290  PRINT : PRINT
300  PRINT "WAIT FOR PROCESS TO LINE UP. PRESS <L> TO CHANGE LEVEL SETPOIN
     T AND PRESS <K> TO CHANGE KC. PRESS <I> TO CHANGE ITAU, AND <D> TO CH
     ANGE DTAU. PRESSING <S> ALLOWS YOU TO CHANGE ST."
310  & PAUSE = 20
320  REM  ALL THE COMMANDS BEGINNING WITH THE AMPERSAND (&) ARE LABSOFT CO
     MMANDS, AN EXTENSION OF APPLESOFT BASIC.
```

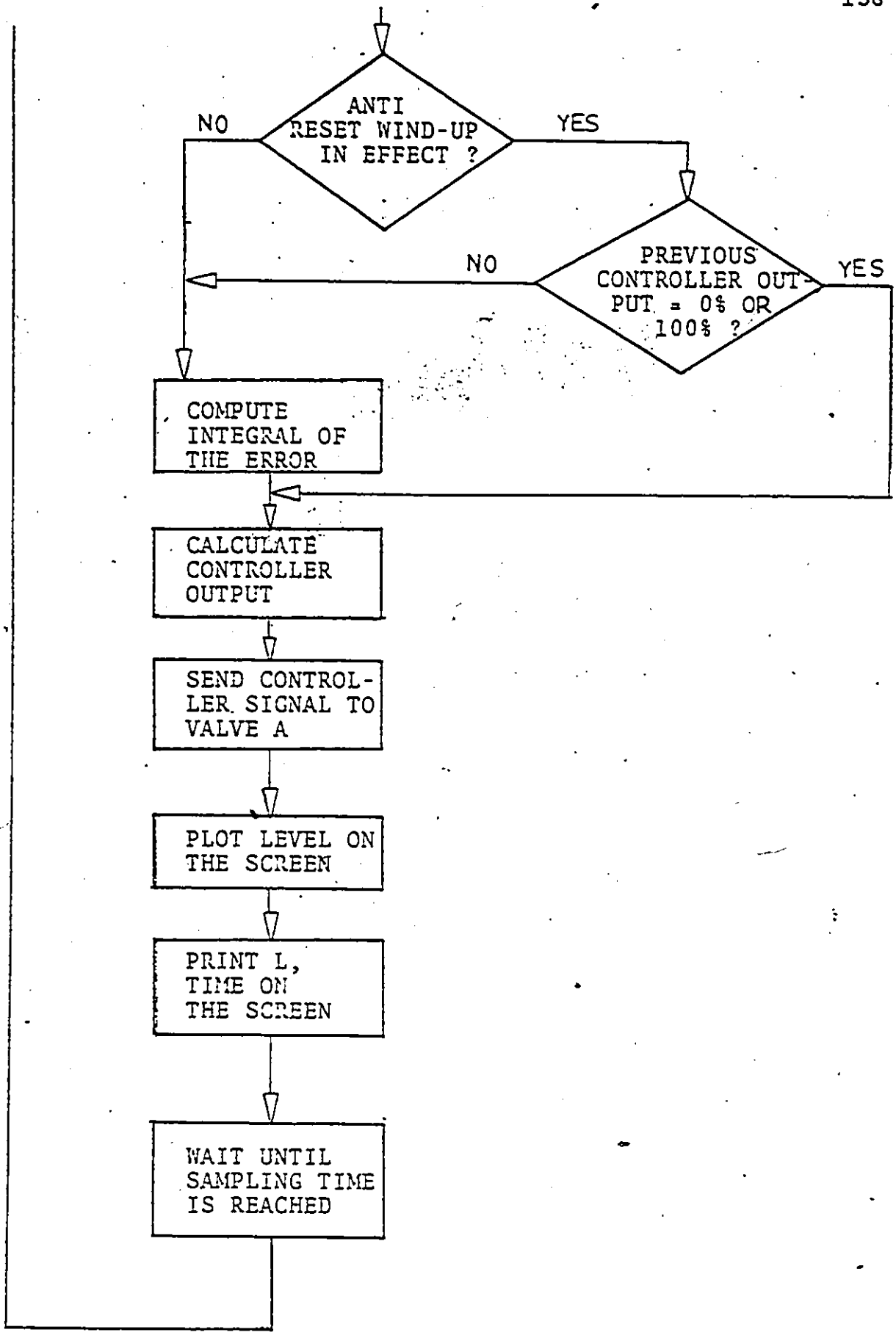
```
325  REM  EXAMPLE: THE LABSOFT PAUSE COMMAND (LINE 310) KEEPS THE PREVIOUS
      MESSAGE ON THE SCREEN FOR 20 SECONDS.
330  REM  DECIMAL VALUES RESULTING FROM A/D AND D/A CONVERSION CAN TAKE ON
     LY INTEGER VALUES BETWEEN 0 AND 4095. OUR A/D AND D/A CONVERTERS HAVE
      A 12-BIT RESOLUTION.
340  HGR
350  REM  HERE WE INTRODUCE HIGH-RESOLUTION GRAPHIC MODE.
360  & SCROLLSET
370  REM  THE &SCROLLSET COMMAND SETS THE LABSOFT SCROLLING GRAPH ROUTINE.

380  & OUTLINE
390  REM  &OUTLINE SIMPLY OUTLINES THE GRAPH SET BY &SCROLLSET.
400  & PLTFMT = 1
410  REM  THE &PLTFMT = "PLOT FORMAT" SETS THE COLOR TO BE USED ON THE GRA
     PH.
415 IERR = 0:E0 = 0
417  REM  THE COMMAND ABOVE INITIALIZES THE INTEGRAL OF THE ERROR AND THE
     PREVIOUS ERROR.
420  & CLRTIMER
430  REM  THE "CLEARTIMER" RESETS THE TIMER TO ZERO AND BEGINS COUNTING IN
      MILLISECONDS.
440 X =  PEEK ( - 16384)
450  IF X > 127 THEN  GOTO 1000
460  REM  THE TWO COMMANDS ABOVE CONTINUOUSLY CHECK WHETHER ANY KEY HAS BE
     EN PRESSED. IF TRUE, PROGRAM COUNTER JUMPS TO LINE 1000.
470  & AIN,(TV) = INVOLT%,(C#) = 0
480  REM  LINE 470 IS AN ANALOG INPUT (THE SIGNAL FROM THE LEVEL TRANSMITT
     ER), THE TARGET VARIABLE IS INVOLT%, AND THE A/D CHANNEL IS # 0.
490 L = 1.24 - INVOLT% / 1651
500  REM  1651 (SCALING FACTOR ) = 4095/2.48 DM, WHERE 2.48 DM IS THE TOTA
     L SPAN OF THE LEVEL TRANSMITTER.
510 E1 = LSET - L
520  REM  E1 IS THE ERROR AT PRESENT SAMPLING TIME.
522 DERR = (E1 - E0) / ST
524  REM  CALCULATES THE DERIVATIVE OF THE ERROR
525  IF  ASC (B$) = 78 THEN  GOTO 540
530  IF M = 4095 THEN  GOTO 560
535  IF M = 0 THEN  GOTO 560
540 IERR = IERR + E1 * ST
550  REM  THE COMMANDS ON LINES 530 AND 535 AVOID RESET WINDUP BY STOPPING
      THE INTEGRATION IF THE CONTROLLER OUTPUT IS ALREADY SATURATED.
560 M = 2048 + KC * (E1 + IERR / ITAU + DTAU * DERR) * ( - 1651)
565  REM  THIS IS THE DISCRETIZED PID ALGORITHM. THE CONSTANT 2048 IS THE
     BIAS TERM, CORRESPONDING TO 50% OUTPUT. THE CONSTANT 1651 IS A SCALIN
     G FACTOR. ITS NEGATIVE SIGN ENSURES THAT THE CONTROL LOOP HAS NEGATIV
     E FEEDBACK.
570  IF M > 4095 THEN M = 4095
580  IF M < 0 THEN M = 0
585  REM  THE CONSTRAINT ABOVE MEANS THAT THE VALUE CANNOT BE MORE THAN FU
     LLY OPEN OR FULLY CLOSED.
```

```
590  & AOUT,(DV) = M,(C#) = 2
600  REM  &AOUT OUTPUTS AN ANALOG SIGNAL TO VALUE C THROUGH CHANNEL # 2.
610  & NXTPLT = (4095 - INVOLT%) / 32
620  REM  THE (4095 - INVOLT%) OPERATION COMPENSATES FOR THE REVERSE ACTIO
     N OF THE LEVEL SENSOR. THE DIVIDE-DOWN CONSTANT 32 (= 4095/128) IS NE
     CESSARY, SINCE THE VERTICAL RESOLUTION OF THE GRAPH SCREEN IS 128 POI
     NTS HIGH.
630  PRINT  TAB( 01)L" DM"; TAB( 25)I * ST" SEC "
640 I = I + 1
645  REM  I IS THE LOOP COUNTER.
650 E0 = E1
660  & LOOK FOR TIMERIN,(TV) = 0,(TH) = ST * 1000: GOTO 420
670  REM  LINE 660 WAITS ELAPSED TIME TO REACH THE THRESHOLD (SAMPLING TIM
     E) BEFORE SAMPLING AGAIN.
690  REM  TIMERIN READS TIME ELAPSED SINCE TIMER HAS BEEN CLEARED (LINE 42
     0).
1000  POKE  - 16368,0
1010  REM  THE COMMAND ABOVE RESETS THE KEYBOARD STROBE.
1020  IF X = 204 THEN  INPUT "NEW LSET BETWEEN -/+ 1.24 DM = ";LSET
1025  IF LSET > 1.24 OR LSET <  - 1.24 THEN  GOTO 1020
1030  IF X = 203 THEN  INPUT "NEW KC = ";KC
1040  IF X = 201 THEN  INPUT "NEW ITAU = ";ITAU
1050  IF X = 196 THEN  INPUT "NEW DTAU = ";DTAU
1060  IF X = 211 THEN  INPUT "NEW ST = ";ST
1065  IF ST > 16 OR ST < 1 THEN  GOTO 1060
1070 I = 0
1080  GOTO 420
3000  LIST 10,50
5000  PRINT "    -WRITING YOUR OWN CONTROL ALGORITHM:"
5010  PRINT "     ----------------------------------"
5020  PRINT "     IN ORDER TO WRITE YOUR OWN CONTROL ALGORITHM, YOU WILL HA
     VE TO REWRITE THE CONTROL EQUATION ON LINE 560."
5030  PRINT "    TYPE (LIST 560) TO HAVE A LISTING OF THE CURRENT CONTROL E
     QUATION."
5040  PRINT "    WRITE YOUR OWN EQUATION AS LINE 560 AND HIT (RETURN). THE
     OLD EQUATION WILL BE SUBSTITUTED BY YOURS."
5050  PRINT "REMARKS: 1) THIS PROGRAM COMPUTES THE PREVIOUS ERROR E0, THE
     PRESENT ERROR E1, THE INTEGRAL OF THE ERROR IERR AND THE DERIVATIVE O
     F THE ERROR DERR."
5070  PRINT "2) THE FULL RANGE OF THE CONTROLLER OUTPUT IS 0 TO 4095, WHIC
     H AFTER THE D/A CONVERSION RESULTS IN 0 TO 5 V.D.C."
5080  PRINT "3) KEEP IN MIND THAT YOUR ALGORITHM SHOULD ALSO BE SCALED TO
     WORK IN THE RANGE 0 TO 4095."
5090  PRINT "ONCE YOUR ALGORITHM IS ON THE PROGRAM, TYPE (RUN) AND HIT (RE
     TURN). GOOD LUCK!"
5100  END
```

PID POSITION ALGORITHM CONTROL PROGRAM FLOW CHART




ANTI RESET WIND-UP IN EFFECT ?
NO
YES
PREVIOUS CONTROLLER OUTPUT = 0% OR 100% ?
