DESIGN AND IMPLEMENTATION OF A DIGITAL CONTROLLER DEVELOPMENT SYSTEM

A thesis submitted to the
Faculty of Graduate Studies
in partial fulfillment of the requirements
for the degree of Master of Science
in the Department of
Electrical Engineering

by
Rodney Dale Gilchrist
Saskatoon, Saskatchewan
November 1977

ACKNOWLEDGEMENTS

The author would like to thank Professors K.E. Bollinger and D.E. Dodds for their assistance, guidance, and encouragement during the course of this project. Ms. V. Mahoney's programming assistance, and the efforts of Mr. R. Winsor of Calgary Power, which were largely responsible for the existence of the field test chapter, were also very much appreciated.

Financial support for this project was received from the National Research Council, grant number A-5124.

UNIVERSITY OF SASKATCHEWAN

Electrical Engineering Abstract 77A179

"DESIGN AND IMPLEMENTATION OF A DIGITAL CONTROLLER DEVELOPMENT SYSTEM"

student: Rod Gilchrist supervisor: K.E. Bollinger

M. Sc. Thesis presented to the College of Graduate Studies

November 1977

ABSTRACT

This thesis describes the design and implementation of a minicomputer based system to develop and evaluate general purpose digital feedback controllers that will operate in the frequency range below ten hertz. An immediate area of application is in the control of turbo generators where additional signals, beyond the standard terminal voltage feedback, are used to provide damping. The feedback of such signals as the machine output electrical power has recently proved advantageous. The controller development system will be a powerful tool in further research in this area. It will allow the development of digital control techniques that could not previously be conveniently tested.

Field test results are described where, as a convenient test case, the digital system satisfactorily emulated an existing power system stabilizer on Calgary Power's Sundance unit 3. Stripchart traces are included that illustrate the performance of the digital controller relative to the existing installed analog controller. These tests were performed with the generator operating on-line near rated load conditions. The tests point out some of the attributes and shortcomings of the prototype digital controller.

This research was supported by the National Research Council under grant A-5124.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

INTRODUCTION

1.1 Overview

Advances in the science of digital electronics are bringing increasingly complex computer applications into the realm of economic feasibility. Digital processors are simultaneously and rapidly becoming more powerful and less expensive. This thesis project is an application of the current level of minicomputer technology to create a research tool in the area of power systems feedback control. The device is to be used as a programmable controller that will enable a wide range of control strategies to be tested at power plants at various times in the future. The application presently requires all of the power of a minicomputer to give performance in the desired range. However, the rapid advance of the state of the art in mini and microcomputers promises to make later versions of the device developed here not only more powerful, but also less expensive.

The project involves the construction of a system, based around a commercial minicomputer, capable of emulating analog feedback controllers and implementing digital controller techniques in systems with a maximum frequency of interest below about ten hertz. Although the digital controller is to be used as a general purpose research device, the design incorporates features to make it practical. An assumption was made in the design specification that the device might be installed in a power system on a temporary basis, and so considerations of reliability and system protection were included in the design.

There is a multitude of advantages to a programmable digital approach for implementing feedback controllers. They include convenience, flexibility, sophisticated control techniques, and self testing. Further, there are several advantages to digital approaches in general. An improvement in noise performance, for example, is often available because once the input signal is digitized it is immune to further analog noise pick up. This could be used to advantage by a digital controller in a noisy environment if the digitization of the signal was done close to the point where it was generated. A second advantage typically gained from a digital, rather than an analog, implementation of a controller is the ability to trade excess processor time directly for parallel controller functions, without requiring additional hardware.

These are some of the advantages of a digital device. The disadvantages are fewer in number. They include the price of the digital hardware and software, the additional complexity of the system, relative to an analog controller, and the relatively low limit on the maximum frequency. In the specific application of this device to power system control these disadvantages are not too significant. The maximum operating bandwidth of the direct digital controller (DDC) system is adequate to accommodate feedback control of a power system synchronous generator, the price of the hardware and software is comparable to the cost of the standard analog approach, and the problems of complexity can be masked by judicious software design.

The additional complexity of a DDC controller over that of its analog counterpart gives rise to difficulties in two areas. Firstly, the component count of the digital system is higher than the analog system with the resultant possibility of lower reliability. Secondly, the additional hardware complexity can give rise to additional operating difficulty. Digital devices are already being used in applications requiring high reliability, such as satelites, manned rocket flights, and military hardware. This is an indication that digital devices can have the necessary reliability. Consequently, reduced reliability is not grounds to disallow digital controllers in general. The DDC controller can be more complicated to tune. Instead of adjusting several calibrated potentiometers and operating a few switches, the controller parameter input device becomes a computer terminal. In order to tune the controller the operator must be able to communicate with, and consequently to some extent understand the functioning of, the controller program which is necessarily more complicated than the analog hardware it replaces. However, with judicious programming, this complication can be made largely transparent. The additional flexibility of the digital controller system more than offsets the additional expertise required of the operator.

This thesis describes a digital controller system that has been constructed to allow convenient implementation of control stategies during tests at power plants. A convenient benchmarking test was conducted in a power system environment. The test was the emulation of a device called a power systems stabilizer (PSS) on a 380 MVA turbo generator at an actual field installation. The next section describes in general the various feedback control loops in such a turbo generator.

1.2 Turbo Generator Control Loops

A turbine-generator system is intended to be a constant voltage and frequency source, independant of power demands made on it.

There are three distinct recovery mechanisms in the system following a load disturbance. The fastest of these is the action of the automatic voltage regulator. The AVR uses a measurement of the terminal voltage to influence the amount of current flowing in the rotor of the machine. A sudden increase in load causes the terminal voltage to drop, which in turn causes the AVR to increase the field current in order to restore the terminal voltage. The effect of the AVR is felt within several tenths of a second after the transient.

Increasing the load also brings into play the second recovery mechanism. The effect of the higher load is a decrease in rotor speed. The electrical power generated and the rotor speed are in fact related by a differentiation in the mathematical model. The decrease in shaft speed is sensed and transmitted to the turbine governor which opens the control valve to increase the amount of steam flowing through the turbine. This brings the mechanical power supply and demand back into line on a short term basis. This control loop has its effect within about five seconds of the transient.

The third loop is concerned with producing the extra steam demanded by the turbine. In the short term there is some steam reserve in the boiler system. In order to balance steam production and demand, however, the cumbustion rate must be increased. The control loop that performs this function is the slowest of the three. Its effect is felt in tens of seconds.

All of these control loops are slow enough to allow digital control elements in them. The feedback control loop of particular interest in this thesis is that of the automatic voltage regulator. The power system stabilizer is added to this loop to provide additional damping.

Professor K.E. Bollinger and others (ref 1-7), have been involved for the last decade in the design of power systems stabilizers to be added to AVR loops in power plants to enhance damping. Of particular interest in this project was his involvement with Calgary Power Limited in the design of a PSS for units 3 and 4 turbo generators at their Sundance plant (ref 8,9,10). The test application, described in the field test chapter, of the digital controller was an emulation of the analog PSS used for tie-line and local power damping at the Sundance plant.

Sundance unit 3 is a 380 MVA turbo generator located at a plant about 60 miles west of Edmonton. It has been on line, with a power stabilizer enhancing its transient response, since September 1976. Sundance unit 4 is a similar unit that is now also in service with a similar stabilizer.

1.3 The Power System Stabilizer (PSS)

A power system stabilizer is defined as a device that feeds back a signal related to the electric power output or rotor shaft speed in order to provide a "stabilizing" voltage through the AVR.

A generator is sometimes modeled as a multivariable system with the two inputs of field voltage and mechanical power and the two outputs of rotor speed and terminal vol-

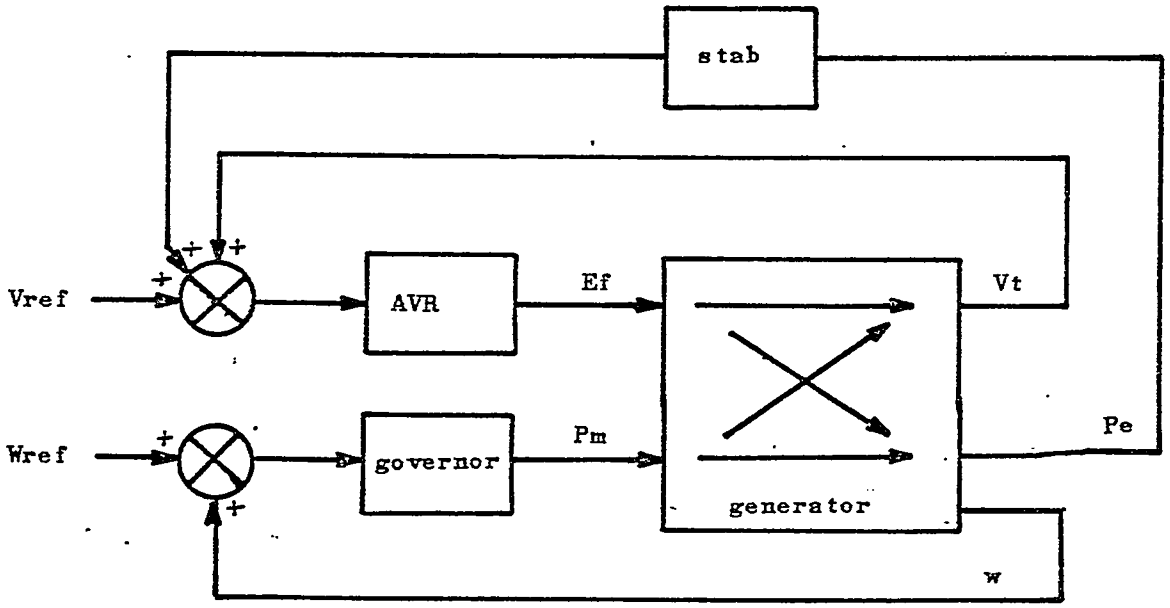


Figure 1.1 Generator Block Diagram

tage as shown in figure 1.1.

The AVR is one of the feedback control loops around the generator. It takes a signal derived from the terminal voltage of the machine, subtracts it from the desired voltage, and uses this error voltage to generate a control voltage for the exciter. The exciter amplifies the control voltage to the level of about a hundred volts and provides a current in the range of a thousand amps to the field of the generator.

The turbine/governor system can be regarded as the other traditional feedback control loop around the generator. The governor measures the speed of the generator shaft, compares it to a reference, and supplies mechanical power as required to keep the shaft speed constant.

A power system stabilizer takes the electrical power signal and feeds it back to the summing junction in the AVR. A "local stabilizer" uses the generator output power as its feed back signal. A "tie-line stabilizer", which is a term that will appear again, is a similar device that measures the power flow along a system tie-line and feeds that signal back, to the AVR, in the expectation of stabilizing the power flow on that line.

The local PSS then, supplies diagonal linkage, or cross coupling, compensation between the speed loop and the terminal voltage elements of the generator transfer function matrix.

The transfer function of the PSS traditionally (ref 2,3,11) consists of three or four poles and three or four zeros. It is not particularilY complex in form, but is difficult to design. The main problem is the difficulty of selecting the correct parameters in order to provide damping. This is attributable to the fact that the power system has interacting control loops that are characterized by high order transfer functions.

Further difficulties arise in the design of a PSS from the rigid time scheduling that is involved in the original construction of a turbo-generator. Time is a tightly budgeted, valuable commodity during such construction. The same time constraints apply to the PSS.

In order for the stabilizer to be installed at the correct phase in the construction of a power plant it must be ordered early. Since the device is typically a custom unit it must be ordered a minimum of a year in advance of the time it is to be delivered. Ideally, the stabilizer should be configured during the commissioning of the generator. The prime benefit of this flexibility would be that the PSS could be configured on the basis of experimental data obtained at the time of commissioning. Practically, the hardware is such that only final adjustment of a few parameters are possible at this point. There are several constaints that lead to this.

A significant point is that specifying a stabilizer that allowed a greater selection of transfer functions during commissioning would result in a much more expensive device that would require even longer to build.

One of the present approaches to PSS design (ref 6) is a preliminary design of the stabilizer with the aid of a computer simulation. The complexity of the multi-machine model, including loads and networks, makes this simulation quite involved. With the generator parameters obtained from the manufacturer and the parameters of existing and proposed transmission lines, system loads and generation facilities, a simulation can be constructed. Using this simulation model a stabilizer can be synthesised.

The controller synthesis is complicated by the fact that the generator is not a single-input single-output device, and is not linear except for small variations about an operating point. The linearization problem is a common one and can be dealt with conventionally. The single-input, single-output assumption considerably simplifies analysis, but must be used carefully, as it may lead to unacceptable perturbations of other system variables.

Once a transfer function form and a set of parameters are settled on, a set of specifications can be drawn up and the construction of the stabilizer tendered. A large part of these specifications will consist of protection criteria such as limiting, disconnect and alarm functions.

When the generator enters the commissioning phase, it suddenly becomes apparent just how good a simulation was obtainable from the design data alone. If all goes well the stabilizer is adjusted to the settings chosen with the simulation and the stabilizer performs as expected. If a similar generator is already operational in the system it is a little easier as simulation variables can be cross checked earlier. If not, the success of the design is heavily dependant on the accuracy of theoretical data and is thus a good correlation of simulated and experimental results may be more difficult to obtain.

It can be seen, then, that the power systems stabilizer is in itself complicated, and that the cost of the hardware is only a part of the price involved.

These factors contribute to making a stabilizer an expensive piece of equipment. They justify what otherwise might appear, without the background, to be an overkill approach in applying the full power of a minicomputer to the problem.

The project described in this thesis ended at the point where the digital controller was constructed and field tested by emulating an existing power system stabilizer on Sundance #3. The performance of the device in this test was satisfactory and the feasibility of the approach was demonstrated.

The device constructed, however, is intended to be more than just a stabilizer. It is intended to be a flexible research tool that will aid in the design of stabilizers and other similar devices.

2 DESIGN OBJECTIVES; A MORE GENERAL DEVICE

This chapter will present an overview of the controller development system from the point of view of what needs to be in such a system, and why. A detailed discussion of the major points can be found in Chapters 3 and 4.

2.1 Flexibility

Flexibility is the principle advantage of a digital implementation. A minicomputer based controller has the potential at least, of not only flexibility with respect to the parameters of any given task, but also flexibility as which task or tasks are to be performed at any given time. Each task, as the word is used here, corresponds functionally to a single-input single-output analog controller.

As presently implemented a typical controller task would consist of an input and output service routine and a series of standard functions such as a washout function with the transfer function:

$$\frac{O(s)}{I(s)} = \frac{Ks}{1+Tds} \quad ; \text{(s is the Laplace operator)} \qquad 2.1$$

This is a function that is often used to reduce the contribution of the controller to zero for a D.C input.

Consequently the effect of the controller would only be felt during transients.

Two more standard functions are the lead-lag function with the transfer function:

$$\frac{O(s)}{I(s)} = \frac{K(1+T1s)}{(1+T2s)} \qquad 2.2$$

and the complex pole zero pair with the transfer function:

$$\frac{O(s)}{I(s)} = \frac{K\,Wn2^2\,(s^2 + 2\,Z1\,Wn1\,s + Wn1^2)}{Wn1^2\,(s^2 + 2\,Z2\,Wn2\,s + Wn2^2)} \;; \qquad 2.3$$

These functions are typically used to synthesize arbitrary transfer functions.

These three functions were all that were required in the Sundance stabilizer and are consequently the only ones presently implemented in the controller development system. It is a simple matter to add further functions should they be required.

In the test case documented in chapter 5, the development system was able to run not only controllers made up of arbitrary blocks with arbitrary parameters, but also was not limited to a single controller at any one time. In that instance a tie-line power stabilizer consisting of a washout and a complex pole zero pair was running simultaneously with a local power stabilizer that contained a washout and two lead-lag terms. During operation it was possible to change any of the fifteen parameters of the two controllers, or to change the form of either controller without opening either loop. Similarily, with the analog controller that was being emulated, one could modify parameters on-line using built in potentiometers, but changing the form of the controller would have been quite difficult. An analog controller is typically a "one of" type of device, individualized for each installation. The digital implementation is much more flexible. It is potentially an extremely useful research device where a wide range of controllers may be investigated during a particular field test situation.

Flexibility is a major advantage, also, in another way. Let's consider a hypothetical stabilizer to be built for permanent installation as a part of a turbo generator excitation system.

The lead time problem mentioned in the previous chapter complicates the synthesis of such a controller somewhat. The difficulty is that the hardware specification must go out a year or so before the device can be delivered. In order that it be integrated and tested in the generator system before rotational tests begin, the controller must be delivered well before the machine installation is complete. Before all this, of course, it is necessary to finalize the controller form and the range of possible values for its parameters. Thus, the simulation work necessary to define the stabilizer must be begun in the order of two years before the machine begins to rotate.

These time constraints can also adversly affect the effectiveness of the controller. Theory and experience advance after the design must be frozen. A better controller (ie one that would provide more damping) could very likely be designed after on-line experience with the machine has been accumulated. In addition, the system itself will change its characteristics over this two year period. As generation and load get larger any given system typically gets more oscillatory. Extrapolation of the system simulation programs gives a good idea of what these changes will be. However, in trying to correct these oscillatory tendencies the situation could come up where the engineer would like to change some already installed stabilizer. With a analog stabilizer, this could be difficult. A digital controller would be quite easy to change because there would be no need to change the hardware at all. Parameters and forms would be widely variable, even after the controller is on-line.

This type of flexibility is inherent in a digital machine. Any desired analog controller can be implemented as a program on a suitably configured computer. The fact that the controller is merely a program allows the hardware (ie the computer) to be specified early, and the form of the controller to be specified (and modified) at a later time.

Yet a third type of flexibility that can be had from a digital controller is the self modifying of transfer function form and controller structure that the computer could allow. A power system is not a linear time invariant system, in spite of how much this assumption simplifies analysis. The transfer function of the generator actually changes both with load and with time as the type and amount of load changes. A given linear time invariant controller can not be expected to be the best possible one for the system at all times. Fortunately, the design procedure based on a linearized time invariant model works well over a relatively wide range. It would seem reasonable to expect, however, that a time varying version of the design, in which some of the parameters changed slightly with the generator transfer function for example, could work better.

A time varying controller is difficult to implement using analog techniques. A digital implementation provides a convenient vehicle for implementing control strategies that utilize control concepts such as optimal, adaptive, and stochastic control.

All of these types of flexibility must be maximized in the design of a digital controller development system. A research tool needs to be as flexible as it is possible to be.

2.2 Performance

Another parameter to be maximized is the speed power product that defines a performance index of the DDC system.

There is a direct trade off between the speed of the controller that can be implemented and the complexity of that controller. The faster the controller must be, the less time is available for computation, and so the simpler the controller must be to run in the available time.

Applications of computers up to this time have tended toward the slow but high task complexity end of this spectrum. This is the area of supervisory control of slow processes, where the power of the computer is used to control many very slow processes with "set points" (desired values) and in logging progressive values of plant outputs, inputs, and internal variables. Computers have been expensive, and consequently they have been used in areas where their contribution to increased system efficiency pays a high rate of return. Computers have been used, for example, to increase the efficiency of data gathering. They can monitor a large number of points, and if necessary can monitor a few points at a very high speed, so that a record of a fault condition can be retained after the event. Improved fault analysis, based on data that would not have been available without the computer, can by itself pay for the computer. Process control is another broad area of application for computers. General purpose computers can handle very many low speed control tasks simultaneously.

The application in this project is about as far in the faster but less complicated direction that it is profitable to go at the present level of technology. The key word here is profitable. Faster tasks can be performed using special purpose computers, discrete logic, or much more expensive but faster minicomputers. The application dealt with in this thesis uses a computer similar in cost to an analog controller, similar in performance, and one that is in high volume production. A computer in volume production gains a number of benefits from the economy of scale, such as lower cost, fewer design bugs, and ease of getting parts, advice and service.

A reasonably complex controller can be implemented in a standard minicomputer if the maximum frequency of interest is below about 10 hertz. The price of minicomputers has fallen dramatically in the last few years to the point where the cost of the computer used in this project is only about $7000. This is in the range of the cost of the analog power system stabilizer installed in Sundance unit 3. In the future the digital hardware cost will become significantly lower.

The applications of supervisory control and implementing compensators (PSS) using DDC are two different points on a given speed-power hyperbola. The speed-power product is a measure, given a particular selection of hardware, of the efficency of the programming of the task. The word speed as it is used in "speed-power" product means the inverse of controller program cycle time. The word power refers to the number of statements, or the complexity of the function performed, in a controller cycle. Programming a controller in Fortran and then in assembler would give two different levels of efficency. If the two tasks performed the same function the "power" part of the speed power product would be the same for each. The speed of the assembler program would very probably be much higher than that of the Fortran program. The assembler program could run up to a factor of ten faster. The Fortran program could probably be written in less than a tenth of the time. This is a common programming trade off. Typically the more general the program the slower it runs. In this application it was necessary to write the controller tasks in assembler and also to rewrite the standard arithmetic floating point functions, trading some unnecessary precision for speed.

The speed power product can be changed only so much by efficient programming techniques. Beyond this point performance increases must come from improvements in hardware. This can involve moving in a number of different directions.

The easiest way to gain an improvement in performance through hardware is to wait until a faster, cheaper computer comes along; hopefully one that is software and hardware compatable with the one you have. At the current rate of progress one needn't wait long for significant improvements.

