DIGITAL COMPUTER SUPERVISION OF AN ANALOG NUCLEAR POWER PLANT SIMULATION

A Thesis

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Master of Science

in

The Department of Nuclear Engineering

by
Ernest Ivry Hamilton, Jr.
B.S., Louisiana State University, 1970
May, 1972

ACKNOWLEDGEMENT

The author is indebted to Dr. Adrian E. Johnson, Jr., of the Chemical Engineering Department, Dr. John C. Courtney, Dr. Frank A. Iddings and Dr. Edward N. Lambremont of the Nuclear Science Center, to Dr. Myron H. Young of Mechanical Engineering, and Mr. Thomas Linton of the Electrical Engineering Department at Louisiana State University for their guidance and encouragement in this work.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

The results of the development of a prototype hybrid computer package to study various proposed digitally controlled systems are presented. The completed package, consisting of a digital control system linked to an analog-simulated process, is the nucleus from which other more elaborate control studies can easily be performed.

The specific physical system implemented in this work to demonstrate the functioning of the package deals with digital computer supervisory control of an analog computer simulated pressurized water nuclear reactor power plant. The prototype was developed utilizing the facilities and equipment of the Louisiana State University Chemical Engineering Hybrid Simulation Laboratory.

The prototype is not limited to study of simulated processes only. By using the analog computer as an interface almost any process with measuring lines generating electrical signals may be studied. Should control be desired there must be controllers that are operable by electrical signals. This means that control studies can be performed with pilot plants (processes) located within communication range of the hybrid interface.

CHAPTER I

INTRODUCTION

History, Background and Data Supporting the Reasons for this Work

A hybrid computer, which has features common to both digital and analog computers, is a very versatile research instrument. It can be used to study many diverse aspects of both similar and dissimilar topics. In the field of nuclear power plant development, for example, the hybrid computer can explore and evaluate such things as control, safety, and plant design, and can be used to train operators by simulating various situations expected in plant operations and management.

Experience has shown that persons trained on a nuclear power plant simulator are better prepared to pass the Operator Licensing Test given by the Atomic Energy Commission.[1] In designing nuclear power plants, the computer can be used to analyze all systems which can be expected to influence reactor behavior. Reactor kinetics equations and auxiliary mathematical expressions of reactor dynamics can be conveniently and rapidly analyzed.

Although the installation of an on-line computer in conventional steam-electric generating plants is becoming quite common today, computer applications in a nuclear plant presents a somewhat different approach. There has been no concerted effort toward direct digital control in nuclear plants because of the extensive precautions necessary in reactor operations.[2]

In the United States, the trend is to use the computer of a nuclear plant only to accumulate data, perform calculations of internal reactor parameters or core performance, and as an operational aid. When a plant's computer is out-of-service, the plant can continue to operate at full power or slightly less using the remaining instrumentation. Operation at less than full power may be required when the plant computer is out-of-service if the computer is being used to determine operating limits.

After a sufficient history of reliable computer operations has been accumulated, control and safety systems will become part of the plant's computer functions. After these functions are assumed, the need for redundancy in control systems in nuclear plants will probably result in dual plant computers. A dual plant computer system would be designed so that failure of one plant computer would automatically allow a second computer to take over the functions of the inoperative computer.

The potential applications of a computer in a nuclear power plant include, in general, the following:

1) Data logging (Scan, Convert, and Alarm)
2) Calculations (Plant efficiencies)
3) Operator Aids (Plant conditions)
4) Safety System (Area radiation monitoring, and systems to reduce the possibility of human error)
5) Supervisory Control

The great advantage of on-line computation is that it can provide the plant operator with almost continuous information which permits operating the plant closer to limits set by engineering, maintenance, safety, or procedural considerations.[3]

A study of digital computer supervision of a nuclear power plant revolves around the very real possibility of optimization of an objective function, which is usually net profit in dollars. The study presents a problem common to many endeavors, namely evaluating the proposed system. In evaluating proposed systems or designs one can generally select either of two options: 1) an experimental evaluation, or 2) an analytical evaluation. An experimental program is usually characterized by a minimum of analysis, the construction of a prototype of the system, and considerable trial and error work with the prototype. The cost and time consumed in an experimental evaluation are normally much greater than that required in an analytical evaluation of the same scope. In the analytical approach, the first task is to derive a set of equations (a mathematical model) whose solution will describe the behavior of the variables of the system. Then these equations instead of the experimental prototype are manipulated to generate the desired results.

Since the derivations of mathematical models nearly always require some degree of approximation, some experimentation usually is required for the verification of the model. However, prototypes designed from analytical investigations hopefully portray reality. Then the only experimental results required are those which validate the mathematical model. Once the model is proven valid, additional data can be generated analytically for various operating conditions, which may result in a considerable cost reduction compared to the experimental approach. The analytical approach is not necessarily always better than the experimental. Both are different forms of analysis. The results of experimentation are required to generate the mathematical relations of parameters (mathematical models) for analytical evaluation, and to determine boundary conditions in some cases.

Electronic computation methods for the solution of mathematical models utilize the digital computer or the analog computer or some combination of both the digital computer and the analog computer called a hybrid computer. In analog computers, the solutions of the mathematical models depend on the analogy between the physical quantities and the mathematical numbers or manipulations. For example, consider the analogy between electrical, mechanical and thermal equations:

$$i = c \frac{de}{dt} \text{ --- (current flow through a capacitor)}$$

$$F = \frac{w}{g} \frac{dv}{dt} \text{ --- (Force acting on a mass)}$$

$$Q = wc \frac{dT}{dt} \text{ --- (Heat flow in a solid)}$$

The form of the differential equation is the same in each case, with the only difference being the constants and the physical meaning of the variables. The variables are represented by scaled voltages in an analog computer. The term "scaled voltages" (Magnitude Scaling) in analog computers means that the voltage output of each amplifier is proportional to the represented problem variable. This proportionality constant is chosen so that the problem variable (temperature, pressure, etc.) will be at maximum or minimum when the analog computer variable (voltage) is maximum or minimum. (Usually this maximum is +10 volts or +100 volts and the minimum is -10 volts or -100 volts, depending on the machine reference voltage.) If a number of events take place at the same time in the real world they will also take place at the same time in the analog computer simulation. This occurrence at the same time in the real world is termed parallel operation. On an analog computer it is called parallel solution.

Digital computers perform all calculations serially, not in parallel, and therefore, unlike the analog computer, require more and more time as the problem becomes even more complex. In reality the machine performs only Boolean Operations, which are used to build an adder which adds numbers. To subtract, the machine adds the complement of the number to itself the number of times equal to the multiplier. Division has a similar algorithm. All operations that can be performed on a digital computer have algorithms that, in their elementary form, depend only on a certain structure of Boolean Logic. Therefore, with the basic set of operations of Boolean Logic (and,or, not) the computer can perform many operations.[(4)]

The Problem

The objective of the work to be described in this thesis was to develop a prototype hybrid computer package to study various proposed systems. The package is the nucleus about which other more elaborate studies can easily be made by other persons by using more detailed analog models and by adding more control functions to the digital section. This package has been developed in a modular form, allowing virtually any problem that can be simulated on the analog to be studied. The specific physical system implemented in this work (to demonstrate that the package functions correctly) deals with digital computer supervision of a simulated nuclear power plant. The prototype was devised utilizing the facilities and equipment of the Louisiana State University Chemical Engineering Simulation Laboratory.

Hybrid Computer Simulation

The simulation of a nuclear power plant and its associated plant digital computer is logically suited to hybrid techniques because a nuclear power plant can be conveniently simulated on the analog computer while the digital computer performs the functions of the digital supervisor. In general, analog computers have the following advantages over the digital computer for simulation of physical systems:

1) The speed of the solution is independent of the problem complexity, and can be chosen to be faster or slower than the physical system being simulated.
2) The analogy between the computer simulation variable and the problem variable is straightforward.
3) The values of parameters and input variables can be easily changed during operation, and the results of the change can be observed at the rate of the time scale of the analog simulation.

Digital computers, on the other hand, have quite different advantages compared to the analog computers:

1) They are more precise, and solutions can be made as accurate as the solution time or the mathematical model will allow.
2) They handle logical operations much better than analogs.
3) They can store and manipulate huge quantities of data.

4) With floating point arithmetic, magnitude scaling is no problem.

To enlarge slightly on the advantages of analog and digital computers, consider the concept of information flow in either continuous or discrete states. Continuous information flow is in the realm of the analog computer. An analog computer is made up of electronic components which function basically as operators in the mathematical sense. Information flow in the discrete form is in the realm of the digital computer. In digital computers, structures depending on Boolean Logic serve as mathematical operators.[5]

CHAPTER II

NUCLEAR REACTOR PLANT SIMULATION

Simulation of Neutron Kinetics

The kinetics of a nuclear fission reactor can be approximated by the time and space dependent diffusion equation:

$$\nabla^2\Phi + B^2\Phi = \frac{\partial n}{\partial t}$$

where Φ is the neutron flux in n/cm^2-sec

B^2 is the buckling in cm^{-2}

n is the neutron density in n/cm^3

By considering a point in a reactor away from source sink and boundary that has no spatial variation of neutron flux, the diffusion equation can be simplified to give:

$$\frac{dn}{dt} = \frac{\delta k-\beta}{l^*} n + \sum_{i=1}^{m} \lambda_i C_i$$

$$\frac{dC_i}{dt} = \frac{\beta_i}{l^*} n - \lambda_i C_i$$

where k is the effective multiplication factor of the reactor

δk is $(k-1)/k$, the reactivity

β is the delayed neutron yield

l^* is the mean effective lifetime of prompt neutrons in the system

m is the number of delayed neutron groups

λ_i is the decay constant of the ith group of delayed neutron precursors.

C_i is the concentration of the ith group of delayed neutron precursors.

With these simplifications the time and space dependent diffusion equation becomes a single-node kinetics simulator. This single-node simulation has thus neglected the space dependence leaving only the time dependent diffusion equation. This time dependent diffusion equation is an ordinary differential equation, which is easily programmed on an analog computer.

There are three basic types of single-node kinetics simulators:

1) One amplifier per group of delayed neutrons connected to a single amplifier.

2) A single amplifier with a complex input impedance.

3) A single amplifier using a complex feedback impedance.

Each of these three types has its place in nuclear reactor studies. The circuit with one amplifier per group of delayed neutrons is used when amplifier availability is not a problem or when the delayed groups are to have initial conditions, such as in the simulation of an old core. The circuit using a single amplifier with complex input impedance is used when there is a shortage of amplifiers but the variable dn/dt must be available, such as in a multinode reactor core simulation. A multinode reactor core simulation uses a number of coupled space and time dependent diffusion equations, thus giving spatial variation and the variation of neutron flux. The circuit using the single amplifier with complex feedback impedance is used when there is a need to conserve amplifiers and the variable dn/dt is not explicitly needed in the simulation. This is accomplished with no loss of accuracy.

Due to the large range of the power level in a reactor, as much as a factor of 10^{14} from source level to peak power level, the simulation of a nuclear reactor on an analog computer presents a difficult magnitude scaling problem. Since it is impractical to cover more than two or three decades in a single run on an analog computer, there have been several alternate methods developed to treat the problem. One of these methods depends on the nature of logarithms. By a substitution of variables, the time dependent kinetics equations can be solved for the logarithm of the power level instead of the power level. This allows the power level itself to vary over a hundred decade range remaining within the magnitude limitations of the analog computer. This advantage is not gained without added problems. These problems stem from the large number of multipliers that must be used (one multiplier for each group of delayed neutrons) and the high accuracy needed in the multipliers to obtain satisfactory results in the simulation. Another method that produces adequate precision depends on the manual rescaling of the problem or the subdividing of the original problem into a number of problems each covering two or three decades of the original problem.[5] In this study the method implemented involves normalizing the neutron flux which is equivalent to simulating the operation of the nuclear reactor from three or four decades below maximum power level up to maximum power level.

The Analog Implementation

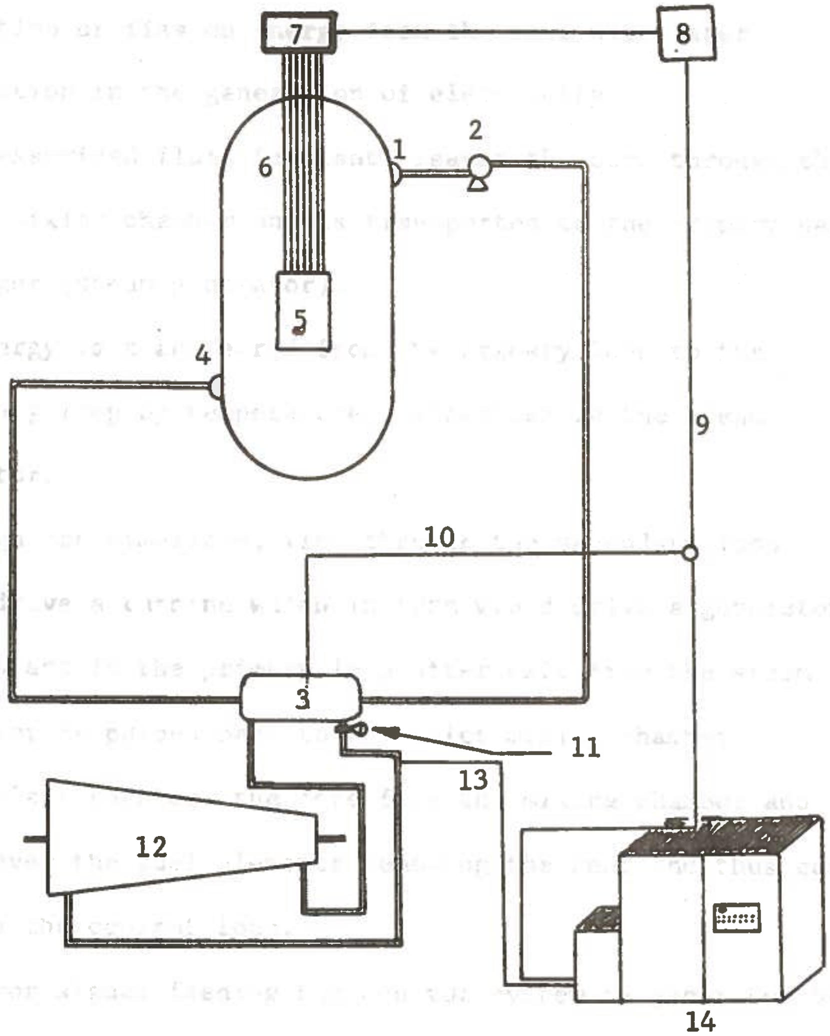
This simulation was adapted from a study performed by EAI (Electronic Associates, Inc., Red Bank, N.J.) on the primary loop of a nuclear power plant.[6] In the EAI study the reactor neutron kinetics was simulated by a passive feedback network. The transport delay of the coolant flow was simulated by a piece of electronic equipment called a capacitor wheel. The scram system consisted simply of a manual switch which was required by the need for a human operator to perform the function of quickly shutting down the simulation of a nuclear reactor in operation. The adaptation for this work consisted of transforming the analog program from dual EAI Model TR-10's to a model EAI-680 along with transforming the passive network for simulation of the nuclear reactor kinetics, the transport delay of the primary coolant loop, and the scram system, all to the equivalent conventional analog patching.

The simulator will reproduce the behavior of the primary loop of a large pressurized water nuclear reactor (PWR) shown in Figure 2-1. In this adaptation, the primary loop is considered to be operating initially under steady-state conditions at one-half of its maximum power. Typical responses studied in this simulation are the response of the reactor to a step change in reactivity; the response of the control system to power demand changes; the reactor response to control system failure during power demand changes; and the response of the reactor when scram rods (large negative reactivity) are inserted into the core.

There are a number of interacting physical systems represented by the simulation:

1) The reactor uses ^{235}U fuel elements as the source of energy (heat) from the fission process.

2) Heat transfer within the fuel element is caused by the temperature difference within the element.



1. Outlet Plenum
2. Primary Coolant Pump
3. Heat Exchanger
4. Inlet Plenum
5. Reactor Core
6. Control Rods
7. Control Rod Drive Unit
8. Cascade Controller
9. Analog Error Signal
10. Average Temperature Across Heat Exchange
11. Valve to Maintain Constant Pressure to Turbine
12. Turbine
13. Steam Temperature
14. Digital Computer

Figure 2-1
Schematic of PWR Primary Loop

3) The fuel element to coolant heat transfer is due to temperature differences between them. This results in the partial extraction of fission energy from the core with later application in the generation of electricity.

4) The pressurized fluid (coolant) leaves the core through the outlet mixing chamber and is transported to the primary heat exchanger (steam generator).

5) The energy is transferred from the primary loop to the secondary loop by temperature differences in the steam generator.

6) Although not simulated, flow through the secondary loop would drive a turbine which in turn would drive a generator.

7) The coolant in the primary loop after exit from the steam generator is pumped back to the inlet mixing chamber.

8) The coolant reenters the core from the mixing chamber and flows over the fuel elements removing the heat and thus completing the coolant loop.

9) The error signal feeding the control system is generated by the difference between the average temperature of the primary loop coolant existing in the steam generator and the setpoint temperature (the temperature at which the steam generator should be operating).

