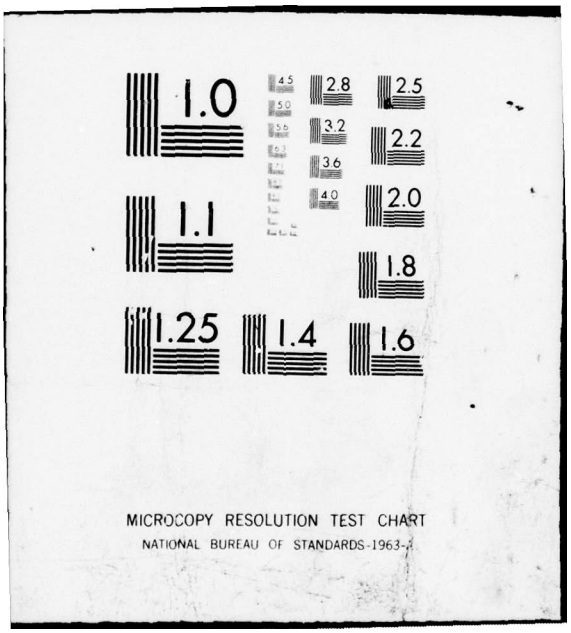
AD-A064 728 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 9/2
MICROPROCESSOR BASED DATA ACQUISITION AND PROCESSING SYSTEM.(U)
DEC 78 S IFTEKHAR

UNCLASSIFIED AFIT/GE/EE/78-29 NL

1 OF 3
AD
AD 64728


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1
8.0
9.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AFIT/GE/EE/78-29

MICROPROCESSOR BASED DATA ACQUISITION AND PROCESSING SYSTEM

THESIS

Saleem Iftekhar
Flt/Lt PAF

AFIT/GE/EE/78-29

Approved for public release; distribution unlimited.

see title page

(14) AFIT/GE/EE/78-29

(6) MICROPROCESSOR BASED DATA ACQUISITION AND PROCESSING SYSTEM.

(9) Master's THESIS,

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science

by

(10) Saleem Iftekhar
~~Flt/Lt~~ PAF

Graduate Electrical Engineering

(11) December 1978

(12) 239 p.

Approved for public release; distribution unlimited.

012225

B

Preface

This report is the result of my attempt to develop a microprocessor based data acquisition and processing system for the Air Force Flight Dynamics Laboratory, Wright-Patterson Air Force Base, Ohio. The report describes the three phases of system development: conceptual phase, design phase, and implementation and test phase. Structured analysis and design technique was used in the conceptual and design phases. In my opinion, this graphical technique makes it possible to follow a top-down design approach. A bench-type model was the result of the implementation and test phase.

I wish to express my appreciation to Dr. Gary B. Lamont, my thesis advisor, for his many hours of help and advise. I greatly appreciate his detailed reading of the report to bring to light errors which I had made and his suggestions for improvement. I would like to thank the committee members, Capt Gregg L. Vaughn and Dr. Thomas C. Hartrum for their valuable guidance during the course of the project.

Also, I would like to thank Dr. Lynn C. Rogers and Mr. Robert W. Gordon of the Air Force Flight Dynamics Laboratory who provided the thesis topic. Their interest in my work and assistance during the project were invaluable.

I would like to mention my appreciation to Lieutenant Peter D. Summers, III, and Mr. Frank E. Beitel of the Air Force Materials Laboratory, Computer Activities Division, Wright-Patterson Air Force Base, Ohio, for extending their help during the software development.

The manuscript in its present form would not be possible without the efforts of my typist, Miss Joyce L. Wilson.

Finally, I wish to acknowledge my indebtedness to my wife and baby for their understanding, patience, and encouragement during the course of this project.

Saleem Iftekhar

Contents

Contents

Contents

List of Figures

List of Figures

List of Figures

List of Tables

Abstract

The Air Force Flight Dynamics Laboratory at Wright-Patterson Air Force Base, Ohio, plans to design a constrained layer damping treatment for aircraft components. This design requires the knowledge of vibration and temperature variations that a component encounters in service.

The development of a microprocessor based data acquisition and processing system (MIBDAPS) to monitor the vibration and temperature variations of a component under test is discussed in this paper. The designed system is capable of acquiring and processing data in real time and generating cumulative damage factor versus frequency and temperature contours. These contours are used in the design of the damping treatment.

The system uses an LSI-11M microcomputer as its central processor. The support hardware for this microcomputer includes an analog-to-digital converter, core memory, real time clock, and serial line interface module. The software developed for this system handles the acquisition and processing of data. The real time processing includes the computation of the FFT and power spectrum of the vibration signal and subsequent calculation of the cumulative damage factor versus frequency and temperature contours. The bench model of the designed system was implemented and tested for functional performance.

I. Introduction

The Air Force Flight Dynamics Laboratory (AFFD), Wright-Patterson Air Force Base, Ohio, plans to develop a constrained layer damping treatment for aircraft components such as the airframe, avionics hardware, and other accessories. The specific damping layer treatment design requires knowledge of temperature and vibration variations encountered by the component. To generate this information, a data acquisition and processing system is required to serve as an in-flight monitor for the component under test. This report describes the design, development, and bench model implementation of such a system.

Background

Research has established that high-cycle fatigue damage due to resonant vibration is one of the major reasons for structural failure of components (Ref 13). A constrained layer damping treatment would be cost effective in extending the life of the component. In order to determine the correct type of damping layer, the temperatures, vibration nodes, and frequencies at which damage occurs in service must be known.

It has been determined that there are approximately ten vibration frequency bands of interest (Ref 13). They lie in the range from 100 HZ to 1000 HZ and are spaced 100 HZ apart (Figure 1). A damage factor (DF) at each frequency of interest can be calculated using the following relationship (Ref 13):

$$DF = \frac{K(\bar{A})^{\beta}}{(f)^{2\beta-1}} \qquad (1)$$

Here, β is approximately equal to 3·2, and $\bar{A}$ is the root-mean-square (RMS) value of acceleration in the frequency band of interest.

A summation of the DF over a predetermined period of time would yield a cumulative damage factor (CDF) and curves as shown in Figure 2 could be drawn.

Existing in-flight data acquisition techniques employ an analog signal conditioner and a multichannel analog tape recorder (Ref 13). The analog signals from the vibration and temperature sensors are conditioned and recorded on a magnetic tape. This system is bulky and takes excessive setup time. Skilled personnel are required to install the equipment on an aircraft, and calibration is difficult and time-consuming.

To simplify this process, the design of a self-contained Microprocessor Based Data and Processing System (MIBDAPS) appeared as an attractive alternative. Such a system should be easy to install and remove, and only minimum operator intervention during its operation should be necessary.

System Concept

The underlying concept of MIBDAPS was a self-contained system that would monitor the analog signals from the temperature and vibration sensors, digitize these signals, process the data in real time, and store the results on an appropriate media. The real time processing would include computation of the power spectrum of the vibration signal and calculation of DF at each frequency of interest. The results of processing would be stored in a CDF versus frequency and temperature array, respectively. Therefore, at the termination of system operation, the CDF versus frequency and temperature contours (Figure 2) should be available.

MIBDAPS should be capable of either 15 to 20 hours of continuous operation (for gathering data on transport aircraft hardware) or time

segmented operation (for gathering data on fighter aircraft hardware which has limited flying endurance). During a time segmented operation, the system should be able to operate for a total aggregate of 15 to 20 hours. The data processed during this time interval would enable determination of vibration nodes at which the damage occurs.

Approach

The development of MIBDAPS was carried out in three phases: the conceptual phase, the design phase, and the implementation and testing phase. During the conceptual phase, the detailed requirements placed on the system were defined using a structured analysis and design technique (SADT) (Ref 1). The design phase included the software/hardware partition of the system, selection of hardware components, and design of software and hardware modules to implement different functions of MIBDAPS. During the implementation and testing phase, the software and hardware modules were given a functional check. The system integration and performance check of the integrated system was also completed during this phase.

The Development Environment

The LSI-11M* microcomputer was used as the system processor because of its powerful instruction set, speed, and availability of support hardware and software. The use of the LSI-11 microcomputer system for MIBDAPS determined the general development requirements for both hardware and software modules of MIBDAPS. The off-the-shelf hardware modules used where analog-to-digital converter (ADC), real time clock (RTC), and the 4K by 16-bits core memory. All of these modules are manufactured by the Digital Equipment Corporation (DEC).

*LSI-11M is the militarized version that is manufactured by Norden Systems, Inc.

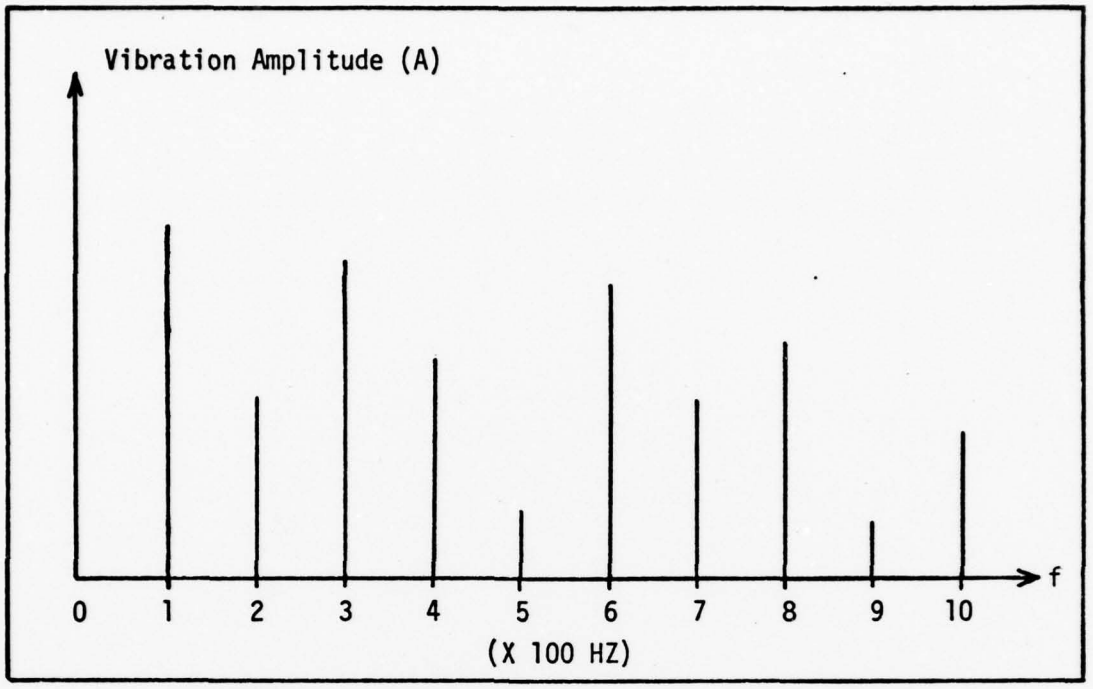


Figure 1. Vibration Frequency Bands

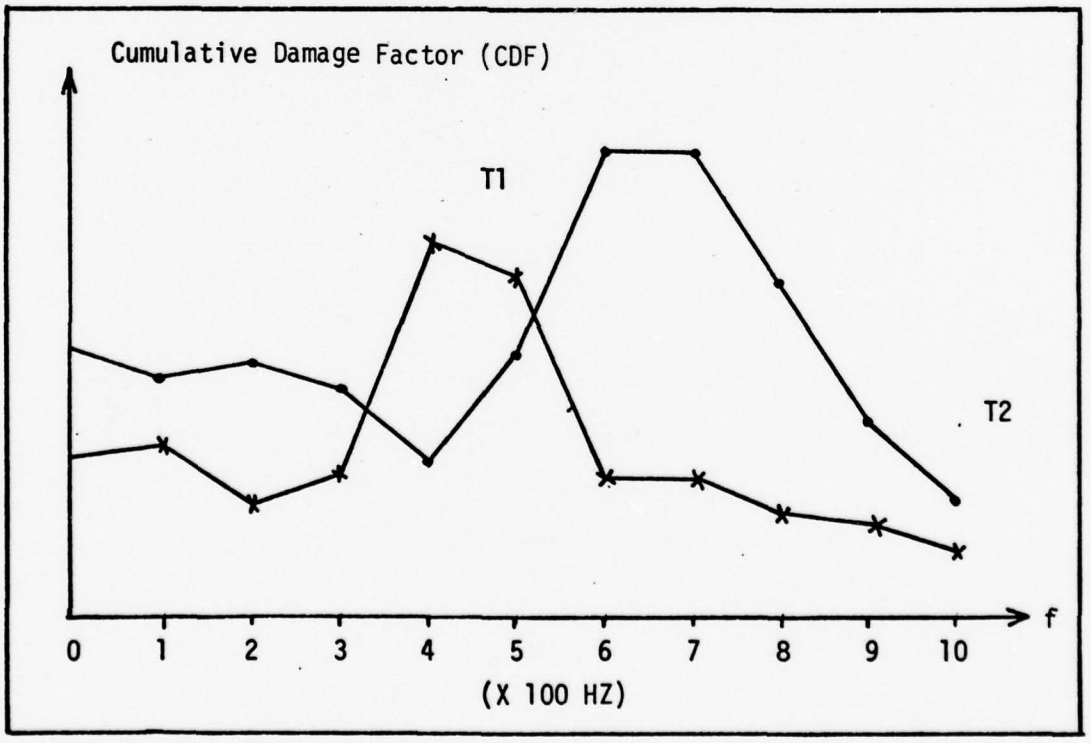


Figure 2. CDF Versus Frequency at Different Temperature Values

The software development system used was a PDP-11/03 minicomputer which has the LSI-11 as the processor. The RXV011 floppy disk based operating system was available on this minicomputer. The development system was located at the Air Force Materials Laboratory (AFML), Wright-Patterson Air Force Base, Ohio.

Scope of Thesis

The structured analysis and design technique was used to develop the activity model of MIBDAPS (Ref Chap II), and the LSI-11M microcomputer was used as the processor for MIBDAPS (Ref Chap III). Hardware and software modules were designed and tested (Ref Chaps IV and V). The hardware/software modules were then integrated into a bench model of MIBDAPS which was then functionally tested (Ref Chap VI). The bench model of MIBDAPS performed the required functions of data acquisition and processing. Testing of the integrated system for computational accuracy has not been completed at this time. Recommendations for further work have been generated in order to complete the implementation of the prototype module (Ref Chap VII).

II. System Requirements for MIBDAPS

Introduction

The conceptual idea is not enough to form the basis of design and implementation of any complex system. The design can be facilitated if the complete system is divided into functional subunits. These subunits are easier to analyze and design. The division of a system into functional subunits according to the requirements placed on the system can be called the "requirement definition" phase of the design. A carefully prepared requirement definition ensures that ambiguities in the design and implementation phase are minimized.

Analysis and Design Methodology

Structured Analysis and Design Techniques (SADT) is a design methodology for performing the functional analysis and design of complex systems (Ref 1). A functional model of the system is created first which helps in understanding what activities the system will perform, followed by a design model to show how the system will be implemented to perform the requisite functions (Ref 1). The functional model also helps to clarify the requirement placed on the system.

Appendix A describes SADT methodology and its application rules. Some changes in convention are made to suit the design of MIBDAPS; these are also included in Appendix A.

The SADT was chosen as a functional analysis and design tool because it is a graphical method of analyzing the functional requirements of a system. The activity model, which shows different functions which the system is required to perform, is independent of the hardware/

software implementation of the system. Thus, any changes in the activity model can be made during this phase of the design without affecting the final shape of the product.

Once the activity model is complete, it is easy to review and correct any oversights that might have occurred during the functional analysis of the system.

Activity Model

Before the activity model is drawn, the system is divided into different nodes. The node index given in Table I shows the hierarchy of the system from activity viewpoint. This node index is based on the initial problem definition by the Air Force Avionics Laboratory (AFAL), Wright-Patterson Air Force Base, Ohio, and subsequent discussions with the sponsor of this thesis. The node index also indicates how the author visualized the system was required to function.

TABLE I

Node Index

A-1	Data Collection and Analysis
A-Ø	Acquire and Process Data
AØ	Acquire and Process Data
A1	Control All Activity
A2	Acquire Data
A3	Process Data
A4	Store Data

The explanation of each node follows.

Node A-1, Data Collection and Analysis (Figure 3). This node shows the overall research activity and the place where MIBDAPS fits in the overall function of the organization. Block 3, Plot Data, activity will be carried out in the laboratories.

Node A-Ø, Acquire and Process Data (Figure 4). Node A-Ø is the model of the MIBDAPS. The node shows the system requirements. Data input comes in the form of analog signals—analog vibration signal (I1) and analog temperature signal (I2). The output O3 of the system is a three-dimensional array. This array contains the cumulative damage factor (CDF) as a function of frequency and temperature. Output O2 is the scale factor for the array elements, and output O1 is the total operational time for which the system was operational.

The MIBDAPS will be able to continuously acquire and process data for a total flying time of 15 to 20 hours. This is the requirement placed by AFAL because the CDF versus temperature and frequency array for this duration will be a good estimate of the fatigue damage profile of the test piece (Ref 13). Input signal specifications are:

Analog Vibration Signal (I1)	0-5V
Analog Temperature Signal (I2)	0-5V

Node AØ, Acquire and Process Data (Figure 5). The MIBDAPS is expected to acquire data, process data, and store the results. Therefore, node AØ shows the decomposition of the system into four primary activities:

Acquire Data
Process Data
Store Data
Control All Processes

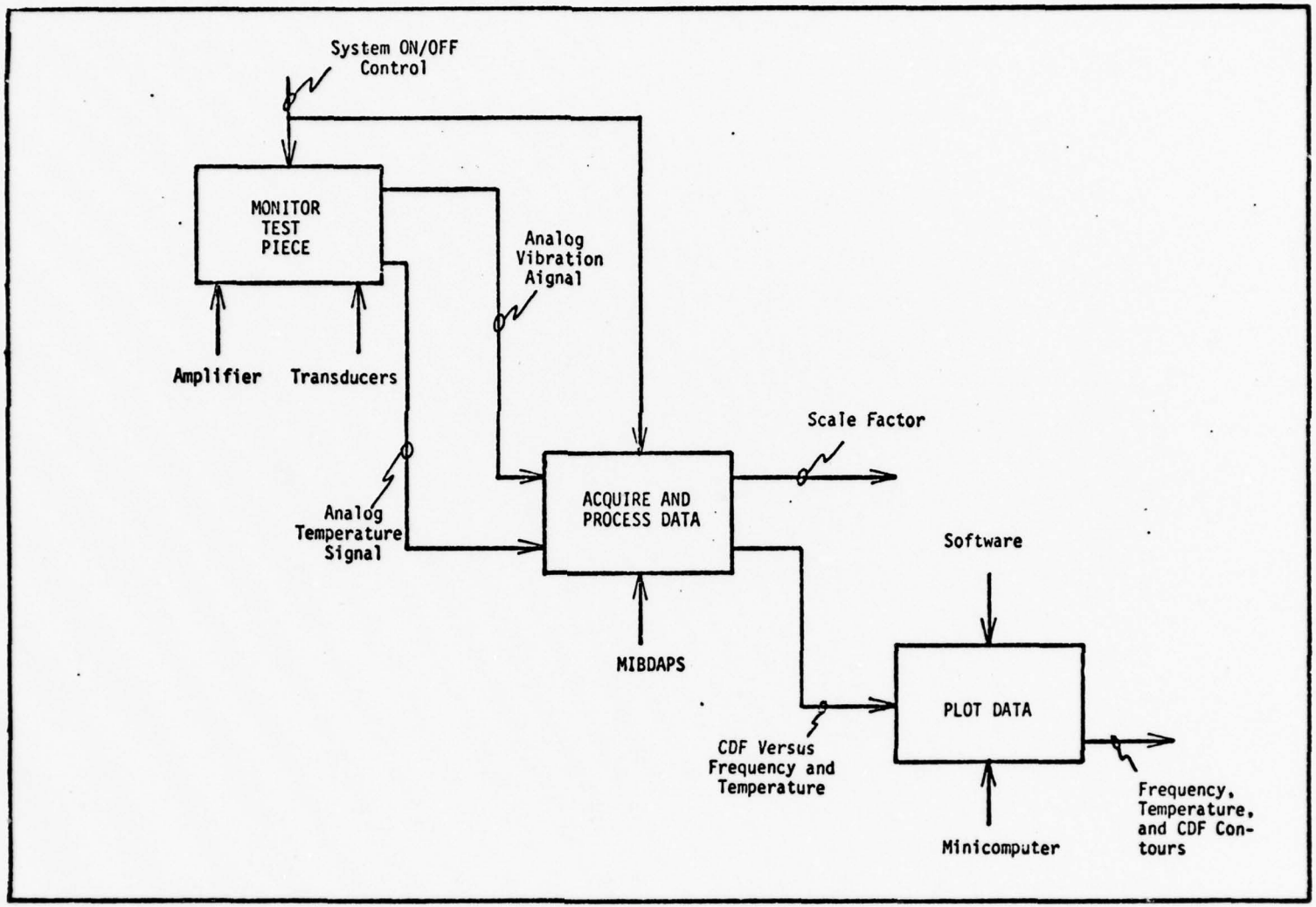


Figure 3. Node A-1, Data Collection and Analysis

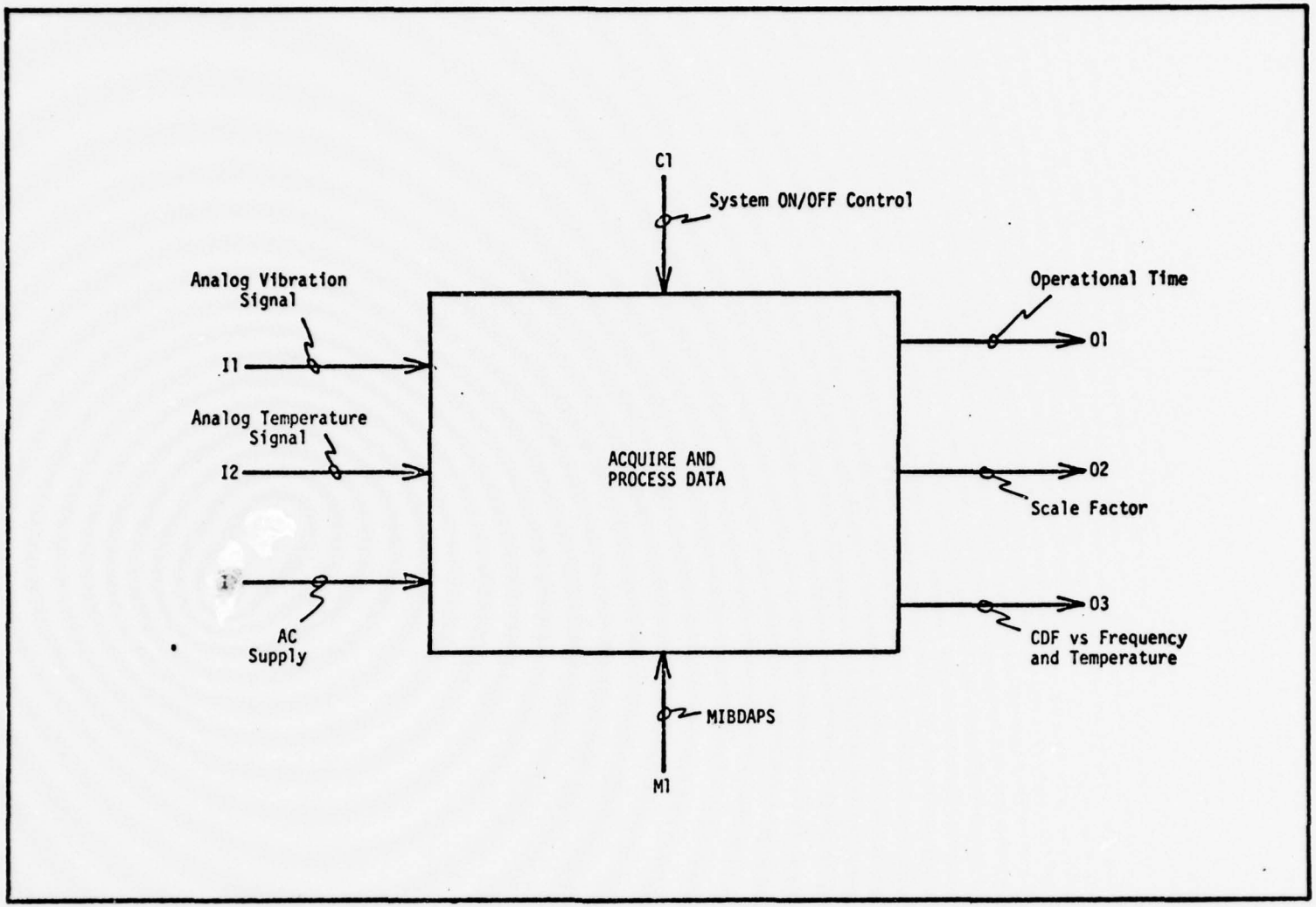


Figure 4. Node A-Ø, Acquire and Process Data

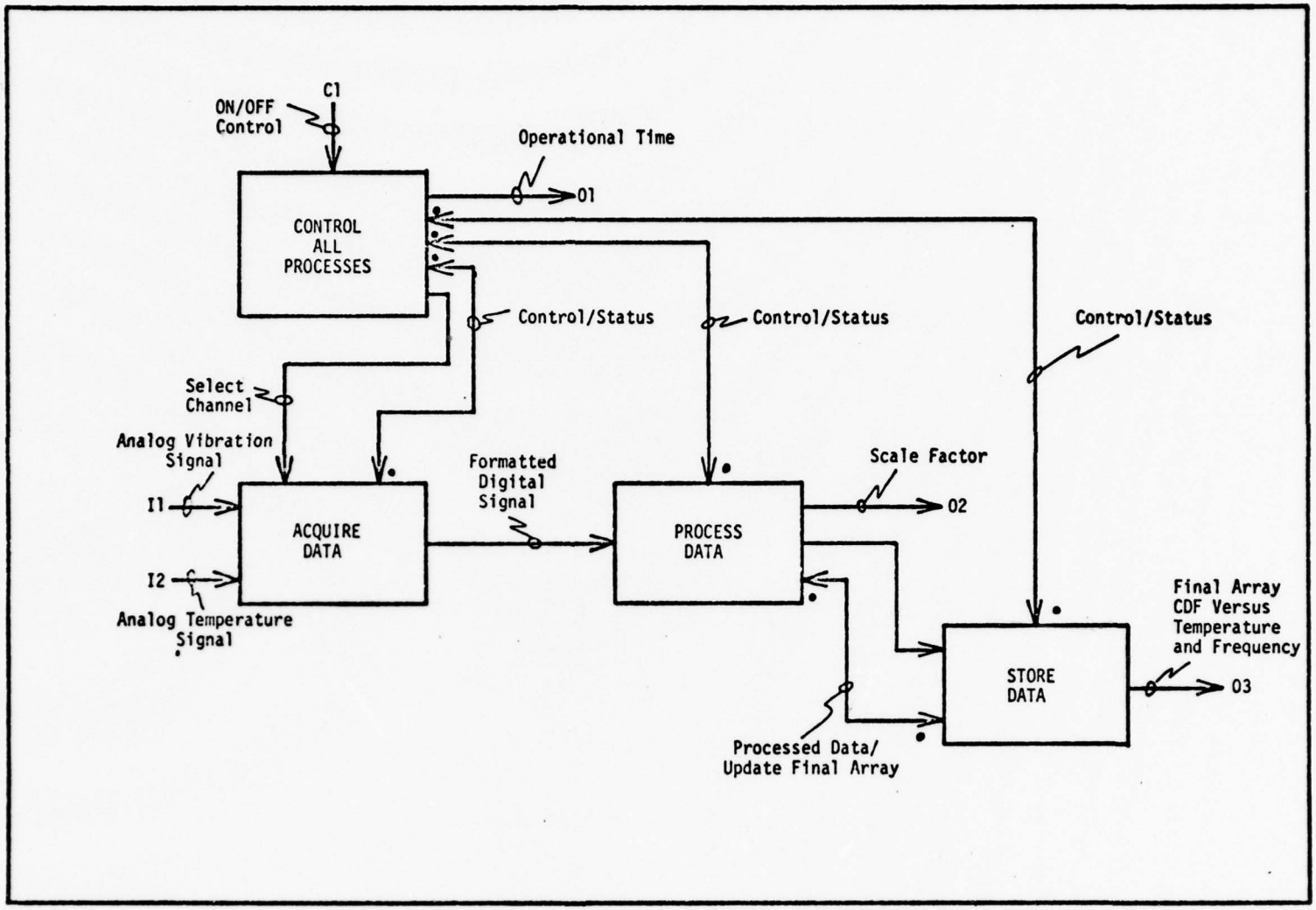


Figure 5. Node AØ, Acquire and Process Data

An explanation of how a signal travels through each subnode is now given. The activity Acquire Data continuously digitizes the input signal under the control from node A1. The digitized signal is queued and sent to the processing node (A2).

The power spectrum of the signal is computed by the processing node. For each frequency of interest, a damage factor is calculated (Ref Chap I) and stored in an array. The activity Store Data adds the most recent set of DF values to the previously stored values. This results in a storage of cumulative damage factors.

Node AØ1, Control All Processes, keeps all events in the system sychronized. It initiates appropriate commands to ensure that data is processed and stored in proper order. The input and output signal specifications are given below:

- Input Signal
  a. Analog Vibration Signal (I1): 0-5V.
  b. Analog Temperature Signal (I2): 0-5V.
  c. AC Supply (I3): 110V, 60 HZ.
- Output Signal
  a. Operational Time (O1): Displayed on the front panel.
  b. Scale Factor (O2).
  c. CDF Versus Temperature and Frequency Array (O3).

Node A1, Control All Activity (Figure 6). Node A1 subdivides the overall control of the system into the following functions:

Control Acquisition of Data
Control Processing of Data
Control Storage of Data
Control Power to the System
Record Operational Time

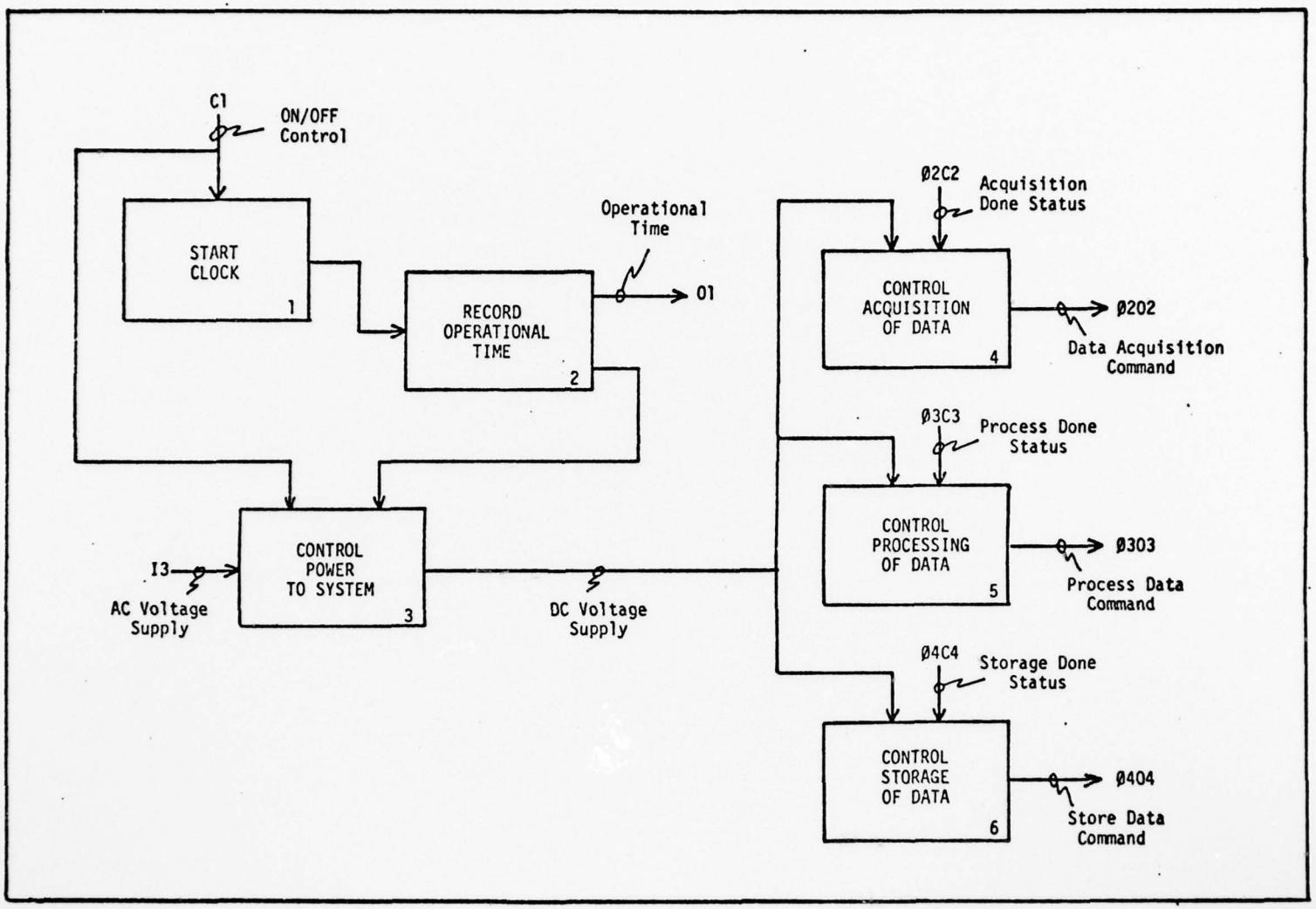


Figure 6. Node A1, Control All Activity

Control signal C1 (switch ON/OFF) activates the system. The power to the system is also controlled by total operational time; therefore, at the end of 15 to 20 hours of operation, the system halts automatically (Ref 13). The output and control signals are listed below along with their explanation.

- Output Signal

a. Operational Time (01): Four binary coded decimal (BCD) numbers, each 4-bit long, display the operational time elapsed in hours and minutes.

b. Data Acquisition Command (Ø2Ø2): This control signal initiates the acquisition of N-data points. This signal acts as a control signal to node A2.

c. Process Data Command (Ø3Ø2): This signal starts the processing of either the raw data or update of CDF versus temperature and frequency array.

d. Store Data Command (Ø4Ø3): This command signal initiates either the storage of the N-processed data element or storage of updated CDF versus temperature and frequency array.

- Control Signal

a. Switch ON/OFF Control (C1): This is a logic "1" when the input switch is closed and a logic "0" when the input switch is open.

b. Acquisition Done Status (Ø2C2): This signal indicates node A14 whenever data acquisition of N-data points is complete.

c. Process Done Status (Ø2C3): This control signal indicates the completion of processing by node A3. The processing is initiated by node A15.

d. Storage Done Status (Ø4C4): Whenever the storage of data is required, node A16 initiates this activity. Once the storage of data is complete, this signal indicates the status.

Node A2, Acquire Data (Figure 7). This node converts the analog vibration and analog temperature signals to an equivalent digital signal. The acquisition of data is initialized by node A1, and either the analog vibration signal or the analog temperature signal is selected. The selected signal is digitized, and the final output is an equivalent digital value of the input signal along with a signal identifier. The input/output and control signals specifications are:

- Input Signal

a. Analog Vibration Signal (I1): 0-5V.

b. Analog Temperature Signal (I2): 0-5V.

- Output Signal

a. Acquisition Done Status (Ø101): This signal indicates the status of the data acquisition process. Whenever N-data elements have been acquired, this status signal informs node A1 about the completion of data acquisition.

b. Formatted Data (Ø302): This is the digitized data along with an identifier which indicates whether the data point corresponds to the analog vibration signal or the analog temperature signal.

- Control Signal

Data Acquisition Control (Ø1C1): This signal initiates the acquisition of N-data elements. This signal controls node A2 in the following manner:

(1) It selects either the analog vibration signal or the analog temperature signal.

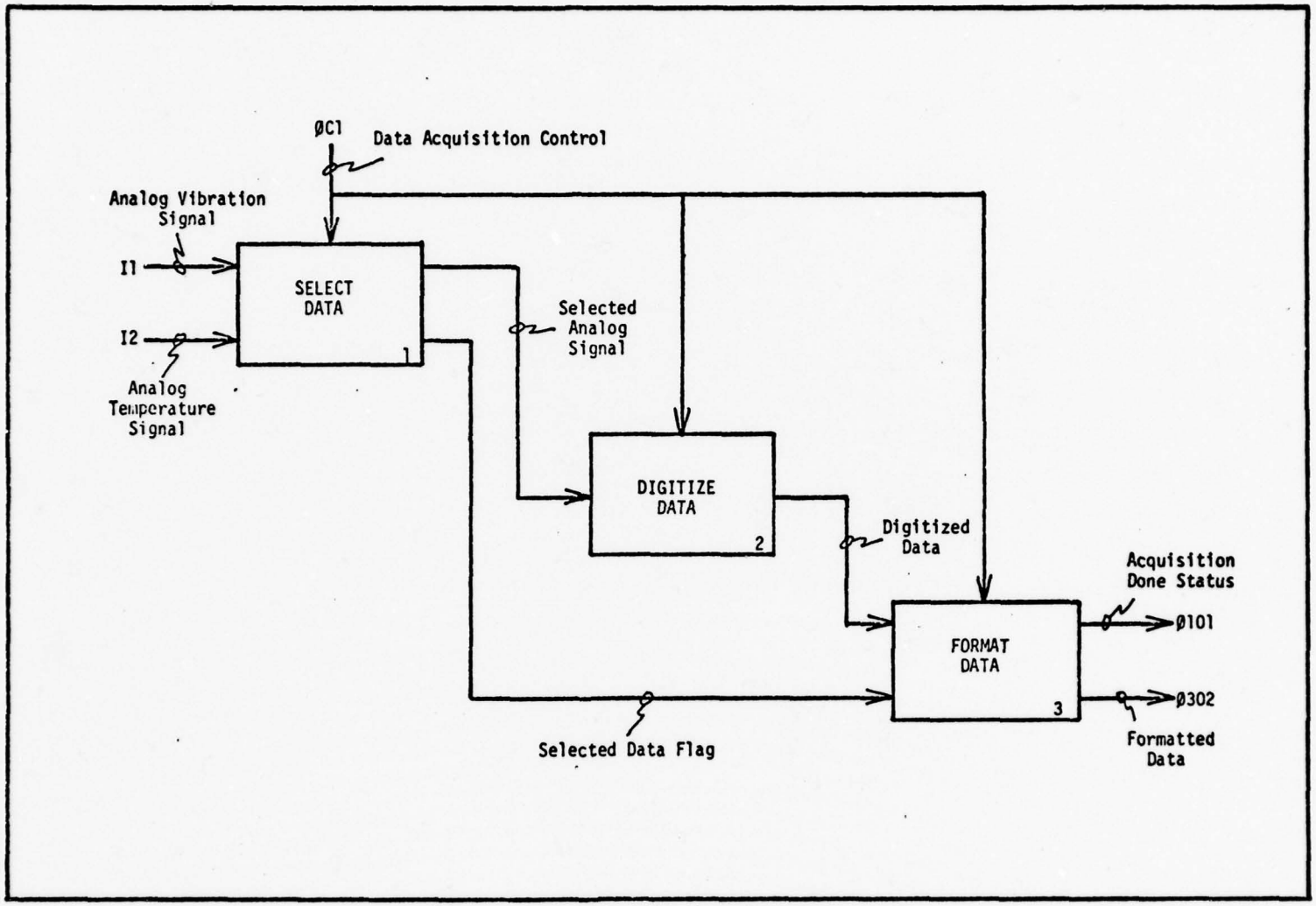


Figure 7. Node A2, Acquire Data

(2) It initiates digitizing of N-samples of an analog vibration signal.

(3) It initiates digitizing of one sample of an analog temperature signal for every N-samples of an analog vibration signal.

(4) It initiates formatting of each digitized data point.

Node A3, Process Data (Figure 8). The activities for this node are listed below:

a. N-elements of raw data array are taken from the raw data storage area and processed. The processing involves Fourier transform of N-raw data points. The power spectrum of first M-frequency components is computed next (M is less than N). The DF is calculated for each of the M-frequency components.

b. M-elements of the final data array (CDF versus temperature and frequency) are updated whenever a new set of DF array is available after processing of N-raw data points.

The input/output and control signals specifications are:

- Input Signal

a. Raw Data From Storage (Ø4I1): The N-element array from the Raw Data Storage area is used by the Process Raw Data module to give an M-element processed array.

b. New Data Array (Ø4I2): This is the data array which has resulted from processing N-element raw data array.

c. Old Data Array (Ø4I3): CDF versus temperature and frequency array which has to be updated by adding in new data array. The Update Data Module creates updated data array.

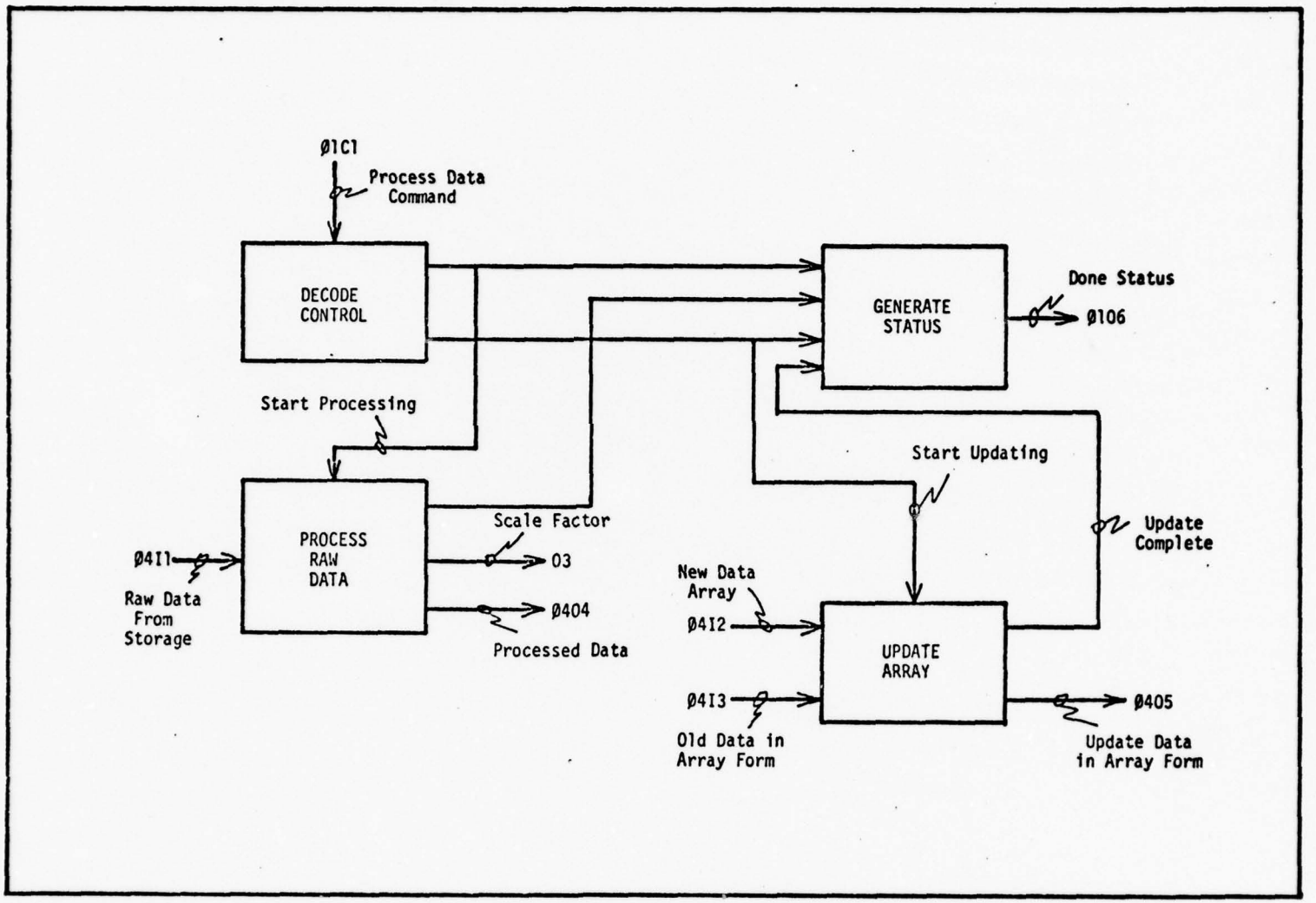


Figure 8. Node A3, Process Data

- Output Signal

a. Scale Factor (03): This is the value which tells by what amount the elements in the final arrays have been scaled.

b. Processed Data (Ø404): This is an M-element data array where the Process Raw Data module stores the results after operating on raw data array.

c. Updated Data Array (Ø405): This is an M-element array. The value of CDF as a function of frequency and temperature is stored in this array. Every time a new set of M-processed data elements is available, the final array is updated by adding these new values to the old array.