A second improvement in hardware that would significantly increase the performance of the computer in the application of this thesis is the addition of a hardware floating point processor. This could add nearly a factor of ten to the performance of the system. The second processor might perhaps double the cost of the computer. The limit of trading money for speed at the current level of minicomputer technology would be a system of similar complexity that would operate with a maximum frequency of perhaps two hundred hertz costing around $50,000. This is a crude estimate based on the use of a DEC PDP11/60 in a very minimal configuration for such a dedicated application. This would not be a particularily practical system considering the additional environmental and maintainance problems introduced, but it does give a comparison point to this project. The speed improvement would come primarily from its ability to do a floating point calculation in less than two microseconds, and the use of a high speed cache memory large enough to hold the entire controller program.

Trends in integrated circuit development suggest minicomputers with microprogrammed integral hardware floating point arithmetic will soon be available at lower prices than current mincomputers. In fact the Hewlett Packard 21MX series of minicomputers already has this capability as a minimal cost option.

2.3 Physical Constraints

A set of criteria of a different type from those already mentioned are the physical and environmental constraints.

The digital controller development system used in research must be easily transportable, because it will be moved from site to site. It was this criteria that lead to limiting the support peripherals to a minimum ; namely a CRT device with two integral cassette tape drives. It was possible with some rewriting of the standard software to construct a programming system capable of the full range of editing and compilation functions in this minimally configured hardware. In the present digital system the hardware consists of a Computer Automation Alpha LSI-2/20 16 bit processor with 16K words of memory, and an HP2644A data terminal. Assembly language programs can be edited on the terminal and cassette drives, or in the Alpha with an editor that is part of a core based operating system.

The system must be capable of operating in an electrically noisy environment. Some concern was felt over transients that might occur in the station 110vac supply. Difficulty was encountered in tests conducted in July 1976 at Sundance with a minicomputer configured as a Fourier Analyser. Supply transients were enough to cause program changes and require reloading of the system software. This occurred during the commissioning of unit 3, however, at the time abnormal switching was being done. No problems were encountered with the DDC during the field test in Feburary

1977, in spite of there not being an additional external supply filter on the computer. Such a filter was planned, but omitted because of time constraints. An external supply filter should be added to the system. If the uninterruptable AVR power supply could be used in future test situations, the problem would be circumvented.

A signifiant amount of electrical and magnetic noise is present in the power station environment. This was considered in the design of the input and output sections of the DDC. The input signal was filtered and buffered with a device to remove common mode noise. The output was made as tough and as fixable as possible, in that output buffering was done with 1458 operational amplifiers, which are very common, quite inexpensive, and also resistant to fault conditions.

The common mode noise problem arises not only from inductive and capacitive coupling to signal lines, but also from the multiplicity of signal grounds in the turbo generator system. The large number of grounds involved imposes the condition that all signal inputs and outputs must be differential.

The input signal is referred to the computer ground with a device called an isolation amplifier. This is a transformer isolated amplifier that can measure low amplitude signals in the presence of up to 2000 volts of offset. The device was developed for biomedical applications and so is specified to have under no circumstances an input current of more than 1 microamp.

The output must be similarily disassociated from the computer ground. Ground loops can cause the turbo generator to trip off line. At Sundance the output isolation was done with custom built transformer isolated buffer amplifiers.

2.4 Reliability and Repairability

A further major design consideration is the problem of reliability and repairability. The particular computer used in this project has a reputation for reliability. The company that manufactures it is the only one in the industry that has a one year warranty. It has a low number of interconnections. The entire processor, for example, is on a single card. The mean time to failure was thought by the manufacturer to be in the order of 40,000 hours. The figure quoted was not thought to be accurate because of the lack of failures up to that time. 40,000 hours is quite a few years operation for most computers. This length of time amounts to about four years of continuous operation. This would probably be a reasonable reliability to expect of a controller.

It is of course completely adequate for a research tool that has by nature much lower reliability requirements.

What happens when the computer does go down? Since an analog controller is a familiar device, if it failed it could probably be fixed on site by an engineer working for the Utility. The purchase of spare boards with the analog controller virtually guarantees this. The engineer on the other hand can not reasonably be expected to be conversant with all the different brands of computers there may be associated with any given generator. This requires that there be a different level of diagnostic associated with the digital controller. It is not reasonable to fly in a specialist every time a given controller stops functioning. The problem is not the cost of getting such a specialist, but the much higher cost of lost generation. If the generator must run at a reduced load because a stabilizer is down, that load must be picked up somewhere else in the grid, usually by more costly generation facilities. The cost of the stabilizer down time could amount to thousands of dollars an hour. Thus it is desirable that the stabilizer be repairable on site.

The digital stabilizer must tell the operator when it is working, if is working, or what is wrong with it. Further there must be spare parts available. The choice is whether to have spare boards or spare units available. Where high reliability is necessary the choice must be both spare boards and units. The AVR for example is typically completely duplicated right up to the SCR's, which also have switchable extra capacity.

High reliability requires both digital and analog controllers alike be completely duplicated. Where the spare board level of reliability is adequate the digital stabilizer must help the engineer or technician by being self diagnosing.

It is much easier for the manufacturer, as opposed to the end user, to repair the individual computer boards. Specialized equipment and specialized skills are required. Typically it requires a computer to efficiently fix a computer, as there are too many signal paths to conveniently test by hand. The fact that the board must be repaired else where need not be a handicap in fixing the overall computer system on site. It is possible to write diagnostic functions into the controller software to monitor the functioning of the computer and to isolate faults to a single printed circuit board.

It is necessary, then to build these diagnostic functions into a digital controller. With them in place it is possible that the digital stabilizer could be easier to repair than the analog stabilizer. This must be made an objective of the design.

The diagnostics should include checking the proper functioning of the processor, the memory, the control algorithms, and the proper functioning and accuracy of the I/O hardware. Software problems should be self-correcting as much as possible. The I/O hardware should be tested by generating and monitoring an analog test signal. If a harware error is diagnosed the computer should trip off line. This should be done in a fail-safe manner, by having an external circuit monitor the presence of a pulse train being output by the program. If the train of pulses stops, or changes frequency, the external circuit opens the feedback loop, containing the computer, in a nondestructive manner. In addition the computer should communicate the nature of the event that caused the disconnection.

If the computer trips itself offline, the appropriate alarms would be given and the standby unit (if there is one) would go on-line. It might not be unreasonable, where high reliability requires a dual processor installation, if the external monitoring function mentioned in the previous paragraph were performed by the second processor. In this case each processor could monitor the functioning of the other by comparing the two outputs resulting from the same input. If one of the computers tripped off line, of course, the monitoring function would have to be taken up again by an external circuit.

This is a minimal statement of the necessary diagnostic functions of the digital controller.

2.5 Support Software

It is desireable to have compilation and editing functions in the development system when it is in the field. This would not be a problem except that for the convenience of increased portability, the reduction of the hardware cost, and the reduction of interconnections for reliability, the number of peripherals in the DDC system was reduced to one. A core based operating system that enabled editing and assembly of programs using the full capabilities of the Hewlett Packard 2644A terminal was not commercially available. It was necessary to partially rewrite and organize the software supplied with the computer to make available these capabilities. In particular, software driver routines to read and write source and binary data onto the terminal cassette tapes were written and interfaced to the standard system editor/assembler "Omega". A bootstrap device was constructed to issue the necessary escape sequence from the computer in order that the computer's hardware bootstrap loader could load binary tapes from the terminal (page 59). Further, the standard relocateable loader was interfaced to Omega to enable binary output to be loaded into a high area of core instead of being output. This enabled modification and testing of programs to be carried out without the input or output of intermediate object files. The last major segment of the core operating system was the interfacing of the standard debugging monitor to all these routines to act both as a monitor and a debugging monitor. An operations and command summary of the core operating system is included as appendix A.

Most of the software development was done using a cross assembler resident on the campus IBM 370/158. The cross assembler received its input from cards and transferred output through specially written transfer and translation routines to the HP 2644A terminal under TSO. The cross assembler was purchased from Computer Automation Inc.

The cross assembler was used at first to develop the operating system, and later to produce good quality listings. Modifications to the standard software and the controller software totalled approximately 3000 assembler statements.

3 SOFTWARE

3.1 Controller Program Overview

The controller program is segmented into four modules connected as per figure 3.1. The numbers in the figure for cycle times and number of passes are as they were in the Sundance field test. They are representative and may be changed through the monitor. The same numbers are used in the following paragraphs to facilitate the discussion.

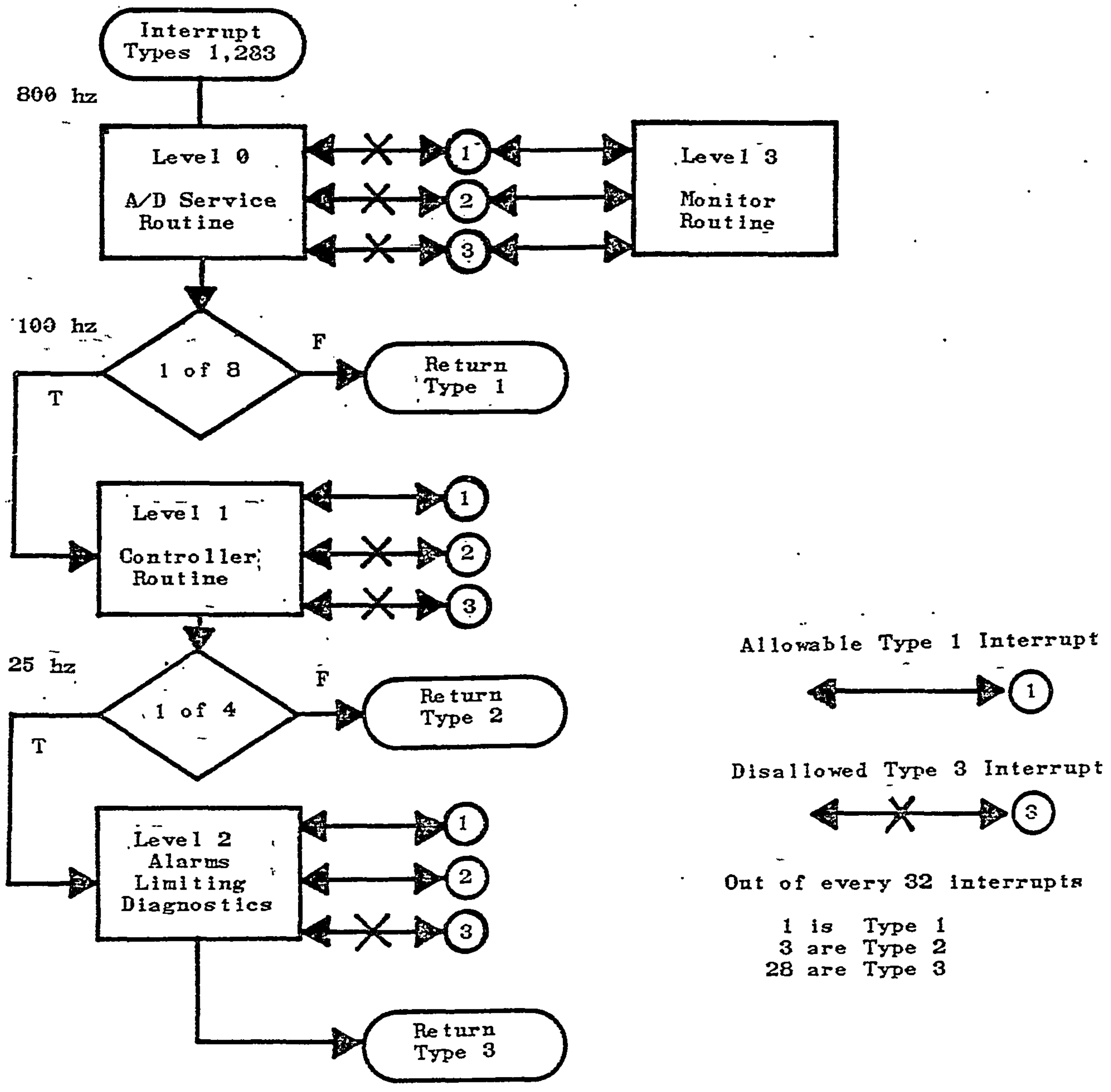
The four modules are named level 0, level 1, etc as a representation of their relative priorities, level 0 being the highest and running the most often.

Levels 0,1 and 2 are interrupt routines. Level 3 is the background (monitor) routine. Each routine is interruptable by the routines with lower numbers. For example, level 2 is interruptable by levels 0 and 1. Figure 3.1 shows the different types of interrupts that are allowed and disallowed in each level.

The interrupt type numbers, shown in the figure, are slightly different than the level numbers. Each time an interrupt of type 2 occurs all the routines of level 0, 1 and 2 are run. Similarily a type 1 interrupt runs the

FIGURE 3.1

CONTROLLER PROGRAM BLOCK DIAGRAM



routines of level 0 and 1. If an interrupt of an illegal type occurs it signifies that the time allowed for the execution of one controller cycle is too short, and consequently results in the orderly shut down of the controller.

Level 0 is the analog to digital converter service routine. This routine is initiated by each real time clock (echo) interrupt. These interrupts occur at a fixed rate, 800 hertz in the diagram, and can interrupt any of the other modules. The service routine requires about 100 microseconds to run. The routine aquires a sample of the voltage on each active input channel, and averages these inputs over eight consecutive clock periods.

The input channel averaging is a crude kind of low pass filter that uses binary instead of floating point arithmetic, and consequently runs much faster. This is possible in the input routine because the range of possible input values is small; limited by what the A/D can distinguish. It is not necessary at the beginning of the controller routine to allow for broadening of the dynamic range of the signal by prior arithmatic operations. The addition of an appropriate set of weighting coefficents could convert the averaging routine to a finite impulse response filter, giving a much better filtering performance. Up to this time only the averaging approach has been used. The aliasing function has been performed with an analog filter. The averaging routine improves the performance when compared to sampling a single point at calculation.

After averaging each of the active input channels the routine exits, reenabling the interrupts, and returning to the monitor, or the module from whence it came. Every eighth time the level 0 routine exits, processing is diverted into the level 1 routine.

Level 1 contains the controller program. The routine first makes a copy of the A/D service routine active input block, and then sets this block to zero to reinitalize the averaging process. At this point the interrupts are reenabled. These inputs are then converted to floating point and passed on to the active controller.

The controller (ie level 1) consists of a number of functional blocks, hereafter referred to as controller segments. The segments are invoked as series of entries in the program stack. Each block has two entries in the stack, a pointer to the routine, and a pointer to its associated parameter block. Typical blocks would be single-input single-output reset, or lead-lag, functions.

The series of functions that make up the active controller is performed as specified by the program stack. The first function is invoked by the A/D service routine and the last entry is always a call to the routine that does the D/A service and then performs all the integrations invoked by the various blocks. The clean up routine then passes control to the level 1 exit routine. Every fourth exit from level 1 includes a call to level 2.

Level 2 contains the alarm and disconnect functions. These are presently in a very rudimentary form. The functions presently implemented include a disconnect with a flag to the terminal if cycle overlap occurs, and a patterned blinking of the console lights to tell the operator that the program is running. Exit from level 2 is always to the monitor routine.

The monitor routine is the interface between the operator and the controller program. This module takes input from the terminal and performs the functions that the operator wants done. In the present version this is also in a rudimentary form. The operations that the monitor can perform include modifing memory locations, resetting the active controllers by zeroing the integrator outputs, disconnecting the active controllers, and starting the execution of a controller. The modify memory command is used to modify controller parameters, by changing their associated parameter blocks, to modify controller forms, by changing the program stack, and to modify global parameters such as the cycle time of the A/D routine and the active or passive status of I/O channels.

3.2 Controller Program Infrastructure

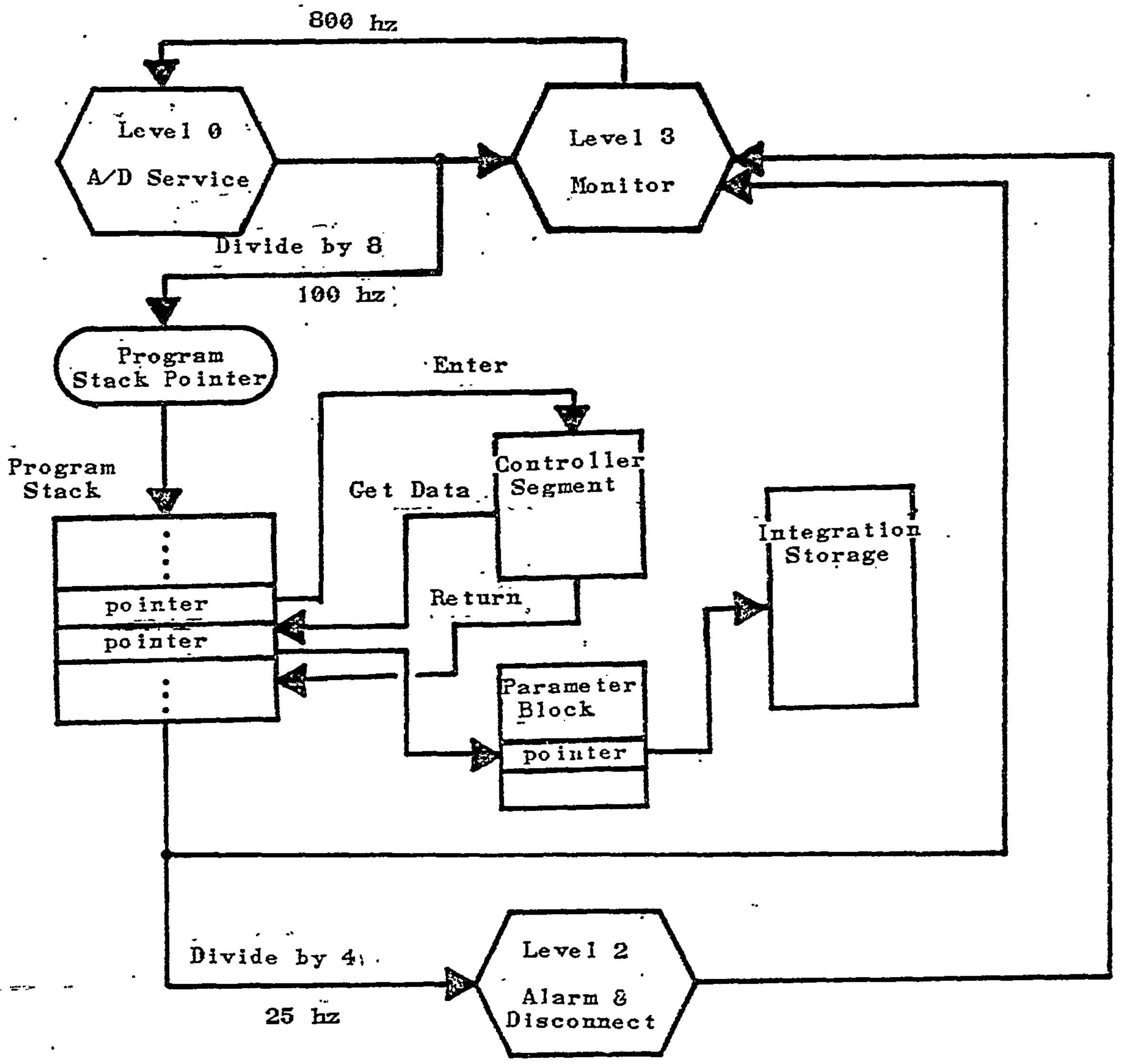
A very convenient feature of the controller program is the ability to modify controller characteristics without opening the control loop. Controller forms and parameters are modifiable while the controller is on line, with a minimum amount of disturbance to the system.

The mechanism for modifying the controllers and their parameters is facilitated by the data structure of the program. All linkage functions are performed through pointers as diagrammed in figure 3.2. Control is transferred from the input service routine through the program stack pointer and the first entry of the program stack to the first controller segment.

The program stack is a block of pointers containing two entries for each controller segment; a control linkage pointer, and a data linkage pointer. The controller segment, which might typically be a lead-lag network, accesses its data through the next entry in the program stack, performs the required controller function,and passes control through the pointer following its data linkage. The data linkage

FIGURE 3.2

POINTER BASED CONTROLLER FUNCTION LINKAGE



pointer points to a block containing both parameters and further pointers. All of the integrator outputs for example are referenced through pointers in this block in order that the integrator outputs for the entire program may be in the same block of memory. This simplifies the limiting and reset functions. Most parameters of the controller segment are contained in the data block.

Integrations required by the controller segment are performed after the D/A service routine, in order that there be the minimum possible delay between input and output. Each segment pushes a pointer corresponding to the integrator onto an active integrator stack, in order to have that particular integration performed.

Besides the input and output service routines, the controller segments are the only blocks containing executable code. These segments are useable more than once in any controller, because their data is specified only with a pointer. Different data pointed to by a different pointer allows the same segments to perform multiple functions.

Each controller segment calls the segment following it on the program stack. The last entry on the program stack is a pointer to a clean up routine. This routine performs the D/A service, and then performs all the integrations

/

requested by entries in the active integrator stack. Control is then passed to the routine that performs an orderly exit from level 1.

The structure of the program allows the controller to be easily modified by a background program while it is in operation. To change the data for a controller segment for example one sets up a second block of data the same as the one in use, modifies that block with the desired changes, and then merely substitutes a pointer to change one block for the other. This substitution requires a single machine instruction, and therefore independant of device interrupts. This allows multiple changes simultaneously, and allows the changing of floating point constants which occupy two consecutive locations in memory without disabling the interrupts.