NO
YES
COMPUTE INTEGRAL OF THE ERROR
CALCULATE CONTROLLER OUTPUT
SEND CONTROLLER SIGNAL TO VALVE A
PLOT LEVEL ON THE SCREEN
PRINT L, TIME ON THE SCREEN
WAIT UNTIL SAMPLING TIME IS REACHED

Appendix H

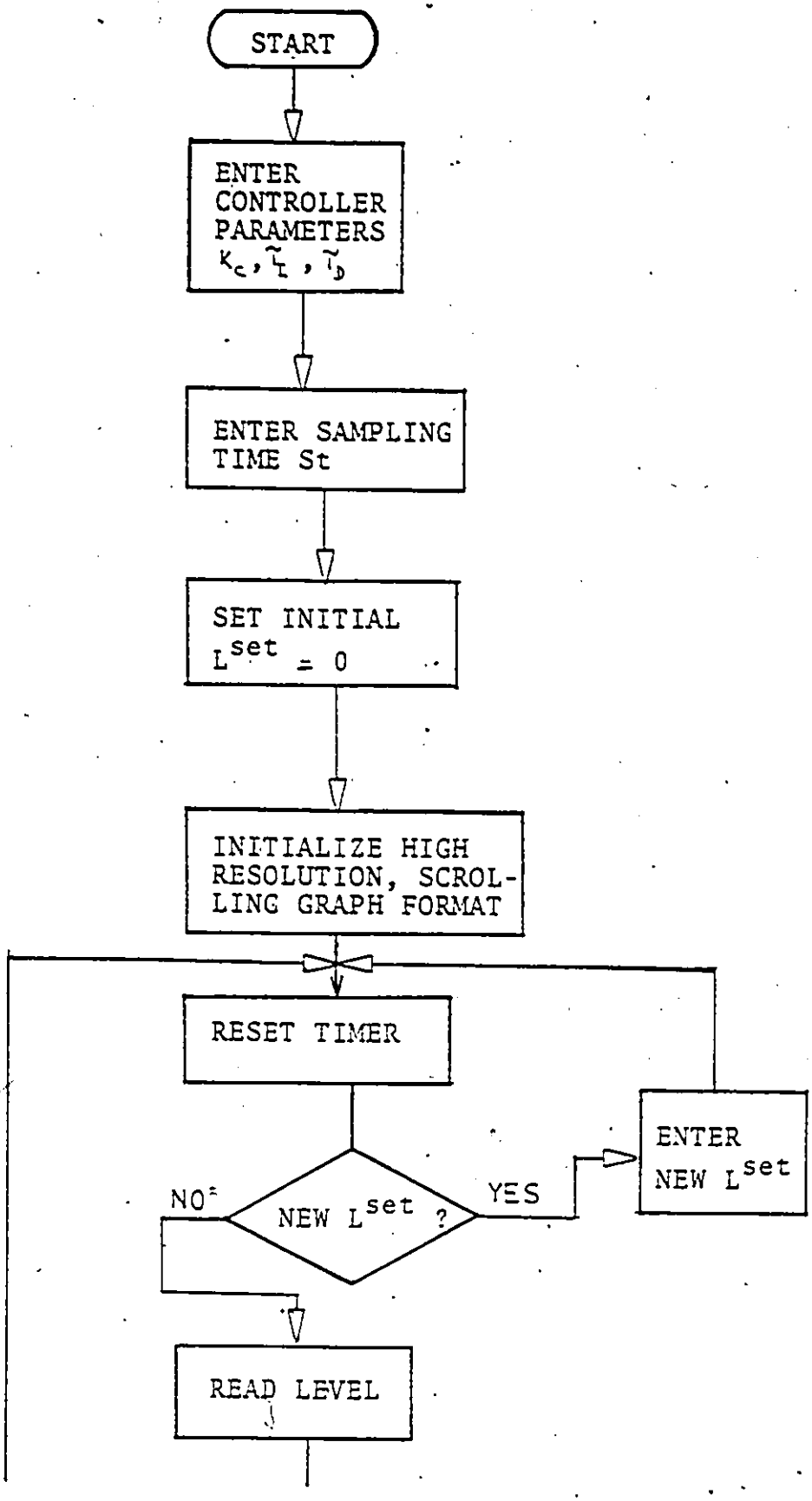
VELOCITY PID LEVEL CONTROL PROGRAM LISTING AND FLOWCHART

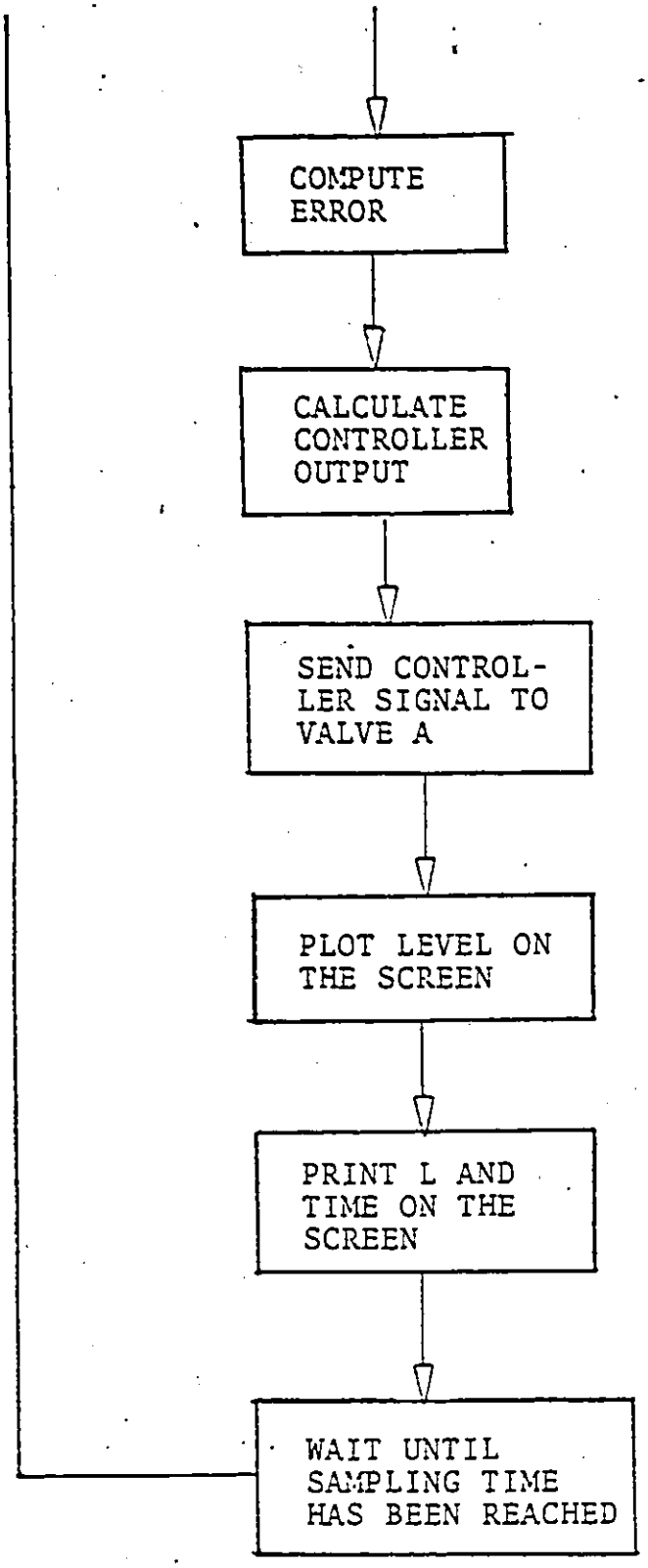
```
10  PRINT "============================"
20  PRINT "DIRECT DIGITAL CONTROL OF A"
30  PRINT "LEVEL LOOP USING THE   PID "
40  PRINT "VELOCITY ALGORITHM.      "
50  PRINT "============================"
60  REM  PROGRAM WRITTEN BY PAULO  SPRINGMANN IN JUNE 1984.
70  PRINT
80  PRINT "REMARKS:"
90  PRINT "========"
100  PRINT
110  PRINT "   1) THIS PROGRAM ALLOWS YOU TO CONTROL THE LEVEL USING PI OR
      PID CONTROL ALGORITHM."
120  PRINT "2) FOR PI-ONLY, SET THE DERIVATIVE TIME (DTAU) = 0."
130  PRINT "3) THE VELOCITY ALGORITHM RESULTS IN VERY POOR CONTROL IF YOU
     USE P-ONLY CONTROL. IF YOU STILL WANT TO TRY IT, SET ITAU = 30000 SEC
     . AND DTAU = 0 SEC."
140  PRINT : PRINT : PRINT
150  INPUT "ENTER CONTROLLER GAIN, KC = ";KC
155  PRINT
160  INPUT "ENTER RESET TIME IN SEC., ITAU = ";ITAU
170  PRINT
180  INPUT "ENTER DERIVATIVE TIME IN SEC., DTAU = ";DTAU
190  PRINT
200  INPUT "ENTER SAMPLING TIME BETWEEN 1 AND 16 SEC., ST = ";ST
203  IF ST > 16 OR ST < 1 THEN  GOTO 200
205 LSET = 0
210  PRINT : PRINT : PRINT
220  PRINT "KC    = ";KC
230  PRINT "ITAU = ";ITAU" SEC"
240  PRINT "DTAU = ";DTAU" SEC"
250  PRINT "ST    = ";ST"   SEC"
260  PRINT "LSET = ";LSET"  DM"
270  PRINT : PRINT
280  PRINT "REMARK : LSET = 0 DM  CORRESPONDS TO ITS DESIGN VALUE."
290  PRINT : PRINT
300  PRINT "WAIT FOR PROCESS TO LINE UP. PRESS ANY KEY FROM THE KEYBOARD I
     F YOU WISH TO CHANGE THE LEVEL SETPOINT."
310  & PAUSE = 4
320  REM  ALL THE COMMANDS BEGINNING WITH THE AMPERSAND (&) ARE LABSOFT CO
     MMANDS, AN EXTENSION OF APPLESOFT BASIC.
325  REM  DECIMAL VALUES RESULTING FROM A/D CONVERSION CAN TAKE ONLY INTEG
     ER VALUES BETWEEN 0 AND 4095. THIS IS BECAUSE THE A/D CONVERSION HAS
     A 12-BIT RESOLUTION.
330  REM  THE LABSOFT PAUSE COMMAND KEEPS THE MESSAGE ON LINE 300 ON THE S
     CREEN FOR 4 SECONDS.
340  HGR
350  REM  HERE WE INTRODUCE HIGH-RESOLUTION GRAPHIC MODE.
360  & SCROLLSET
370  REM  THE &SCROLLSET COMMAND SETS THE LABSOFT SCROLLING GRAPH ROUTINE.

380  & OUTLINE
```

```
390  REM  &OUTLINE SIMPLY OUTLINES THE GRAPH SET BY &SCROLLSET.
400  & PLTFMT = 1
410  REM  THE &PLTFMT = "PLOT FORMAT" SETS THE COLOR TO BE USED ON THE GRA
     PH.
420  & CLRTIMER
430  REM  THE "CLEARTIMER" RESETS THE TIMER TO ZERO AND BEGINS COUNTING IN
      MILLISECONDS."
440 X =  PEEK ( - 16384)
450  IF X > 127 THEN  GOTO 1000
460  REM  THE 2 COMMANDS ABOVE CONTINUOUSLY CHECK IF ANY KEY HAS BEEN PRES
     SED, IN WHICH CASE PROGRAM JUMPS TO LINE 1000.
470  & AIN,(TV) = INVOLT%,(C#) = 0
480  REM  LINE 470 CAUSES ISAAC TO READ THE ANALOG SIGNAL FROM THE LEVEL T
     RANSMITTER.
490 L = 1.24 - INVOLT% / 1651
500  REM  1651 IS A SCALING FACTOR (=4095 / 2.48 DM), WHERE 2.48 DM IS THE
      FULL SPAN OF THE LEVEL TRANSMITTER."
510 E2 = LSET - L
520  REM  E2 IS ERROR AT PRESENT TIME.
530  REM  E1 IS ERROR FROM PREVIOUS SAMPLE.
540 M1 = M0 + KC * ((E2 - E1) + ST / ITAU * E2 + DTAU / ST * (E2 - 2 * E1 +
     E0)) * ( - 1651)
550  REM  THIS IS THE VELOCITY FORM OF THE PID ALGORITHM. THE SCALING FACT
     OR 1651 HAS A NEGATIVE SIGN TO ENSURE THAT THE CONTROL LOOP HAS NEGAT
     IVE FEEDBACK."
560  IF M1 > 4095 THEN M1 = 4095
570  IF M1 < 0 THEN M1 = 0
580  REM  THE CONSTRAINT ABOVE MEANS THAT THE VALVE CANNOT BE MORE THAN FU
     LLY OPEN OR FULLY CLOSED.
