DIGITAL COMPUTER PROCESS CONTROL

By

Alan Kwok-Keung CHING

A thesis submitted to The School of Graduate Studies

of

University of Ottawa, Ottawa, Ontario, Canada,

in partial fulfillment of the requirement for the

degree of

Master of Applied Science

in the

DEPARTMENT OF CHEMICAL ENGINEERING

October, 1977

ABSTRACT

A shell and tube heat exchanger was interfaced with a NOVA digital computer to allow automatic data logging and direct digital computer control. A real time computer operating system was developed for this special application. The Characteristics of a non-linear Multiplicative control algorithm were studied using this system.

As a result of the study, a feed back loop was added to the Multiplicative control algorithm to compensate the offset caused by the non-linear characteristics of the heat exchanger. The stability region of the Multiplicative control algorithm was mapped and showed high and low stability limits for Multiplicative gain settings.

ACKNOWLEDGEMENTS

The author of this thesis wishes to thank Dr. F. D. F. Talbot, the supervisor of this work, for his patient guidance throughout the course of this work.

The author would also like to express his gratitude to his parents whose enthusiastic support and continuous encouragement have made this work possible.

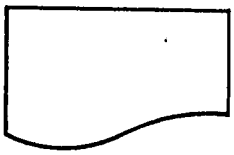
TABLE OF CONTENTS

LIST OF SYMBOLS

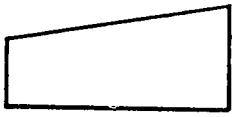
The following symbols appear in the software program logic flow charts.

Start or End of a logic path

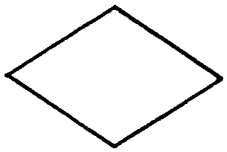
A function performed by the computer



A message printed on the teletype writer



An operator's entry through the teletype writer



A decision made by the computer

A computer output through the paper tape punch

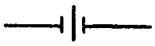
A logic path connection

The Following symbols appear in the figures

Steam trap

Hand valve

Pneumatic control valve

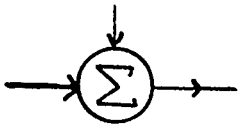


Critical flow orifice

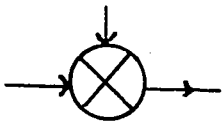
Signal amplifier

Operational amplifier

Resistor

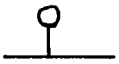


Summation of two input signals

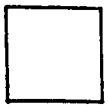


Multiplication of two input signals

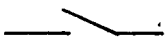
Water flow rate indicator



Pressure indicator



Functional Block



Digital switch

LIST OF FIGURES

NOMENCLATURE

A	Multiplicative controller gain constant (1/PSI)
AC	Accumulators of the NOVA digital computer
ADCV	Analog to digital converter
ASCII	American Standard Committee for Information Interchange
$b, b_1 - b_{n+1}$	Multiplicative feed back value (°F)
c	Process output (°F)
CFO	Critical flow orifice
CPU	Central process unit
d	Differential
DDC	Direct digital control
DGC	Data General Corporation
DACV	Digital to analog converter
e	Error for the control algorithm (°F), or Natural log base
E_A	Error for the Modified Multiplicative control (°F) algorithm
exp	Exponential
FI	Flow indicator
G	Process gain (°F/PSI)
$G_1 - G_{n+1}$	Multiplicative control system functions

G_p	The simplified heat exchanger model
IPC	Current to pressure converter
I_o	Current output (mA)
I/O	Input-output (of the digital computer)
K	Proportional gain (PSI/°F)
K_I	Gain constant for the integral control (1/PSI.°F)
K_L	Lower limit for the proportional gain (PSI/°F)
L	Load
M	Intermediate variable (PSI)
PI	Pressure indicator, or
	Proportional-Integral controller
PTP	Paper tape punch
PTR	Paper tape reader
r	Set point of a controller (°F)
R_1 - R_3	Resistors (ohm)
RTC	Real time clock
S	Laplace transform operator
t	Time, function of
T_c	Multiplicative control time constant (sec)
T_I	Integral time constant (sec)
T_p	Time constant of the heat exchanger system (sec)
T_s	Sampling time (sec)
T_1,T_2	Process time constant (sec)
T/C	Thermocouple

TTI	Teletype writer input (to the computer)
TTO	Teletype writer output (from the computer)
u_1	Load for the heat exchanger (PSI)
u_2	Constant for thermocouple conversion function (°F)
V_I	Voltage input (volt)
W_c	Corner frequency (1/sec)
X	Manipulated process variable (PSI)
$y, y_1 - y_{n+1}$	Output of the Multiplicative system functions
$Z_1 - Z_{n+1}$	Variables of Integration
Z	Z-transform operator
$\mathcal{Z}$	Z-transformation, function of

GLOSSARY

This is a collection of some computer terms used in the thesis. Most of the definitions given were extracted from COMPUTER DICTIONARY AND HANDBOOK, compiled by Charles J. Sippl and Charles P. Sippl and published by Howard W. Sams & Co. Inc. 1972.

ACCUMULATOR:	A part of the logic-arithmetic unit of a computer. It may be used for immediate storage, to form algebraic sums, or for other intermediate operations.
ADDRESS:	A label, name or number identifying a memory location where information is stored.
ADDRESS, RELOCATABLE:	Address coded in a special routine or program whose instructions are written in a special manner so that it can be loaded and executed in many areas of the memory. Relocatable programs allow for the highly flexible real time use of the main memory.
ADDRESS, RETURN:	The address provided for a return to a previous point in the usual sense. In particular, the address provided by a main program to a subroutine so that at the end of the subroutine execution, system control can be returned to the proper point in the main program.
BRANCH:	To depart from the normal sequence of executing of instructions in a computer program.
FLAG:	An indicator used to tell some later part of a program that some condition occurred earlier.
HARDWARE:	The mechanical, magnetic, electrical and electronic device of a computer system.
INTERRUPT:	A break in the normal flow of a system or a routine such that the flow can be resumed from that point at a later time. An interrupt is usually caused by a signal from an external source.

JUSTIFY:	To move a data item so that a particular part of the item assumes a particular position relative to some reference point in a storage medium; for instance, to adjust the print on a printed page so that the left, right or both margins are aligned; also to shift the item in a register to position specifically the most or least significant digit.
MEMORY:	Any device into which a unit of information can be copied, which will hold this information, and from which the information can be obtained at a later time. (The terms Memory and Storage are interchangeable).
NOISE:	Random variations of one or more characteristics of any entity such as voltage, current and data.
OCTAL NOTATION:	A number of one or more figures, representing a sum in which the quantity represented by each figure is based on a radix of eight. The figures used are 0,1,2,3,4,5,6 and 7.
PARITY:	As regards to computer operations, parity relates to the maintenance of a sameness of level or count. i.e. keeping the same number of binary one in a computer word to thus be able to perform a check based on an even or odd number for all words under examination.
PRIORITY:	The sequence in which various entries and tasks will be processed.
PROGRAM COUNTER:	A register in which the address of the current instruction is recorded. (synonymous with instruction counter).
REAL TIME:	Relating to the performance of computing during the specific time in which the related process, event, problem or communication is taking place, i.e. the computer must be fast enough, during the process of the happening of the event for the related process or result.

REAL TIME CLOCK:	The clock that develops readable digits or periodic signals for the computer to allow computation of elapsed time between events, and to initiate the performance of time initiated processing.
REAL TIME OPERATING SYSTEM:	A computer operating system which allows concurrent operations for data processing (computing) and physical processing in such a way that the result of the computing operations are available whenever needed by the physical processing operations, and visa versa.
ROUTINE:	A set of coded instructions arranged in proper sequence to direct the computer to perform a desired operation or series of operations.
SCAN:	To examine the analog input and update the computer data table periodically.
SCHEDULER:	A special system of the executive software. The scheduler controls the time that a program is to be executed.
SOFTWARE:	The internal programs or routines professionally prepared for a specific operation.
SYMBOLIC DEBUGGER:	A program allows to utilize symbolic commands to assist in the procedure of debugging.
SYSTEM INITIALIZATION:	To originate or establish the basic conditions or start up state. Such procedures might be used to set an initial value for the address of an operand, establish the initial control value for a loop, set all registers to a preset value prior to running etc.
WAITING MODE:	Tasks in "waiting mode" are voluntarily suspended until some operation is completed. Upon completion of the awaited operation, the waiting task is resumed.

INTRODUCTION

The early development of process control emphasized the dynamics and stability of individual control loops. However, today with more complex industrial processes, emphasis has shifted to the whole control system and its optimum operating condition. The control system of a modern process plant consists of a large number of individual control loops which must be manipulated in a consistent manner according to some overall control or optimization strategy. This can only be accomplished efficiently by using a computer (1).

After the commercialization of digital computers in the 1950's, the potential for digital computer applications in process control was quickly recognized. The development of the integrated circuit and the mini-computer in the 1960's has made the digital computer competitive with the conventional analog controller. More recently, the micro-processor and verious new digital interfaces have provided the user with more flexibility at considerably lower cost and will undoubtedly result in still more applications.

Process computer control systems are usually implemented in one of the following three levels (2):

1. Off-line Off-line computer systems are mainly used for data logging and estimating optimum conditions. Most of the time, a set of good data and analysis is not sufficient to justify a process computer. However data logging is essential for every control computer system.

2. On-line : On-line computer control or so called direct digital control (DDC) uses the computer in the place of the conventional analog controller. The output of the computer connects directly to the final control element. In the event of computer failure a backup system is required for critical control loops.

3. Supervisory control Supervisory control is also known as set-point control. The computer manipulates the set-point of the conventional analog control loop. The set-point value may be calculated according to process constraints, a desired optimization strategy or a special control algorithm.

Any digital computer process control system may consist of one or all of the above features. The present work was the developement of an on-line direct digital computer control system for a NOVA digital computer and the study of a non-linear Multiplicative servo control algorithm.

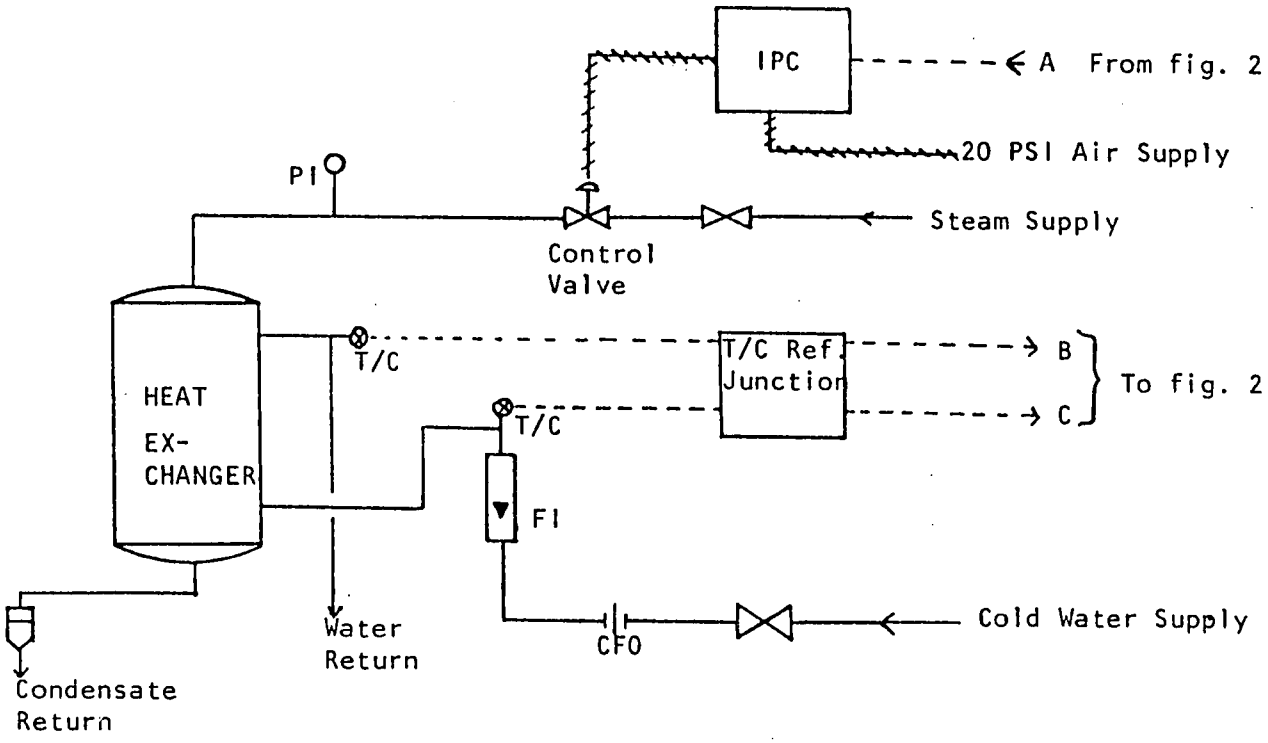
THE DIRECT DIGITAL CONTROL SYSTEM

The development of the direct digital computer control system included the hardware configuration as well as the software function development. The following sections will discuss these topics individually.

The Hardware :

The hardware configuration may be divided into two different groups; a shell and tube heat exchanger and a mini-computer with suitable peripheral equipment.

The heat exchanger, as shown in figure (1) was an American Standard Products Ltd. model BCF-302 one-pass shell and tube heat exchanger which consists of 52-0.25 inch O.D. tubes in a 3 1/8 inch I.D. shell that provided 7.34 square feet of heat transfer area. The entire 27 inch long heat exchanger and associated steam piping were insulated with asbestos. Cold water flowed into the shell side of the heat exchanger and was heated by steam in the tube side. The



PI	Pressure Indicator	———	Piping
T/C	Thermocouple	- - - - -	Electrical Connection
FI	Flow Indicator	~~~~~	Pneumatic Connection
CFO	Critical Flow Orifice		
IPC	Current to Air Pressure Converter		

Figure 1 The Heat Exchanger Mechanical Configuration

flow rate of water was regulated at 2 U.S. gallons per minute using a Dole Valve Company critical flow orifice. The steam flow rate was controlled using a Minneapolis-Honeywell Company model 1403 control valve with 1/2 inch plug and a model MP953 air-to-open valve operator. The pneumatic motor valve was operated by a Foxoboro model 69TA-1 current-to-air-pressure transducer. The temperatures of the inlet and outlet water were sensed by copper-constantin thermocouples. The thermocouples signals were compensated with an ACROMAG 325 zero degree centigrade electronic reference and then transmitted to the NOVA digital computer.

The digital computer and the peripheral configuration are shown in figure (2). A Data General Corporation (DGC) NOVA computer central processor unit (CPU) with 8192 (8K) 16 bit word core memory was the basic equipment. An ASR-33 teletype writer served as an operator's console and low speed data logging device. A DGC power failure monitor and auto-restart provided protection against AC line power failure. A DGC high speed paper tape reader and punch served the need for software development and high speed data output. A DGC programmable real time clock (RTC) was used as the system timer. A DGC 4033D eight channel analog-to-digital converter (ADCV) and a DGC 4037B two channel digital-to-analog converter (DACV) handled the interface between the analog (the process

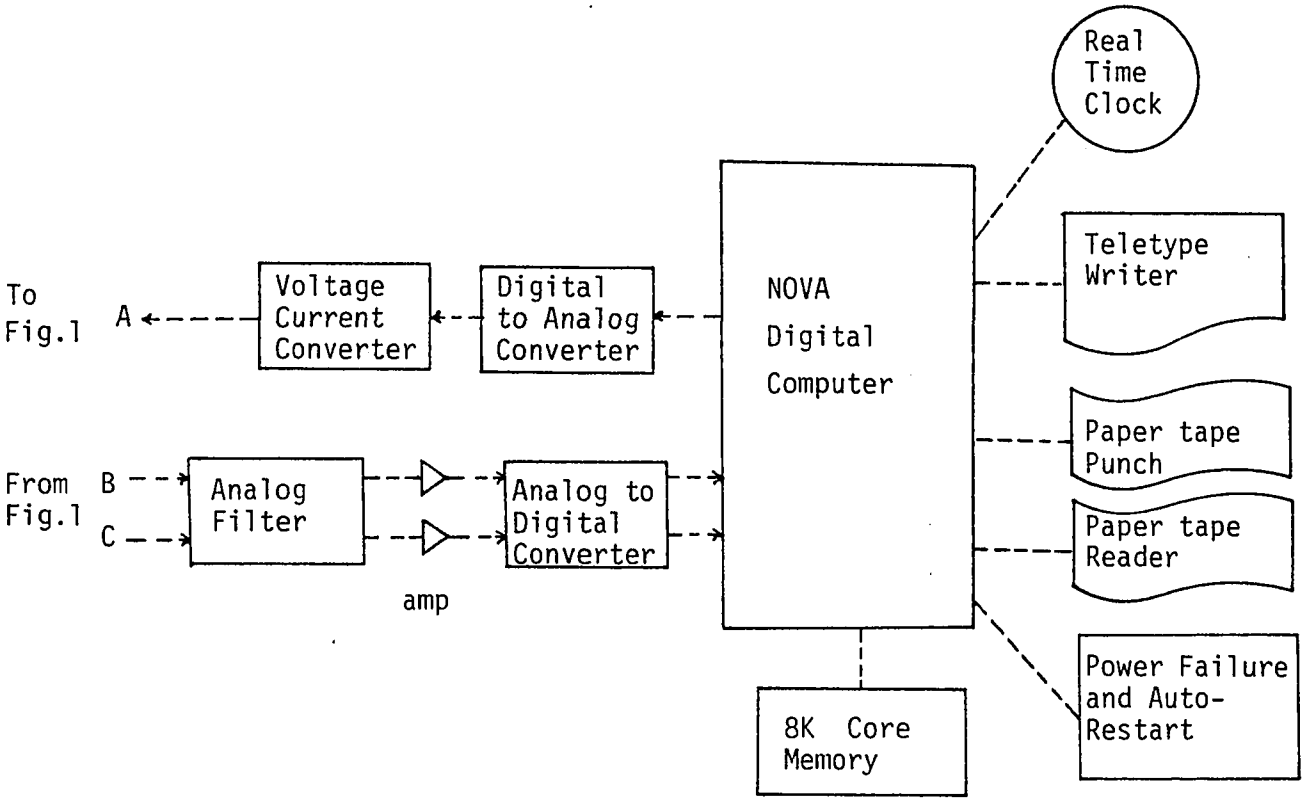
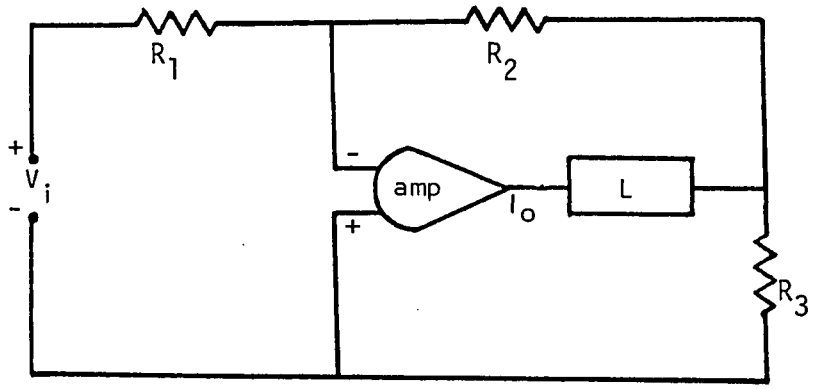


Figure 2 The Digital Computer Configuration

veriable sensor and driver) and the digital devices (the NOVA digital computer). The output of the digital-to-analog converter, which was ±10 VDC was converted to 4-20 mA current signal to drive the currnet-to-air-pressure converter. The voltage-to-current converter circuit is shown in figure (3).

During the debugging stage of the software development, the digital computer system was interfaced to a comdyna GP-6 analog computer by means of which the operation of the heat exchanger was simulated.



R_1 510K ohm resistor

R_2 220K ohm resistor

R_3 220K ohm resistor

amp Burr Brown 3164/25 operational amplifier

L Load (The current-to-air-pressure transducer. The input coil with 187 ohm DC resistance)

V_i Input voltage (0 - 10 VDC)

I_o Output current (0 - 20 ma DC with the above load)

Figure 3 circuit of the voltage to current converter

Software :

Software is defined as all the programs used in the computer to achieve a common goal. The software developed for this study is described as follows :

1) Program scheduling : A program was needed to schedule the execution of the programs according to their priority, timing or special request. In the present work, program scheduling was accomplished by using a hardware interrupt program (INTRP). This program also contained a special power failure interrupt routine (PWFSV) to protect the system from temporary line power failure.

2) Input-Output device handler : This was a group of subroutines that directly interfaced with the I/O devices in the system. Different I/O devices needed different handlers. Subroutines PTPO, OPT and IOXOS were written for this purpose.

3) Analog signal input scan and processing : This program was required to scan the process input periodically (according to the sampling time) and to convert the input signal (usually mv or ma)

into engineering units (°F etc.). The converted data was used for data logging and control calculation. Program READR was written for this purpose.

4) Control module : This was the control calculation for direct digital control. Subroutines PICNTL and MULPLTV were written for Proportional-Integral control and Multiplicative control algorithms respectively.

5) Analog output : A program was required to output the result from the computer to the process via the digital-to-analog converter. In the present work, this was built into the control module subroutines.

6) Data logging and operator communication : This group of programs permitted off-line data gathering and allowed the operator to communicate with the computer. Programs PTPSV, TTOSV and TTISV were written to provide these services.

7) System initialization : When the computer was first started up, all the data tables needed to be initialized and many system or control variables needed to be set up. Program HEEXEC provided these services.

The above seven functions were essential for the process control computer software system. The individual programs in the software system is discussed in detail in the following sections. Logic flow charts follow each program description and complete program listings are provided in appendix III.

All programs were written and tested individually under simulated operating conditions. Once all programs were working satisfactorily, they were linked together and the digital computer was interfaced with a Comdyna GP-6 analog computer on which a simple model of the heat exchanger was simulated. A common Proportional-Integral (PI) control algorithm (2,11) was used to test the software system. Once the system was operating satisfactorily, the Multiplicative control algorithm was tested. After all programs were proved, the system was interfaced with the shell and tube heat exchanger to study the stability of the Multiplicative servo control algorithm.

When the digital computer was first interfaced with the heat exchanger, a serious noise problem in the input signal was encountered which made the control action almost impossible.

The noise problem was solved by adding an analog filtering device and a digital filtering feature in the analog input scan program READR. The analog filtering device was simply a filtering capacitor. The digital filter was a first order filter which is discussed in detail in Appendix III.

The following sections will discuss each individual programs in detail.

1) Program scheduling and power failure protection programs :

a) Program INTRP

Program INTRP was the system interrupt handling program. It was turned on whenever there was a system hardware interrupt. There were four different types of system interrupts in this particular system, they are :

1. Power failure and auto-restart
2. Real time clock
3. Paper tape punch
4. Teletype writer input (operator's interrupt)

Program INTRP acknowledged the interrupt source and branched to service the highest priority interrupt. A higher priority device can interrupt the execution of a lower priority device service program but a lower priority device cannot interrupt a higher priority device service program.

A real time clock servicing routine was built into the interrupt handler. When the real time clock interrupted the

system, the sampling time counter (set up during system initialization) was counted down by one. If the counter was not zero, the sampling program would not be turned on and the lower priority program would be served. If the sampling counter was counted to zero, the analog scan and data conversion program READR was turned on and executed at priority 2.

The address of this program (INTRP) was stored in memory location 1. When a system interrupt occurred, the CPU generated instructions to interrupt the program being executed, store the interrupted program counter in memory location zero and then branch to program (INTRP).

A logic flow chart of this program is shown in figure (4).

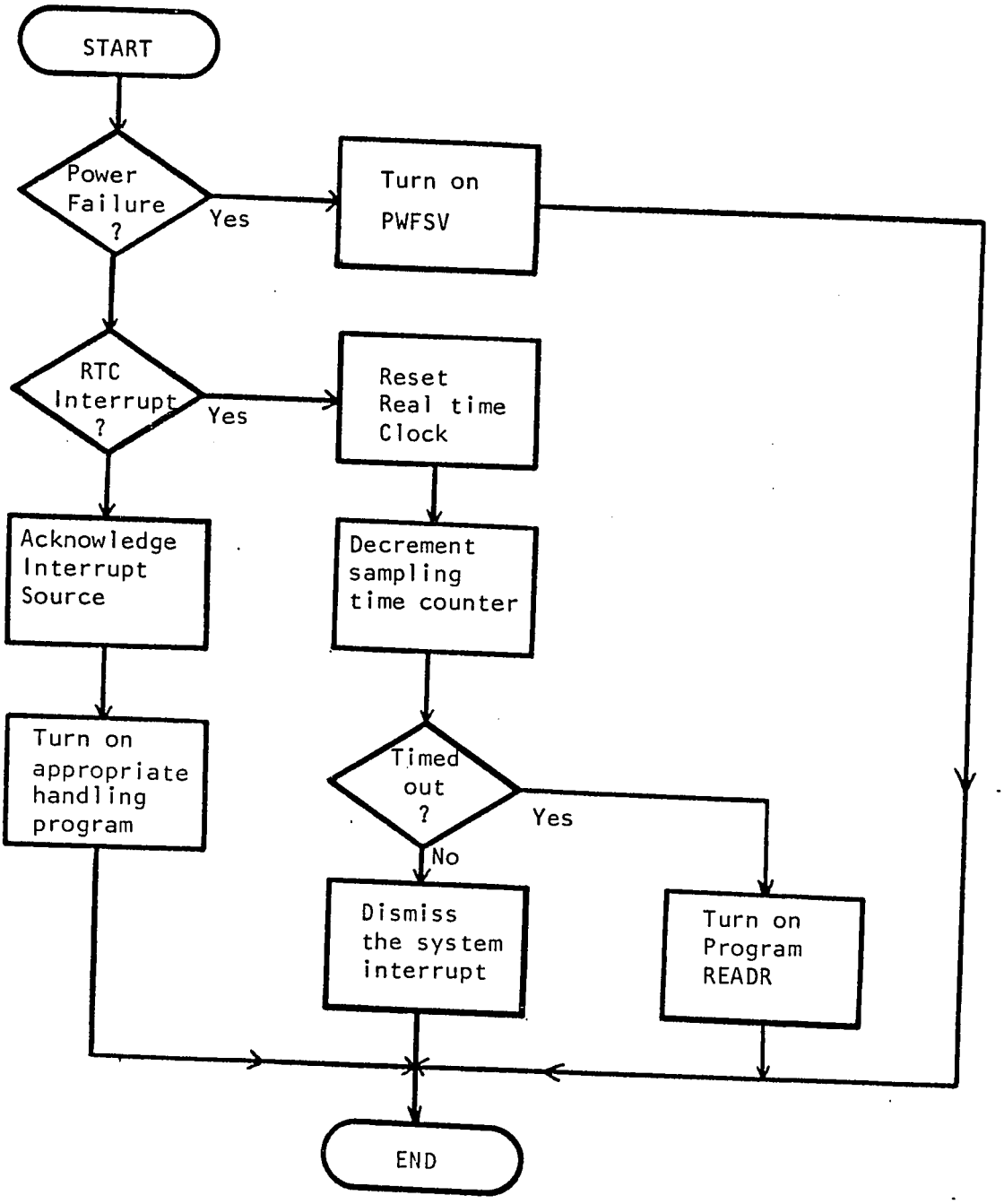


Figure 4 Program INTRP Logic Flow Chart

b) Program PWFSV :

This was the program used to protect the control system during short periods of power failure. The program was turned on whenever a power failure occurred which caused the hardware Power Failure and Auto-Restart to interrupt the system.

Program PWFSV consists of two parts, the power failure service and the power recover system restore service. There are only few milli-seconds to save the system if power should fail, thus this program was given priority one which had a response time of approximately 10 micro-seconds.

The power failure service part of this program saved all four accumulators, carry bit and the return address. The address of the automatic restart routine was set up in memory location zero. When the power resumed, the automatic restart hardware generated an instruction to branch through location zero to the automatic restart routine of this program. The routine then restored all four accumulators and the carry

bit. Normal operation was resumed after the system interrupt and the real time clock was turned on by the restart routine.

A short period power failure did not affect the process. If power loss was longer than the response time of the control valve, a process upset would occurr. For this reason, if the power failure period was long enough for the operator to notice, the control system was restarted using the initialization program.