10) The error signal is operated on by a cascade controller to generate the control rod position which in turn determines the reactivity which through the reactor kinetics equation dictates the neutron level. The cascade controller takes the error signal and operates on it with a PI (proportional,

Reactor Kinetics: $$\frac{dn(t)}{dt} = \frac{\rho n(t)}{l^*} - \frac{\beta n(t)}{l^*} + \sum_{i=1}^{2} \lambda_i C_i$$

$$\frac{dC_i}{dt} = \frac{\beta_i n(t)}{l^*} - \lambda_i C_i$$

Fuel Element Heat Transfer:

$$\frac{dT_f}{dt} = \frac{n\Delta H_f}{M_f C_f} - \frac{UA}{M_f C_f} T_f + \frac{UA}{M_f C_f} T_c$$

Fuel Element to Coolant Heat Transfer:

$$\frac{dT_c}{dt} = \frac{UA}{M_c C_c} (T_f - T_c) - \frac{2W_c}{M_c} (T_c - T_{ic})$$

Out of Reactor Core:

$$T_{oc} = 2T_c - T_{ic}$$

Outlet Plenum:

$$\frac{dT_o}{dt} = \frac{W_c}{M_o} (T_{oc} - T_o)$$

Piping Delay to Steam Generator:

$$T_{ix} = T_o(t-D)$$

Steam Generator:

$$\frac{dT_x}{dt} = \frac{W_c}{M_x} (T_{ix} - T_{ox}) - \frac{U_x A_x}{M_x C_c} (T_x - T_s)$$

Out of Steam Generator:

$$T_{ox} = 2T_x - T_{ix}$$

Figure 2-2.1
Heat Generation and Coolant Transfer Loop Equations

Piping Delay to Inlet Plenum

$$T_{ix} = T_{ox}(t-D)$$

Inlet Plenum

$$\frac{dT_{ic}}{dt} = \frac{W_c}{M_i}(T_i - T_{ic})$$

Figure 2-2.2
Heat Generation and Coolant Transfer Loop Equations

Average Temperature:

$$T_{avg} = 0.5(T_{ox} + T_{ix})$$

Error Signal:

$$\epsilon(t) = T_{ref}-T_{avg}$$

PI Controller:

$$\frac{d^2\mu(t)}{dt} = \frac{1}{\tau_m}\frac{d\mu(t)}{dt} = \frac{K_m}{\tau_c}\left(\frac{n_o-n}{n}\right)$$

Reactivity:

$$\delta k = \delta k_p + \delta k_f + \delta k_c + \delta k_t$$

$$\delta k_t = \alpha(T_f-T_o)$$

Figure 2-3
Control Systems Equations

Reactor Kinetics:

$$\frac{d}{dt}[10n^*] = (0.5)[2000\delta k][10n^*] - 10(0.64)[10n^*] + 10\sum_{i=1}^{2}\lambda_i C_i^*$$

$$\frac{d}{dt}\left[\frac{c_i^*}{10}\right] = \frac{1}{10^2}\frac{\beta_i}{l^*}[10n^*] - \lambda_i\left[\frac{c_i^*}{10}\right]$$

Fuel Element Heat Transfer:

$$\frac{d}{dt}\left[\frac{T_f}{200}\right] = 0.1\left(\frac{\Delta H_f}{200\, M_f C_f}\right)[10n^*] - \left(\frac{UA}{M_f C_f}\right)\left[\frac{T_f}{200}\right] + 0.1\left(\frac{5UA}{M_f C_f}\right)\left[\frac{T_c}{100}\right]$$

Fuel Element to Coolant Heat Transfer:

$$\frac{d}{dt}\left[\frac{T_c}{100}\right] = \left(\frac{2UA}{M_c C_c}\right)\left[\frac{T_f}{200}\right] - \left(\frac{UA + 2W_c C_c}{M_c C_c}\right)\left[\frac{T_c}{100}\right] + \left(\frac{2W_c C_c}{M_c C_c}\right) T_{ic}$$

Out of Reactor Core:

$$\left[\frac{T_{oc}}{100}\right] = 10(0.2)\left[\frac{T_c}{100}\right] - \left[\frac{T_{ic}}{100}\right]$$

Outlet Plenum:

$$\frac{d}{dt}\left[\frac{T_o}{100}\right] = \left(\frac{W_c}{M_o}\right)\left[\frac{T_{oc}}{100}\right] - \left(\frac{W_c}{M_o}\right)\left[\frac{T_o}{100}\right]$$

Piping Delay to Steam Generator:

$$\left[\frac{T_{ix}}{100}\right] = \left[\frac{T_o}{100}\right](t-D)$$

Steam Generator:

$$\frac{d}{dt}\left[\frac{T_x}{100}\right] = \frac{2W_c}{M_x}\left[\frac{T_{ix}}{100}\right] + \frac{U_x A_x}{M_x C_c}\left[\frac{T_s}{100}\right] - \frac{U_x A_x}{M_x C_c}\left[\frac{T_x}{100}\right] - \frac{2W_c}{M_x}\left[\frac{T_x}{100}\right]$$

Figure 2-4.1
Analog Magnitude Scaled Heat Generation and Coolant Transfer Loop Equations

Out of Steam Generator:

$$\left[\frac{T_{ox}}{100}\right] = 10(0.2)\left[\frac{T_x}{100}\right] - \left[\frac{T_{ix}}{100}\right]$$

Piping Delay to Inlet Plenum:

$$\left[\frac{T_{ix}}{100}\right] = \left[\frac{T_{ox}}{100}\right](t-D)$$

Inlet Plenum:

$$\frac{d}{dt}\left[\frac{T_{ic}}{100}\right] = \frac{W_c}{M_i}\left[\frac{T_i}{100}\right] - \frac{W_c}{M_i}\left[\frac{T_{ic}}{100}\right]$$

Figure 2-4.2
Analog Magnitude Scaled Heat Generation and Coolant Transfer Loop Equations

Average Temperature:

$$\left[\frac{T_{avg}}{100}\right] = 0.5\left(\left[\frac{T_{ox}}{100}\right]+\left[\frac{T_{ix}}{100}\right]\right)$$

Error Signal:

$$\left[\frac{\epsilon}{100}\right] = [10]\left(\frac{T_{ref}}{1000}\right) - \left[\frac{T_{avg}}{100}\right]$$

Cascade Controller:

$$[2(n_o-n^*)] = (2\text{x}10^3K_c)\int_o^t \frac{1}{10}\left[\frac{\epsilon}{100}\right]dt + (2000K_c\tau_c)\left[\frac{\epsilon}{100}\right]$$

$$-0.200[10^*] + 2n_o(0)$$

Control Rod Drive Unit:

$$\frac{d^2}{dt^2}[2000\mu] = \left(\frac{1}{\tau_m}\right)\frac{d}{dt}[2000\mu] = \left(\frac{K_M 10^3}{\tau_m}\right)\frac{[20(n_o-n^*)]}{[10n^*]}$$

Reactivity:

$$[2000\delta k] = 10(20k)[10] - 10(4\text{x}10^4|\alpha|)\left[\frac{T_f}{200}\right] + [2000\mu]$$

$$k = \delta k_f + \delta k_c(0) + \delta k_p - \alpha T_o$$

Figure 2-5
Analog Magnitude Scaled Control Systems Equations

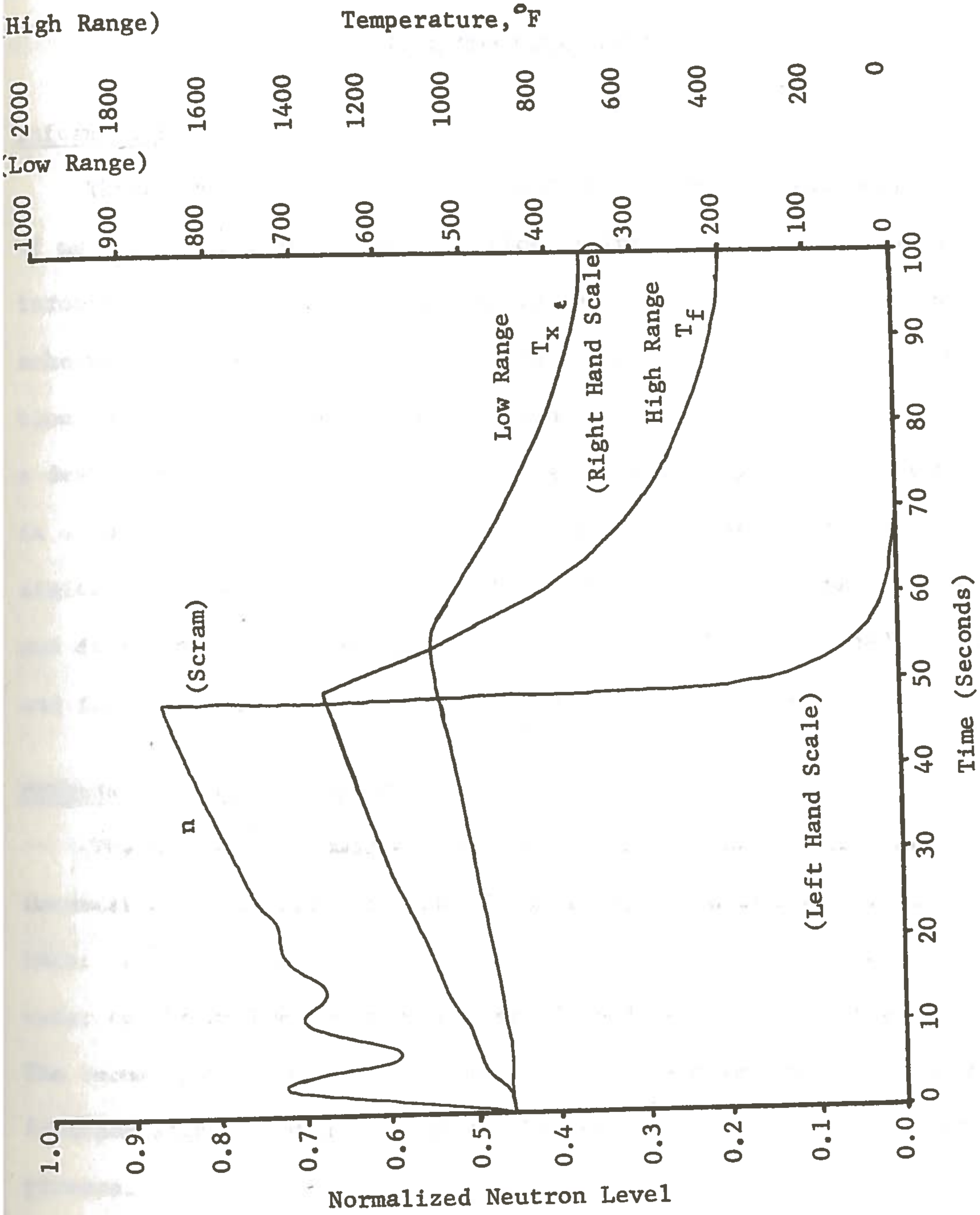


Figure 2-6

Reactor Scram Response

CHAPTER III

CONTROL COMPUTER CONCEPTS

Information Flow

Three types of flow can be considered in process analysis: flow of material, flow of energy, and flow of information. The flow of information is enhanced by the incorporation of control computers in the scheme. Information flow is essential to control for without information in its many forms the control function (the ability to accomplish a desired end) is effectively lost. A digital computer control system is a tool which may be applied in the area of information flow. The digital computer has the ability to quickly acquire, assimilate, analyze, and disseminate large amounts of information with great speed, accuracy, and flexibility.

Methods of Computer Implementation

The methods of computer implementation of control which are of interest are: 1) off-line, and 2) on-line. Each can be further divided into: a) open-loop, and b) closed-loop. The terms off-line and on-line refer to the method by which process data is entered into the computer. The terms open-loop and closed-loop refer to the method by which the feedback signal that is calculated by the computer is passed to the process.

The off-line computer receives information about the process from a human intermediary and the resulting calculated control actions are applied to the process through the human operator (open-loop mode). In the closed-loop mode, the computer applies the control action to the process through an electronic interface that converts the signals into settings for the controlled elements in the process.

As the intricacies of the process unfold, the potential of information flow in the process is enhanced by on-line computers. The term "on-line" refers to the ability of the computer to accept signals directly from process instruments and to convert them into a form suitable for computer processing. The on-line ability greatly reduces the data accumulation time. An on-line computer could operate in the open-loop mode, again referring to the fact that the resultant calculated actions are applied to the process through a human operator. In on-line, closed-loop computer control the computer receives information strictly from the process and its calculated control actions are applied directly to the process through suitable instruments. In this last mode the faster information flow results in the least delay time from the time something happens in the process to the time the process receives a resulting control action.(7)

Software Requirements

In the past in computer control implementation, there have been many instances of underestimation of the cost of user-written software. There are several reasons for this:

1) Since every process is unique there will be a considerable amount of effort expended initially in developing the custom software required by each process. Cost for the effort depends upon many things, one of which is the previous experience of the personnel on the project.

2) The software was often not designed to be easily expandable. The once-and-for-all concept was used with little or no thought for future expansion.

The development of the philosophy and general operating characteristics of a complete set of real-time programs is an extremely important task. The success of the entire project may hinge upon successful software design and development. To some degree the structure of the software system differs between DDC (Direct Digital Control) and Supervisory Control. In DDC, the tasks to be performed are usually simple, but must be performed at frequent intervals. These tasks are usually very similar. In contrast, in Supervisory Control the tasks are usually longer and more complex, but the system is usually composed of fewer tasks. Supervisory Control can be thought of as setpoint control. The computer outputs the setpoint to the controlled device (analog controller) which in turn maintains the controlled variable at setpoint conditions by manipulating a control element. In DDC the computer outputs (position, etc.) directly to the control element, thereby replacing the function performed by the analog controller should the controlled variable experience a disturbance. While the demands on the software packages differ between DDC and Supervisory Control, both must operate in real time.

In either Supervisory Control or DDC, the monitor or executive system can be divided into three parts:

1) Interrupt Servicing,

2) Cyclic Program Servicing,

3) Free-time Servicing.

The basis for this structure revolves about the idea that the accomplishment of some tasks are more important than the accomplishment of others. Therefore the tasks that are more important are assigned a higher priority than the less important tasks. Depending upon this pre-assigned priority, the tasks are assigned to initiators or interrupt levels within the computer.

Interrupt Servicing

A process control computer may have any number of levels of interrupts. Each level has associated with it a priority. When the computer is operating and an interrupt request occurs that has a higher priority than the current task which the computer is executing, the current task is interrupted with the contents of all registers saved and the higher priority task enters execution. When the higher priority task has been completed the interrupted task registers are restored and execution continues from the point of interruption. If an interrupt request occurs that has lower priority than the current task which the computer is executing, there is no interruption of the higher priority task to service the lower priority interrupt request. Instead when all higher priority requests have been serviced, it will be serviced. An interrupt request of the same priority level of a current task will not interrupt the current task but will be serviced later.

To illustrate the operation of interrupts refer to Figure 3-1. The computer is in the idle state at the start of this example, which is graphically represented at the top of the time line. As time passes, one moves down the time line until Interrupt 1 occurs. The computer responds by servicing this interrupt since none of higher priority are

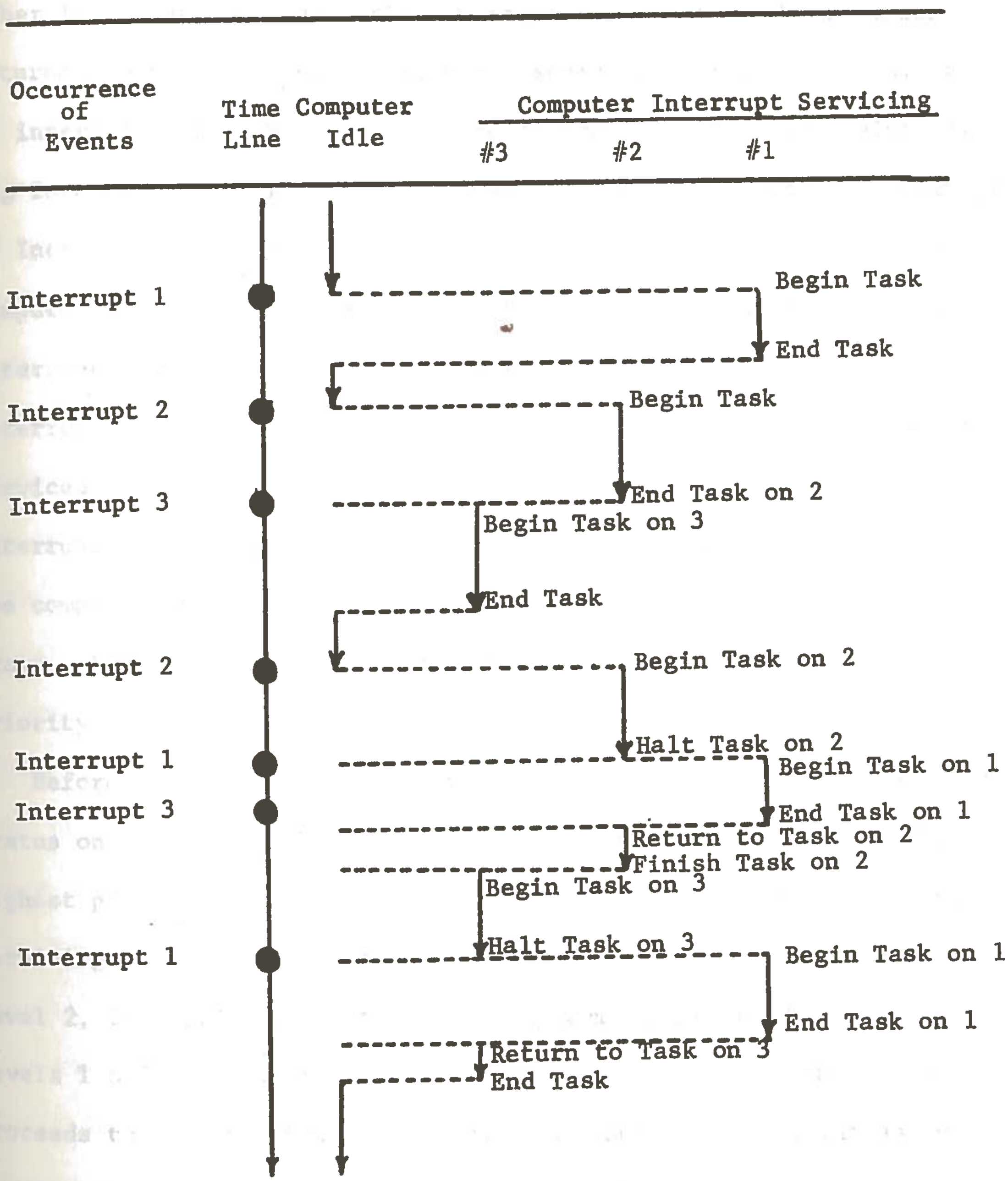


Figure 3-1

Interrupt Operation

utilizing the computer. When this servicing is completed, since no other interrupts of lower priority require servicing, the computer is returned to the idle state. As time passes, Interrupt 2 occurs. Since no interrupts of higher priority are active, the computer begins servicing Interrupt 2. Before the computer has finished servicing Interrupt 2, Interrupt 3 occurs. Interrupt 3 does not receive control of the computer now since an interrupt of higher priority has control. When Interrupt 2 is completed the computer gives control to level 3 since no interrupt of higher priority is active and Interrupt 3 is waiting to be serviced. Since no other interrupts occur during the servicing of Interrupt 3, the level is not interrupted and when service is completed, the computer is returned to the idle state. Later Interrupt 2 occurs again. Since the computer is in the idle state, there are no high priority interrupts active and the computer gives control to Interrupt 2. Before level 2 can finish, Interrupt 1 occurs. The computer saves status on level 2 and begins servicing Interrupt 1 since it is the highest priority active in the computer. Interrupt 3 occurs during the servicing of Interrupt 1, but since level 3 is lower than level 1 or level 2, it will have to wait until the completion of the servicing of levels 1 and 2. Interrupt 1 has now been completed and the computer proceeds to give control to Interrupt 2, since it is waiting and now has the highest priority. When the computer finishes servicing level 2 it will find that level 3 now has the highest priority and gives control to Interrupt 3. During the servicing of level 3, Interrupt 1 occurs and the computer gives control to level 1 putting level 3 in the wait state. When level 1 is completed, level 3 is restored and continues servicing from the point at which Interrupt 1 had occurred. When level 3 is completed there are no interrupts active and the computer again returns to the idle state.[8]

In general, the assignment of interrupts to their associated tasks is not a simple matter. In most cases, monitor interrupts have the highest priority, level 1. The next level, level 2, may be assigned to disastrous plant alarms. The groups that would be attached at this level are equipment shutdown tasks. This level could also include some type of logging function which after an emergency condition could, upon request, produce the record of events during the emergency condition. With this information the operator should be able to decide what caused the condition. On level 3, there could be some type of timing function or clock task which would be the scheduler for programs that need to run on a cyclic base. The different control programs would be attached to different interrupts within level 3. When the clock task detects that it is time for one of the tasks to run, the clock task turns on the interrupt associated with the tasks that need servicing. On level 4, there could be a task that allows the human operator to communicate with the control computer. The usual name for this task is operators console. This interface relationship between the human and the computer is very important. The function of operators console is to allow the operator to use the computer to expand the control capability.