- Control Signal

Process Data Command (Ø1C1): This command initiates the processing of N-raw data elements. After the processing is completed, the CDF versus temperature and frequency array is updated. The update takes place by adding the newly processed data array to the existing final array (CDF versus temperature and frequency).

Node A4, Store Data (Figure 9). This node also performs multiple activities on the data. The different activities are enumerated below.

a. The Formatted Raw Data points are stored sequentially in an N-point array. This array is called Raw Data Array, and it serves as a temporary storage area.

b. Whenever raw data has to be processed, N-elements of the raw data array are transferred to node A3 for processing.

c. The processed data is stored in M-elements processed data array. This storage of processed data points is temporary.

d. Whenever the final output arrays have to be updated, M-data elements from the Process Data Array are transferred to node A3.

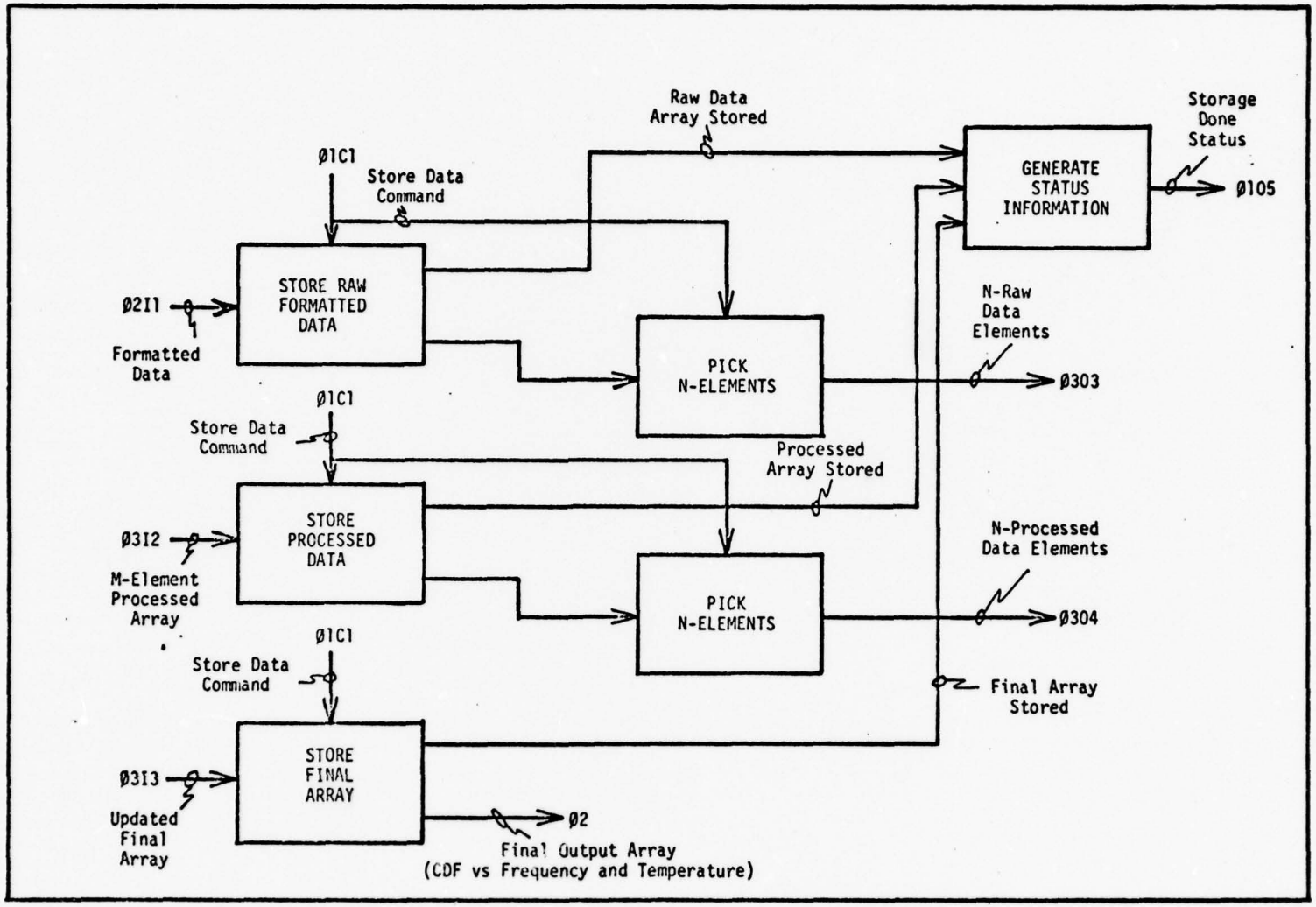


Figure 9. Node A4, Store Data

e. The Update Final Array (Ø3I3) is sent from node A3, and it is stored as the final CDF versus temperature and frequency array (02).

The input/output and control signal specifications are:

- Input Signal

a. Formatted Data Input (Ø2I1): This is the formatted digitized version of input analog signals. The data points at this stage are also known as Raw Data Elements.

b. M-Element Processed Array (Ø3I2). This is an M-element processed array, which is the output from node A3. This array is generated every time N-raw data elements are processed.

c. Updated Final Array (Ø3I3): The final output array (CDF versus frequency and temperature) has to be updated whenever a new M-element processed array is ready. This input is the result of updating the existing final output array.

- Output Signal

a. N-Raw Data Elements (Ø303): Node A3 requires N-raw data elements whenever processing has to be done. This output is actually an N-raw data element array, which is sent to node A3 for processing.

b. M-Processed Data Elements (Ø304): This output is used to update the final output array, so the CDF versus frequency and temperature is continuously updated.

c. Storage Done Status (Ø105): This signal is used to synchronize different storage operations. The Control All Process, node A1, is informed about the storage status of node A4.

d. Final Output Array (02): This array stores CDF as a function of frequency and temperature and is the final output for MIBDAPS.

Summary

In this chapter, the overall requirements placed on MIBDAPS are defined. All possible activities which MIBDAPS is expected to perform, along with their interrelationships, have been consistently and completely documented. At this stage, engineering decisions about the mechanization are not taken; however, the next phase of the design would be the engineering decision regarding the mechanization of the system. Hardware and software subdivision would also take place in the next phase.

III. Software/Hardware Division of MIBDAPS

Introduction

The functional analysis model for MIBDAPS, which was developed in Chapter II, does not discuss mechanization of the system. The choice of the hardware for mechanization of MIBDAPS will, to a large extent, determine the subdivision of the system functions. In this chapter, the engineering decisions that define the realization of MIBDAPS are presented. The implementation of different modules of MIBDAPS and the subdivision of different modules into software and hardware activities are also presented. SADT is again employed in this development of MIBDAPS' activities.

Choice of Microcomputer System

One of the objectives in the development of MIBDAPS was a self-contained microprocessor based data acquisition and processing system. Thus, there were two possibilities for the mechanization of MIBDAPS:

i) Selection of an off-the-shelf microprocessor with the requisite word length and instruction set, followed by the development of a useable microcomputer system with the required amount of read/write memory (RAM), input/output (I/O) interface circuits, and development of application software.

ii) Selection of an industry standard microcomputer board with already developed I/O interface circuits, on-board memory and memory expansion capability, and software development tools.

At this stage, the approach taken for implementation of MIBDAPS was, in part, guided by the time available for the project. The first

possibility where the system could be developed around a selected microprocessor would encompass design of I/O interfaces, memory boards, and finally the development of software. This would have taken longer than the allotted time. Therefore, it was decided that an industry standard microcomputer board along with its existing I/O interface circuits, memory options, and software development tools would be used in the implementation of MIBDAPS.

Selection of Microcomputer Board

The main features which were considered in the selection of the microcomputer board are given below:

Word Size
I/O Capability
Instruction Set
Memory Capacity
Software Development Tools

Since the MIBDAPS is to process the input signals in real time, the choice of the microcomputer word size and the instruction set would determine the processing speed of the system. The word size determines the number of memory accesses required for execution of each instruction. For the same instruction set, a machine with a smaller word size requires more memory access as compared with a machine with a larger word size (Ref 4). Thus, a 16-bit microcomputer was an obvious choice. The signal processing function (Ref Chap I) of MIBDAPS required that multiply and divide instructions should be a part of the instruction set. Although subroutines can be written to carry out multiplication/division of binary numbers, this technique is orders of magnitude slower than hardwired multiply/divide

instructions. This is because the hardwired instructions are micro-programmed, which is a lower primitive than the software implemented subroutine. Table II lists the candidate microcomputer boards. Texas Instruments' TMS 9900 and the Digital Equipment Corporation's LSI-11 have multiply/divide instructions as part of their instruction set. Also, both of these mucrocomputers have a 16-bit word length. Therefore, the choice was narrowed down to these two boards.

The data acquisition function of MIBDAPS (Ref Chap I) dictated the need for an analog-to-digital converter (ADC) board which would be compatible with the selected microcomputer. Also, the need for sychronization and timing of real time signal processing made it desirable to have a hardwired system to perform this function. The Digital Equipment Corporation (DEC) manufactures both an ADC board and a real time clock (RTC) board that are compatible with the LSI-11. This feature of the LSI-11 made it an obvious choice. Recently, Norden Systems Incorporated, which is a subsidiary of United Technology Incorporated, announced the manufacturer of a bit-sliced based microcomputer board (designated LSI-11M) which is six times faster than its commercial counterpart (see Table II). This board also conforms to military specifications (mil specs). Thus, the mil spec LSI-11M was selected as the microcomputer board to be used for MIBDAPS.

LSI-11 Microcomputer System

The LSI-11 microcomputer system is a microprocessor based version of DEC's PDP-11 series minicomputer. The PDP-11 development software (assemblers, debuggers, loaders, compilers) is useable on the LSI-11 based system.

TABLE II

Candidate Microcomputer Boards

Manufacturer	Candidate Microcomputer Board	Word Length	Resident RAM	Multiply Instruction Time (μsec)	Memory Expansion	I/O Lines	Vectored Interrupt (levels)
Norden Systems Inc.	LSI-11M*	16-bits	4K	12·5F**	32K	17	Yes (1)
Digital Equipment Corp.	LSI-11 (with KEV-11)	16-bits	4K	75/120F**	32K	17	Yes (1)
Texas Instruments Inc.	TI 990/4	16-bits	4K	18·0 μsec	20K	16	Yes (8)
Intel Corp.	SBC 80/XX	8-bits	256-4K	None	Yes	22-48	Yes (4-8)
Data General Corp	μnova 61	16-bits	2K/4K	None	32K	-	Yes

*Reference Norden Systems Product Brochure on LSI-11M.

**F = fixed-point arithmetic only.

(Adapted From Ref 4)

In the following paragraphs, a brief description of the LSI-11 processor, interface/memory options, and software development tools that were used in the implementation if MIBDAPS is given. These options were selected because they support either the hardware or the software realization of different modules of MIBDAPS. The detailed description of each of the above modules and the installation instructions are given in Ref 11.

KD11-F Microcomputer (Ref 11:87-99). The KD11-F microcomputer is contained on a single 8·5-inch by 10-inch printed circuit board. The module also contains resident dynamic 4K by 16-bit semiconductor RAM. Figure 10 is a functional block diagram of the microcomputer board. The salient features of this microcomputer are given below:

- This is a 16-bit microcomputer board using four custom LSI integrated circuit chips.

- Direct addressing of 32K 16-bit words or 64K 8-bit bytes (K = 1024).

- Efficient processing of 8-bit bytes without the need to rotate, swap, or mask.

- Hardware memory stock for handling structured data, subroutines, and interrupts.

- Direct memory access (DMA) for high data rate devices inherent in the bus architecture.

- Eight general purpose registers that are available for data storage, pointers, and accumulators; two are dedicated—stack pointer (SP) and program counter (PC).

- A bus structure that provides position-dependent priority to the peripheral devices that are connected to the I/O bus.

- Vectored interrupt facility.

- Typical operating speed is 400 nsec based on a 10 MHZ clock.

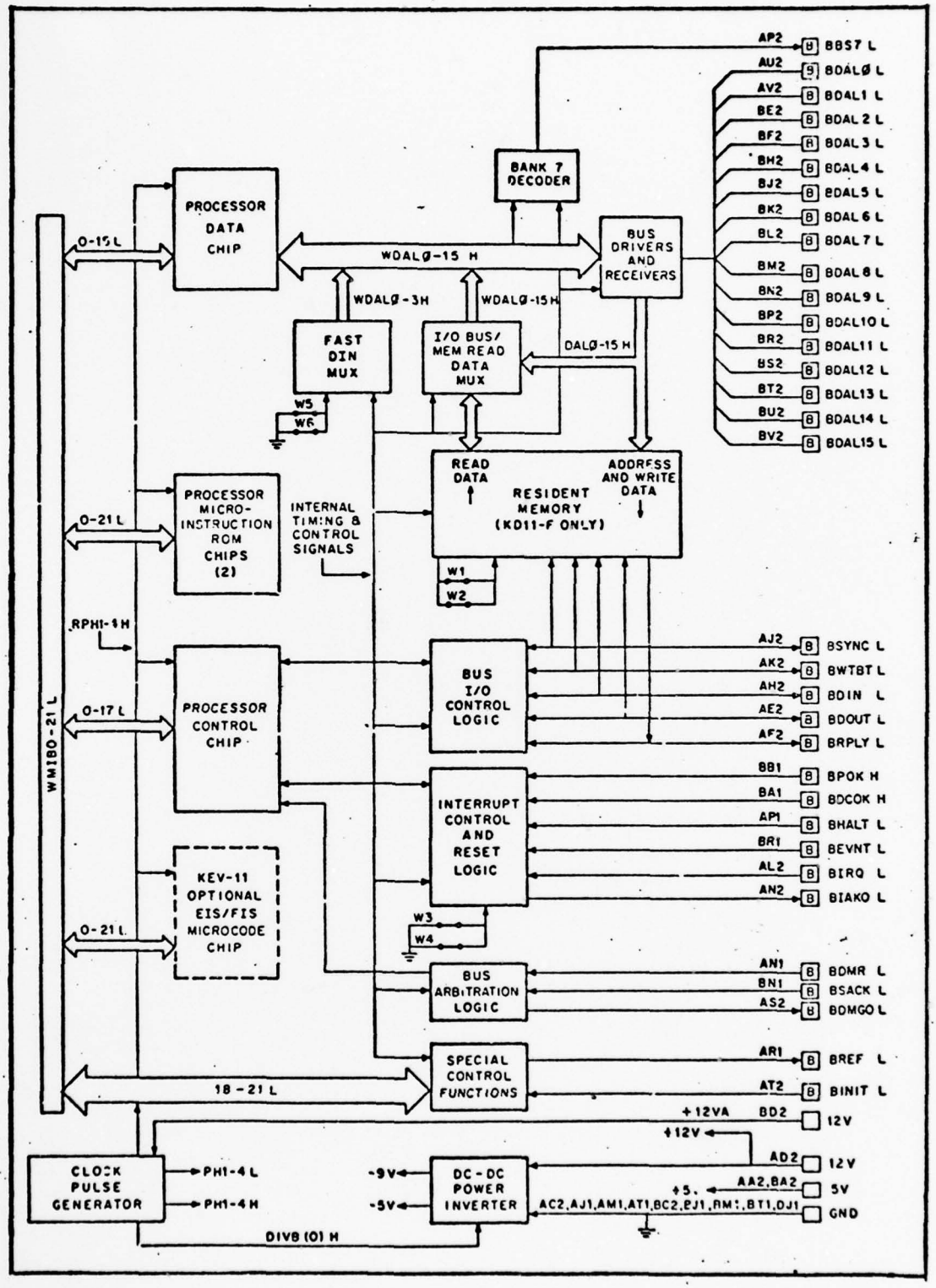


Figure 10. KD11-F Microcomputer Logic Block Diagram (From Ref 11:38)

- The power supply requirements are:

+5V ±5% at 2·4A maximum current

+12V ±5% at 1·1ØA maximum current

MMV11 4K by 16-Bit Core Memory (Ref 11:101-102). The MMV11-A 4K by 16-bit core memory option provides a nonvolatile read/write storage of the user's program and data (Figure 11). This optional module was used so that system software which would be developed for MIBDAPS could be stored permanently. This would eliminate the need to reload the program whenever the system is switched ON. The processed data would also be stored in the core memory for either updating or retrieval at a later time. The salient features of the MMV11-A core system are:

- 4096 by 16-bit capacity.

- Typical access time is equal to 425 nsec (475 nsec maximum); full read/restore cycle time is equal to 1·15 nsec.

- User-selected bank address; this allows the user to assign any 4K bank address.

- Power requirements are:

+5V ±5% at 2·ØA maximum

+12V ±5% at 0·56A maximum

ADV11-A Analog-to-Digital Converter (Ref 11:186-189). The ADV11-A is a 12-bit successive approximation analog-to-digital converter with a built-in multiplexer and sample-and-hold for use on the LSI-11 bus (Figure 12). The multiplexer selection accommodates 16 single-ended or 8 quasi-differential inputs.

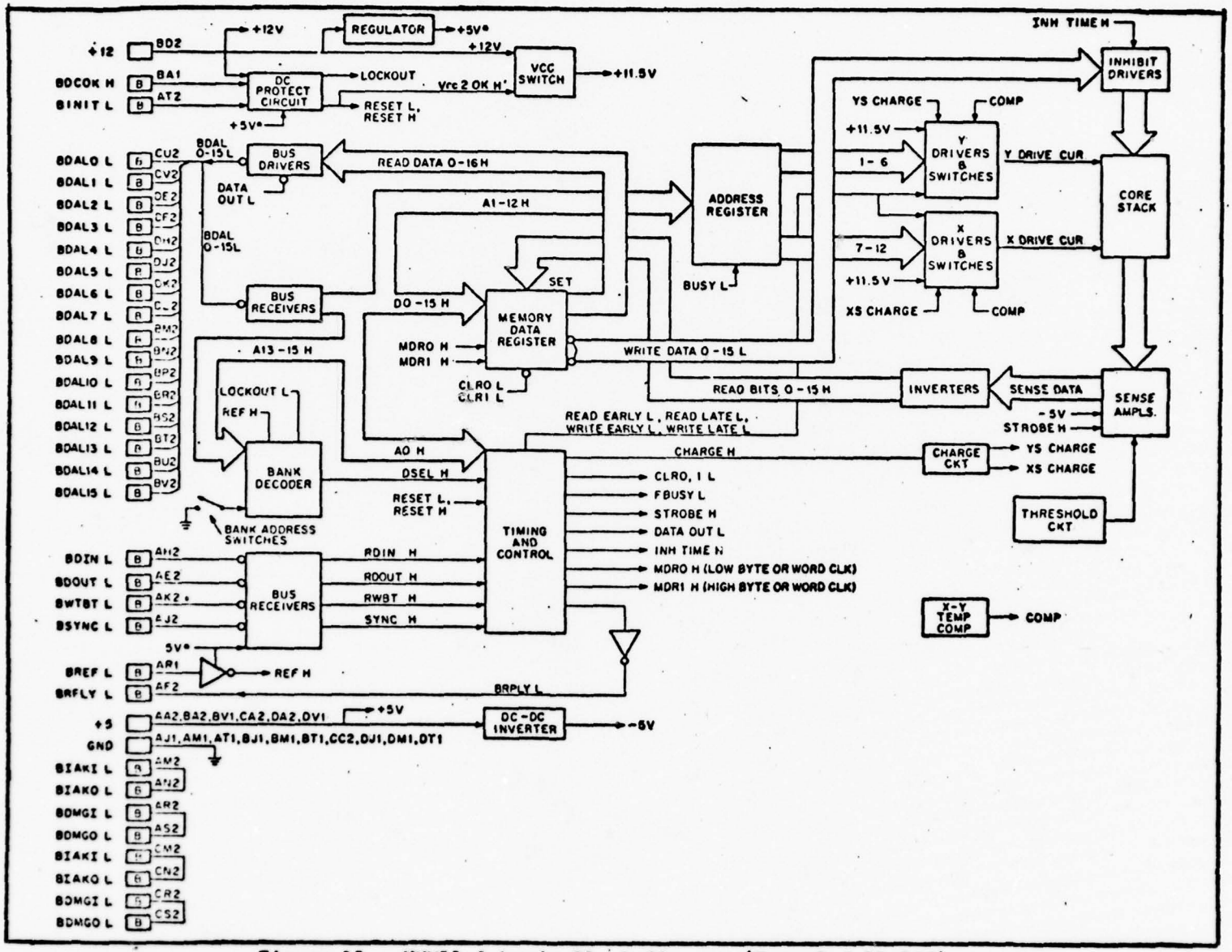


Figure 11. NMV11-A Logic Block Diagram (From Ref 11:105)