3.3 Controller Implementation; Mathematical Structure

There are a number of possible ways of implementing controller type structures in a digital form. For example it is possible to implement a controller in a completely general way using the state-space approach as shown in figure 3.3. Here all that is required to obtain the next output point is to perform a simple series of matrix multiplications and additions. Individual operations are quite

FIGURE 3.3

State Space Controller Implementation Example

$$\frac{Y(s)}{X(s)} = \frac{Ks\,(s+1)\,(s+6)}{(s+2)\,(s+3)\,(s+5)}$$

$$Y(s)\,(s^3 +5s^2 +6s +5s^2 +25s +30) = X(s)\,(Ks^3 +7s^2 +6s)$$

$$sY(s) -KsX(s) = 7X(s) -10Y(s) +1/s[6X(s) -31Y(s) +1/s[-30Y(s)]]$$

$$\dot{Y1} = -30Y(s)$$

$$\dot{Y2} = 6X(s) -31Y(s) +Y1$$

$$\dot{Y3} = 7X(s) -10Y(s) +Y2$$

$$Y3 = Y(s) - KX(s)$$

$$\begin{bmatrix} \dot{Y1} \\ \dot{Y2} \\ \dot{Y3} \end{bmatrix} = \begin{bmatrix} 0 & 0 & -30 \\ 1 & 0 & -31 \\ 0 & 1 & -10 \end{bmatrix} \begin{bmatrix} Y1 \\ Y2 \\ Y3 \end{bmatrix} + \begin{bmatrix} -30 \\ -25 \\ -3 \end{bmatrix} X(s)$$

$$\dot{\bar{Y}} = \bar{A}\ \bar{Y} + \bar{B}\ X(s)$$

$$\bar{Y} = \bar{Y} + \dot{\bar{Y}}\,\Delta T$$

similar,thus few statements and consequently relatively little program memory is required. A drawback of this approach is that the generality of the mathematics does not take full advantage of the sparseness that is usually present in the matrices.

The state space method, as well as any other method of software simulation of analog controllers, suffers from the problem of limiting. In an analog controller integrators are limited to the maximum voltage they can output. They can not put out a voltage higher than their supply voltage. This is a useful characteristic under some circumstances. If a feedback loop is opened, for example, an integrator might be put in the position of having a constant non-zero input. Its output would then integrate up to the supply voltage and remain there. A software integrator is not naturally limited in such a manner. It is capable of outputting values up to the largest number representable in the computers number system. If it has integrated to a large number, and is then put back on line a significant transient will result.

In an analog controller the problem is dealt with by the supply limiting, and by the addition of limiters to the circuit to keep the voltage at a given point within given limits. In a digital controller the problem can be dealt with in a number of different ways. All the integrator inputs could be set to zero when the loop was opened, for example. This would be a good approach if the digital controller knows when it is off line. However, it might not be practical to inform the controller in all circumstances when the loop is opened. It would seem reasonable then to include facilities in the controller for limiters.

In the state space simulation approach, unless care is taken to relate state variables to real world variables (which is not unreasonably difficult), it can be difficult to put limiters into the controller.

The state space matrix multiplication method is definately the most convenient simulation technique to use. The limiting problem mentioned in the previous paragraphs does not make it much more difficult to program. The zero multiplication difficulty, mentioned at the beginning of the section, effectively disallows it from use in a time critical application with a serial (as opposed to array) processor. Another factor that would also remove it from consideration is the difficulty involved in changing the order of the controller. In order to change the order of the controller transfer function it would be necessary to (dynamically) change the order of all of the matrices associated with that simulation, which is a programming problem of significant difficulty with a serial processor.

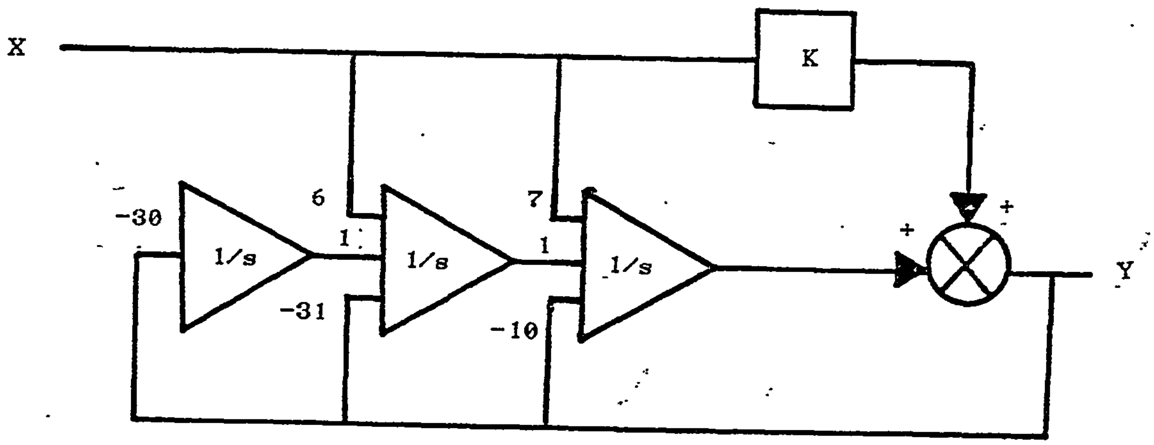
All three of these difficulties are overcome by simulating the blocks of the equivalent analog computer diagram that can be derived from the transfer function, as in figure 3.4. By using this stucture limits can easily be applied to the outputs of each integrator, and the many zero multiplications inherent in the matrix statement of state space simulation are eliminated. As well this approach has the advantage that most engineers have had previous exposure to this analog simulation method. The primary advantage however is the modularity of the resulting simulation program. It is possible to express any given transfer function as a serial sequence of operations on an input signal. These operations would be blocks of code that performed for example reset, lead-lag, or complex pole pair functions.

This last property, that of the controller being segmentable into removeable blocks, is in fact a necessary property of any controller expression to be used in the development system. The application requires a modifiable controller structure. There are are two more controller implementations outlined hereafter that fulfil this requirement.

One of these approaches is to break the controller transfer function up in a partial fractions expansion and implement each of the resultant sections independantly.

FIGURE 3.4 DISCRETE INTEGRATORS

(Same transfer function as fig 3.3)



$$Y(s) - KX(s) = 1/s(\ 7X(s) - 10Y(s) + 1/s(\ 6X(s) - 31Y(s) + 1/s(\ -30Y(s)\ )))$$

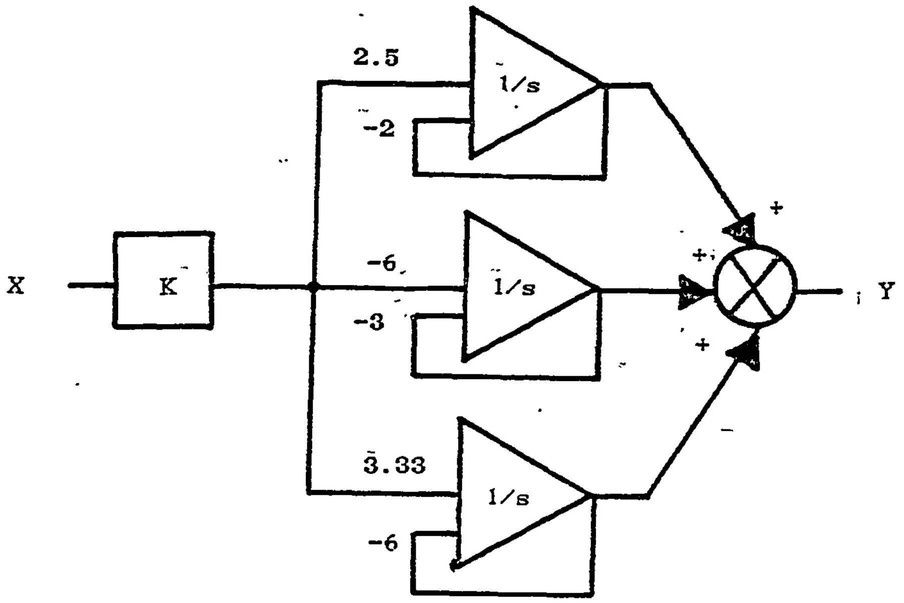


FIGURE 3.5 PARTIAL FRACTIONS

(Same transfer function as fig 3.3)

$$\frac{Ks\ (s+1)\ (s+6)}{(s+2)\ (s+3)\ (s+6)} = \frac{2.5\ K}{(s+2)} + \frac{(-6K)}{(s+3)} + \frac{3.33K}{(s+6)}$$

This idea gives some advantages in a digital implementation. All of the sections receive the same input data, and so there is no need to set up a structure to transmit output values to the next stage. This simplifies the programming involved. A second advantage is that the outputs of of the stages are combined in an additive manner. A multiplicative product would not give as much accuracy as a additive product. Disadvantages include difficulty with multiple poles, which change the form, and complicated calculation of parameters when the transfer function contains many poles and zeros. This development is shown in figure 3.5

The other approach is one that is referred to here as the double adder expression. This is similar to a recursive formulation in a digital filter (ref 12 pp 43). The form is obtained from the transfer function:

$$H(s) = \frac{N(s)}{D(s)} = \frac{Y(s)}{X(s)} \qquad 3.1$$

by creating a variable A :

$$\frac{Y(s)}{N(s)} = A = \frac{X(s)}{D(s)} \qquad 3.2$$

$$Y(s) = AN(s) \quad ; \quad X(s) = AD(s) \qquad 3.3$$

The form can be set up by subtracting some terms in A from the input to give A, and then summing A and some terms in A to give the output. The flow diagram that results is interestingly symmetrical. This form has some of the advantages of the partial fractions approach in the additive products, without the associated difficulties with multiple poles and complicated calculations. An example of the double adder form is developed in figure 3.6

All of the above implementations suffer from inaccuracies in integration if delta t is too long with respect to the fastest time constant in the controller. Uram (ref 13) suggests that a factor of at least 8 or 10 is required for the ratio of the shortest time constant to delta t if 2 or 3 percent accuracy is enough.

A z transform method could eliminate this type of inaccuracy. A z transform approach, however, adds a significant amount of difficulty in return for this exact solution.

In order to successfully emulate controllers it is necessary to be able to implement specific analog blocks, such as a reset function for example, in a form that lends itself to flow diagram algebra. There are two significant points here.

First an analog function must be translated into a z transform version. This translation does not give as

FIGURE 3.6 DOUBLE ADDER IMPLEMENTATION

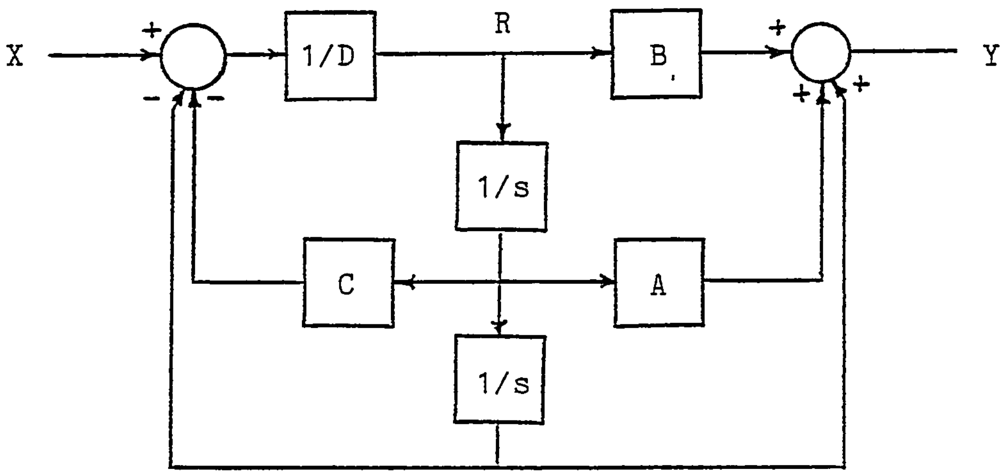
$$\frac{Y(s)}{X(s)} = \frac{K\,(s^2 + As + B)}{(s^2 + Cs + D)}$$

$$\frac{Y(s)}{K\,(s^2 + As + B)} = R = \frac{X(s)}{(s^2 + Cs + D)}$$

$$Y(s) = R\,K\,(s^2 + As + B)$$

$$X(s) = R\,(s^2 + Cs + D)$$

where A,B,C,D, and K are constants, and s is the Laplace operator.



satisfactory a result as developing the controller function in the z domain. Gold and Rabiner (ref 12) show four methods of translation from the s plane to the z plane. None of these are completely accurate. Each method has its advantages and disadvantages.

Secondly, the translated version has to consist of blocks that are multiplicatively cascadable, in the same sense as the Laplace transform expressions of the blocks. That is to say that the translation of the Laplace form of a transfer function to the z plane image must be a linear operation. This is not the case for all four of the above mentioned mappings.

In the present version of the controller system the standard, conceptually simpler, simulation techniques based on the approximate solution of the controller differential equations by piecewise integration were used. They proved to be an adequate first order approach. The performance of the system as it is will provide a base on which to judge more sophisticated techniques.

In the prototype system tested at Sundance #3 different simulation techniques were used in the different controller segments. In particular the complex pole zero pair block uses the double adder implementation, while the reset and lead-lag blocks use the discrete integrator approach. Each approach has its advantages and disadvantages. The present segment structures were arrived at by trading off the speed of the implementation against the simplicity of expression of the parameters. Different controller segments gave different best compromises.

3.4 Mathematical Operations

The limiting factor in the digital implementation of controllers with respect to the speed-power product is largely the time that it takes to do a mathematical operation. If one uses a floating point format of some sort for internal numerical representation the representation of numbers encountered can vary over a large range. On the other hand, the amount of precision that is required is determined by the signal to noise ratio obtainable in the real world. One can buy eight or ten bit D/A converters for a reasonable price, but sixteen bit ones are extremely expensive. The internal numerical representation needs to be precise enough to retain the accuracy necessary for the output device, through a short series of calculations.

Some typical numerical representation forms and associated operation times are as follows:

-16 bit fixed point addition	2us

-16 bit fixed point multiplication	15us
-32 bit fixed point addition	30us
-32 bit fixed point multiplication	100us
-24 bit floating point addition	150us
-24 bit floating point multiplication	250us

All of the above operations are typically available in supplied hardware or software. The 16 bit fixed point operations are not feasible because of the lack of resolution. The 32 bit fixed point operations have the necessary resolution but could over or underflow under some conditions. They would also require careful controller segment design to maintain the resolution. The floating point is the best approach. It allows a large dynamic range. However, when implemented in software as assumed in the times above it is slow and more precise than necessary. A good compromise would be to implement a reduced precision floating point software package that is faster. Such a package was implemented (see appendix D) with a precision of 14 binary digits plus sign (a convenient number from algorithm considerations) and a speed for both addition and multiplication of 100us.

Both the 14 bit and 24 bit floating point storage requires two words of storage. In the standard software the first of the two words contains 8 bits of exponent, seven bits of mantissa, and a one bit sign. The 14 bit floating point format uses this word for an eight bit exponent and a one bit sign only.

Speed is obtained in the 14 bit floating point operations because it is no longer necessary to maintain multiword precision in the mantissa operations, and it is not necessary to shift the extra seven bits of mantissa into the first word of the output. In the multiplication routine, where the largest increase in speed is obtained, this translates into one sixteen bit integer multiplication being necessary instead of three, and at most two shifts to perform the normalization.

If still more speed were required a floating point processor would be the route to take. Such a processor could do a floating point addition or multiplication in a minimum of about 10us (a floating point processor does the operations in hardware).

The floating point package written for the digital controller was written with the supposition that a floating point processor might be added to the controller at a later data, such as when bit slice processor technology reaches the point where such an application becomes relatively uncomplicated. It should be relatively easy to interface such a device to the existing controller program.

3.5 The Controller Maximum Frequency Limitation

The Sundance unit 3 analog stabilizer has adjustable time constants with minimum settings of .01 seconds. In operation the minimum time constant was set at .12 seconds. The analog stabilizer consists of a reset function, and two lead-lag segments. This could quite reasonably be considered a typical application of the controller development system.

What is the maximum performance capability of the DDC system? A reasonable way to answer this question would be to examine the timing limitations in such a typical application.

The Sundance unit 3 DDC stabilizer was implemented in the field test with the cycle time of the controller program set at 10 milliseconds. Of this time the analog to digital conversion routine took up the a larger fraction than any other routine. The input channel was sampled 8 times per cycle, at 1.25 millisecond intervals, using a total of 800 microseconds of processor time. These eight samples were averaged, and the resulting output was passed to the controller proper.

The controller reset function required one floating point addition and two multiplications to complete; a total of 350 microseconds. The two lead-lag segments required two additions and two multiplications each; a total of 450 microseconds for each segment. The digital to analog convertion required about 100 microseconds.

The last part of the controller routine was the integration routine that performed the necessary three integrations. Euler integration required one multiplication and one addition for each integrator. This amounted to 200 microseconds for each integration.

The total time required for the controller routine was then about 2.8 milliseconds. In addition to this it was (and is) necessary to have some time in the cycle for the "overhead" functions such as limiting, alarms, and the monitor. Since these functions can be spread over many cycles of the controller, it is only necessary to allow about 500 microseconds, or 5 percent of the cycle for their execution.

Consequently the minimum cycle time of the controller was about 3.5 milliseconds. A second controller of the same complexity would boost this time to about 5.5 milliseconds. This is not twice the time required for one controller because the overhead functions and the A/D functions need be performed only once per cycle.

In the field test, at one point both the tie-line and the local power stabilizer were running simultaneously, both using the more time consuming trapezoidal integration method, with the cycle time of the controller set at 12 milliseconds. This is about the useful limit of the present software and hardware of the current version of the DDC system.

The cycle time of the controller could have been reduced about forty percent in this particular application if the tie-line stabilizer task had been run in level 2, instead of level 1. This would have been possible in this case because of the disparity in maximum bandwidth between the two stabilizers. The maximum external power system frequency of interest in the tie-line stabilizer was below one hertz. This approach is the most promising of the software methods for improving the capabilities of the DDC system.

3.6 Development system software

The software required to do the necessary assembler programming and debugging consists mostly of standard programming aids supplied by the manufacturer with the computer. However the constraint on the system of portability made it desirable to modify this software and modularize it for use in the system.

A standard Computer Automation assembler/editor called Omega (ref 15) was used in the system. This is a completely core resident program that allows a programmer to enter and edit assembler source, and then to compile the source into a loadable format. It was necessary to modify this program to support the cassette tape drives on the HP terminal as input and output devices. A further modification was a linkage through a loader program to an area of memory in order that core could be specified as a valid output device.

A second major support program from Computer Automation is a debugging monitor called Debug (ref 15). This program allows the programmer to modify and read data in the memory through the keyboard, and control program execution. The program was modified with the addition of a few functions to act as both a debugging monitor and a system moniter in the system of programs used to develop the controller programs.

The first version could be exited to from Omega and the loader programs and had a swap function added that would replace the lower 2k of Omega with the output core buffer of Omega. This allowed the assembly of a program into a core buffer, exit to Debug, swapping the newly assembled version of the program into the base page, performing debugging modifications, and then swapping Omega back in with a intact text buffer. The only reasons for loading any thing from tape would be a runaway program that modified program memory, or a desire to work on a new program. This made programming very convenient.

A third support program from the manufacturer was a cross assembler (ref 17) that ran on the IBM 370 and produced binary for the Alpha from assembler statements read in as cards. Difficulty with getting binary data out of the 370 resulted in several consecutive program systems to perform the transfer. The final system used involved getting the output from the cross assembler in printable characters (hex) and transferring them to the HP 2644A terminal on a TSO link. This version was then translated to binary in a program running on the Alpha. The hex data was read off of one tape, translated to binary and written on the other tape.

This cross assembler was used because it allowed very large programs to be assembled, it gave a very nice listing, and the card medium is well suited to both editing and permanent storage of programs.

4 HARDWARE

4.1 The Hardware Description

The primary controller system component is a 16 bit Computer Automation Inc Alpha LSI 2/20 minicomputer with 16k words of memory. The computer itself consists of four printed circuit boards mounted in a 7.5 inch high, 19 inch rack mountable chassis. The chassis consists of a power supply, a mother board with the computer bus on it, and a programmer's console. The four circuit boards are: the processor, 8k words of core memory, 8k words of semiconductor memory, and an input/output board.

The processor is a 15 inch square circuit board containing all the processor functions, including standard hardware binary multiplication and division, a real time clock, a jumper speed selectable 110 to 9600 baud serial interface, and a hardware bootstrap loader.

The 8k word core memory board is also a standard full board 15 inches square. It contains high speed core memory (980ns cycle) intended to contain the controller program in the final configuration.

The 8k semiconductor memory board is a half size board.

It is inexpensive volatile memory intended to be used during the development of the controller software providing space for the editing and compilation functions.

The fourth board is the full sized input/output board. This board contains eight channels of analog to digital converters, four channels of digital to analog converters, and the interfacing required to tie them to the computer bus. The board was constructed on a standard wire wrap bread board purchased from the computer manufacturer. The circuitry consists of about 75 MSI packages ranging in complexity from quad bus drivers to a complete eight channel 12 bit A/D converter system.