A logic flow chart of this program is shown in figure (5).

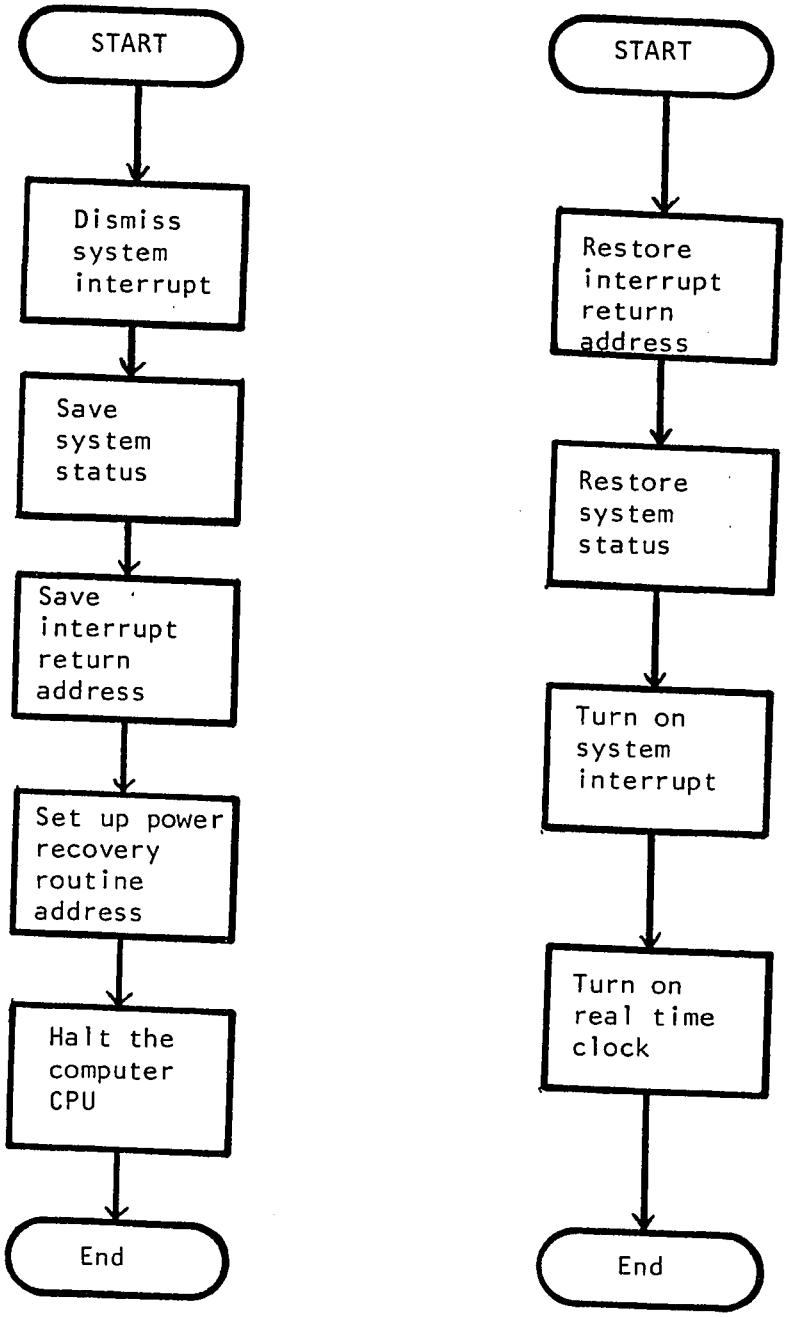


Figure 5 Power Failure and Recovery Handler

2) Input-output device handlers :

a) Subroutine PTPO

Calling sequence :

(output data in AC0, right justified)

JSR @42

(return)

This subroutine outputed one character through accumulator zero (AC0) to the paper tape punch. The right most eight bits were punched. No parity was generated or checked. If parity was required, the calling program was responsible for it. The output was made in the waiting mode of the paper tape punch.

For the convenience of the calling program, the address of this subroutine was stored in memory location 42_8 (octal notation). A logic flow chart of this subroutine is shown in figure (6).

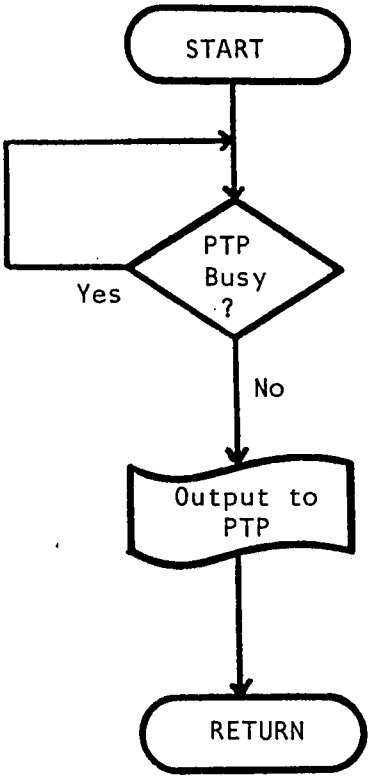


Figure 6 Subroutine PTPO Logic Flow Chart

b) Subroutine OPT

Calling sequence :

(output data in AC0, right justified)

JSR @41

(return)

This subroutine outputed one character through accumulator zero (AC0) to the teletype writer. The right most eight bits were printed. No parity was generated or checked. If parity was required, the calling program was responsible for it. The output was made in the waiting mode of the teletype writer.

For the convenience of the calling program, the address of this subroutine was stored in memory location 41_8 (octal notation). A logic flow chart of this subroutine is shown in figure (7).

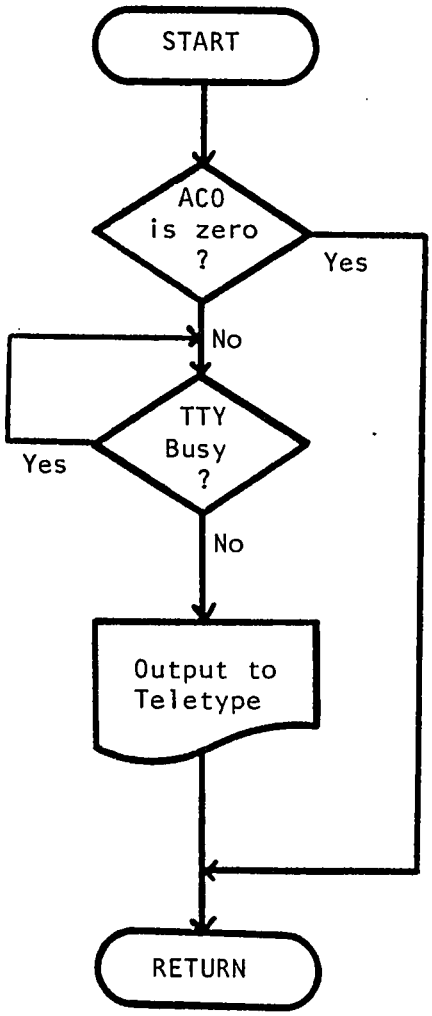


Figure 7 Subroutine OPT Logic Flow Chart

c) Subroutine IPT

Calling sequence :

JSR @40

(return with input character in AC0, right justified)

This subroutine inputed one ASCII character from the teletype writer. The parity bit was masked out before return to the calling program. The input was made in the waiting mode of the teletype writer.

For the convenience of the calling program, the address of this subroutine was stored in memory location 40_8(octal notation). A logic flow chart of this subroutine is shown in figure (8).

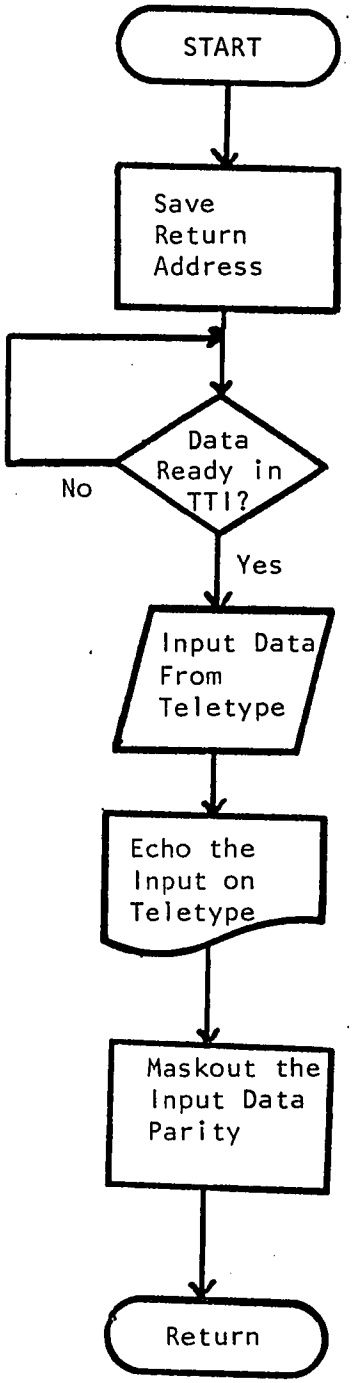


Figure 8 Subroutine IPT Logic Flow Chart

d) Subroutine IOXOS

Calling sequence :

(store address of the message in AC2)

JSR @4

(return)

This subroutine outputed a string of ASCII character whose starting address was stored in accumulator two (AC2). The ending character of the ASCII string must be the same as specified in DGC SOURCE-EDITOR program (an ASCII null). This subroutine called subroutine OPT to output the ASCII characters to the teletype writer.

For the convenience of the calling program, the address of this subroutine was stored in memory location 4. A logic flow chart of this subroutine is shown in figure (9).

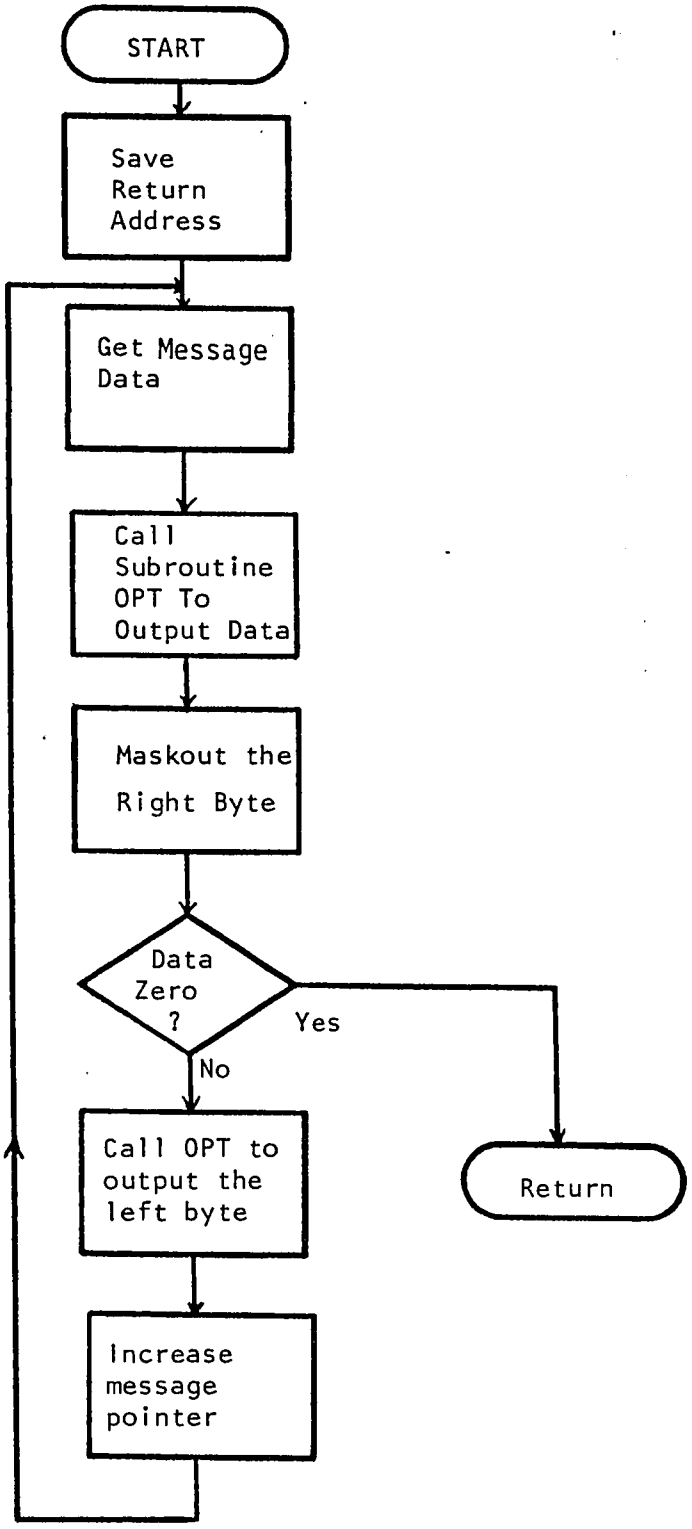


Figure 9 Subroutine IOXOS Logic Flow Chart

3) Analog signal input scan and processing :

Program READR

Program READR was the analog input program. The program was executed whenever the sampling timer was due.

This program read the input form both channels of the analog-to-digital converter, applied a digital filter to the input signals, converted the signals into engineering unit (°F) and stored them in the data table. If the control switch was on, the proper control algorithm subroutine would be turned on. If the control switch was off, the control subroutine would not be turned on and program READR would exit.

There were two control algorithm subroutines written for this system, the Multiplicative control and the PI control algorithms. Only one of the two subroutines were executed at one time. The choice of control algorithm was determined during system loading. The address of the control subroutine is stored at relocatable address CNALM.

A logic flow chart of this program is shown in figure (10).

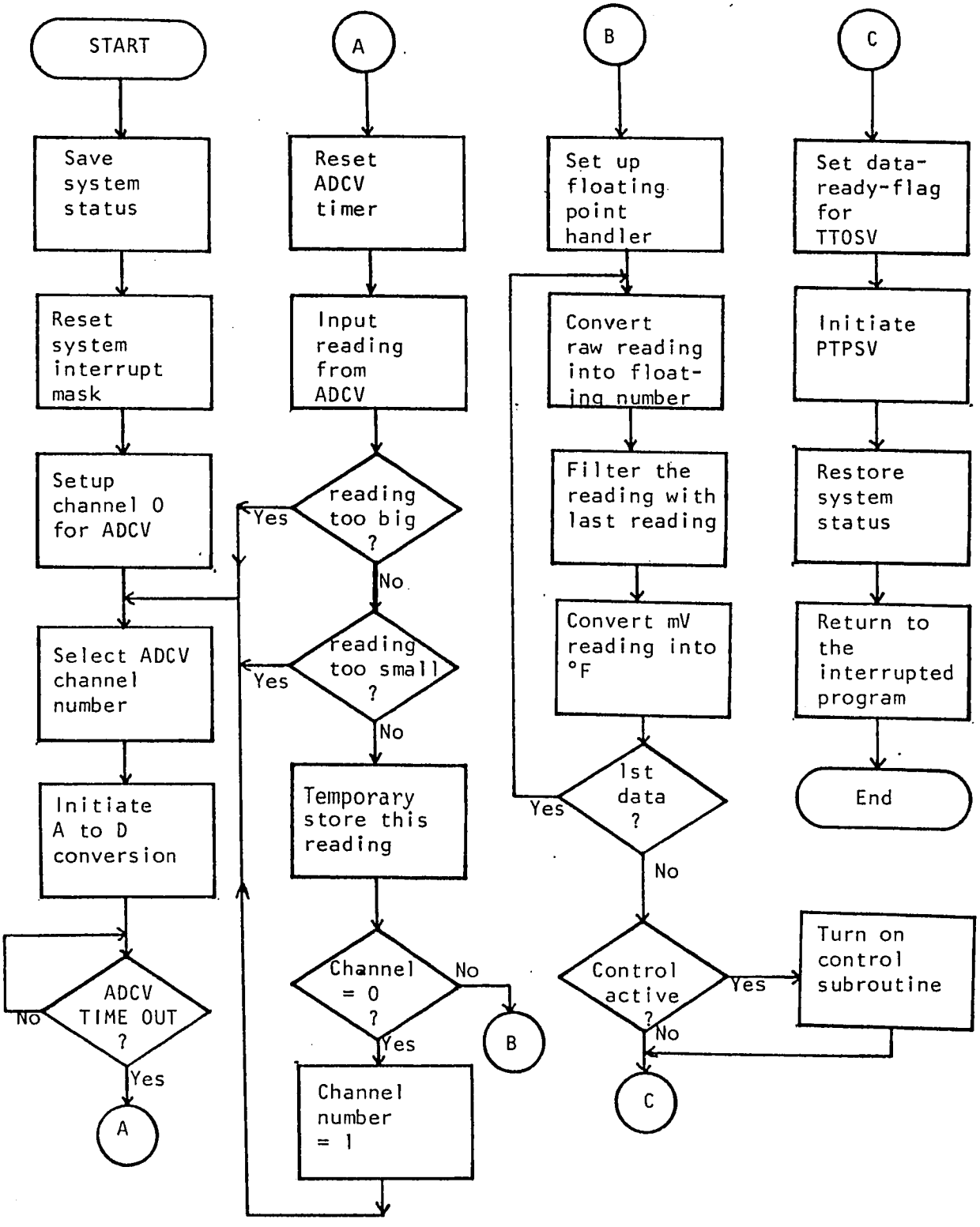


Figure 10 Program READR Logic Flow Chart

4) Control programs :

a) Subroutine ALPAR

This subroutine was called by programs HEEXEC and TTISV to request the operator to enter the parameters for the Multiplicative control algorithm. The parameters for the Multiplicative control algorithm are controller time constant (T_c), controller gain (K), and process gain (G). A logic flow chart of this subroutine is shown in figure (11).

b) Subroutine MULPLTV

This subroutine was the Multiplicative control algorithm for servo control. The functions of this subroutine might be summarized as follows :

1. Based on the input from the process and the set-point, calculates the theoretical output signal in PSI for the control valve using the Multiplicative control algorithm.

2. The above calculated value is limited by a maximum and mininum value to prevent control system wind up.
3. The value is converted to an integer value for the digital-to-analog converter to give an appropriate air pressure at the control valve.
4. Output the integer value to the digital-to-analog converter.
5. Return to the calling program.

A logic flow chart of the Multiplicative control algorithm is shown in figure (12).

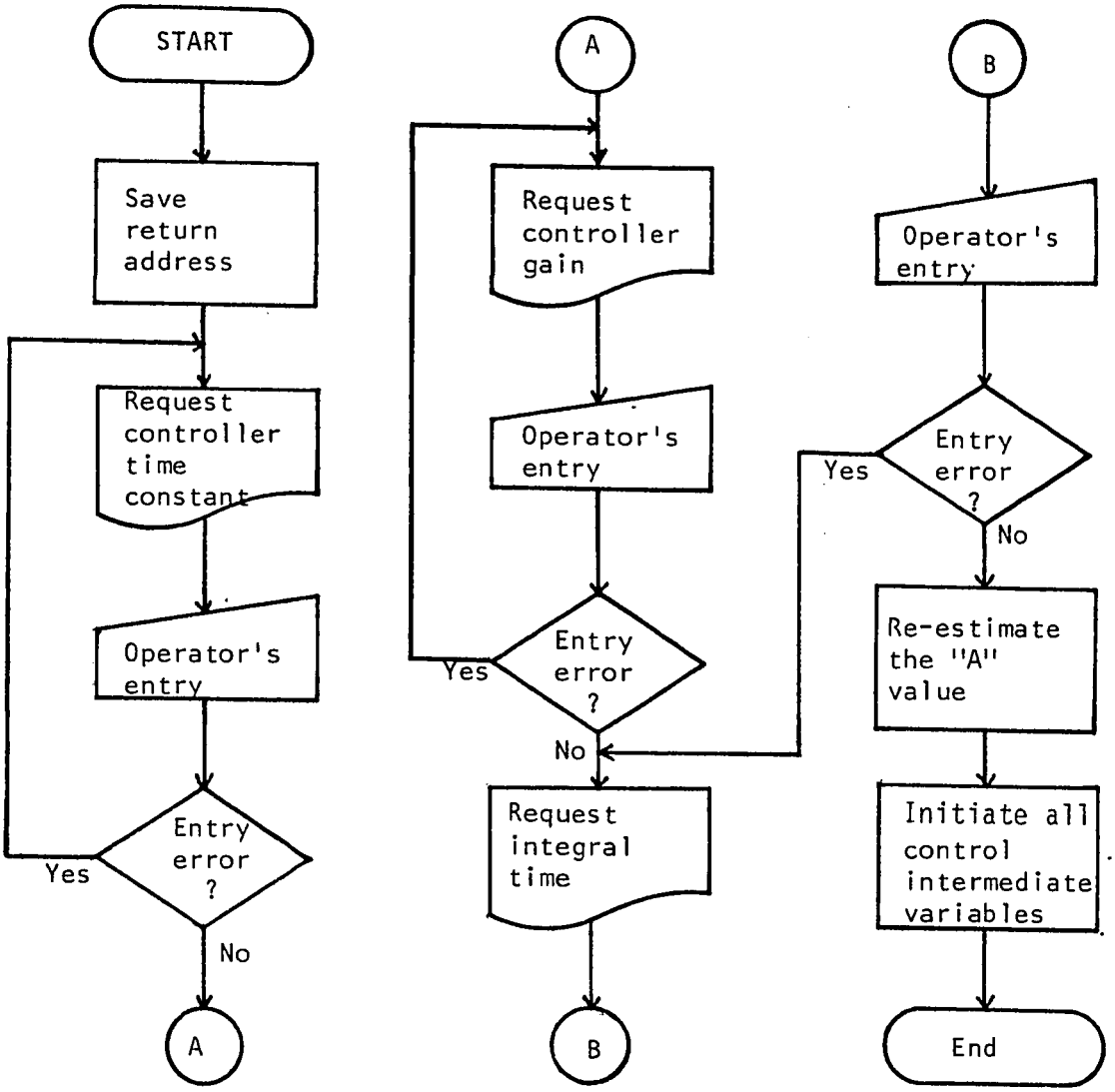


Figure 11 Subroutine ALPAR for Multiplicative Control

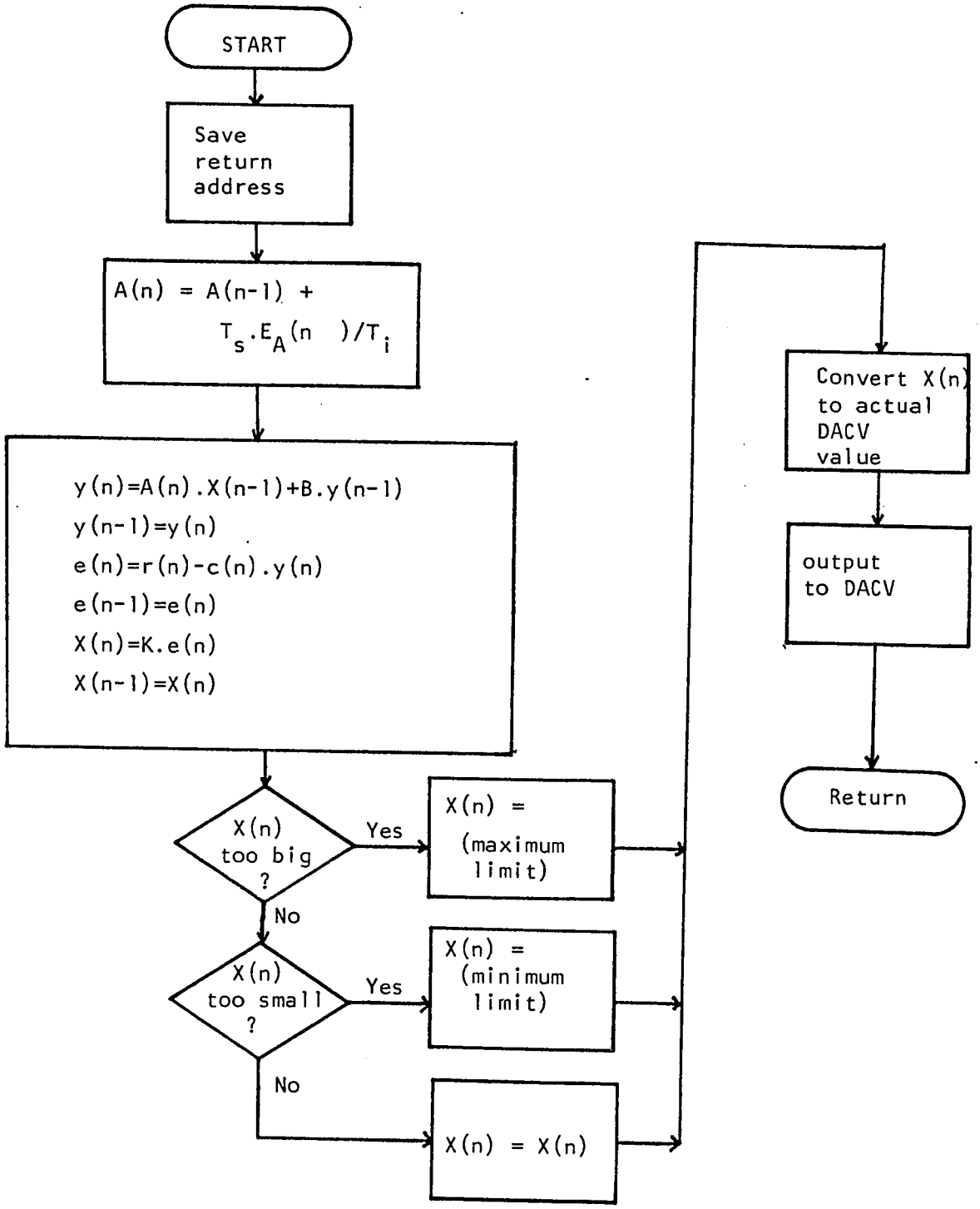


Figure 12 Program MULPLTV Logic Flow Chart

c) Subroutine ALPAR

This was a different version of subroutine ALPAR for the PI control algorithm. This subroutine is called by programs HEEXEC and TTISV if the PI control algorithm is being used in the system. This subroutine requests the operator to enter the parameters for the PI control algorithm. The parameters for the PI control algorithm are proportional gain (K_p) and integral time constant (T_I).

A logic flow chart of this subroutine is shown in figure (13).

d) Subroutine PICNTL

This was the proportional-integral control algorithm in velocity form. The logic of this subroutine was basically the same as that of the Multiplicative control algorithm subroutine except that the PI control algorithm was used.

A logic flow chart of this subroutine is shown in figure (14).

The name of the parameter-requesting subroutine of the Multiplicative and Proportional-Integral control algorithms was the same. Actually, there were many common variables defined in the two control algorithm subroutines. This is the reason why only one of these two control subroutines could be in the system at any one time. Otherwise, a multiple definition error will be generated by the DGC STAND ALONG RELOCATABLE LOADER during system loading time.