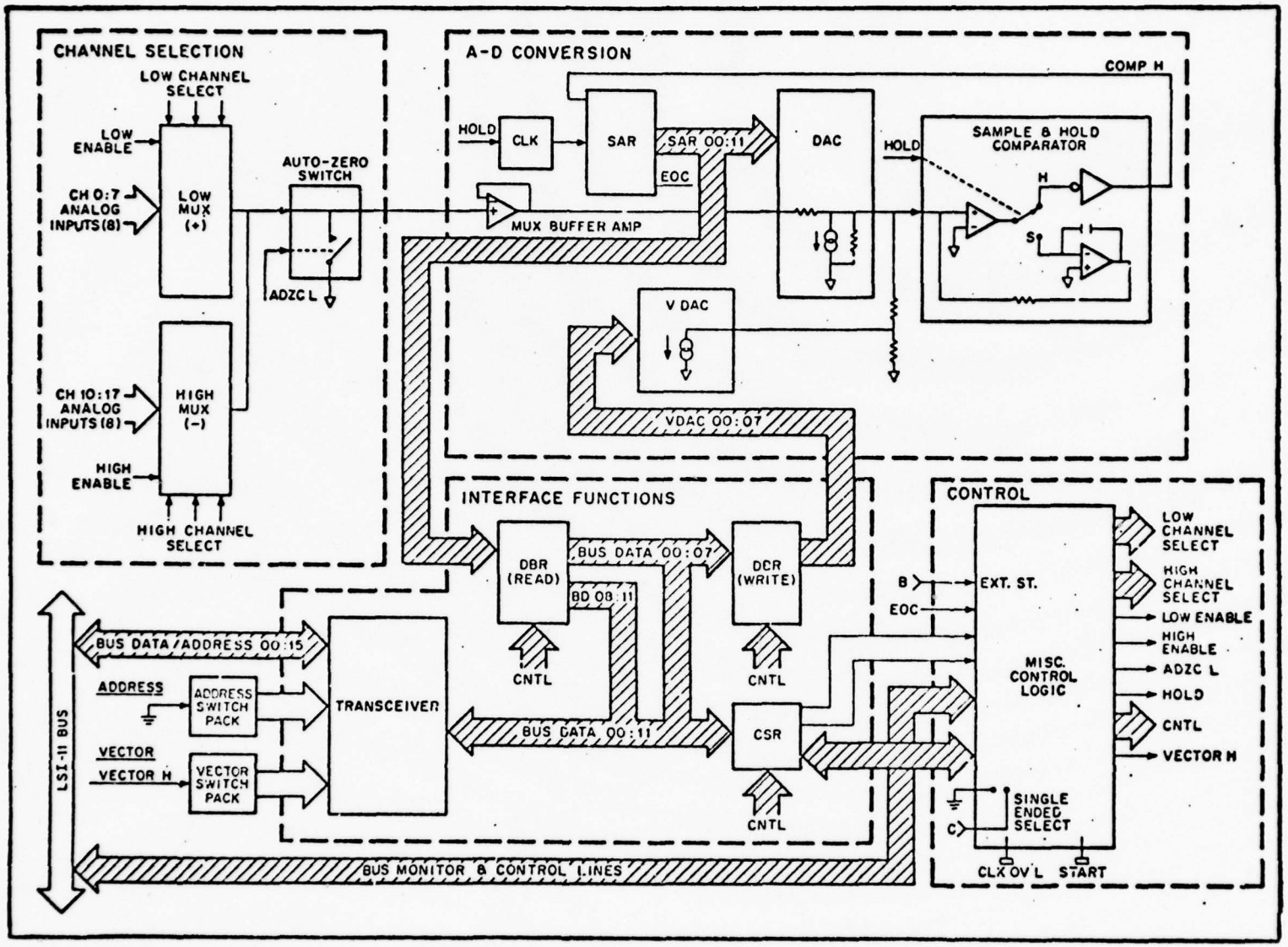


Figure 12. ADV11-A Function Block Diagram (From Ref 11:191)

This A/D converter was adequate for MIBDAPS requirements because only two analog signals needed to be digitized. Also, it was required that the conversion error should not exceed ±1% of the full scale (Ref 13). For analog signals of 0-5V, this requirement translates to an error of ±5mv. The one least significant bit (LSB) value for the ADV11-A is 2·5mv which means that the maximum error that can be introduced is ±2·5mv, which is within the requirement.

The main features of ADV11-A are:

- Analog input full scale range is 10·24V bipolar (-5·12V to +5·12V).
- Logic input voltages are:

low (logic "0") = 0·0 to +0·7V

high (logic "1") = +2V to +5V

- Resolution of ADV11-A 12 12-bit binary weighted. The format of the digitized signal is parallel offset binary right justified.
- The timing requirements are:

a. External start should be a low-level pulse, 50 nsec minimum duration and 10 µsec maximum duration. Conversion starts on the leading edge.

b. Conversion time is 16 times the clock period. The maximum clock frequency is 100 KHZ.

- Power requirements are:

+5V ±5% at 2·ØA maximum current

+12V ±5% at 450mA maximum current

KWV11-A Programmable Real Time Clock (Ref 11:211-212). The functional diagram for the real time clock is shown in Figure 13. This programmable clock/counter provides a variety of means for determining time

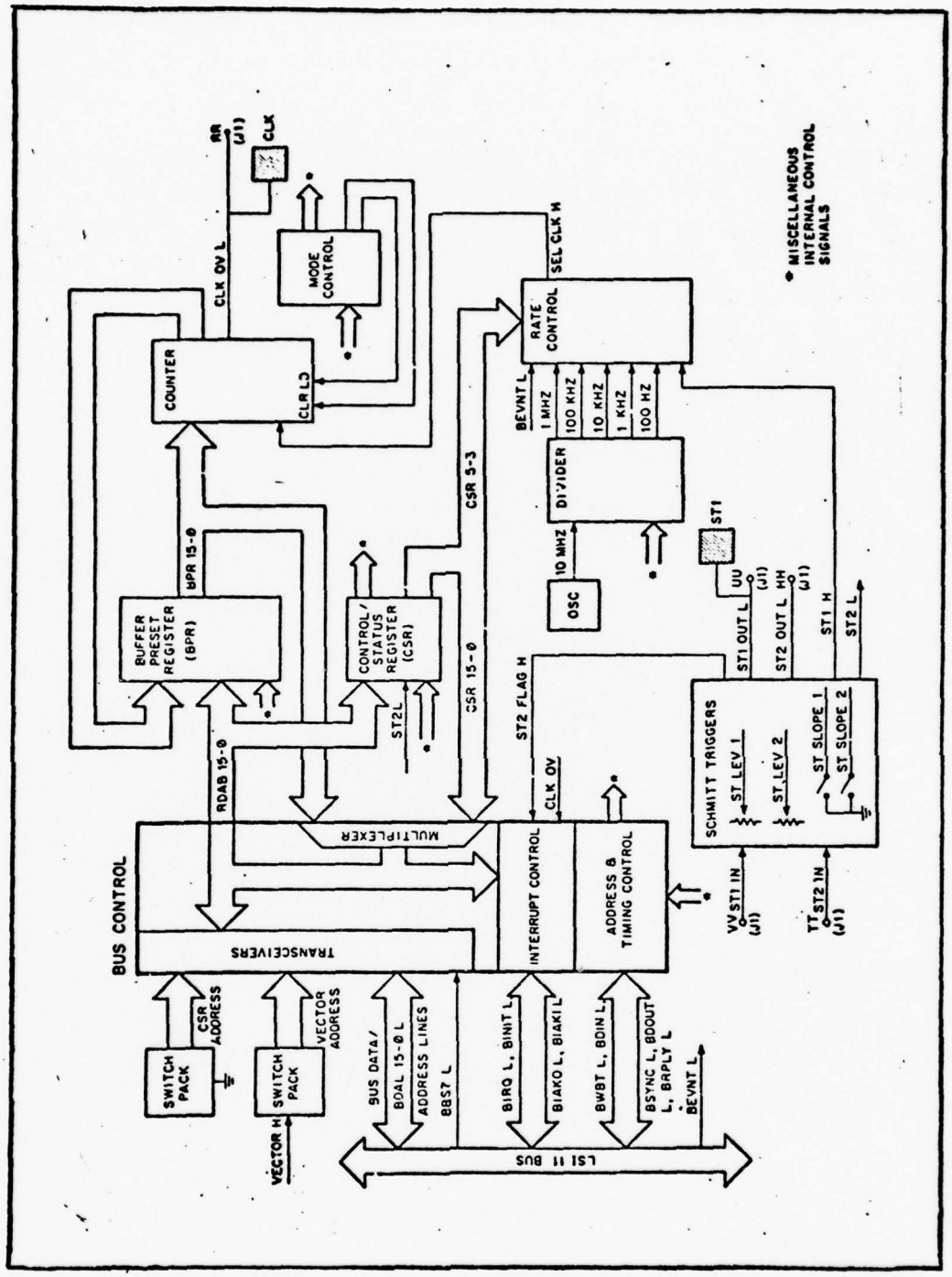


Figure 13. KWV11-A Real Time Clock (From Ref 11:213)

intervals or counting levels. It can be used to generate interrupts to the LSI-11 processor at predetermined intervals to synchronize the processor to external events or to measure time intervals or establish programmed ratios between input and output events. It can also be used to start the ADV11A analog-to-digital converter either by clock counter overflow or by the firing of a Schmitt trigger circuit.

The clock counter has a resolution of 16 bits and can be driven from any five internal crystal-controlled frequencies (100 HZ to 1 MHZ). It can also be driven from a line frequency input or from a Schmitt trigger fired by an external clock input. The performance specifications for the KWV11-A are:

- Internal Clock

The internal clock has a frequency accuracy of ±0.01%. The base frequency is 10 MHZ, and it is divided into five selectable rates (1 MHZ, 100 KHZ, 10 KHZ, 1 KHZ, and 100 HZ).

- External Clock

The external clock can either be the line frequency input or a clock generator having the desired frequency.

- Output Signals

a. A clock overflow signal occurs whenever a certain timing event is completed. The characteristics of this signal are: 500 nsec low asserted level pulse which is TTL compatible.

b. Two trigger outputs, Schmitt trigger 1 and Schmitt trigger 2, are also available for either being used as a trigger for the ADV11-A (A/D converter board) or for any other external device. The characteristics are the same as that for the clock overflow signal.

DLV11 Serial Line Unit (Ref 11:147-148). The DLV11 is the basic interface module used for connecting asynchronons serial line devices to the LSI-11 bus. The functional diagram for the DLV11 is shown in Figure 14. The salient features of the DVL11 unit are:

- Either an EIA RS232 or a 20mA current loop device can be interfaced to an LSI-11 bus.

- There are 13 baud rates that can be selected: 50, 75, 110, 134·5, 150, 200, 300, 600, 1200, 1800, 2400, 4800, and 9600.

- The LSI-11 bus interface and control logic for interrupt processing and vector generation are also available on the board.

- The control/status register (CSR) and data buffer register (DBR) for programmed or interrupt driven transfer of data are available.

Software Development Tools. One of the reasons why the LSI-11 microcomputer system was chosen for mechanization of MIBDAPS was because it is software compatible with the PDP-11 computer family. The following systems that were available on Wright-Patterson Air Force Base could, therefore, be used for software development:

- The Department of Electrical Engineering at the Air Force Institute of Technology (AFIT), Wright-Patterson Air Force Base, Ohio, has a PDP-11/20 along with a paper tape software system. (A single floppy disk drive has been added; therefore, the RXV11 operating system could also be used.)

- The AFAL has a cross-assembler for PDP-11 assembly language on its DEC-10 computer system. Programs could be created on the DEC-10 system using text editors. The PDP-11 cross-assembler (MACY11) can then be used to create absolute loader formatted paper tapes. There is, however, no facility to simulate execution of these programs.

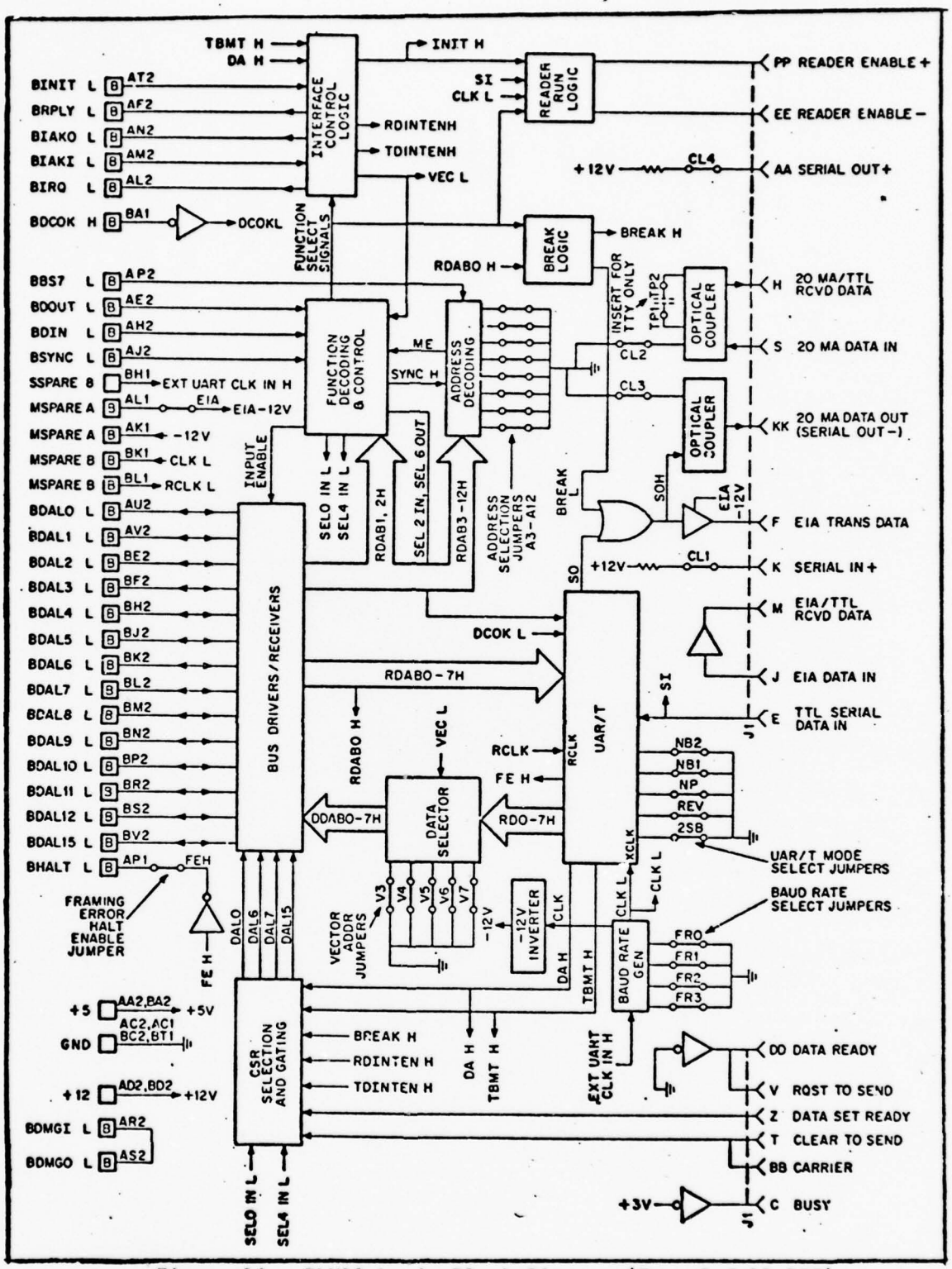


Figure 14. DLV11 Logic Block Diagram (From Ref 11:150)

- The AFML has a PDP-11/03 with dual floppy disk drive and an RXV11 operating system. This system uses an LSI-11 microcomputer and all interface boards. The RXV11 operating system has a text editor, macro assembler, linker, FORTRAN compiler, and peripheral interchange program (PIP) available on a floppy disk. This system was suitable not only for developing and debugging programs, but also for real time program execution.

The PDP-11/03 system at AFML was eventually used for the development of software for MIBDAPS.

Analysis and Design of MIBDAPS

The decisions about the implementation of different functional modules of MIBDAPS, either in software or hardware, were guided by the following factors:

- Capabilities and limitations of the KDF-11 processor module along with 4K words of core memory used as the microcomputer system.

- Capabilities and limitations of support hardware modules, in particular ADV11-A (ADC) and KWV11 (RTC).

- Speed of software execution.

The KDF-11 processor along with 4K words of core memory (subsequently referred to as "microcomputer system") would be used to implement the controlling and processing functions of MIBDAPS in software. The ADC board (ADV11-A) and RTC (KWV11) along with a system clock circuit (which would be designed and fabricated) were visualized as the only hardware needed to implement the data acquisition, signal processing, and synchronization functions of MIBDAPS. Keeping these two factors as guidelines, another in-depth study of the system requirement model developed in Chapter II was carried out. SADT was employed to carry out a functional analysis

of the system and a new activity model, keeping implementation in view, was developed. The node index for this activity model is given in Table III. It should be noted that this time the mechanization (M) for each node is also defined as hardware or software, or both. Software implementation implies programs which would be executed on the microcomputer system. Hardware implementation implies use of ADC, RTC, or an external system clock.

TABLE III

Node Index for New Activity Model

A-Ø	Acquire and Process Data (same as in Chap II)
AØ	Acquire and Process Data
A1	Control All Activity
A13	Switch Power to System
A14	Stop Execution
A15	Start Execution
A16	Start Acquisition and Processing of Data
A162	Initiate Data Acquisition
A163	Initiate Signal Processing
A2	Acquire Data
A3	Process Data
A32	Compute Fourier Transform
A33	Compute Power Spectrum
A34	Compute Damage Factor
A35	Update CDF Array

The explanation of each node along with data structures used and input/output signal specifications follows.

Node AØ, Acquire and Process Data (Figure 15). The difference between node AØ as compared with node AØ* (Figure 4) is the absence of module A4 (Store Data). This difference has come about because the Process Data Module will perform the task of storing and updating result that are stored. The various activities taking place in this node are listed below.

- The Control All Activity module performs the function of managing the control and execution of all other activities. It also provides the DC supply to all hardware modules.

- The Acquire Data module continuously samples and digitizes the input analog signals, analog vibration signal and analog temperature signal. Whenever a set of data values have been digitized, it sets the Acquisition Done flag.

- The Process Data module takes in the input raw data and processes it to give an array of damage factor versus frequency. This array is added on to one of the cumulative data factor versus frequency arrays. There are 11 CDF versus frequency arrays, one for each expected temperature intervals (Ref Chap II).

The input/output and control signal specifications are:

- Input Signal

  a. Analog vibration signal (I1): 0-5 volts.

  b. Analog temperature signal (I2): 0-5 volts.

  c. AC power (I3): 110 volts, 60 cycles.

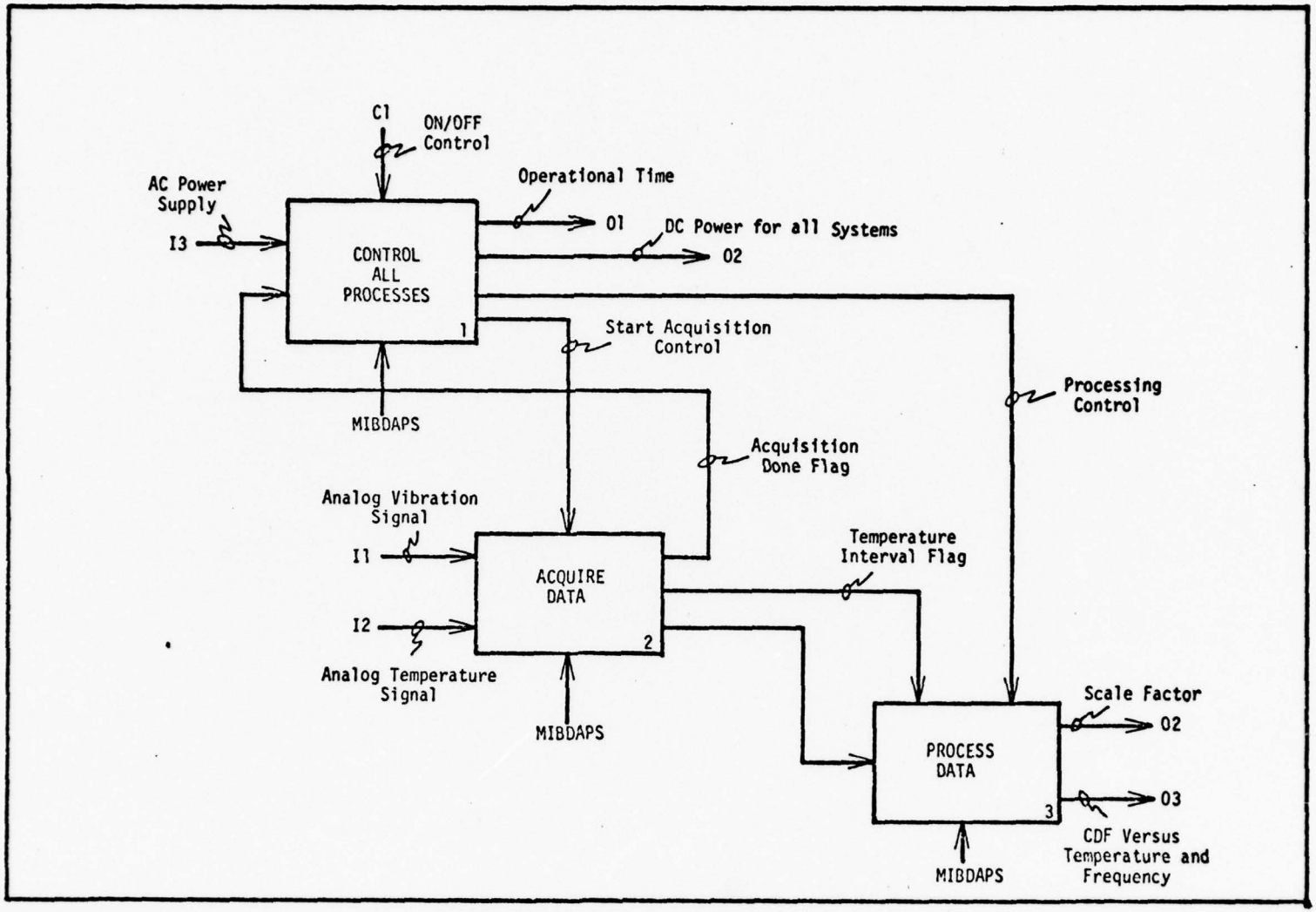


Figure 15. Node AØ, Acquire and Process Data

- Output Signal

a. Total Operational Time (01): A display of operational time in hours and minutes.

b. DC Power to System (02):

c. Cumulative Scale Factor (03): This is a 16-bit octal number which gives the scale factor for the output arrays.

d. CDF Versus Frequency and Temperature Arrays (03): These are 13-word long arrays having the data structure as shown in Figure 16.