The eight channel A/D converter system is a hybrid package purchased from Data Translation Inc. The package contains eight differential inputs multiplexed into a single twelve bit ten microsecond successive approximation analog to digital converter. Individual channels are addressable, or automatic sequencing over a range of channels may be initiated under program control.

The D/A converters are eight bit multiplying devices from Motorola. External 741 type operational amplifiers perform the output buffering and current to voltage conversion.

The I/O board supports four of the five possible I/O transfer methods to the CPU. It is capable of input block transfer at 100 kilohertz on one channel while leaving about 30 percent of the processor time available for background tasks. Alternatively all eight of the input channels can be sampled, each at 150 microsecond intervals, while leaving close to two thirds of the processor's time free.

Additional system hardware in the cabinet includes the I/O buffer. This contains sixteen channels of filters and one (expandable to four) isolation amplifier. The isolation amplifier is from Analog Devices. It is a transformer isolated device with a 2.5 kilohertz bandwidth and 2000 volt isolation capability. The maximum input current with input voltages below the isolation maximum is one microamp.

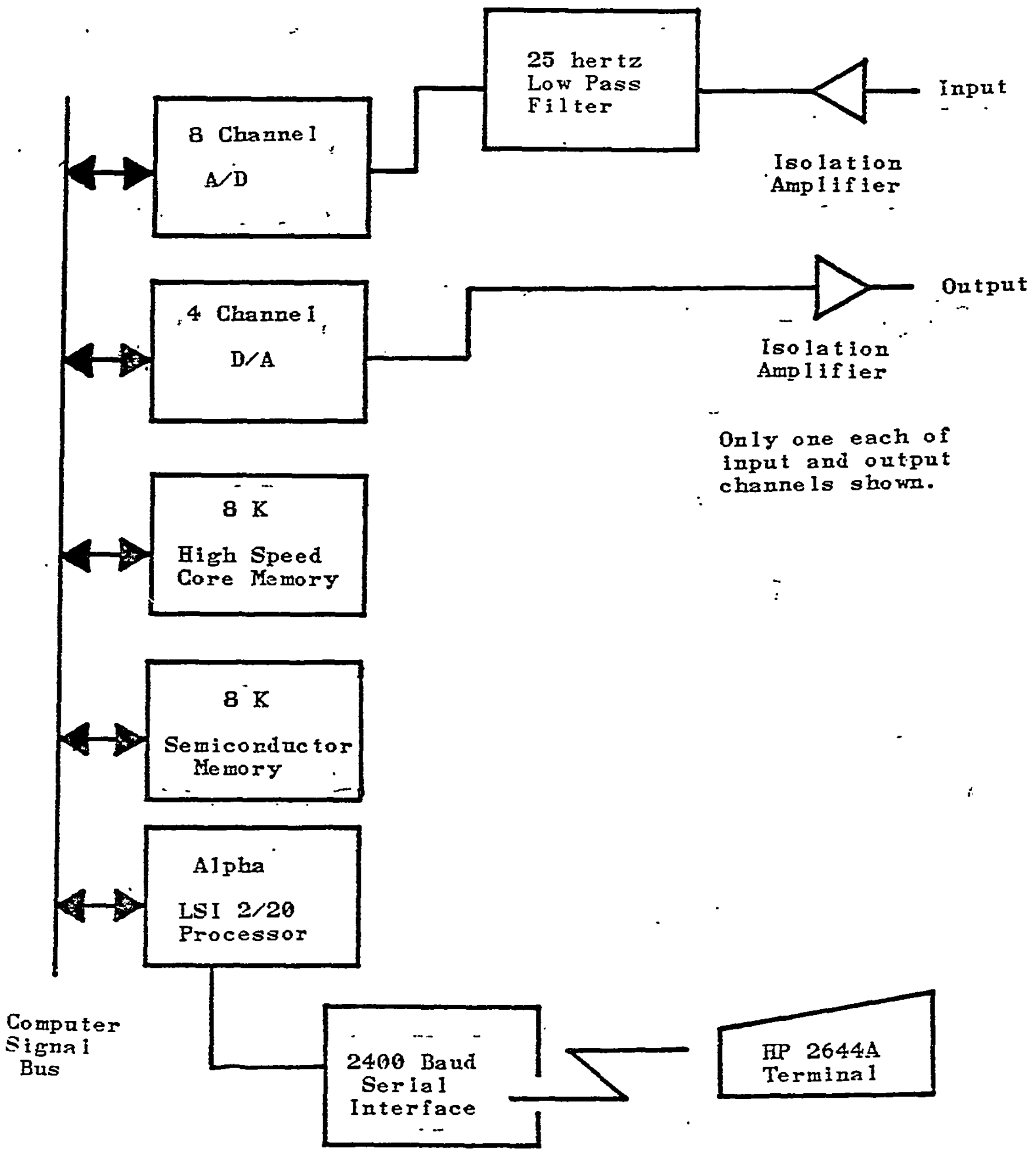
The cabinet also contains a device to trigger binary dump operations from the terminal. The device injects an escape sequence onto the computer's end of the serial communication line.

A block diagram of the complete development system hardware is figure 4.1.

4.2 I/O Hardware Optimization

The particular computer purchased for this project has

FIGURE 4.1 BLOCK DIAGRAM OF SYSTEM HARDWARE



quite a flexible I/O structure. There are five I/O methods, with rationals varying from very simple hardware but, costly in processor time and programming overhead, to very fast, but requiring complicated hardware (ref 16). All five of these approachs were evaluated with the goal of minimizing the processor time required to service the interface, while still keeping the hardware compexity within reason. Four of the methods required small amounts of hardware. Consequently all of these input structures were built into the I/O interface. Direct memory access was not included.

Three of the five methods mentioned are not unique to this particular computer, but are standard approaches. These are programmed I/O, interrupts, and direct memory access. The other two are variations on the first two for block transfers, making the standard techniques more powerful.

Programmed I/O is the simplest approach, and a good benchmark. The hardware required consists of a data buffer connected through three state buffers to the data bus of the computer, and a flag flip flop to say when the data in the buffer is valid. Three or four packages of gates decode the device address and input control signal and put the data or flag on the bus when requested. In order to transfer data the program samples the flag from time to time and, when there is data to be transfered, executes an instruction to input the data word. This hardware is a subset of the interface implemented.

In order to transfer a number of channels of data using this method the procedure for each would be as follows. Send the desired channel and a start signal to the A/D. Wait for the a/d done flag. Read the resultant data and store it in the appropriate location. The time for a channel, using the A/D purchased that has a conversion time of ten microseconds, would be about twenty microseconds. Added to this to give the total input subroutine overhead would be the time required to enter the routine, save the registers, and exit from the routine, amounting to perhaps another thirty microseconds. The total system overhead would also include the service routine for the real time clock, a minimum of twenty five microseconds set up time plus five microseconds per clock tick.

The time required to acquire a single channel would be 80-100 microseconds. A group of eight channels could be sampled in about 330 microseconds or 40 microseconds per channel. Sampling a number of channels at the same time is faster because the overhead routines would only have to be executed once per group of inputs.

With an external clock (ie no clock servicing) these rates would be 20 khz and 40 khz respectively. Unless a crystal clock is used externally, some loss of precision would result.

At the other end of the spectrum is a technique known as direct memory access. This approach requires quite complicated hardware, but has the advantages of high speed and little or no program intervention being necessary after the original start up. This is the only one of the five techniques explained here that wasn't implemented in the analog interface.

A typical DMA write cycle consists of the following series of events: The interface first issues a signal that tells the processor that it would like control of the bus. After the processor is finished the current instruction, it relinquishes the bus, the interface puts an address on the address bus and the first data point on the data bus. It also must issue a memory write signal, and wait for the memory done signal, before incrementing the address, and putting a new data point on the bus. On completing the transfer the interface removes the bus aquistion signal and allows the processor to continue.

The time that the processor is not in operation amounts to one bus cycle, about one microsecond per data point.

The hardware required in the interface includes the data buffer and associated logic mentioned in the programmed I/O section above, plus two other registers that can also be loaded from the data bus. One of these registers will contain the address that will appear on the address bus as the destination of the data. This register must have associated with it the logic required to increment and decrement its contents, so that different addresses can be selected for different channels. The second register will be loaded with the numbers of the first and last channels to be sampled. The logic involved in the interface must look after incrementing the address and current channel after each DMA write, restoring the address after the last write of the group, and then signaling the processor with an interrupt. The complexity involved here in the order of a 50-100 MSI packages.

The I/O technique between programmed I/O and DMA in speed is the use of interrupts. The sequence of events in an interrupt in the Alpha CPU is as follows.

The peripheral generates an interrupt request when it has data to transfer. After the processor is finished the current instruction it requests a vector address from the peripheral and executes the instruction at this address in memory. This technique is called vectored interrupts because control is transferred to a different address for each peripheral.

The interrupt instruction is typically a service subroutine call. The processors registers are saved by the service routine, the required servicing is performed, the registers are restored, and control is returned to the main line routine.

If the interrupt instruction does not modify the program counter (ie is not a jump or call instruction) control is immediately returned to the main line. This is unique to the Alpha series of minicomputers. Special "automatic I/O" instructions take advantage of this anomaly to transfer data directly from the device to memory without disturbing the processor registers. This is the fourth type of I/O instuction available in the Alpha. Automatic I/O consumes five bus cycles per data word in a block transfer rather than the one necessary for DMA tranfers. The programming overhead is the same as for DMA, the speed is higher than that of interrupts, and the hardware is slightly more complicated than that required for vectored interrupts.

The fifth and last type of I/O operation is called block I/O. This is also unique to the Alpha series of computers. It is an extension of programmed I/O. The index register is loaded with a word count, and data is transferred through a pointer in the instruction from the device when the device ready flag goes high. This can be a very high speed transfer with data rates to 600k words per second. This is half the maximum data rate available under DMA.

All of the I/O methods but DMA were implemented in the analog interface. The transfer method chosen for the controller program was a combination of block I/O and interrupt. The service routine overhead for the input of eight channels of data is about 60 microseconds. In a typical controller cycle, with eight input services this amounts to nearly 500 microseconds. This amounts to five percent of a typical cycle. With a simple interrupt interface the fraction would have been closer to fifteen percent.

A slightly different technique, using the automatic I/O tranfer method, could reduce the overhead to 220 microseconds for two input channels.

A multiplicity of I/O methods were built into the analog interface in order that the best method to use could be chosen in the program. No one technique was demonstratably superior to all the others. As it turned out it is still not possible to choose a best method. Only after some field experience further defines the exact program requirements will it be possible to make such a choice, and simplify the analog interface. Fortunately not that much complication was introduced into the hardware by making it as flexible as it is. Consequently the decision to add the flexibility was the best choice.

It would seem likely that floating point hardware will be added to the system at some future time. When this happens the I/O overhead could become a significant factor in the minimum cycle time of the controller.

5 FIELD TESTS

5.1 Discussion of results

In the early part of March 1977 the controller system was transported to the Sundance thermal plant fifty miles west of Edmonton. The Sundance station is the largest of Calgary Power's generation installations. At present it consists of two 300 MVA (units 1 & 2) and two 380 MVA turbo electric generators fueled by coal mined on site. Units 5 and 6, presently under construction, will also be 380 MVA units. Unit 3 through 6 will have static exciters capable of changing field current, in accordance to voltage regulation signals, with step responses of about 150 milliseconds.

The primary part of the test involved the digital implementation of the existing "power system stabilizer" that feeds the output electrical power of the generator back into the voltage regulator and provides damping of local power oscillations. The voltage regulator was at one point receiving inputs from four sources, the reference input, and conditioned signals corresponding to the machine's terminal voltage, the machine's output power, and the power flow in the Alberta/B.C. tie-line.

Calgary power is at present experiencing some difficulty with oscillations in the power flow through the tie-line between their system and B. C. Hydro. It is hoped that a feedback system into the voltage regulators of the four 380 MVA Sundance generators can afford some stabilization of these oscillations. The natural frequency of the power variations on the tie-line are in the order of .2 hertz. The telemetry (a 2 hertz wide channel on a microwave link) to feed back the power flow, was in place at the time of the field tests. Implementation of a "tie-line stabilizer" to condition this signal for feedback into the voltage regulator of unit 3 was a part of the controller system test.

The transfer function of the existing analog PSS for local power oscillation suppression was :

$$\frac{O(s)}{I(s)} = \frac{4.5s\,(1+.12s)\,(1+.12s)}{(1+.21s)\,(1+.5s)\,(1+.5s)} \qquad 5.1$$

and the transfer function of the tie-line stabilizer was;

$$\frac{O(s)}{I(s)} = \frac{10s\,(s^2+2s+100)}{(1+.2s)\,(s^2+20s+100)} \qquad 5.2$$

Figure 5.1 shows a block diagram of unit 3, the relevant control loops and the input and output points for

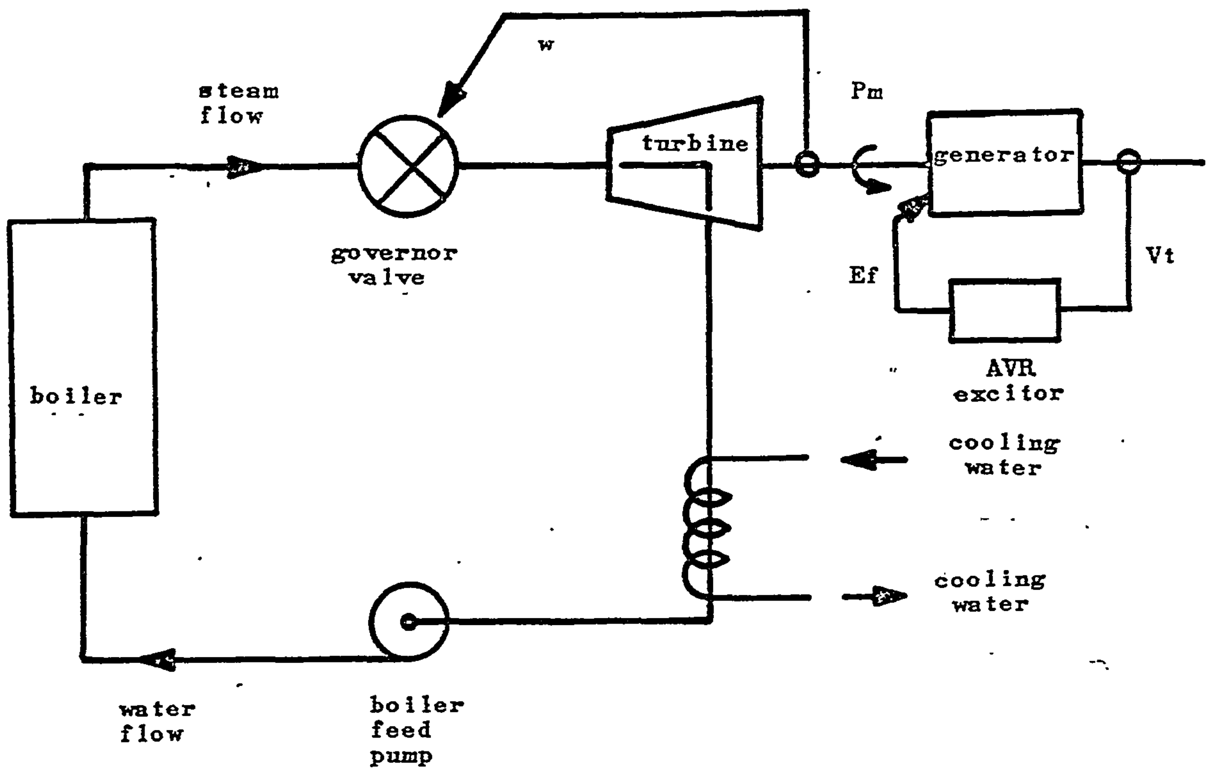


Figure 5.1 Block diagram of Sundance #3

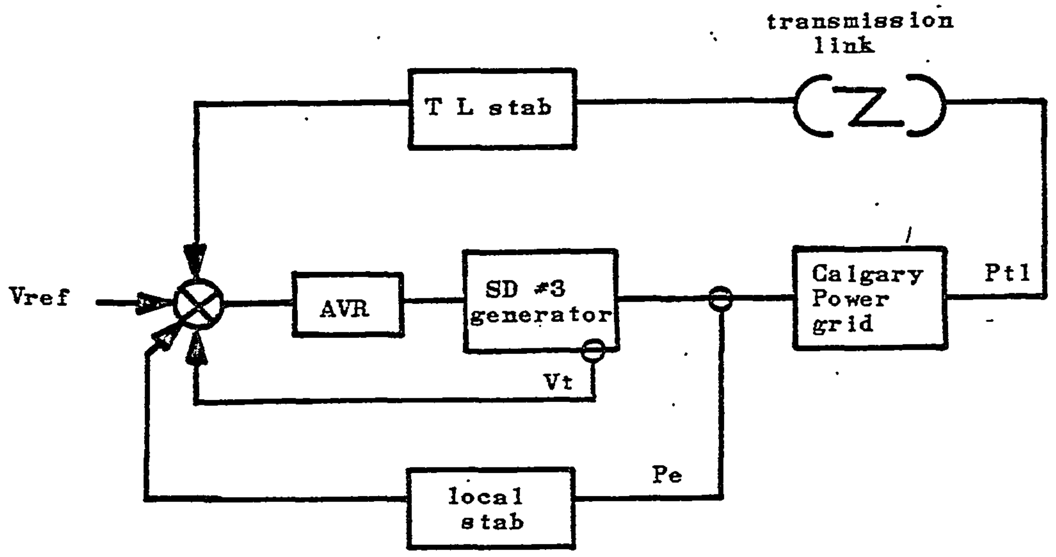


Figure 5.2 Stabilizer Inputs and Outputs

the digital controller. Figure 5.2 shows a diagram of the relevant blocks in the digital controller and its interfacing to the voltage regulator.

A series of strip chart recordings shows a comparison of the responses of the digital and analog power systems stabilizers. Figure 5.3 shows the outputs of the two controllers for a step disturbance on the reference input of the voltage regulator. These two responses are similar but not exact.

For this figure and 5.4 the integration technique used was the rectangular or Euler method (ref 14). The algorithm for this method is given by:

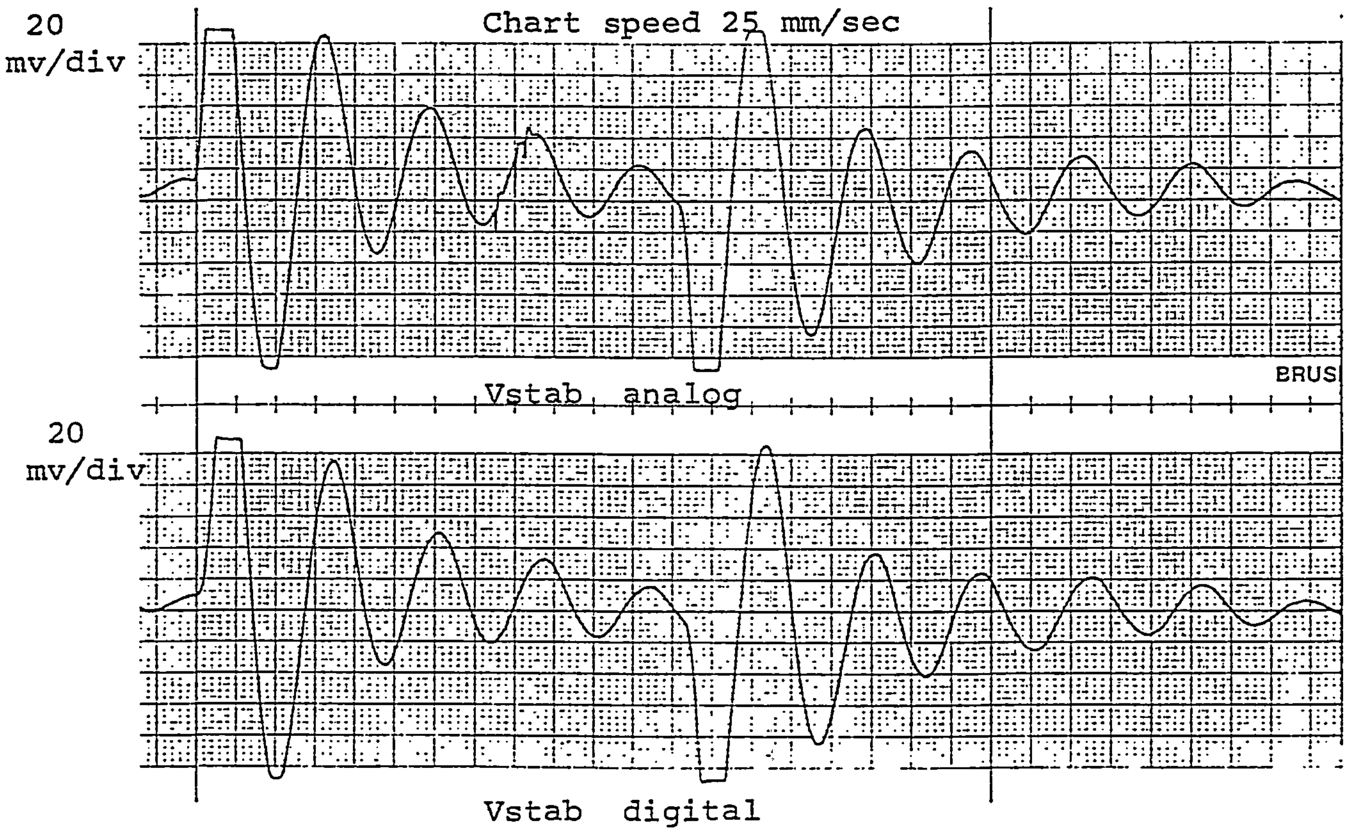
$$out(t+\Delta t)= out(t)+ in(t)\Delta t \qquad 5.3$$

where Δt was .01 seconds

Figure 5.4 compares the closed loop performance of the two controllers. Figure 5.4a shows the power transient resulting from a step input to the voltage regulator with the analog stabilizer in the loop. The analog stabilizer had the parameters shown in equation 5.1.