590  & AOUT,(DV) = M1,(C#) = 2
600  REM  &AOUT OUTPUTS AN ANALOG SIGNAL TO VALVE C THROUGH CHANNEL # 2.
610  & NXTPLT = (4095 - INVOLT%) / 32
620  REM  THE (4095 - INVOLT%) OPERATION COMPENSATES FOR THE REVERSE ACTIO
     N OF THE LEVEL SENSOR. 32 IS A DIVIDE-DOWN VALUE, SINCE THE VERTICAL
     RESOLUTION OF THE GRAPH WINDOW IS 128.
630  PRINT  TAB( 01)L" DM"; TAB( 25)I * ST" SEC"
640 I = I + 1
650 E0 = E1:E1 = E2:M0 = M1
660  & LOOK FOR TIMERIN,(TV) = Q,(TH) = ST * 1000: GOTO 420
670  REM  LINE 660 READS TIME ELAPSED SINCE TIMER WAS CLEARED IN LINE 420.
      IF TIME ELAPSED HAS REACHED THE THRESHOLD (SAMPLING TIME), A NEW SAM
     PLING CYCLE BEGINS.
1000  POKE  - 16368,0
1010  REM  RESETS KEYBOARD STROBE.
1020  INPUT "ENTER NEW LSET WITHIN +/- 1.24 DM, LSET = ";LSET
1030  IF LSET > 1.24 THEN  GOTO 1090
1040  IF LSET <  - 1.24 THEN  GOTO 1090
1050  PRINT "YOUR NEW LSET = ";LSET" DM"
1060  PRINT
1070 I = 0
1080  GOTO 420
1090  PRINT "YOUR NEW LSET IS NOT WITHIN THE OPERABLE RANGE. TRY AGAIN!"
1100  GOTO 1020
```

PID VELOCITY ALGORITHM CONTROL PROGRAM FLOW CHART




COMPUTE ERROR
CALCULATE CONTROLLER OUTPUT
SEND CONTROL-LER SIGNAL TO VALVE A
PLOT LEVEL ON THE SCREEN
PRINT L AND TIME ON THE SCREEN
WAIT UNTIL SAMPLING TIME HAS BEEN REACHED

Appendix I

LEVEL/TEMPERATURE CONTROL PROGRAM LISTING AND FLOWCHART

```
10  PRINT "----------------------------------------"
20  PRINT "DIRECT DIGITAL CONTROL OF A"
30  PRINT "LEVEL/TEMPERATURE MULTIVARIABLE SYSTEM."
40  PRINT "----------------------------------------"
50  REM  PROGRAM WRITTEN BY PAULO SPRINGMANN IN JUNE 1984.
60  PRINT "   THIS PROGRAM ALLOWS YOU TO CONTROL SIMULTANEOUSLY THE LEVEL
     AND THE TEMPERATURE IN THE TANK."
70  PRINT "   PID POSITION AND VELOCITY ALGORITHMS ARE AVAILABLE FOR BOTH
     LOOPS."
80  PRINT "   DYNAMIC DECOUPLING IS ALSO AVAILABLE, IF YOU WISH."
100  PRINT "REMARKS: 1) FOR BOTH LOOPS YOU CAN CHOOSE AMONG P, PI OR PID M
     ODES.IF YOU CHOOSE P-ONLY MODE, THE POSITION ALGORITHM MUST BE USED."

110  PRINT " 2)IF YOU WANT TO DECOUPLE THE LOOPS OR USE P-ONLY FOR THE LEV
     EL, CHOOSE THE POSITION ALGORITHM FOR THE LEVEL."
115  PRINT " 3) THE DECOUPLING EFFECT IS ONLY NOTICEABLE IF LEVEL IS LOOSE
     LY TUNED (KL <= 1)."
120  PRINT
130  REM  VARIABLE DICTIONARY:
140  REM  ====================
150  REM   LEVEL LOOP:
154  REM  -----------
158  REM  ST = SAMPLING TIME IN SECONDS.
159  REM  RSP = REMOTE SETPOINT IS THE SETPOINT SIGNAL SENT TO THE MICROPR
     OCESSOR.
160  REM  L = LEVEL READING AT PRESENT TIME.
161  REM  LSET = LEVEL SETPOINT.
162  REM  KL = CONTROLLER GAIN.
166  REM  ILTAU = CONTROLLER INTEGRAL TIME IN SECONDS.
170  REM  DLTAU = CONTROLLER DERIVATIVE TIME IN SECONDS.
174  REM  LIERR = INTEGRAL OF THE ERROR.
178  REM  LDERR = DERIVATIVE OF THE ERROR.
180  REM  M1 = OUTPUT FROM CONTROLLER AT PRESENT TIME.
184  REM  M0 = OUTPUT FROM CONTROLLER 1 SAMPLING TIME AGO.
188  REM  L2ERR = ERROR AT PRESENT SAMPLING TIME.
192  REM  L1ERR = ERROR 1 SAMPLING TIME AGO.
195  REM  L0ERR = ERROR 2 SAMPLING TIMES AGO.
197  REM  LEVISE = INTEGRAL OF THE SQUARED ERROR.
200  REM  TEMPERATURE LOOP:
203  REM  -----------------
204  REM  T = TEMPERATURE READING AT PRESENT TIME.
205  REM  TSET = TEMPERATURE SETPOINT.
206  REM  KT = CONTROLLER GAIN.
208  REM  ITTAU = CONTROLLER INTEGRAL TIME IN SECONDS.
212  REM  DTTAU = CONTROLLER DERIVATIVE TIME IN SECONDS.
215  REM  TIERR = INTEGRAL OF THE ERROR.
217  REM  TDERR = DERIVATIVE OF THE ERROR.
220  REM  N1 = PRESENT OUTPUT FROM CONTROLLER.
222  REM  N0 = PREVIOUS OUTPUT FROM CONTROLLER.
225  REM  T2ERR = ERROR AT PRESENT TIME.
```

```
227  REM  T1ERR = ERROR 1 SAMPLING TIME AGO.
230  REM  T0ERR = ERROR 2 SAMPLING TIMES AGO.
235  REM  TEMPISE = INTEGRAL OF THE SQUARED ERROR.
240  REM  DISTURBANCE:
245  REM  ------------
250  REM  TH = TEMPERATURE OF THE INLET HOT WATER, CONTROLLED BY THE MICRO
     PROCESSOR.
255  REM
257  REM
258  REM  DECOUPLER:
259  REM  ----------
260  REM  D1 = PRESENT DECOUPLER SIGNAL.
270  REM  D0 = PREVIOUS DECOUPLER SIGNAL.
280  REM
290  REM
300  INPUT "LEVEL LOOP: VELOCITY (V) OR POSITION (P) ALGORITHM ? (V/P) ";A
     $
305  REM  THE VELOCITY ALGORITHM CANNOT BE USED FOR P-ONLY CONTROL, BECAUS
     E OF THE CANCELATION OF THE SETPOINT.
310  PRINT
320  INPUT "TEMPERATURE LOOP: VELOCITY (V) OR POSITION (P) ALGORITHM ? (V/
     P) ";B$
325  REM  AGAIN, DO NOT USE THE VELOCITY ALGORITHM FOR TEMPERATURE P-ONLY
     CONTROL.
330  PRINT
340  INPUT "LEVEL CONTROLLER: CHOOSE AMONG P-ONLY (1), PI (2) OR PID (3) C
     ONTROL ACTION. (1/2/3) ? ";LEVEL%
350  PRINT
360  INPUT "TEMPERATURE CONTROLLER: CHOOSE AMONG P-ONLY (1), PI (2) OR PID
      (3) CONTROL ACTION. (1/2/3) ? ";TEMP%
370  FOR I = 1 TO LEVEL%
380  IF I = 1 THEN  INPUT "LEVEL CONTROLLER GAIN KL = ";KL
390  IF I = 2 THEN  INPUT "LEVEL CONTROLLER RESET TIME IN SEC., ILTAU = ";
     ILTAU
400  IF I = 3 THEN  INPUT "LEVEL CONTROLLER DERIVATIVE TIME IN SEC., DLTAU
      = ";DLTAU
410  NEXT I
420  FOR J = 1 TO TEMP%
425  PRINT
430  IF J = 1 THEN  INPUT "TEMPERATURE CONTROLLER GAIN KT = ";KT
440  IF J = 2 THEN  INPUT "TEMPERATURE CONTROLLER INTEGRAL TIME ITTAU = ";
     ITTAU
450  IF J = 3 THEN  INPUT "TEMPERATURE CONTROLLER DERIVATIVE TIME DTTAU =
     ";DTTAU
460  NEXT J
470  PRINT
480  INPUT "ENTER SAMPLING TIME BETWEEN 1 AND 16 SEC., ST = ";ST
481  IF ST > 16 OR ST < 1 THEN  GOTO 480
482  REM  USE THE FASTEST SAMPLING TIME FOR BETTER CONTROL.
484  PRINT : PRINT
486  INPUT "ENTER INLET HOT WATER TEMPERATURE (BETWEEN 50 AND 70 C), TH =
     ";TH
488  REM  TH IS THE REMOTE SETPOINT FOR THE HONEYWELL MICROPROCESSOR.
490  PRINT
493  INPUT "ENTER LSET (WITHIN +/- 1.24 DM), LSET = ";LSET
```

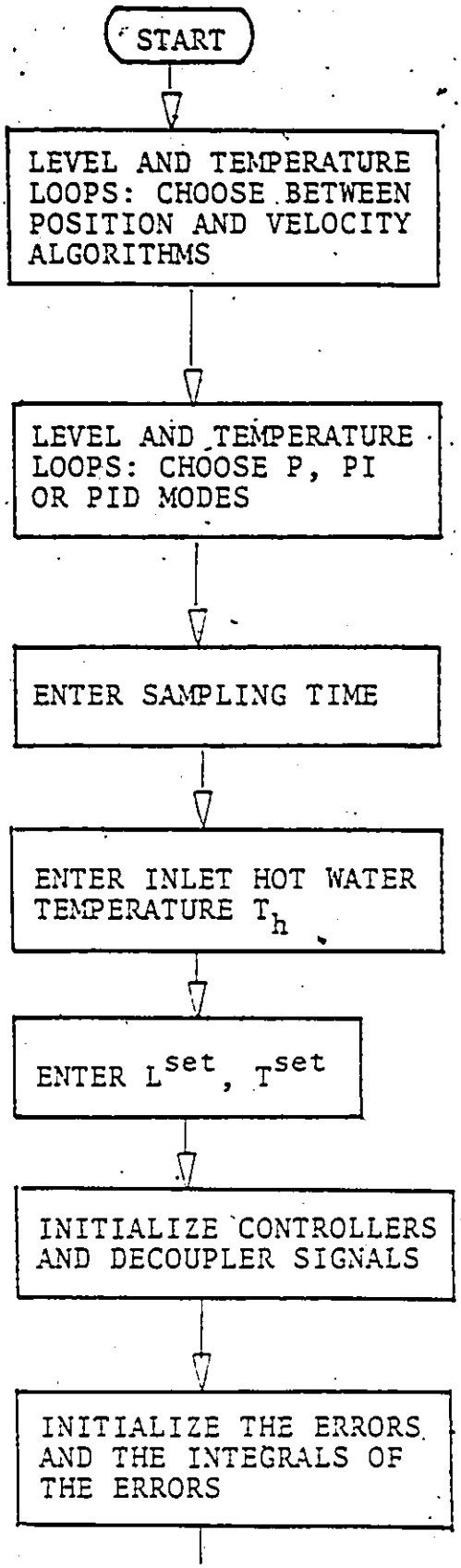
```
494  REM  LEVEL IS A PERTURBATION VARIABLE, THE DESIGN SETPOINT IS 0 DM AN
     D THE TOTAL SPAN IS -1.24=< L =< +1.24 DM.
495  PRINT
497  INPUT "ENTER TSET (BETWEEN 25 AND 35 C), TSET = ";TSET
498  REM  THE DESIGN TSET VARIES ACCORDING TO THE INLET COLD WATER TEMPERA
     TURE TC. TC = 4 C DURING WINTER AND TC=20 C DURING SUMMER.