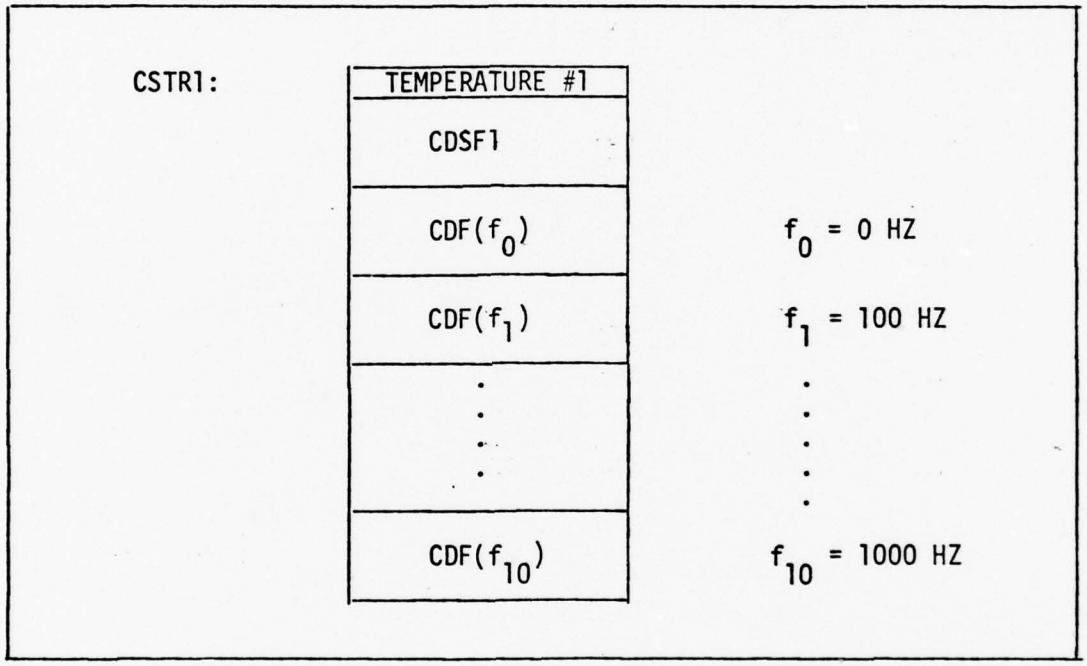


Figure 16. Data Structure for CDF Vs Frequency and Temperature Array

- Control Signal

System ON/OFF Control (C1): Once the system is switched ON and the AC power is available, the system starts functioning.

Node A1, Control All Activity (Figure 17). This node performs the activities necessary to control the synchronization of all other activities

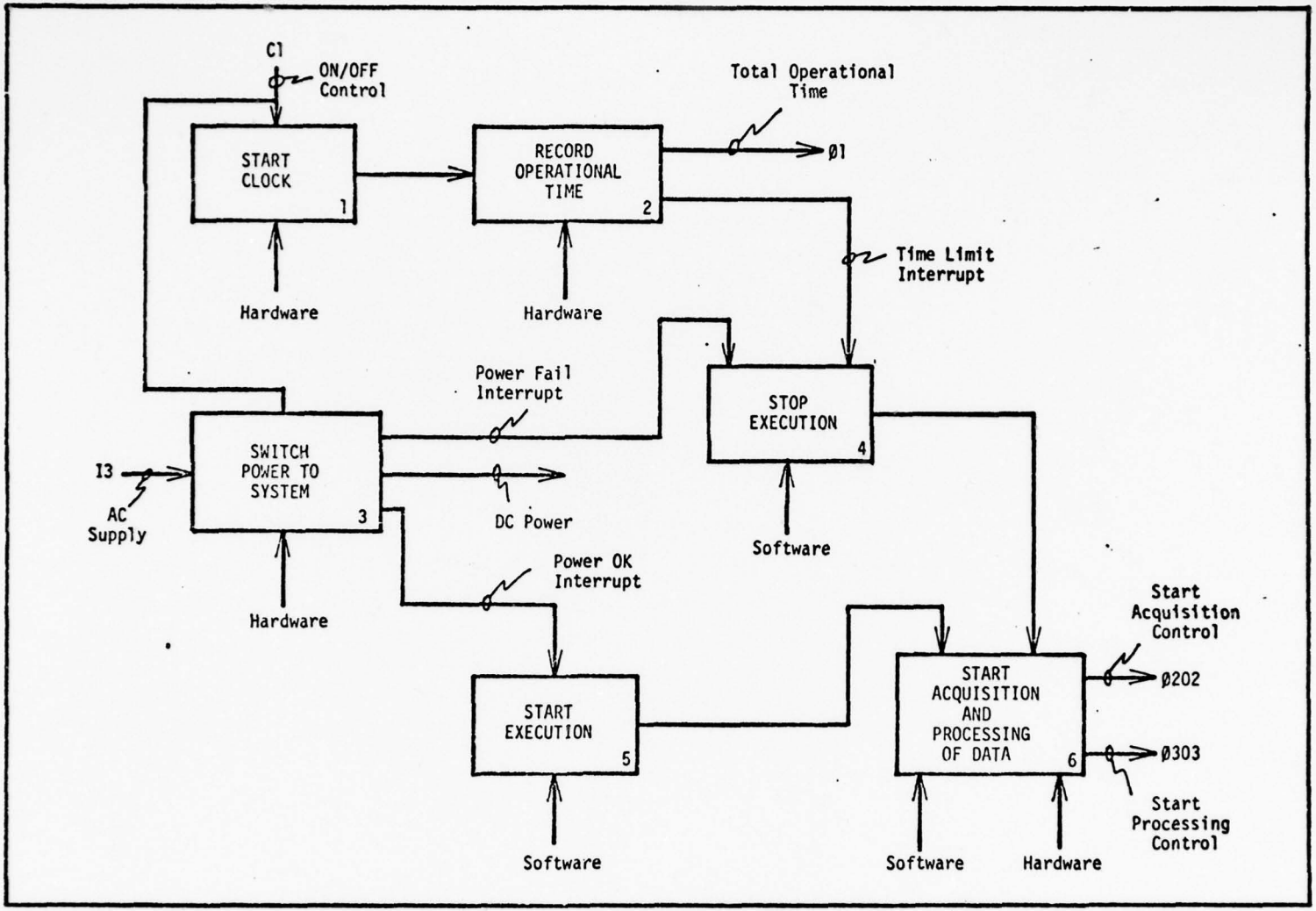


Figure 17. Node A1, Control All Activity

of MIBDAPS. The mechanization of different activities in this node are either done by hardware or software or a combination of both. It is assumed that the LSI-11 computer is used to execute the software; however, hardware mechanization includes various other modules which are being interfaced to the microcomputer. The various activities performed in this node are listed below.

- The DC power required by different modules of the LSI-11 microcomputer system is provided by this node. The 110-volt AC supply is converted to +5V and +12V DC supplies. Whenever the system's ON/OFF switch is selected to the ON position, the DC supply is switched ON. This is done by a Switch Power to System module. This module also generates an interrupt in case the DC power is within an acceptable limit. In case of a power failure, an interrupt is also generated.

- Total operational time for the system is also computed by the Record Operational Time module. In case the operational time reaches the selected time limit (15 to 20 hours), an interrupt is generated.

- The Start Execution module, after receiving the Power OK interrupt, initializes the system and sends a signal to Start Acquisition and Processing of Data.

- The Stop Execution module stops execution of the system function in case the following interrupts occur:

Power Fail Interrupt

Time Limit Interrupt

- The Start Acquisition and Processing of Data module initializes different variables which are used by the software. It then signals nodes A2 and A3 to start acquisition and processing of data.

Specifications for the input/output and control signals are:

- Input Signal

AC Supply (I3): 110 volts, 60 HZ.

- Output Signal

a. Total Operational Time (01): Four 4-bit BCD numbers.

b. Start Acquisition Control (Ø202): A software-generated command. The control/status register of the ADV11-A A/D converter (ADC) is loaded with the appropriate value to enable the ADC. This initiates the data acquisition cycle.

c. Start Processing Control (Ø303): This is also a software-generated command, which activates the processing modules for input data processing.

- Control Signal

System ON/OFF Control (C1): A signal which initiates the operation of the system. A logic "1" value is available when the system is turned ON and logic "0" value when the system is turned OFF.

Node A13, Switch Power to the System (Figure 18). This node carries out the following activities:

- Converts the AC power supply to +5 volts and +12 volts DC.

- Generates an interrupt to the microcomputer when DC power is available.

- Detects a power failure and generates a Power Fail interrupt to the microcomputer.

The input and output signal specifications are:

- Input Signal

AC Supply (I3): This is the only input to this node. The system operates on 110V AC, 60 HZ.

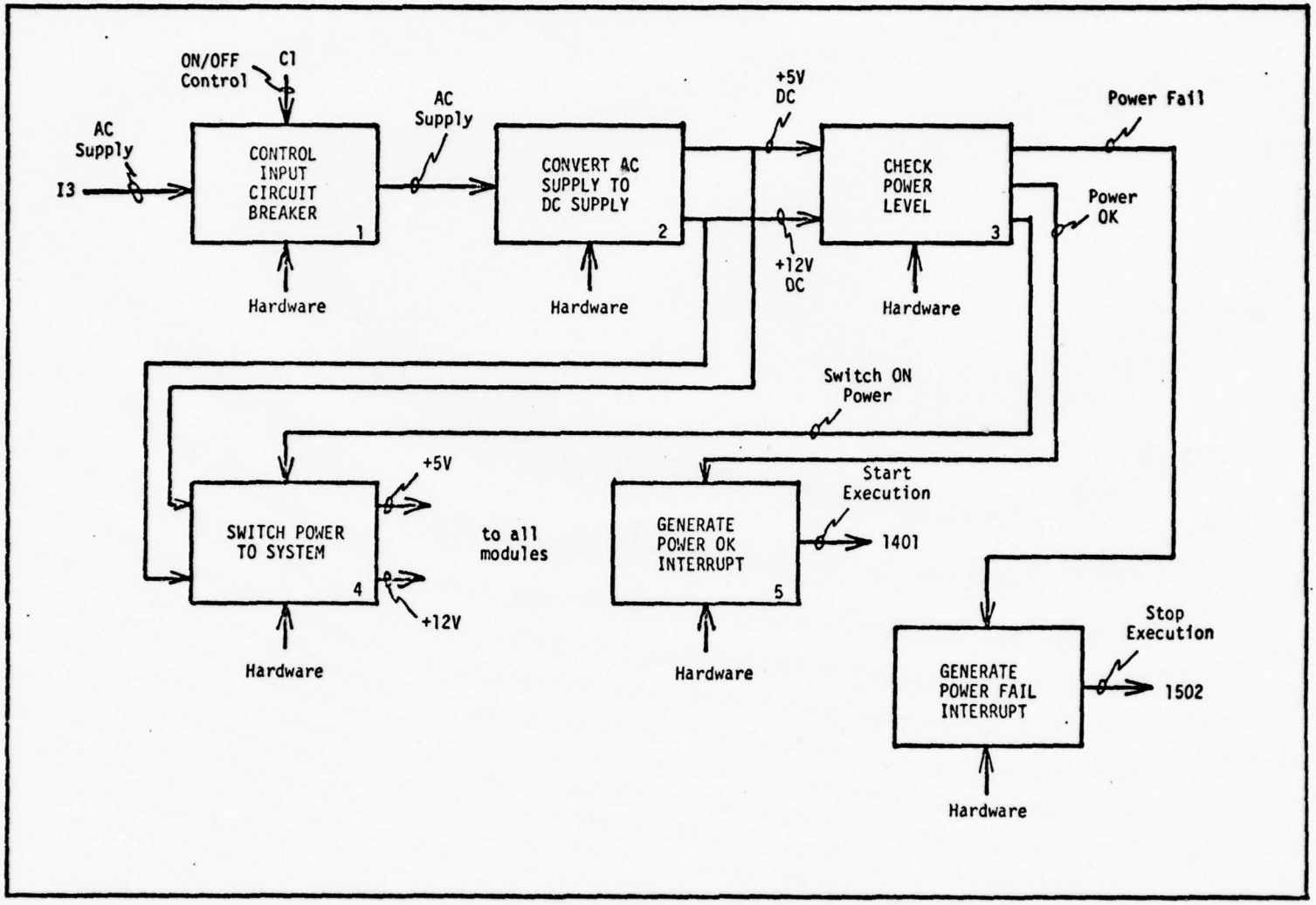


Figure 18. Node A13, Switch Power to the System

- Output Signal

a. DC Power Supply: The +5V and +12V DC power supply is routed to all modules of MIBDAPS.

b. Start Execution (1401): A software-generated signal which is used to start execution of the program. Bit-Ø of variable STEXEC is set to initiate data acquisition and processing.

c. Power Fail Interrupt (1501): A software-generated signal which stops execution of the program. Bit-Ø of variable STPXC is set to stop execution of the program.

Node A14, Stop Execution (Figure 19). This node receives the Power Fail interrupt; it then accesses the vector interrupt address for the service routine. The service routine is executed and bit-Ø of the variable STPXC is set to a logic "1." This variable is sent as an output signal to stop execution of the system execution.

Node A15, Start Execution (Figure 20). This node receives the Power OK interrupt; it then accesses the vector interrupt address for the service routine. The service routine is executed and bit-Ø STEXEC switch is set to logic "1." This variable is sent as to the GO Command to start system execution.

Node A16, Start Acquisition and Processing of Data (Figure 21). Once the system has been turned ON and DC power is made available to the system, this node starts execution of the data acquisition and data processing functions of MIBDAPS.

The Stop Execution Command (14C2) and Start Execution Command (14C1) are used to start and halt the execution of data activity. The input/output signal specifications are:

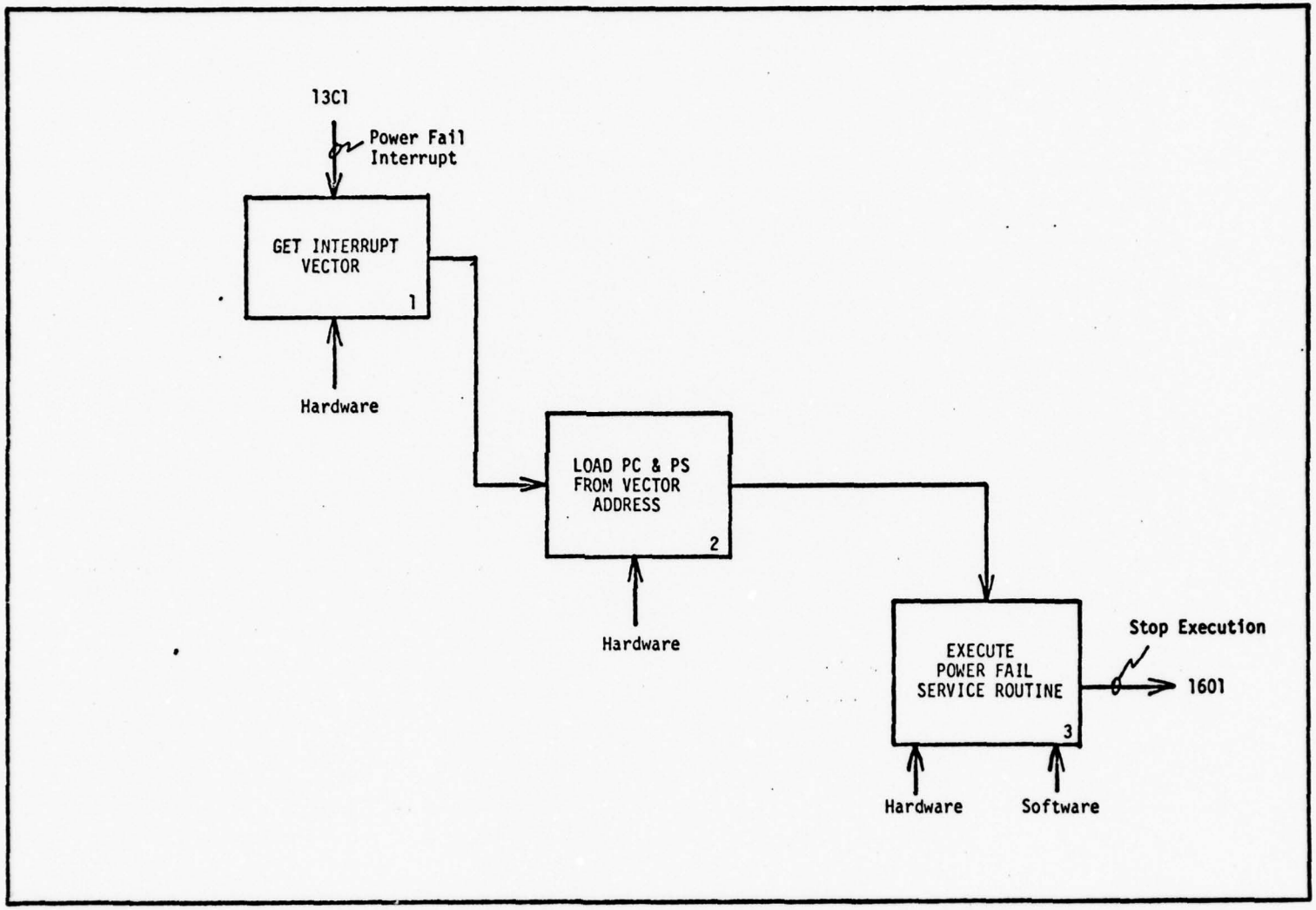


Figure 19. Node A14, Stop Execution

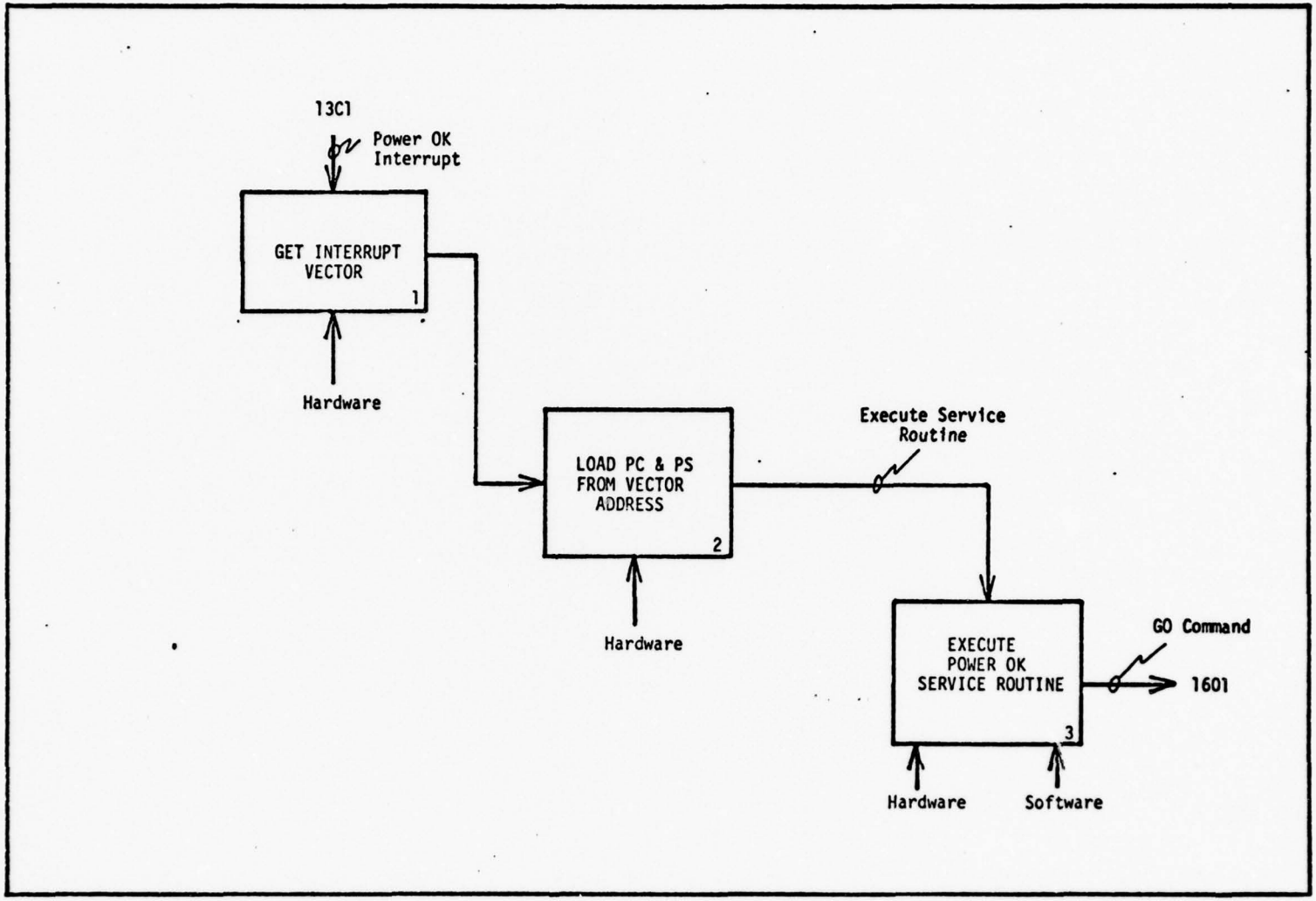


Figure 20. Node A15, Start Execution

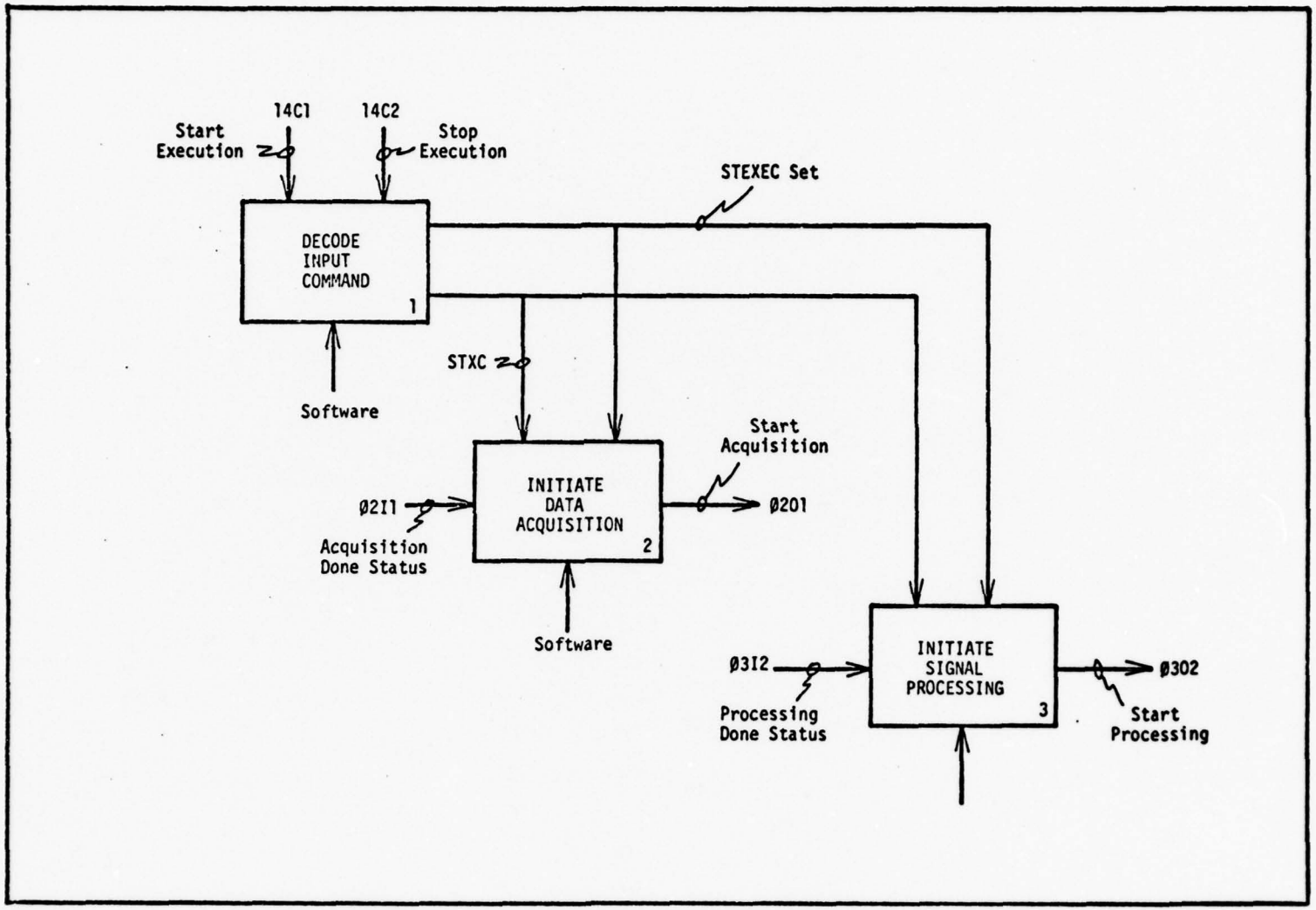


Figure 21. Node A16, Start Acquisition and Processing of Data

- Input Signal

a. Acquisition Done Status (Ø2I1): Whenever the acquisition of n-data points is complete, the variable ADFLG is set to logic "1." This variable along with ADCNT (number of data points to be converted) and CDATA (starting address of data buffer where data is to store) are passed to the Initiate Data Acquisition module. All of these variables need to be initialized whenever data acquisition has to be reinitiated.

b. Processing Done Status (Ø3I2): The following variables are passed to the Initiate Processing module for initialization: DATA - Contains starting address of data to be processed. SFACTR, CUMSF1, and CDSFi (i=1, 2...) - Scale factors used in node A3, which are explained later; however, each has to be cleared before processing.

- Output Signal

a. Start Acquisition (Ø2O1): This signal is generated when the control/status register (ADCSR) for the A/D converter is loaded with the appropriate bit pattern. Once the ADCSR is loaded, the acquisition of data starts.

b. Start Processing (Ø3O2): This is a software-generated command, and it initiates the processing of data.

- Control Signal

a. Start Execution (14C1): This is a software switch, STEXEC, which is set to logic "1" whenever DC power is available and the system has been initialized.

b. Stop Execution (14C2): This is also a software switch, STPXC, which is set to logic "1" whenever a power failure occurs.

Node 162, Initiate Data Acquisition (Figure 22). This node shows explicitly the functional decomposition of the module Initiate Data

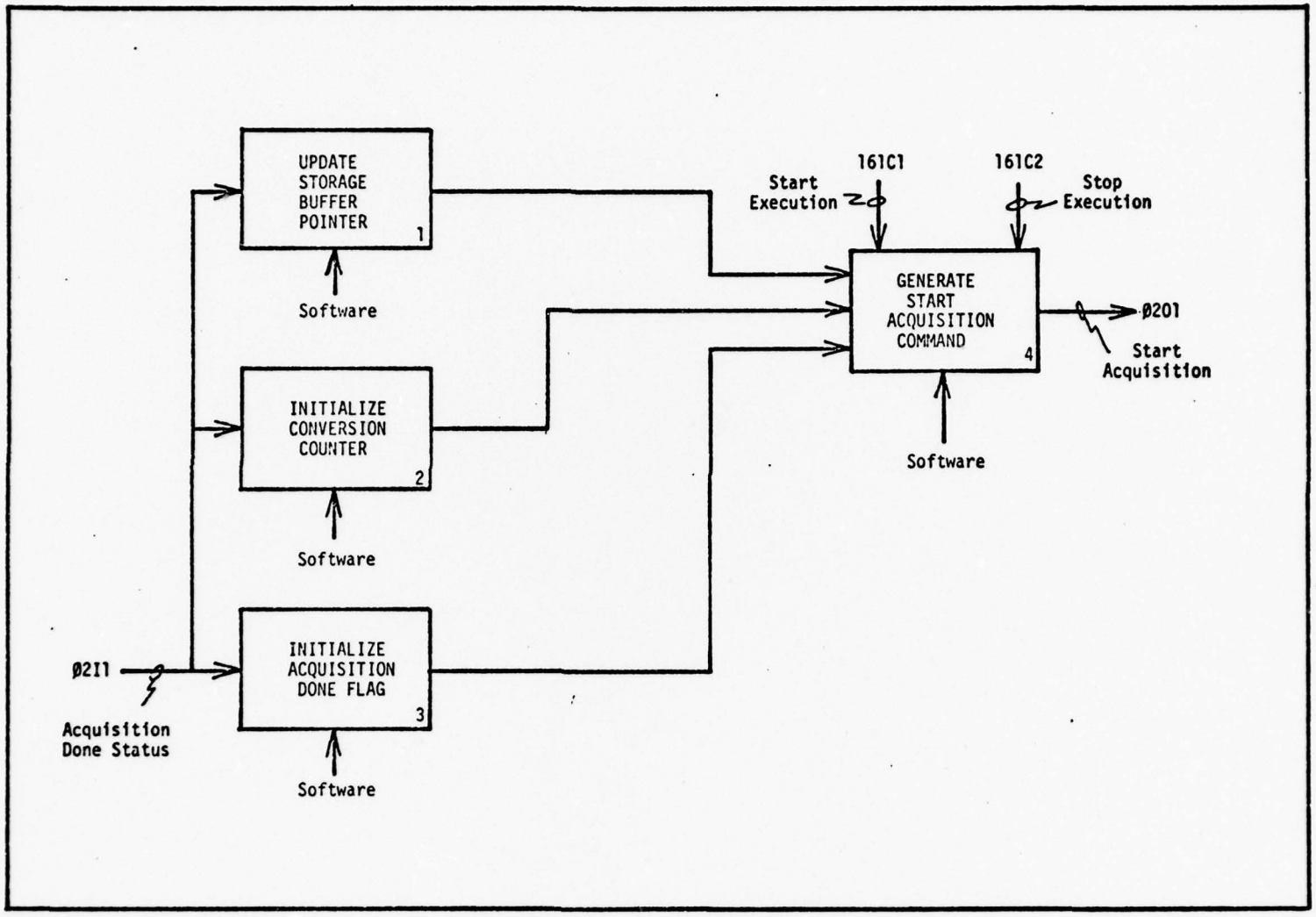


Figure 22. Node A162, Initiate Data Acquisition

Acquisition of node A16. The STEXEC switch (161C1) and STPXC switch (161C2) control the activity in this node. The input signal Ø2I1—CDATA, ADCNT, and ADFLG—is initialized by modules 1, 2, and 3, respectively. The Generate Start Acquisition Command loads the ADCSR with appropriate bits to initiate acquisition of data by ADC, only after all of the above mentioned switches are initialized and STEXEC is set to logic "1."

Node 163, Initiate Signal Processing (Figure 23). This node shows the functional decomposition of the Initiate Signal Processing module of node A16. The activities in this node are similar to node A162. The input signal Processing Done Status (Ø321) and output signal Start Processing (Ø301) are different. The input includes SFACTR, CUMSF1, CDSFi SPCTRA, and DATA switches. These are initialized in this node. The Start Processing Command generated in this node starts the execution of the processing of acquired data.

Node A2, Acquire Data (Figure 24). This node performs the activity of acquiring data and storing it in temporary storage. Once the acquisition of N-data points is complete, the acquisition is stopped and the ADFLG switch is set to logic "1." When the analog temperature signal is selected for acquisition, the temperature interval flag, TFLG, is loaded with a number from Ø to 11. The temperature interval number that is loaded in TFLG depends on the value of the temperature signal at the time of conversion. The input and output signal specifications are:

- Input Signal
  a. Analog vibration signal: 0-5V.
  b. Analog temperature signal: 0-5V.

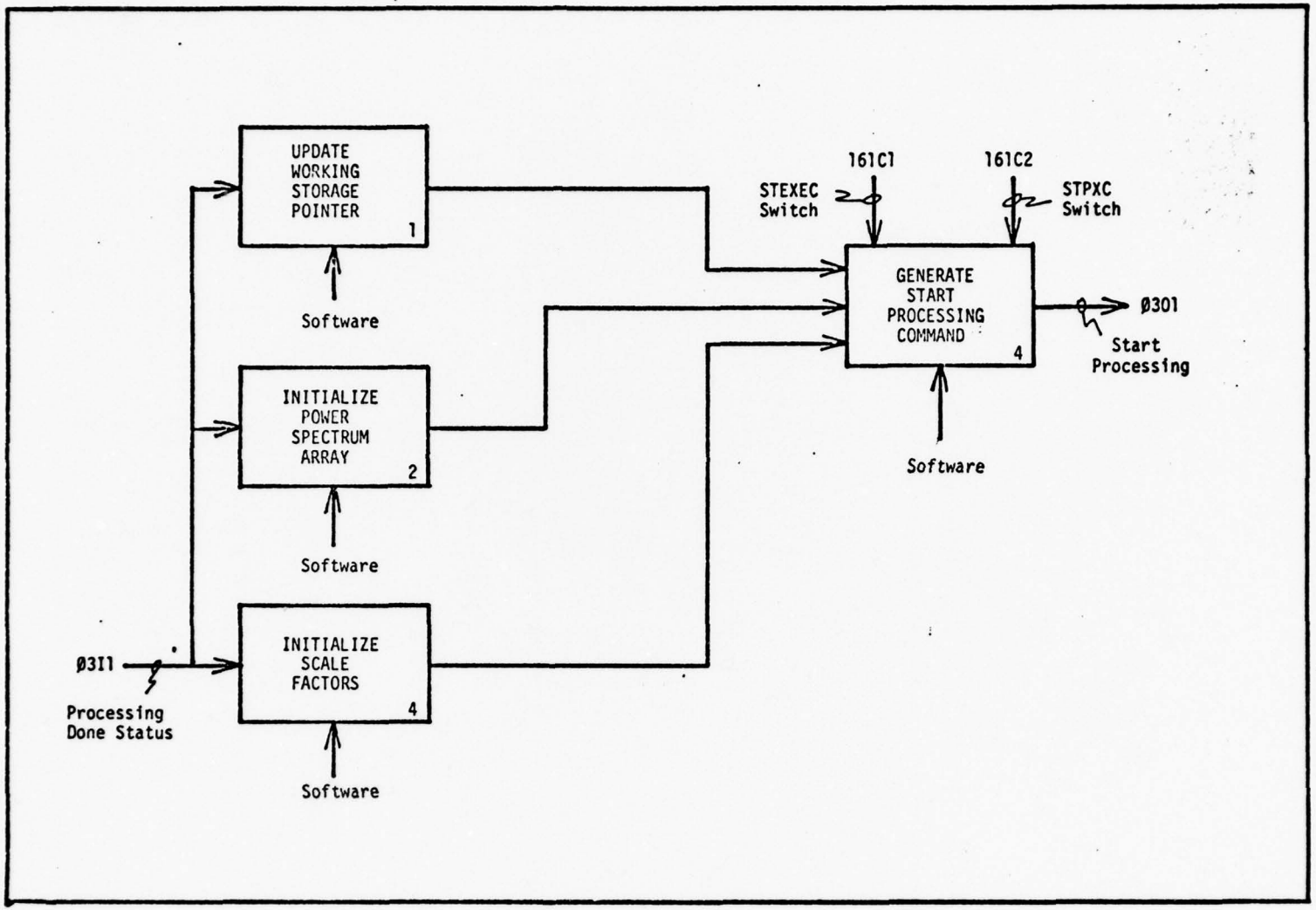


Figure 23. Node 163, Initiate Signal Processing

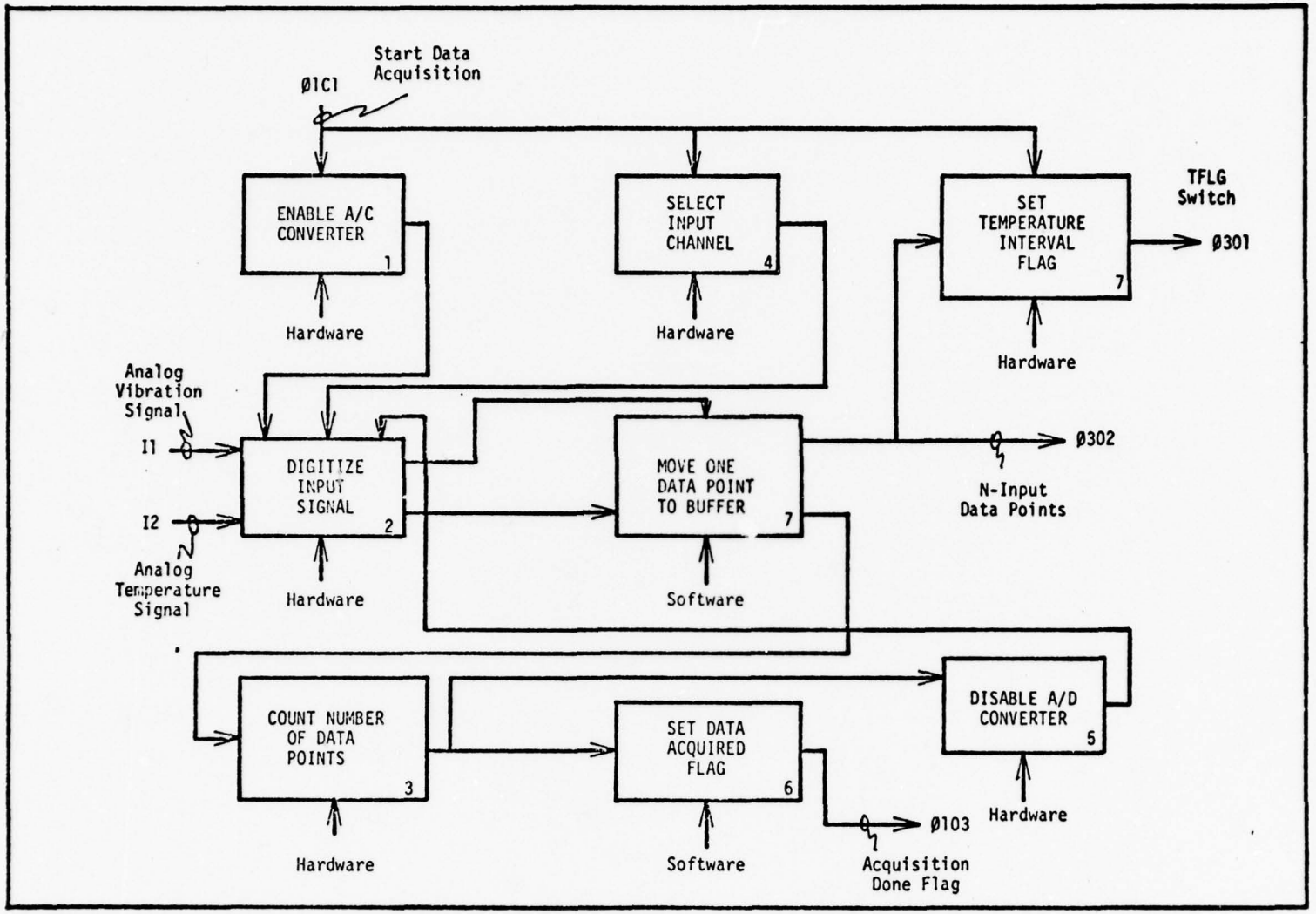


Figure 24. Node A2, Acquire Data

- Output Signal

a. Temperature Interval Flag (Ø301): A decimal number between Ø and 11 is loaded in TFLG. This number depends on the value of the temperature signal at the time of acquisition.

b. N-Input Data Points (Ø302): The data structure for the data buffer storage is given in Figure 25. CDATA is the switch which contains the starting address of the data buffer to be used. ISTR1 and ISTR2 are the starting addresses of the two buffers. Each buffer is N-words long.

c. Acquisition Done Flag (Ø103): Bit-Ø of switch ADFLG is set to logic "1" whenever data acquisition of N-points is complete.

Node A3, Process Data (Figure 26). The activities performed by this will be easier to understand if the flow of data through each module is explained. The Input Data (Ø2I1) is N-points of acquired data which are stored in a buffer area. The starting address of this buffer is available in switch DATA. The fourier transform of these N-points is computed, and the results are stored back in the same buffer area. Switch SFACTR contains the scale factor for the resulting array. The fourier transformed array serves as an input to the Computer Power Spectrum module. This module computes the power spectrum for first M-frequency components (including DC term). The output for this module is M-element power spectrum array and switch CUMSF1, which has the scale factor for this array. The M-element DF array is added to one of the existing CDF versus frequency arrays. The value of switch TFLG is used to select the final array, where the DF array is added. The data structure associated with each module of this node is explained in the subordinate nodes.

Node A32, Computer Fourier Transform (Figure 27). This node takes the input data from the buffer storage, computes the fourier transform