Figure 5.4b is the best power transient obtained with the digital controller by varying the parameters in a narrow range about those of the analog stabilizer. Again the

FIGURE 5.3 Transient response of analog and digital stabilizers (open loop) for a reference voltage step.



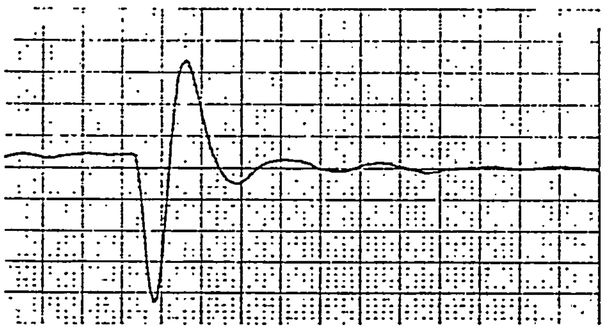
March 2, 1977

March 2, 1977

FIGURE 5.4a

Closed loop.

Analog stabilizer.

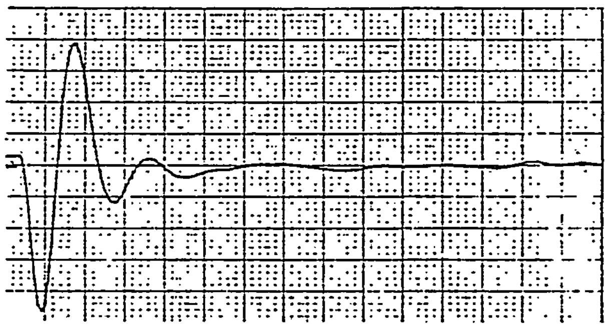


all traces are generator electric power

FIGURE 5.4b

Closed loop.

Digital stabilizer.



all traces 1Mw/div vertical 25 mm/sec horizontal

FIGURE 5.4c

Open loop.

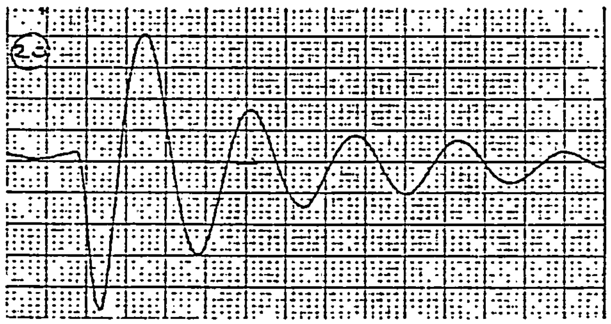


FIGURE 5.4 Power transient responses for step in reference voltage.

responses are not exactly the same and possible sources of the differences will be discussed later. Both responses are significant improvements over the open loop power transient figure 5.4c. The parameters of the digital stabilizer for figure 5.4b were:

$$\frac{O(s)}{I(s)} = \frac{2s\ (1+.2s)\ (1+.2s)}{(1+.21s)\ (1+.5s)\ (1+.5s)} \qquad 5.4$$

Figure 5.5 compares the two stabilizer outputs for the case of trapezoidal integration. Here both stabilizers again have the original parameters as shown in equation 5.1. These two traces are closer than those of figure 5.3. Substituting the parameters of equation 5.4 gave the closed loop performance shown in figure 5.6.

At this point, the tie-line stabilizer was added to the digital system and the two stabilizers run simultaneously. Traces showing the effect on the tie-line power can be seen as figure 5.7. A marginal improvement was found. More significant improvement was seen during tests conducted in July 1976 when unit 4 was not yet on line. Possibly future tests will be carried out when unit 4 is down, or when it too can be driven from the controller.

5.2 Functional Considerations

March 2, 1977

FIGURE 5.5

Open loop response of stabilizers for a reference voltage step. Same parameters for both stabilizers. Trapezoidal integration in the digital stabilizer.

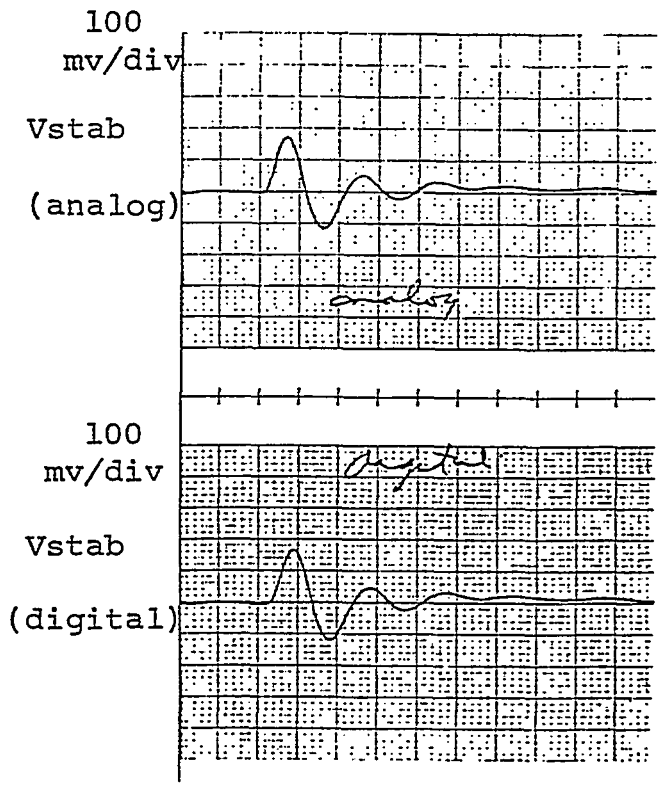


Chart speed 25 mm/sec

Chart speed 25 mm/sec

FIGURE 5.6

Closed loop response. Trapezoidal integration, digital parameters modified from above.

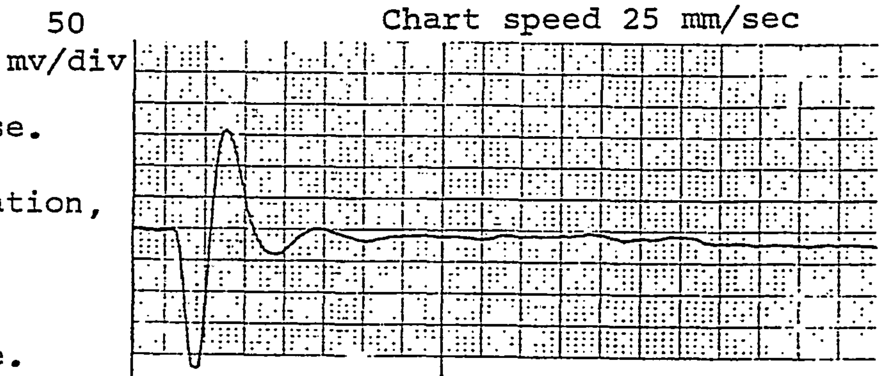


FIGURE 5.6a Vstab (analog)

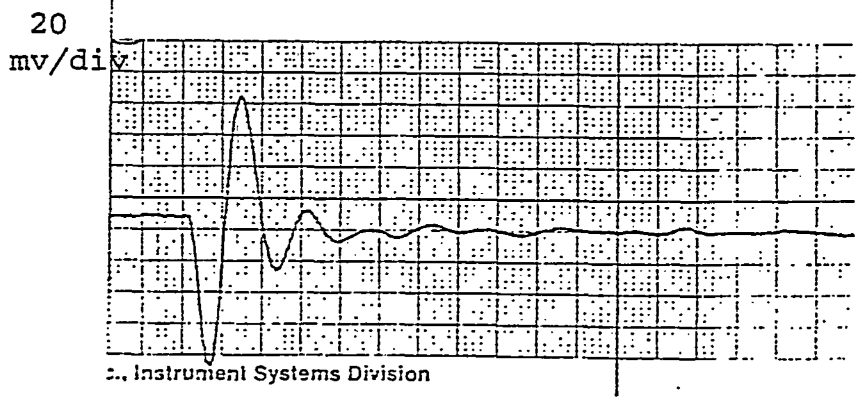


FIGURE 5.6b Vstab (digital)

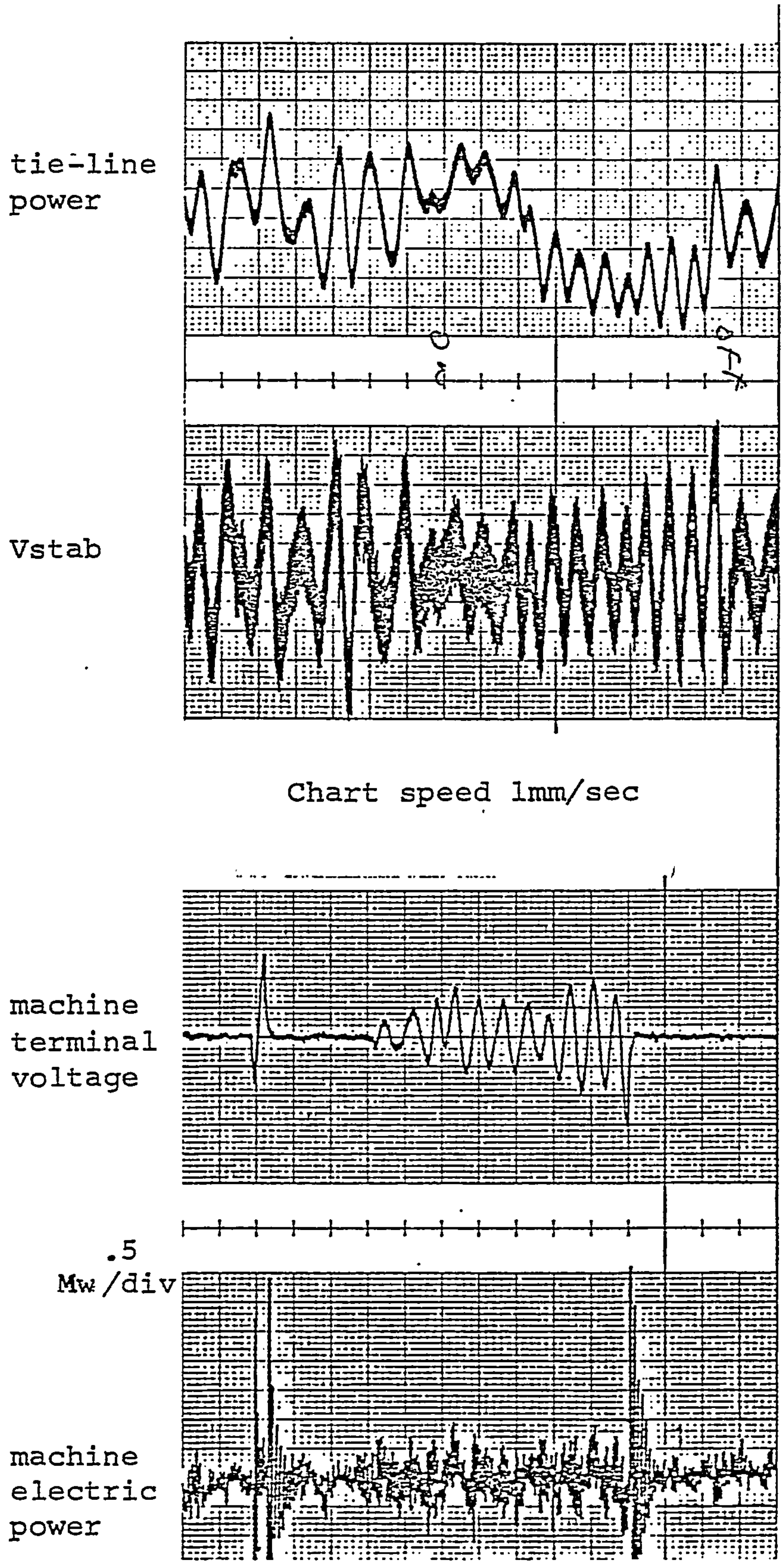


FIGURE 5.7 Performance of the digital stabilizer with both tie-line and local stabilizer functions being performed simultaneously.

It was felt that the performance of the digital system was adequate during these field tests. The output of the digital stabilizer was not exactly the same as that of the analog stabilizer. From figures 5.3 and 5.5 it can be seen that with the same parameters the digital output is about 5 percent smaller and 40 milliseconds behind the analog output under open loop conditions. These discrepancies required that the lead terms in the digital stabilizer be increased to give similar performance to the analog controller.

The discrepancies could arise from a number of different sources. A discussion by Uram (ref 13) suggests that a significant source of error could be inaccuracies due to too large a sampling interval in the integration routine. Uram suggests that a sampling time of .02 to .05 times the smallest time constant is necessary for one percent accuracy. If accuracies of 3 to 5 percent are tolerable, as they should be in this case, the factor might increase to .1 or greater. In the case of figure 5.3 and 5.5 this factor was .083. The effect of a number of different sampling rates on the step response of a digital simulation of this digital stabilizer is shown in figure 5.8. From these graphs it would appear that cycle times up to 30 milliseconds might be useable with trapezoidal rule integration.

From this it would seem that little of the observed

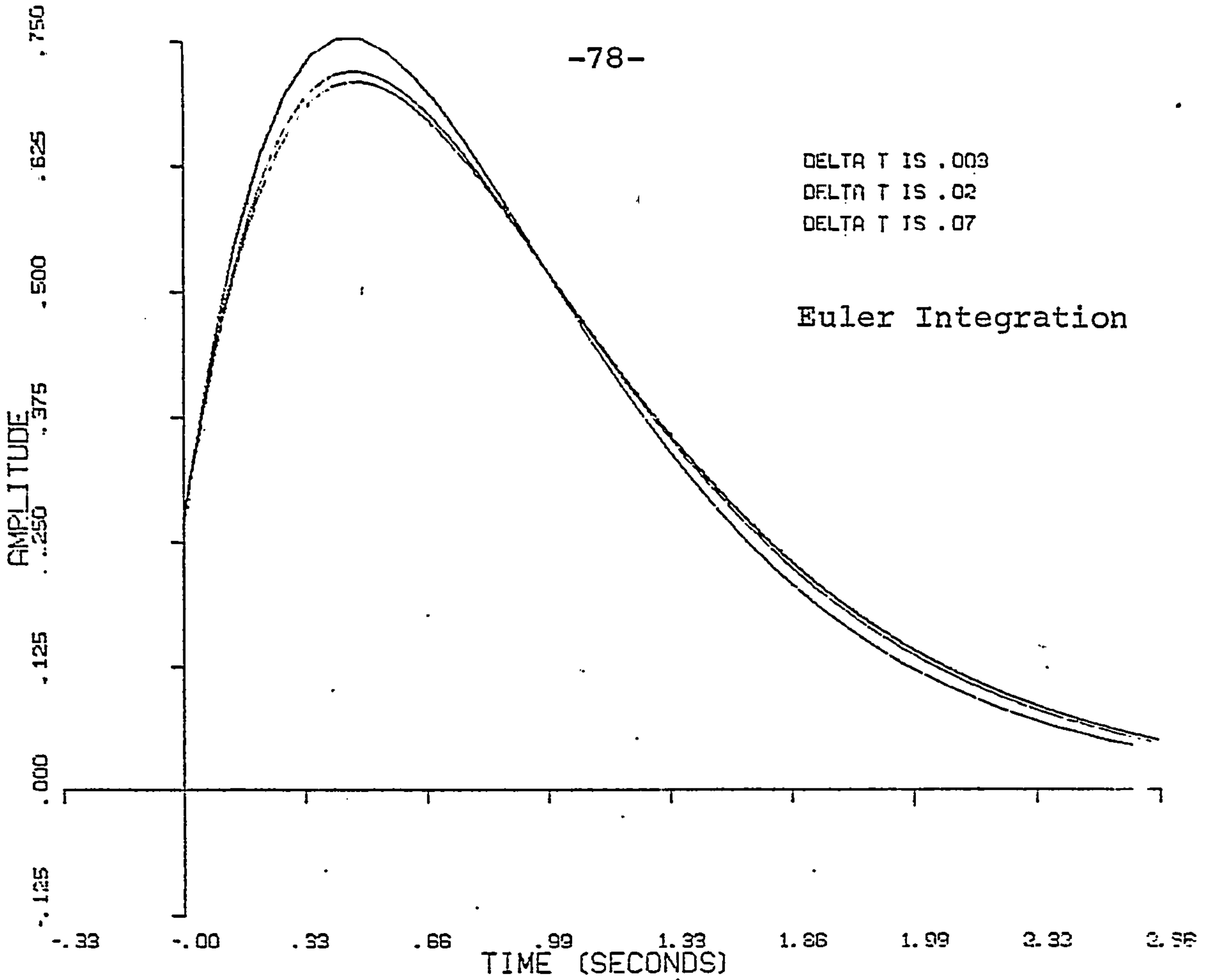
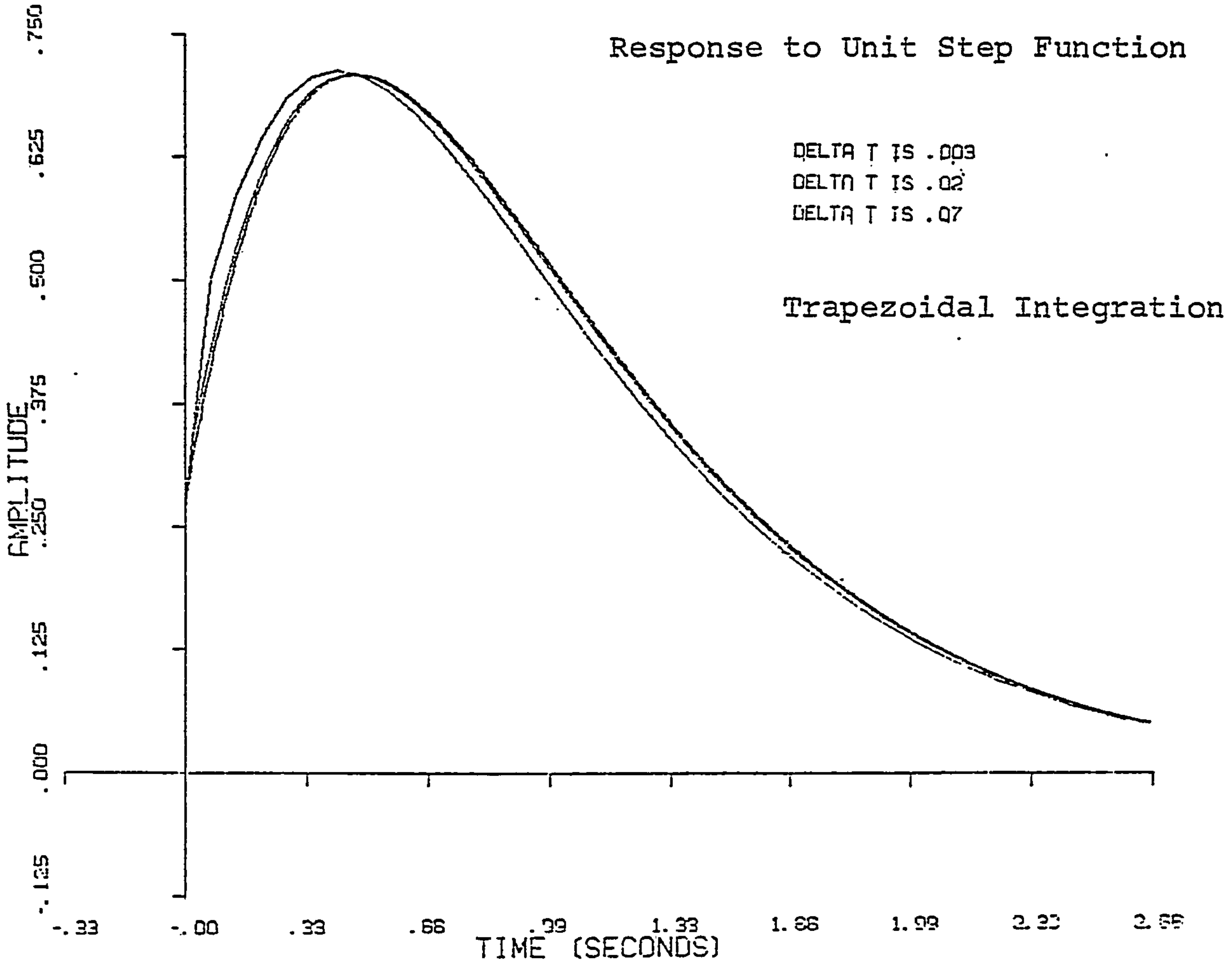


Figure 5.8 Computer Simulation of Digital Stabilizer

Response to Unit Step Function



discrepancy is attributable to integration inaccuracies.

The five percent difference in gain is in fact not a serious concern. It is within the tolerence of the components used, most notably the output voltage divider that was used to give adequate resolution. These resistors were ordinary components with standard 10 percent tolerance. This gain error could in any case be corrected by changing a single parameter in the digital system.