499  PRINT : PRINT
500  PRINT "   WAIT FOR PROCESS TO LINE-UP: TO CHANGE THE LEVEL SETPOINT P
     RESS <L>. TO CHANGE THE TEMPERATURE SETPOINT PRESS <T>. TO CHANGE THE
      INLET HOT WATER TEMPERATURE PRESS <D>."
520  IF LEVEL% = 1 THEN ILTAU = 1000:DLTAU = 0
530  IF LEVEL% = 2 THEN DLTAU = 0
540  IF TEMP% = 1 THEN ITTAU = 1000:DTTAU = 0
550  IF TEMP% = 2 THEN DTTAU = 0
560  PRINT
570  INPUT "DO YOU WANT DYNAMIC DECOUPLING TO BE IN EFFECT ? (Y/N) ";C$
575  REM  USE DYNAMIC DECOUPLING ONLY WITH THE LEVEL POSITION ALGORITHM. T
     HE TEMPERATURE ALGORITHM CAN BE EITHER POSITION OR VELOCITY.
576 MR = 2048:NR = 1638
577  IF  ASC (C$) = 89 THEN MR = 1671
578  REM  MR AND NR ARE RESPECTIVELY THE LEVEL AND THE TEMPERATURE STEADY-
     STATE CONTROLLER OUTPUT (BIAS).
579  REM  IF DECOUPLER IS IMPLEMENTED, ITS STEADY-STATE OUTPUT IS D1=377.
     IN THIS CASE, MR=2048 HAS TO BE REDUCED TO MR = 2048 - 377 = 1671, TO
      AVOID LEVEL OFFSET.
580 A = .011:B = 25.962
590 C =  - .755:D = .035
595  REM  THE CONSTANTS ABOVE ARE THE PARAMETERS FROM THE THERMOCOUPLE'S P
     OLYNOMIAL.
600 M0 = 1671:N0 = 1638:D0 = 0:D1 = 0
605  REM  M0 AND N0 ARE THE CONTROLLERS' OUTPUTS AT STEADY-STATE. DECOUPLI
     NG IS NOT COMPUTED FOR THE FIRST LOOP.
610 K = 0:L0ERR = 0:L1ERR = 0
615  REM  K IS THE LOOP COUNTER.
620 T0ERR = 0:T1ERR = 0
630 L1ERR = 0:T1ERR = 0
635  REM  THE ERRORS AND THE INTEGRALS OF THE ERRORS ARE INITIALIZED.
640  HGR
650  & SCROLLSET
655  REM  &SCROLLSET IS A LABSOFT COMMAND THAT SETS THE SCROLLING GRAPH.
660  & PLTFMT = 1,2
665  REM  &PLTFMT STANDS FOR PLOT FORMAT. IT SETS THE COLORS ON THE SCREEN
      DISPLAY.
670  & OUTLINE
675  REM  &OUTLINE OUTLINES THE GRAPH.
680  & CLRTIMER
685  REM  THE "CLEAR TIMER" COMMAND RESETS THE TIMER TO ZERO AND BEGINS CO
     UNTING IN MILLISECONDS"
690 K = K + 1
700 X =  PEEK (-16384)
710  IF X > 127 THEN  GOTO 5000
715  REM  LINES 700 AND 710 CONTINUOUSLY CHECK IF THE KEYBOARD HAS BEEN PR
     ESSED. IF TRUE, PROGRAM COUNTER JUMPS TO 5000.
720 RSP = 819.2 + 32.758 * TH
```

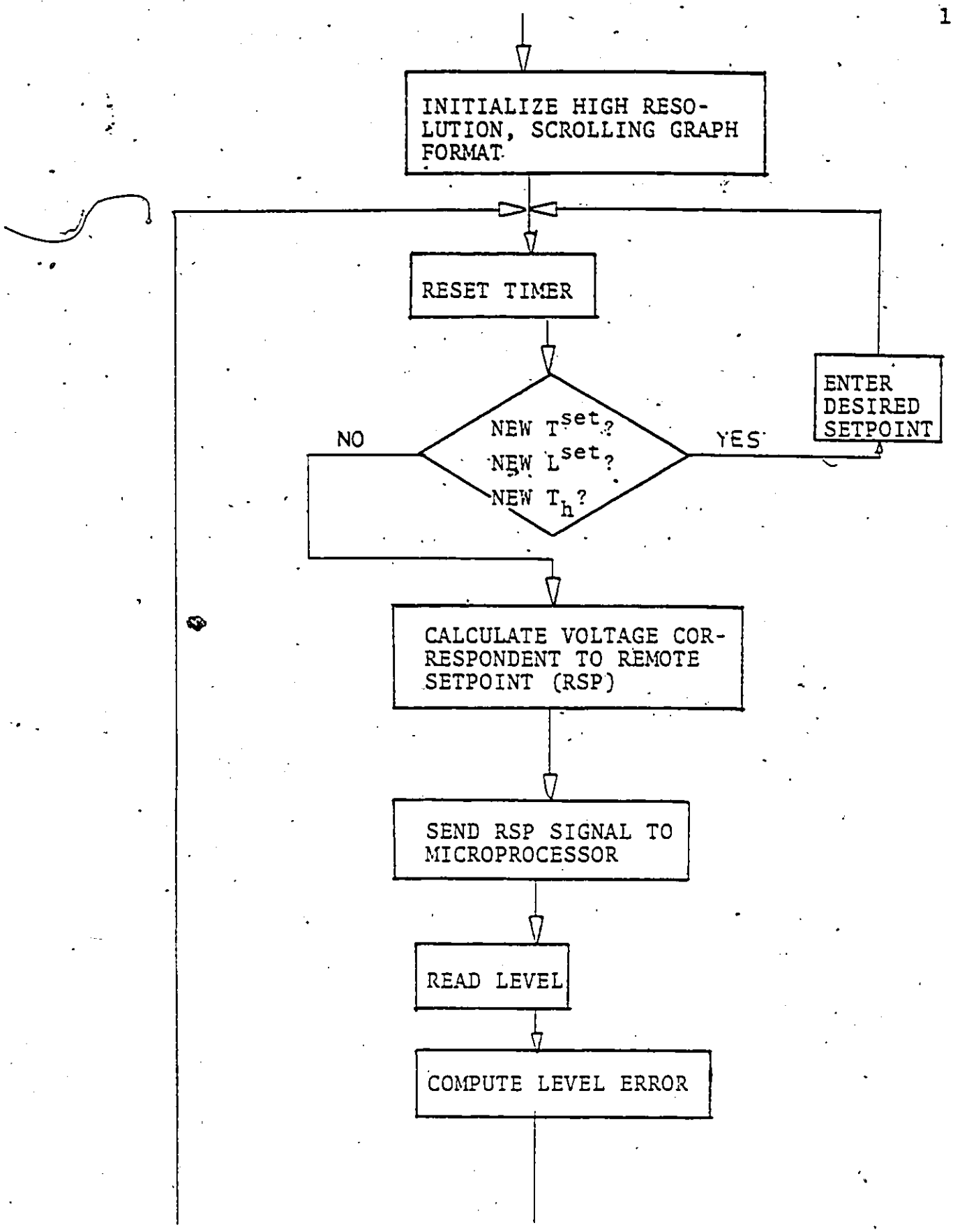
```
725  REM  RSP STANDS FOR REMOTE SETPOINT. RSP IS THE TH SETPOINT SIGNAL SE
     NT TO THE MICROPROCESSOR. WHEN TH = 0C, RSP MUST BE 1 VDC (=819.2 IN
     DIGITAL FORM). WHEN TH = 100 C, RSP MUST BE 5 VDC (=4095).
730  & AOUT,(DV) = RSP,(C#) = 0
735  REM  ANALOG OUTPUT, DATA VALUE=RSP, D/A CHANNEL # 0. THIS COMMAND OUT
     PUTS THE RSP SIGNAL TO THE MICROPROCESSOR.
740  & AIN,(TV) = INVOLT%,(C#) = 0
745  REM  ANALOG INPUT, TARGET VARIABLE = INVOLT%, A/D CHANNEL # 0. THIS C
     OMMAND INPUTS THE LEVEL TRANSMITTER SIGNAL (0 TO 5 V.D.C.)
750 L = 1.24 - INVOLT% / 1651
755  REM  1651 IS A SCALING FACTOR (=4095/2.48 DM), WHERE 2.48 DM IS THE T
     OTAL LEVEL SPAN.
760 L2ERR = LSET - L
765  REM  COMPUTES THE PRESENT ERROR.
770 LEVISE = LEVISE + L2ERR ^ 2 * ST
775  REM  COMPUTES THE INTEGRAL OF THE SQUARED ERROR.
777  IF ASC (A$) = 86 THEN  GOTO 900
780  REM  CHECKS IF VELOCITY ALGORITHM IS TO BE USED.
790  IF LEVEL% < 3 THEN  GOTO 810
800 LDERR = (L2ERR - L1ERR) / ST
805  REM  CALCULATES THE DERIVATIVE OF THE ERROR.
810  IF LEVEL% < 2 THEN  GOTO 830
820 LIERR = LIERR + L2ERR * ST
825  REM  CALCULATES THE INTEGRAL OF THE ERROR.
830 M1 = MR + KL * (L2ERR + LIERR / ILTAU + LDERR * DLTAU) * ( - 1651) + D
     1
835  REM  EQUATION ABOVE IS THE LEVEL POSITION ALGORITHM. IF NO DECOUPLING
      IS IMPLEMENTED, MR=2048, WHICH IS EQUIVALENT TO THE VALVE 50% OPEN.
     THE SCALING FACTOR 1651 HAS NEGATIVE SIGN SO THAT THE CONTROL LOOP HA
     S NEGATIVE FEEDBACK.
836  REM  D1 IS THE DECOUPLER SIGNAL.
840  GOTO 920
900 M1 = M0 + KL * ((L2ERR - L1ERR) + ST / ILTAU * L2ERR + DLTAU / ST * (L
     2ERR - 2 * L1ERR + L0ERR)) * ( - 1651)
910  REM  EQUATION ABOVE IS THE LEVEL VELOCITY ALGORITHM.
920  IF M1 > 4095 THEN M1 = 4095
940  IF M1 < 0 THEN M1 = 0
945  REM  THE CONSTRAINTS ABOVE REPRESENT THE PHYSICAL LIMITATION OF THE V
     ALVE OPENING.
950  & AOUT,(DV) = M1,(C#) = 2
955  REM  ANALOG OUTPUT, DATA VALUE = M1, D/A CHANNEL # 2. THIS IS THE OUT
     PUT TO VALVE A.
960  & NXTPLT = (4095 - INVOLT%) / 32
965  REM  THE COMMAND "NEXT PLOT" PLOTS THE LEVEL ON THE SCREEN. THE SUBTR
     ACTION (4095 - INVOLT%) COMPENSATES FOR THE REVERSE ACTION OF THE LEV
     EL TRANSMITTER.
966  REM  THE CONSTANT 32 (=4095/128) IS A DIVIDE DOWN VALUE, SINCE THE VE
     RTICAL RESOLUTION OF THE GRAPH WINDOW IS 128.
970  & WRDEV,(DV) = 2,(W#) = 2,(D#) = 0
975  REM  WRITE DEVICE SPECIFIES THE TYPE OF COLD JUNCTION COMPENSATION WE
      ARE USING. SEE I-130 MANUAL.