CHAPTER IV

DIGITAL DATA ACQUISITION AND CONTROL PACKAGE

The Function of the Package

This package performs the necessary digital data acquisition and data manipulations required to accomplish a set of control tasks. The package is connected to an interface through which process variables are communicated. The process in this study was simulated on an analog computer. The process need not be simulated should the real process be accessible to the computer. The interface communicates signals between -10 and +10 volts. The analog computer operates with signals in the range of -10 to +10 volts. Should the real process not have signals initially in this range, the signals can be transformed into the required voltage. The real process variables are then connected into the interface instead of the simulated process variables. Thus the researcher can study a simulated process or a real process with equal ease.

The Description of the Package

The package was designed with further expansion in mind. This capability for further expansion stems from its modular structure, which was designed at the outset of the project. The structure consists of several modules each composed of groups of related tasks. The basic modules required are: 1) time module, 2) process-computer interface module, 3) control module, 4) human-computer interface module and 5) intra-structure communication module; refer to Figure 4-1.

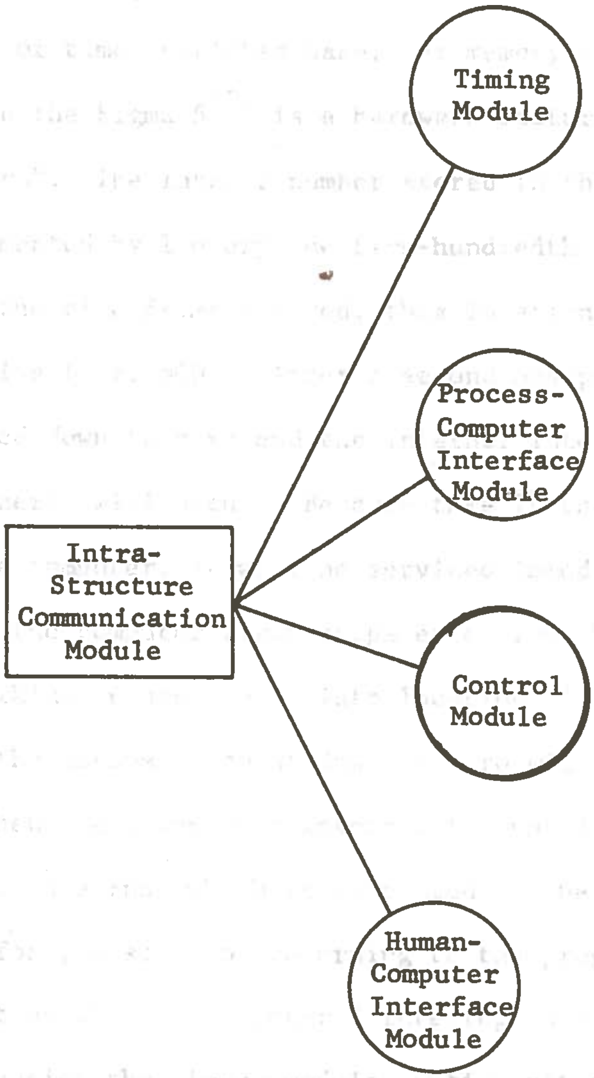


Figure 4-1

Structure of Hybrid Package

Timing Module

The timing module consists of a clock function for counting elapsed time and a triggering function for initiating all tasks that run on a cyclic or time initiated base. At memory location X'5A' (hexadecimal) in the Sigma 5[9] is a hardware feature called "counter 3 interrupt if zero". The integer number stored in this location is automatically decremented by 1 every one five-hundredth of a second. Depending upon the time frame desired, this location is initially set at a certain value (e.g. 500). After 1 second has passed the location will have counted down to zero and the internal interrupt from "counter 3 interrupt if zero" will occur. Because this is the highest priority interrupt in the computer, it will be serviced immediately. To service this interrupt, the computer first stops executing whatever program it is currently working on and stores into location X'5A' the same value as before so that the automatic count-down to zero will occur again the next second. Then the computer triggers interrupt level X'60', the highest external interrupt which is connected to the timing module routine. Therefore, instead of returning to the program which it may have been executing when the counter 3 interrupt occurred, the computer next begins executing the timing module routine after first saving the necessary registers to permit it to return to the interrupted program later. The timing module updates the time-of-day, a record it is keeping, and decrements by 1 second all the contents of locations associated with the time-to-run (ITRUN) table, which keeps track of when each task that runs on a cyclic base should be initiated. The timing module then checks to see if any of these tasks have a time-to-run equal to zero. If so the interrupt to which the task is connected is triggered by the timing module and the task run interval is stored in the time-to-run table. None of these triggered tasks start immediately because the timing module routine is running at the highest priority external interrupt level. When it is finished with the time-to-run table, the timing module interrogates the intra-structure communication module to determine if any control programs need servicing. These control programs could need servicing because of either unusual conditions in the process detected by the process-computer interface or by request from other modules. If any need servicing, the interrupt levels associated with them are triggered. Therefore, the control programs may run on a cyclic time base and also on requests. This describes the present design of the timing module structure. The timing module is easy to add to or modify.

<u>Process-Computer Interface Module</u>

The process-computer interface module performs four functions and is connected to the second highest external interrupt level X'61'. These four functions are: 1) receive, 2) filter, 3) send, and 4) convert. In receive, the computer interrogates the process to determine the current values of the variables in the process which have been chosen as process inputs and therefore are patched into the Analog Digital Converters (ADC units). The digital values which the computer receives from the ADC units are normalized quantities (between -1 and +1). These normalized values must be multiplied by a scaling constant for each individual variable to obtain the value in engineering units. These scaling constants are read into a table as data when the package is first loaded into the computer. Thus the values in engineering units, E.U., have dimensions, whereas the values initially read from the process as normalized variables are undimensioned. The E.U. values next could be digitally filtered to reduce the effect of any random error or "noise" in the readings from the process. This filtering function was not implemented in this study since no need for it was detected. Later applications involving readings from actual instruments on pilot plant equipment will likely require filtering. The position to insert filtering has been denoted in the program listing should the need develop. The filtered E.U. values are stored in a table in the intra-structure communication module for access by other modules. Thus there exists in memory at all times while the package is in operation a table of the most recent E.U. values of the process inputs. The process-computer interface next checks to see if any other modules have entered any values to be sent to the process (setpoints, etc.). Should there be outputs to be sent to the process, the process-computer interface normalizes the E.U. values of the outputs then sends them to the process via the Digital to Analog Converters (DAC units). At this point the module has completed its function for this cycle.

<u>Control Module</u>

The control module has slots for eight control programs, one of which was implemented in this study. Expansion of the number of control slots beyond eight could be easily implemented. Each control program has associated with it a run interval since control programs normally run on a cyclic base. If the control program has been previously turned on, the time-to-run for the control program (ITRUNC) is being decremented each cycle by the clock module. The control module merely interrogates the time-to-run of each control program and executes those that have zero entries. Thus the control module itself runs on the shortest cycle when initiated by the timing module, and each control program is called at its appropriate time by the control module. All run interval values are stored in the intra-structure communication module and they can be changed with the human-computer interface module. When a control program needs to output information back to the process, it stores the value in the intra-structure communication module and sets a flag that indicates the value is new and awaiting transmission to the process. The next time the process-computer interface module runs, the set flag is detected and the value is sent to the process. The control module is connected to both external interrupt level X'62' and X'63'. Should a control program be requested to run by another control program or by any module, interrupt level X'62' will be activated. If the control is requested by the timing module, interrupt X'63' will be activated. This allows control programs with a higher need to run than the normal cyclic operation to cut in line. Control programs also have built-in priorities with respect to each other depending upon the order in which they are interrogated by the control module.

Human-Computer Interface Module

The human-computer interface is located at the lowest priority in the interrupt structure because of the relatively long response time of a human operator. The computer can respond to an operator request via the human-computer interface module and still perform all the higher priority tasks described above, normally without the human operator detecting a lull in the transmissions. Were this module given higher priority, the human operator-computer communication would not increase noticeably, but the computer would spend excessive time in the idle state since it cannot service lower level interrupts until higher levels are completed. Consequently, many of its tasks would simply not be executed. Instead, the priority structure assigning the human-computer interface module the lowest priority results in the best overall performance. The human-computer module is termed the "operators console program" by most users in industry. The operators console program should be broad enough in scope so that the human operators abilities to interpret and run the plant are improved, not hindered. To accomplish this task effectively, the module performs two basic functions. The first gives the human operator the capability of changing values stored in memory locations with the computer, and the second gives him the capability of looking at listings of values stored in the computer. The human operator would be required to remember the locations of the values and exactly what the value had to be to accomplish the desired action if these two module functions were not expanded. In the expanded form implemented in this study, instead of only one general purpose memory change function and one general purpose memory printout function, there are a number of specific change and print functions designed to change or print out specific types of quantities. Each function is requested by the operator by entering three codes into the computer. The first code, IFUNCOD (Function Number), identifies the function. The second code, IDPT (ID point), identifies a sub-element within the specific function and the third code, VALUE, when applicable designates the new value to be entered into memory. When a human operator attempts to change any value by more than five percent, the operator console program requires a verification from him before the change is affected. The operators console functions implemented in this work are listed in Figure 4-2. The values that the operators console program can change or print out are stored in the intra-structure communication module.

Intra-Structure Communication Module

A Fortran programmer would recognize the intra-structure communication module as "Common". Common is a storage area that can be used by the different modules, but it is not located within the modules. In this study, values that are parameters are stored in common in the CT (control table). These parameters can be changed or printed out by an operators console function. The values that are variables, i.e., values that are either read from the process or generated by a module, are stored in common in the VT (variable table). These values cannot be changed directly by the human operator, but they can be printed out. Typical CT and VT constants are listed in Figure 4-3 and 4-4 along with the meaning of each element.

The Use of the Package

The process if simulated is patched on the analog board. The inputs and outputs to the simulated or real pilot plant are selected and patched into the interface section on the analog board (ADC units and DAC units). With knowledge of the inputs and outputs, the parameter cards are punched and added to the program deck. The program is then loaded into the computer. The setup of the deck will not be discussed here since appropriate comment cards are included within the listing of the program deck in the appendix. The parameter cards which follow the program deck are also shown in the listing in the appendix. They are described by comment cards in the listing and their formats are defined by format statements in the program.

FUNCTION	IDPT	VALUE	DESCRIPTION
01	Idpt	Value	Update of Constants in Control Table (Idpt) to (Value)
02	Idpt	Value	Update of Task Run Interval (Idpt) to (Value)
03	Idpt	Value	Update of Alarm Lower Limit (Idpt) to (Value)
04	Idpt	Value	Update of Alarm Upper Limit (Idpt) to (Value)
05	Void	Void	List on Logging Device Alarm Limits
06	Idpt	Void	Change Name of Analog Input (Idpt)
07	Void	Void	List on Logging Device Names of Analog Inputs
08	Idpt	Void	Start Trending Log, Idpt = Number of Variables to be Trended
09	Void	Void	Stop Trending Log
10	Void	Void	Add One Variable to Trend Log (Max.9)
11	Idpt	Void	Delete One Variable From the Trending Log, Idpt = the Number of the Variable Deleted
12	Void	Void	To Give Clock Correct Time of Day
13	Idpt	Void	To Turn on a Control Program (Idpt)
14	Idpt	Void	To Turn Off a Control Program (Idpt)
15	Void	Void	To List on Logging Device Status of Control Programs
16	Idpt	Void	To Allow Control Program to Print Messages
17	Idpt	Void	To Turn Off Messages From Control Program

Figure 4-2

Operator Console Functions

CONTROL TABLE

IDPT	VALUE	DESCRIPTION
1	1.0	Maximum Value Neutron Flux
2	20000.0	" " Avg. Fuel Temp. F
3	1000.0	" " Temp. Out of Core F
4	1000.0	" " " " " Mixing F
5	1000.0	" " " in Exch. — F
6	1000.0	" " Avg. Temp. Exch. F
7	1000.0	" " Cool in Core F
8	1000.0	" " Temp. Out Exch. — F
9	1000.0	" " Inlet Mixing F
10	1000.0	" " T Inlet Core F
11	1.0	" " Control Rod Position
12	10000.0	" " Steam Temp. — F
13	1000.0	" " Temp. Avg. Compar. F
14	1000.0	" " Error Temp. F
15	1.0	
16	1.0	
17	0.005	" " Del K
18	1000.0	" " Temp. Ref. F
19	1.0	
20	0.0	
21	0.0	
22	0.0	
23	0.0	
24	0.0	
25	1000.0	" " Setpoint Controller F
26	1.0	" " Digital Load Change
27	0.7	Position of Valve on Heat Exchanger
28	0.0	
29	0.0	
30	0.0	
31	0.0	
32	0.0	
33	0.0	
34	0.0	
35	0.0	
36	0.0	
37	400.0	Temp. Steam Target F
38	0.1	KP For CONTR1
39	0.05	KI For CONTR1
40	0.0	

Figure 4-3.1

Control Table

CONTROL TABLE

IDPT	VALUE	DESCRIPTION
41	60.0	CONTR1 Run Interval Seconds
42	60.0	CONTR2 " " "
43	60.0	CONTR3 " " "
44	60.0	CONTR4 " " "
45	300.0	CONTR5 " " "
46	300.0	CONTR6 " " "
47	300.0	CONTR7 " " "
48	300.0	CONTR8 " " "
49	0.0	
50	0.0	
51	1.0	Flag to Print Cont. Prog. 1 Run
52	1.0	" " " " " 2 "
53	1.0	" " " " " 3 "
54	1.0	" " " " " 4 "
55	1.0	" " " " " 5 "
56	1.0	" " " " " 6 "
57	1.0	" " " " " 7 "
58	1.0	" " " " " 8 "
59	0.0	
60	400.0	Temp. Setpoint F

Figure 4-3.2

Control Table

VARIABLE TABLE

IDPT	VALUE	DESCRIPTION
1	Current Values	Current Value(EU) of Variable Described CT(IDPT)
2	" "	" " " " " "
3	" "	" " " " " "
4	" "	" " " " " "
5	" "	" " " " " "
6	" "	" " " " " "
7	" "	" " " " " "
8	" "	" " " " " "
9	" "	" " " " " "
10	" "	" " " " " "
11	" "	" " " " " "
12	" "	" " " " " "
13	" "	" " " " " "
14	" "	" " " " " "
15	" "	" " " " " "
16	" "	" " " " " "
17	" "	" " " " " "
18	" "	" " " " " "
19	Available for future use	
20	" " " "	
21	" " " "	
22	" " " "	
23	" " " "	
24	" " " "	
25	" " " "	
26	Current Values	Position of Valve on Heat Exchanger
27	Available for future use	

Figure 4-4

Variable Table

CHAPTER V

ACTION OF A DIGITAL SUPERVISORY CONTROL PROGRAM

Digital Control Program

The digital control program implemented in this study demonstrates the utility of the hybrid package for making studies of various proposed digitally controlled systems. The cascaded control system included in the analog simulation of the PWR is not capable of maintaining constant steam temperature when a load disturbance occurs in the system. With the addition of a supervisory digital control program to maintain constant steam temperature the total system adjusts automatically to load disturbances and returns the steam temperature to its desired value. The objective of maintaining constant steam temperature stems from the characteristics of turbines. Turbines perform better if they are operated at constant pressure.[10] Since the steam in this model is considered to be saturated, constant pressure means constant temperature. Therefore the objective of better performance of the steam turbines is met by maintaining a constant temperature steam flow leaving the heat exchanger.

Each time it runs, the digital control program uses the velocity form of the PI control algorithm to compute an adjustment to the manipulated variable (temperature setpoint for the analog cascaded PI controller) based on the error in the controlled variable (desired steam temperature — actual steam temperature). The velocity algorithm can be expressed as:

$$M = K_p(EN-EO) + K_I(EN)$$

where M = Change in the manipulated variable

K_p = Proportional constant

K_I = Integral constant

EN = Error now

EO = Error at previous time increment

There was no attempt made in this study to develop optimal values for the two digital controller settings, the proportional constant and the integral constant.

Response of Process to Digital Supervisory Control

The above described digital supervisory control program was written and inserted into the digital package to illustrate the application of the digital control package to control an analog simulated process. As explained above, the digital control program, when turned on by the human operator through an operator console function, adjusts the setpoint of the analog cascade controller to maintain a constant temperature of the steam generated in the boiler. The graphs in Figure 5-1 to 5-4 represent the responses produced by the hybrid computer package to a ten percent load change. The load was varied by adjusting a parameter in the analog simulation representing the opening of a valve in the steamline to the turbine. The plotted information was obtained while the simulated process was running via the operators console function. Different responses were obtained by changing the digital controller proportional and integral constants. Values of the controller constants are listed on the individual graphs.