The 40 millisecond delay observed is more serious. Some modification of time constants was required to correct for it. The delay is of course in the input and output analog filters. These are 16 hertz low pass filter with two poles on the real axis at -100 radians. Each filter has a group delay (ref 12) that was measured at 18 to 20 milliseconds. Some calculation showed that the delay is nearly constant over the pass band of the PSS with a theoretical value of 20 milliseconds.

The input filter is to prevent aliasing. The passband of this filter could conceivably be doubled. The output filter was intended only to smooth the output. It could have a cut off frequency a factor of twenty higher. These two modifications would reduce the observed delay by a factor of four.

In the A/D service routine the actual sampling is done at 800 hertz. Every eight points are averaged together to give one input point to the controller routine. It would be possible then, and probably a good idea, to back the analog input filter off to one or two hundred hertz and implement the aliasing filter digitally. This could enable the cut off frequency of the filter to be linked to the cycle time of the program. Thus the highest possible cut off frequency and the lowest possible group delay could be had under all circumstances. The Nyquist frequency for the input to the controller is 1/(2*cycle time). The group delay of the required aliasing filter can be anticipated to be at least the cycle time of the routine.

This necessary delay may well be the most significant difference between the analog and digital implementations. As was mentioned previously it is possible to live with 40 milliseconds of delay in this particular application. The delay causes a discrepancy between the analog and digital stabilizers, but both devices give a significant improvement in transient response. The performance of the digital device is then quite adequate.

5.3 System Considerations

A number of problems with the DDC system as it is now surfaced while field tesing it.

The most significant operator difficulty encountered was that controller parameters could not be entered and examined in decimal notation. This facility had been planned for but not yet implemented at the time of the test. It became apparent that it is very important to know exactly what controller and what parameters are currently active. It is difficult to enter and read constants through interpretation by a printed table. The problem is greatly exaggerated by the "hostile to humans" environment. Day-long exposure to 100 dba noise levels is detrimental to operator efficiency. It was found that on the second day of such exposure this operator was much more prone to make mistakes. Unnecessary complication of the human interface to the system was definately unappreciated under these conditions. This effect must be allowed for by making the human interface section of the program as simple to understand as possible.

In particular parameters should be entered and listed in a form similar to that in which they appear in a transfer function. For example entering the parameters K, T1 and T2 in decimal is preferrable to entering K*T1/T2 and 1/T2 in a binary representation of a floating point format as was the case at Sundance during the field tests. A second important point is that the parameters be easily accessable and labeled with meaningful mnemonics. For example typing the command "List C 1" and getting an output such as:

```
Controller 1 consists of:

A washout term with the parameters:
        K=4.5
        Td=.21

A lead lag term with the parameters:
        K=1.0
        T1=.12
        T2=.5
```

can be seen to be a significant advantage. This type of output is quite possible and very desirable.

Another problem encountered was the difficulty of changing from Euler or rectangular rule integration to trapezoidal rule integration. At the time of the field test each controller segment had a "second half" routine that was invoked after the D/A service to perform the required integration. This was (and is) the fastest, in terms of execution time, way of doing this. More general routines typically run slower. In the case of the present digital controller there was a significant penalty for lack of generality in that these routines had to be completely rewritten in the field to change the method of integration. This structure has now been changed so that a general purpose routine to perform all the required integrations is a part of the D/A service routine. This results in a slight time penalty but use of the same code for all the integrations makes the integration method easy to change. In the current implementation the integration method is a specification made in the global data block.

At the time this modification was carried out a preliminary investigation was carried out with regard to higher order methods of integration. In particular the fourth order Milne method predictor/corrector (with one step of corrector, ref 14) was used. This method is very attractive because most of the multiplicative constants used are powers of two, and so lend themselves to a very fast multiplication technique that can be used with floating point numbers. In fact the method required less than twice the time of the trapezoidal rule to give a result with a nominally fifth order error term. Unfortunately significant mathematical stability problems were encountered and the method abandoned.

It is felt that further investigation into higher order integration techniques is warranted, but that this is beyond this scope of this project.

Some plots of the step response of the Sundance controller algorithm with the different integration techniques used in the field test and representative integration times are shown in figure 5.8 as previously mentioned. In these plots the time constants and gains are as those for the installed analog stabilizer shown in equation 5.1.

These were the two major system related difficulties encounterd at Sundance. As may be expected a host of minor problems were also encountered, a few of which are listed here.

The D/A routine allowed wrap around on too small or large an output value. This was quite a dangerous oversight but easily corrected with a limiting routine.

The resolution of the 8 bit D/A turned out to be insufficient. The temporary fix was to boost up the gain through the controller and then reduce it with a resistive divider after the D/A. The difficulty arose from the "arbitrary" requirement that the output swing from plus ten volts to minus ten volts. This gave a resolution of 78 millivolts. This was thought to be adequate before the test but on hook up it was found that under some circumstances the maximum output excursion was only several hundred millivolts. The

8-bit D/A were chosen for a number of reasons including significant hardware simplification and the cost of a large number of channels. A permanent fix to this problem will take the form of generating the reference voltage (which specifies the positive and negative limit of the output) for the output D/A from a second D/A. Consequently the output will be specified by one D/A and a limiting function performed by the second. There are advantages and disadvantages of this approach as compared to using a D/A of sufficient precision to give both adequate range and precision under all circumstances. The fix specified will function as well as the second solution and yet retain the hardware advantages of the original design.

A third minor problem which should be corrected was that the D/A outputs were all referenced to a single ground. This also was a product of the time constraints imposed by the field test, as isolation was designed into the system but not yet implemented. The problem was solved by using external isolation amplifiers previously constructed by Calgary Power. In a power station there are typically many different grounds. For two signals to have the same ground is the exception rather than the rule. Both inputs and outputs must be provided with isolation from the computer ground.

This is a relatively complete list of the difficulties encountered during the field test. This is not, of course, to say that with these modifications the system will be perfect. It is to be expected that the problems thus far recognized no doubt masked a good many others.

6 CONCLUSIONS

6.1 Project Summary

This thesis describes a digital controller development system that was constructed around an especially purchased commercial minicomputer. The system is intended to be a relatively general purpose power systems research tool. The thesis dealt with the particular application of emulation of the existing analog power systems stabilizer in the excitation system of Calgary Power's Sundance #3 turbo generator. A field test was conducted wherein the digital processors performance was compared directly to that of the analog stabilizer, both under open and closed loop conditions. A second part of the test consisted of emulating the PSS with the digital processor in the closed loop, simultaneous with the implementation of a tie-line stabilizer of similar complexity. The tie-line stabilizer for Sundance #3 had not yet been installed at the time of the test.

The digital controller performed satisfactorally during the tests.

6.2 Recomendations For Further Work

The device would benefit from some further work.

Most notably the controller program lacks an adequate monitor module for the interface of the operator and the controller program. Such a routine should ask questions regarding what kind of controller and what parameters are desired, while providing an easy to use facility for obtaining information on the status and parameters of operational controllers. The present monitor routine performs these functions, but requires that the operator have extensive knowledge of the controller program.

A second significant programming area that needs work is the inclusion of a set of limiting and alarm routines. The limits should be acèssible through the monitor. The programming of the existing controller routine has been done in such a way that it is possible to put a limit on the output of each integrator. It would be desirable to implement this function. Also a set of modes should be added such that each controller can be in only one of the two possible modes; pot set and run. The modes named would have similar meanings to those assigned in analog computer jargon. Each controller should be individually settable to only one of the modes, as well as being resetable to intial conditions.

A third programming task that must be done requires the adaptation of the standard computer diagnostic program to run with the controller system. Much of the program can probably be excised as irrelavent, but some things will have to be added. Diagnostics of the real time clock, the A/D's and the D/A's that will run simultaneously with the controller system must be written. The A/D's could also be well tested by measuring the voltage output by each of the D/A channels with a spare A/D channel and comparing the result with the programmed output value.

The system could also benefit from some hardware improvements.

The most significant improvement would be the development of a "hardened" memory module that more exactly suited the requirements of the application than any available modules. Neither core nor semiconductor memory is adequately nonvolatile. Semicondutor memory requires a battery backup for limited nonvolatility. Core memory can be disturbed by power transients. The contents of both types of memory can be modified by runaway programs caused by a bug in the code. The standard fix is to add some kind of magnetic mass storage to the system and reload the program under these circumstances.

Recent improvements in the technology now offer a better (and cheaper) alternative. Intel has developed a 5 volt UV eraseable ROM. The chip contains 2k by 8 bits of storage, is individual byte programmable, and requires only one supply plus the programming supply. Combining this type of memory (EPROM) with memory made up of the new 4k static RAMs and a small fuse link memory would allow an 8 or 12k word memory module to be built on a single half board with the desired "hardness".

The lower 16 words of the memory would be a bootstrap routine, in fuse link PROM, at the power up interrupt vector location. The rest of the lower 4k words would be EPROM. The EPROM would be normally write protected by a program settable latch. Setting the latch would allow programming (ie writing into) the memory. The upper 4k words on the half board would be memory constructed of 4k static RAMs.

On power up the EPROM would be copied into the RAM by the bootstrap routine and control transferred to the program in the RAM. If the EPROM was empty (ie had just be UV erased) the autoload routine could be used to enter a program into RAM. This program could be run until controller forms and parameters were developed, and then it could copy itself into the EPROM memory. Interruptions in power after this point would result in the controller being completely reset and restarted with parameters previously saved, when power was restored. Since the Intel EPROM can be programmed a word at a time, it would also be possible to change the controller parameters and forms without erasing the EPROMs.

A second improvement that would be desirable would be the shrinking of the analog interface to half board size. This would allow the entire system to run using the microprocessor LSI 3/05 version of the Alpha CPU. Improvements in the density of the necessary functions such as the development of programable interval timers and four bit bus tranceivers would allow the necessary shrinkage.

A third possibility is the addition of a floating point processor. Plessy Systems sells a bit-slice based microprocessor that could profitably be used as such a floating point processor. It is doubtful that it would be wise to convert the system base from the Alpha to the Miproc processor because of the lack of sofware support for the latter. However, it would not require a tremendous amount of work to put the floating point routines into the microprocessor with a resulting speed improvement of more than five times.

Off loading some of the intelligence of the analog interface into the microprocessor might also increase throughput.

All these additions will involve a considerable amount of work. The system is useable without them, however, and they could be implemented gradually.

LIST OF REFERENCES

1. Schlief F.R., Hunkins H.D., Martin G.E., Hatton E.A, "Excitation Control to Improve Power-Line Stability", IEEE Transactions, Vol-PAS-87, No. 6, pp 1426-1434, June 1968.

2. Warchol E.J., Schleif F.R., Gish W.D., Church J.R., "Alignment and Modelling of Hanford Excitation for System Damping", IEEE Transactions, Vol-PAS-90, pp 714-724, March/April 1971.

3. Bollinger K.E., Laha A.K., Hamilton R., Harris T., "Power Stabilizer Design Using Root Locus Methods", IEEE Transactions, VOL-PAS-94, pp 1484-1488, Sept/Oct 1975.

4. Bollinger K.E., Winsor R.A., Laha A.K., Gilchrist R., "On-Line Identification of Power System Transfer Functions", presented at the CEA meeting March 17-19, 1975 and published in the CEA Transactions.

5. Laha A.K, Bollinger K.E., "Power System Stabilizer Design Using Pole Placement Techniques on Approximate Power System Models", IEE Proceedings, September 1975, pp 903-907, Vol 102, No. 9.

6. Bollinger K.E., Laha A.K., "Designing and Field

Testing Power Stabilizers", Presented at the CEA Meeting October 23, 1974, and published in the CEA Transactions.

7. Hughes F.M., Hamdan A.M., "Design of Turbo Alternator Excitation Controllers Using Multi-Variable Frequency Response Methods", IEE Proceedings, Vol 123, No. 9, September 1976, pp 901-906.

8. Winsor R.A., Bollinger K.E., "Sundance Plant Dynamic Stability Improvement Utilizing a Power System Stabilizer Tuned by the Root Locus Method", accepted for presentation at IEEE Summmer Power Meeting, Mexico City. Abstract published.

9. Bollinger K.E., Laha A.K., Winsor R.A., "System Models from Transient Stability Programs", Presented at IEEE PICA Conference, New Orleans, June 2, 1975 and published in the IEEE PICA Proceedings TP XII-6, pp 330-334.

10. Bollinger K.E., Winsor R.A., "A Survey of Power System Stabilizer Tuning Techniques", Presented at the CEA conference, Toronto, March 1976, to be published in the CEA Transactions.

11. Keay F.W., South W.H., "Design of a Power System Stabilizer Sensing Frequency Deviation", IEEE Transactions, Vol PAS-90, pp 705-713, March/April 1971.

12. Rabiner L.R., Gold B., Theory and Applications of Digital Signal Processing, Prentice Hall, Englewood Cliffs N.J., 1975

13. Uram R., "Choosing Numerical Methods for Direct Digital Control Algorithms", Control Engineering Journal, May 1970, pp 94-97

14. Carnahan B., Luther H.A., Wilkes J.O., "Applied Numerical Methods", John Wiley and Sons, New York, 1969

15. Naked Mini LSI/ALPHA LSI Software Manual, Software order number 20025-00A0 from Computer Automation Inc., Irvine California, 1973.

16. Naked Mini LSI Series Computer Handbook, Software part number 91-20400-00A2 from Computer Automation Inc., Irvine California, 1974.

17. ALPHA 16/LSI Cross-Assembler (XASM3) Installation and User's Guide, for Software part number 94001-00B0 cross-assembler by Computer Automation Inc., Irvine California, 1974.

8.1 APPENDIX A; THE CORE OPERATING SYSTEM

I. Loading Procedure

1. Insert the system tape in the left drive of the 2644.
2. Sense switch on, console sense register set to zero.
3. Reset and autoload.
4. Hit the Esc e switch on underside of the table top to the right of the front door of the computer enclosure.

When the load is finished the moniter (Debug2) will be automatically started.

II. Modifications

The operating system consists of an interelation of standard CAI software some new I/O drivers, and some extra functions.

Software in the system:

1. Omega (not Omega2 because Omega2 uses some byte mode addressing) starts at :100
2. Lambda (not Lambda2, Lambda is shorter, fully

documented and adequate) starts at :35A0

3. BDP/VER (again the old version for the same reasons as for Lambda) starts at :37D0
4. Debug2 (the new version of DBG has some useful features) starts at :39C0

III. Modifications to Omega:

1. I/O drivers overlayed

   I4 is now HP right tape instead of card reader. L2 is now HP right tape instead of line printer. O2 is now HP left tape instead of DIO HSP. O3 is now device 2 instead of 6 for compatability with Doug's HSP
2. Commands overlayed
   - B command is a break or return to the monitor
   - T command is now a translate command to read out of the text buffer hex characters and translate them 2 for 1 to binary. This is for output from the cross assembler on the 370.
3. New Software
   - Core load routine uses Omega to assemble the program currently in the text buffer, and load the resulting binary into a buffer

from :3001 to :3500, swap this buffer for the lower :4FF words of Omega and then return to Debug.

NOTE: programs must be shorter than :500 locations and all ABS statements must be replaced by REL statements with the same arguements. Also labels in DATA statements will be :3001 more than they should be.

- Restore Omega routine swaps the lower portion of Omega back from the save buffer, if necessary, and transfers control to Omega, presumably with the text buffer intact.
- Swap routine exchange the save buffer (:3001 t0 :3500) word for word with the area from 0 to :4FF. This is the routine that actually performs the swaps mentioned above.

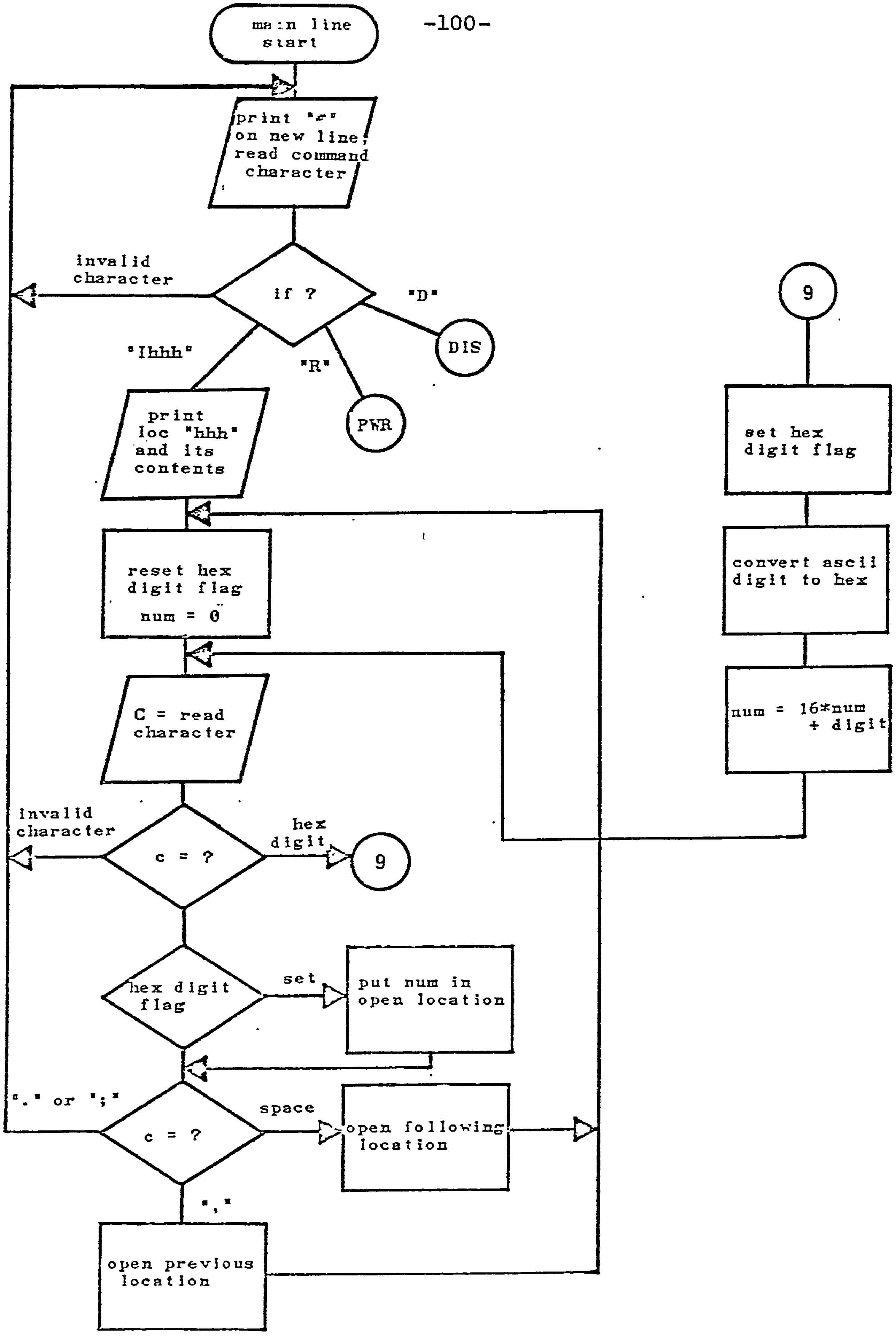
4. Standard locations now pointed to by relocation registers
   - RM=:35A0 start of Lambda
   - RD=:37D0 start of BDP/VER
   - RL=:3920 start of Core Load
   - RS=:3982 start of Swap
   - RO=:3917 start of Restore Omega

each of these routine can be executed by putting a J in front to make it a Debug command (IE JRL. would invoke the core load routine the core load routine from Debug)