980  & AIN,(D#) = 0,(TV) = MV,(C#) = 0
985  REM  LINE 980 INPUTS THE THERMOCOUPLE MILLIVOLT SIGNAL THROUGH THE I-
     140 BOARD.
```

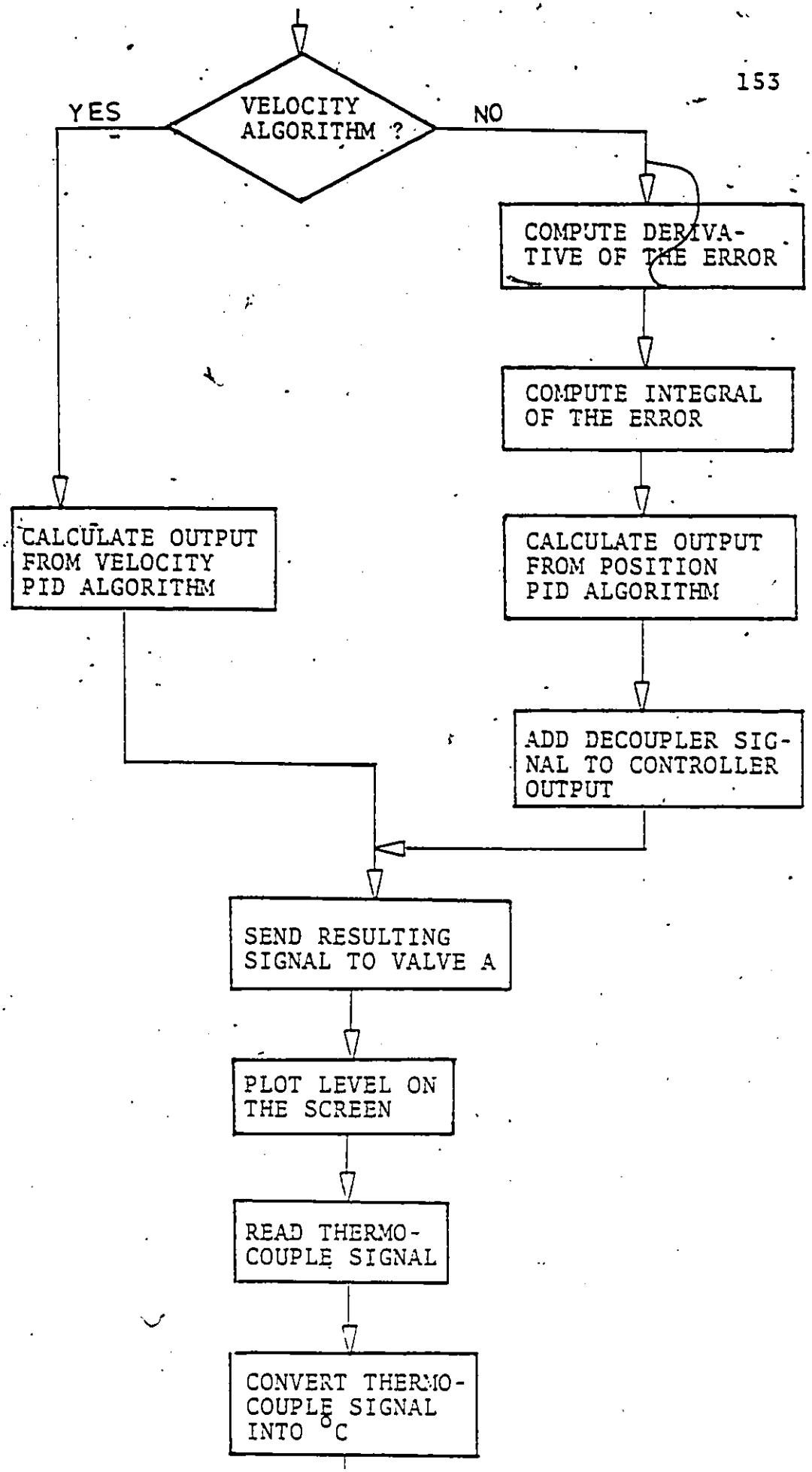
```
990 V = 5000 * (MV - 2048) / (248 * 2048)
995  REM  LINE 990 CONTAINS A DIVIDE DOWN VALUE DETERMINED BY THE GAIN EQU
     ATION OF THE THERMOCOUPLE BEING USED. SEE I-130 MANUAL.
1000 T = Z + A + V * (B + V * (C + D * V))
1005  REM  LINEARIZES THE THERMOCOUPLE OUTPUT. IT IS THE FINAL STEP OF THE
      TRANSFORMATION OF MILLIVOLTS INTO DEGREES CELSIUS.
1010 T2ERR = TSET - T
1015  REM  COMPUTES THE PRESENT ERROR.
1020 TEMPISE = TEMPISE + T2ERR ^ 2 * ST
1025  REM  COMPUTES THE INTEGRAL OF THE SQUARED ERROR.
1030  IF  ASC (B$) = 86 THEN  GOTO 1100
1035  REM  CHECKS IF VELOCITY ALGORITHM IS TO BE USED.
1040  IF TEM% < 3 THEN  GOTO 1060
1050 TDERR = (T2ERR - T1ERR) / ST
1055  REM  CALCULATES THE DERIVATIVE OF THE ERROR.
1060  IF TEMP% < 2 THEN  GOTO 1080
1070 TIERR = TIERR + T2ERR * ST
1075  REM  CALCULATES THE INTEGRAL OF THE ERROR.
1080 N1 = NR + KT * (T2ERR + TIERR / ITTAU + TDERR * DTTAU) * ( - 409.5) *
     .67
1085  REM  THIS IS THE TEMPERATURE POSITION ALGORITHM. THE BIAS TERM NR=16
     38 CORRESPONDS TO A 2 VDC CONTROLLER OUTPUT (VALVE B 40% OPEN).
1086  REM  THE SCALING FACTOR 409.5 (=4095/10 C). 10 C IS THE TEMPERATURE
     FULL SPAN. ITS NEGATIVE SIGN IS NECESSARY SO THAT THE CONTROL LOOP HA
     S NEGATIVE FEEDBACK.
1090  GOTO 1110
1100 N1 = N0 + KT * ((T2ERR - T1ERR) + ST / ITTAU * T2ERR + DTTAU / ST * (
     T2ERR - 2 * T1ERR + T0ERR)) * ( - 409.5) * .67
1105  REM  EQUATION ABOVE IS THE TEMPERATURE VELOCITY ALGORITHM.
1110  IF N1 > 4095 * .67 THEN N1 = 4095 * .67
1120  IF N1 < 0 THEN N1 = 0
1125  REM  CONSTRAINTS ABOVE ARISE FROM THE LIMITATION IN VALVE OPENING. V
     ALVE B CANNOT OPEN MORE THAN 67% (3.35 VDC), OTHERWISE INLET MAXIMUM
     FLOWRATE > OUTLET MAXIMUM FLOWRATE.
1130  & AOUT,(DV) = N1,(C#) = 1
1135  REM  ANALOG OUTPUT, DATA VALUE = N1, D/A CHANNEL #1. OUTPUTS THE TEM
     PERATURE CONTROLLER SIGNAL.
1140  & NXTPLT =  - 25.6 + 2.56 * T
1145  REM  EQUATION ABOVE TRANSFORMS A 10 TO 60 C TEMPERATURE SCALE INTO 0
      TO 128 SCALE, WHERE 128 IS THE VERTICAL RESOLUTION OF THE GRAPH WIND
     OW.
1150  IF  ASC (C$) = 78 THEN  GOTO 1200
1155  REM  CHECKS IF DYNAMIC DECOUPLING IS IN EFFECT. IF NOT, D1 = 0 VDC.
1160 D1 = N1 * (2.2 + .115 * ST) - N0 * (2.2 - .115 * ST) + D0 * (7.86 -
     ST / 2)) / (7.86 + ST / 2)
1165  REM  EQUATION ABOVE COMPUTES THE DECOUPLER OUTPUT, BASED ON A LEAD-L
     AG APPROXIMATED BY TRAPEZOIDAL INTEGRATION.
1170 D0 = D1
1200 L0ERR = L1ERR:L1ERR = L2ERR
1210 T0ERR = T1ERR:T1ERR = T2ERR
1220 M0 = M1:N0 = N1
```

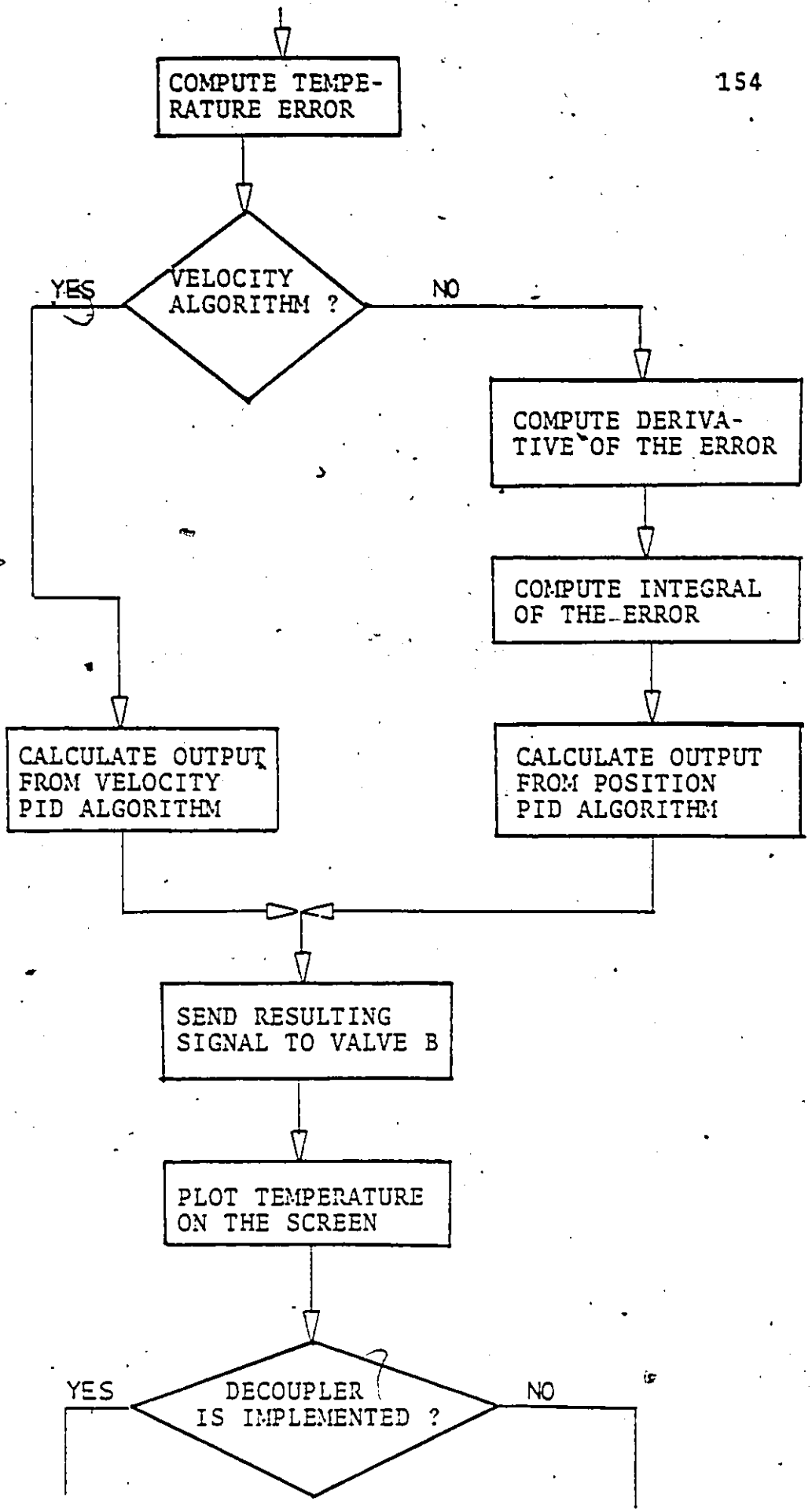
```
1230  PRINT  TAB( 01)"*  TIME = ";(K - 1) * ST; TAB( 16)D1
1240  PRINT  TAB( 01)"** TSET =  ";TSET; TAB( 16)"** T = ";T
1250  PRINT  TAB( 01)"***LSET =  ";LSET; TAB( 16)"***L = ";L
1260  & LOOK FOR TIMERIN,(TV) = Q,(TH) = ST * 1000: GOTO 680
1300  REM  COMAND ABOVE READS TIMER UNTIL TIME ELAPSED EQUALS THE THRESHOL
    D (SAMPLING TIME).