Figure 5-1 illustrates the response without the digital control program in operation. The steam temperature does not return to its original position before load change. Figure 5-2 illustrates the response with only proportional action. In this case the steam temperature does not have as much steady-state off-set from the set-point as in the case without any control. Figure 5-3 and 5-4 illustrate the response with both proportional and integral action. There is no permanent off-set if integral action is employed. Different values of integral action produce differences in the periodic time constant and the stability of the system. Increased integral action decreases the response time of the system, but at the same time decreases the stability of the system.

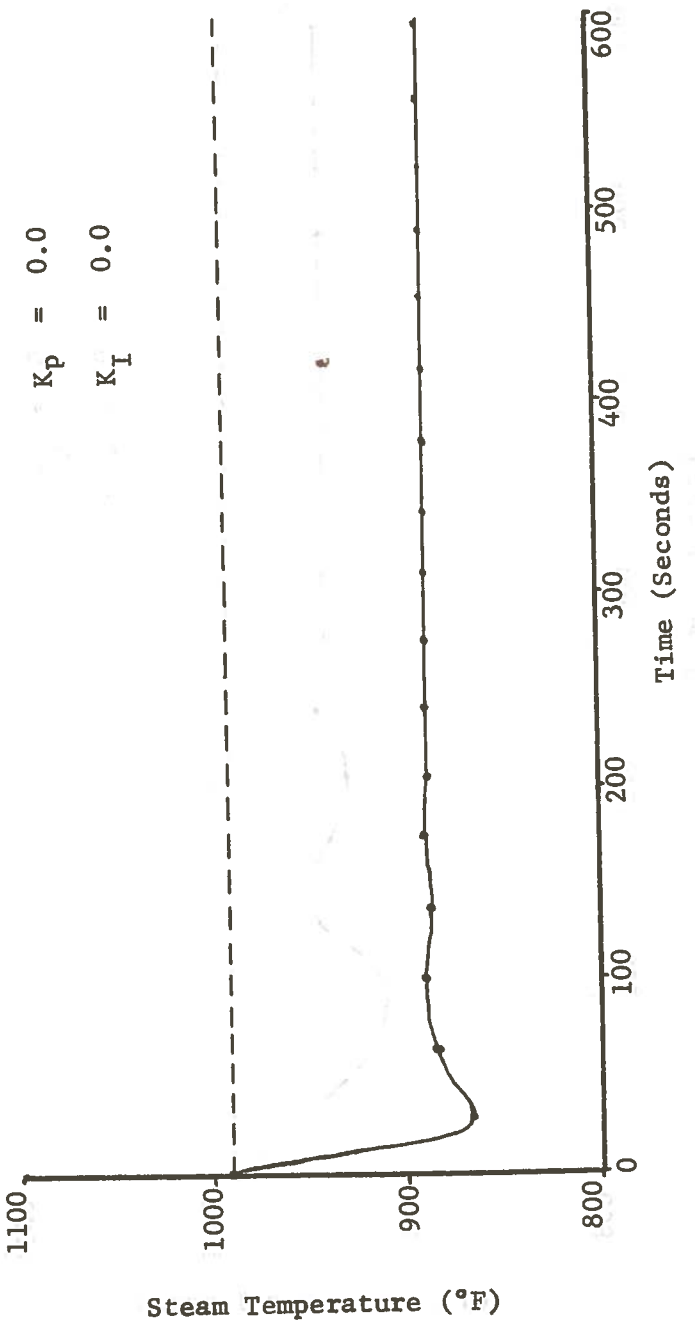


Figure 5-1

System Response

to Step Change in Steam Valve Opening

with no Supervisory Control

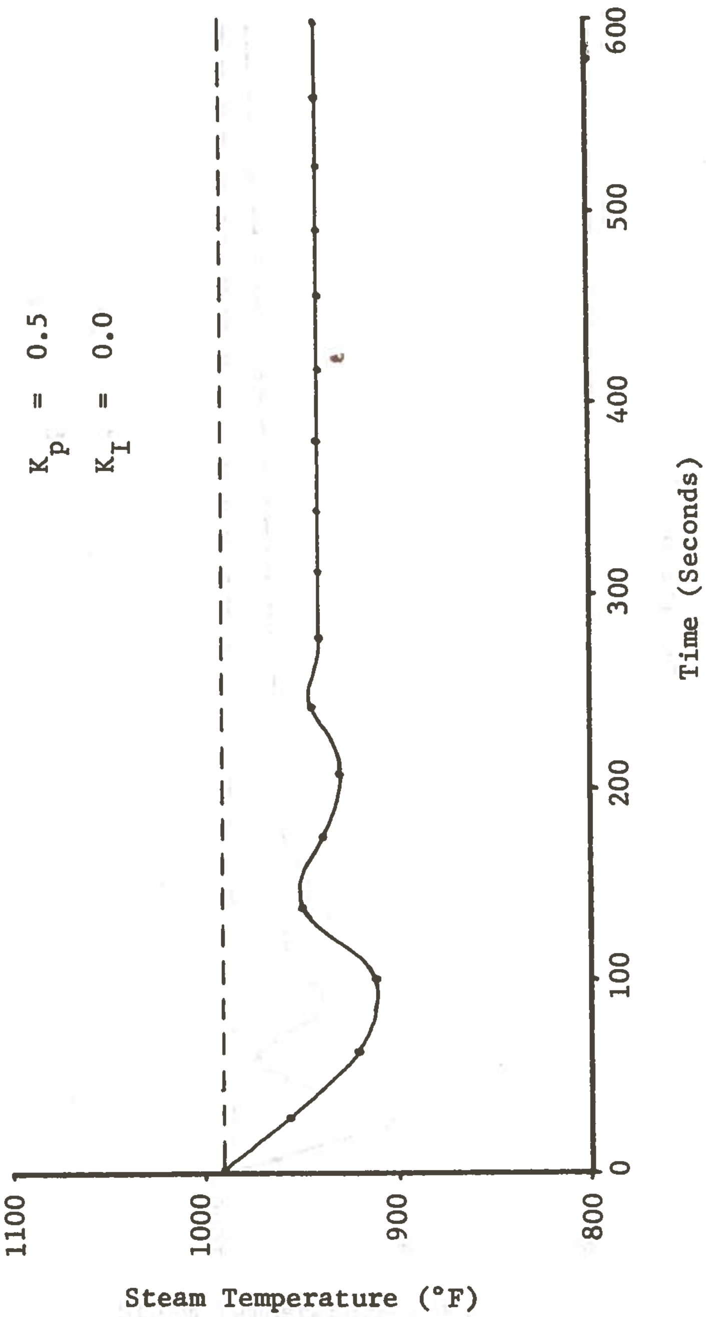


Figure 5-2

System Response

to Step Change in Steam Valve Opening

with only Proportional Control

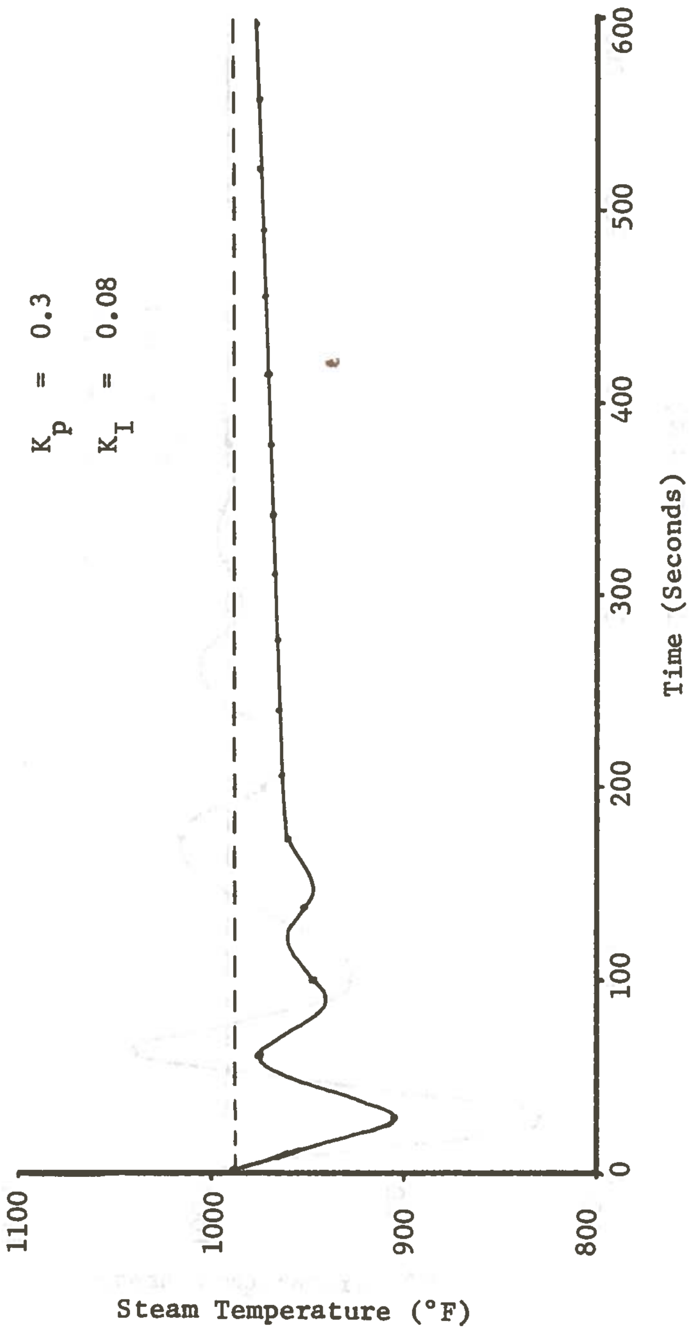


Figure 5-3
System Response
to Step Change in Steam Valve Opening
with Proportional and Integral Control

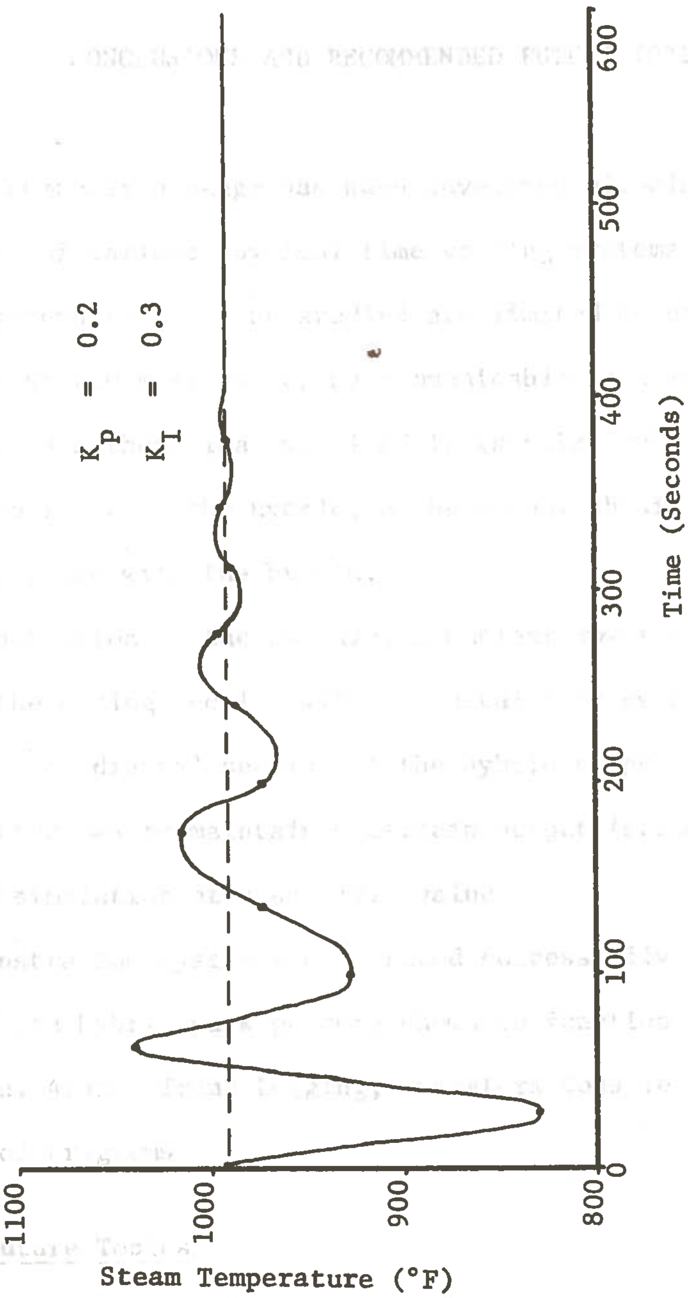


Figure 5-4

System Response

to Step Change in Steam Valve Opening

with Proportional and Integral Control

CHAPTER VI

CONCLUSIONS AND RECOMMENDED FUTURE TOPICS

Conclusions

A hybrid computer package has been developed allowing the study of many aspects of various physical time varying systems. The variety of physical systems that can be studied are limited by only two constraints. The system must either be communicable in a mathematical sense such that a mathematical model of it (simulation) can be patched on the analog portion of the hybrid, or be actual physical processes that can communicate with the hybrid.

For demonstration of the package, a nuclear power reactor was simulated on the analog section with a digital process control computer implemented on the digital section of the hybrid computer. The digital control objective was to maintain a certain output (steam temperature) of the analog simulation at a specific value.

The demonstration system was operated successfully, and all components of the hybrid package were shown to function as designed, including Scan, Alarm, Trend Logging, Operators Console Functions, and Digital Control Programs.

Recommended Future Topics

The hybrid package could be expanded in various ways to yield an even more powerful research tool.

The capability of maintaining in the computer's auxiliary memory a historical record recallable on demand (e.g. 24 hours) of all the data being generated would produce for the researcher a more complete picture of what was occurring.

At present when the package is running both the analog and digital machines are dedicated to the hybrid package. The digital section of the hybrid package could be further developed so as to time-share the digital computer with other digital programs. The use of the hybrid computer to control either simulated or analog processes would not interfere greatly with normal batch-type use of the digital computer.

For any specific study, additional control programs could be developed relying on more advanced control techniques. Two such techniques are Feedforward Control and Adaptive Control. Feedforward control generates control actions based upon measurement of inputs to the process instead of the controlled variable. Should the inputs change, a feedforward control program computes the corrective action needed to maintain the controlled variable at the desired value.(11) Adaptive control is defined as a system which is provided with a means of continuously monitoring its own performance in relation to a given index of performance and modifying its own parameters by closed loop action so as to approach optimal performance.(12)

Should the control programs exceed the available core storage, the structure could be modified to allow the control programs to reside in auxiliary storage. The programs could then be brought into core when needed (e.g. overlay structure).

LITERATURE CITED

(1) Collins, Paul F., Skovholt, Donald J., Operator Licensing Experience Involving Nuclear Power Plant Simulators, paper presented at American Nuclear Society in Washington, D.C. on November, 1970, by Division of Reactor Licensing, U.S. Atomic Energy Commission.

(2) Lewis, James G., Application of On-Line Computer System at Big Rock Point Nuclear Plant, paper presented October 4-7, 1965, before the 20th Annual ISA Conference and Exhibit, Los Angeles, California. Preprint Number 34.1-3-65.

(3) Bevilacqja, F., Yuile, E., Application of Digital Computer Systems in Nuclear Power Plants, paper presented October 23-25, 1968, at 15th Annual IEEE/GNS SYMPOSIUM, Montreal, Canada.

(4) Carlson, A., Hannauer, G., Carey, T. and Holsberg, P. J., Handbook of Analog Computation, 2nd Ed., pp. 1-5, Electronic Associates, Inc., New Jersey, 1967.

(5) McLeod, John, Editor, Simulation, pp. 1-14, McGraw Hill Book Co., New York, 1968.

(6) Electronic Associates, Inc., Simulation of the Primary Loop of a Nuclear Power Plant with a Small General Purpose Analog Computer, Application Study 13.4.2a, 1964.

(7) Savas, Emanuel S., Computer Control of Industrial Processes, McGraw Hill Book Co., New York, 1965.

(8) Smith, Cecil L., Digital Control of Industrial Processes, Computing Surveys, Vol. 2, No. 3, pp. 211-41, September, 1970.

(9) Xerox Data Systems, 701 South Aviation Boulevard, El Segundo, California, 90245, Sigma 5 (Model).

(10) Schultz, M. A., Control of Nuclear Reactors and Power Plants, pp. 130, McGraw Hill Book Co., 1955.

(11) Murrill, Paul W., Automatic Control of Processes, International Textbook Co., 1967.

(12) Williams, Theodore J., Present Status and Outlook for Adaptive and Learning Control in Industrial Computer Control System, Paper presented June, 1970, at Annual Meeting, 1970, of DECHEMA, Deutsche Gesellschaft Fur Chemische Apparatewesen, Frankfurt/Main, Germany.