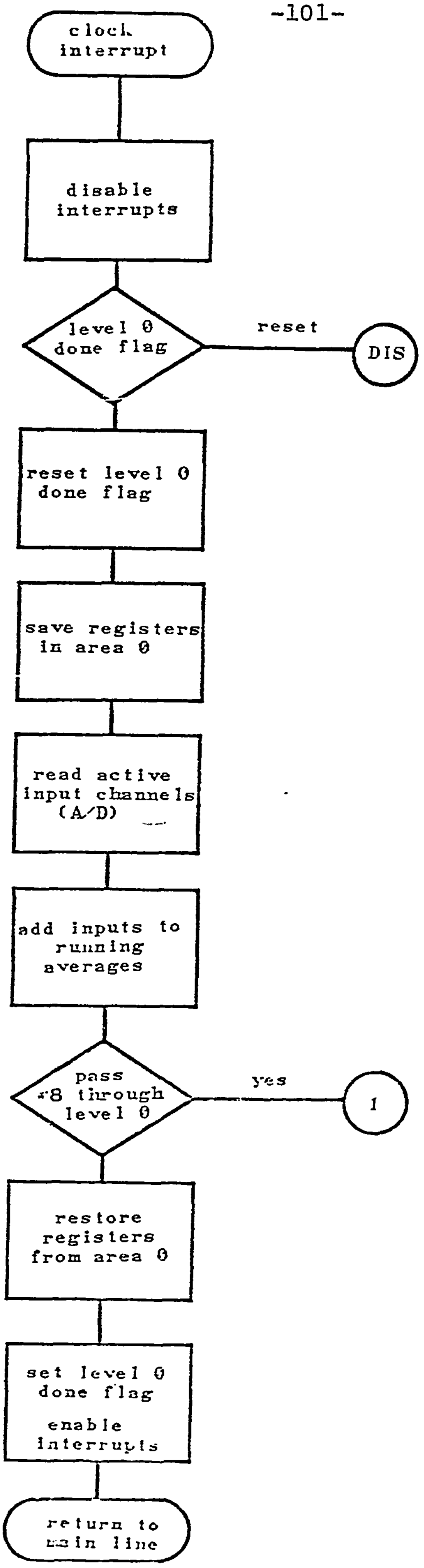
III. Creation from paper tapes

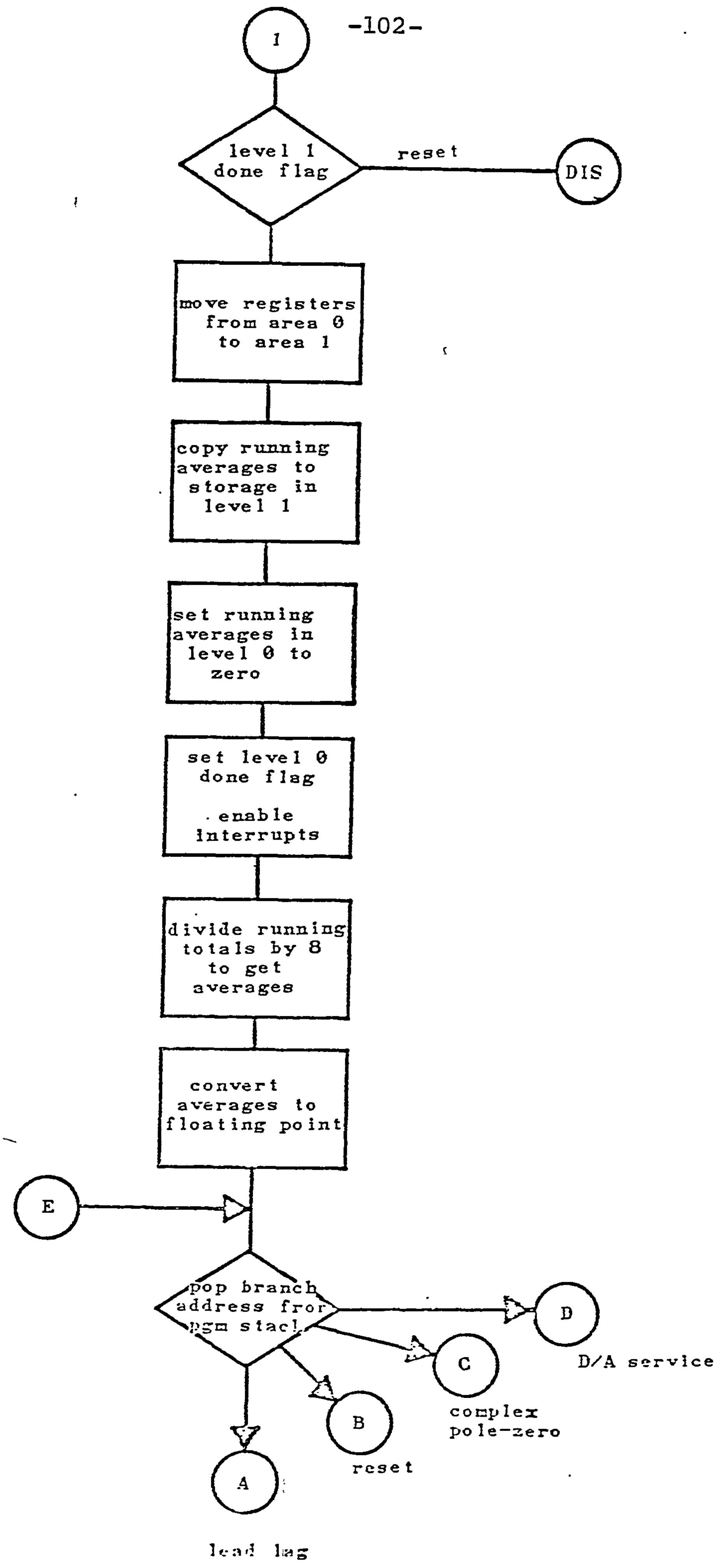
1. Load Debug2 at :39C0, BDP/VER at :37D0, and Lambda at :35A0
2. Load BDP/VER overlay. This is now the 16k monitor.
3. Load the Core Load overlay. This moved the BDP buffer and driver linkage to point at the driver and buffer in Omega.
4. Load Omega.
5. Load the HP Driver overlay into Omega. This is now the operating system.

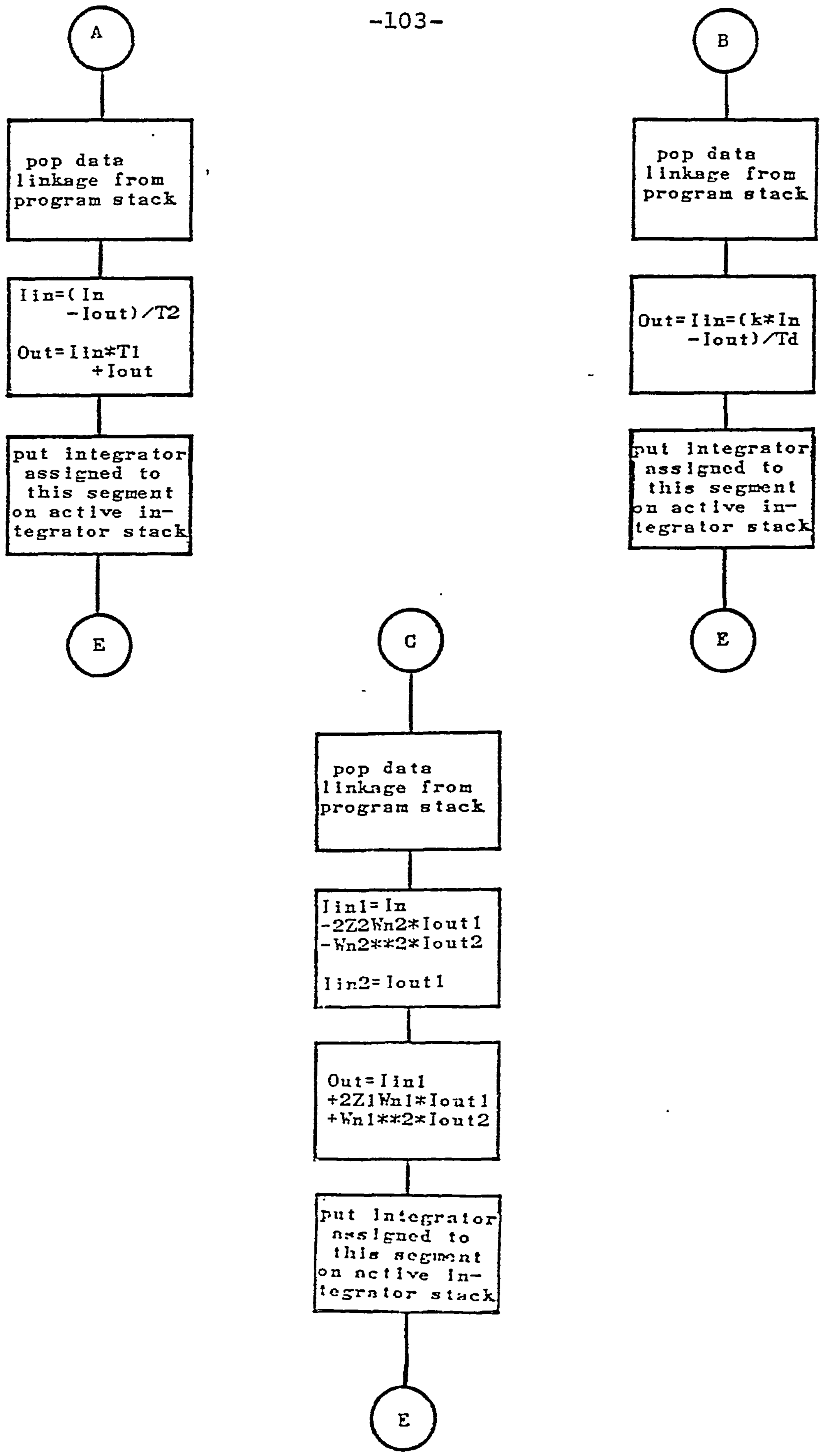
APPENDIX B; CONTROLLER SOFTWARE SUMMARY

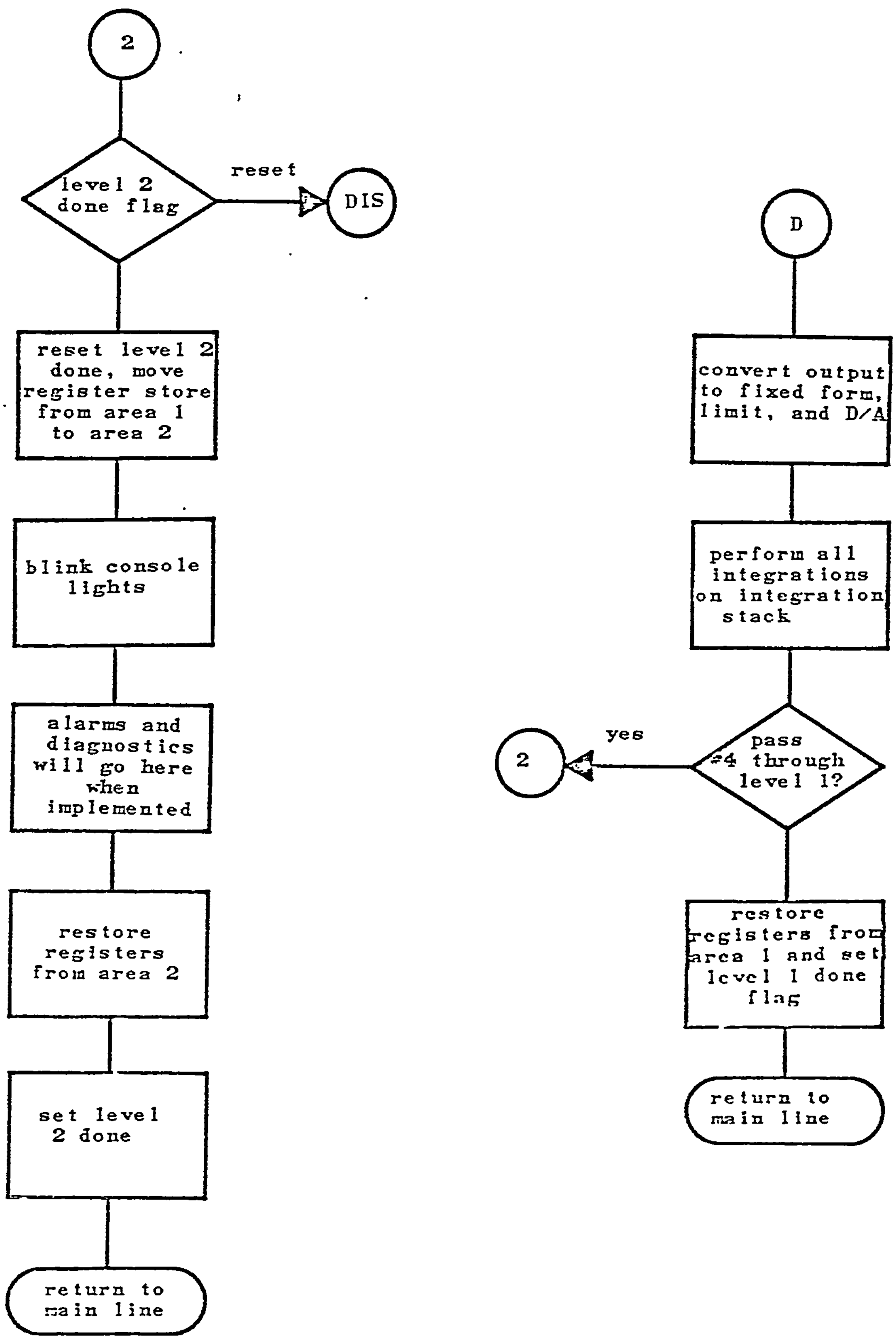


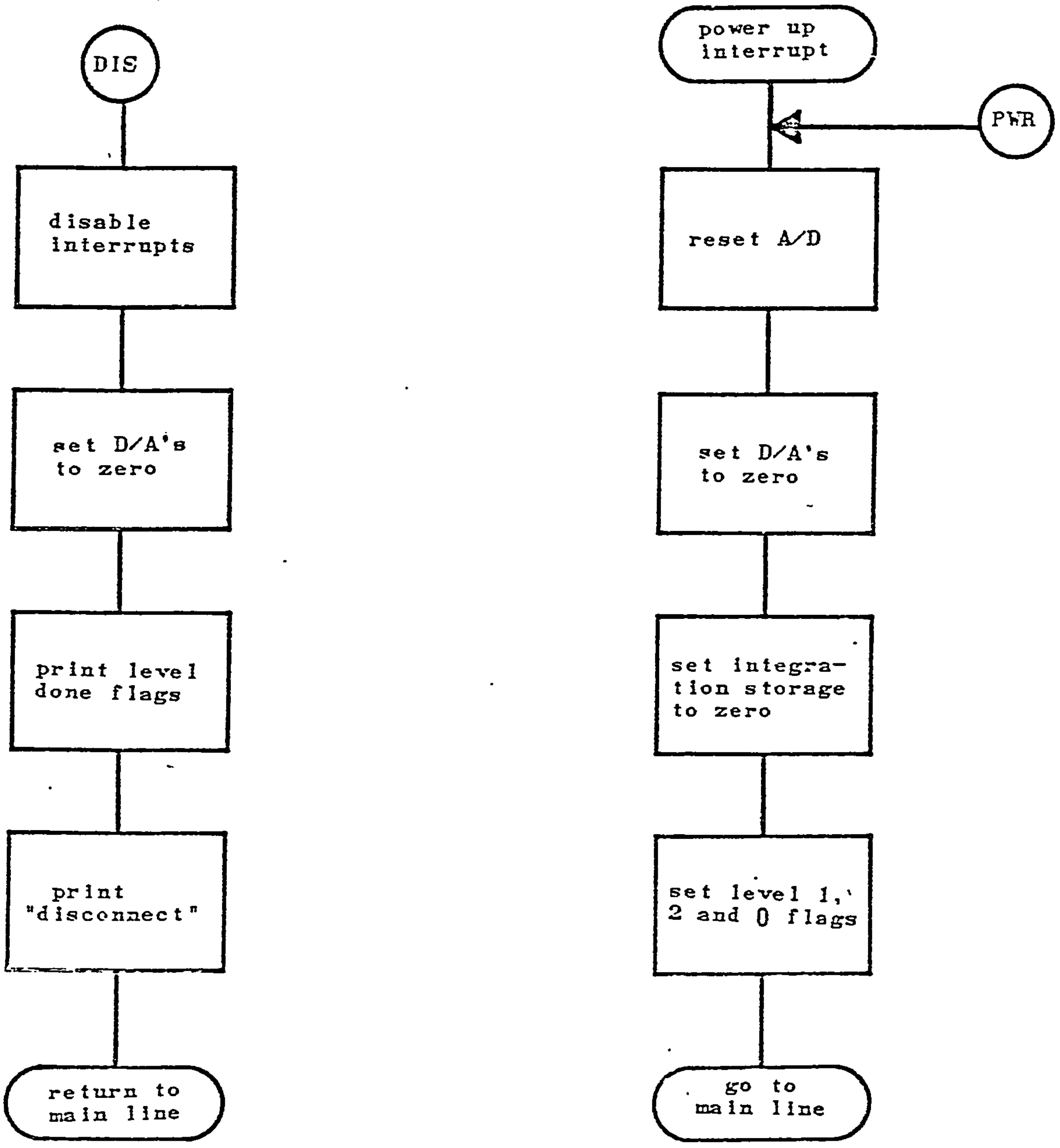
main line start
print "≠" on new line; read command character
if ?
invalid character
"D"
DIS
"R"
PWR
"Ihhh"
print loc "hhh" and its contents
reset hex digit flag
num = 0
C = read character
c = ?
invalid character
hex digit
9
hex digit flag
set
put num in open location
c = ?
"." or ";"
space
open following location
","
open previous location
9
set hex digit flag
convert ascii digit to hex
num = 16*num + digit


clock interrupt
disable interrupts
level 0 done flag
reset
DIS
reset level 0 done flag
save registers in area 0
read active input channels (A/D)
add inputs to running averages
pass #8 through level 0
yes
1
restore registers from area 0
set level 0 done flag
enable interrupts
return to main line


I
level 1 done flag
reset
DIS
move registers from area 0 to area 1
copy running averages to storage in level 1
set running averages in level 0 to zero
set level 0 done flag
enable interrupts
divide running totals by 8 to get averages
convert averages to floating point
E
pop branch address from pgm stack
D
D/A service
C
complex pole-zero
B
reset
A
lead lag


A
pop data linkage from program stack
Iin=(In -Iout)/T2
Out=Iin*T1 +Iout
put integrator assigned to this segment on active integrator stack
E
B
pop data linkage from program stack
Out=Iin=(k*In -Iout)/Td
put integrator assigned to this segment on active integrator stack
E
C
pop data linkage from program stack
Iin1=In -2Z2Wn2*Iout1 -Wn2**2*Iout2
Iin2=Iout1
Out=Iin1 +2Z1Wn1*Iout1 +Wn1**2*Iout2
put integrator assigned to this segment on active integrator stack
E


2
level 2 done flag
reset
DIS
reset level 2 done, move register store from area 1 to area 2
blink console lights
alarms and diagnostics will go here when implemented
restore registers from area 2
set level 2 done
return to main line
D
convert output to fixed form, limit, and D/A
perform all integrations on integration stack
pass #4 through level 1?
yes
2
restore registers from area 1 and set level 1 done flag
return to main line


DIS
disable interrupts
set D/A's to zero
print level done flags
print "disconnect"
return to main line
power up interrupt
PWR
reset A/D
set D/A's to zero
set integration storage to zero
set level 1, 2 and 0 flags
go to main line

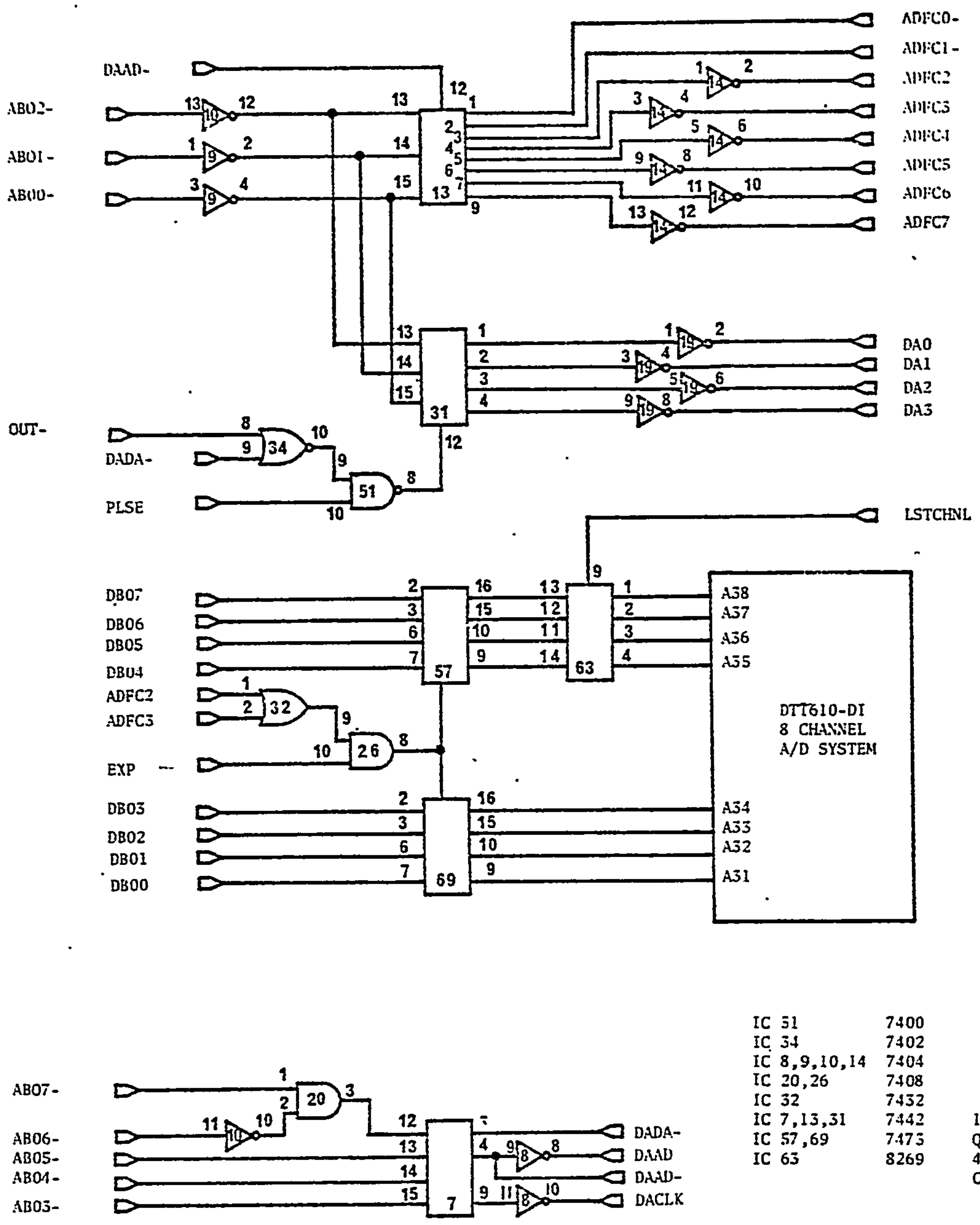
APPENDIC C; DESCRIPTION OF TH A/D AND D/A INTERFACE

Commands:

SEA 12,0	- Copy A register to on board clock latch. This value is stobed into a counter that is decremented every 16 microseconds. Underflow will trigger the A/D when enabled.
SEN 12,1	- Function code 1 reads the state of the data present latch onto the SER-line of the CPU. Data is present after all the channels specified have been sampled, and not all of them have been read out of the push down stack.
SEA 12,2	- Trigger the A/D. Sample channels a to b where the A register contains :00ba, and b>a.
SEA 12,3	- Trigger the A/D and allow it to be triggered by whichever of the following conditions are enabled: 1. An echo signal on the bus. (ie a RTC echo) 2. An underflow from the on board clock.