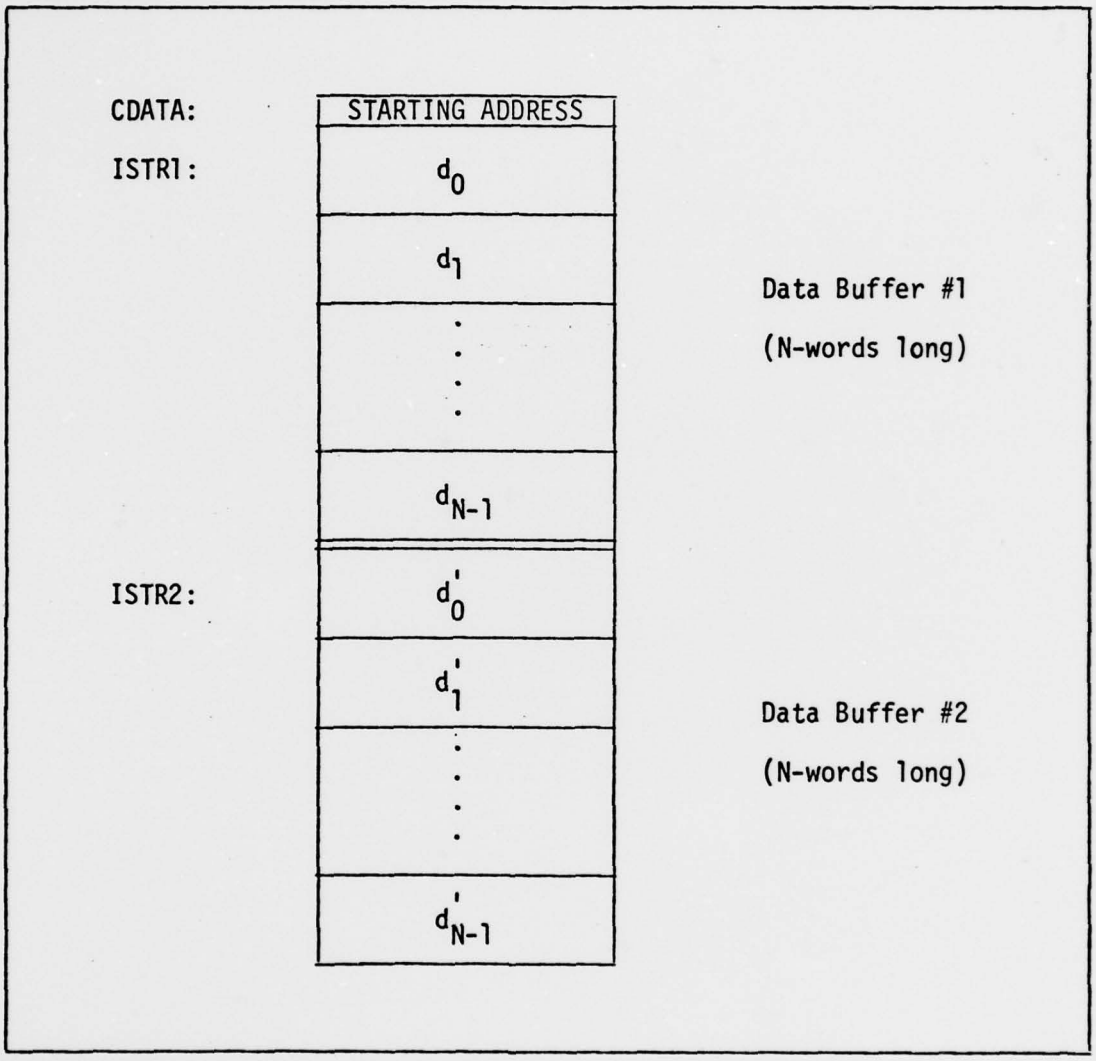


Figure 25. Data Structure for Buffer Storage

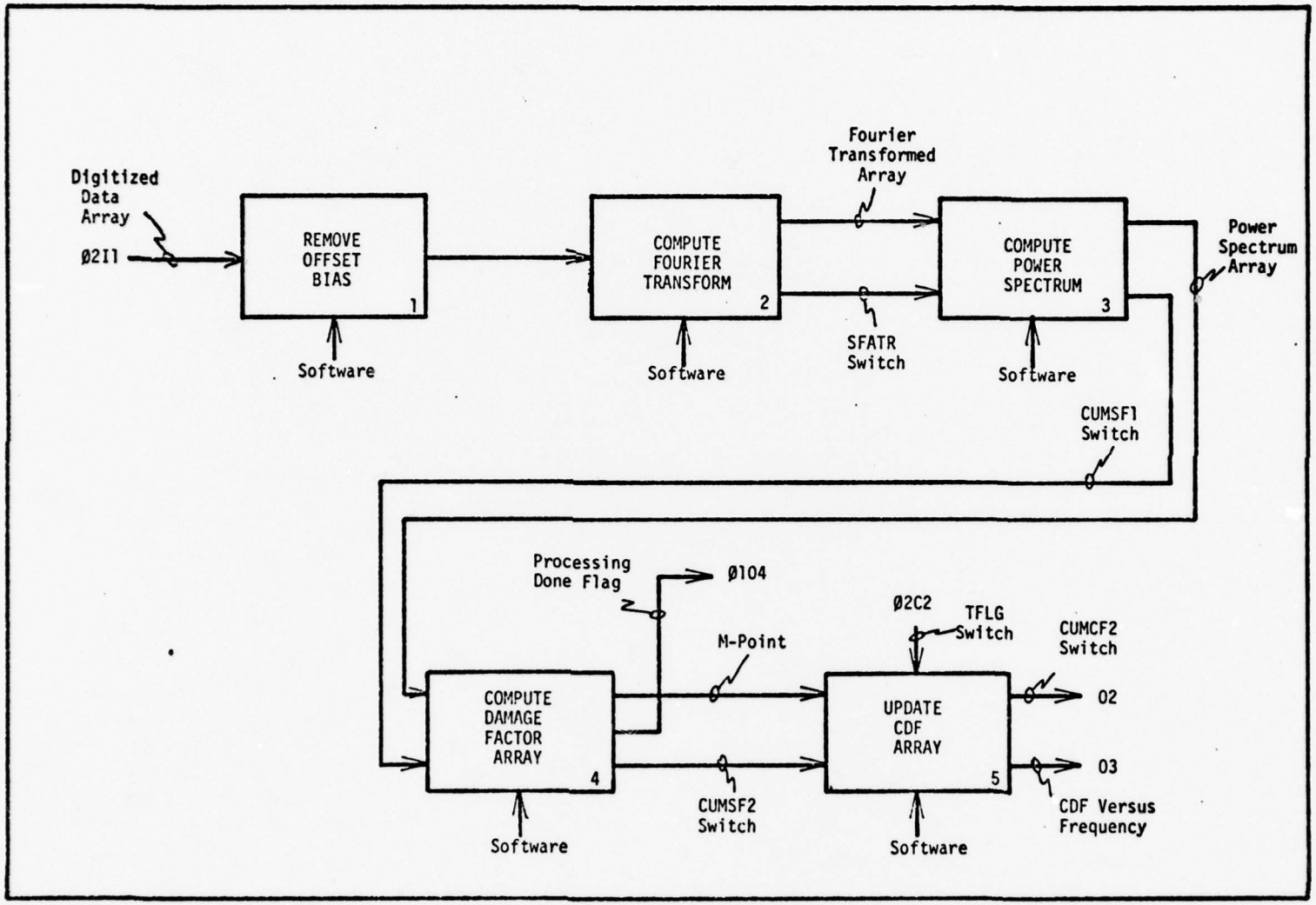


Figure 26. Node A3, Process Data

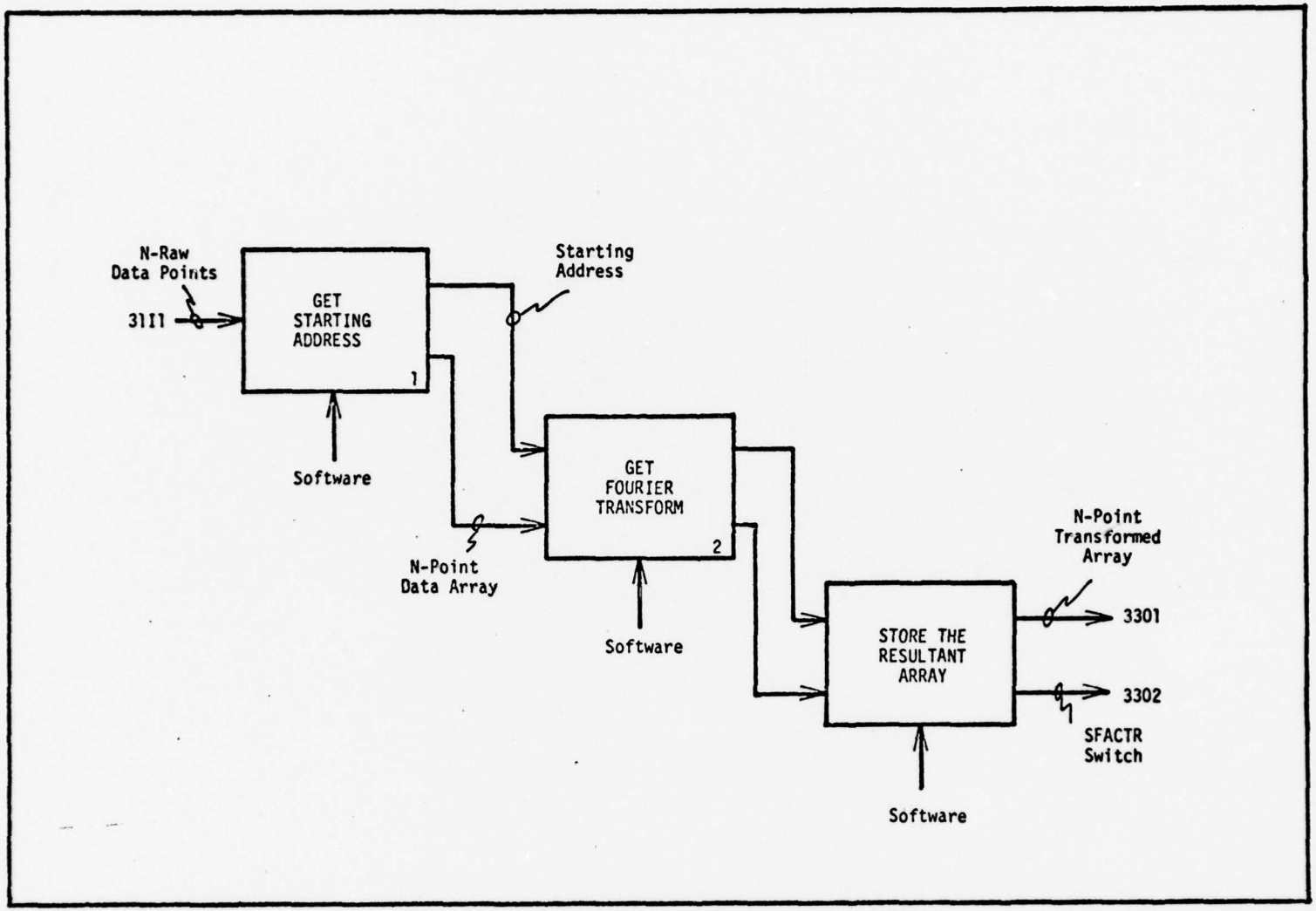


Figure 27. Node A32, Computer Fourier Transform

of the N-data points, and stores the results in the original data buffer array. The input/output data structure is shown in Figure 28. Switch DATA contains the starting address for the input/output data buffer area. Each data buffer area is N-words long.

Node A33, Computer Power Spectrum (Figure 29). This node takes the N-point transform array and computes the power spectrum for first M-frequency components. The resultant M-point array is stored in another temporary data storage area whose starting address is stored in switch SPCTRA. The input/output data structure is shown in Figure 30.

Node A34, Computer Damage Factor (Figure 31). The M-point power spectrum array is input to this node (33I1). It takes the starting address from the SPCTRA switch and computes the damage factor for each M-data points in the input array. The resultant DF array is passed on to Node A35.

Node A35, Update CDF Array (Figure 32). This node takes the damage factors computed by node A34 and updates one of the 11 CDF versus frequency arrays. Switch TFLG (Ø2C1) is used to select the starting address of the output array which corresponds to the correct temperature interval. The output data structure for CDF versus frequency arrays is given in Figure 33.

Summary

In this chapter, the selection of a microcomputer system to implement MIBDAPS was ~~made~~ accomplished. Based on the selected microcomputer system, the partitioning of MIBDAPS into hardware or software realizable modules was performed. A detailed design of MIBDAPS, including mechanization of different modules, input/output data structures, and breakdown of different modules into primitives, was carried out using SADT.

FLOW INTO NEXT CHAPTER

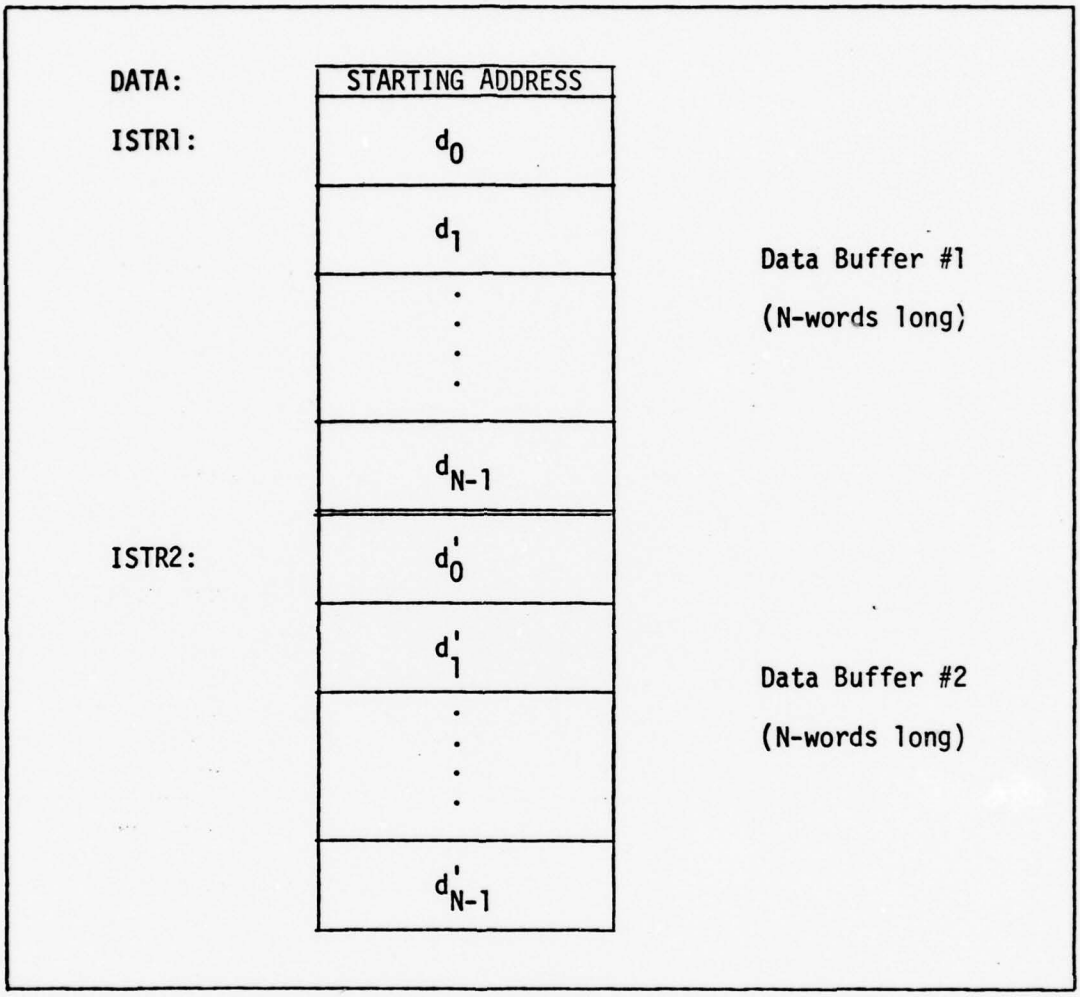


Figure 28. Input/Output Data Storage for Node A32

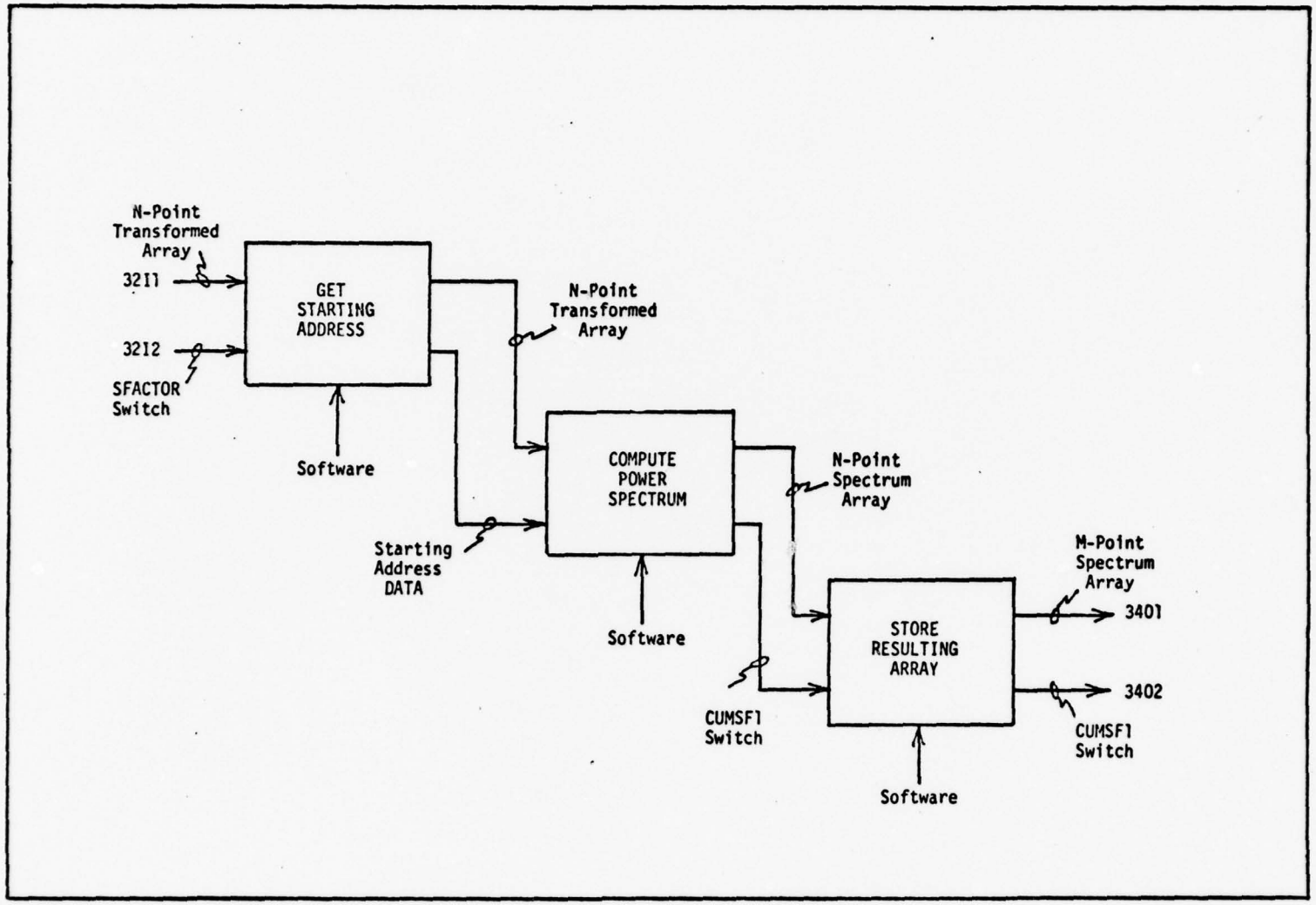


Figure 29. Node A33, Computer Power Spectrum

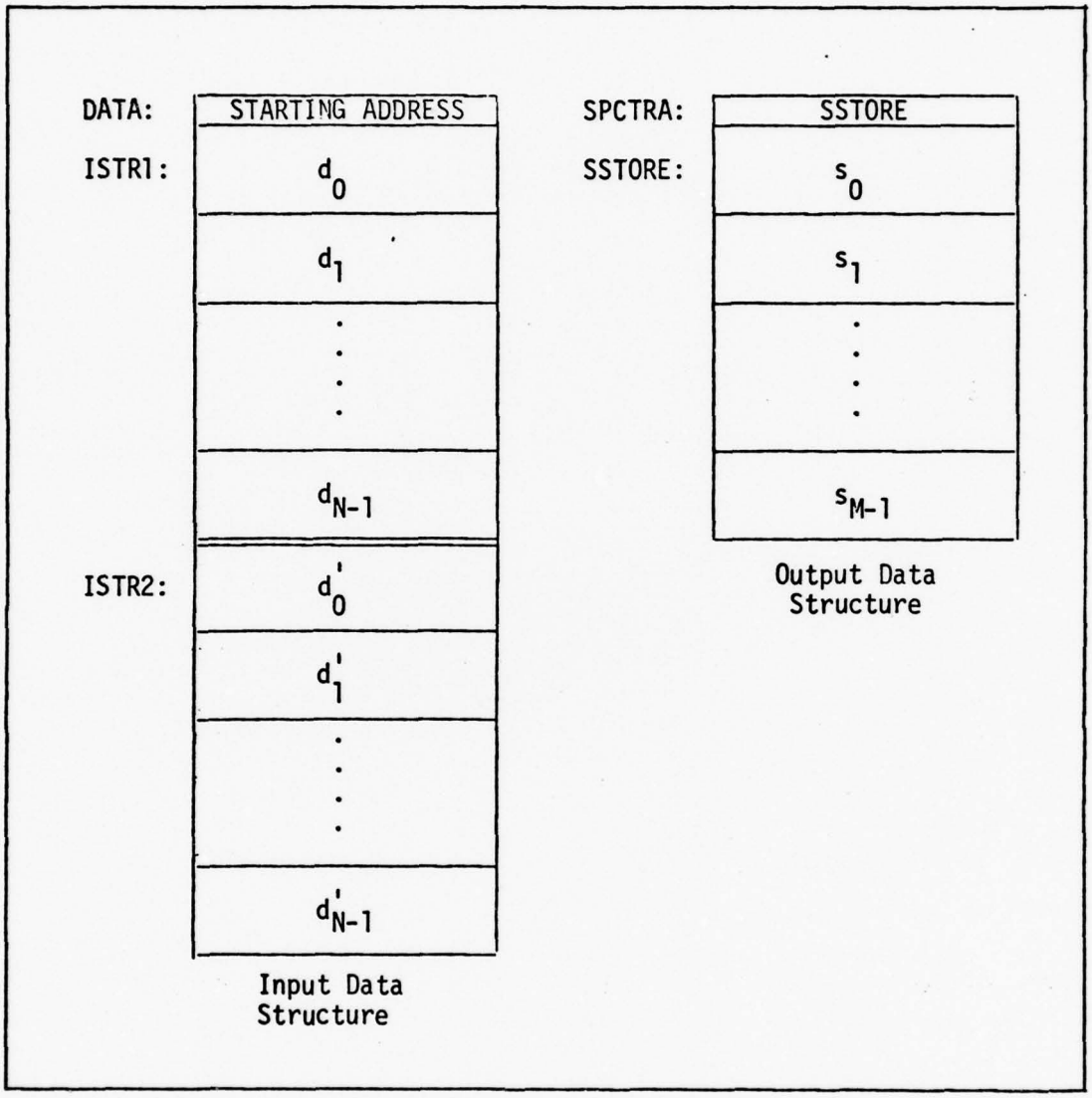


Figure 30. Input/Output Data Structure for Node A33.