5000  POKE  - 16368,0
5005  REM  RESETS KEYBOARD STROBE
5010 LEVISE = 0:TEMPISE = 0
5015  REM  INITIATES LEVISE AND TEMPISE.
5020  IF X = 212 THEN  INPUT "ENTER NEW TSET = ";TSET
5030  IF X = 204 THEN  INPUT "ENTER NEW LSET = ";LSET
5040  IF X = 196 THEN  INPUT "ENTER NEW INLET HOT WATER TEMPERATURE TH = "
    ;TH
5050  GOTO 680
```

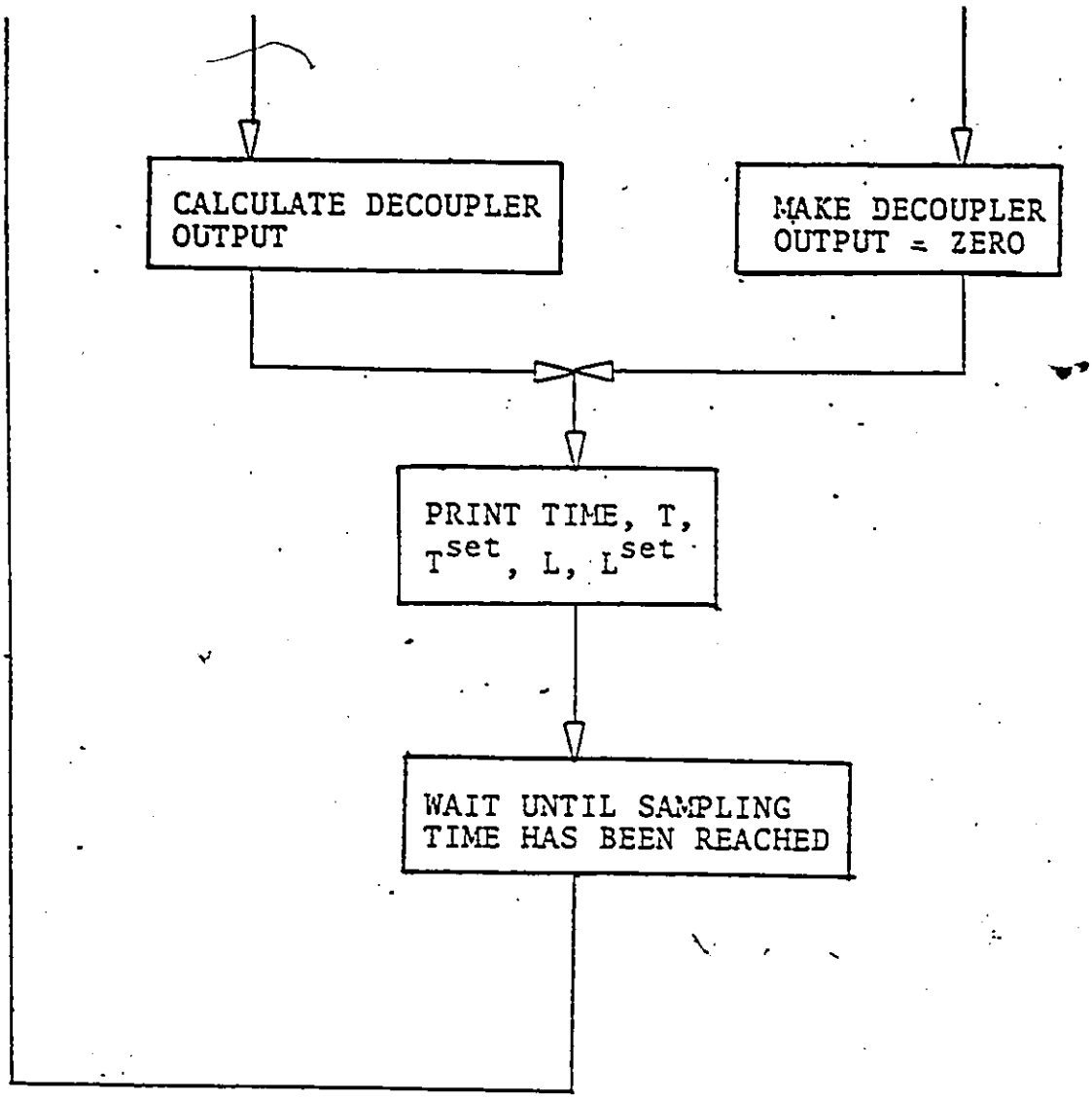
LEVEL & TEMPERATURE CONTROL PROGRAM FLOW CHART




INITIALIZE HIGH RESOLUTION, SCROLLING GRAPH FORMAT
RESET TIMER
NEW T^{set}?
NEW L^{set}?
NEW T_h?
NO
YES
ENTER DESIRED SETPOINT
CALCULATE VOLTAGE CORRESPONDENT TO REMOTE SETPOINT (RSP)
SEND RSP SIGNAL TO MICROPROCESSOR
READ LEVEL
COMPUTE LEVEL ERROR


VELOCITY ALGORITHM ?
YES
NO
COMPUTE DERIVATIVE OF THE ERROR
COMPUTE INTEGRAL OF THE ERROR
CALCULATE OUTPUT FROM VELOCITY PID ALGORITHM
CALCULATE OUTPUT FROM POSITION PID ALGORITHM
ADD DECOUPLER SIGNAL TO CONTROLLER OUTPUT
SEND RESULTING SIGNAL TO VALVE A
PLOT LEVEL ON THE SCREEN
READ THERMOCOUPLE SIGNAL
CONVERT THERMOCOUPLE SIGNAL INTO °C


COMPUTE TEMPE-
RATURE ERROR
VELOCITY
ALGORITHM ?
YES
NO
COMPUTE DERIVA-
TIVE OF THE ERROR
COMPUTE INTEGRAL
OF THE ERROR
CALCULATE OUTPUT
FROM VELOCITY
PID ALGORITHM
CALCULATE OUTPUT
FROM POSITION
PID ALGORITHM
SEND RESULTING
SIGNAL TO VALVE B
PLOT TEMPERATURE
ON THE SCREEN
DECOUPLER
IS IMPLEMENTED ?
YES
NO


CALCULATE DECOUPLER OUTPUT
MAKE DECOUPLER OUTPUT = ZERO
PRINT TIME, T, T^{set}, L, L^{set}
WAIT UNTIL SAMPLING TIME HAS BEEN REACHED

Appendix J

PROCESS IDENTIFICATION PROGRAM LISTINGS

```
5  TEXT
10  PRINT "-----------------------"
20  PRINT "PROCESS IDENTIFICATION:"
40  PRINT "G11 RESPONSE CURVE."
60  PRINT "=========================="
80  PRINT "REMARKS:"
100  PRINT "   1) THE OBJECTIVE OF THIS PROGRAM IS TO OBTAIN THE LEVEL RES
     PONSE CURVE TO A VOLTAGE STEPCHANGE ON THE SIGNAL TO VALVE A."
110  PRINT "   2) THE LEVEL RESPONSE WILL BE PLOTTED ON THE CHART RECORDER
     , WHICH MUST BE ON-LINE. SET THE CHART SPEED = 5 MM/MINUTE, AND HOT W
     ATER FLOWRATE AT 30% ROTAMETER READING."
120  PRINT "   3) THE MAGNITUDE OF THE STEPCHANGE MUST BE IN THE ORDER OF
     0.1 V.D.C, OTHERWISE THE TANK MAY RUN DRY OR OVERFLOW BEFORE NEW EQUI
     LIBRIUM HAS BEEN REACHED."
160  PRINT "   4) THE STEPCHANGE WILL TAKE PLACE AT THE MOMENT YOU TOUCH T
     HE KEYBOARD AFTER PROGRAM IS RUNNING. AT THAT MOMENT THE LOOP WILL OP
     EN, AND THE CHOSEN OPEN-LOOP VOLTAGE WILL BECOME THE SIGNAL TO THE VA
     LVE."
165  INPUT "CONTINUE ? (Y/N)";A$
167  IF  ASC (A$) = 78 THEN  STOP
170  PRINT"   5) IT IS ADVISABLE THAT YOU RUN THE PROGRAM IN CLOSED-LOOP
     FOR A WHILE SO THAT YOU GET AN IDEA ABOUT THE MAGNITUDE OF THE PRESEN
     T CLOSED-LOOP SIGNAL WHEN PROCESS IS LINED-UP.
175  PRINT "WHEN ASKED TO INPUT VOLTS (THE OPEN-LOOP VOLTAGE), ENTER ANY V
     ALUE, IT IS NOT IMPORTANT NOW."
180  PRINT "   6) ONCE YOU KNOW THE CURRENT VOLTAGE SIGNAL, BREAK THE PROG
     RAM AND RUN IT AGAIN. WHEN ASKED ABOUT YOUR STEPCHANGE (ACTUALLY THE
     OPEN-LOOP VOLTAGE), ENTER A VALUE WHICH IS 0.1 VDC GREATER OR SMALLER
      THAN THAT CURRENT VOLTAGE."
190  PRINT "   7)SET THE LEVEL SETPOINT CLOSE TO EITHER EXTREME OF THE RAN
     GE (THE OPERABLE LEVEL RANGE IS +/-1.24 DM), AND INTRODUCE A STEPCHAN
     GE THAT WILL CAUSE THE LEVEL TO RESPOND TOWARDS THE OPPOSITE EXTREME.
     "
195  INPUT "CONTINUE ? (Y/N) ";B$
196  IF  ASC (B$) = 78 THEN  STOP
200  PRINT "FOR EXAMPLE, IF YOUR LEVEL SETPOINT WERE + 1.1 DM, AND YOUR CL
     OSED LOOP VOLTAGE AFTER PROCESS LINED UP WERE 2.5 VDC, A GOOD GUESS F
     OR YOUR OPEN-LOOP VOLTAGE WOULD BE 2.6 VDC."
230  PRINT : PRINT
240  INPUT "ENTER LEVEL SETPOINT BETWEEN -/+ 1.24 DM , LSET = ";LSET
250  PRINT
260  PRINT "REMARK: STRICTLY SPEAKING, THE LOOP SHOULD BE OPEN BEFORE THE
     STEPCHANGE IS INTRODUCED. HOWEVER, IF SYSTEM IS LINED UP, WE CAN BEGI
     N FROM A CLOSED LOOP AND OPEN IT AT THE MOMENT WE INTRODUCE THE STEPC
     HANGE."
265  PRINT "   ALSO: DO NOT FORGET TO TOUCH THE KEYBOARD AND THE EVENT BUT
     TON ON THE CHART RECORDER AT THE SAME TIME."
```

```
266  PRINT
267  PRINT "   IF YOU WISH A LISTING OF THIS PROGRAM, TYPE <RUN 10000> AND
      HIT <RETURN>."
269  PRINT : PRINT
270  INPUT "ENTER OPEN-LOOP VOLTAGE : VOLTS = ";VOLTS
280 FLAG% = 0
285 I = 0
290  HGR
300  & SCROLLSET
310  & OUTLINE
320  & PLTFMT = 1
330  & CLRTIMER
335 I = I + 1
340 X =  PEEK ( - 16384)
350  IF X > 127 THEN FLAG% = 1
360  & AIN,(TV) = INVOLT%,(C#) = 0
370 L = 1.24 - INVOLT% / 1651
380  IF FLAG% = 1 THEN  GOTO 430
390 M = 2048 + 10 * (LSET - L) * ( - 1651)
400  IF M > 4095 THEN M = 4095
410  IF M < 0 THEN M = 0
420  GOTO 440
430 M = VOLTS * 819
440  & AOUT,(DV) = M,(C#) = 2
450  & NXTPLT = (4095 - INVOLT%) / 32
455  PRINT
460  PRINT  TAB( 01)"L(DM)=";L; TAB( 22)"*VOLTS=";M / 819
480  PRINT  TAB( 01)"TIME = ";I" SECONDS"
580  & LOOK FOR TIMERIN,(TV) = Q,(TH) = 1000: GOTO 330
```

```
10  TEXT
20  PRINT "-----------------------"
25  PRINT "PROCESS IDENTIFICATION:"
30  PRINT "G12 RESPONSE CURVE."
40  PRINT "========================"
50  PRINT "REMARKS:"
60  PRINT "   1) THIS PROGRAM ALLOWS YOU TO OBTAIN THE LEVEL RESPONSE TO A
      STEPCHANGE IN THE VOLTAGE SIGNAL TO VALVE B. AGAIN, YOU WILL NEED TH
     E CHART RECORDER TO BE ON-LINE."
70  PRINT "   2) SET THE CHART RECORDER SPEED AT 10 MM/MIN. THE HOT WATER
      INLET FLOWRATE SHOULD BE SET AT 15% ROTAMETER READING. THE STEAM DOES
       NOT NEED TO BE ON, SINCE WE WILL NOT READ TEMPERATURES."
80  PRINT "   3) YOU WILL BE ASKED TO INPUT TWO VOLTAGES, V1 AND V2. THE M
      AGNITUDE OF YOUR STEPCHANGE WILL BE THE DIFFERENCE BETWEEN V1 AND V2
      . USE SMALL STEPCHANGES, WHERE ABS(V1-V2) <= 0.5 VOLTS."