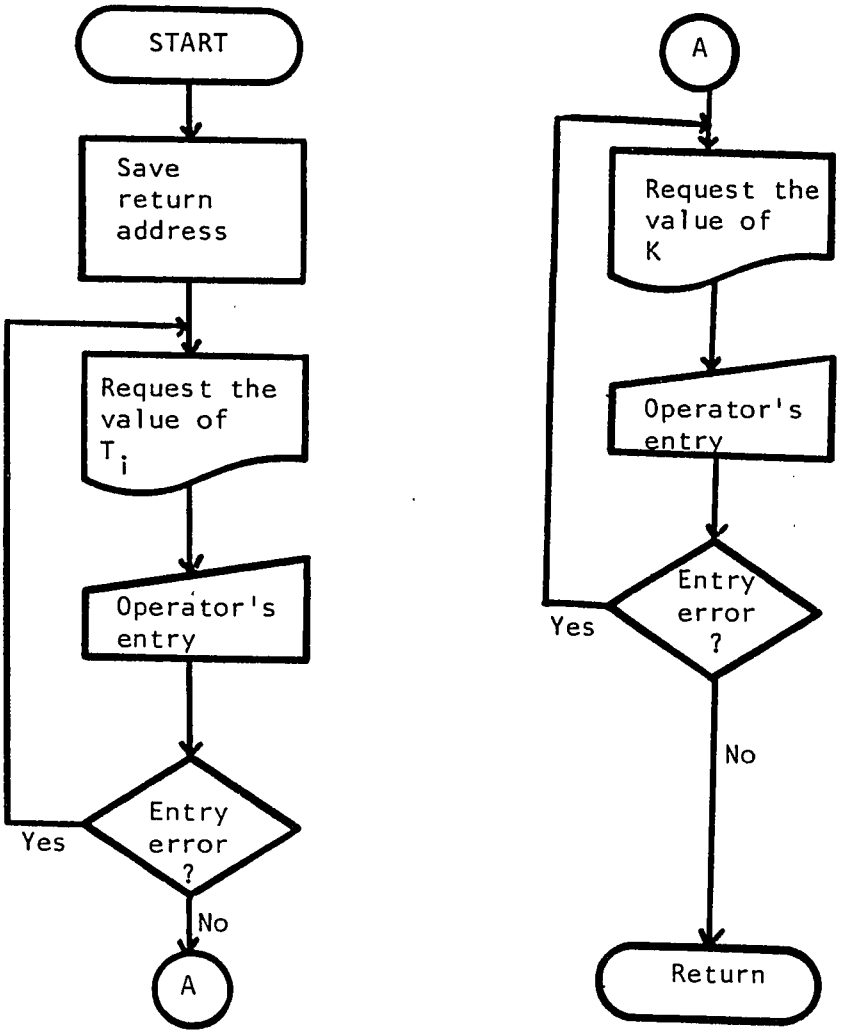


Figure 13 Parameter Request Subroutine ALPAR For the PI Control Algorithm

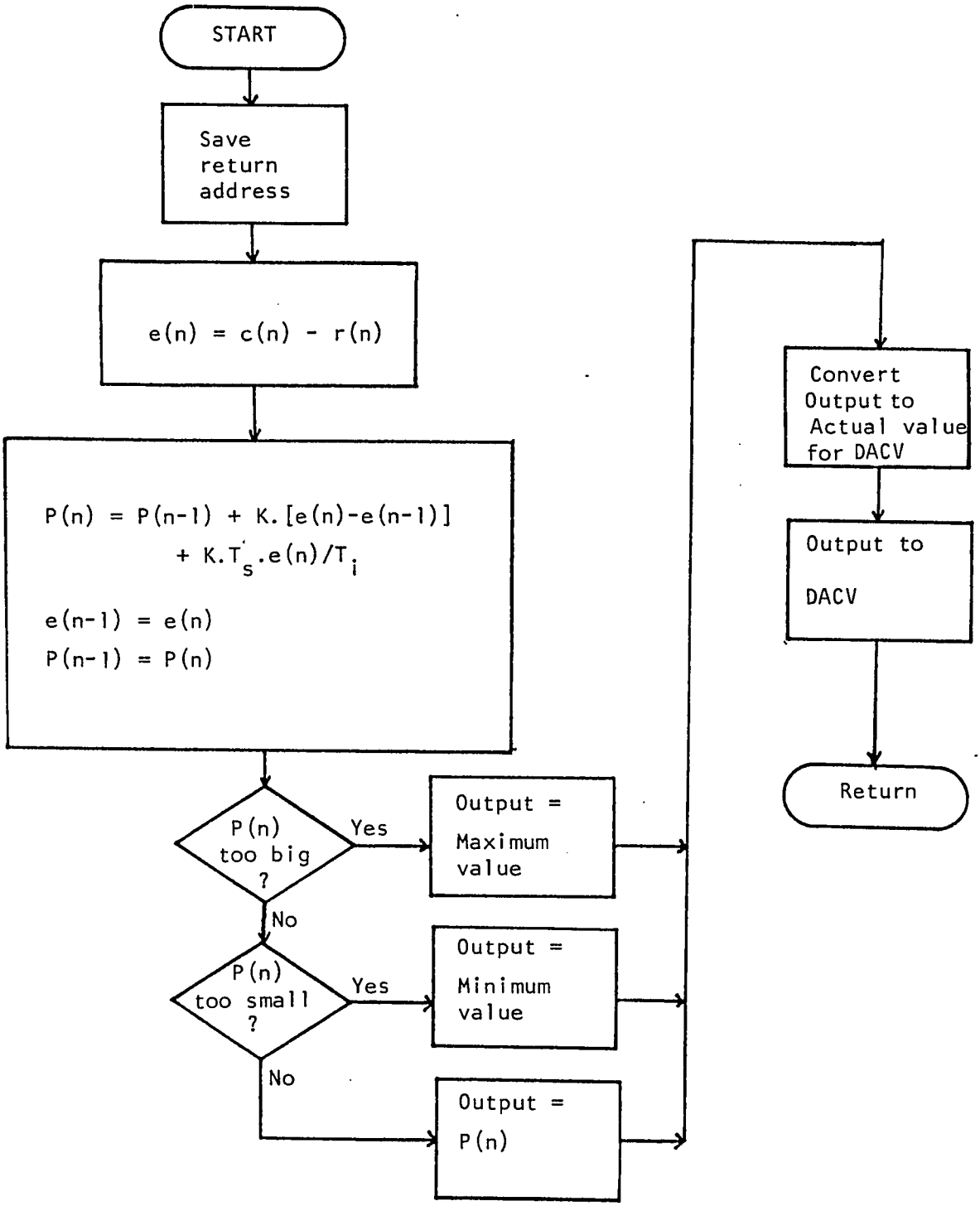


Figure 14 Subroutine PICNTL Logic Flow Chart

4) Data logging and operator communication programs :

Data logging in this system simply means the display or storage of the heat exchanger inlet and outlet temperature for process response tracing or for future study. Since a low speed teletype writer (10 characters per second) was the only hard copy device in the system, data logging speed was very much limited and valuable data could be lost due to the low speed of the output device. The problem was solved by storing the data on paper tape for future study. The teletype was only used as an instantaneous process response tracing.

Computer-operator communication was provided through the teletype writer. A set of commands allows the operator to make a step change in the set point of the controller to initialize the control system and to change the control settings. Details of all the commands are discussed in the write up for program TTISV.

a) Program PTPSV

Program PTPSV was executed whenever the paper tape punch interrupted the system and there was no higher priority program waiting to be executed. The program outputed the heat exchanger inlet and outlet temperature to the paper tape punch in DGC floating point format; two floating point numbers (total eight bytes) with one blank frame following. Priority of this program was just higher than the teletype output handler, which was the lowest priority program in the system. A logic flow chart of this program is shown in figure (15).

b) Program TTOSV

TTOSV was the lowest priority program in the system. When no other program was being executed, the system would branch to this program. If there were new data to be outputed to the teletype writer, this program would print the current heat exchanger inlet and outlet temperature on the teletype writer. If no data were to be outputed, the program would branch to an idle loop and do nothing. A logic flow chart of this subroutine is shown in figure (16).

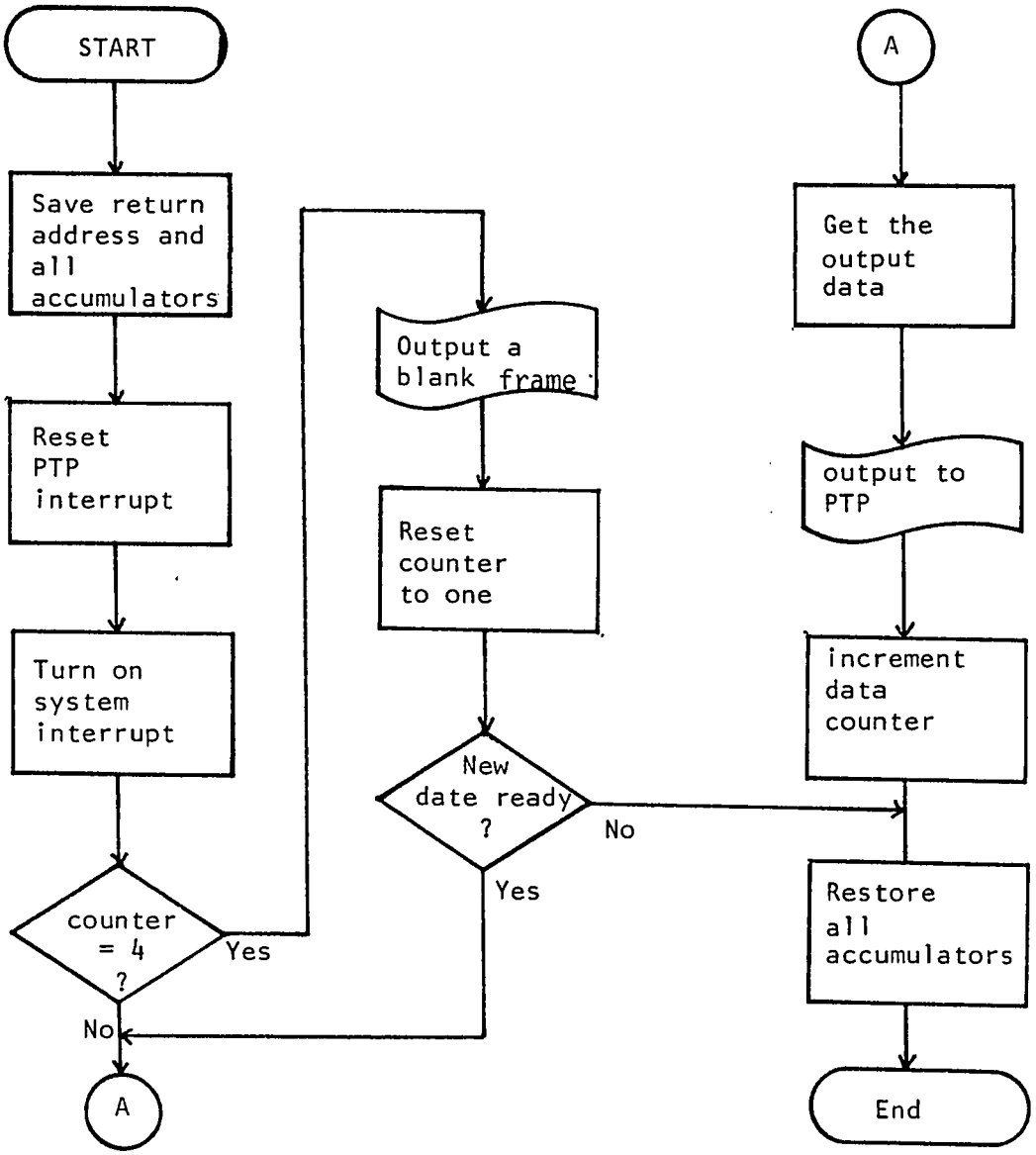


Figure 15 Program PTPSV Logic Flow chart

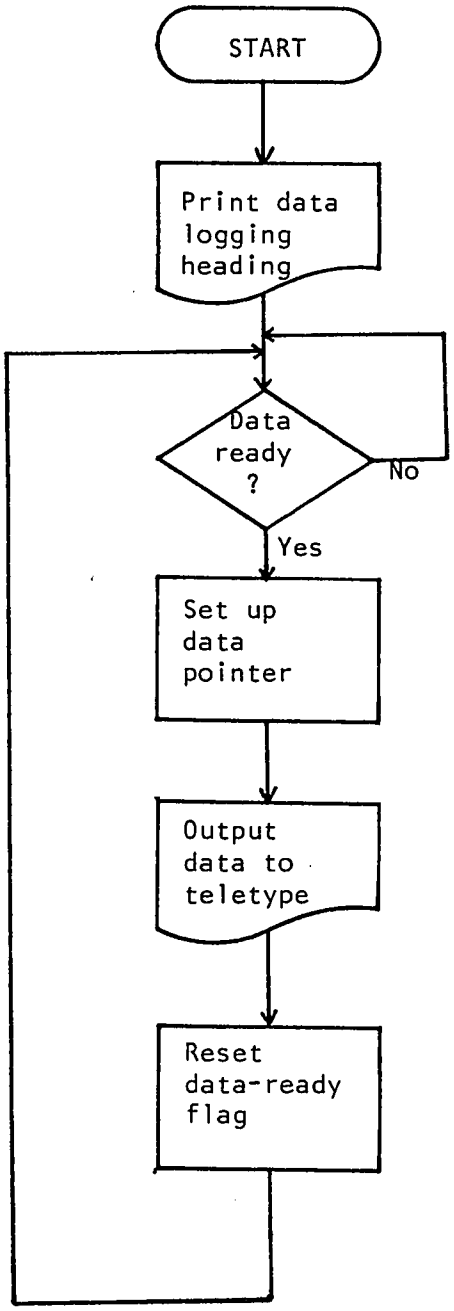


Figure 16 Program TTOSV Logic Flow Chart

c) Program TTISV

This program was executed whenever there was an operator input through the teletype writer. The hardware interrupt of the teletype activated this program. This program was an off-line program to permit the operator to communicate with the control system. When this program was being executed, all the control and data logging activities were terminated and the system was waited for the operator's command.

The operator commands allowed in this program are :

1. S A single S command caused a step change in the controller set point. The value of the step change was stored in relocatable address STEPC. After the set point change, normal control was resumed.
2. R A single R command caused the system to be reinitialized. Control of the system was transfered to the initialization program HEEXEC.
3. RD The RD command caused the DGC symbolic debug processor to be turned on and control was transfered to the debug processor. This command was specially designed for

early stage debugging and used later, as a means of changing the system variable.

4. RA XXXXX The RA command following a five digit octal number caused the system control to be transfered to the address specified by the five digit number XXXXX. This command was designed to test the individual programs.

5. RL The RL command caused subroutine LISTR to be executed to list all the control parameters on the teletype writer. After the listing, the system would wait for the operator for another command through the teletype writer.

6. RSC This command provided the operator with a means of resetting all the control parameters. Subroutine ALPAR of the appropriate control algorithm was turned on and the operator was led through a question and answer type conversation for the required control parameters. After all the parameters were entered, the system would resume control using the new parameters.

A logic flow chart of the operator communication program TTISV is shown in figure (17).

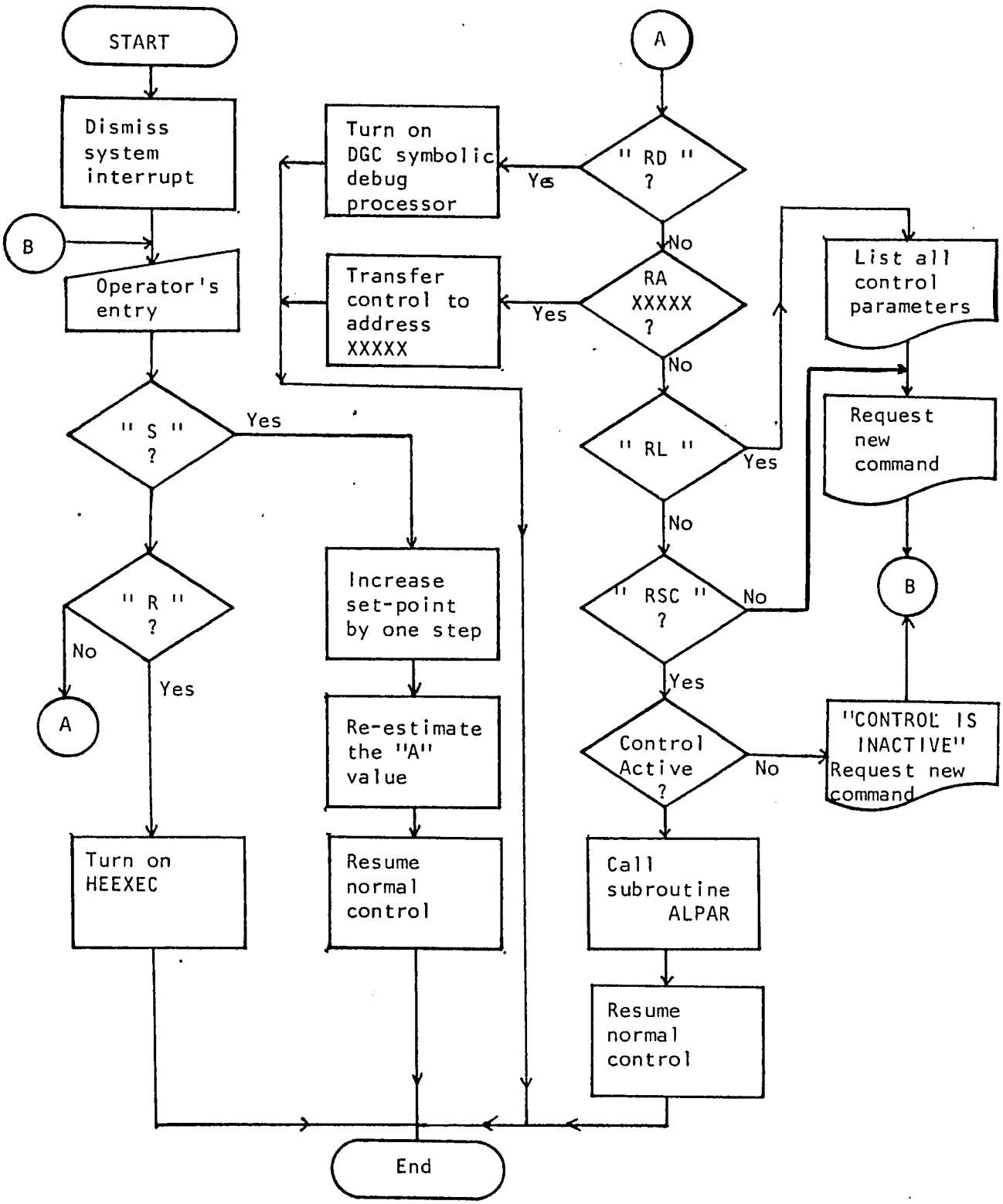


Figure 17 Program TTISV Logic Flow Chart

6) The system initialization program

a) Program HEEXEC

This program was executed when the computer control system underwent a system start up or when the operator reinitialized the control system. The program initialized the system in the following manner :

1. Set up the DGC floating point number handling package. The floating point number error checking subroutine , ERROR,was also set up to provide a floating point checking service to the other system programs.
2. Lead the operator by question and answer to enter system parameters and variables through the teletype writer.
3. Set up a control or no control mode.
4. Initialize the system data tables and reset all pointers and flags.
5. Set up the system interrupt and turn on the real time clock.
6. Wait for a system interrupt to occur.

A logic flow chart of this program is shown in figure (18).

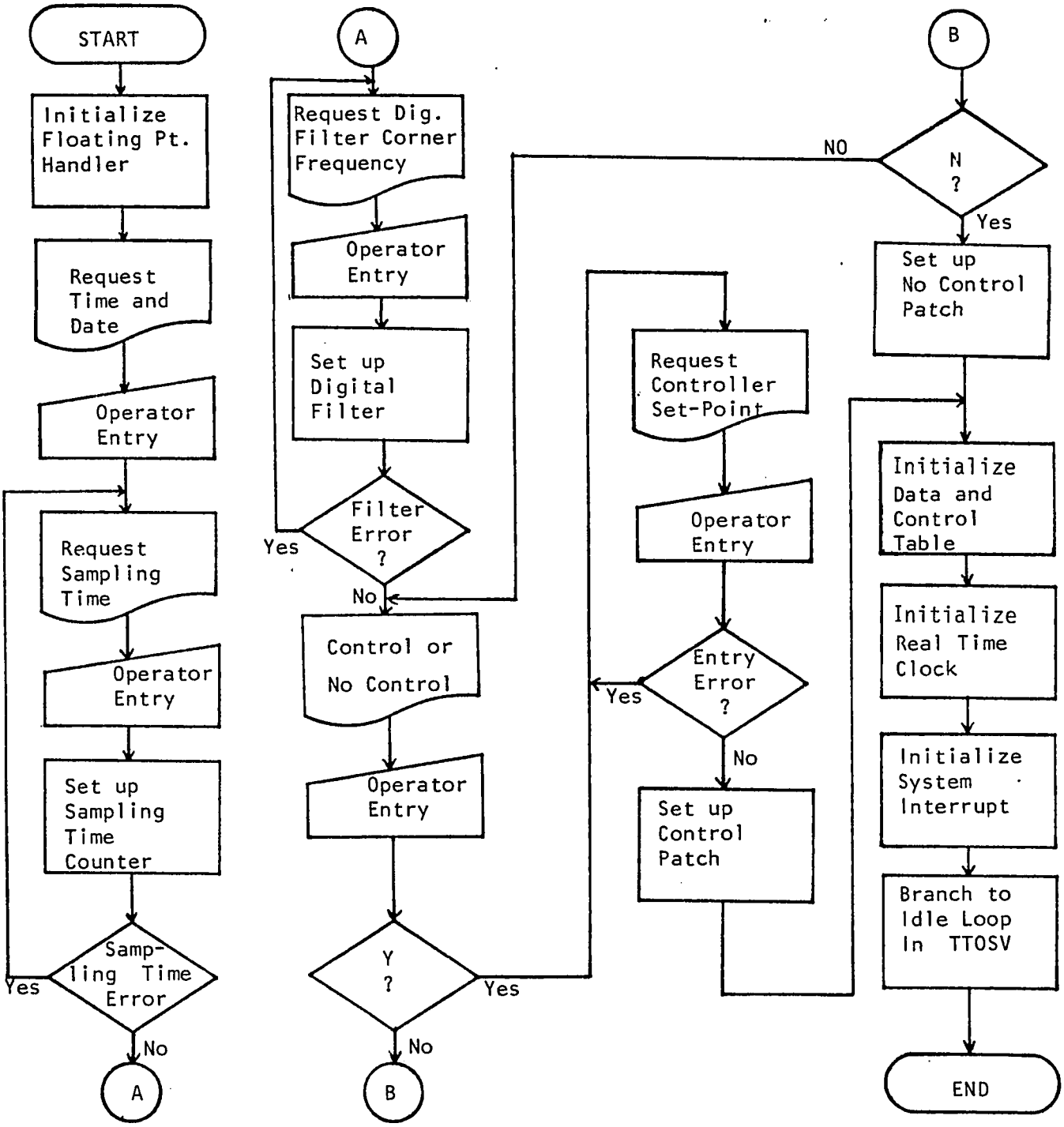


Figure 18 Program HEEXEC Logic Flow Chart

b) Subroutine ERROR

Calling sequence :

JSR @6

(error return)

(normal return)

This subroutine checked the data of the DGC floating point number handler. If overflow, underflow or any other floating point calculation error was detected, the subroutine made an error return (branch to the calling address plus one). If no error was found, a normal return would be assumed (branch to the calling address plus two).

All programs doing floating point number calculations used this subroutine to ensure the calculated result was correct. If an error return was made, the calculated result was not used and the computer halted.

For the convenience of the calling program, the address of this subroutine was stored in memory location 6. A logic flow chart of this subroutine is shown in figure (19).

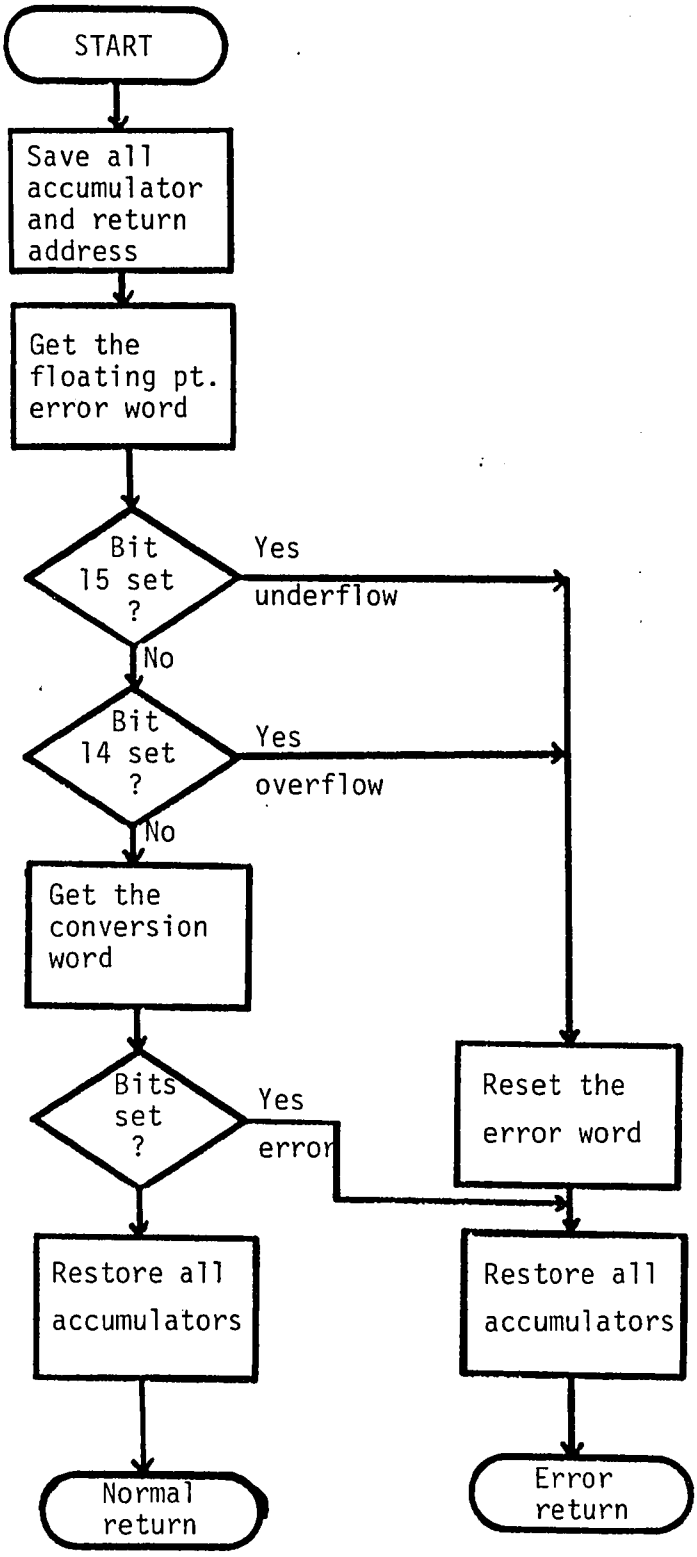


Figure 19 Subroutine ERROR Logic Flow Chart

THE MULTIPLICATIVE CONTROL SYSTEM

Introduction

The theory and application of the linear feedback control are quite well developed and have been used in the chemical and petroleum industries for years. The application of linear network theory to process control has led to the use of mathematical techniques such as the Laplace transformation, frequency response methods, signal flow diagrams and root locus plots. This has resulted in a great emphasis on the use of a linear model when studying process dynamics (3). In practice, almost every process is nonlinear. Linearity can only be applied to a small range around the operating point and this range depends very much upon the non-linearity of the process and control instruments. Engineers quickly realized that few models existed that accurately described dynamic behavior and that very little hardware existed to implement process control.

Meanwhile, great theoretically strides had been taken in the areas of non-linear and optimum control. The requirements of the Aerospace, electrical and nuclear industries, and the military

have led to a new body of knowledge generally referred to as modern control theory. The recent introduction of digital computers to process control was the hardware breakthrough that has permitted the use of non-linear and modern control theories on the actual chemical processes.

The concept of introducing non-linearities into the feedback path was proposed by J. B. Lewis (4) in 1953. Y. H. Ku (5,6) studied the introduction of a non-linear element to the forward and feedback path of a linear control system. Jafri (7) studied the Multiplicative servo control of some simple systems on an analog computer. J. Auns (3) compared the transient response of proportional-integral control and first order Multiplicative control of a concentric tube heat exchanger and suggested that Multiplicative control algorithm is more suitable for servo control purpose. The present work is a study of the stability region and various response characteristics of the simple Multiplicative system suggested by Jafri (7).