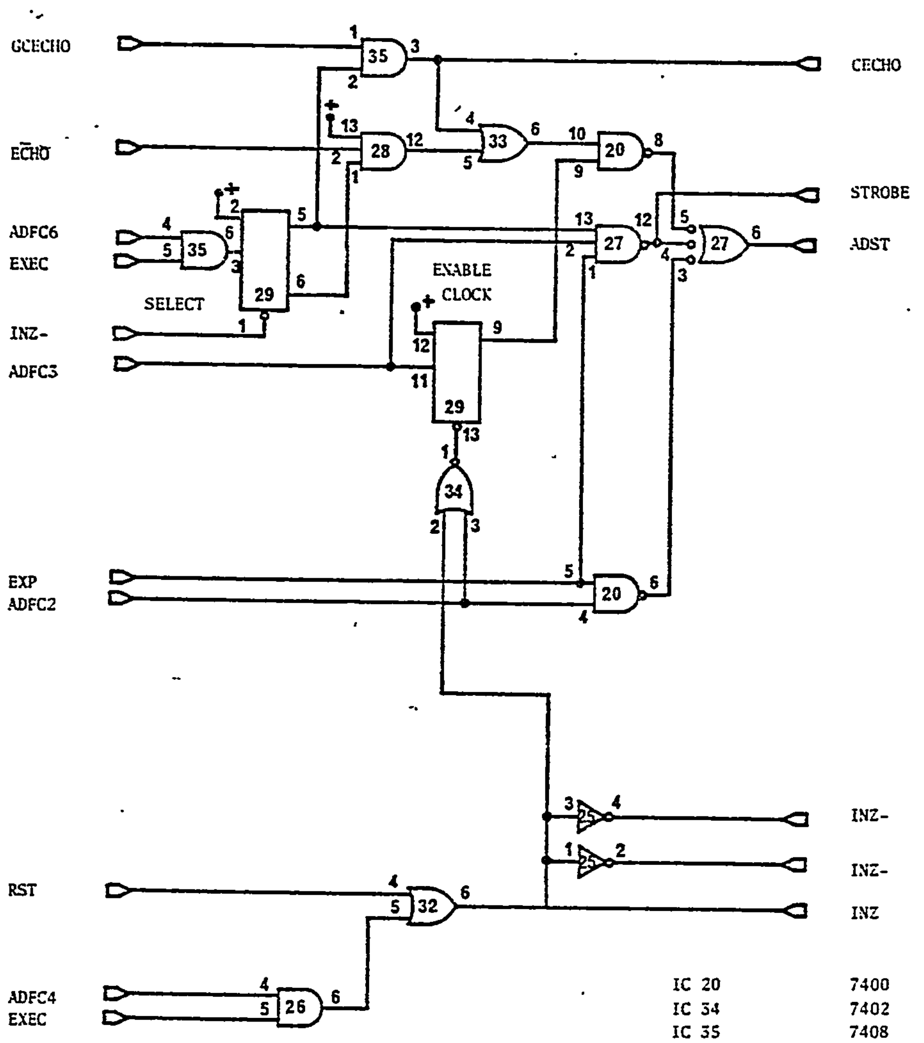
The channels to be sampled at each trigger are specified in the A register as in the case of function code 2.

SEL 12,4	- Initialize the A/D interface
SEL 12,5	- Enable A/D done to cause interrupt to location :72
SEA 12,6	- If A register is 0 enable A/D to be triggered from on board clock underflow. If A register is 1 enable on board clock underflow to cause an interrupt to location :76, as well as triggering the A/D
SEL 12,7	- Disable interrupts
SEA 13,0	- Value in A register is strobed into latch on D/A channel 0.
SEA 13,1	- Value in A register is strobed into latch on D/A channel 1.
SEA 13,2	- Value in A register is strobed into latch on D/A channel 2.
SEA 13,3	- Value in A register is strobed into latch on D/A channel 3.



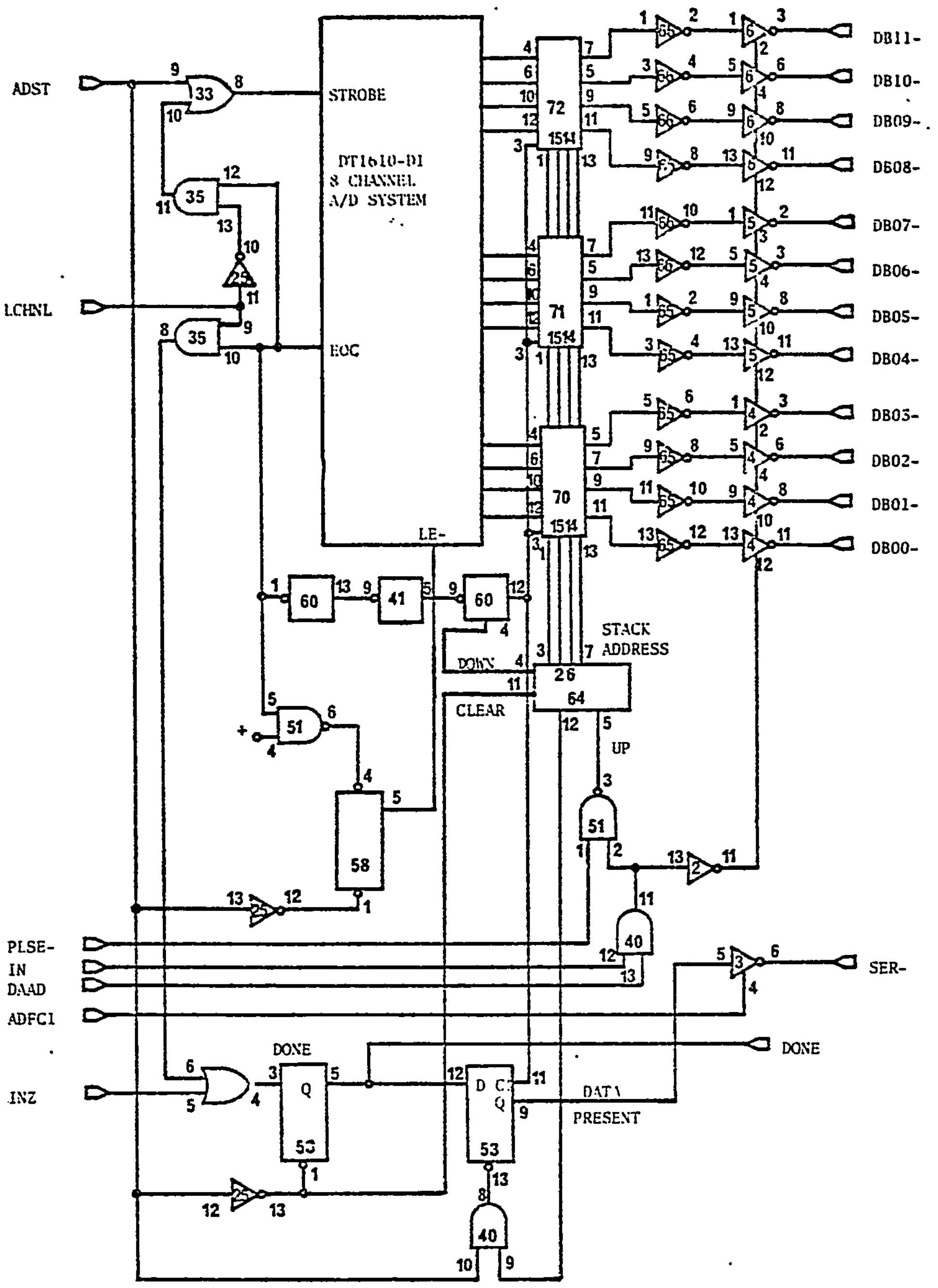
IC 51	7400	
IC 34	7402	
IC 8,9,10,14	7404	
IC 20,26	7408	
IC 32	7432	
IC 7,13,31	7442	1 of 10
IC 57,69	7475	QUAD LATCH
IC 63	8269	4 BIT COMPARATOR

Channel Selection and Device Decoding

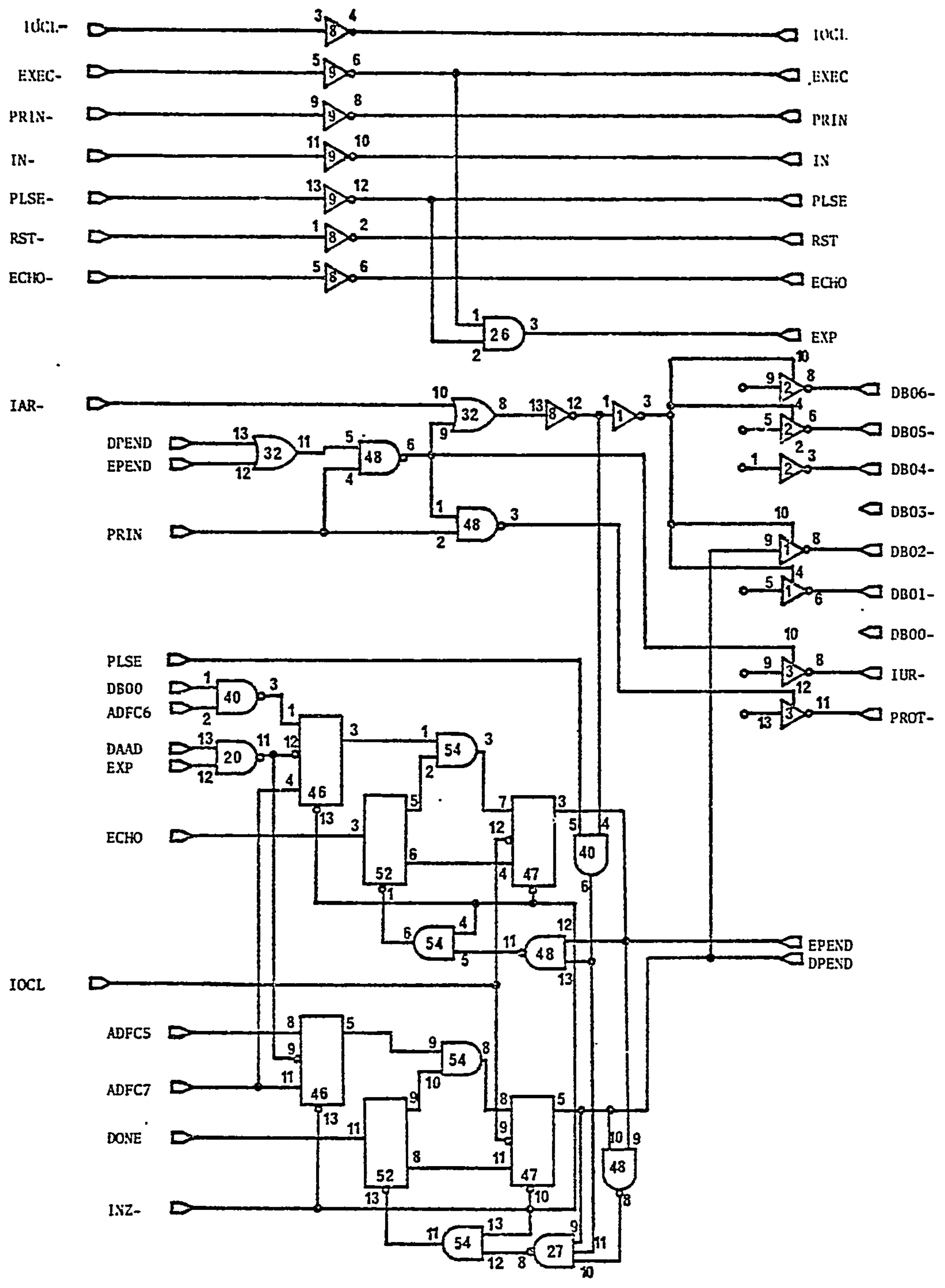


IC 20	7400	
IC 34	7402	
IC 35	7408	
IC 27	7410	
IC 28	7411	
IC 33	7432	
IC 29	7474	DUAL D TYPE FLIP FLOP

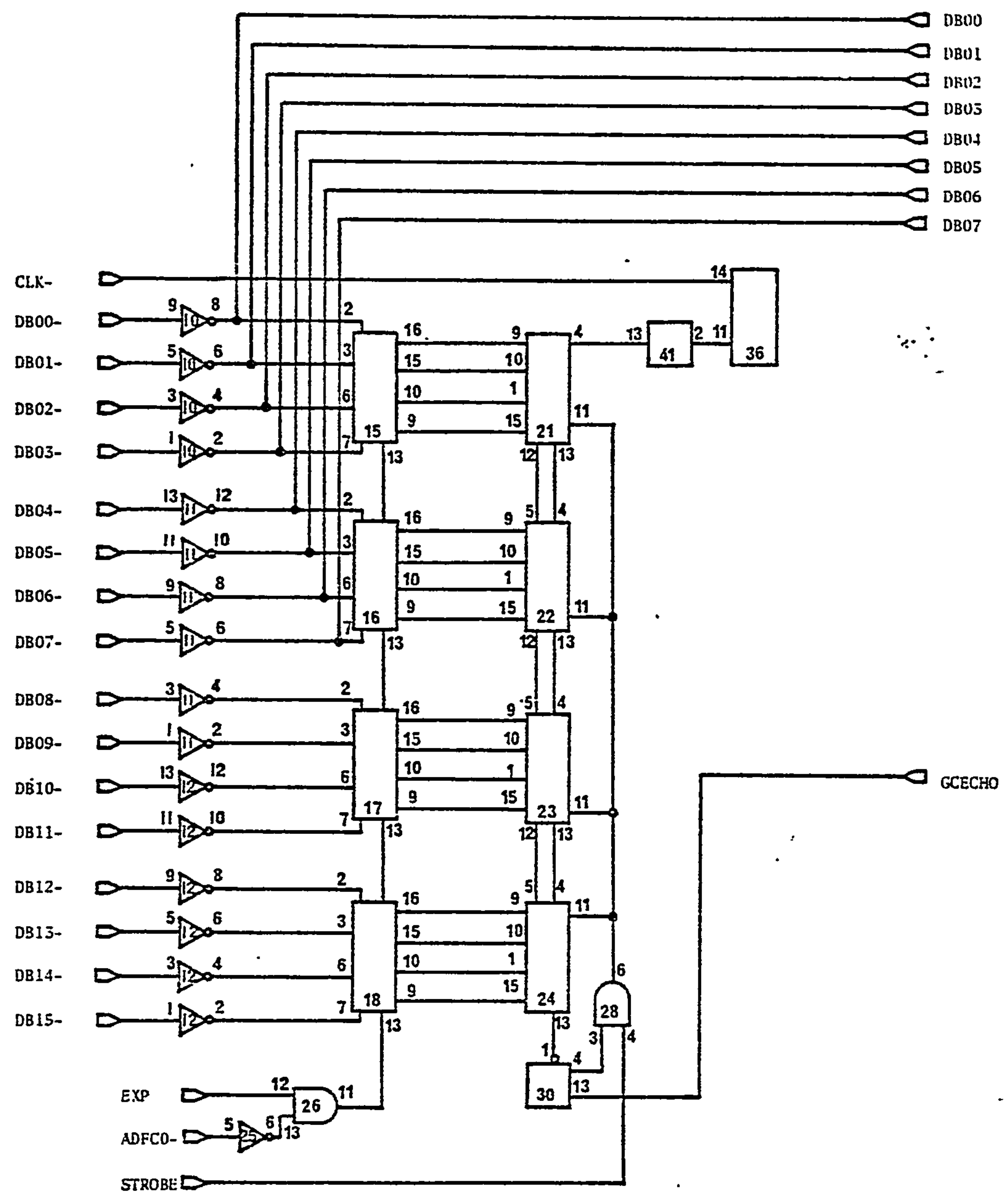
A/D Start Select



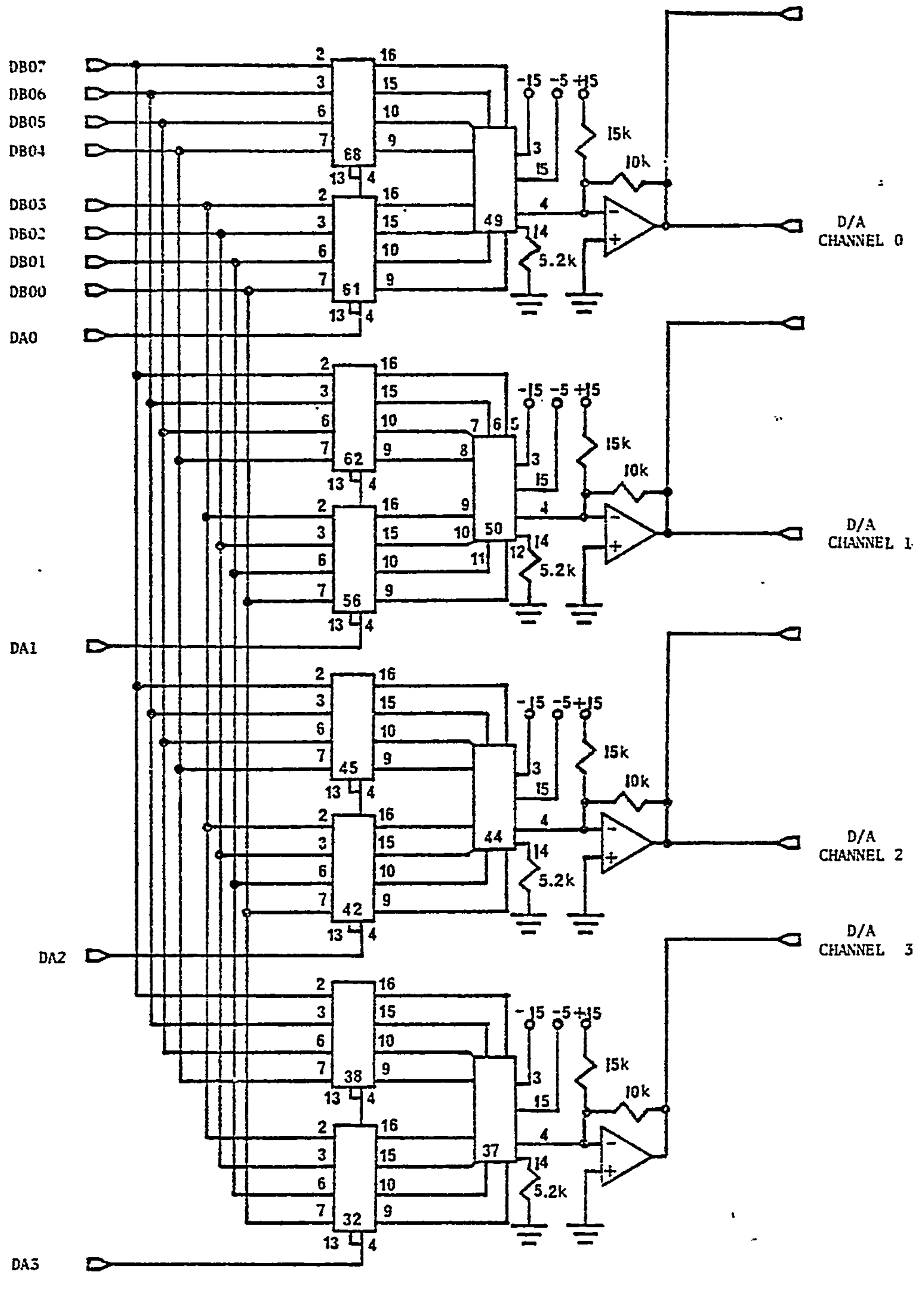
Push Down Stack and Channel Sequencing



Interrupt Generation



Real Time Clock



D/A Output Section

APPENDIX D;FLOATING POINT FORMAT DESCRIPTION

```
Word 0     sign  0 0 0   0 0 0 0   x Expone  ntx x x
            bit

Word 1           0 0 1.x   x Manti   ssa x x   x x x x
```

The first byte of word 0 of a floating point· number contains the sign bit followed by seven binary zeros. The second byte of the word is the base 2 exponent of the floating point number with a bias of 80 (in hexidecimal) added to it.

Word 2 of the number contains the mantissa of the number, with the first three binary digits being 001, and the implied binary point after the third bit.

For example the floationg point representation of the number 1 would be 80, 2000 in hexidecimal. The number -1.5 would be 8080, 3000.