90  PRINT "OTHERWISE, TANK MAY RUN DRY OR OVERFLOW BEFORE NEW EQUILIBRIUM
     HAS BEEN REACHED."
100  INPUT "CONTINUE ? (Y/N)";A$
110  IF  ASC (A$) = 78 THEN  STOP
120  PRINT "   4) LEVEL SETPOINT SHOULD BE CLOSE TO EITHER EXTREME OF ITS
      RANGE (+/- 1.24 DM). INTRODUCE A STEPCHANGE SO THAT THE LEVEL RESPONS
      E BE TOWARDS THE OPPOSITE EXTREME OF THE RANGE."
130  PRINT "    FOR EXAMPLE, IF V1 = 1 VDC AND V2 = 1.5 VDC, BEGIN WITH LSE
      T = -1.1 DM, SO THAT YOU ALLOW PLENTY OF SPACE FOR THE LEVEL TO RISE.
      "
140  PRINT "   5) AS WITH THE OTHER PROCESS IDENTIFICATION PROGRAMS (G11 A
      ND G22), PRESS SIMULTANEOUSLY THE COMPUTER KEYBOARD AND THE EVENT BUT
      TON IN THE CHART RECORDER TO INITIATE THE STEPCHANGE."
150  PRINT "   6) IF YOU WISH A LISTING OF THIS PROGRAM, TYPE (RUN 10000)
      AND HIT (RETURN)."
160  PRINT : PRINT
170  INPUT "ENTER V1 IN VOLTS, V1 = ";V1
175  PRINT
180  INPUT "ENTER V2 IN VOLTS, V2 = ";V2
185  PRINT
186  INPUT "ENTER LEVEL SETPOINT LSET = ";LSET
190 FLAG% = 0
200 I = 0
210  HGR
220  & SCROLLSET
230  & OUTLINE
240  & PLTFMT = 6
250  & CLRTIMER
260 I = I + 1
270 X =  PEEK ( - 16384)
280  IF X > 127 THEN FLAG% = 1
290  & AIN,(TV) = INVOLT%,(C#) = 0
300 L = 1.24 - INVOLT% / 1651
310  IF FLAG% = 1 THEN  GOTO 350
320 M = 2048 + 10 * (LSET - L) * ( - 1651)
330  IF M > 4095 THEN M = 4095
```

```
340  IF M < 0 THEN M = 0
350  & AOUT,(DV) = M,(C#) = 2
360  IF FLAG% = 1 THEN  GOTO 390
370  & AOUT,(DV) = V1 * 819,(C#) = 1
380  GOTO 400
390  & AOUT,(DV) = V2 * 819,(C#) = 1
400  & NXTPLT = (4095 - INVOLT%) / 32
405  IF FLAG% = 1 THEN  GOTO 430
407  PRINT
410  PRINT  TAB( 01)"L(DM)=";L; TAB( 22)"**VOLTS="V1
420  GOTO 440
430  PRINT  TAB( 01)"L(DM)=";L; TAB( 22)"**VOLTS="V2
440  PRINT "TIME = ";I" SECONDS"
450  & LOOK FOR TIMERIN,(TV) = 0,(TH) = 1000: GOTO 250
```

```
10  TEXT
20  PRINT "========================="
30  PRINT "PROCESS IDENTIFICATION:"
40  PRINT "G22 RESPONSE CURVE.      "
50  PRINT "-------------------------"
60. PRINT "REMARKS:"
70  PRINT "   1) THIS PROGRAM ALLOWS YOU TO OBTAIN THE TEMPERATURE RESPONS
    E CURVE TO A VOLTAGE STEPCHANGE TO VALVE B. VALVE B CONTROLS THE COLD
     WATER INLET FLOWRATE."
80  PRINT "   2) THE RESPONSE CURVE WILL BE RECORDED IN THE CHART RECORDER
    , WHICH WILL HAVE TO BE ON-LINE. SET CHART RECORDER SPEED AT 10 MM/MI
    NUTE."
90  PRINT "   3) THE STEPCHANGE IS INITIATED BY SIMULTANEOUSLY PRESSING TH
    E COMPUTER KEYBOARD AND THE EVENT BUTTON IN THE CHART RECORDER."
100 PRINT "   4) YOU WILL BE ASKED TO INPUT TWO VOLTAGES V1 AND V2. THE M
    AGNITUDE OF YOUR STEPCHANGE WILL BE THE DIFFERENCE BETWEEN V1 AND V2.
    "
110 PRINT
120 INPUT "CONTINUE ? <Y/N)";A$
130 IF  ASC (A$) = 78 THEN  STOP
140 PRINT "   4) WAIT FOR PROCESS TO LINE UP WITH V1. INTRODUCE STEPCHANG
    E BY PRESSING THE KEYBOARD AND THE EVENT BUTTON SIMULTANEOUSLY."
150 PRINT "   5) THE LEVEL LOOP WILL BE CLOSED AND TIGHTLY TUNED WHILE YO
    U OBTAIN THE TEMPERATURE RESPONSE."
160 PRINT "THIS IS NOT A REASON FOR CONCERN, SINCE IT WAS SHOWN AT THE PR
    OCESS MODELLING STAGE THAT THE LEVEL LOOP DOES NOT AFFECT THE TEMPERA
    TURE LOOP."
165 PRINT "   6) SET THE MICROPROCESSOR SETPOINT AT 60 C, AND THE HOT WAT
    ER AT 15% ROTAMETER READING. V1 AND V2 SHOULD NOT EXCEED 3.35 VDC."
167 PRINT "   7) IF YOU WISH A LISTING OF THIS PROGRAM, TYPE <RUN 10000>
    AND HIT <RETURN>."
170 PRINT : PRINT
180 LSET = 0
190 ST = 1
200  INPUT "ENTER V1 IN VOLTS, V1 = ";V1
210  PRINT
220  INPUT "ENTER V2 IN VOLTS, V2 = ";V2
230 A = .011:B = 25.962
240 C =  - .755:D = .035
250 FLAG% = 0
260  HGR
270  & SCROLLSET
280  & OUTLINE
290  & PLTFMT = 1,2
300 I = 0
310  & CLRTIMER
320 I = I + 1
330 X =  PEEK ( - 16384)
340  IF X > 127 THEN FLAG% = 1
```

```
350  & AIN,(TV) = INVOLT%,(C#) = 0
360 L = 1.24 - INVOLT% / 1651
370 M = 2048 + 10 * (LSET - L) * ( - 1651)
380  IF M > 4095 THEN M = 4095
390  IF M < 0 THEN M = 0
400  & AOUT,(DV) = M,(C#) = 2
410  IF FLAG% = 1 THEN  GOTO 440
420  & AOUT,(DV) = V1 * 819,(C#) = 1
430  GOTO 450
440  & AOUT,(DV) = V2 * 819,(C#) = 1
450  & NXTPLT = (4095 - INVOLT%) / 32
460  & WRDEV,(DV) = 2,(W#) = 2,(D#) = 0
470  & AIN,(D#) = 0,(TV) = MV,(C#) = 0
480 V = 5000 * (MV - 2048) / (248 * 2048)
490 T = A + V * (B + V * (C + D * V))
500  & NXTPLT =  - 25.6 + 2.56 * T
505  PRINT
510  PRINT  TAB( 01)"L(DM)=";L; TAB( 22)"**T(C)=";T
520  PRINT "TIME = ";I" SECONDS"
530  & LOOK FOR TIMERIN,(TV) = Q,(TH) = 1000: GOTO 310
```

Appendix K

SAMPLE 2-POINT METHOD MODEL FITTING

Figure K.1 shows an experimental level reaction curve to a voltage stepchange V_a. This curve can be described by the model

$$g_{11}(s) = \frac{L(s)}{V_a(s)} = \frac{K_p\, e^{-Ds}}{\tau_p s + 1} \quad (dm/VDC)$$

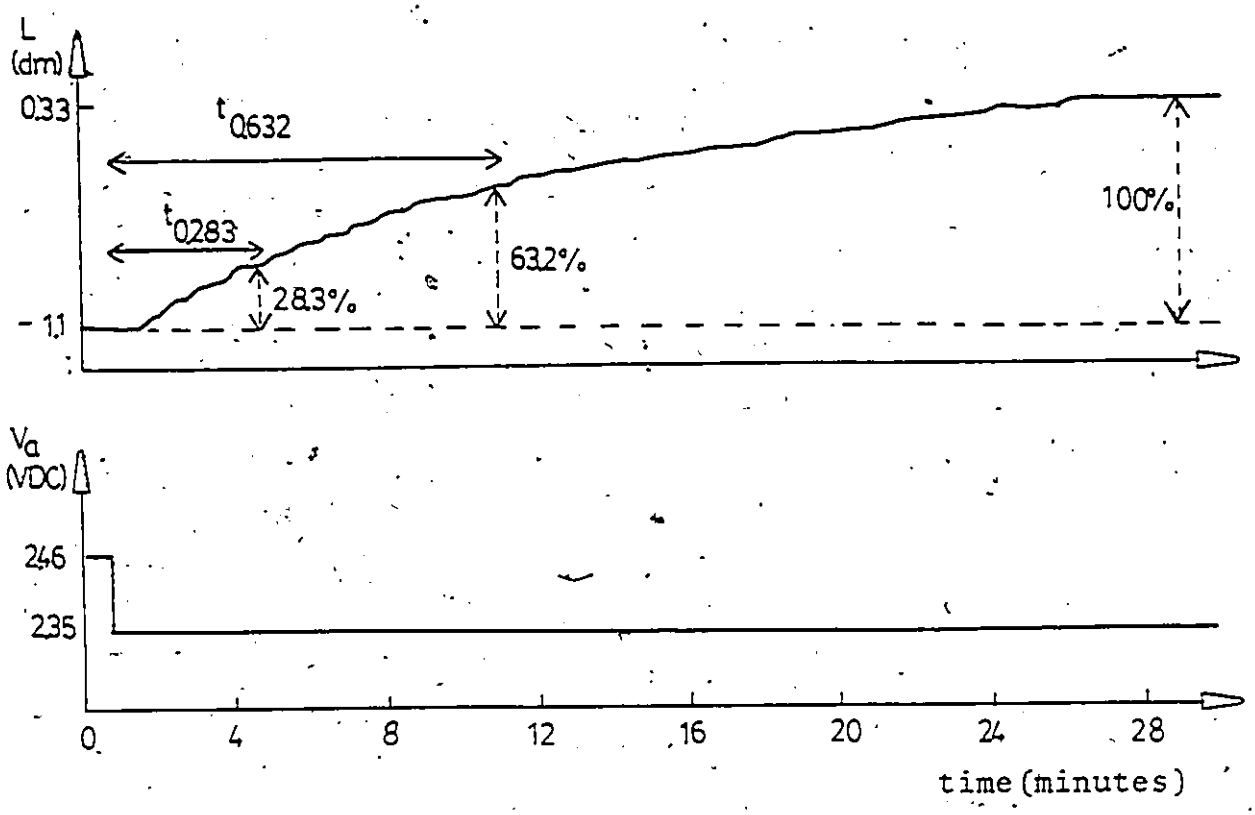


Figure K.1: Sample Level Reaction Curve.

where $K_p = \Delta L(\infty)/\Delta V_a$ (dm/VDC)

$$t_{0.283} = \tau_p/3 + D \qquad (4.3)$$

$$t_{0.632} = \tau_p + D \qquad (4.4)$$

From Figure K.1, $t_{0.283} = 3.8$ minutes and

$t_{0.632} = 10.3$ minutes.

Therefore $K_p = (+0.33 - (-1.10))/(2.35-2.46)$ (dm/VDC)

$K_p = -13.00$ (dm/VDC)

Solving simultaneously equations (4.3) and (4.4) yields

$\tau_p = 9.75$ minutes and

$D = 0.55$ minutes.