The Multiplicative Control Algorithm

The class of multiplicative control system to be studied is shown in figure (20). It can be represented mathematically by the equation :

$$b_n(t) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\cdots\int_{-\infty}^{+\infty} [G_1(Z_1).x(t-Z_1)+u].G_2(Z_2).X(t-Z_2).G_3(Z_3).X(t-Z_3).....$$

$$...G_n(Z_n).X(t-Z_n).G_{n+1}.X(t-Z_{n+1}).dZ_1.dZ_2...dZ_n.dZ_{n+1}$$

----------(1)

or equation (1) can be rearranged to become :

$$b_n(t) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\cdots\int_{-\infty}^{+\infty} [G_1.(Z_1).X(t-Z_1)+u].G_2(Z_2).G_3(Z_3).....G_n(Z_n).G_{n+1}(Z_{n+1}).$$

$$X(t-Z_2).X(t-Z_3)....X(t-Z_n).X(t-Z_{n+1}).dZ_1.dZ_2.$$

$$dZ_3.....dZ_n.dZ_{n+1}$$ ----------(2)

or equation (2) be further arranged to become :

$$b_n(t) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\cdots\int_{-\infty}^{+\infty} G_1(Z_1).G_2(Z_2).G_3(Z_3).....G_n(Z_n).G_{n+1}(Z_{n+1}).$$

$$X(t-Z_1).X(t-Z_2).X(t-Z_3)....X(t-Z_n).X(t-Z_{n+1}).$$

$$dZ_1.dZ_2.dZ_3.....dZ_n.dZ_{n+1} +$$

$$... \quad u.G_2(Z_2).G_3(Z_3)...G_{n+1}(Z_{n+1}).X(t-Z_2).X(t-Z_3).....$$

$$X(t-Z_{n+1}).dZ_1.dZ_2.dZ_3.....dZ_n.dZ_{n+1}$$ ----------(3)

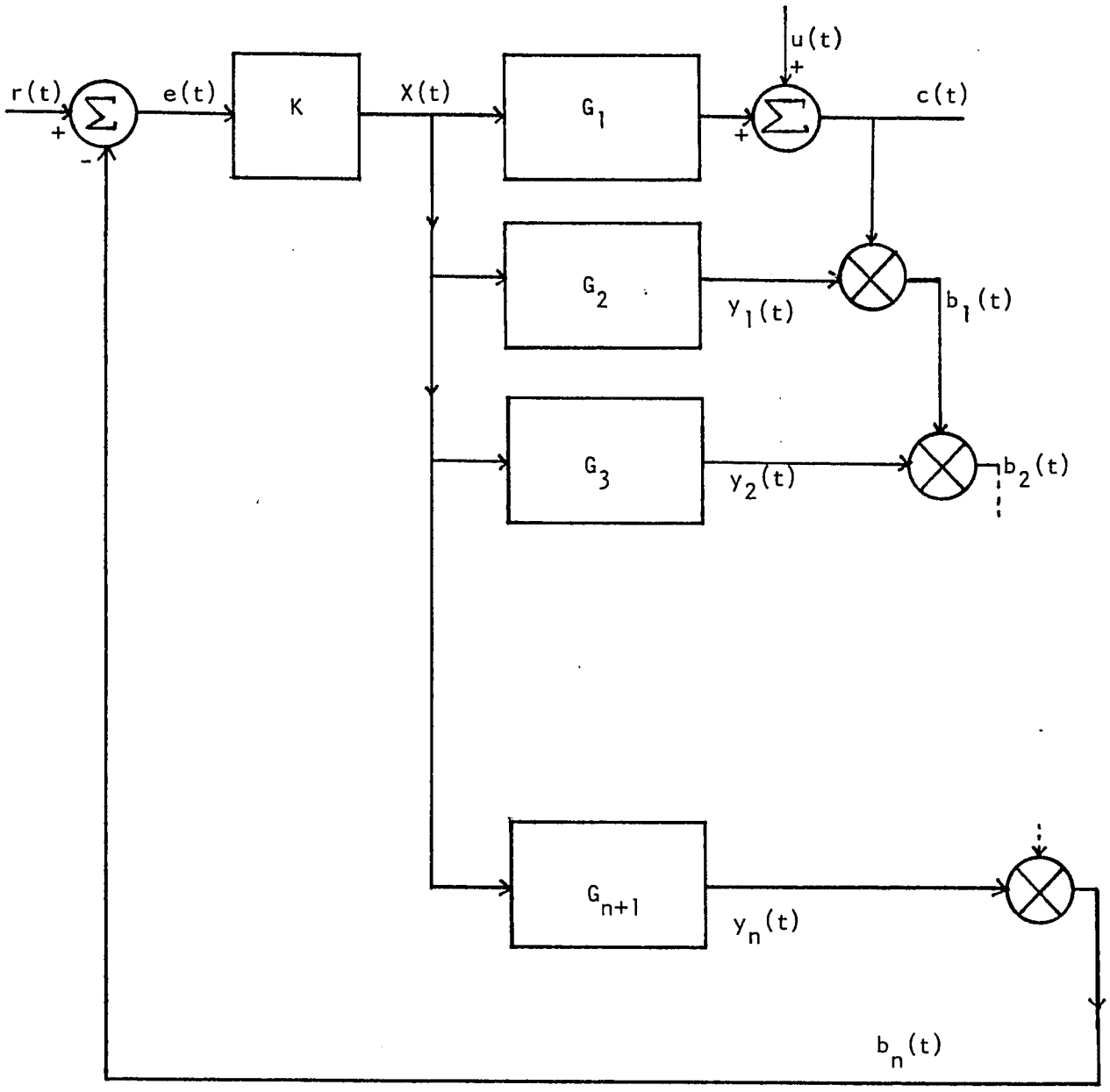


Figure 20 The Multiplicative Control System

The above equation shows that the performance of the Multiplicative control systems can be expressed by (n+1) dimension kernel $G_1(Z_1)....G_{n+1}(Z_{n+1})$ and the expression is known as a "regular homogeneous functional" (3). This kind of functional was introduced by Volterra (8) and applied to non-linear system representation and analysis by Wiener (9) and Barret (10).

G_1 in figure (20) represents the actual process whose natural dynamic performance is to be modified by the controller. r(t) is the set point of the controller. The rest of figure (20) is referred to as the Multiplicative control algorithm. Since most of the process control computer are mini-computers, it is economical to simplify the complex system of (n+1)th integration shown in equation (3) to reduce the loading of the process computer. According to Jafri (7), the system was reduced to the form of a first order Multiplicative control system shown in figure (21). G_2 represents the Multiplicative control functional. K is the Multiplicative gain. r(t) is the set point of the controller. c(t) represents the temperature of the outlet water from the heat exchanger.

It is clearly shown in figure (21) that G_2, a non-linear multiplication, is introduced into the feedback path of a proportional

control system. For r(t) to be equal to c(t) at steady state, a judicious choice of the value "A" in G_2 would be required. The basis for the calculation of "A" for this condition is shown in figure (22).

In figure (22), u_1 is the load function and u_2 is a constant added to the thermocouple reading during the conversion of the analog signal to engineering unit (°F). Both u_1 and u_2 are constants. The derivation of the "A" value is shown in the following section.

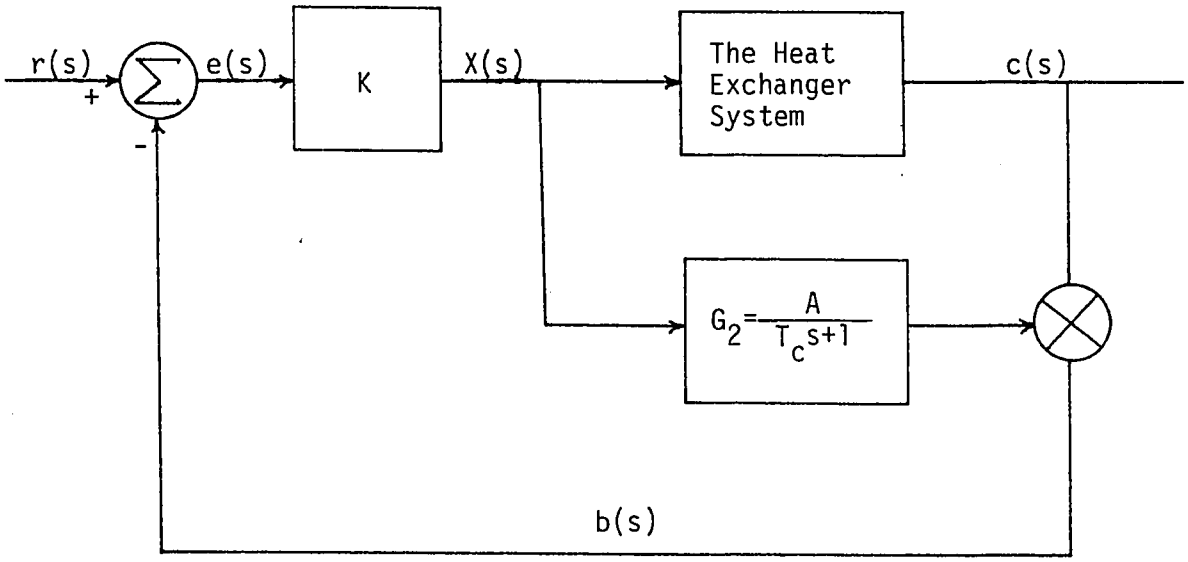


Figure 21 Simplified Multiplicative Control System

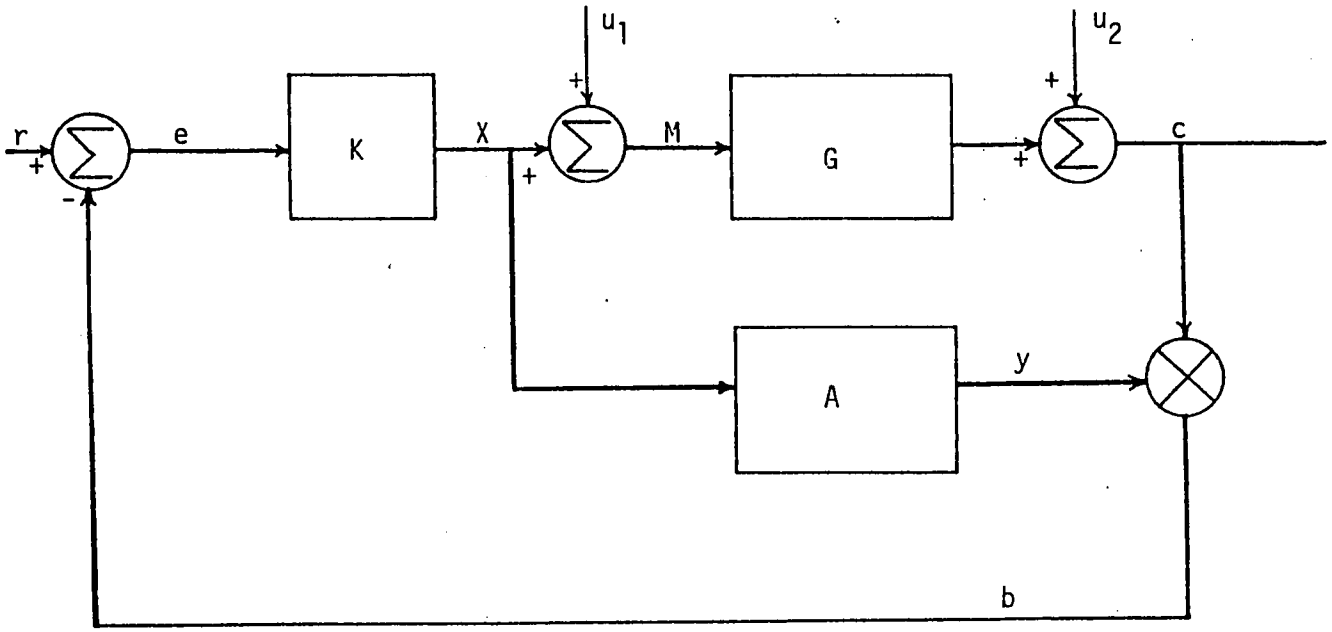


Figure 22 The Multiplicative Control System At Steady State

According to figure (22), the steady state configuration, we have the following set of relationship :

$$e = r - b \quad \text{---------- (4)}$$
$$X = K.e \quad \text{---------- (5)}$$
$$y = X.A \quad \text{---------- (6)}$$
$$M = X + u_1 \quad \text{---------- (7)}$$
$$c = M.G + u_2 \quad \text{---------- (8)}$$
$$b = c.y \quad \text{---------- (9)}$$

The present work concerned about the Multiplicative servo control. u_2 thus became a constant. The derivation of the "A" value starts with equation (8).

$$\begin{aligned} c &= M.G + u_2 \\ &= (X + u_1).G + u_2 \\ &= G.X + G.u_1 + u_2 \\ &= G.K.e + G.u_1 + u_2 \\ &= G.K.(r - b) + G.u_1 + u_2 \\ &= G.K.r - G.K.b + G.u_1 + u_2 \\ &= G.K.r - G.K.c.y + G.u_1 + u_2 \\ &= G.K.r - G.K.c.X.A + G.u_1 + u_2 \\ &= G.K.r - G.K.c.A.(M - u_1) + G.u_1 + u_2 \\ &= G.K.r - G.K.c.A.M + G.K.c.A.u_1 + G.u_1 + u_2 \\ &= G.K.r - K.c.A.(c - u_2) + G.K.c.A.u_1 + G.u_1 + u_2 \\ &= G.K.r - K.c^2.A + K.c.A.u_2 + G.K.c.A.u_1 + G.u_1 + u_2 \end{aligned}$$

$$c = G.K.r - K.c^2.A + K.c.A.u_2 + G.K.c.A.u_1 + G.u_1 + u_2 \quad \text{----- (10)}$$

The aim of the control action is to make the control variable (c) equal to the control set point (r). To satify this condition, c in equation (10) was replaced by r.

$$r = G.K.r - K.r^2.A + K.r.A.u_2 + G.K.r.A.u_1 + G.u_1 + u_2$$

or

$$r = G.K.r - G.u_1 + u_2 + A.(K.r.u_2 + K.G.r.u_1 - K.r^2)$$

or

$$A = \frac{r - G.K.r - G.u_1 - u_2}{K.r.u_2 + K.r.G.u_1 - K.r^2}$$

or

$$A = \frac{K.G.r + G.u_1 + u_2 - r}{K.r^2 - G.K.r.u_1 - K.u_2.r} \qquad \text{---------- (11)}$$

Equation (11) is the steady state value of the Multiplicative constant "A". Equation (11) was derived under the assumption that all variables in the right hand side of the equation are constant. This is discussed in more detail in the section concerning the modified Multiplicative control algorithm.

Figure (23) shows the Multiplicative control system in discrete time format. The derivation from continuous time (Laplace transformation) to discrete time (Z-transformation) is shown in Appendix I. The following is the derivation of the digital Multiplicative control algorithm.

Directly from figure (23), the following relationships can be obtained.

$$\frac{y(Z)}{X(Z)} = \frac{A.(1-\exp(-T_s/T_c)).Z^{-1}}{1 - \exp(-T_s/T_c).Z^{-1}} \quad \text{---------- (12)}$$

$$e(n) = r(n) - c(n).y(n) \quad \text{---------- (13)}$$

$$X(n) = K.e(n) \quad \text{---------- (14)}$$

$$\text{Let } B = \exp(-T_s/T_c) \quad \text{---------- (15)}$$

Substitute equation (15) into equation (12)

$$\frac{y(Z)}{X(Z)} = \frac{A.(1-B).Z^{-1}}{1 - B.Z^{-1}} \quad \text{---------- (16)}$$

Take the inverse Z-transformation of equation (16) :

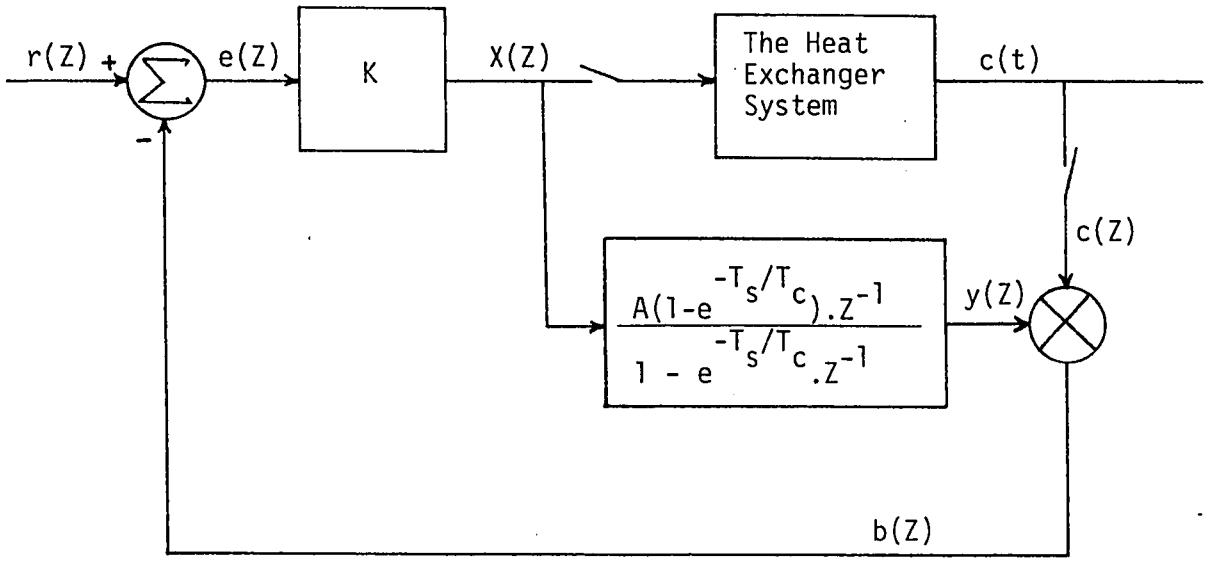


Figure 23 The Multiplicative Control Algorithm in Discrete Time Format

$$y(n) - B.y(n-1) = A.(1-B).X(n-1)$$

or

$$y(n) = A.(1-B).X(n-1) + B.y(n-1) \quad \text{---------- (17)}$$

The basic Multiplicative control algorithm used in this project consisted of equations (11), (13), (14), (15) and (17). In the control program, the calculation sequence may be summarized as follows :

1. The "A" value was calculated using equation (11) with the assumption that the process gain was constant.
2. A sample of c(n) was taken from the analog-to-digital converter and converted into engineering unit (°F).
3. The intermediate variable y(n) was calculated using equation (17).
4. The instantaneous error e(n) was calculated using equation (13) and the intermediate variable y(n).
5. Controller output X(n) was calculated using equation (14).
6. X(n) was limited between maximum and minimum values and converted into the output format.
7. The final value of X(n) was then outputed to the digital-to-analog converter to control the process.

8. This sequence was repeated form step 2 for each sampling period.

The above procedure was programmed in the Multiplicative control subroutine MULPLTV.

The Modified Multiplicative Control Algorithm

The derivation of the Multiplicative control constant "A" was discussed in the previous section. However, equation (11) is based on the assumption that all variables in the right hand side of the equation are constant.

$$A = \frac{K.G.r + G.u_1 + u_2 - r}{K.r^2 - G.K.r.u_1 - K.u_2.r} \qquad \text{---------- (11)}$$

It is obvious that if any of the variables in the right hand side is not constant, error would be introduced through "A" which would appear as an offset at steady state. The phenomenon was also noticed by Auns (3).

As we look closely at equation (11), K and u_2 undoubtedly are constants. u_1 can be considered as a constant for servo control because the actual experiment water flow rate was held constant. r, the controller set point is changed during servo control, but the exact value of the set point is known and the respective "A" value can be calculated. The process gain G is the only possible error introduced to equation (11). As a matter of fact, the shell and tube heat exchanger

is a non-linear distributed parameter system. It was very difficult to determine the exact gain and incorrect value of the process gain would cause an offset at steady state. This kind of error is very common in feed-forward control (2,11). A technique used in feed-forward control to eliminate offset is to introduce a feedback loop. A similar scheme was developed for the Multiplicative control algorithm. As shown in figure (24), the control variable c(Z) is compared with the set point r(Z) to generate an error $E_A(Z)$ which is fed into an integrator to provide a modified "A" value. The rest of figure (24) is the Multiplicative control algorithm. The equation for the manipulation of the "A" value is shown below :

$$A(n) = A(n-1) + K_I.T_s.E_A(n)/T_I \quad \text{---------- (18)}$$

The modified Multiplicative control algorithm consisted of six equations, namely equation (11), (13), (14), (15), (17) and equation (14).

Equation (11) was used to calculate an initial estimated "A" value for the controller. The other five equations were programmed in the modified version of the Multiplicative control program MULPLTV to perform on-line direct digital control.

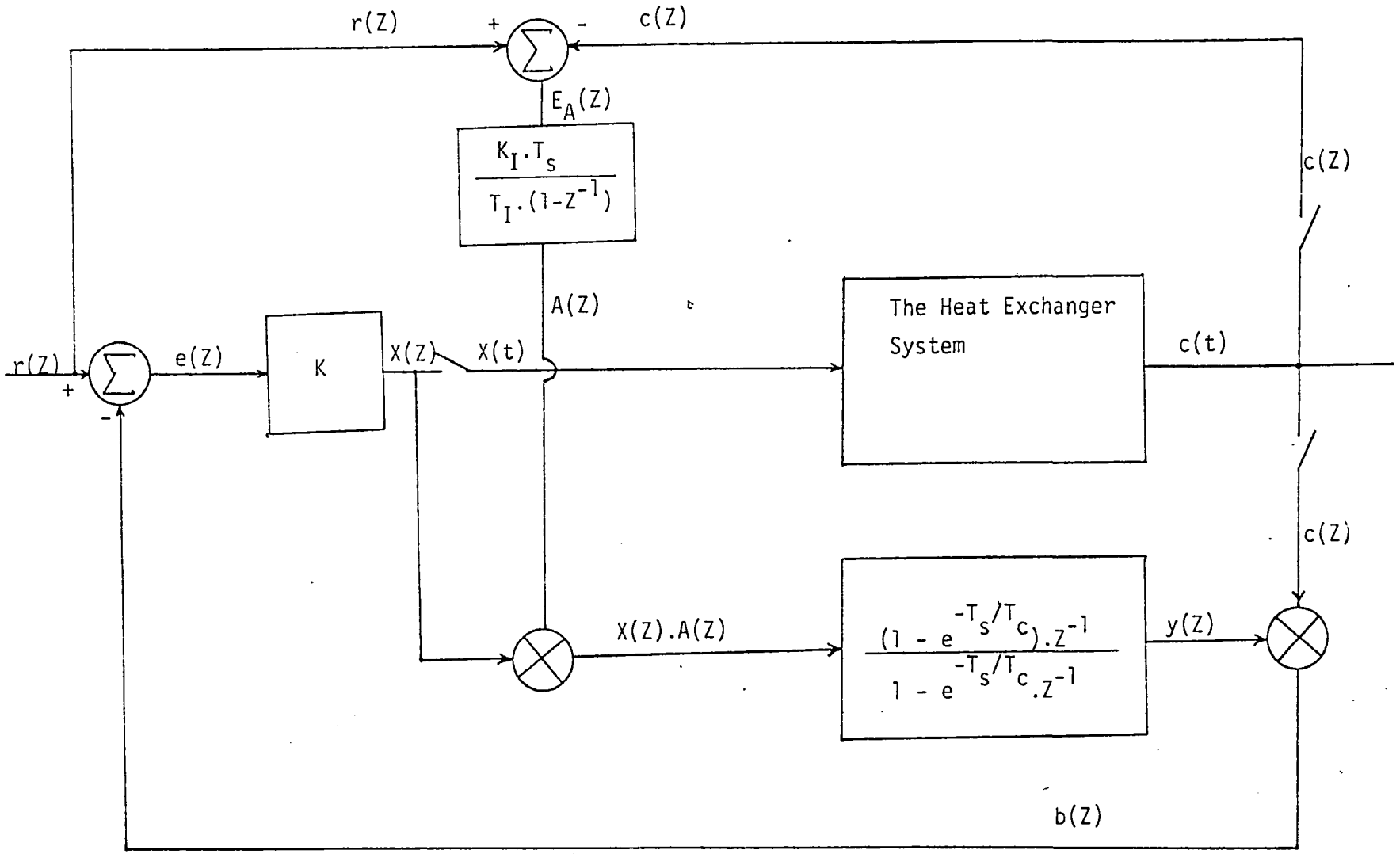


Figure 24 The Modified Multiplicative Control Algorithm

The experiment

The experiments of this project can be divided into three parts :

1. Determination of a simplified model of the heat exchanger system.
2. On-line Multiplicative control of the analog simulation of the heat exchanger system.
3. On-line Multiplicative control of the heat exchanger system.

The details of each part of the experiment is discussed in the following sections.

1. Heat exchanger system modeling :

A simple second order model of the heat exchanger system which consisted of the shell and tube heat exchanger, the pneumatic control valve, the valve positioner and the thermocouple was developed using frequency response and step tests. The Bode diagram from the frequency response tests is shown in figure (25) and the step change response is shown in figure (26).