LIST OF SYMBOLS

SYMBOL	MEANING	UNITS
A	Heat transfer area in the reactor	ft^2
A_x	Heat transfer area in the steam generator	ft^2
A'	Cross sectional flow area	ft^2
C_c	Specific heat of the coolant	Btu/lb-°F
C_f	Specific heat of the fuel, moderator,etc.	Btu/lb-°F
C_j	Concentration of neutrons from delayed neutron group "j"	Neu/sec
C	Fluid specific heat	Btu/lb-°F
k	Reactivity constant	Dimensionless
K	Control rod drive unit gain	ft/sec
K_c	Controller gain	$\frac{\text{neutrons}}{°\text{F-sec-cm}^2}$
M_c	Mass of coolant in the reactor	lbs.
M_f	Mass of fuel, moderator, etc.	lbs.
M_i	Mass of collant in the inlet plenum chamber	lbs.
M_o	Mass of coolant in the outlet plenum chamber	lbs.
M_x	Mass of coolant in the steam generator	lbs.
S	Operator	Seconds^{-1}
T_a	Ambient temperature	°F
T_c	Average coolant temperature in the reactor	°F
T_f	Average fuel temperature	°F
T_i	Coolant temperature at the input to the reactor inlet plenum chamber	°F
T_o	Coolant temperature at the outlet plenum chamber, or temperature co-efficient of reactivity reference temp-erature (temperature of which temp-erature contribution is zero).	°F

SYMBOL	MEANING	UNITS
T_s	Average temperature of secondary fluid in steam generator	°F
T_x	Average coolant temperature in the steam generator	°F
T_{ic}	Temperature of coolant entering the reactor	°F
T_{ix}	Temperature of coolant entering the steam generator	°F
T_{oc}	Temperature of coolant leaving the reactor	°F
T_{ox}	Temperature of coolant leaving the steam generator	°F
$T_{ref.}$	Reference temperature	°F
$T_{ave.}$	Average system temperature	°F
T	Fluid temperature	°F
U	Overall coefficient of heat transfer in the reactor core, or	$\frac{Btu}{sec\text{-}ft^2}$
	Control rod reactivity variable	Dimensionless
U_x	Overall coefficient of heat transfer in the steam generator	Dimensionless
U'	Overall coefficient of heat transfer	Dimensionless
V	Mean fluid velocity	ft/sec
W_c	Mass flow rate of coolant	lbs/sec
W	VA, mass rate of flow of the fluid	lbs/sec
X	Control rod position	Dimensionless
a	Heat transfer area per unit length of conduit	ft.
i	Index for delayed neutron groups	Dimensionless
l_i	Effective length of inlet piping system	ft.
l_o	Effective length of outlet piping system	ft.
l^*	Effective neutron lifetime	seconds

SYMBOL	MEANING	UNITS
n	Neutron density	Neutron/cu.cm.
n^*	Reduced neutron density	Dimensionless
n_m	Maximum practical neutron density	Neutron/cu.cm.
n_o	Demand-power level neutron density	Neutron/cu.cm.
t	Time	seconds
v_o	Mean velocity of coolant in outlet piping system	ft/sec.
V_i	Mean velocity of coolant in inlet piping system	ft/sec.
x	Position along conduit	ft.
α	Temperature coefficient of reactivity	$°F^{-1}$
β_j	Fraction of prompt neutrons appearing in delayed neutron group "j"	Dimensionless
δ	Reactivity	Dimensionless
δk_c	Reactivity contribution of control rod positions	Dimensionless
$\delta k_c(0)$	Initial reactivity contribution of control rods	Dimensionless
δk_f	Built-in reactivity of fuel	Dimensionless
δk_p	Reactivity contribution of reactor poisons	Dimensionless
δk_t	Reactivity contribution due to the fuel temperature	Dimensionless
ΔH_f	Heat of fission	$\frac{\text{Btu-cu.cm.}}{\text{sec. Neu}}$
$\epsilon(t)$	Error signal	°F
λ_j	Decay constant associated with group "j"	Dimensionless
$\mu(t)$	Departure of control rod reactivity from its initial value	Dimensionless
τ_o	Outlet piping system delay time	seconds

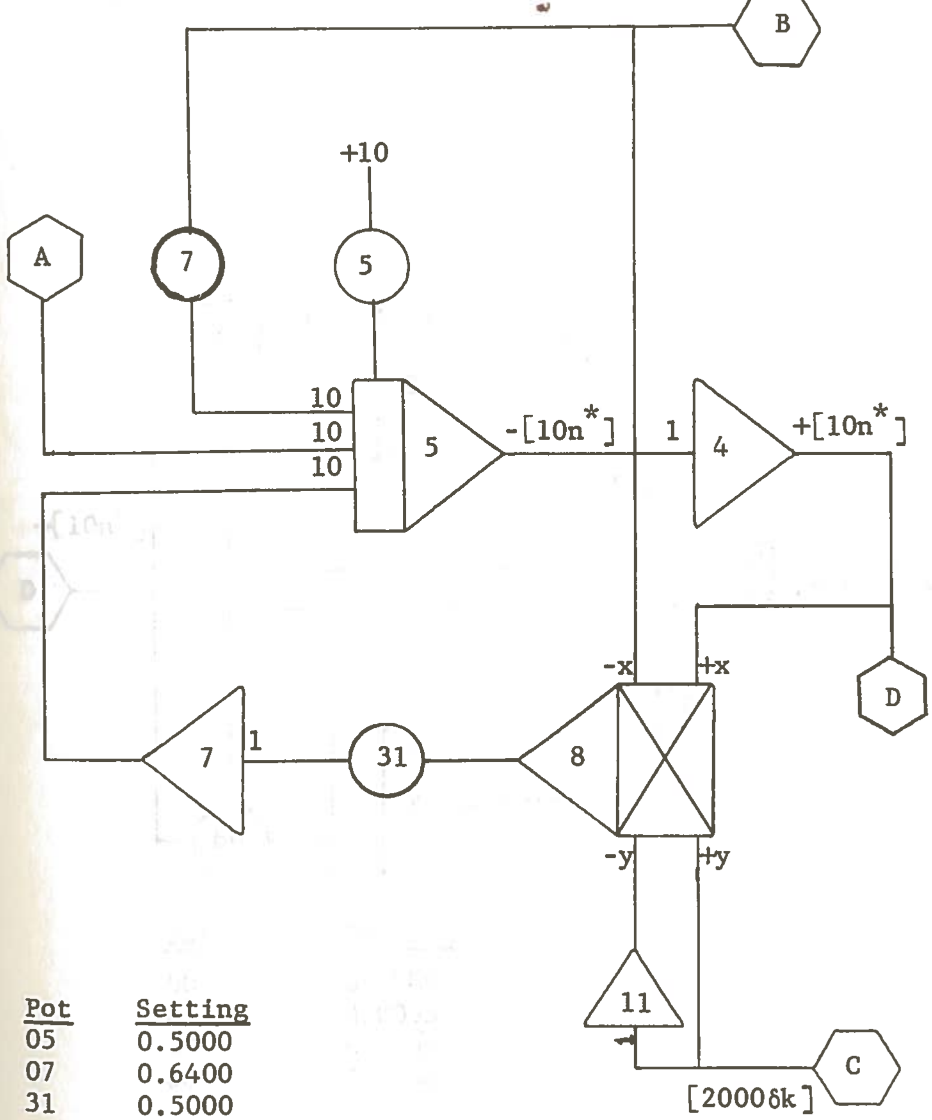
SYMBOL	MEANING	UNITS
τ_i	Inlet piping system delay time	seconds
τ_m	Control rod drive unit time constant	seconds
ρ	Fluid density	seconds

APPENDIX A

MODULE	OFF PAGE CONNECTORS
Reactor Kinetics	A B C D
Delayed Neutron Groups	A D
Fuel Element Heat Transfer	B E F
Fuel Element to Coolant Heat Transfer	E F G H
Out of Reactor Core	F G I
Outlet Plenum	I J
Piping Delay to Steam Generator	J K
Steam Generator	K L M
Out of Steam Generator	K L N
Piping Delay to Inlet Plenum	N O
Inlet Plenum	G O
Average Temperature	K N P
Comparator	P Q
PI Controller	D Q R
Control Rods	R S
Reactivity	E S T
Automatic Scram	D
Boiler-Power Demand	L M

Reactor Kinetics:

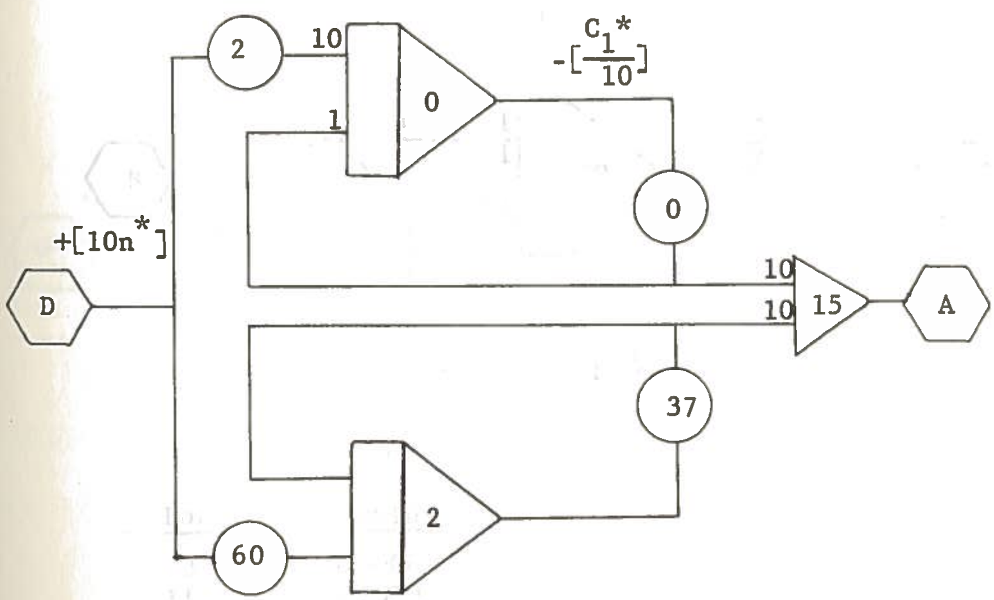
$$\frac{d}{dt}[10n^*] = (0.5)[2000\delta k]\,[10n^*] - 10(0.64)[10n^*] + 10\sum_{i=1}^{2}\lambda_i C_i^*$$



Pot	Setting
05	0.5000
07	0.6400
31	0.5000

Delayed Neutron Groups:

$$\frac{d}{dt}\left[\frac{C_i^*}{10}\right] = \frac{1}{10^2}\,\frac{\beta_i}{l^*}\,[10n^*] - \lambda_i\left[\frac{C_i^*}{10}\right]$$

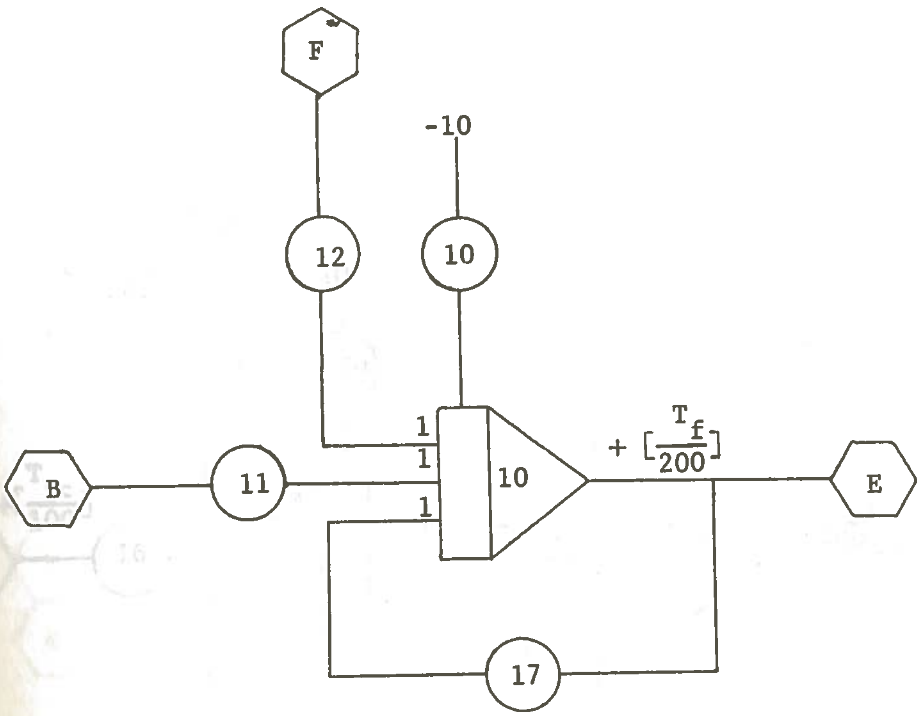


Pot	Setting
00	0.7580
02	0.0033
37	0.0950
60	0.0041

Fuel Element Heat Transfer:

$$\frac{d}{dt}\left[\frac{T_f}{200}\right] = 0.1\left(\frac{\Delta H_f}{200\ M_f C_f}\right)[10n^*] - \left(\frac{UA}{M_f C_f}\right)\left[\frac{T_f}{200}\right] +$$

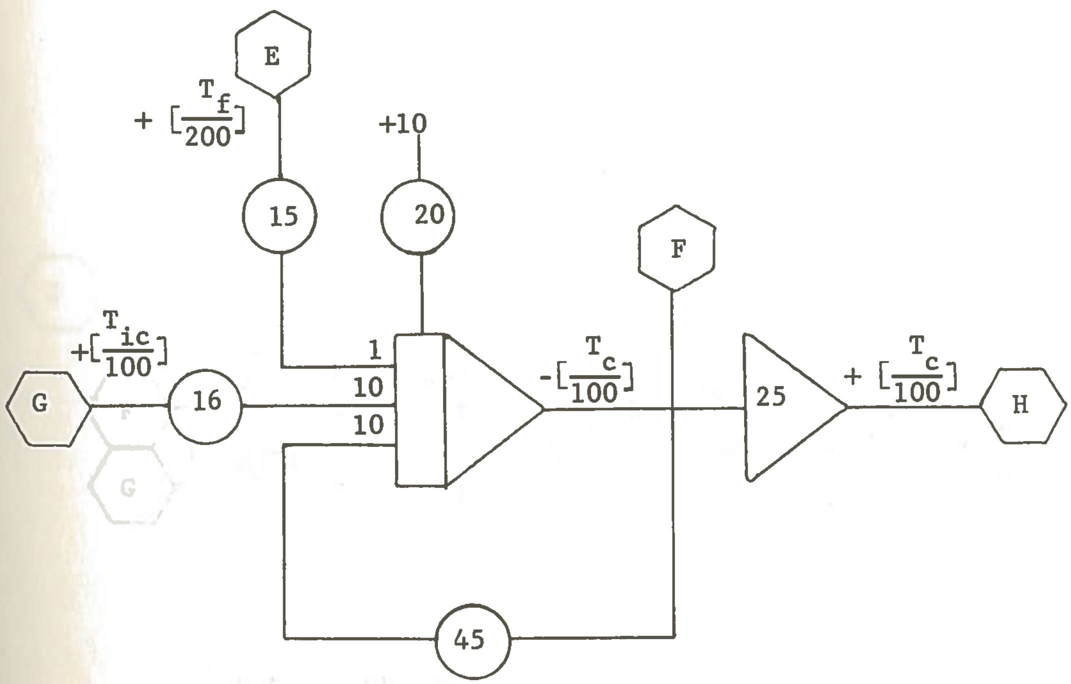
$$0.1\left(\frac{5UA}{M_f C_f}\right)\left[\frac{T_c}{100}\right]$$



Pot	Setting
10	0.5000
11	0.1000
12	0.1000
17	0.2000

Fuel Element to Coolant Heat Transfer:

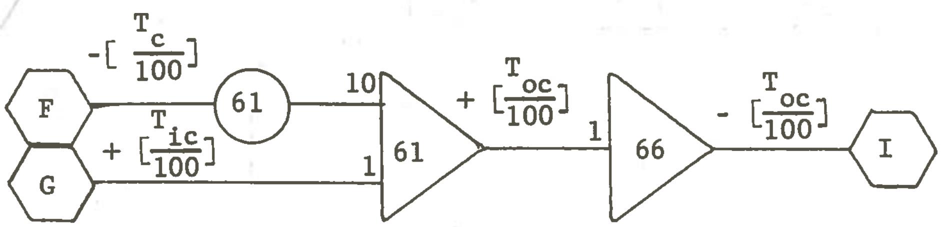
$$\frac{d}{dt}\left[\frac{T_c}{100}\right] = \left(\frac{2UA}{M_c C_c}\right)\left[\frac{T_f}{200}\right] - \left(\frac{UA + 2\,W_c C_c}{M_c C_c}\right)\left[\frac{T_c}{100}\right] + \left(\frac{2\,W_c C_c}{M_c C_c}\right)\left[\frac{T_{ic}}{100}\right]$$



Pot	Setting
15	0.9999
16	0.2500
20	0.5000
45	0.3000

<u>Out of Reactor Core</u>:

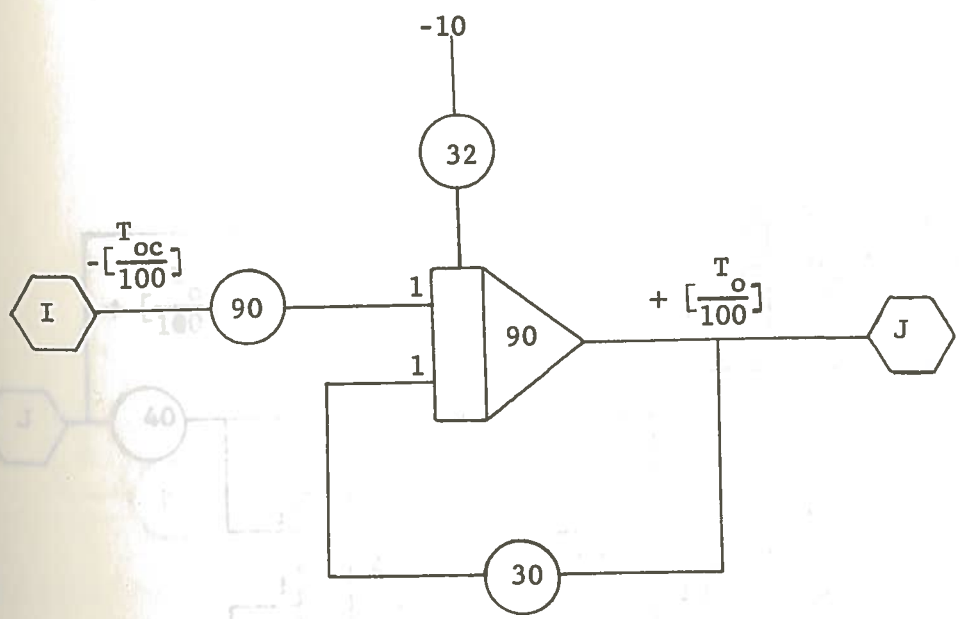
$$\left[\frac{T_{oc}}{100}\right] = 10(0.2)\left[\frac{T_c}{100}\right] - \left[\frac{T_{ic}}{100}\right]$$