The final fitted model is

$$g_{11}(s) = \frac{-13.00\, e^{-0.55\, s}}{9.75\, s + 1} \quad \text{(dm/VDC)}$$

Appendix L

LEVEL AND TEMPERATURE CONTROLLER TUNING

1) LEVEL

In order to determine the ultimate gain and period for the level loop, the temperature loop was open. The level controller was put in effect with a position P-only algorithm. Setpoint stepchanges were introduced for progressively higher controller gains until the level began oscillating at a constant amplitude. This was the ultimate gain, and the period of oscillation, the ultimate period. Figure L.1 (a) shows some level responses to level setpoint stepchanges, with the ultimate gain Ku = 30. One can notice that, because of process nonlinearities, the ultimate gain is higher for higher levels. We determined the ultimate gain at a low level, the least favorable condition. From Figure L.1 (a) the ultimate period of oscillation was found to be Pu = 12 seconds. Since level control is used mainly for inventory purposes, small offsets do not pose a problem and P-only control can be used. Figure L.1 (b) shows some level responses to level setpoint stepchanges using P-only control with Ziegler-Nichols setting:

$K_c = Ku/2 = 30/2 = 15$. One can notice that some overshoot is present. An offset of approximately 8% can also be seen. Control performance is satisfactory.

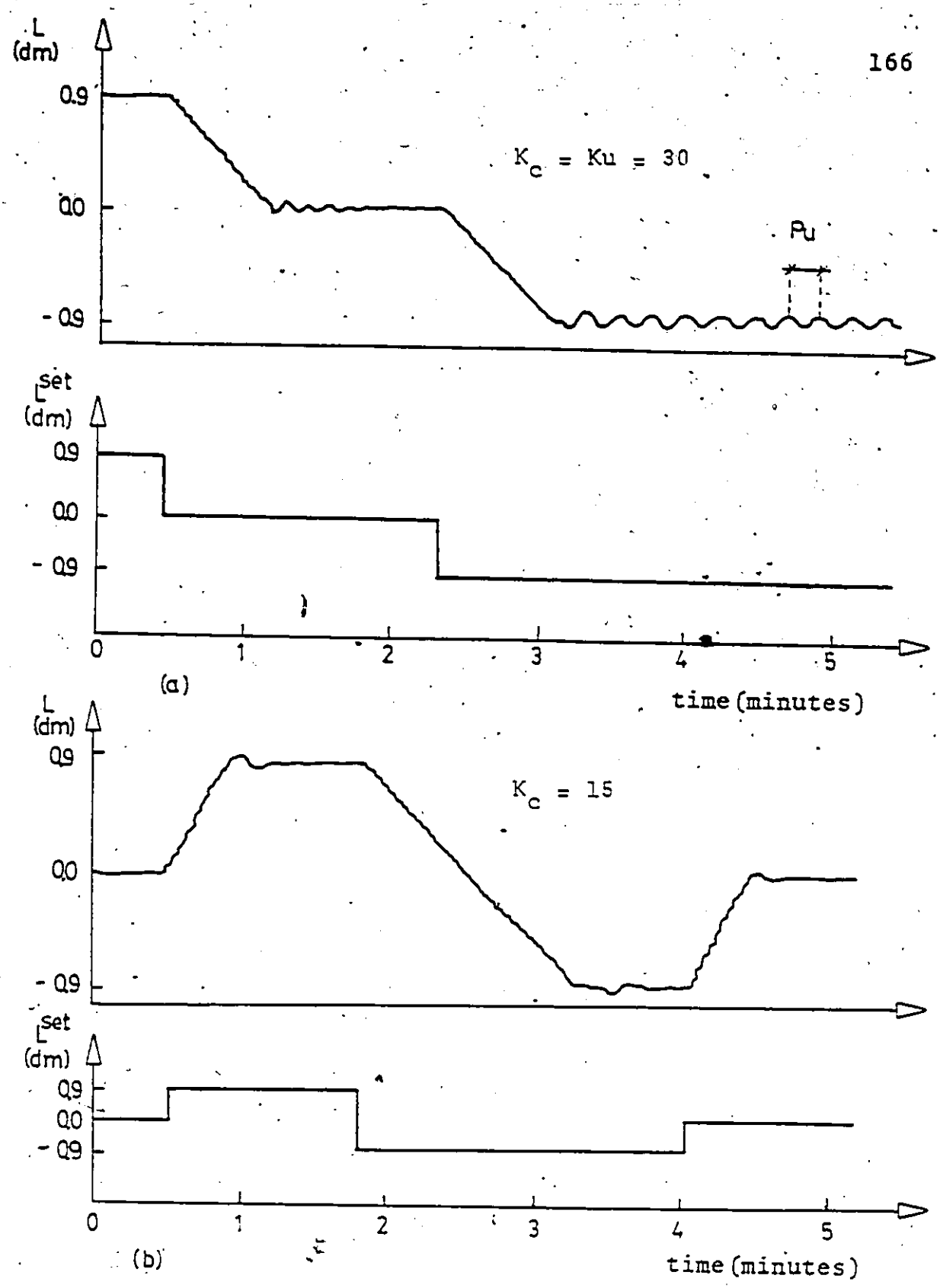


Figure L.1: Level Controller Tuning: Determining the ultimate gain Ku (a) and controller performance under Ziegler-Nichols setting (b). F_h = 14.1 L/minute and F_c = 0 L/minute.

2) TEMPERATURE:

In order to determine the ultimate values for the temperature loop, the level loop (P-only mode) was kept tightly tuned with K_c = 30, to minimize coupling. The level loop was kept closed because there is no coupling in the direction level to temperature.

The remaining of the procedure was identical to the one for the level loop described above. The ultimate gain was found to be Ku = 1.75 and the ultimate period Pu = 45 seconds, as can be seen in Figure L.2 (a).

Offsets are not likely to be tolerated in temperature loops. Therefore P-only control cannot be used. On the other hand, PID control may not provide adequate control, because the derivative term amplifies the noise usually present in the thermocouple signal. Thus we decided to use PI control in the temperature loop. The Ziegler-Nichols settings are:

K_c = Ku/2.2 = 0.8 and

τ_I = Pu/1.2 = 38 seconds.

Temperature responses to temperature setpoint stepchanges, the PI controller tuned using the settings above, are shown in Figure L.2 (b). Despite the overshoot mainly due to the integral action, and the noise in the signal, control performance is satisfactory and offset is zero.

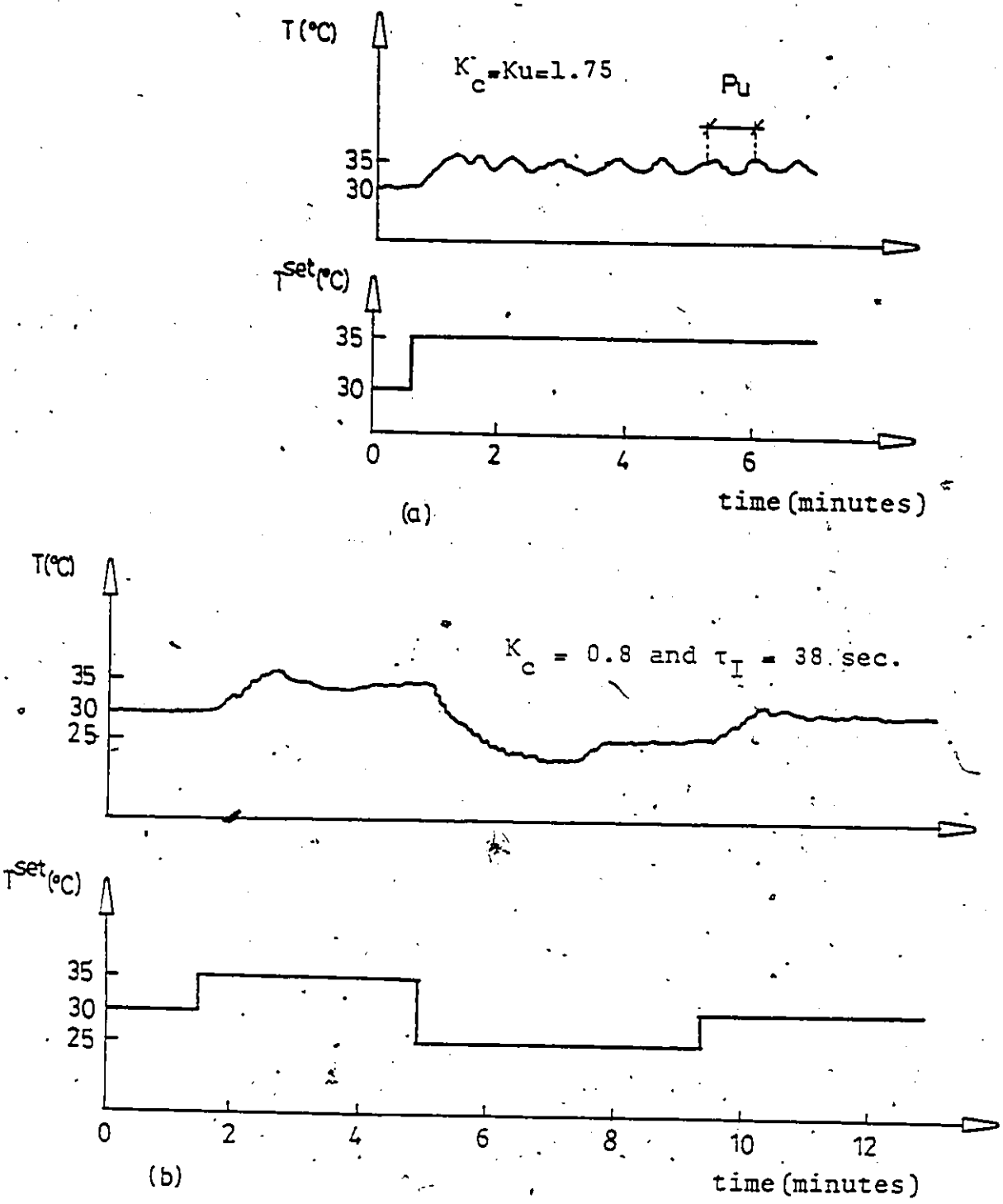


Figure L.2: Temperature Controller Tuning: Determining the ultimate gain Ku (a) and controller performance under Ziegler Nichols settings (b). Level loop closed with K_c = 30.
L = 9.48 dm, F_h = 7.6 L/min., T_h = 60°C.

Appendix M

STATISTICAL ANALYSIS OF THE DECOUPLER PERFORMANCE.

In order to assess any improvement in control performance achieved with decoupling, we introduced 4 replicated temperature setpoint stepchanges, respectively with and without decoupling. The response recorded was the Integral of the Squared Error (ISE) of the level response, computed during 3 minutes. Having obtained 2 sets of data with 4 datapoints each, we compared both populations.

Level controller Tuning:

P-only control with $K_c = 1.0$.

Temperature controller tuning:

PI control with $K_c = 0.8$. and $\tau_I = 38$ seconds.

Dataset #1 (with decoupling)	Dataset #2 (without decoupling)
$n_1 = 4$	$n_2 = 4$
$\nu_1 = 3$	$\nu_2 = 3$
$\bar{y}_1 = 1.59$	$\bar{y}_2 = 4.15$
$s_1^2 = 0.02$	$s_2^2 = 0.11$

A 95% Confidence Interval for σ_1^2 / σ_2^2 is given by:

$$\frac{s_1^2}{s_2^2 F_{\upsilon_1,\upsilon_2,\alpha/2}} \text{ to } \frac{s_1^2}{s_2^2 F_{\upsilon_1,\upsilon_2,1-\alpha/2}}$$

Substituting numerical values:

$$\frac{0.02}{0.11\ (9.28)} \text{ to } \frac{0.02}{0.11\ (1/9.28)} = (0.02 \text{ to } 1.69)$$

Therefore $\sigma_1^2 = \sigma_2^2$ is a reasonable assumption.

Pooling the variances:

$$s_p^2 = \frac{0.02(3) + 0.11(3)}{3 + 3} = 0.07$$

A 95% C.I. for $\mu_1 - \mu_2$ is given by:

$$(\bar{y}_1 - \bar{y}_2) +/- t_{\upsilon,\alpha/2} S_p (1/n_1 + 1/n_2)^{1/2}$$

Substituting numerical values:

$$(1.59 - 4.15) +/- 2.447\ (0.26)\ (1/4 + 1/4)^{1/2} =$$

$$(-3.01 \text{ to } -2.11)$$

Since the interval above does not include zero, the ISE with decoupling is significantly smaller than the ISE without decoupling, based on a 95% probability level.