As shown in figure (25), the slope of the asymptote of the amplitude ratio vs frequency plot is -2 which indicated that the system can be represented as a second order system. It was found that two first order systems with the same time constant closely represented the data. The time constant T_p was determinated at $1/W_c$, the reciprocal of the corner frequency W_c. The corner frequency in figure (25) is 0.1885 rad/sec. The basic form for the heat exchanger system as shown in equation (19).

$$G_p = \frac{G}{(5.305S + 1)^2} \qquad \text{----------} \qquad (19)$$

From the phase angle vs frequency plot, it can be recognized that the system contained dead time. The dead time was evaluated using a step response test. As shown in figure (26), approximately one second dead time was encounted. Equation (20) is the simple model obtained and used through out the analog computer simulation.

$$G_p = \frac{G.\exp(-S)}{(5.305S + 1)^2} \qquad \text{----------} \qquad (20)$$

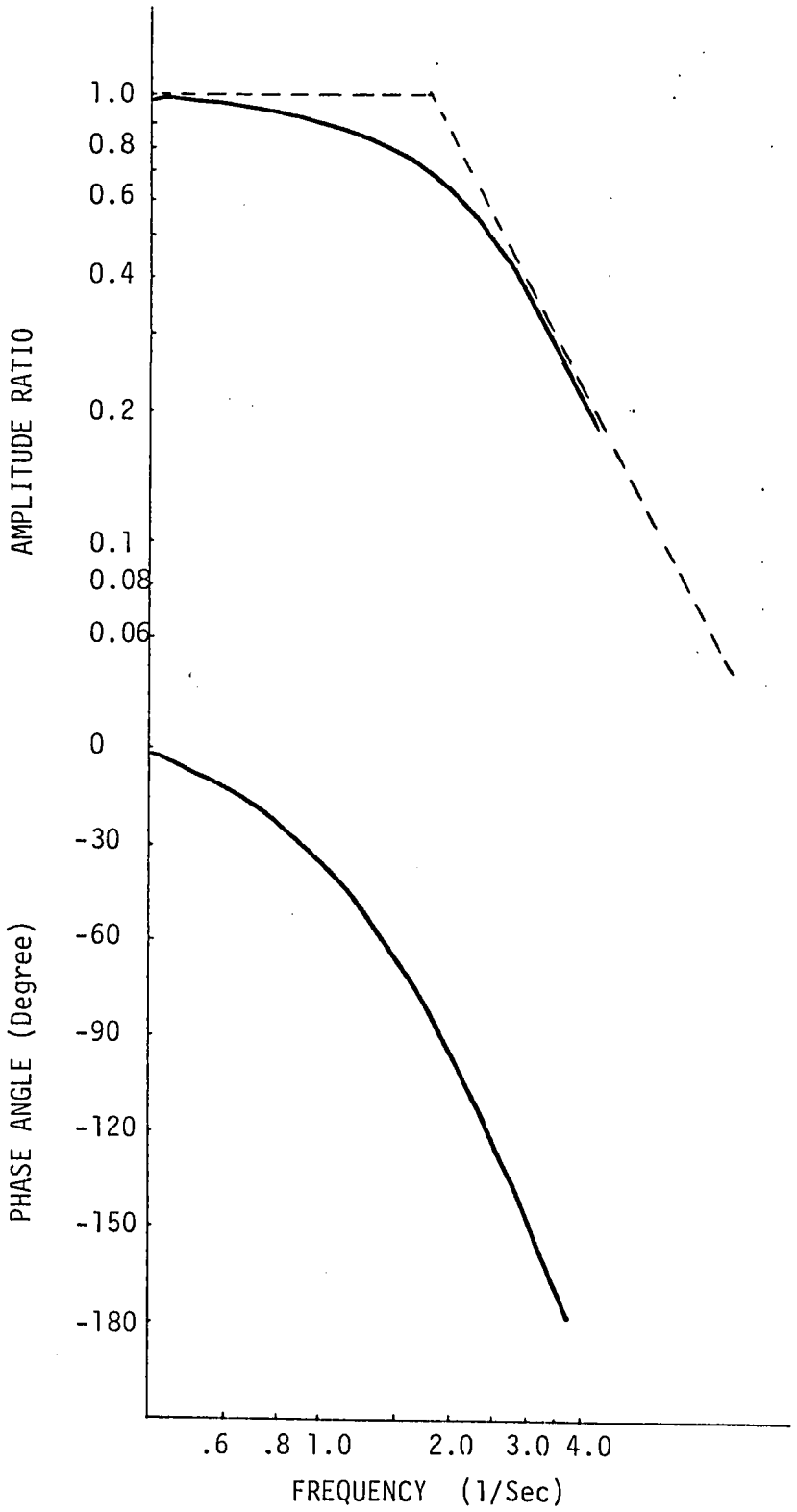


Figure 25 Bode Plot for the Heat Exchanger System

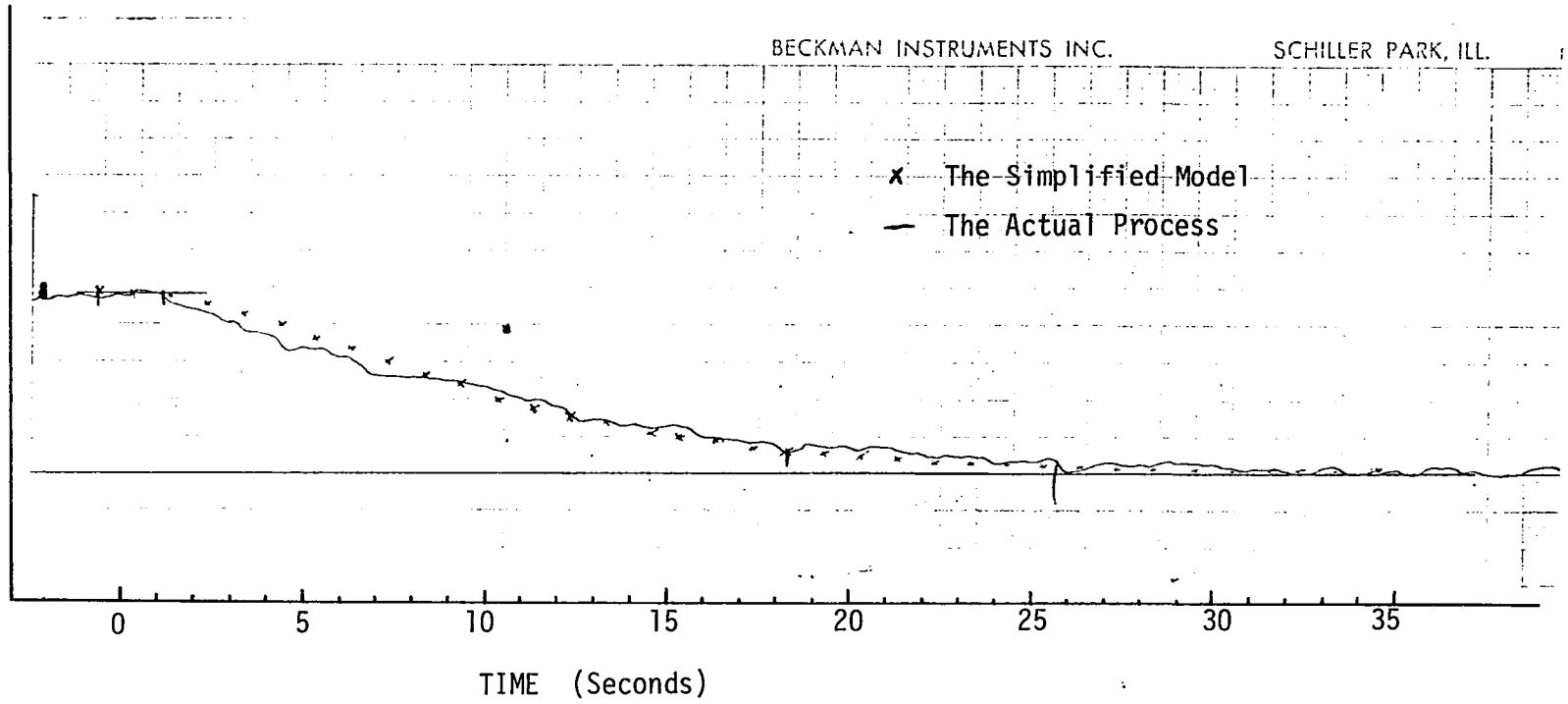


Figure 26 Comparision of Step Response for the Simplified Model vs the Actuat Process

The term G is the process gain which is a function of control temperature but was kept constant during the analog simulation experiments.

2) On-line Multiplicative control on analog computer :

Equation (20) was programmed on a Comdyna GP-6 analog computer. The analog computer was interfaced with the NOVA digital computer. The digital computer was programmed with the software described in the software section using the unmodified Multiplicative control algorithm. The reason for using the unmodified Mulplicative control algorithm is that the simulated model was a linear model and the process gain was known.

The experimental procedure to study the unstable region of the Multiplicative control algorithm with the simulated heat exchanger system is as follows :

1. Turn on the digital computer and the analog computer. Make sure the input and output signals are properly connected.

2. Set the controller set point to 140 °F. The selection of 140 °F placed the control valve at the middle position which was the most linear portion.
3. Wait until the system reaches steady state.
4. Introduce a 5 °F step change in the set point and observe the system response.
5. If another steady state were reached, reset the controller set point to 140 °F and tune the controller settings K and T_c manually to wards the unstable region of the controller.
6. Repeat the procedure from step 3.

The above procedure was followed to tune the Multiplicative controller towards the unstable region until the system started to oscillate. The first setting that made the system started to oscillate was recorded. Different combinations of the control settings K and T_c were recorded by repeating the above procedure. The values of the critical settings are plotted in figure (27) showing the unstable region of the Multiplicative controller for the simulated heat exchanger.

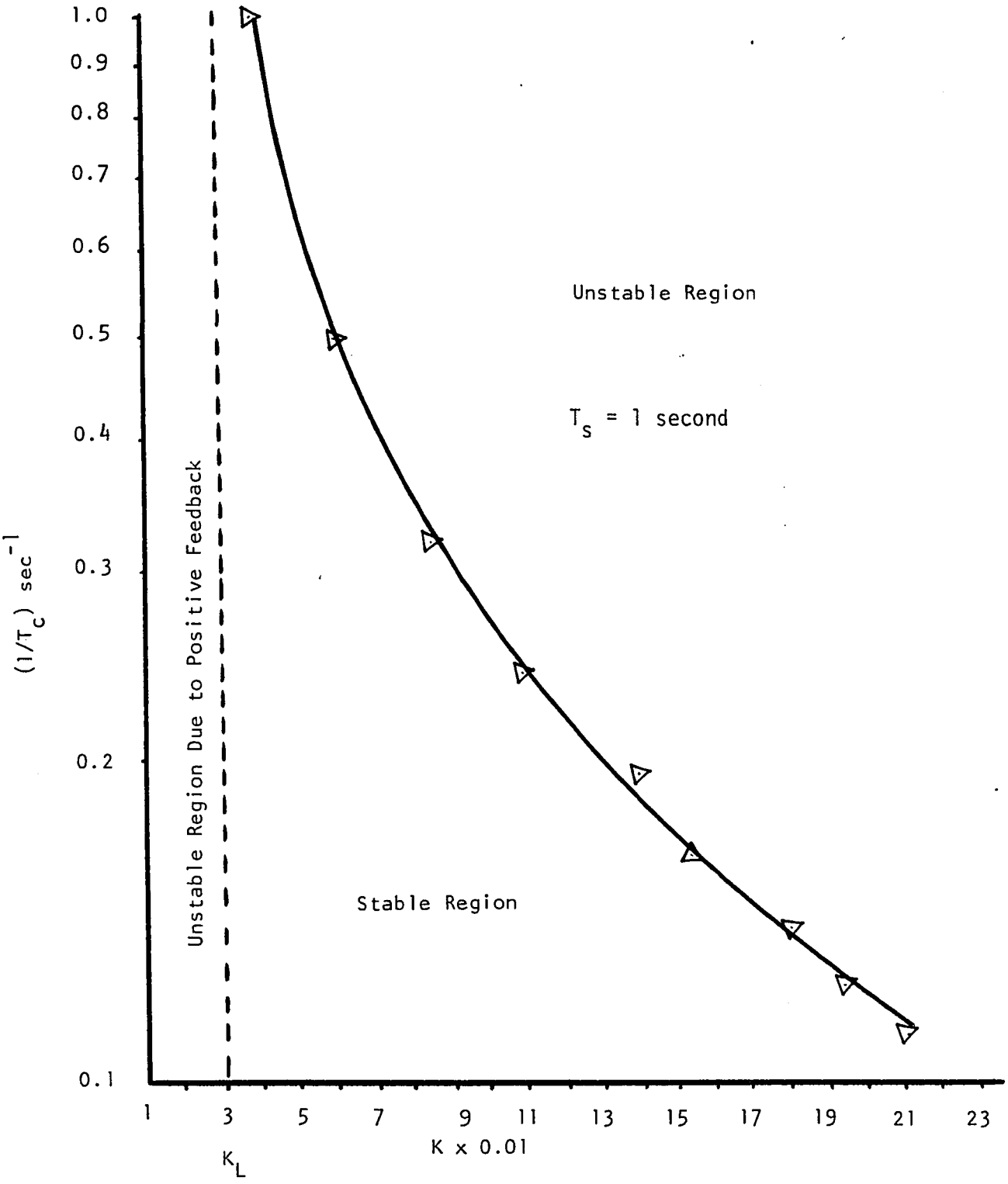


Figure 27 Stability Region of Multiplicative Control Through Analog Computer Simulation

3) On-line Multiplicative control of the heat exchanger system :

After the experiments using the analog computer, the NOVA digital computer was interfaced with the real process, the shell and tube heat exchanger. Thermocouple readings were scanned and a voltage was generated by the computer to drive the steam flow rate control valve.

The same six step procedure used in the analog simulation study was used to determine the stability limits. The stability region of the modified Multiplicative control algorithm controlling the real heat exchanger is plotted in figure (28). Figure (29) compares the analog computer simulation with the real heat exchanger.

Another set of experiments to study transient response of the Multiplicative control algorithm was also conducted. This set of experiments studied the effect of individual tuning parameters on transient response. There were four basic tuning variables, the proportional gain (K), the Multiplicative time constant (T_c), the sampling time (T_s) and the integral time constant (T_I). Three of the four tuning constants were fixed while the fourth was tuned and the results were recorded. Four sets of representative results are shown in figures (30) - (33). The time scale for the figures is three seconds per division.

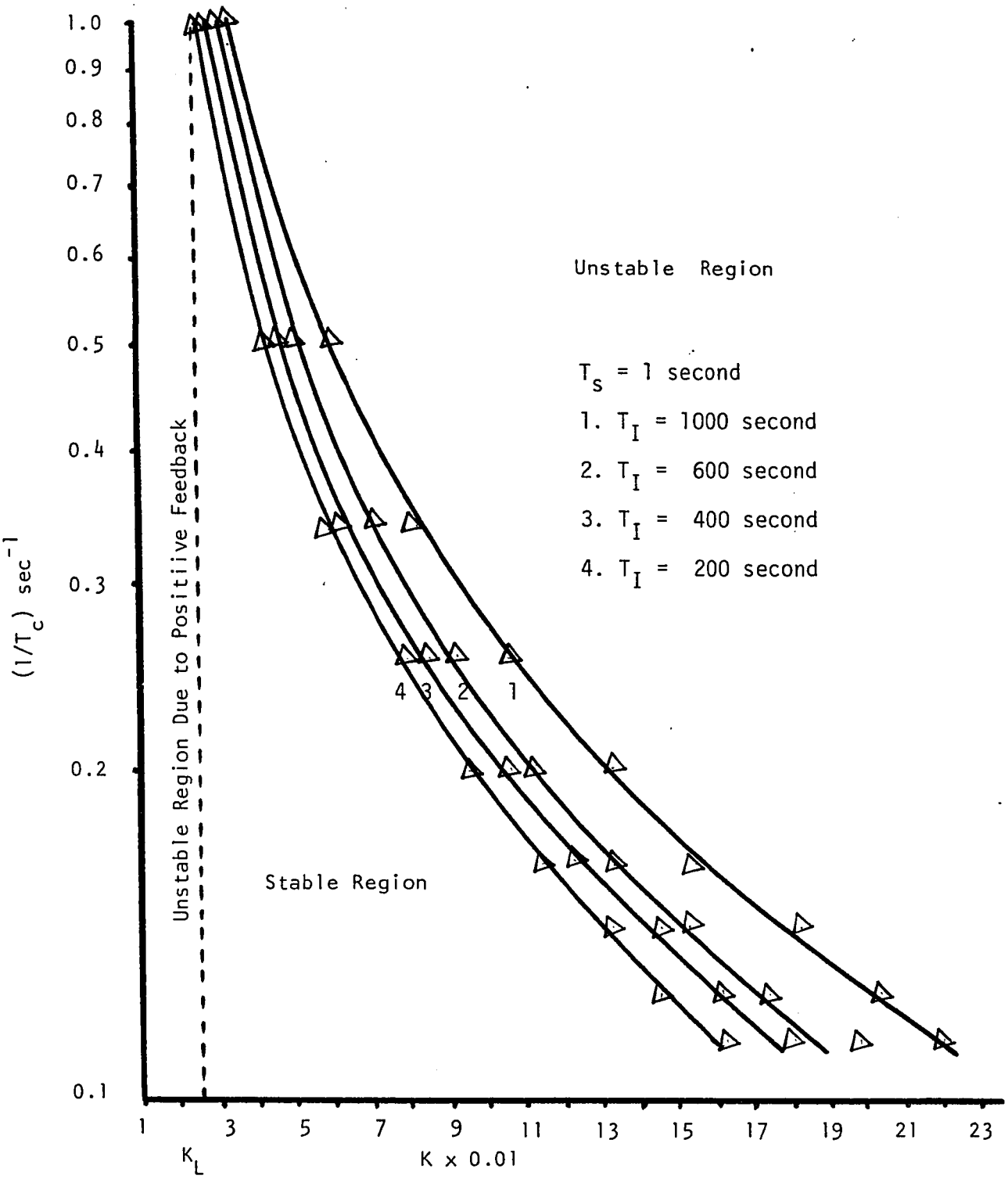


Figure 28 Stability Region of Multiplicative Control Applied to the Heat Exchanger

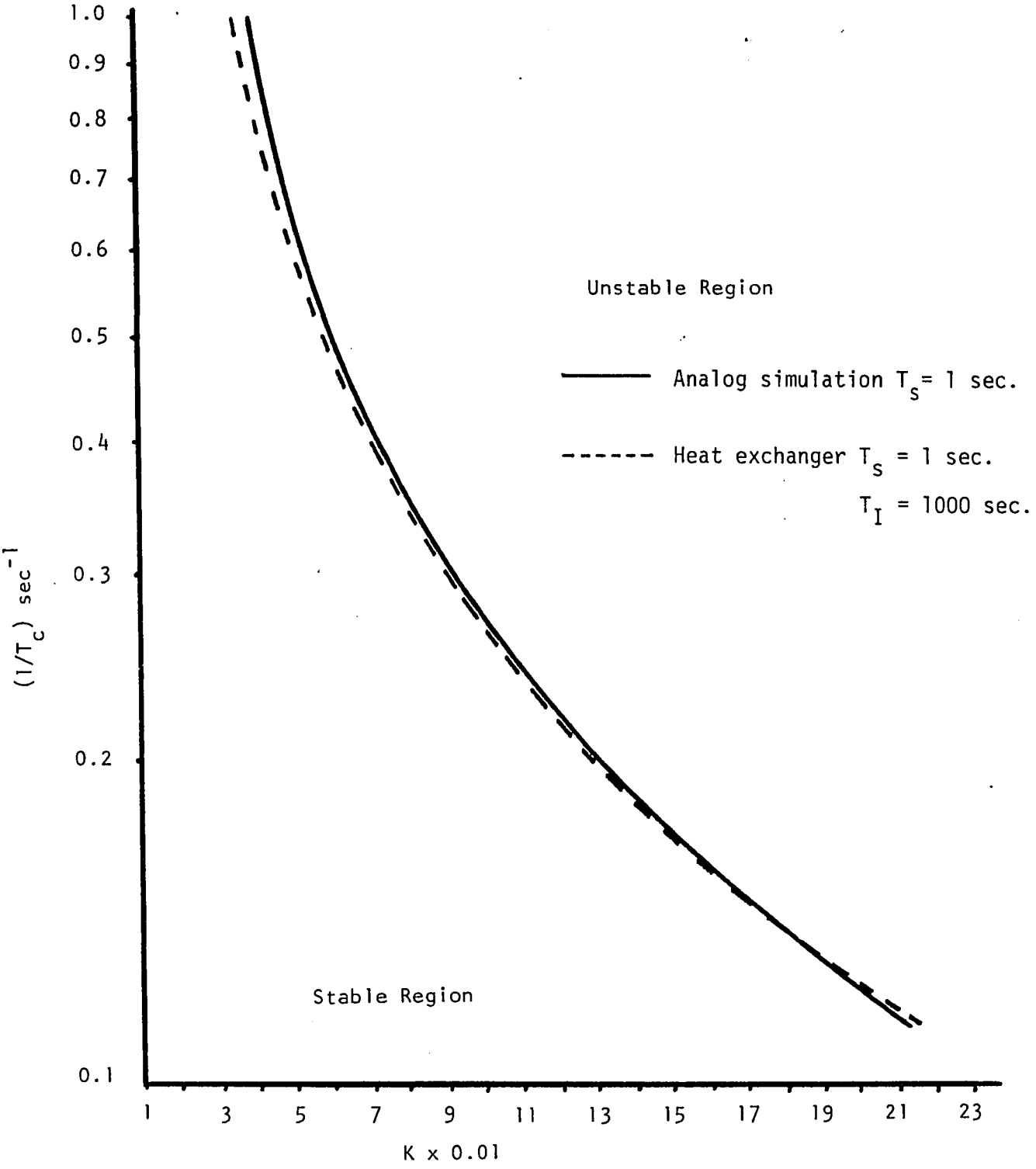


Figure 29 Comparison of Stability Region for the Heat Exchanger and the Simulated Model