Pot	Setting
61	0.2000

Outlet Plenum

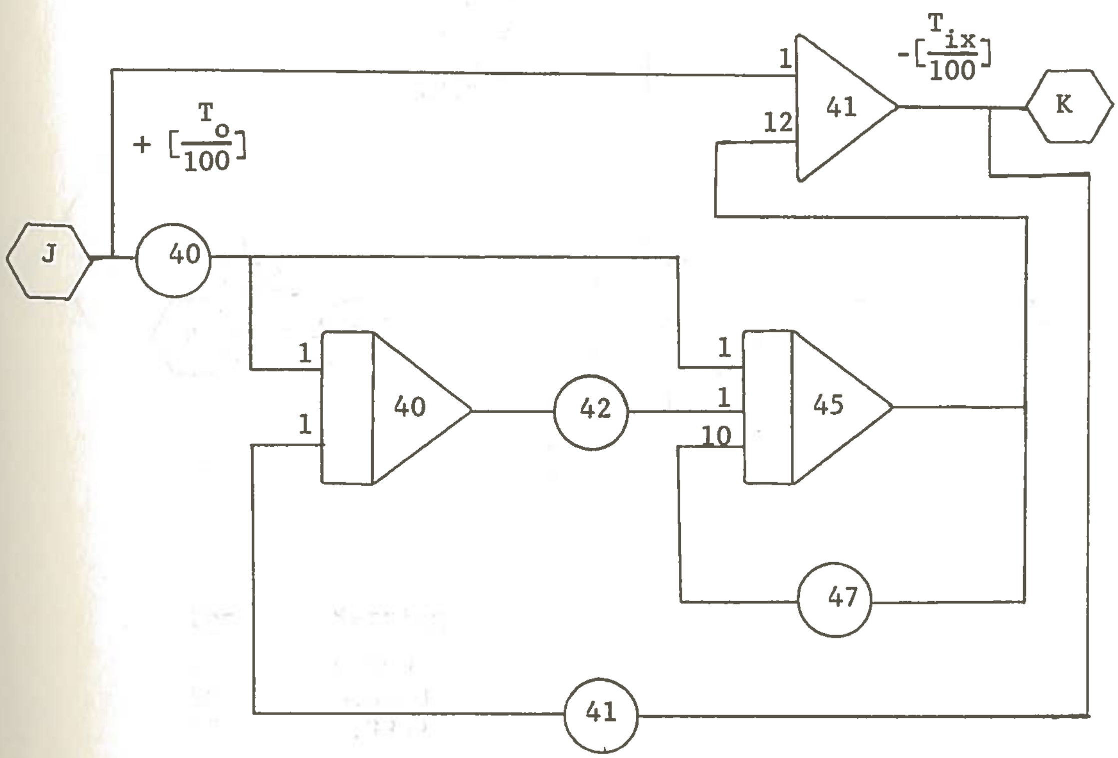
$$\frac{d}{dt}\left[\frac{T_o}{100}\right] = \left(\frac{W_c}{M_o}\right)\left[\frac{T_{oc}}{100}\right] - \left(\frac{W_c}{M_o}\right)\left[\frac{T_c}{100}\right]$$



Pot	Setting
30	0.5000
32	0.6000
90	0.5000

Piping Delay to Steam Generator:

$$\left[\frac{T_{ix}}{100}\right] = \left[\frac{T_o}{100}\right](t-D)$$

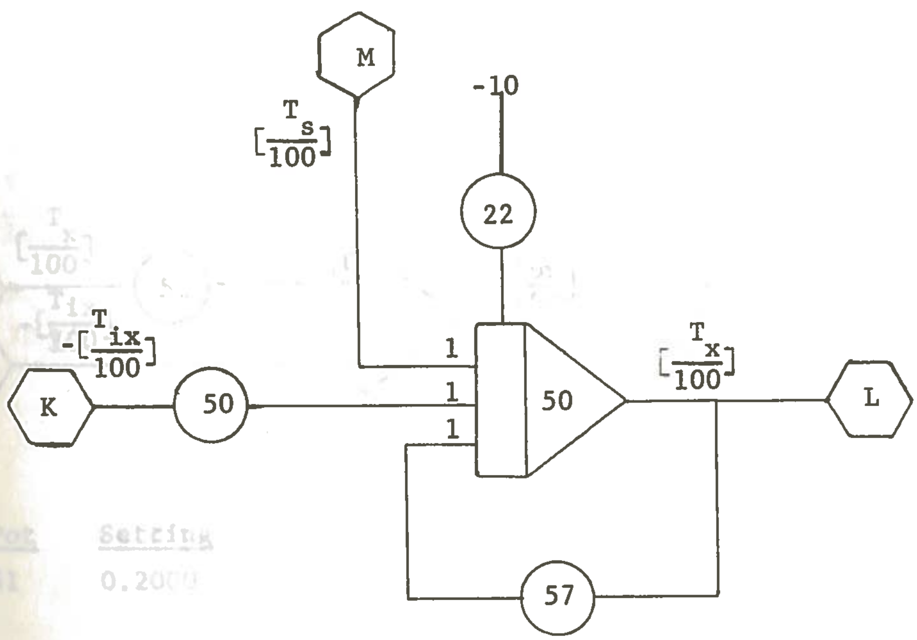


Pot	Setting
40	0.9999
41	0.9999
42	0.9999
47	0.6000

Steam Generator:

$$\frac{d}{dt}\left[\frac{T_x}{100}\right] = \frac{2W_c}{M_x}\left[\frac{T_{ix}}{100}\right] + \frac{U_xA_x}{M_xC_c}\left[\frac{T_s}{100}\right] - \frac{U_xA_x}{M_xC_c}\left[\frac{T_x}{100}\right] -$$

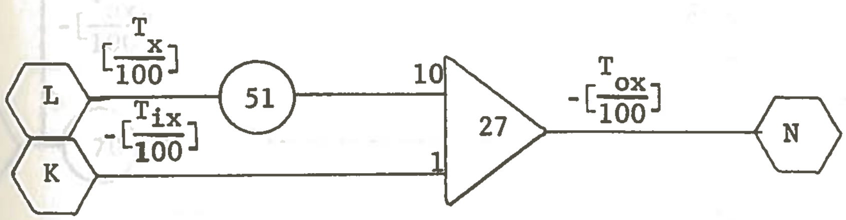
$$\frac{2W_c}{M_x}\left[\frac{T_x}{100}\right]$$



Pot	Setting
22	0.5000
50	0.6000
57	0.9999

Out of Steam Generator:

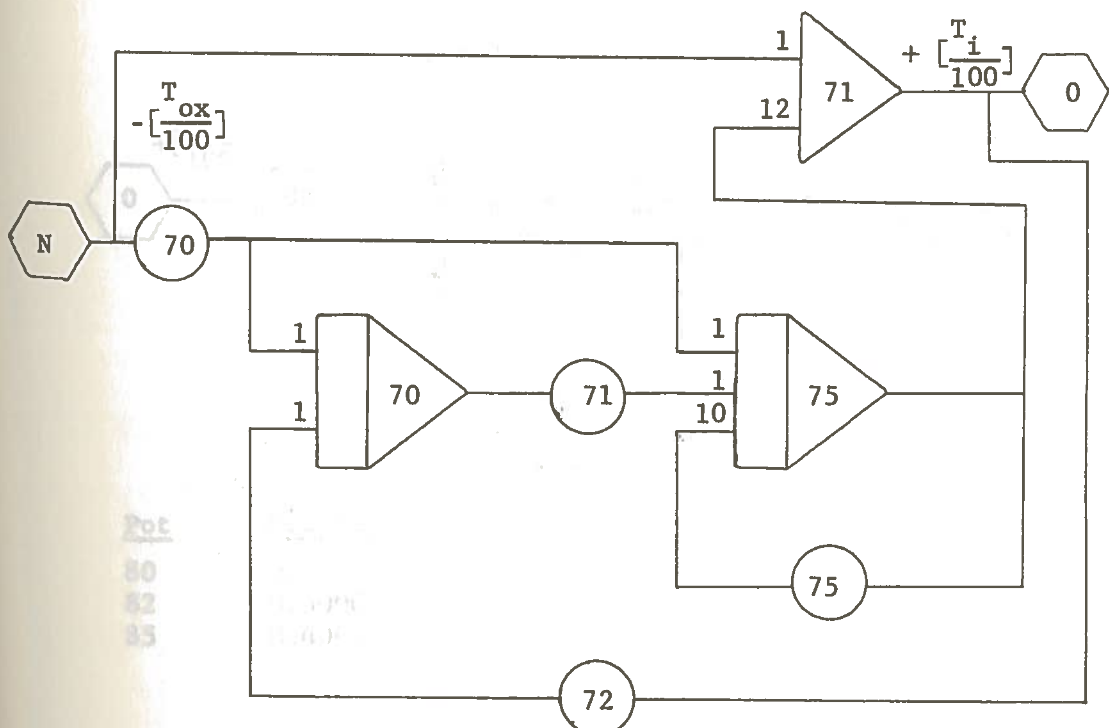
$$\left[\frac{T_{ox}}{100}\right] = 10(0.2)\left[\frac{T_x}{100}\right] - \left[\frac{T_{ix}}{100}\right]$$



Pot	Setting
51	0.2000

Piping Delay to Inlet Plenum:

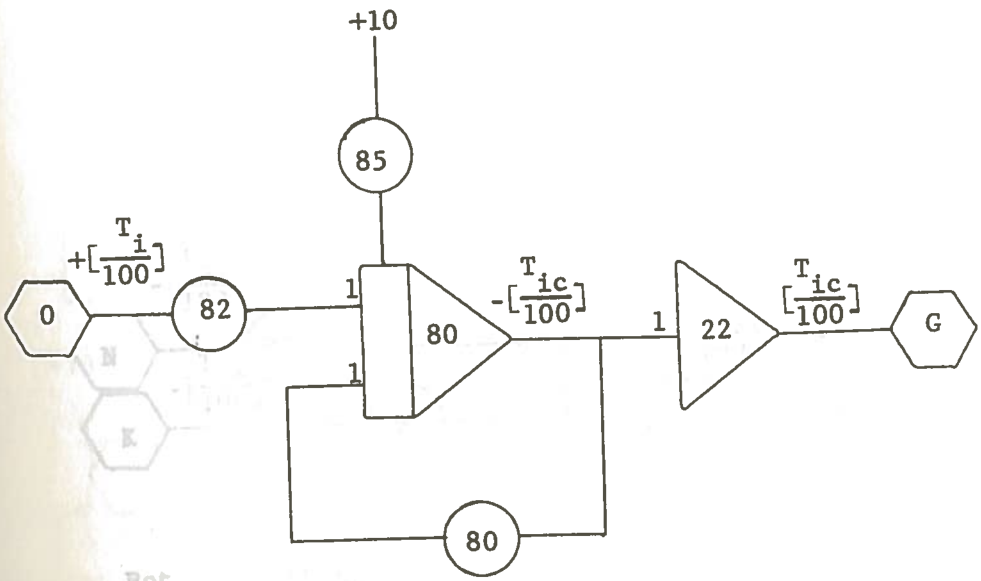
$$\left[\frac{T_{ix}}{100}\right] = \left[\frac{T_{ox}}{100}\right](t-D)$$



Pot	Setting
70	0.9999
71	0.9999
72	0.9999
75	0.6000

Inlet Plenum:

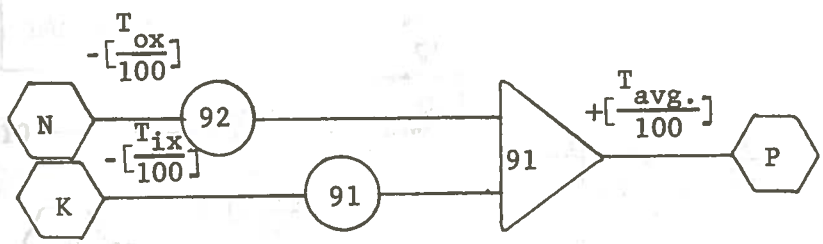
$$\frac{d}{dt}\left[\frac{T_{ic}}{100}\right] = \frac{W_c}{M_i}\left[\frac{T_i}{100}\right] - \frac{W_c}{M_i}\left[\frac{T_{ic}}{100}\right]$$



Pot	Setting
80	0.5000
82	0.5000
85	0.4000

Average Temperature:

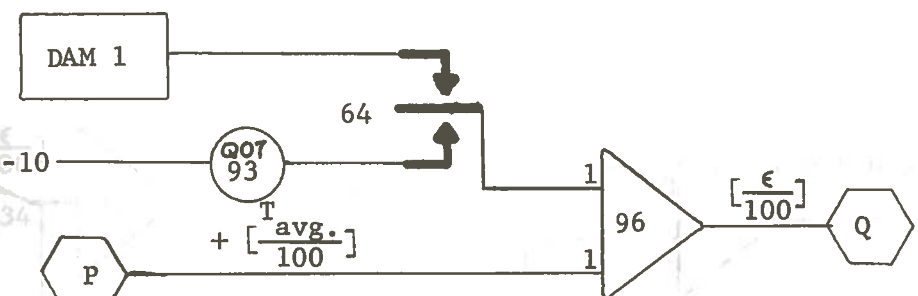
$$\left[\frac{T_{avg.}}{100}\right] = 0.5\left(\left[\frac{T_{ox}}{100}\right] + \left[\frac{T_{ix}}{100}\right]\right)$$



Pot	Setting
91	0.5000
92	0.5000

Comparator:

$$\left[\frac{\epsilon}{100}\right] = [10]\left(\frac{T_{ref.}}{1000}\right) - \left[\frac{T_{avg.}}{1000}\right]$$

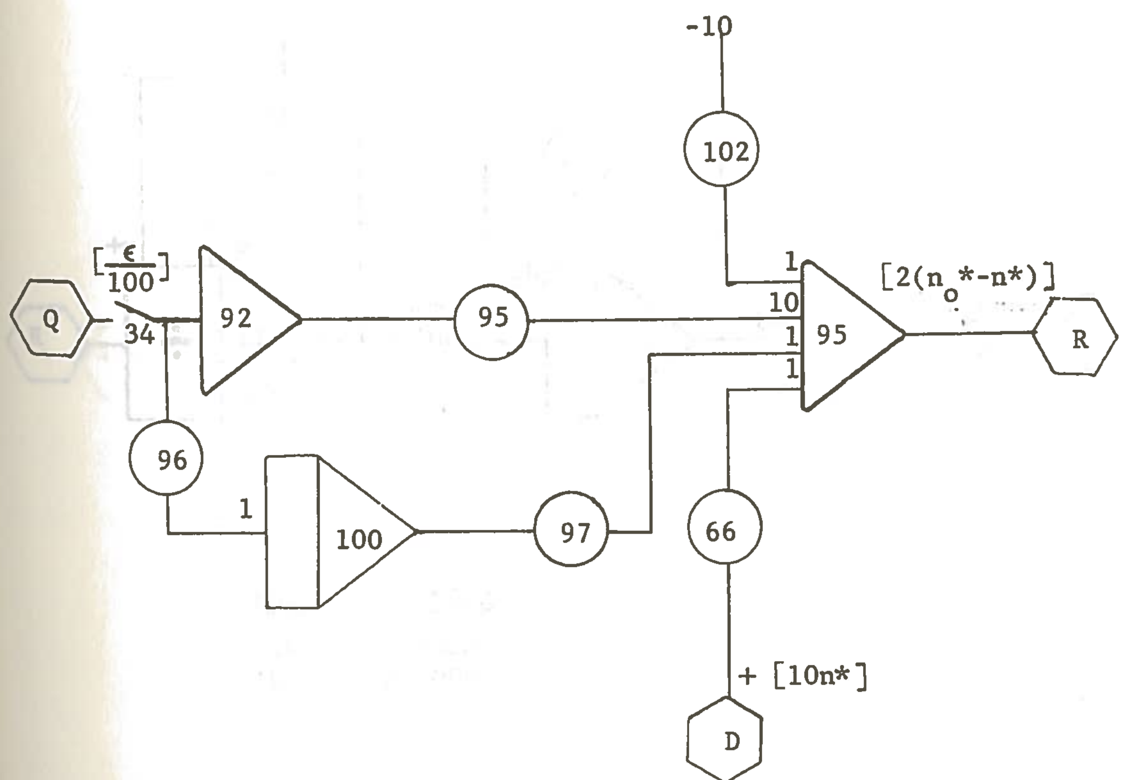


Pot	Setting
Q07	0.4000

PI Controller:

$$[2(n_o^*-n^*)] = (2\times10^3 K_c^*)\int_o^t \frac{1}{10}\left[\frac{\epsilon}{100}\right]dt + (2000\ K_c^*\tau_c)\left[\frac{\epsilon}{100}\right] -$$

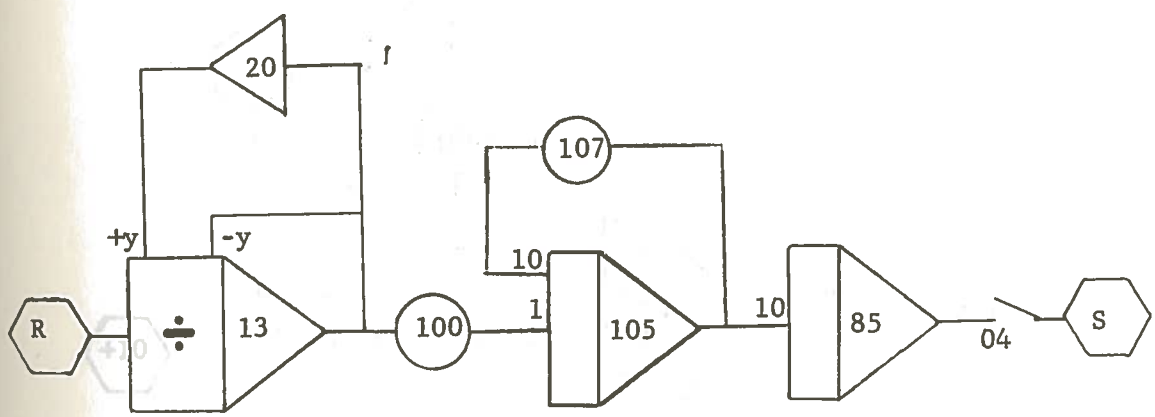
$$0.200[10n^*] + 2\ n_o^*(0)$$



Pot	Setting
66	0.2000
95	0.0200
96	0.1000
97	0.2000

Control Rods:

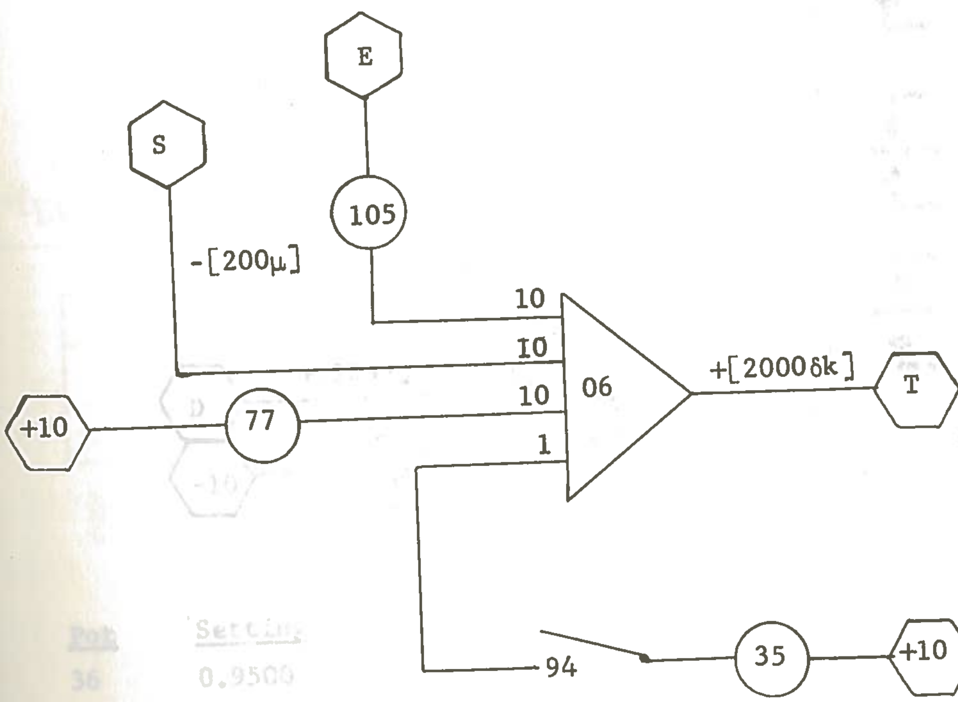
$$\frac{d^2}{dt^2}\,[2000\mu] + \left(\frac{1}{\tau_m}\right)\frac{d}{dt}\,[2000\mu] = \left(\frac{K_m 10^3}{\tau_m}\right)\frac{[20(n_o{*}-n{*})]}{[10n{*}]}$$



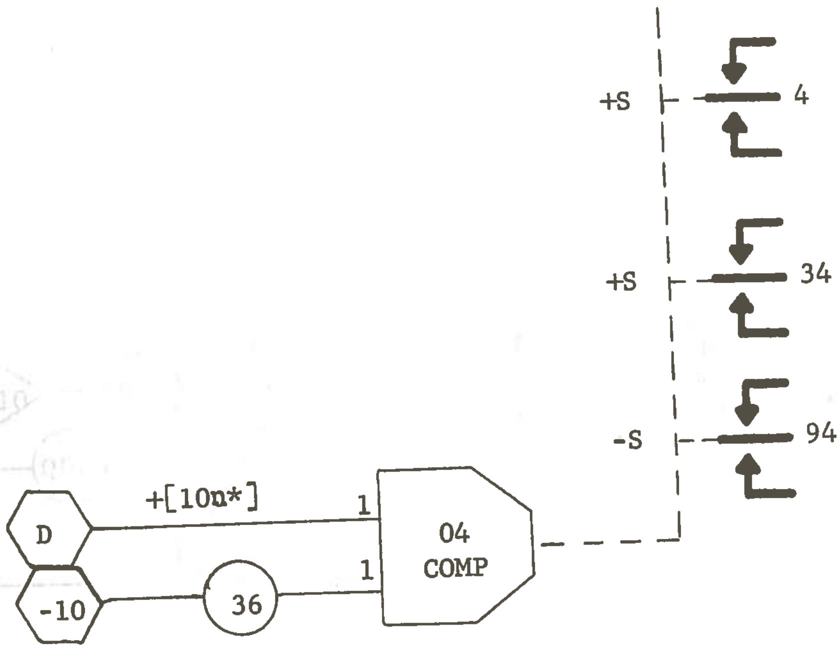
Pot	Setting
100	0.1000
107	0.2000

Reactivity:

$$[2000\delta k] = 10(20k)\ [10] - 10(0.8)\ [\frac{T_f}{200}] + 2000\mu$$



Automatic Scram:



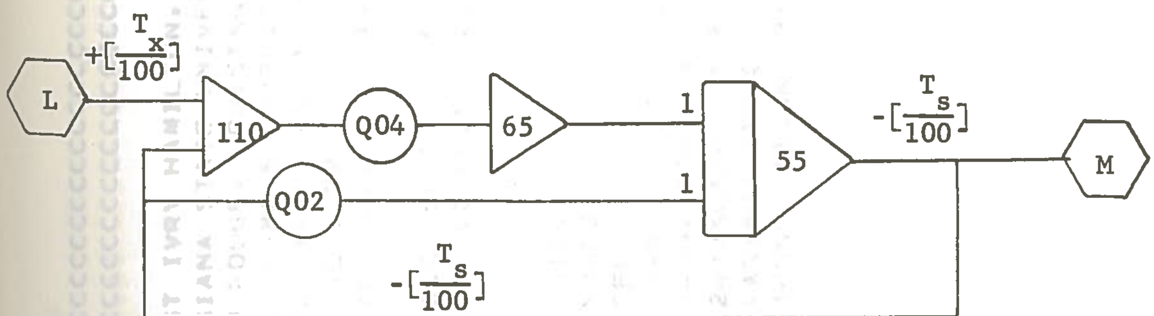
Pot	Setting
36	0.9500

Boiler - Power Demand:

$$\frac{d}{dt}\left[\frac{T_s}{100}\right] = K_1\left(\left[\frac{T_x}{100}\right] - \left[\frac{T_s}{100}\right]\right) - K_2A$$

Where A = Throttle Opening

A is the fraction of full open throttle.