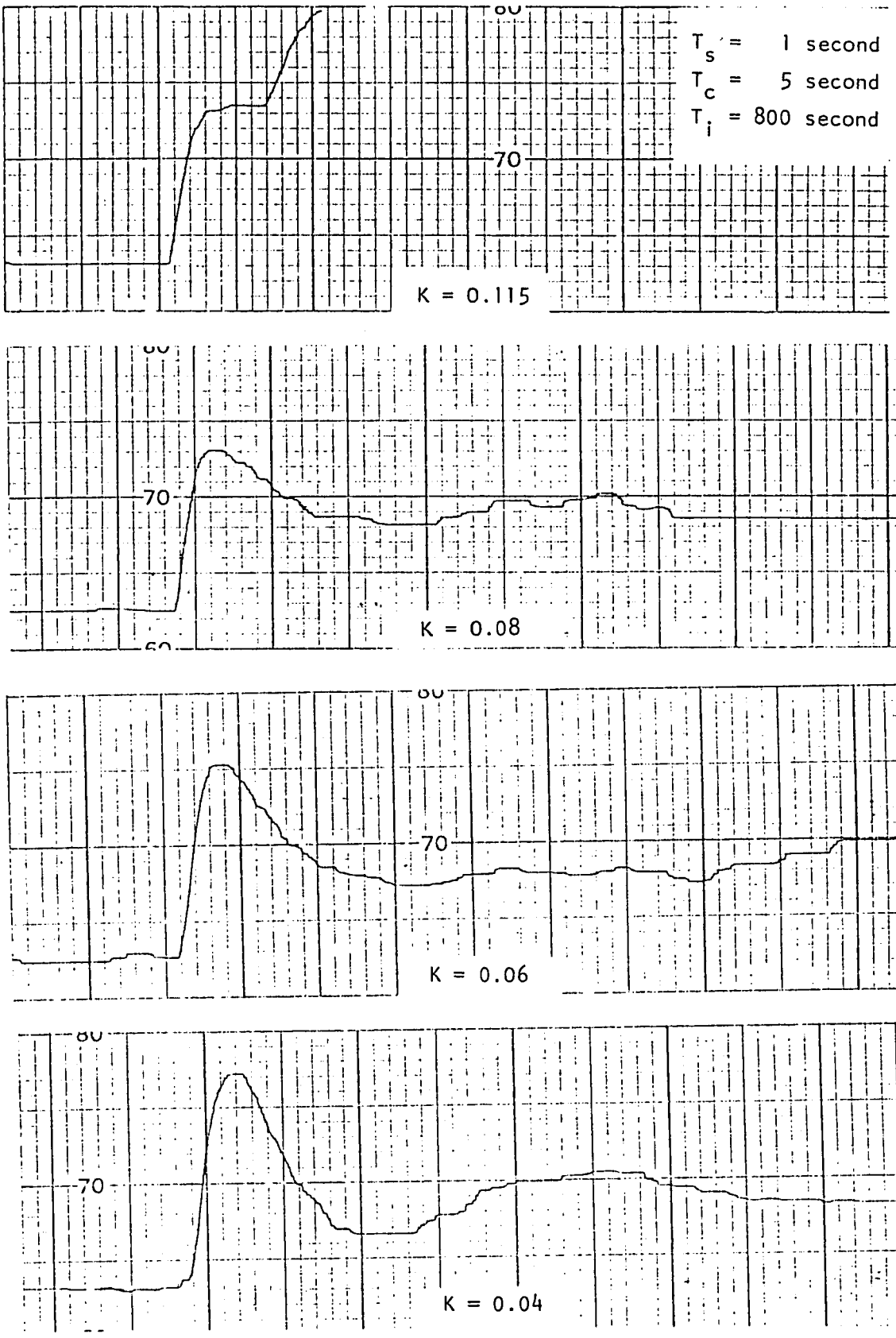


Figure 30 Effect of K in the Control System Stability

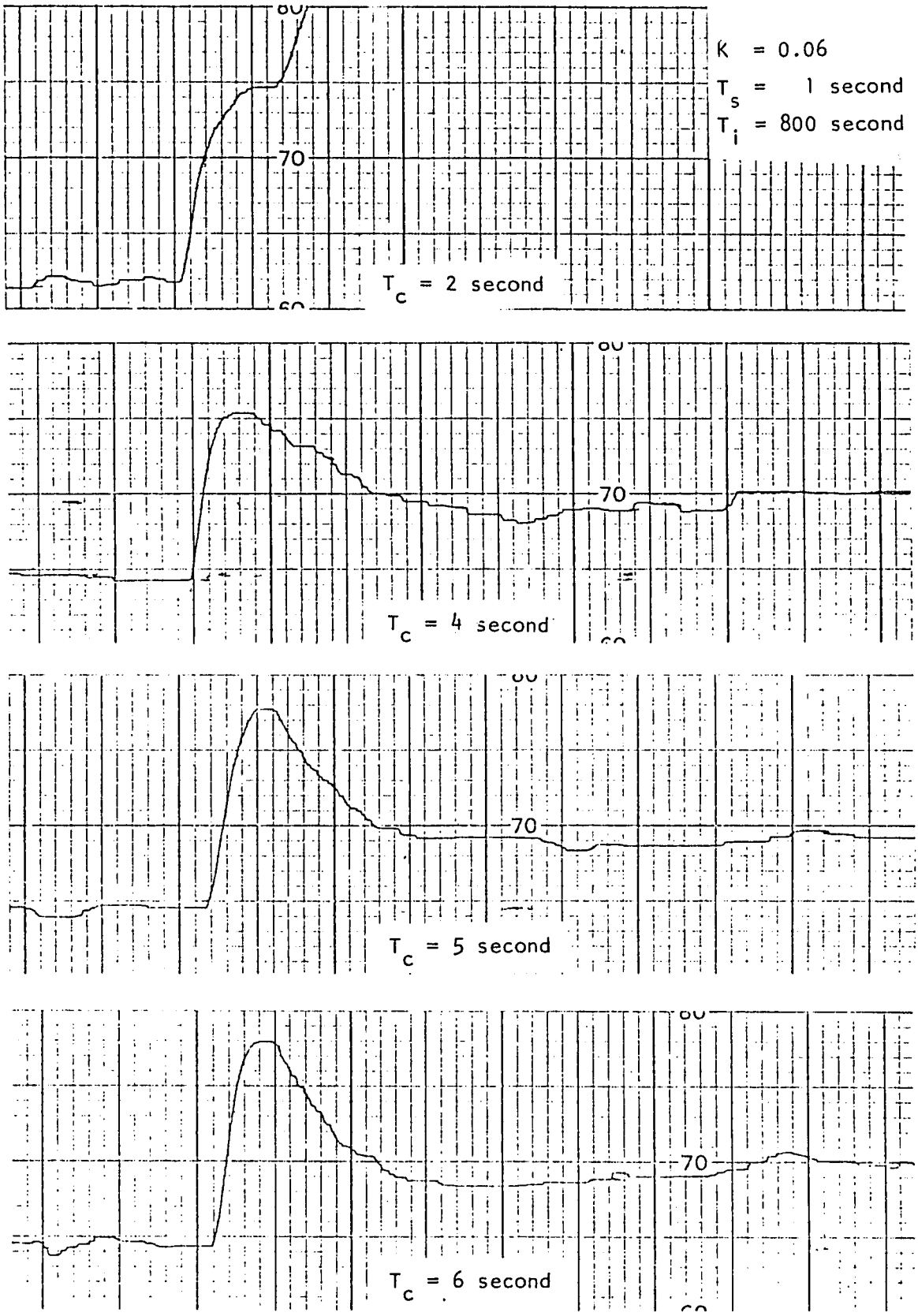


Figure 31 Effect of T_c in the Control System Stability

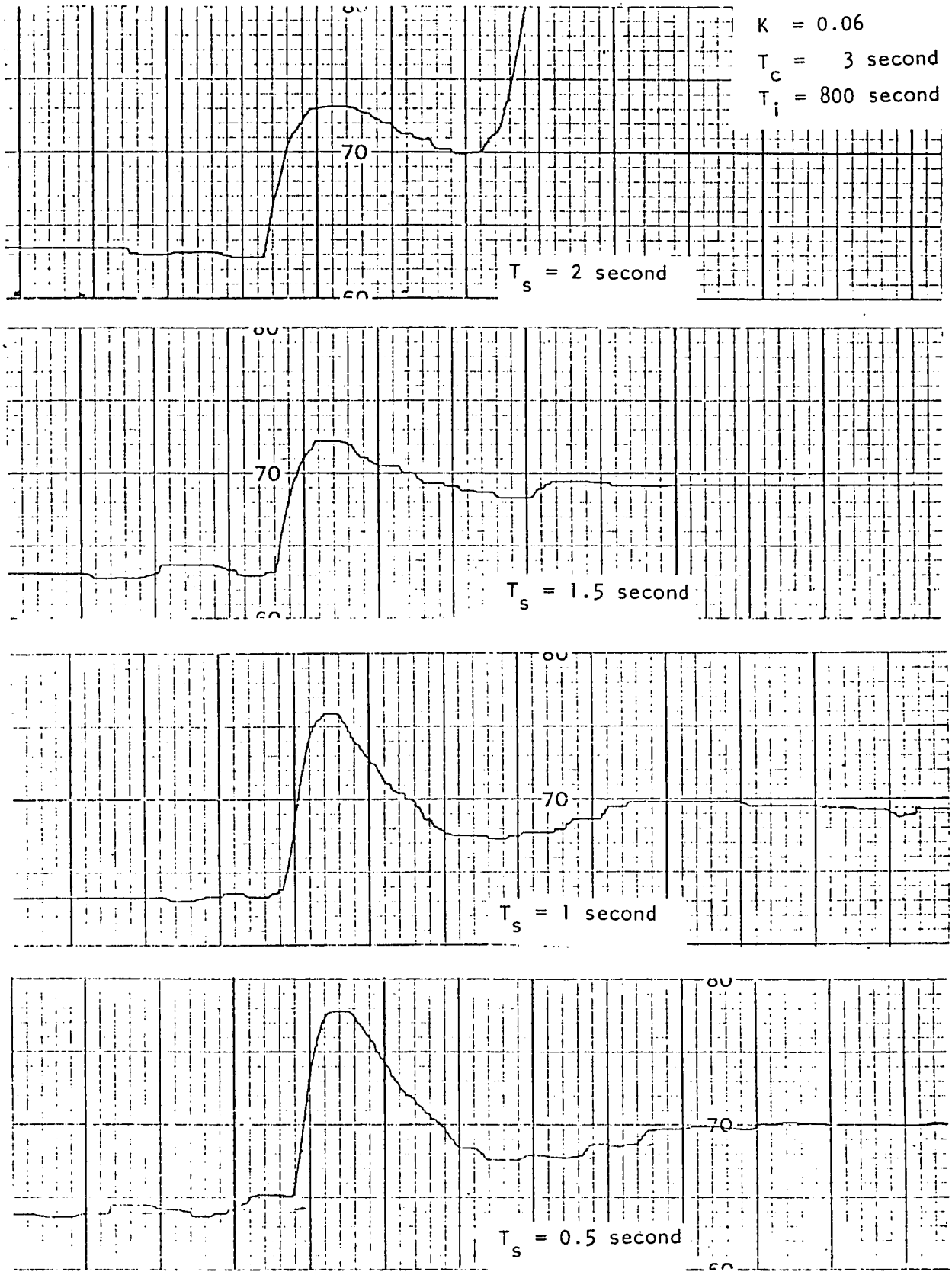


Figure 32 Effect of T_s in the Control System Stability

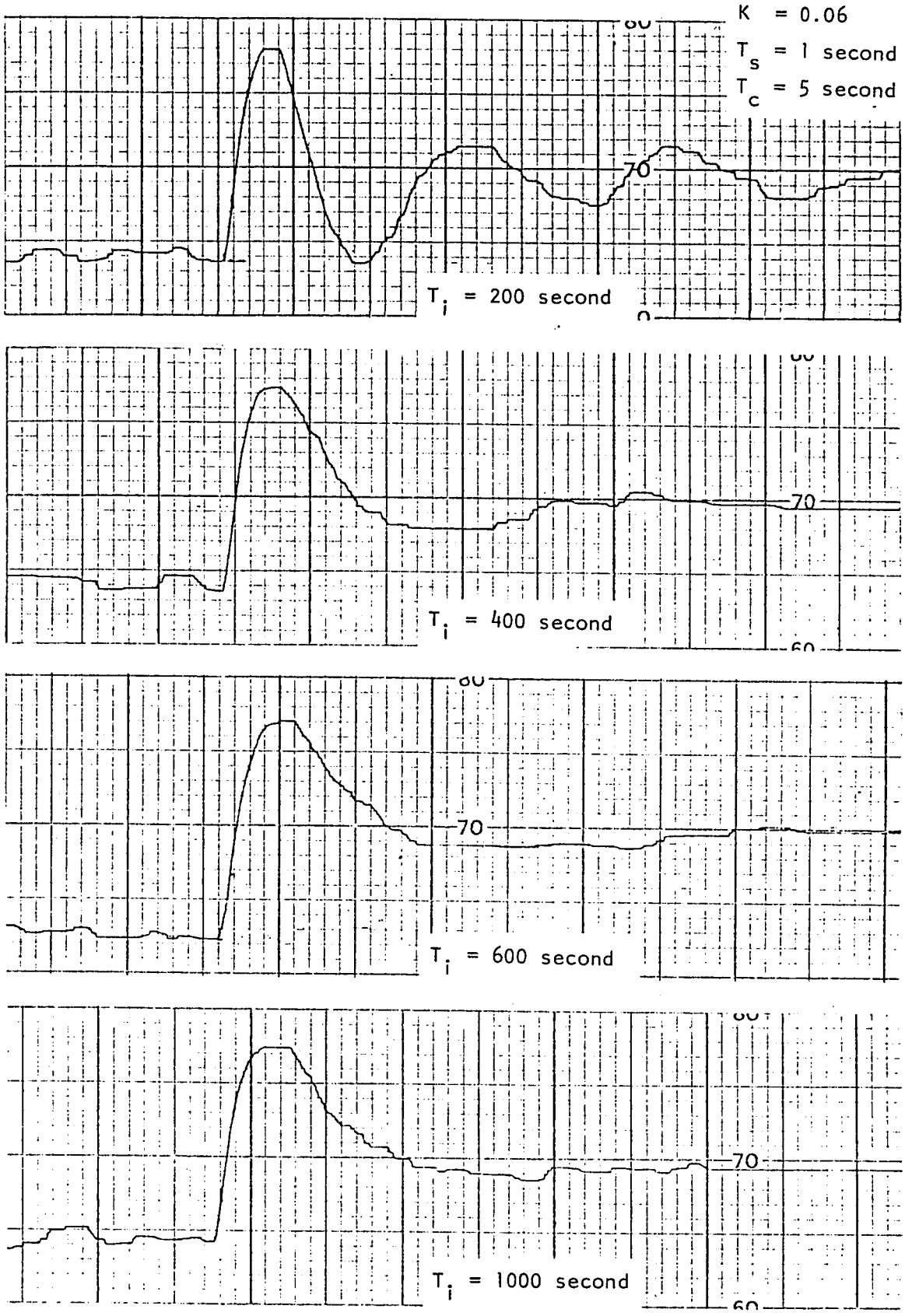


Figure 33 Effect of T_i in the Control System Stability

RESULTS AND DISCUSSIONS

1. The stability region for the analog computer simulation and the actual process are very close as shown in figure (29). Figures (27) - (29) also show that the smaller the proportional gain (K) and the larger the Multiplicative time constant (T_c), the more stable the system. This is true only if the value of the proportional gain (K) is greater than a lower limit (K_L). If the value of K goes below the lower limit K_L, the system would suddenly become unstable. This behavior is caused by a negative Multiplicative gain (A) and can be explained as follows :

$$A = \frac{K.G.r + G.u_1 + u_2 - r}{K.r^2 - K.G.r.u_1 - K.u_2.r} \quad \text{---------- (11)}$$

or

$$A = \left(G.r + \frac{G.u_1 + u_2 - r}{K} \right) . \frac{1}{r^2 - G.r.u_1 \quad u_2.r}$$

$$= A_1.A_2$$

Where

$$A_1 = G.r + \frac{G.u_1 + u_2 - r}{K}$$

$$A_2 = \frac{1}{r^2 - G.r.u_1 - u_2.r}$$

The ranges for each individual variables for this particular system are listed below.

r	=	120	- 160	°F
G	=	20	- 45	°F/psi
u_1	=	0.05	- 0.07	psi
u_2	=	30	- 36	°F

It is obvious that A_2 is always a positive number within the variable range. Then, A_1 is the only source causing the Multiplicative gain (A) to become negative. The value of K_L, the lower limit of stability, can be calculated by letting A_1=0.

$$A_1 = G.r + \frac{G.u_1 + u_2 - r}{K_L} = 0$$

$$K_L = \frac{r - G.u_1 - u_2}{G.r} \qquad \text{---------- (22)}$$

For different operating conditions k_L may vary, but the value can always be calculated from equation (22).

2) It is clearly shown in figure (29) that the longer the Multiplicative time constant, the more stable the system. This was rather expected.

3) Figures (31) - (34), the modified Multiplicative control transient responses show a very interesting result. It shows that when the controller is tuned towards the unstable region, the overshoot value decreases.

4) The software system includes all the elements of a basic digital computer control system - real time operating, operator communication, analog scan, data logging, control and input-output conversion. The analog alarm scan feature was not included because the system was designed for experimental purpose. The software system can be easily expanded to provide service to other experiments provided there are no hardware limitations.

CONCLUSIONS

1) This work has shown that the first order Multiplicative control algorithm does control the process properly provided that the integral feedback loop is added to compensate for the inaccuracy of the model. It is also recommended that for any model which can not accurately describe the process gain, the feedback loop is required for the Multiplicative control algorithm.

2) The Multiplicative control algorithm has both high and low stability limits for the controller gain constant K.

3) The stability region of the second order model compared with that of the real heat exchanger shows that even though the heat exchanger is a distributed parameter system, it was adequately represented by a simple second order system plus dead time.

REFERENCES

1) Williams, T.J. and Ryan, F.M. "Progress in Direct Digital Control" ISA 1969

2) Smith, C.L. "Digital Computer Process Control" Intex Educational Publishers, 1972

3) Auns, J. PhD thesis, University of Ottawa, Ottawa, Ontario, 1972

4) Lewis, J.B. Trans AIEE 71 part II 449, Jan 1953

5) Ku, Y.H. Trans ASME 79 1897 1959

6) Ku, Y.H. Trans AIEE 75 part II 402, Jan 1957

7) Jafri, M.N. PhD Thesis, University of Ottawa, Ottawa, Ontario 1967

8) Volterra, V. "Theory of Functionals and Integral and Integral-Differential equations", Dover Publication N.Y. 1959

9) Wiener, N. "Non-linear Problems in Random Theory" The Tech. Press, MIT and John Willy & sons, N.Y. 1958

10) Barret, J.F. "The use of functionals in the analysis of Nonlinear Physical Systems", J. of Electronics and Control, Vol 15 1963

11) Shinsky, F.G. "Process Control Systems", McGraw-Hill, 1967

12) Cadzow, J.A. and Martens, H.R. "Discrete time and Computer Control Systems" Prentice-Hall 1970

13) Luyben W.L. "Process Modeling, Simulation and Control for Chemical Engineers", McGraw-Hill, 1973

APPENDIX I DERIVATION OF THE DISCRETE MULTIPLICATIVE FUNCTION

This section covers the derivation of the difference equation from the transfer functions shown in figure (21).

Figure (21) shows the Multiplicative control system expressed in Laplace transform. The output of the controller X(s) outputs directly and continuously to the process. For a computer controlled process, the controller does not continuously output to the process. Instead, the output is a sequence of impulses. A hold device is always required in a computer control system to convert the sequence of impulses into a continuous staircase function. There are several types of mathematical hold functions. The one that is used in practically all real systems is called zero-order hold. The function of a zero-order hold is illustrated in figure (A1-1) and mathematically expressed as equation A1-1 (13).

$$H(s) = \frac{1 - \exp(-sT_s)}{s} \quad \text{---------- (A1-1)}$$

The zero-order hold device actually existed as part of the digital-to-analog converter. The same zero order hold function was added to the Multiplicative path to compensate for the digital-to-analog converter (see figure A1-3).

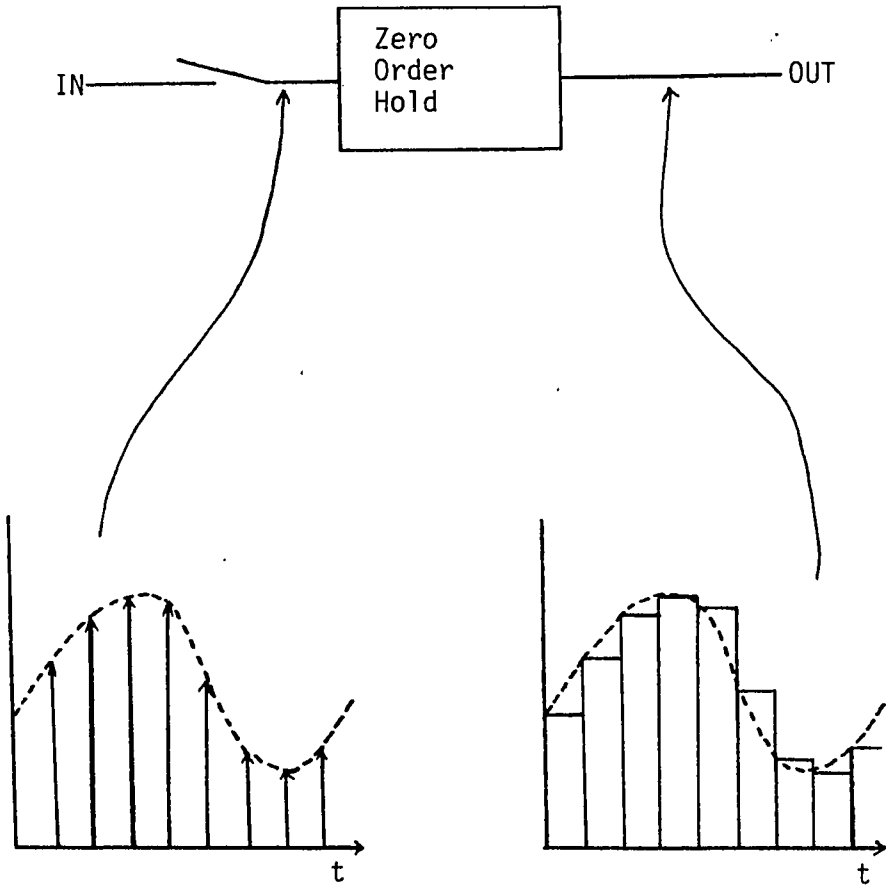


Figure A1-1 Zero-Order Hold

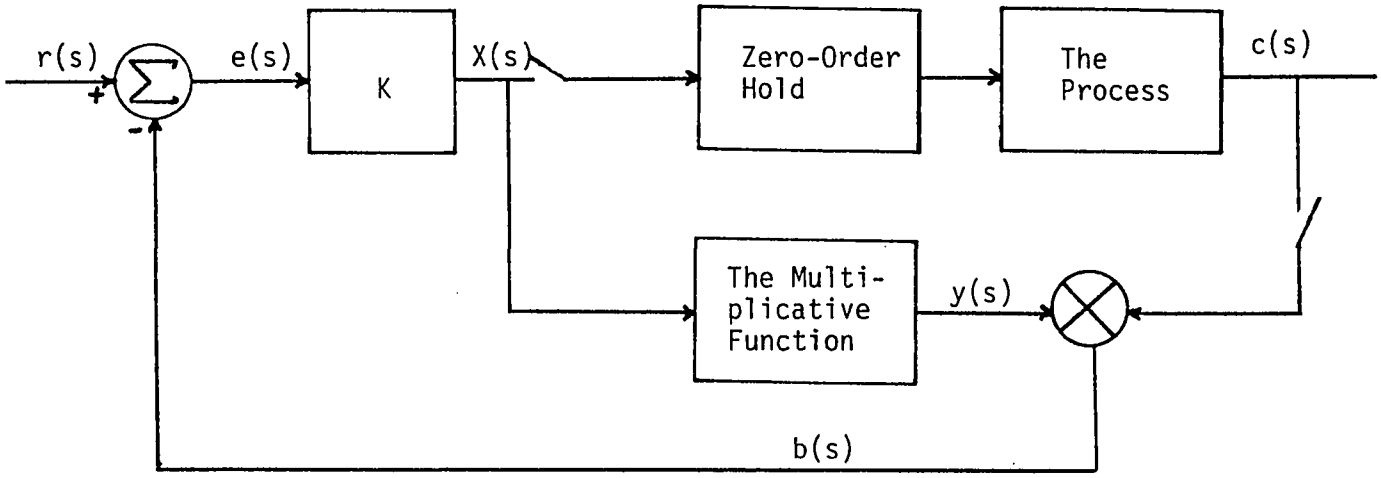


Figure A1-2 The System with Zero-Order Hold in the Output Path

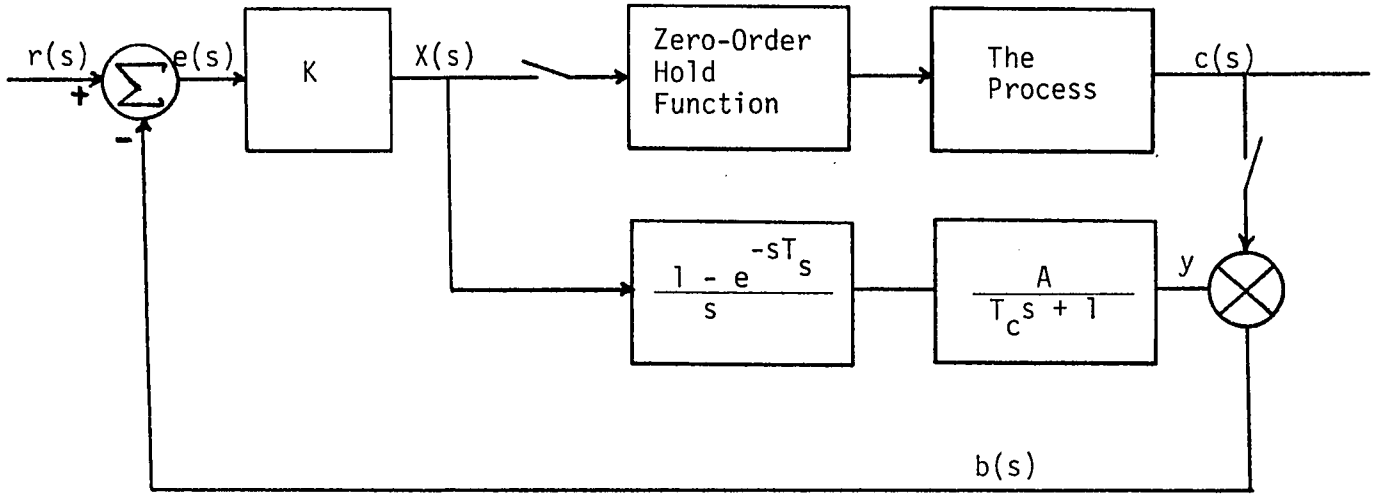


Figure A1-3 The System with Zero-Order Hold in both the Output and the Multiplicative Path

The Multiplicative function G_2 is now redefined as a combination with the zero-order hold function. The discrete time format is derived as follows:

$$G_2(s) = \frac{1 - \exp(-sT_s)}{s} \cdot \frac{A}{T_c s + 1}$$

$$= \frac{1 - \exp(-sT_s)}{s} \cdot \frac{A/T_c}{s + 1/T_c}$$

$$= \frac{A/T_c}{s(s + 1/T_c)} - \frac{A/T_c . \exp(-sT_s)}{s(s + 1/T_c)}$$

$$G_2(Z) = \mathcal{Z}\left(\frac{A/T_c}{s(s + 1/T_c)} \right) - \mathcal{Z}\left(\frac{A/T_c . \exp(-sT_s)}{s(s + 1/T_c)} \right)$$

Where $\mathcal{Z}$ is the Z-transformation of the function. Since the expression $\exp(-sT_s)$ in the second term is a unit delay, it can be replaced with a unit delay expressed in the Z-transformation form Z^{-1} as follows :

$$G_2(Z) = \mathcal{Z}\left(\frac{A/T_c}{s(s + 1/T_c)} \right) - Z^{-1} . \mathcal{Z}\left(\frac{A/T_c}{s(s + 1/T_c)} \right)$$

$$= (1 - Z^{-1}) . \mathcal{Z}\left(\frac{A/T_c}{s(s + 1/T_c)} \right)$$

or G_2 can be expressed as :

$$G_2(Z) = (1 - Z^{-1}).[\ \mathcal{Z}\left(\frac{A}{s}\right) - \mathcal{Z}\left(\frac{A}{s + 1/T_c}\right)]$$

To convert the Laplace transfer function into the Z-transformation, the conversion table compiled by C. Smith (2) was used. The following Z-transfer function was obtained as a result of the conversion.

$$G_2(Z) = (1 - Z^{-1}).\left(\frac{A}{1 - Z^{-1}} - \frac{A}{1 - \exp(-T/T_c).Z^{-1}}\right)$$

$$= A - \frac{A(1 - Z^{-1})}{1 - \exp(-T/T_c).Z^{-1}}$$

$$= \frac{A - A.\exp(-T/T_c).Z^{-1} - A + A.Z^{-1}}{1 - \exp(-T/T_c).Z^{-1}}$$

$$= \frac{A.Z^{-1} - A.\exp(-T/T_c).Z^{-1}}{1 - \exp(-T/T_c).Z^{-1}}$$

The variable T used in this derivation is the time interval for the discrete time system. Since the Multiplicative control algorithm

was calculated every time the sampling timer was due, the discrete time interval T can be replaced by the sampling time T_s which will give the following equation.

$$G_2(Z) = \frac{A.(1 - \exp(-T_s/T_c)).Z^{-1}}{1 - \exp(-T_s/T_c).Z^{-1}} \quad \text{---------- (A1-2)}$$

Equation (A2-1) is the transfer function in figure (23) for the Multiplicative control algorithm.

APPENDIX II DERIVATION OF THE FIRST ORDER DIGITAL FILTER

The digital filter used in the analog input program READR is a first order filter. The filter can be represented as a first order lag in Laplace transformation as shown in figure (A2-1) and mathematically as the following equation,

$$\frac{y(s)}{x(s)} = \frac{1}{Ts + 1} \quad , \quad \text{---------- (A2-1)}$$

where y(s) and x(s) are the Laplace transforms of the system input and output functions respectively. The first difference method (12) was used to convert equation (A2-1) into digital format. The differential equation governing this system is found by first multipling both sides of equation (A2-1) by (Ts+1).x(s), which gives

$$y(s).(Ts + 1) = x(s)$$

and by noting that the inverse transform of s.y(s) is dy(t)/dt (with the initial condition to be zero). This results in

$$T.\frac{dy(t)}{dt} + y(t) = x(t) \quad \text{---------- (A2-2)}$$

It is desired to simulate the dynamic characteristics of this

differential equation, which in turn, necessitates the determination of the first derivative of the function y(t). Since the simulation is to be carried by means of a digital computer, it is natural to solve equation (A2-2) at computation time that is taken to be spaced at T_s second intervals, that is

$$T.\left.\frac{dy(t)}{dt}\right|_{t=N} + \left.y(t)\right|_{t=N} = \left.x(t)\right|_{t=N} \quad \text{---------- (A2-3)}$$

In order to determine $\left.\frac{dy(t)}{dt}\right|_{t=N}$, we shall employ the following differential approximation.

$$\dot{y}(N) = \left.\frac{dy(t)}{dt}\right|_{t=N} \doteq \frac{y(N) - y(N-1)}{T_s}$$

Where N is the number of intervals (starting at time equal to zero) in discrete time. T_s is the sampling time (the length of the discrete time interval). Substitute the above approximation into equation (A2-3), the following expression can be obtained.

$$T.\frac{y(N) - y(N-1)}{T_s} + y(N) = x(N)$$

$$\frac{T}{T_s}.y(N) - \frac{T}{T_s}.y(N-1) + y(N) = x(N)$$

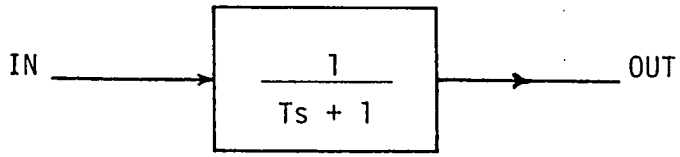


Figure A2-1 The First Order Filter

$$\left(\frac{T}{T_s} + 1\right).y(N) = \frac{T}{T_s}.y(N-1) + x(N)$$

$$y(N) = \frac{(T/T_s).y(N-1) + x(N)}{(T/T_s) + 1}$$

$$y(N) = \frac{(T/T_s).y(N-1) + (T_s/T_s).x(N)}{(T + T_s)/T_s}$$

$$y(N) = \frac{T.y(N-1) + T_s.x(N)}{T + T_s}$$

$$y(N) = \frac{T}{T + T_s}.y(N-1) + \frac{T_s}{T + T_s}.x(N) \quad \text{----------(A2-4)}$$

Equation (A2-4) is the first order filter used in program READR to eliminate noise in the input signal. It is a first order lag with a time constant T, or in other words, with a corner frequency of 1/T. Any noise with a frequency higher than 1/T will be filtered.

In practice, the filter can be viewed as a kind of rolling average type of filter. The weighting factor between the last filtered value y(N-1) and the current reading x(N) depends on the time constant T and the sampling time T_s respectively. Time constant T is a value entered by operator. The larger the time constant T, the more weight will be added to the last filtered value y(N-1). The longer the sampling time T_s, the more weight will be added to the current value x(N).

APPENDIX III SOFTWARE PROGRAM LISTINGS

SOFTWARE PROGRAM LISTINGS

```
        ;THE INTERRUPT SERVICE PROGRAM


        .TITL   INTRP
        .ENT  TABLE, INTRP, AC3,   CARRY, SATAM
        .ENT  PCBCK, TTOCN, OUTPT, CONON, TTOSV
        .EXTN READ1, FENT,  H2OTM, RESTR, CNTLS
        .EXTN ALPAR, SAMMT, CRERO
        .ZREL

TABLE:  PWFSV   ; 0
        NODVC   ; 1
        NODVC   ; 2
        NODVC   ; 3
        NODVC   ; 4
        NODVC   ; 5
        NODVC   ; 6
        NODVC   ; 7
        TTISV   ;10 TTI SERVICE ROUTINE
        NODVC   ;11 TTO SHOULD NOT INTERRUPT
        NODVC   ;12
        PTPSV   ;13 PTP SERVICE ROUTINE
        RTCSV   ;14 RTC SERVICE ROUTINE

        .NREL

INTRP:  SKPDZ CPU       ;IS POWER FAILURE INTERRUPT?
        JMP  @TABLE     ;GOTO SERVE PWF
        STA  3,AC3      ;STORE AC3
        MOVL 3,3        ;SAVE CARRY
        STA  3,CARRY
        SKPDN RTC       ;IS RTC INTERRUPT?
        JMP  NOTHI      ;NO, GOTO LOW PRIORITY SERVICE
RTCSV:  NIOS RTC        ;REINITIALIZE REAL TIME CLOCK
        DSZ  SATIM      ;DECREASE TIME COUNTER
        JMP  WAITA      ;COUNTER NOT ZERO, WAIT
        LDA  3,SATAM    ;COUNTER ZERO, RESET IT
        STA  3,SATIM
        JMP  @.READ     ;GOTO DATA READING ROUTINE

WAITA:  LDA  3,CARRY    ;RESTORE CARRY
        MOVR 3,3
        LDA  3,AC3      ;RESTORE AC3
        NIOS CPU        ;TURN ON INTERRUPT AGAIN
        JMP  @0                 ;RETURN TO MAIN PROGRAM

NOTHI:  INTA 3          ;INTERRUPT ACKNOWLEDGE
        JMP  @TABLE,3   ;GOTO PROPER SERVICE ROUTINE
```

```
AC3:    3
SATIN:  10
SATAM:  10
.READ:  READ1
CARRY:  0


        ;INTERRUPT DEVICE ERROR DETECTING ROUTINE

NODVC:  MOV   3,1           ;SAVE ERROR DEVICE CODE IN AC1
        LDA   2,.MESS       ;SEND DEVICE ERROR MESSAGE
        JSR   @4                    ;GOTO OUTPUT ROUTINE
        HALT                ;STOP THE SYSTEM
        JMP   .-1           ;DON'T CONTINUE

.MES :  MESSA
```

```
        ;POWER FAILURE SERVICE ROUTINE

PWFSV:  INTDS               ;DISMISS INTERRUPT
        STA   0,PFAC0       ;SAVE AC0
        STA   1,PFAC1       ;SAVE AC1
        STA   2,PFAC2       ;SAVE AC2
        STA   3,PFAC3       ;SAVE AC3
        MOVL  3,3           ;SAVE CARRY
        STA   3,PFCRY
        LDA   0,0           ;SAVE ADDRESS 0
        STA   0,PFAD0
        LDA   0,.PRST       ;SET RESTART ADDRESS
        STA   0,0
        HALT                ;STOP THE SYSTEM
        JMP   @0                    ;RESTART

PFCRY:  0
PFAC0:  0
PFAC1:  0
PFAC2:  0
PFAC3:  0
PFAD0:  0
.PRST:  PFRST

PFRST:  LDA   0,PFAD0       ;RESTORE ADDRESS0
        STA   0,0
        LDA   3,PFCRY       ;RESTORE CARRY
        MOVR  0,3
        LDA   0,PFAC0
        LDA   1,PFAC1
        LDA   2,PFAC2
        LDA   3,PFAC3
        INTEN               ;TURN ON INTERRUPT
        JMP   @0                    ;RETURN TO NORMAL OPERATION
```

```
        ;PTP INTERRUPT SERVICE ROUTINE
        ;PUNCHES 4 DATA FRAMES AT A TIME


        ;THE PUNCH CONTROL BLOCK

PCBCK:  0           ; 0   NOTHI   NO USE
        0           ; 1   PPCRY   SAVE CARRY
        0           ; 2   PPADD   SAVE RETURN ADDRESS
        0           ; 3   PPAC0   SAVE AC0
        0           ; 4   PPAC1   SAVE AC1
        0           ; 5   PPAC2   SAVE AC2
        0           ; 6   PPAC3   SAVE AC3
        0           ; 7   FIRTM   CYCLE START INDICATOR
        4           ;10   PHCNT   DATA COUNTER
        0           ;11   PDATA   PUNCH DATA BUFFER
        0           ;12   BYTEP   BYTE POINTER
        20          ;13   PADDR   PUNCH DATA ADDRESS POINTER
        4           ;14   PHCST   PUNCH DATA COUNTER CONSTANT
        CARRY       ;15   .CARY   INTERRUPT CARRY SAVE ADD
        AC3         ;16   .AC3    INTERRUPT AC3 SAVE ADD

P=PCBCK
```

```
        ;THE SERVICE ROUTINE

PTPSV:  LDA   3,@P+15     ;SAVE CARRY
        STA   3,P+1
        LDA   3,@P+16     ;SAVE AC3
        STA   3,P+6
        STA   0,P+3       ;SAVE AC0
        STA   1,P+4       ;SAVE AC1
        STA   2,P+5       ;SAVE AC2
        LDA   0,0         ;SAVE RETURN ADD
        STA   0,P+2
        NIOC  PTP         ;IDLE PTP
        INTEN             ;TURN ON INTERRUPT AGAIN
        LDA   0,P+7       ;CHECK FOR THE BEGINNING OF
        MOV   0,0,SNR     ;  PUNCHING CYCLE
        JSR   @CYCL       ;YES, GOTO RESTART THE CYCLE
        LDA   0,P+11      ;GET THE PUNCH DATA
        MOV   0,0,SNR     ;CHECK IF DATA IS ZERO
        JMP   PRETN       ;DISMISS THE INTERRUPT
        LDA   1,P+12      ;GET THE BYTE POINTER
        DOAS  0,PTP       ;OUTPUT THE DATA
        MOV   1,1,SZR     ;CHECK THE BYTE POINTER
        JMP   .+6         ;POINTER ZERO
        MOVS  0,0         ;SHIFT THE DATA
        STA   0,P+11      ;RESTORE THE DATA
        SUBZL 1,1         ;CLEAR BYTE POINTER
        STA   1,P+12
        JMP   .+3
        JSR   @CYCL       ;BYTE POINTER IS ZERO
        DSZ   P+13        ;DECREASE DATA POINTER
        JMP   PRETN       ;COUNTER NOT ZERO, RETURN
        LDA   1,P+14      ;COUNTER ZERO, RESET IT!
        STA   1,P+10
        SUB   0,0         ;CLEAR DATA BUFFER
        STA   0,P+11
PRETN:  LDA   0,P+1       ;RESTORE CARRY
        MOVR  3,3
        LDA   0,P+3       ;RESTORE AC0
        LDA   1,P+4       ;RESTORE AC1
        LDA   2,P+5       ;RESTORE AC2
        LDA   3,P+6       ;RESTORE AC3
        JMP   @P+2        ;RETURN

@CYCL:  LDA   2,P+13      ;GET DATA POINTER
        LDA   0,0,2       ;GET THE DATA
        INC   2,2         ;INCREASE THE POINTER
        STA   2,P+13      ;RESTORE THE POINTER
        STA   0,P+11      ;SET UP DATA BUFFER
        SUB   2,2
        STA   2,P+12      ;CLEAR BYTE POINTER
        INC   2,2
        STA   2,P+7
        JMP   0,3         ;RETURN
```

```
;TTI INTERRUPT SERVICE
;
;COMMANDS :  S        SET POINT STEP CHANGE
;           R        RESTART AT EXEC
;           RD       RESTART AT DEBUG
;           RA(ADD)RESTART AT ADDRESS
;           RL       LIST ALL PARAMETERS
;           RSA      RESET ALL PARAMETERS
;           RSC      RESET ALL CONTROL PARAMETERS


TTISV:  INTDS             ;DISMISS INTERRUPT
        LDA  0,CRLF       ;SEND CRLF
        JSR  @5
        JSR  @40          ;INPUT ONE CHARACTER
        LDA  1,S          ;TO CHECK FOR SET POINT INTERRUPT
        SUB# 0,1,SNR      ;FIRST
        JMP  STPTC        ;GOTO SET POINT UNIT STEP CHANGE
                          ;INTERRUPT ROUTINE
        LDA  1,R          ;NO, CHECK FOR CHAR R
        SUB# 0,1,SZR      ;IS IT "R" ?
        JMP  EROR1        ;NO, GOTO ERROR ROUTINE
        JSR  @40          ;YES, INPUT ANOTHER CHARACTER
        LDA  1,CR
        SUB# 0,1,SNR      ;IS IT "CR" ?
        JMP  2            ;YES, RESTART AT EXEC
        LDA  1,L          ;CHECK FOR CHAR L
        SUB# 0,1,SNR
        JMP  LISTR        ;YES, GOTO LIST ROUTINE
        LDA  1,S          ;NO, IS IT S?
        SUB# 0,1,SZR
        JMP  .+11         ;NO, GOTO TO TEST OTHERS
        JSR  @40          ;YES, INPUT ANOTHER CHARACTER
        LDA  1,A          ;IF IT IS A?
        SUB# 0,1,SNR
        JMP  2            ;ALL PARAMETERS RESET, RESTART!
        LDA  1,C          ;IS IT C?
        SUB# 0,1,SNR
        JMP  CONRS        ;ONLY MODIFY CONTROL PARAMETERS
        JMP EROR1         ;INPUT ERROR!
        LDA  1,D
        SUB# 0,1,SNR      ;NO, IS IT "D" ?
        JMP  3            ;YES, RESTART AT DEBUG
        LDA  1,A
        SUB# 0,1,SZR      ;NO, IS IT "A" ?
        JMP  EROR1        ;NO, GOTO ERROR ROUTINE
        JSR  @..BIN       ;YES, ENTER STARTING ADDRESS
        STA  1,ADDRS      ;STORE THE ADDRESS
        LDA  1,CR         ;GET CR
        SUB# 0,1,SZR      ;IS BREAK CHARACTER CR?
        JMP  EROR1        ;NO, GOTO ERROR ROUTINE
        JMP  @ADDRS       ;GOTO THE STARTING ADDRESS
```

```
ERROR1:  LDA   2,.MES7     ;SEND ERROR MESSAGE
         JSR   @4
         LDA   0,CRLF      ;SEND CRLF
         JSR   @5
         JMP   TTISV       ;WAIT FOR COMMAND AGAIN


CONRS:   LDA   1,CONON     ;TEST IF CONTROL IS ACTIVE
         MOV   1,1,SNR     ;IS INDICATOR SET?
         JMP   NOCTL       ;NO, DO NOT MODIFY
         JSR   @.ALPA      ;GOTO CONTROL PARAMETER MODIFICATION
         JMP   NOCTL-3     ;WAIT FOR COMMAND

.ALPA:   ALPAR
```

```
; CONVERT AN ASCII OCTAL CHARACTER STRING TO A BINARY
; NUMBER

; INPUT:          CALLS A GET CHARACTER ROUTINE WHOSE
;                 ADDRESS MUST BE STORED
;                 IN LOCATION 40 OF PAGE 0.
;                 CHARACTERS MUST BE RETURNED,
;                 RIGHT ADJUSTED IN AC0 WITH BIT 8=0

;                 INPUT OF FORM:
;                         00...0(BREAK)
;                 WHERE "0" REPRESNETS AN OCTAL DIGIT AND
;                 BREAK IS ANY OTHER CHARACTER

; OUTPUT:         AC0 CONTAINS THE BREAK CHARACTER
;                 AC1 CONTAINS THE BINARY NUMBER (MOD
;                 200000 OCTAL)

; CALLING SEQUENCE:
;       JSR       .OBIN
;       RETURN

; IF AN INDICATION IS DESIRED TO SIGNAL CHARACTERS ARE
; REQUESTED, CALLING SEQUENCE:
;       JSR       .OBNI
;       RETURN

; AN ASCII "0" FOLLOWED BY A NULL CHARACTER
; WILL BE TRANSMITTED VIA AC0
; TO USER PUT CHARACTER ROUTINE WHOSE
; ADDRESS MUST BE STORED IN LOCATION 41 OF PAGE 0

; CAUTION:        RESULT IS N MOD 200000 (OCTAL)
;                 E.G. 576452* CONVERTS TO 176452

; DESTROYED:      AC0, AC1, AC3, CARRY
; UNCHANGED:      AC2


.OBNI:  STA 3,.EE03       ; SAVE RETURN
        STA 2,.EE02       ; SAVE AC2
        LDA 0,.EE22
        JSR @.EE41        ; SEND "0"
        SUB 0,0
        JSR @.EE41        ; SEND NULL
        JMP .+3
.OBIN:  STA 3,.EE03       ; SAVE RETURN
        STA 2,.EE02       ; SAVE AC2
        SUB 1,1                   ; CLEAR RESULT WORD
        STA 1,.EE10
```

```
.EE99:  JSR @.EE40      ; GET A DIGIT
        LDA 2,.EE20     ; OCTAL 60
        LDA 3,.EE21     ; OCTAL 67
        ADCZ# 3,0,SNC   ; TEST FOR 60 <=N<= 67
        ADCZ# 0,2,SZC
        JMP .EE99       ; NO - MUST BE BREAK CHARACTER
        SUB 2,0                 ; PUT N IN RANGE 0-7
        LDA 1,.EE10
        MOVZL 1,1       ; SHIFT SUM
        MOVZL 1,1
        MOVZL 1,1
        ADD 0,1
        STA 1,.EE10
        JMP .EE98       ; LOOP TILL BREAK RECEIVED

.EE99:  LDA 2,.EE02     ; RESTORE AC2
        LDA 1,.EE10     ; ANSWER TO AC1
        JMP @.EE03      ; AND RETURN

.EE02:  0               ; SAVE AC2
.EE03:  0               ; SAVE RETURN

.EE10:  0               ; STORAGE FOR RESULTS

.EE20:  60              ; ASCII "0"
.EE21:  67              ; ASCII "7"
.EE22:  "0              ; ASCII "0"

.EE40=40                ; PAGE 0 ADDRESS OF GET A
                        ; CHARRACTER ROUTINE
.EE41=41                ; PAGE 0 ADDRESS OF PUT A
                        ; CHARACTER ROUTINE
```

```
;SET POINT STEP CHANGE SUBROUTINE:
;
;    THIS ROUTINE WILL GIVE A STEP CHANGE OF 5 DEG F
;    TO THE TEMPERATURE SET POINT.  IF ANY OTHER
;    VALUE ARE ASSIGNED, ENTER THE FLOATING FORMAT
;    OF YOUR STEP CHANGE AT LOCATIONS  "STEPC" AND
;    "STEPC+1".
;
;      AFTER THE SET POINT CHANGED, THE WHOLE SYSTEM
;      WILL BE RESTARTED AT  "RESTR"  OF THE EXECUTIVE
;      PROGRAM.

STPTC:  LDA  2,.MS11      ;SEND MESSAGE
        JSR  @4
        FENT
        FLDA 0,@.H2OT     ;GET TEMPERATURE SET POINT
        FFDCF 0                   ;OUTPUT TEMP SET POINT
        FLDA 1,STEPC      ;GET THE STEP CHANGE
        FADD 1,0          ;SET STEP CHANGE
        FSTA 0,@.H2OT     ;RESTORE TEMP SET POINT
        FEXT
        LDA  2,.MS12
        JSR  @4
        FENT
        FLDA 0,@.H2OT
        FFDCF 0                   ;OUTPUT THE NEW SET POINT
        FEXT
        LDA  2,.MS21
        JSR  @4
        SUB  0,0          ;SEND BELL ON PAPER TAPE
        JSR  @4?          ;FOR SEPARATION
        DSZ  COUNT        ;DECREASE DATA COUNT
        JMP  .-2
        LDA  0,CR
        STA  0,COUNT      ;RESET COUNTER
        JMP  @.RESR       ;GOTO RESTART THE SYSTEM

.MES?:  MESS?
.MS21:  MES21
.MS11:  MES11
.MS12:  MES12
.RESR:  RESTR
COUNT:  15
STEPC:  0.5E+1
CRLF:   5015      ;"CR AND "LF
R:      "R
D:      "D
A:      "A
S:      "S
L:      "L
C:      "C
CR:     15
ADDRS:  0
..BIN:  .OBIN
```

```
;LIST OF ALL VARIABLE PARAMETERS IN THE SYSTEM
;THE PARAMETERS :
;       SAMTM:    SAMPLING TIME (SEC)
;       H2OTM:    TEMPERATURE SET POINT (DEG F)
;       CONFQ:    PERIOD FOR DIGITAL FILTER CORNER FREQUENCY
;
;       AND OTHER CONTROL PARAMETERS


LISTR:  LDA   2,.MS13
        JSR   @4
        LDA   2,.SAMT       ;GET SAMPLING TIME POINTER
        JSR   OUTPT         ;OUTPUT SAMPLING TIME
        LDA   2,.MS14
        JSR   @4
        LDA   2,.H2OT       ;GET TEMP SET POINT POINTER
        JSR   OUTPT
        LDA   2,.MS15
        JSR   @4
        LDA   2,.CNFQ       ;GET CORNER FREQ POINTER
        JSR   OUTPT
        LDA   2,.MS16
        JSR   @4
        LDA   2,CONON       ;TEST IF CONTROL IS ACTIVE
        MOV   2,2,SNR
        JMP   NOCTL         ;NO CONTROL
        LDA   2,.MS17
        JSR   @4
        JSR   @.CTLS        ;GOTO OUTPUT CONTROL PARAMETERS
        LDA   2,.MS19
        JSR   @4
        JMP   TTISV+1
NOCTL:  LDA   2,.MS18
        JSR   @4
        JMP   .-5

OUTPT:  STA   3,BACKW       ;RETURN ADDRESS
        FENT
        FLDA 0,0,2          ;GET THE DATA
        FFDCF 0                     ;OUTPUT THE DATA
        FEXT
        JMP   @BACKW        ;RETURN

.MS13:  MES13
.MS14:  MES14
.MS15:  MES15
.MS16:  MES16
.MS17:  MES17
.MS18:  MES18
.MS19:  MES19
.SAMT:  SAMMT
.H2OT:  H2OTM
.CNFQ:  CRFRQ
.CTLS:  CNTLS
CONON:  0
BACKW:  0
```

```
        ;TTO OUTPUT ROUTINE

MASK1:  177771          ;PTC,PTP AND TTI CAN INTERRUPT
MASK2:  177773          ;ONLY RTC AND PTP CAN INTERRUPT


TTOSV:  LDA   2,.MES9   ;OUTPUT HEADING
        JSR   @4
        LDA   2,TTOCN   ;CHECK FOR THE DATA READING
        MOV   2,2,SNR   ;IS THE INDICATOR ZERO?
        JMP   .-2       ;YES, DATA NOT READY, WAIT!
        SUB   2,2       ;GET DATA POINTER
        JSR   ..FDC     ;OUTPUT THE DATA
        LDA   0,SPSP    ;SEND SPACES
        JSR   @5
        INC   2,2       ;INCREASE POINTER
        INC   2,2
        JSR   ..FDC     ;OUTPUT ANOTHER DATA
        LDA   0,.CRLF   ;OUTPUT CR AND LF
        JSR   @5
        SUB   0,0       ;RESET DATA READY INDICATOR
        STA   0,TTOCN
        JMP   TTOSV+2


.MS10:  MES10
.MES9:  MESS9
DADDS:  13000
RETRN:  0
.CRLF:  5015
SPSP:   20040
TTOCN:  0


..FDC:  STA   3,RETRN   ;SAVE RETURN
        INTDS
        LDA   3,MASK2
        DOBS  3,CPU
        FENT
        FLDA  0,20,2    ;GET THE DATA
        FFDCF 0                 ;OUTPUT THE DATA
        FEXT
        INTDS
        LDA   3,MASK1
        DOBS  3,CPU
        JMP   @RETRN    ;RETURN
```

```
MESS7:  .TXT /<15><12><12><12>COMMAND ERROR, TRY AGAIN!<15><12><12>/
MESS8:  .TXT /<15><12><12><12>DEVICE INTERRUPT ERROR.<15><12>
DEVICE CODE IN AC1; CHECK YOUR SYSTEM!!<15><12><12>/
MESS9:  .TXT /<15><12><12>INLET TEMP  OUTLET TEMP<15><12>/
MES10:  .TXT /<15><12><12>ANOTHER SET OF DATAS<15><12><12>/



MES11:  .TXT /<15><12><12>H2OT# : /
MES12:  .TXT / F  TO /

MES13:  .TXT /<15><12><12>  SAMT#:  /
MES14:  .TXT /  SECOND<15><12>  H2OT#:  /
MES15:  .TXT /  DEG F <15><12>  CONFG:  /
MES16:  .TXT /  SECOND<15><12>/
MES17:  .TXT /<15><12>CONTROL ACTION IS ACTIVE. THE PARAMETERS ARE
MES18:  .TXT /<15><12>CONTROL IS INACTIVE/
MES19:  .TXT /<15><12>INPUT YOUR COMMAND PLEASE<15><12>/
MES21:  .TXT / F<15><12>/



        .END
```

```
        ;I/O DEVICE HANDLERS

        .TITL IOHDLR

        .NRFL



IOXOS:  STA   3,..SAV
        LDA   0,0,2         ;OUTPUT THE CHARACTER
        JSR   @41
        LDA   1,MASKF       ;MASK THE LEFT MOST 9 BITS
        MOVS  0,0
        AND#  1,0,SNR       ;CHECK IF THE CHARACTER IS NULL
        JMP   @..SAV        ;YES, OUTPUT COMPLETE. RETURN
        JSR   @41           ; NO, OUTPUT THE CHARACTER
        INC   2,2           ; INCREASE DATA POINTER
        JMP   IOXOS+1       ; DO THE NEXT WORD

IOXIS:  STA   3,..SAV
        LDA   1,CR          ;GET CR
        JSR   @40           ;INPUT THE CHARACTER
        SUB#  0,1,SZR       ;IS THE CHARACTER CR?
        JMP   .-2           ;NO,INPUT THE NEXT ONE
        LDA   0,LF          ;YES, SENT LF
        JSR   @41
        JMP   @..SAV        ;RETURN TO MAIN PROGRAM

IOXOZ:  STA   3,..SAV
        JSR   @41           ;OUTPUT IT
        MOVS  0,0           ;SHIFT TO LEFT 8 MOST BITS
        JSR   @41           ;OUTPUT
        JMP   @..SAV        ;RETURN


MASKF:  177
..SAV:  0
CR:     15
LF:     12

PTPO:   SKPBZ PTP           ;IF THE PTP IS BUSY
        JMP   .-1
        DOAS  0,PTP         ;OUTPUT THE DATA
        JMP   @3
```

```
OPT:    MOV  0,0,SNR     ;CHECK IF AC0 IS ZERO
        JMP  0,3         ;YES, RETURN!
        SKPBZ TTO        ;NO, CHECK IF TTO IS BUSY
        JMP  .-1
        DOAS 0,TTO       ;NO, OUTPUT TO TTO
        JMP  0,3         ;RETURN

IPT:    STA  3,SAVER     ;SAVE RETURN
        SKPDN TTI        ;SEE IF TTI IS BUSY
        JMP  .-1         ;YES, WAIT!
        DIAC 0,TTI       ;NO, INPUT THE DATA
        JSR  OPT         ;ECHO ON TTY
        LDA  3,MASK      ;MASK TO 7 BITS
        AND  3,0
        JMP  @SAVER      ;RETURN
SAVER:  0
MASK:   177

        .END
```

```
        ;THE DATA TAKING ROUTINE
        ;READS TWO READINGS FROM ADCV CHANNELS 0 AND 1
        ;


        .TITL READR
        .EXTN ITMP,  CONTL, CARRY, FENT,  FINT,
        .EXTN PCBCK,  AC3 , TTOCN
        .ENT  SAMMT, CREAD, GOCNT, VOLT0
        .ENT  SANMT, READ1, INCNT

        .ZREL

ADRED:  .BLK 2              ;READING TEMPORARY STORAGE
VOLT0:  .BLK 2              ;FIRST READING FPP
VOLT1:  .BLK 2              ;SECOND READING FPP
CNALM:  CONTL               ;CONTROL ROUTINE ADDRESS
PREAD:  0.54E+2                     ;LAST READING CHANNEL 0
        0.54E+2                     ;LAST READING CHANNEL 1


        .NREL

READ1:  LDA  3,@.AC3        ;SAVE ACS AND CARRY
        STA  3,SYAC3
        LDA  3,@.CARY
        STA  3,SYCRY
        STA  0,SYAC0
        STA  1,SYAC1
        STA  2,SYAC2
        LDA  0,0            ;SAVE ADDRESS 0
        STA  0,SYAD0
        LDA  0,MASK2        ;RESET PRIORITY
        DOBS 0,CPU          ;TURN ON INTERRUPT AGAIN
        SUB  3,3            ;SET ADCV CHANNEL POINTER
READ2:  DOAS 3,ADCV         ;SELECT ADCV CHANNEL
        LDA  0,ADCST        ;GET ADCV READING TIME CONSTANT
        INC  0,0,SZR        ;WAIT FOR END OF CONVERSION
        JMP  .-1            ;WAIT AGAIN
        LDA  0,.ADST        ;RESET TIME COUNTER
        STA  0,ADCST
        DIC  0,ADCV         ;INPUT THE DATA
        NEG  0,0            ;TURN TO POSITIVE
        LDA  1,MAX          ;GET MAX READING LIMITS
```

```
        ADCZ# 0,1,SNC    ;IS READING GREATER THAN MAX?
        JMP  READ2       ;YES, READ AGAIN
        SUBZL 1,1        ;GET THE LOWER LIMIT
        ADCZ# 1,0,SNC    ;IS READING SMALLER THAN MIN?
        JMP  READ2       ;YES, READ AGAIN
        STA  0,ADRED,3   ;STORE THE READING
        MOV  3,3,SZR     ;IS CHANNEL POINTER ZERO?
        JMP  .+3
        INC  3,3         ;INCREASE CHANNEL POINTER
        JMP  READ2       ;GOTO READ ANOTHER CHANNEL
        SUB  3,3
        LDA  1,ADRED     ;GET FIRST READING
        LDA  2,ADRED+1   ;GET THE SECOND READING
        STA  3,VOLT0     ;CLEAR IT
        STA  1,VOLT0+1   ;STORE TO FPP AREA
        STA  3,VOLT0+2   ;CLEAR IT
        STA  2,VOLT0+3   ;STORE TO FPP AREA



        ;CONVERT THE READING INTO TEMPERATURE

        LDA  0,2         ;SET UP ANOTHER FPP SYSTEM
        STA  0,FPWSA
        LDA  0,#.ITMP
        STA  0,SITMP
        LDA  0,.WSA
        STA  0,2
        FINT             ;INTTIALIZE THE NEW FPP
TEMP:   SUB  2,2
        FENT             ;ENTER FPP
        FFLO VOLT0,2     ;CONVERT INTO FPP
        FLDA  2,VOLT0,2          ;GET THE READING  X(N)
```

```
;DIGITAL FILTERING
;THE FILTER ALGORITHM IS
;
;     YN=(T/(T+TS))*Y(N-1)+(TS/(T+TS))*XN

FLDA 1,PREAD,2   ;GET PREVIOUS READING   Y(N-1)
FLDA 3,SAMMT     ;GET SAMPLING TIME      TS
FLDA 0,CRFRQ     ;GET CORNER FREQUENCY   T
FADD 0,3         ;T+TS
FDIV 3,0         ;T/(T+TS)
FMPY 0,1         ;Y(N-1)*(T/(T+TS))
FLDA 3,SAMMT     ;TS
FLDA 0,CRFRQ     ;T
FADD 3,0         ;T+TS
FDIV 0,3         ;TS/(T+TS)
FMPY 3,2         ;X(N)*(TS/(T+TS))
FADD 1,2         ;Y(N)=Y(N-1)*T/(T+TS)+X(N)*TS/(T+TS)
FSTA 2,PREAD,2   ;Y(N-1) IN PREAD+AC2
FJMP  CNVRT      ;GOTO CONVERSION ROUTINE
```

```
SYAC0:  0
SYAC1:  0
SYAC2:  0
SYAC3:  0
SYCRY:  0
SYADR:  0
SITMP:  0
FPWSA:  0
SAMMT:  0
SAMMT:  1
CRFRQ:  .BLK 2
.WSA:   SYWSA
.AC2:   AC3
.CARY:  CARRY
.ITMP:  ITMP
.PCRK:  PCRCK
.TOCN:  TTOCN
BETA:   1.11719
ALPHA:  -5.37497E-4
GAMMA:  35.9827
MAX:    400
SEPAT:  377
ADCST:  177771
.ADST:  177771
MASK1:  177771
MASK2:  177773
```

```
        ;TEMPERATURE CONVERSION
        ;CONVERSION ALGORITHM:
        ;    TEMP=ALPHA*(READING)**2+BETA*READING+GAMMA

CNVRT:  FLDA  3,BETA        ;GET THE CONSTANT BETA
        FMPY  2,3           ;BETA*V
        FMPY  2,2           ;V*V
        FLDA  0,ALPHA       ;GET THE CONSTANT ALPHA
        FMPY  0,2           ;ALPHA*V*V
        FLDA  0,GAMMA       ;GET GAMMA
        FADD  3,0           ;BETA*V+GAMMA
        FADD  2,0           ;ALPHA*V**2+BETA*V+GAMMA
        FSTA  0,VOLT0,2     ;RESTORE THE TEMPERATURE
        FSTA  0,20,2        ;STORE TO OUTPUT BUFFER
        FEXT
        MOV   2,2,SZR       ;IS DATA POINTER ZERO?
        JMP   .+4           ;NO, GO AHEAD
        INC   3,2           ;INCREASE DATA POINTER
        INC   2,2           ;BY TWO
        JMP   TEMP+1        ;GOTO CONVERT ANOTHER DATA
        JMP   .+1           ;RESERVE ONE SPACE
        SUBZL  3,3
        STA   3,@.TOCN      ;CONVERT CYCLE INDICATION
GOCNT:  JSR   @CNALM        ;GOTO CONTROL SUBROUTINE
```

```
;END OF ONE CONTROL CYCLE
;INITIALIZE THE PUNCH OF DATAS
;AND RESTORE COMPUTER STATUS

LDA   0,C20         ;GET DATA POINTER
LDA   2,.PCBK       ;GET PUNCH CONTROL BLOCK
STA   0,13,2        ;RESET PUNCH DATA POINTER
SUB   0,0
STA   0,7,2         ;RESET PUNCH CYCLE START INDICATOR
NIOC PTP            ;IDLE PTP
LDA   0,SEPAT       ;GET DATA SEPARATE WORD
DOA   0,PTP         ;SEND SEPARATE WORD TO PTP
LDA   0,FPWSA       ;RESTORE WRITABLE AREA ADD
STA   0,7
LDA   0,SITMP       ;RESTORE FPP TEMP STORAGE
STA   0,@.ITMP
LDA   0,SYCRY       ;RESTORE CARRY
MOVR 0,0
LDA   1,SYAC1       ;RESTORE ACS
LDA   2,SYAC2
LDA   3,SYAC3
INTDS               ;DISABLE INTERRUPT
LDA   0,MASK1       ;RESET PRIORITY
DORS 0,CPU          ;TURN ON INTERRUPT AGAIN
NIOS PTP            ;START PTP PUNCHING
LDA   0,SYAC0       ;RESTORE AC0
JMP   @SYADD        ;RETURN TO MAIN PROGRAM


C20:    20
INCNT:  JSP @CUALM
SYMSA:  .BLK 100.


        .END
```

```
        ;MULTIPLICATIVE CONTROL ALGORITHM

        .TITL MULPLTV
        .EXTN VOLTO, SAMMT, FENT
        .ENT  ALPAR, H2OTM, CONTL, CNTLS, ACALC

        ;CONTROL PARAMETER REQUESITION SUBROUTINE
        ;
        ;  FILCT:        CONTROL TIME CONSTANT
        ;  KPROC:        PROCESS GAIN
        ;  KCONT:        CONTROLLER GAIN
        ;  U1:           LOAD 1
        ;  U2:           LOAD 2
        ;  A :           CONTROLLER CONSTANT


        .NREL

ALPAR:  STA   3,RETRN     ;SAVE RETURN
RPKST:  LDA   2,.MES0     ;SEND REQUEST MESSAGE
        JSR   @4
        FENT
        FDEC  0           ;INPUT THE PARAMETER
        FSTA  0,FILCT     ;STORE TO CONTROL AREA
        FEXT
        JSR   @6
        JMP   RPKST
RSTEP:  LDA   2,.MES2     ;SEND REQUEST MESSAGE
        JSR   @4
        FENT
        FDEC  0
        FSTA  0,KCONT
        FEXT
        JSR   @6
        JMP   RSTEP
RITIM:  LDA   2,.MES6
        JSR   @4
        FENT
        FDEC  0
        FSTA  0,ITIME
        FEXT
        JSR   @6
        JMP   RITIM
RU1:    LDA   2,.MES3
        JSR   @4
        FENT
        FDEC  0
        FSTA  0,U1
        FEXT
        JSR   @6
        JMP   RU1
        JMP   ACALC+1
```

```
        ; A VALUE AND B VALUE CALCULATION
        ;
        ; A=(K1*K2*K*R+K1*K2*U1+U2-R)/
        ;    (K*R*R-K1*K2*K*R*U1-K*R*U2)
        ;
        ; B=EXP(-TS/TC)

ACALC:  STA  3,RETRN
        FENT
        FLDA 0,H2OTM     ;CALCULATE KPROC BY
        FLDA 1,ALPHA     ;    KPROC = ALPHA*H2OTM^2
        FLDA 2,BETA      ;            + BETA*H2OTM+GAMMA
        FMPY 0,2         ;BETA*H2OTM
        FMPY 0,0         ;H2OTM^2
        FMPY 0,1         ;ALPHA*H2OTM^2
        FLDA 0,GAMMA     ;GAMMA
        FADD 1,0         ;ALPHA*H2OTM^2+GAMMA
        FADD 2,0         ;KPROC
        FSTA 0,KPROC
        FLDA 3,H2OTM     ;R
        FLDA 2,KCONT     ;K
        FLDA 1,U2        ;U2
        FMPY 3,2         ;K*R
        FMPY 2,1         ;K*R*U2
        FMPY 2,0         ;K*R*G
        FMPY 3,2         ;K*R*R
        FLDA 3,U1        ;U1
        FMPY 0,3         ;K*R*G*U1
        FADD 3,1         ;K*R*G*U1+K*R*U2
        FSUB 1,2         ;K*R*R-K*R*G*U1
        FLDA 1,H2OTM     ;R
        FSUB 1,0         ;K*G*R-R
        FLDA 1,KPROC     ;G
        FLDA 3,U1        ;U1
        FMPY 1,3         ;G*U1
        FADD 3,0         ;K*G*R+G*U1-R
        FLDA 3,U2        ;U2
        FADD 3,0         ;K*G*R+G*U1-R+U2
        FDIV 2,0         ;(K*G*R+G*U1-R+U2)/
                         ;(K*R*R-K*R*G*U1-K*R*U2)
        FSTA 0,AVALU     ;STORE TO A
        FLDA 0,@SAMTM    ;TS
        FLDA 2,FILCT     ;TC
        FDIV 2,0         ;TS/TC
        FNEG 0,0         ;-TS/TC
        FEXP 0,0         ;EXP(-TS/TC)
        FSTA 0,BVALU     ;B=EXP(-TS/TC)
        FSUB 0,0
        FSTA 0,PRMN1
        FSTA 0,EREN1
        FSTA 0,YBEFE
        FEXT
        JMP  @RETRN      ;RETURN TO EXEC PROGRAM
```

```
        .ZRTL


.MES0:  MESS0
.MES1:  MESS1
.MES2:  MESS2
.MES3:  MESS3
.MES4:  MESS4
.MES5:  MESS5
.MES6:  MESS6
RETRN:  0
FILCT:  .BLK 2
KPROC:  .BLK 2  ;PROCESS GAIN
H2OTM:  .BLK 2  ;TEMPERATURE SET POINT
KCONT:  .BLK 2
PRMN1:  .BLK 2  ;P(N-1)
FPEN1:  .BLK 2
DIGOT:  .BLK 2  ;DACV OUTPUT BUFFER
AVALU:  .BLK 2
BVALU:  .BLK 2
U1:     .BLK 2
U2:     35.98
.CNTM:  VOLTO   ;TEMP READING POINTER
SAMTM:  SAMMT   ;SAMPLING TIME PARAMETER POINTER
UPLMT:  0.17E+2         ;UPPER LIMIT OF P(N)
LWLMT:  0.7E+1  ;LOWER LIMIT OF P(N)
PRCO1:  9.0     ;D/A CONVERSION CONSTANT 1
PRCO2:  29.0    ;D/A CONVERSION CONSTANT 2
FPONE:  1.00    ;+1
YBEFE:  0.00
ITIME:  20.00
ALPHA:  6.23816E-4
BETA:   -4.30665E-1
GAMMA:  91.616
```

```
        ; ON LINE TUNING OF "A" VALUE
        ; BY INTEGRATION
        ;
        ; A(N)=A(N-1)+TS'*E/TI



        .NREL

AMUB-I: FLDA 0,H20TM    ;R
        FST3 BACK       ;SAVE RETURN ADDRESS
        FLD3 .CNTM
        FLDA 2,2,3      ;C
        FSUB 0,2        ;E=-(R-C)
        FLDA 0,@SAMTM   ;TS'
        FMPY 0,2        ;TS'*E
        FLDA 0,ITIME    ;TI
        FDIV 0,2        ;TS'*E/TI
        FLDA 0,AVALU    ;A(N-1)
        FADD 2,0        ;A(N)=A(N-1)+TS'*E/TI
        FSTA 0,AVALU    ;A(N-1)=A(N)
        FJMP @BACK      ;RETURN

BACK:   0               ;RESERVE FOR RETURN ADDRESS
```

```
        ;MULTIPLICATIVE CONTROLLER FOR THE SYSTEM :
        ;
        ;    H(S)=G/(5.305S+1)^2
        ;
        ; A=(K1*K2*K*R+K1*K2*U1+U2-R)/
        ;    (K*R*R-K1*K2*K*R*U1-K*R*U2)
        ; Y(N)=A*X(N-1)*(1-B)+Y(N-1)*B
        ; E(N)=R-Y(N)*C(N)
        ; X(N)=K*E(N)
        ; B=EXP(-TS/TC)


CONTL:  STA  3,RETRN     ;SAVE RETURN ADD
        FENT             ;ENTER THE FPP
        FJSR AMUMI       ;MUNIPULATE THE "A" VALUE
        FLDA 0,AVALU     ;A
        FLDA 1,FPONE     ;+1
        FLDA 3,BVALU     ;B
        FSUB 3,1         ;1-B
        FMPY 1,0         ;A*(1-B)
        FLDA 1,PRMN1     ;X(N-1)
        FMPY 1,0         ;A*X(N-1)*(1-B)
        FLDA 2,YBEFE     ;Y(N-1)
        FMPY 2,3         ;Y(N-1)*B
        FADD 0,3         ;Y(N)=A*X(N-1)*(1-B)+Y(N-1)*B
        FSTA 3,YBEFE     ;Y(N-1)=Y(N)
        FLD3 .CNTM
        FLDA 2,2,3       ;C(N)
        FMPY 2,3         ;C(N)*Y(N)
        FLDA 0,H20TM     ;R
        FSUB 3,0         ;E(N)=R-C(N)*Y(N)
        FLDA 2,KCONT     ;K
        FSTA 0,EREN1     ;E(N-1)=E(N)
        FJMP .+3
        0
        0                ;RESERVE 3 LOCATIONS FOR
        0                ;PROGRAM MODIFICATION
        FMPY 0,2         ;X(N)=K*E(N)
        FMOV 2,1
        FLDA 0,LWLMT     ;GET +7
        FSTA 1,PRMN1
        FADD 0,1         ;X(N)=X(N)+7
        FJMP .+3         ;RESERVE 3 LOCATIONS
        0
        0
        FLDA 0,UPLMT     ;GET UPPER LIMIT (12PSAG)
        FSUB# 0,1,FSLT   ;SKIP IF VALUE < UPPER LIMIT
        FJMP SETUP
        FLDA 0,LWLMT     ;GET LOWER LIMIT (2PSIG)
        FSUB# 0,1,FSGT   ;SKIP IF VALUE>LOWER LIMIT
        FLDA 1,LWLMT     ;SET OUTPUT=LWLMT
        FJMP NOROT       ;GOTO NORMAL OUTPUT
```

```
SETUP:  FLDA  1,UPLMT     ;SET OUTPUT=UPLMT
NOROT:  FJMP  .+1         ;STORE PRESENT PRESSURE TO PL
        FLDA  0,PRCON     ;CONVERT TO HARDWARE PRESSURE
        FLDA  2,PROFS
        FMPY  0,1
        FSUB  2,1
        FSTA  1,DIGOT     ;STORE TO DIGITAL OUTPUT AREA
        FFIX    DIGOT     ;CONVERT TO FIX POINT
        FEXT
        LDA   1,DIGOT+1   ;GET THE OUTPUT SIGNAL
        SUB   0,0
        DOB   0,DACV      ;DACV CHANNEL ZERO SELECTOR
        DOAS  1,DACV      ;OUTPUT THE CONTROL SIGNAL
        JMP   @RETRN      ;GO BACK TO READ1
```

```
        ;PROPORTIONAL AND INTEGRAL CONTROL ALOGRITHM

        .TITL PICONTL
        .EXTN VOLTO, SAMMT, FENT
        .ENT  ALPAR, H2OTM, CONTL, CNTLS


        ;CONTROL PARAMETER REQUESITION SUBROUTINE
        ;
        ;  ITIME  :      INTEGRAL TIME  (SEC)
        ;  KCONT  :      PROPORTIONAL CONSTANT (PSI/DEG F)


        .NREL

ALPAR:  STA  3,RETRN     ;SAVE RETURN
RITIM:  LDA  2,.MES0     ;SENT REQUEST MESSAGE
        JSR  @4
        FENT
        FDEC 0           ;INPUT THE PARAMETER
        FSTA 0,ITIME     ;STORE TO CONTROL AREA
        FEXT
        JSR  @6                  ;GOTO ERROR DETECT ROUTINE
        JMP  RITIM       ;ERROR! REQUEST AGAIN
RPKST:  LDA  2,.MES1     ;SEND REQUEST MESSAGE
        JSR  @4
        FENT
        FDEC 0           ;INPUT THE PARAMETER
        FSTA 0,KCONT     ;STORE TO CONTROL AREA
        FEXT
        JSR  @6
        JMP  RPKST
        JMP  @RETRN      ;RETURN TO EXEC PROGRAM


CNTLS:  STA  3,RETRN     ;SAVE RETURN
        LDA  2,.MS20     ;SEND MESSAGE
        JSR  @4
        FENT
        FLDA 0,KCONT     ;PRINT OUT CONT CONSTANT
        FFDC 0
        FEXT
        LDA  2,.MS21     ;SEND MESSAGE
        JSR  @4
        FENT
        FLDA 0,ITIME     ;PRINT OUT INTEGRAL TIME
        FFDC 0
        FEXT
        JMP  @RETRN      ;RETURN
```

```
.MES0:  MESS0
.MES1:  MESS1
.MS20:  MES20
.MS21:  MES21

MESS0:  .TXT /<15><12>PI CONTROLLER :<15><12>INTEGRAL TIME (SEC) =
MESS1:  .TXT !<15><12>PROPORTIONAL CONSTANT (PSI/DEG F) =  !
MES20:  .TXT /<15><12>  KCONT:  /
MES21:  .TXT ! PSI/DEG F<15><12>  ITIME:  !

RETRN:  0
ITIME:  .BLK 2  ;INTEGRAL TIME OF PI CONTROL
KCONT:  .BLK 2  ;PI CONTROL CONSTANT
H2OTM:  .BLK 2  ;TEMPERATURE SET POINT
ERRN1:  .BLK 2  ;ERROR(N-1)
PRMN1:  .BLK 2  ;P(N-1)
DIGOT:  .BLK 2  ;DACV OUTPUT BUFFER
.CNTM:  VOLTOL  ;TEMP READING POINTER
SAMT":  SAMMT   ;SAMPLING TIME PARAMETER POINTER
UPLMT:  0.1*E+2         ;UPPER LIMIT OF P(N)
LWLMT:  0.3E+1  ;LOWER LIMIT OF P(N)
PRCON:  2.99779E-2      ;D/A CONVERSION CONSTANT 1
PROFS:  3.32507         ;D/A CONVERSION CONSTANT 2
```

```
        ;PI CONTROL ALGORITHM SUBROUTINE
        ;
        ;P(N)=P(N-1)+KC*(E(N)-E(N-1))+KC*TS'*E(N)/TI


CONTL:  STA  3,RETRN     ;SAVE RETURN ADD
        FENT             ;ENTER THE FPP
        FLD3  .CNTM      ;GET DATA POINTER
        FLDA 0,+2,3      ;GET THE CONTROL TEMP  TR
        FLDA 1,H2OTM     ;GET TEMP SET POINT  TS
        FSUB 0,1         ;E(N)=TS-TR
        FLDA 0,@SAMTM    ;TS'
        FMPY 1,0         ;TS'*E(N)
        FLDA 2,KCONT     ;KC
        FMPY 2,0         ;KC*TS'*E(N)
        FLDA 3,ITIME     ;TI
        FDIV 3,0         ;KC*TS'*E(N)/TI
        FLDA 3,EREN1     ;E(N-1)
        FSTA 1,EREN1
        FSUB 3,1         ;E(N)-E(N-1)
        FMPY 2,1         ;KC*(E(N)-E(N-1))
        FLDA 2,PRMN1     ;P(N-1)
        FADD 2,1         ;P(N-1)+KC*(E(N)-E(N-1))
        FADD 0,1  ;P(N-1)+KC*(E(N)-E(N-1))+KC*TS'*E(N)/TI
        FLDA 0,UPLMT     ;GET UPPER LIMIT (18PSAG)
        FSUB# 0,1,FSLT   ;SKIP IF VALUE < UPPER LIMIT
        FJMP SETUP
        FLDA 0,LWLMT     ;GET LOWER LIMIT (2PSIG)
        FSUB# 0,1,FSGT   ;SKIP IF VALUE>LOWER LIMIT
        FLDA 1,LWLMT     ;SET OUTPUT=LWLMT
        FJMP NORUT       ;GOTO NORMAL OUTPUT
SETUP:  FLDA 1,UPLMT     ;SET OUTPUT=UPLMT
NORUT:  FSTA 1,PRMN1     ;STORE PRESENT PRESSURE TO ML
        FLDA 0,PRCON     ;X=(Y-PRCON)/PROFS
        FLDA 2,PROFS     ;Y: CALC. PRESSURE
        FSUB 2,1         ;X: CORRELATE FACTOR
        FDIV 0,1         ;X IN FAC1
        FSTA 1,DIGOT     ;STORE TO DIGITAL OUTPUT AREA
        FFIX   DIGOT     ;CONVERT TO FIX POINT
        FEXT
        LDA  1,DIGOT+1   ;GET THE OUTPUT SIGNAL
        SUB  0,0
        DOB  0,DACV      ;DACV CHANNEL ZERO SELECTOR
        DOAS 1,DACV      ;OUTPUT THE CONTROL SIGNAL
        JMP  @RETRN      ;GO BACK TO READ1
```

```
PCNTL:  FLDA 2,KCONT     ;GET P CONSTANT
        FLDA 0,EREN1     ;E(N-1)
        FSTA 1,EREN1
        FSUB 0,1         ;E(N)-E(N-1)
        FMPY 2,1         ;KC*(E(N)-E(N-1))
        FLDA 0,PRMN1     ;P(N-1)
        FADD 0,1         ;P(N)=P(N-1)+KC*(E(N)-E(N-1))
        FSTA 1,PRMN1
        FJMP 0,3


        .END
```

```
        ;SYSTEM INITIALIZATION PROGRAM


        .TITL HEEXEC
        .EXTN DEBUG, ALPAR, H20TM, INTRP, CPERQ, SAMMT
        .EXTN FENT,  FINT,  TTOSV, SAMMT, SATAM
        .EXTN PCHCK, TTOCN, GOCNT, CONON, INCNT
        .EXTN IOXOS, IOXOS, ERROR
        .EXTN OPT,   IPT,   PTPO
        .ENT  STEXC, IOXOS, IOXO2, IOXIS, EXEC
        .ENT  ERROR, PTPO,  IPT,   RESTR
        .ENT  OPT,   GETC,  PUTC,  WSA


        .LOC 0

        0                   ; 0 INTERRUPT RETURN ADDRESS
.INTP:  INTRP               ; 1 INTERRUPT MASTER ROUTINE
STEXC:  JMP @.CEXE          ; 2 START AT EXECUTIVE PROGRAM
STDEB:  JMP @.DEBU          ; 3 START AT DEBUG PROGRAM
.IOXS:  IOXOS               ; 4 TTO OUTPUT A STRING OF CHARACTERS
.IOX2:  IOXO2               ; 5 TTO OUTPUT 2 CHARACTERS
.ERRO:  ERROR               ; 6 FPP INPUT ERROR DETECT
WSA:    FPPWA               ; 7 FPP WRITABLE AREA ADDRESS


        .LOC 20

        13000   ;TTO OUTPUT DATA POINTER
        13000   ;DATA STORAGE POINTER


        .LOC 40

.IPT:   IPT     ;40 TTI INPUT ROUTINE
.OPT:   OPT     ;41 TTO OUTPUT ROUTINE
.PTPO:  PTPO    ;42 PAPER TAPE OUTPUT ROUTINE
.ALPA:  ALPAR   ;43 ALGORITHM REQUEST ROUTINE
        0       ;44 POINTER FOR RELOCATABLE LOADER
GETC=.IPT
PUTC=.OPT



        .ZREL

.CEXC:  EXEC    ;EXEC PROGRAM ADDRESS
.DEBU:  DEBUG   ;DEBUG PROGRAM ADDRESS
```

```
        .NREL

.IOXI:  IOXIS   ;TTI INPUT (UNSTORED)
.CRFR:  CRFRQ
.MES1:  MESS1   ;HEADING
.MES2:  MESS2   ;ASK FOR DATE
.MES3:  MESS3   ;ASK FOR TIME
.MES4:  MESS4   ;ASK FOR SAMPLING TIME
.MES5:  MESS5   ;ASK FOR TEMPERATURE SET POINT
.MES6:  MESS6   ;ASK FOR FILTER CORNER FREQUENCY
.MES7:  MESS7   ;ASK FOR CONTROL OR NOT
.GOCN:  GOCNT   ;GOTO CONTROL ADDRESS
.PCBK:  PCBCK   ;PTPSV CONTROL BLOCK ADDRESS
.TOCN:  TTOCN   ;TTOSV DATA READY INDICATOR
.H2OT:  H2OTM   ;TEMP SET POINT
WIDTH:  10      ;FPP F-FORMAT PARAMETERS
DECIM:  4
CR:     15
LF:     12
CY:     "Y      ;CHARACTER Y
CN:     "N      ;CHARACTER N
FREQ:   1       ;RTC INTERRUPT FREQ=100 CPS
MASK1:  177771  ;INTERRUPT MASK
STADD:  13000   ;OUTPUT DATA STARTING ADDRESS
..TTO:  TTOSV   ;TTI OUTPUT SERVICE ROUTINE ADD
.SAMT:  SAMMI
        SAMMT
.SATA:  SATA
FPONE:  1.00
ISJMP:  JMP  .+1
.CNON:  CONON
.INCN:  INCNT
.CNTM:  10.00
FPRAM:  .BLK 100
```

```
EXEC:   LDA   0,WIDTH     ;SET UP FPP F-FORMAT
        LDA   2,WSA       ;PARAMETERS
        STA   0,121,2
        LDA   0,DECIM
        STA   0,122,2
        FINT              ;INITIALIZE FPP
        LDA   2,.MES1     ;GET HEADING ADDRESS
        JSR   @.IOXS      ;PRINT HEADING
        LDA   2,.MES2     ;ASK FOR DATE
        JSR   @.IOXS
        JSR   @.IOXI
        LDA   2,.MES3     ;ASK FOR TIME
        JSR   @.IOXS
        JSR   @.IOXI      ;INPUT TIME
RDLIT:  LDA   2,.MES4     ;ASK FOR SAMPLING TIME
        JSR   @.IOXS
        FENT
        FDEC  0           ;INPUT DATA SAMPLING TIME
        FLDA  2,.CN10     ;GET 10.00
        FMPY  0,2         ;SECOND*10=HZ
        FSTA  2,@.SAMT    ;STORE HZ TO BUFFER
        FFIX  @.SAMT
        FLD3  @.SAMT+1    ;SAVE THE FIXED POINT DATA
        FSTA  0,@.SAMT    ;STORE THE FPP DATA AGAIN
        FST3  @.SAMT      ;STORE TO DATA POOL
        FEXT
        JSR   @6
        JMP   RDLIT
RCNFQ:  LDA   2,.MES6     ;ASK FOR FILTER FREQ
        JSR   @.IOXS
        FENT
        FDEC  0
        FLDA  1,FPONE     ;GET FPP ONE
        FDIV  0,1         ;T=1/F
        FSTA  1,@.CRFQ    ;STORE CORNER FREQ
        FEXT
        JSR   @5
        JMP   RCNFQ
RCNTL:  LDA   2,.MES7     ;ASK FOR CONTROL OR NOT
        JSR   @.IOXS
        JSR   @40         ;INPUT A CHARACTER
        MOV   0,1         ;SAVE INPUT
        LDA   0,CR        ;SEND CR
        JSR   OPT
        LDA   0,LF        ;SEND LF
        JSR   OPT
        LDA   0,CY        ;GET CHARACTER Y
        SUB#  1,0,SNR     ;IS INPUT "Y"?
```

```
        JMP   RH2OT       ;YES, GOTO ASK CONTROL PARAMETERS
        LDA   0,CN        ;GET CHARACTER N
        SUB#  1,0,SZR     ;IS INPUT "N"?
        JMP   RCNTL       ;NO, WRONG INPUT, ASK AGAIN
        LDA   1,ISJMP     ;YES, DISMISS CONTROL
        STA   1,@.GOCN
        SUB   0,0         ;RESET CONTROL INDICATOR
        STA   0,@.CNON
        JMP   RESTR
RH2OT:  LDA   2,.MES5     ;ASK FOR TEMPERATURE SET POINT
        JSR   @.IOXS
        FEXT              ;ENTERS FPP
        FDFC  0           ;INPUT THE SET POINT
        FSTA  0,@.H2OT    ;STORE TO BUFFER
        FEXT
        JSR   @6                  ;GOTO ERROR ROUTINE
        JMP   RH2OT
        SUBZL 0,0         ;SET CONTROL INDICATOR
        STA   0,@.CNON
        LDA   0,@.INCN    ;GET  JMP  @CNALM  INSTRUCTION
        STA   0,@.GOCN    ;STORE TO PROPER ROUTINE
        JSR   @.ALPA      ;GO TO ALGORITHM REQUEST ROUTINE
RESTR:  LDA   2,.PCBK     ;RESET ALL PARAMETERS BEFORE
        LDA   0,STADD     ;DOING THE DATA LOGGING AND
        STA   0,20        ;CONTROL FUNCTION
        STA   0,21
        STA   0,13,2
        LDA   0,DECIM
        STA   0,10,2
        SUB   0,0
        STA   0,12,2
        STA   0,@.TOCN
        NIOC  PTP         ;IDLE PTP
        LDA   0,FREQ      ;INITIALIZE RTC
        DIAS  0,RTC
        LDA   0,MASK1     ;MASK FOR INTERRUPT
        DORS  0,CPU       ;MASK OUT AND TURN ON INTERRUPT
        JMP   @..TTO      ;GOTO DATA OUTPUT ROUTINE
```

```
MESS1:  .TXT /<15><12><12>
         DIRECT DIGITAL CONTROL DEMONSTRATION<15><12>/
MESS2:  .TXT /<15><12>DATE :    /
MESS3:  .TXT /TIME :    /
MESS4:  .TXT /<15><12>SAMPLING TIME (SEC) = /
MESS5:  .TXT /<15><12>TEMPERATURE SET POINT (F) = /
MESS6:  .TXT "<15><12>CORNER FREQUENCY (C/S) =  "
MESS7:  .TXT /<15><12>CONTROL ( Y ) OR NOT ( N ).  /

        .END
```

```
        ;FLOATING POINT CALCULATION ERROR CHECK

        .TITL ERROR

        .NREL

ERROP:  STA   3,SAVE3       ;SAVE RETURN
        STA   0,SAVE0       ;SAVE ACS
        STA   1,SAVE1
        STA   2,SAVE2
        LDA   2,7,0         ;GET ERROR TABLE POINTER
        LDA   0,0,2         ;CHECK FOR OF/UF
        MOVZR 0,0,SZC       ;IS 15TH BIT SET?
        JMP   ERR0A         ;YES, UNDERFLOWED
        MOVZR 0,0,SZC       ;IS 14TH BIT SET?
        JMP   ERR0A         ;YES, OVERFLOWED!
        LDA   0,1,2         ;NO, CHECK CONVERSION OK
        MOV   0,0,SZR       ;IS COK ZERO?
        JMP   .+3           ;YES, NORMAL RETURN
        JSR   RSTOR         ;RESTORE ACS
        JMP   @SAVE3        ;ERROR RETURN
        JSR   RSTOR         ;RESTORE ACS
        LDA   3,SAVE3       ;NORMAL RETURN
        JMP   1,3

RSTOR:  LDA   0,SAVE0       ;RESTORE AC0
        LDA   1,SAVE1       ;RESTORE AC1
        LDA   2,SAVE2       ;RESTORE AC2
        JMP   0,3           ;RETURN


ERR0A:  SUB   1,1
        STA   1,0,2         ;RESET OF/UF
        JSR   RSTOR         ;RESTORE ACS
        JMP   @SAVE3        ;ERROR RETURN


SAVE0:  0
SAVE1:  0
SAVE2:  0
SAVE3:  0

        .END
```