APPENDIX B

```
 JOB
 FORTRANH  LS,GO,RT,S
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CC                                                                   CC
CC      PROGRAMMER      ERNEST IVRY HAMILTON, JR.                    CC
CC                      LOUISIANA STATE UNIVERSITY                   CC
CC                      BATON ROUGE, LOUISIANA                       CC
CC      MACHINE         XEROX SIGMA 5  HYBRID EAI 680                CC
CC      LANAGUAGE       FORTRAN IV-H, ASSEMBLE                       CC
CC      DATE            24 APRIL 1972                                CC
CC      PURPOSE         DIGITAL PROCESS CONTROL COMPUTER SOFTWARE    CC
CC                                                                   CC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      DOUBLE PRECISION NADC
      DOUBLE PRECISION N2ADC
      COMMON /ADC/ ADC(24),OADC(24)
      COMMON /NADC/ NADC(24),N2ADC(24)
      COMMON /TIME/ IHR,MIN,ISEC
      COMMON /CONTAB/ CT(70)/IDAMSET/ IDS(12)
      COMMON /RUNI/ IRUNI(4)/TRUN/ ITRUN(4)
      COMMON /ALARML/ ALARML(24)/ALARMU/ ALARMU(24)
      COMMON /IDEM/ IDEM(13)/VARTAB/ VT(70)
      COMMON /FLAG/ IFLAG(8)/TRUNC/ ITRUNC(8)
      CONNECT (2Z60,CLOCK)
      CONNECT (2Z61,SUB1)
      CONNECT (2Z62,SUB2)
      CONNECT (2Z63,SUB3)
      CONNECT (2Z64,SUB4)
      CONNECT (2Z65,SUB5)
```

```
C
C     READ IN THE RUN INTERVALS
C
      READ(5,10)(IRUNI(I),I=1,4)
10    FORMAT(4I5)
C
C     READ IN LOWER ALARM LIMITS
C
      READ(5,40)(ALARML(I),I=1,8)
      READ(5,40)(ALARML(I),I=9,16)
      READ(5,40)(ALARML(I),I=17,24)
40    FORMAT(8F10.5)
C
C     READ IN UPPER ALARM LIMITS
C
      READ(5,40)(ALARMU(I),I=1,8)
      READ(5,40)(ALARMU(I),I=9,16)
      READ(5,40)(ALARMU(I),I=17,24)
      IHR=0
      MIN=0
      ISEC=0
C
C     READ IN NAMES OF VARIABLES
C     1ST LINE OF NAME IS 8 COL. WIDE
C     AND STARTS IN COL. THAT IS 1
C     PLUS MULTIPLE OF 10
C
      READ(5,50)(NADC(I),I=1,8)
      READ(5,50)(NADC(I),I=9,16)
      READ(5,50)(NADC(I),I=17,24)
50    FORMAT(8(A8,2X))
```

```
C
C    2ND LINE OF NAME IS 8 COL. WIDE
C    AND STARTS IN COL. THAT IS 1
C    PLUS MULTIPLE OF 10
C
      READ(5,50)(N2ADC(I),I=1,8)
      READ(5,50)(N2ADC(I),I=9,16)
      READ(5,50)(N2ADC(I),I=17,24)
C
C    READ IN THE MAXIMUM VALUES
C    OF THE ANALOG VARIABLES
C    WITH ADC NUMGER PLUS 1 EQUAL
C    TO ADC NUMBER IN DIGITAL
C
      READ(5,60)(CT(I),I=1,8)
      READ(5,60)(CT(I),I=9,16)
      READ(5,60)(CT(I),I=17,24)
C
C    READ IN THE MAXIMUM VALUES OF DAMS--
C    (DIGITAL ANALOG MULTIPLIERS)
C    AND DAMSET NUMBER PLUS 24 EQUAL
C    TO 1 GREATER THAN DAC NUMBER IN
C    THE DIGITAL.
C
      READ(5,60)(CT(I),I=25,32)
      READ(5,60)(CT(I),I=33,39)
60    FORMAT(8E10.0)
C
C    READ IN CONTROL PROGRAM RUNI
C
      READ(5,150)(CT(I),I=41,48)
150   FORMAT(8F10.2)
```

```
      DO 160 I=1,8
      J=I+40
160   ITRUNC(I)=CT(J)
C
C   TO ZERO CONTROL RUN INTERVAL FLAGS
C
      DO 200 I=1,8
200   IFLAG(I)=3
      DO 100 I=1,12
100   IDS(I)=0
      IDEM(13)=0
      DO 20 I=1,4
20    ITRUN(I)=IRUNI(I)
C
C   TO SET INDICATORS TO NOT ALLOW MESSAGES TO BE
C   PRINTED FROM CONTROL PROGRAMS
C
      DO 300 I=51,58
300   CT(I)=0.0
      CALL LTDA(1,0.400)
      VT(25)=400.0
      CT(26)=0.700
      CALL SSRM(64)
      CALL CONEC
      STOP
      END


      SUBROUTINE CLOCK
      COMMON /CONTAB/ CT(70) /IDAMSET/ IDS(12)
      COMMON /TIME/ IHR,MIN,ISEC
```

```
      COMMON /RUNI/ IRUNI(4)/TRUN/ ITRUN(4)
      COMMON /IDEM/ IDEM(13)/VARTAB/ VT(70)
      COMMON /FLAG/ IFLAG(8)/TRUNC/ ITRUNC(8)
C
C   THE TIME FOR THE REAL TIME CLOCK
C
      ISEC=ISEC+1
      IF(ISEC.GE.60)GOTO1
3     IF(MIN.EQ.60)GOTO2
      GOTO4
1     ISEC=0
      MIN=MIN+1
      GOTO3
2     MIN=0
      IHR=IHR+1
      IF(IHR.LE.24)GOTO4
      IHR=0
4     CONTINUE
C
C   TO DOWN COUNT TRUN
C
      DO 20 I=1,4
20    ITRUN(I)=ITRUN(I)-1
C
C   TO TRIGGER THE DIFFERENT INTERRUPTS
C
      IF(ITRUN(1).GT.0)GOTO 30
S        LI,1       X'0061'
S        CAL1,5     1
      ITRUN(1)=IRUNI(1)
30    IF(ITRUN(2).GT.0)GOTO 40
S        LI,1       X'0062'
```

```
S          CAL1.5     1
         ITRUN(2)=IRUNI(2)
40       IF(ITRUN(3).GT.0)GOTO 50
S           LI.1        X'0063'
S           CAL1.5     1
         ITRUN(3)=IRUNI(3)
50       IF(IDEM(13).EQ.0)GOTO60
         IF(ITRUN(4).GT.0)GOTO 60
S           LI.1        X'0064'
S           CAL1.5     1
         ITRUN(4)=IRUNI(4)
60       CONTINUE
1000     CONTINUE
         DO 70 I=1,8
         IF(IFLAG(I).GE.1)GOTO69
         ITRUNC(I)=ITRUNC(I)-1
         IF(ITRUNC(I).GT.0)GOTO70
65       IFLAG(I)=2
         J=I+40
         ITRUNC(I)=CT(J)
69       CONTINUE
S           LI.1        X'0063'
S           CAL1.5     1
         ITRUN(3)=IRUNI(3)
70       CONTINUE
         END


         SUBROUTINE SUB1
         COMMON /CONTAB/ CT(70)/IDAMSET/ IDS(12)
         COMMON /ADC/ ADC(24),OADC(24)/VARTAB/ VT(70)
```

```
      DO 5 I=1,24
C
C     TO SAVE THE LAST VALUE IN ENGINEERING UNITS
C     BY STORING INTO OADC(I)
C
5       OADC(I)=ADC(I)
C
C     TO SCAN PROCESS
C
        I=0
        N=0
10      N=N+1
        CALL CRAC (I,ADC(N))
        I=I+1
        IF(I.LT.24)GOTO 10
C
C     TO CONVERT TO ENGINEERING UNITS
C
        DO 30 I=1,24
30      ADC(I)=ADC(I)*CT(I)
C
C     TO FILTER
C     AND STORE IN VARTAB--VT(I)
C     AT PRESENT THERE IS NO FILTERING
C
        DO 40 I=1,24
40      VT(I)=ADC(I)
        VT(26)=CT(26)
C
C     USED TO SET DAMS--(DIGITAL TO ANALOG
C     MULTIPLIERS)
C
```

```
      DO 50 I=1,12
      J=I+24
      IF(IDS(I).NE.0)GOTO60
50    CONTINUE
      GOTO70
60    K=I+59
      VT(J)=CT(K)
      SEND=VT(J)/CT(J)
      ICHAN=IDS(I)
      CALL LTDA(ICHAN,SEND)
      IDS(I)=0
      GOTO50
70    CONTINUE
      END


      SUBROUTINE SUB2
      COMMON /FLAG/ IFLAG(8)
      IF(IFLAG(1).EQ.1)CALL CONTR1
      IF(IFLAG(2).EQ.1)CALL CONTR2
      IF(IFLAG(3).EQ.1)CALL CONTR3
      IF(IFLAG(4).EQ.1)CALL CONTR4
      IF(IFLAG(5).EQ.1)CALL CONTR5
      IF(IFLAG(6).EQ.1)CALL CONTR6
      IF(IFLAG(7).EQ.1)CALL CONTR7
      IF(IFLAG(8).EQ.1)CALL CONTR8
      END
```

```
      SUBROUTINE SUB3
      COMMON /FLAG/ IFLAG(8)
      IF( IFLAG(1).EQ.2)CALL CONTR1
      IF( IFLAG(2).EQ.2)CALL CONTR2
      IF( IFLAG(3).EQ.2)CALL CONTR3
      IF( IFLAG(4).EQ.2)CALL CONTR4
      IF( IFLAG(5).EQ.2)CALL CONTR5
      IF( IFLAG(6).EQ.2)CALL CONTR6
      IF( IFLAG(7).EQ.2)CALL CONTR7
      IF( IFLAG(8).EQ.2)CALL CONTR8
      END


      SUBROUTINE CONTR1
      COMMON /TIME/ IHR.MIN.ISEC
      COMMON /COLDST/ ICOLST(8)
      COMMON /CONTAB/ CT(70) /IDAMSET/ IDS(12)
      COMMON /FLAG/ IFLAG(8)/VARTAB/ VT(70)
      IF( ICOLST(1).EQ.0)GOTO40
C   VT(12) IS ACTUALLY A NEGATIVE NUMBER.
      ERRNOW=CT(37)+VT(12)
      ACTKP=CT(38)*(ERRNOW-ERROLD)
      ACTKI=CT(39)*ERRNOW
      DELM=ACTKP+ACTKI
      ERROLD=ERRNOW
      CT(60)=VT(25)+DELM
      IDS(1)=1
      GOTO50
 40   ERROLD=0.0
      CT(37)=-VT(12)
      ICOLST(1)=1
```

```
50    CONTINUE
      IF(CT(51).NE.1.0)GOTO51
      WRITE(7,9999)IHR,MIN,ISEC,ERRNOW,DELM,CT(38),
     *CT(39),ACTKP,ACTKI,CT(37),CT(60)
9999  FORMAT('TIME ',3(I2,1X),' CONTROL PROGRAM 1 RAN ',
     *' ERROR NOW = ',F10.2,'  DEG. F',/,
     *'DELM = ',F10.2,'  KP = ',F10.2,'  KI = ',F10.2,/,
     *'PROP ACTION = ',F10.2,' INT ACTION = ',F10.2,
     *'   TARGET = ',F10.2,/,'MANIPULATED VAR = ',F10.2)
51    CONTINUE
      IFLAG(1)=0
      END


      SUBROUTINE CONTR2
      COMMON /FLAG/ IFLAG(8)/TRUNC/ ITRUNC(8)
      WRITE(7,9999)
9999  FORMAT('*******CONTROL PROGRAM 2 RAN')
      IFLAG(2)=0
      END


      SUBROUTINE CONTR3
      COMMON /FLAG/ IFLAG(8)/TRUNC/ ITRUNC(8)
      WRITE(7,9999)
9999  FORMAT('*******CONTROL PROGRAM 3 RAN')
      IFLAG(3)=0
      END
```

```
      SUBROUTINE CONTR4
      COMMON /FLAG/ IFLAG(8)/TRUNC/ ITRUNC(8)
      WRITE(7,9999)
9999  FORMAT('*******CONTROL PROGRAM 4 RAN')
      IFLAG(4)=0
      END


      SUBROUTINE CONTR5
      COMMON /FLAG/ IFLAG(8)/TRUNC/ ITRUNC(8)
      WRITE(7,9999)
9999  FORMAT('*******CONTROL PROGRAM 5 RAN')
      IFLAG(5)=0
      END


      SUBROUTINE CONTR6
      COMMON /FLAG/ IFLAG(8)/TRUNC/ ITRUNC(8)
      WRITE(7,9999)
9999  FORMAT('*******CONTROL PROGRAM 6 RAN')
      IFLAG(6)=0
      END


      SUBROUTINE CONTR7
      COMMON /FLAG/ IFLAG(8)/TRUNC/ ITRUNC(8)
      WRITE(7,9999)
9999  FORMAT('*******CONTROL PROGRAM 7 RAN')
      IFLAG(7)=0
```

```
      END

      SUBROUTINE CONTR8
      COMMON /FLAG/ IFLAG(8)/TRUNC/ ITRUNC(8)
      WRITE(7,9999)
9999  FORMAT('*******CONTROL PROGRAM 8 RAN')
      IFLAG(8)=0
      END


      SUBROUTINE SUB4
      COMMON /TIME/ IHR,MIN,ISEC/VARTAB/ VT(70)
      COMMON /CONTAB/ CT(70)/IDAMSET/ IDS(12)
      COMMON /ADC/ ADC(24),OADC(24)
      COMMON /IDEM/ IDEM(13)
C
C   TO PRINT THE VALUES THAT ARE IN VARTAB
C   FOR THE TREND LOG
C
      IDPT=IDEM(13)
      WRITE(6,10)IHR,MIN,ISEC,(VT(IDEM(I)),I=1,IDPT)
10    FORMAT(1X,I2,2X,I2,2X,I2,12(1X,F10.3,1X))
      END


      SUBROUTINE SUB5
      DOUBLE PRECISION NADC
      DOUBLE PRECISION N2ADC
```

```
      DOUBLE PRECISION TEMP1
      DOUBLE PRECISION TEMP2
      COMMON /COLDST/ ICOLST(8)
      COMMON /TIME/ IHR,MIN,ISEC
      COMMON /FLAG/ IFLAG(8)/TRUNC/ ITRUNC(8)
      COMMON /CONTAB/ CT(70)/IDAMSET/ IDS(12)
      COMMON /RUNI/ IRUNI(4)/TRUN/ ITRUN(4)
      COMMON /ADC/ ADC(24),OADC(24)
      COMMON /ALARML/ ALARML(24)/ALARMU/ ALARMU(24)
      COMMON /NADC/ NADC(24),N2ADC(24)
      COMMON /IDEM/ IDEM(13)/VARTAB/ VT(70)
C
C   TO COMMUNICATE WITH THE HUMAN OPERATOR
C
      WRITE(7,10)
10    FORMAT('IFUNCODE IDPT VALUE')
      CONTINUE
      WRITE(7,11)
11    FORMAT('II II FFFFFFFFF  NOTED EXAMPLE TYPE-IN0'/
     *'02 04 10.0000000 FOLLOWED BY NL THEN EOM')
      READ(7,12)IFNCOD,IDPT,VALUE
12    FORMAT(I2,1X,I2,F10.5)
      GOTO(21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,
     *36,37,38),IFNCOD
      GOTO41
C
C   UPDATE OF CONSTANTS IN CONTROL TABLE(IDPT) TO VALUE.
C
21    OVALUE=CT(IDPT)
      WRITE(7,100)IDPT,VALUE
100   FORMAT('**** UPDATE OF CONSTANTS ***************'/
     *' IN CONTROL TABLE(',I2,') TO ',F10.5)
```

```
      IF(IDPT.GT.50)GOTO1110
1100  CALL QUEST(OVALUE,VALUE,IXX)
      IF(IXX.EQ.2)GOTO40
      CT(IDPT)=VALUE
      GOTO40
1110  IF(IDPT.GT.58)GOTO1100
      J=IDPT-51
      IDS(J)=1
      GOTO40
C
C   UPDATE RUN INTERVAL(IDPT) TO VALUE.
C
22    OVALUE=IRUNI(IDPT)
      WRITE(7,101)IDPT,VALUE
101   FORMAT('**** UPDATE OF CONSTANTS ***************'/
     *' IN TASK RUN INTERVAL(',I2,') TO ',F10.5)
      CALL QUEST(OVALUE,VALUE,IXX)
      IF(IXX.EQ.2)GOTO40
      IRUNI(IDPT)=VALUE
      GOTO40
C
C   UPDATE ALARM LOWER LIMIT(IDPT) TO VALUE.
C
23    OVALUE=ALARML(IDPT)
      WRITE(7,102)IDPT,VALUE
102   FORMAT('**** UPDATE OF CONSTANTS ***************'/
     *' IN LOWER ALARM LIMIT(',I2,') TO ',F10.5)
      CALL QUEST(OVALUE,VALUE,IXX)
      IF(IXX.EQ.2)GOTO40
      ALARML(IDPT)=VALUE
      GOTO40
C
```

```
C     UPDATE ALARM UPPER LIMIT(IDPT)TO VALUE.
C
24      OVALUE=ALARMU(IDPT)
        WRITE(7,103)IDPT,VALUE
103     FORMAT('**** UPDATE OF CONSTANTS ***************'/
       *' IN UPPER ALARM LIMIT(',I2,') TO ',F10.5)
        CALL QUEST(OVALUE,VALUE,IXX)
        IF(IXX.EQ.2)GOTO40
        ALARMU(IDPT)=VALUE
        GOTO40
C
C     LIST ON LOGGING DEVICE ALARM LIMITS.
C
25      DO 200 N=1,8
        M=N+8
        L=N+16
        WRITE(6,200)N,ALARML(N),N,ALARMU(N),
       *M,ALARML(M),M,ALARMU(M),
       *L,ALARML(L),L,ALARMU(L)
200     FORMAT(1X,'ALARML(',I1,')',F10.2,
       *2X,'ALARMU(',I1,')',F10.2,
       *2X,'ALARML(',I2,')',F10.2,
       *2X,'ALARMU(',I2,')',F10.2,
       *2X,'ALARML(',I2,')',F10.2,
       *2X,'ALARMU(',I2,')',F10.2)
        GOTO40
C
C     CHANGE NAME OF ANALOG INPUT(IDPT).
C
26      WRITE(7,1000)IDPT
1000    FORMAT('**** CHANGE NAME ************************'/
       *' OF ANALOG VARIABLE(',I2,')')
```

```
      WRITE(7,201)
201   FORMAT('TYPE IN 16 CHAR. NAME')
      READ(7,202)TEMP1,TEMP2
202   FORMAT(A8,A8)
      WRITE(7,1001)NADC(IDPT),N2ADC(IDPT),TEMP1,TEMP2
1001  FORMAT(' DO YOU WANT TO CHANGE NAME '/
     *'FROM ',A8,A8,' TO ',A8,A8/
     *' TYPE IN 1 IF CORRECT OR 2 IF NOT CORRECT')
      READ(7,1002)IEC
1002  FORMAT(I1)
      IF(IEC.EQ.2)GOTO1003
      NADC(IDPT)=TEMP1
      N2ADC(IDPT)=TEMP2
      WRITE(7,1004)NADC(IDPT),N2ADC(IDPT)
1004  FORMAT(' THE NAME IS CHANGED TO ',A8,A8)
      GOTO40
1003  WRITE(7,1005)NADC(IDPT),N2ADC(IDPT)
1005  FORMAT(' THE NAME REMAINS ',A8,A8)
      GOTO40
C
C   LIST ON LOGGING DEVICE NAMES OF ANALOG INPUTS.
C
27    DO 205 J=1,8
      M=J+8
      N=J+16
205   WRITE(6,206)J,NADC(J),N2ADC(J),M,NADC(M),
     *N2ADC(M),N,NADC(N),N2ADC(N)
206   FORMAT(1X,'NADC(',I1,')',A8,A8,4X,
     *'NADC(',I2,')',A8,A8,4X,'NADC(',I2,')',A8,A8)
      GOTO40
C
C   START TRENDING LOG, IDPT = NUMBER OF VARIABLES
```

```
C     TRENDED
C
28        CONTINUE
          IF(IDEM(13).GE.10)GOTO230
          DO 209 J=1,IDPT
232       WRITE(7,208)
208       FORMAT('TYPE IN ID OF ANALOG INPUT. FORMAT I2')
209       READ(7,210)IDEM(J)
210       FORMAT(I2)
219       WRITE(6,220)(NADC(IDEM(I)),IDEM(I),I=1,IDPT)
220       FORMAT(1X,'HR MIN SEC',10(1X,A8,1X,I2))
          WRITE(6,221)(N2ADC(IDEM(I)),IDEM(I),I=1,IDPT)
221       FORMAT(11X,10(1X,A8,1X,I2))
          IDEM(13)=IDPT
          GOTO40
C
C     STOP TREND LOGGING
C
29        IDEM(13)=0000
          GOTO40
C
C     ADD ONE VARIABLE TO THE TREND LOG
C     IDEM(13) EQUALS NUMBER OF VARIABLES ON TREND
C     LOG THE MAXIMUN IS 10
C
30        IF(IDEM(13).EQ.10)GOTO230
          IDEM(13)=IDEM(13)+1
          J=IDEM(13)
          IDPT=J
          GOTO 232
230       WRITE(7,231)
231       FORMAT('YOU CANNOT ADD TO THE TREND ',
```

```
      *'LOG MORE THAN 10 VARIABLES')
       GOTO 40
C
C  TO DELETE ONE VARIABLE FROM THE TREND LOG
C
31     MMM=IDEM(13)
       DO 240 JJ=1,MMM
       IF(IDPT.EQ.IDEM(JJ))GOTO 250
240    CONTINUE
       WRITE(7,241) IDPT
241    FORMAT('YOU ARE NOT TRENDING IDEM(',I2,')')
       GOTO40
250    M=IDEM(13)-JJ
       IDEM(13)=IDEM(13)-1
       DO 251 KK=1,M
       MM=JJ+1
       IDEM(JJ)=IDEM(MM)
251    JJ=JJ+1
       J=IDEM(13)
       IDPT=J
       GOTO219
C
C   TO GIVE THE CLOCK THE CORRECT TIME OF DAY
C
32     WRITE(7,60)
60     FORMAT('MY CLOCK SEEMS TO HAVE ',
      *'LOST THE TIME OF DAY.  COULD YOU',/,
      *'TELL ME THE TIME   IF IT WERE ',
      *'TO BE 15 SECONDS AFTER 8030',/,
      *'YOU WOULD TYPE THAT IN AS 083015  ',/,
      *'  NOTEO  I LIKE MILITARY TIME ALSO.')
       READ(7,61)IHR,MIN,ISEC
```

```
61    FORMAT(3I2)
35    WRITE(7,62)IHR,MIN,ISEC
62    FORMAT('THANK YOU FOR THE TIME.',/,
     *'  THE TIME IS NOW',1X,3I2)
      GOTO40
C
C   TO TRIGGER CONTROL PROGRAMS
C
33    IF(IFLAG(IDPT).EQ.0)GOTO350
      WRITE(7,300)IDPT
300   FORMAT(' CONTROL PROGRAM ',I2,
     *'  ALREADY WAITING TO RUN')
350   IFLAG(IDPT)=1
      WRITE(7,360)IDPT
360   FORMAT(' CONTROL PROGRAM ',I2,' TRIGGERED')
      GOTO40
C
C   TO TURN OFF A CONTROL PROGRAM
C
34    IF(IFLAG(IDPT).EQ.3)GOTO370
      IFLAG(IDPT)=3
      J=50+IDPT
      CT(J)=0.0
      ICOLST(IDPT)=0
      WRITE(7,374)IDPT
374   FORMAT(' CONTROL PROGRAM ',I2,' MASKED')
      GOTO40
370   WRITE(7,375)IDPT
375   FORMAT(' CONTROL PROGRAM',I2,' ALREADY MASKED')
      GOTO40
C
C   TO LIST STATUS OF CONTROL PROGRAMS
```

```
C
35    DO 390 I=1,8
      WRITE(7,380)I,IFLAG(I)
380   FORMAT('   CONTROL PROGRAM ',I2,'  FLAG SET ',I2)
390   CONTINUE
      WRITE(7,400)
400   FORMAT(//' 3 MEANS MASKED ',/,' 2 MEANS TIMED OUT',
     */,' 1 MEANS TRIGGERED',/,' 0 MEANS NOT MASKED')
      GOTO40
C
C   TO ALLOW CONTROL PROGRAM(IDPT) TO PRINT MESSAGES
C   TO OPERATOR
C
36    IF(IDPT.LT.1)GOTO410
      IF(IDPT.GT.8)GOTO410
      J=IDPT+50
      CT(J)=1.0
      WRITE(7,430)IDPT
430   FORMAT('CONTROL PROGRAM ',I2,'WILL BE ALLOWED',
     *' TO PRINT MESSAGES')
      GOTO40
410   WRITE(7,420)IDPT
420   FORMAT('THERE IS NOT A CONTROL PROGRAM ',I2)
      GOTO40
C
C   TO STOP A CONTROL PROGRAM(IDPT) FROM
C   PRINTING MESSAGES TO OPERATOR
C
37    IF(IDPT.LT.1)GOTO510
      IF(IDPT.GT.8)GOTO510
      J=IDPT+50
      CT(J)=0.0
```

```
      WRITE(7,530)IDPT
530   FORMAT('CONTROL PROGRAM ',I2,
     *' WILL STOP PRINTING MESSAGES')
      GOTO40
510   WRITE(7,520)IDPT
520   FORMAT('THERE IS NOT A CONTROL ',
     *'PROGRAM ',I2)
      GOTO40
C
C   TO CHANGE LOAD ON HEAT EXCHANGER
C
38    WRITE(7,550)
550   FORMAT('CHANGE IN LOAD ON HEAT EXCHANGER')
      OVALUE=CT(26)
      CALL QUEST(OVALUE,VALUE,IXX)
      IF(IXX.EQ.2)GOTO40
      CT(26)=VALUE
      CALL LTDA(2,CT(26))
      CALL SSRM(9)
      GOTO40
C
C   TO PRINT THAT ILLEGAL FUNCTION CODE
C   WAS ENTERED
C
41    WRITE(7,42)IFNCOD
42    FORMAT(1X,'YOU HAVE ENTERED AN ILLEGAL ',
     *'FUNCODE ',I2)
40    CONTINUE
      END
```

```
      SUBROUTINE QUEST(VALO,VALN,IXX)
      IXX=0
      PERCEN=ABS(((VALN-VALO)/VALO)*100.0)
      IF(PERCEN.LT.5.0)GOTO20
1     WRITE(7,10)VALO,VALN
10    FORMAT('****  DO YOU WANT TO CHANGE ',
     *'THE VALUE FROM ',F10.2,'  TO  ',F10.2,/,
     *' TYPE IN 1 IF CORRECT OR 2 IF NOT CORRECT')
      READ(7,11)IXX
11    FORMAT(I1)
      IF(IXX.EQ.1)GOTO20
      IF(IXX.EQ.2)GOTO100
12    FORMAT('YOU TYPED IN A  ',I1)
      GOTO1
100   WRITE(7,13)
13    FORMAT('****  THERE WILL BE NO CHANGE ',
     *'IN THE VALUE')
      GOTO25
20    WRITE(7,14)VALO,VALN
14    FORMAT('**********  THE VALUE WILL BE ',
     *'CHANGED FROM ',F10.2,'  TO  ',F10.2)
25    CONTINUE
      RETURN
      END


  SYMBOL GO,SI,LO


        DEF      CONEC
```

```
         REF       9ITOR
         REF       9RTOI
TYPE     STW,1     SAVE
         LI,1      500
         STW,1     X'18C'
         LI,1      X'8000'
         WD,1      X'1702'
         LW,1      SAVE
         LPSD,3    TCB
         BOUND     8
TCB      DATA      0,0,TYPE,0
SAVE     RES       1
XP       XPSD,0    TCB
MTW      MTW,-1    X'18C'
CONEC    LW,3      XP
X        XPSD,0    PSD
         LI,4      11
         AWM,4     PSD
         STW,3     X'5A'
         LI,4       500
         STW,4     X'18D'
         STW,4     X'18C'
         LW,3      MTW
         STW,3     X'54'
         LI,1      X'2080'
         WD,1      X'1200'
         LPSD,0    PSD
         B         *15
         BOUND     8
PSD      DATA      0,0,X+1,0
         END
```

```
OLOAD (GO),(TEMP,1900),(UDCB,4),(FORE,1600),(TASK,6),(LIB,SYSTEM,USER)
OASSIGN (F01,BO)
OASSIGN (F05,SI)
OASSIGN (F06,LO)
OASSIGN (F07,OC)
 ROV

    1    1   60    15
       0.0       0.0       0.0       0.0       0.0       0.0       0.0       0.0
       0.0       0.0       0.0       0.0       0.0       0.0       0.0       0.0
       0.0       0.0       0.0       0.0       0.0       0.0       0.0       0.0
       0.0       0.0       0.0       0.0       0.0       0.0       0.0       0.0
       0.0       0.0       0.0       0.0       0.0       0.0       0.0       0.0
       0.0       0.0       0.0       0.0       0.0       0.0       0.0       0.0
REDUCED   AVG FUEL  TEMP OUT  TEMP OUT  T. INLET  AVG TEMP  COOL IN   TEMP OUT
T  INLET  T  INLET  CNTL  RD  STEAM     TEMP AVG    ERROR   55555555  66666666
77777777  TEMP.     LOAD ON   00000000  11111111  22222222  33333333  44444444
N  DEN.   TEMP.  F  CORE   F  MIXING F  EXCH  -F  EXCH   F  CORE   F  EXCH  -F
MIXING F  CORE   F  POSITION  TEMP. -F  COMPAR F  TEMP.  F  55555555  66666666
77777777  SET PT.   EXCH.     00000000  11111111  22222222  33333333  44444444
1.0       2000.0    1000.0    1000.0    1000.0    1000.0    1000.0    1000.0
1000.0    1000.0    1.000     10000.0   1000.0    1000.0    1.0       1.0
0.005     1000.0    1.0       0.0       0.0       0.0       0.0       1.00
1000.0    1.0       0.0       0.0       0.0       0.0       0.0       0.0
0.0       0.0       0.0       0.0       0.0       0.1       0.01
10.0      60.0      60.0      60.0      300.0     300.0     300.0     300.0
```

VITA

Ernest Ivry Hamilton, Jr. was born in Lake Charles, Louisiana, 17 July, 1947.

Secondary education was obtained in Hackberry at Hackberry High School, from which he graduated in 1965. In June, 1965, he entered Louisiana State University at Baton Rouge, where he received a Bachelor of Science in Chemical Engineering in May, 1970.

In September, 1970, he enrolled in the Graduate School of Louisiana State University. At present he is a candidate for a degree of Master of Science in the Department of Nuclear Engineering.