APPENDIX IV DATA FOR THE EXPERIMENTS

Test	Freq 1/sec	Amplitude ratio	Output lag degree
1	0.6	0.95	- 8.4
2	0.8	0.90	- 29.
3	1.0	0.89	- 39.
4	1.2	0.84	- 52.
5	1.5	0.72	- 63.
6	1.8	0.68	- 91.
7	2.0	0.56	- 95.
8	2.4	0.50	-118.
9	2.8	0.42	-133.
10	3.0	0.33	-146.

Table 1 Data for Bode Diagram of the Heat Exchanger System

T_c (sec)	$1/T_c$ (1/sec)	K (PSI/°F)
1	1.0	4.0
2	0.5	6.3
3	0.333	8.7
4	0.25	11.0
5	0.20	13.99
6	0.1666	15.2
7	0.1429	18.0
8	0.125	19.5
9	0.111	21.0

Table 2 Data for the stability region of the Multiplicative control system controlling the analog computer simulated model.

T_c (sec)	$1/T_c$ (1/sec)	K (PSI/°F) at K_I=1 (1/°F.PSI), and			
		T_I=1000 sec	T_I=600 sec	T_I=400 sec	T_I=200 sec
1	1.0	4.0	3.5	3.0	2.5
2	0.5	6.5	5.5	5.0	4.55
3	0.333	8.5	7.55	7.0	6.55
4	0.25	11.0	9.5	9.0	8.5
5	0.2	13.5	11.5	11.0	10.0
6	0.1666	15.5	13.5	12.5	11.5
7	0.1429	18.5	15.5	15.0	13.5
8	0.125	20.5	17.5	16.5	14.5
9	0.111	22.0	20.0	18.5	16.5

Table 3 Data for the stability region of the Multiplicative control system controlling the heat exchanger system.