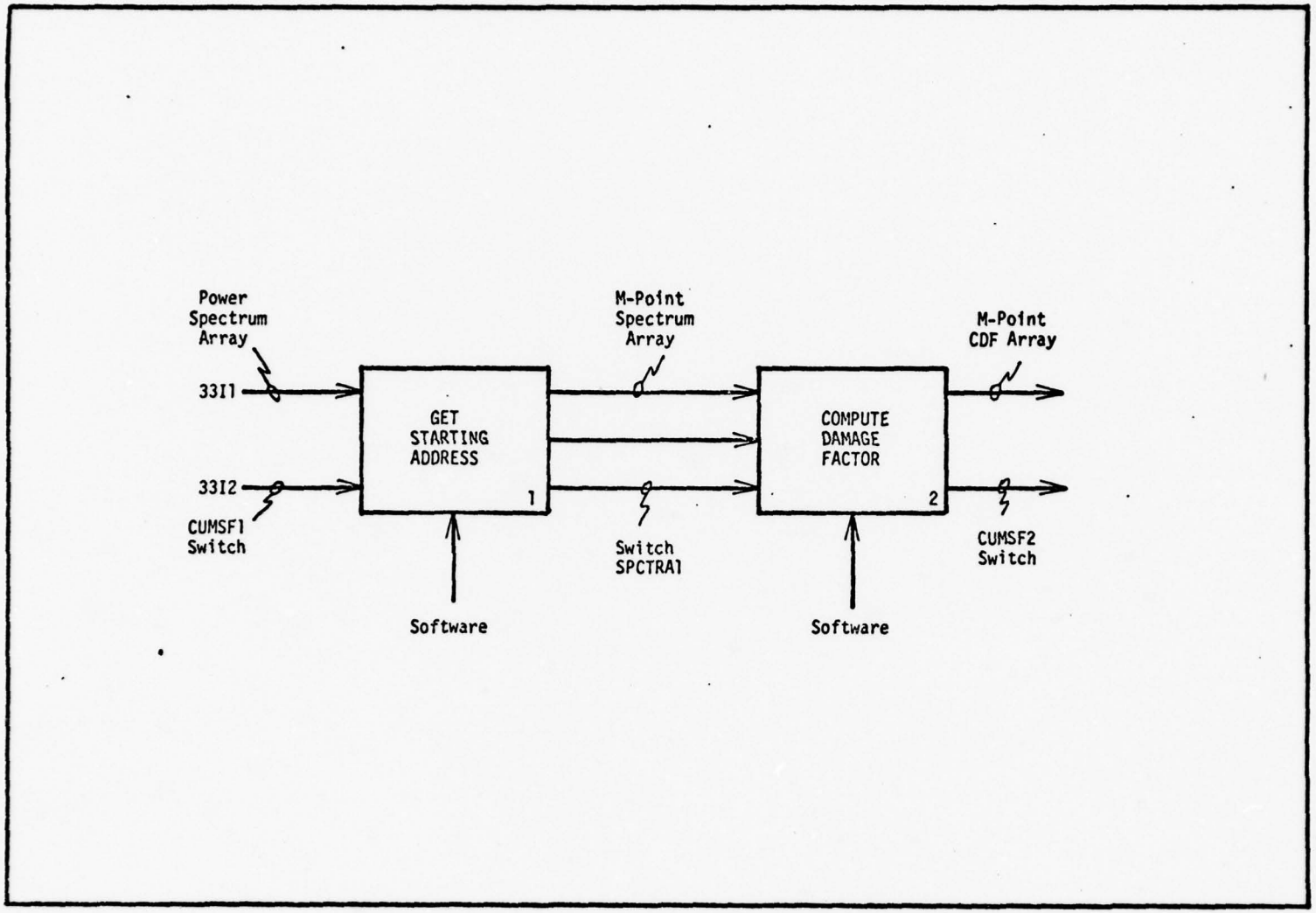


Figure 31. Node A34, Computer Damage Factor

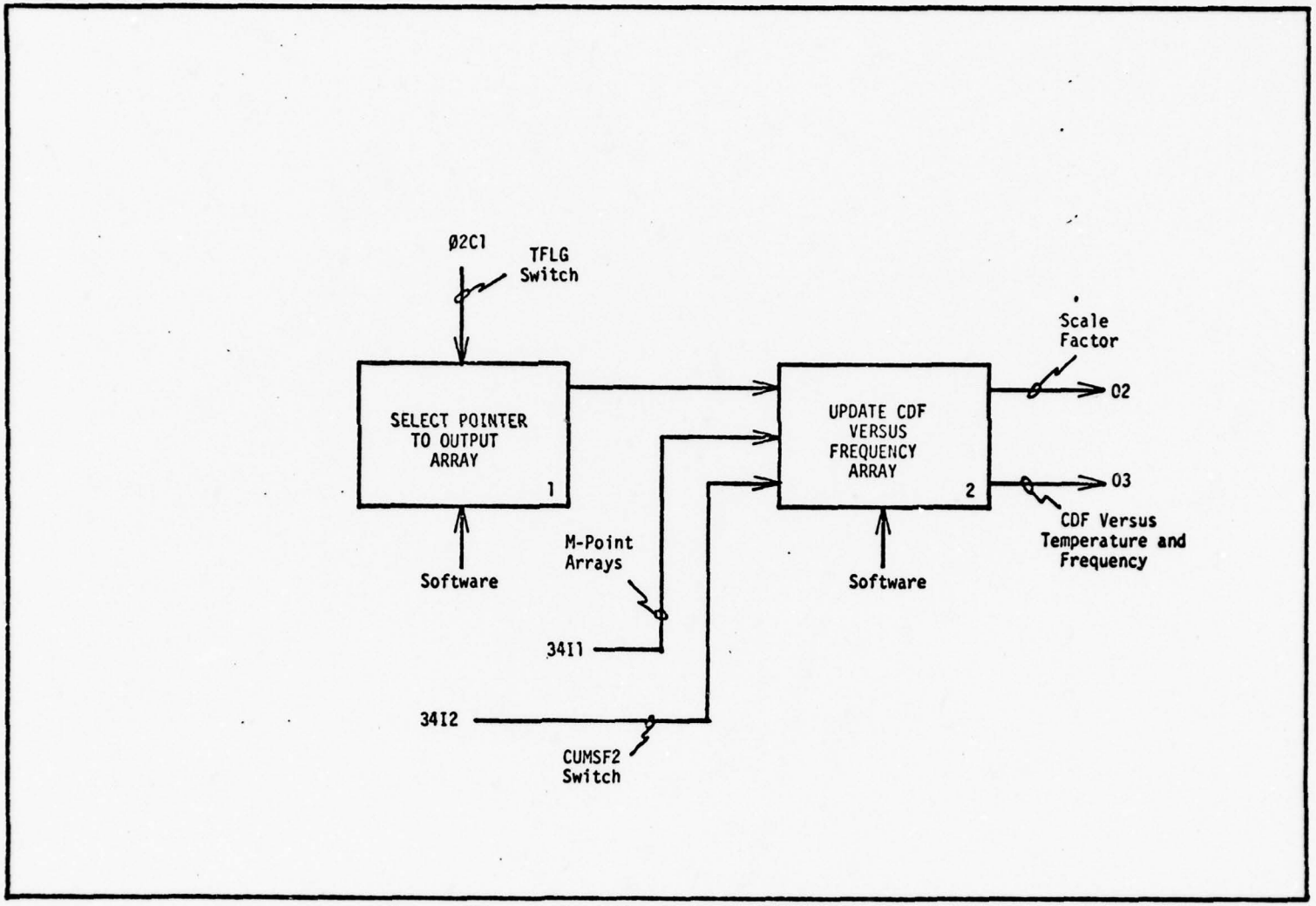


Figure 32. Node A35, Update CDF Array

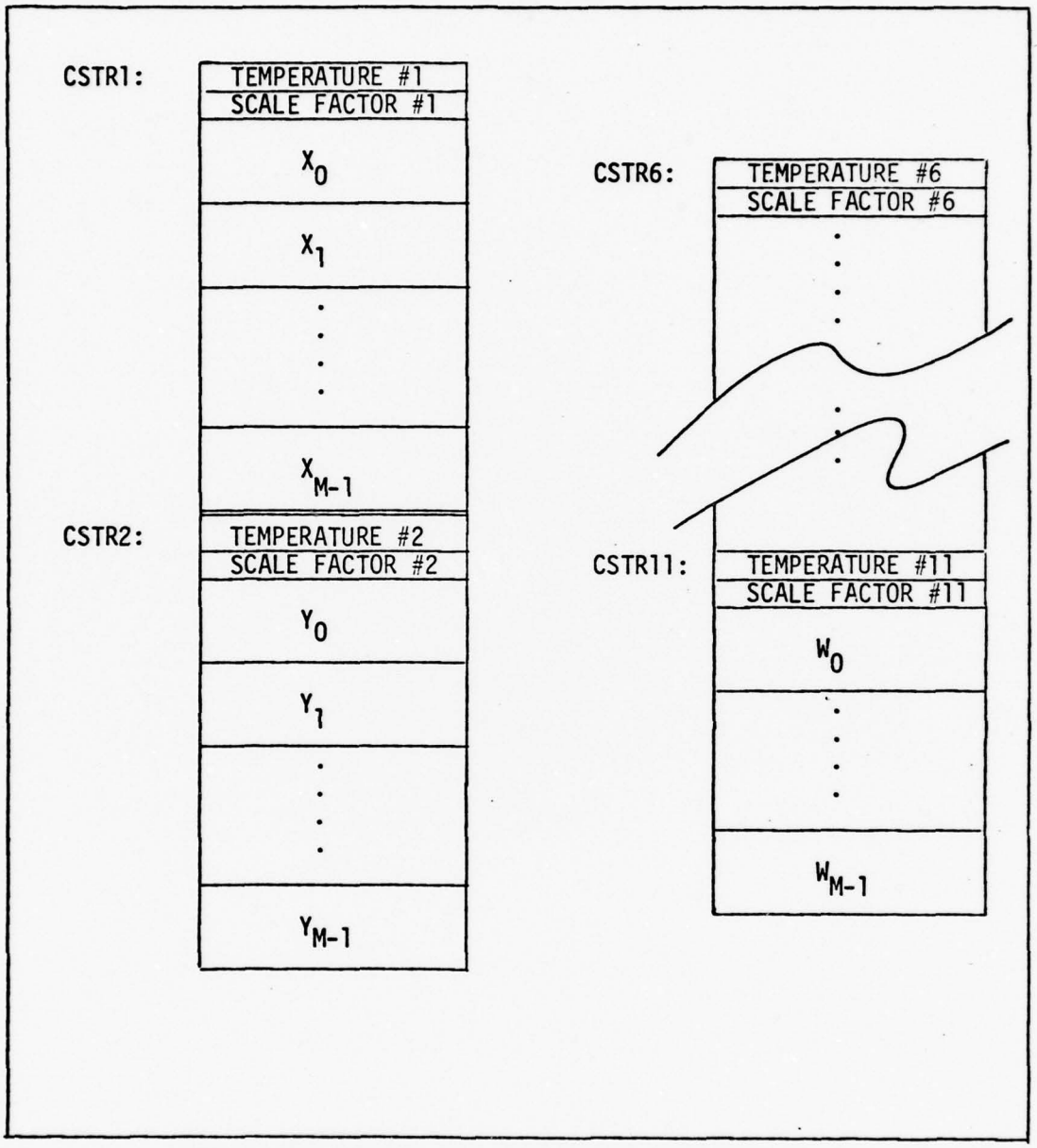


Figure 33. Output Data Structure for Node A35

IV. Design and Implementation of Software

The general design of MIBDAPS discussed in Chapter III presented the foundation for system implementation. The division of the overall system into software and hardward modules was also accomplished. In this chapter, the algorithm for each software module is developed. Subsequently, the LSI-11 assembly language code is written for the designed algorithms. The source listing for each software module is included in Appendices E, F, and G. Before the software algorithms are designed, a mathematical analysis of the signal processing function of MIBDAPS is presented. This analysis is performed to determine system parameters and their variation during the signal processing of data.

Determination of System Parameters

The development of any software module requires a priori knowledge of the computation taking place within the module and the relationship of input and output parameters. Therefore, a mathematical analysis of the data acquisition and signal processing functions of MIBDAPS is required. This analysis should result in the type and amount of computation required within each software module of MIBDAPS and the different input/output parameters. Because the processing of the acquired data is to be done in real time (Ref Chap I), timing constraints placed on the system would also be determined in this analysis.

Data Acquisition Analysis. The MIBDAPS is required to digitize the analog vibration signal and the analog temperature before any signal processing is carried out. The acquisition analysis for the analog vibration signal is carried out first. This would be followed by the acquisition analysis for the analog temperature signal.

The analog-to-digital conversion of a continuous signal is a two-step progress, as shown in Figure 34. The analog signal is first sampled, and then followed by quantization of the amplitudes of the sampled signal (Ref 2:155).

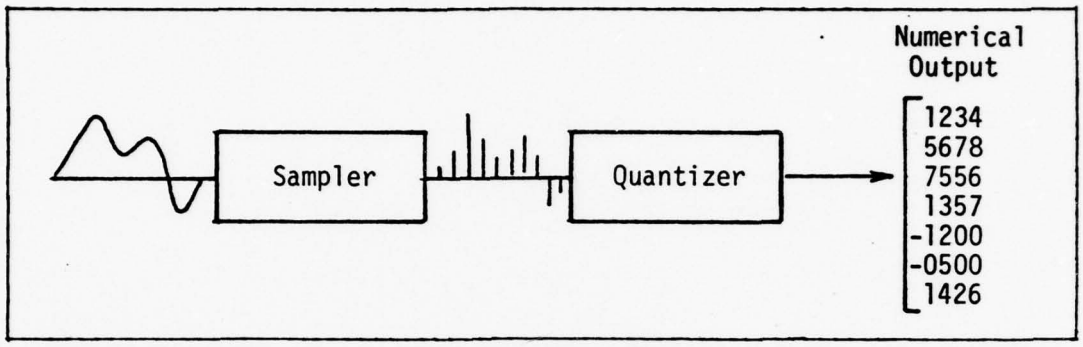


Figure 34. Sampling and Quantization of an Analog (From Ref 2:155)

The limitations and applicability of these two steps will first be considered and, particularly, the consequences arising from the selection of a given sampling rate and number of quantizing levels.

The first problem that arises from sampling a continuous signal is that of aliasing. The nature of this problem is illustrated in Figure 35 which shows that the same set of sampled data points can describe a number of time series histories, which are indistinguishable to the digital computer.

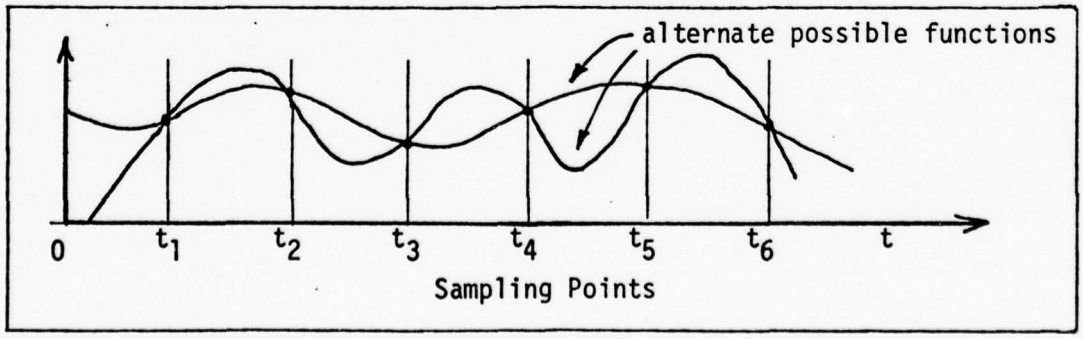


Figure 35. Aliasing (From Ref 2:157)

To overcome this problem, the continuous signal which has a finite band width up to and including B rad/sec should be sampled at π/B sec intervals (Ref 2:160). Therefore, the sampling interval ΔT is given as:

$$\Delta T = \pi/B \tag{2}$$

$$B = 2\pi f_m \tag{3}$$

where

f_m = maximum frequency content of the signal

From Eq (2) and Eq (3) we get

$$\Delta T = 1/2f_m \tag{4}$$

The sampling frequency F, which is the reciprocal of ΔT, is given by

$$F = 1/\Delta T = 2f_m \tag{5}$$

Equation (5) defines the sampling frequency required to reduce the effect of aliasing. This rate is also known as the Nyquist sampling rate (Ref 6: 225). The analog vibration signal input to MIBDAPS is expected to have a finite band width extending from OHZ to 1000 HZ (Ref Chap I); therefore, Eq (5) specifies that the sampling frequency for this signal should be equal to or greater than 2000 HZ.

The second step in data acquisition is quantization. "Quantization" is defined as the representation of a variable amplitude series of discrete sample values as an equivalent series of discrete numbers representing their amplitude values. This process can only be an approximation because the number of bits in a digital representation is limited, even though the analog signal can assume an infinite number of values. The

numerical values of the quantized variable may be represented by some form of binary code to permit entry into the digital computer (Ref 2:162). Figure 36 shows the transfer characteristics of the quantizer which gives a 3-bit code for an input analog signal. The analog values are quantized by partitioning the continuum into eight discrete ranges. All analog values within a given range are represented by the same digital code, which corresponds to the nominal mid-range value. There is, therefore, an inherent quantization uncertainty of $\pm\frac{1}{2}$ LSB. The only sure way to reduce this quantization uncertainty is to increase the number of bits for the representation of output and have more quantization levels (Ref 8:69).

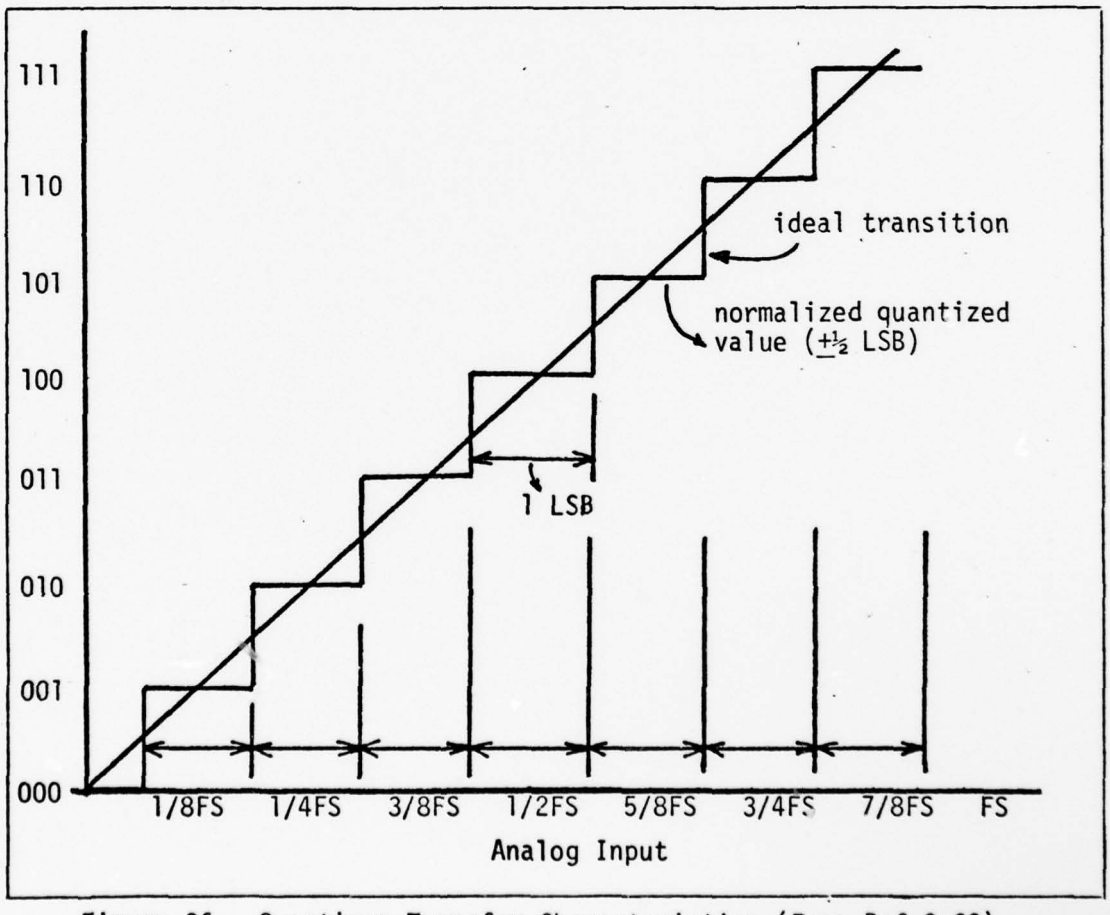


Figure 36. Quantizer Transfer Characteristics (From Ref 8:69)

The ADC used for data acquisition in MIBDAPS is a 12-bit successive approximation converter which has one LSB value equal to 25mv (Ref Chap III). Thus the quantization error, which is $\pm\frac{1}{2}$ LSB, is equal to $\pm$12·5mv. The analog vibration and temperature signal will have a full-scale value of 5 volts; therefore, the quantization error will be $\pm$0.25 percent of the full-scale value.

Signal Processing Analysis. The signal processing function of MIBDAPS has been broken down into three sub-functions (Ref Chap III, Node AØ). The mathematical analysis of the signal processing in each subunit will now be carried out.

1. Processing an Analog Vibration Signal. The power spectral density of the digitized vibration signal needs to be computed first before the damage factor for each frequency of interest is computed (Ref Chap III). This can be accomplished by first determining the frequency domain spectrum of the input time domain signal. The Fourier transform is a method that converts a signal from time to frequency domain. Since the input signal is a sampled version of the continuous time domain signal, the discrete fourier transform (DFT) method was used for the transformation. Equation (6) gives the expression for the conversion of N sampled data points from time domain to frequency domain:

$$F(n) = \sum_{k=0}^{N-1} x(k)\exp\left(\frac{-2\pi nk}{N}\right), \quad n = 0, 1, 2 \ldots N-1 \qquad (6)$$

where F(n) is the nth point in the frequency domain and n(k) the kth point in the time domain. The x(k) may be complex and the F(n)'s are always complex. We can rewrite Eq (6) as

$$F(n) = \sum_{k=0}^{N-1} x(k)W^{nk}, \quad n = 0, 1, 2 \ldots N-1 \tag{7}$$

where

$$W = \exp\left(\frac{2\pi i}{N}\right) \tag{8}$$

Figure 37 shows the required frequency domain spectrum (Ref Chap I) from 0 HZ to 1000 HZ. The frequency interval Δf, which is 100 HZ, will be used to determine the number of time domain samples required for the Fourier transformation. Equation (9) gives the relationship between Δf and the sampling interval T (Ref 3:87).

$$\Delta f = 1/T \tag{9}$$

or

$$T = 1/\Delta f \tag{10}$$

also

$$T = M \times \Delta T \tag{11}$$

therefore

$$M = T/\Delta T \tag{12}$$

Equation (12) gives the number of sampled points M required to get a frequency spectrum shown in Figure 37. The relationship between the signal in the frequency domain and time domain is shown in Figures 37 and 38, respectively. From Eq (4) ΔT is computed equal to 0.5 msec, and Eq (10) gives T equal to 10 msec. Therefore, the number of sampled points M required to get the desired frequency spectrum is 20, which is calculated using Eq (12). Once the frequency spectrum of the time domain signal is computed, the next step in signal processing is to compute the power spectral density for each frequency component. After the Fourier transformation, each frequency component F(n) in Eq (6) is a complex number as given in Eq (13):

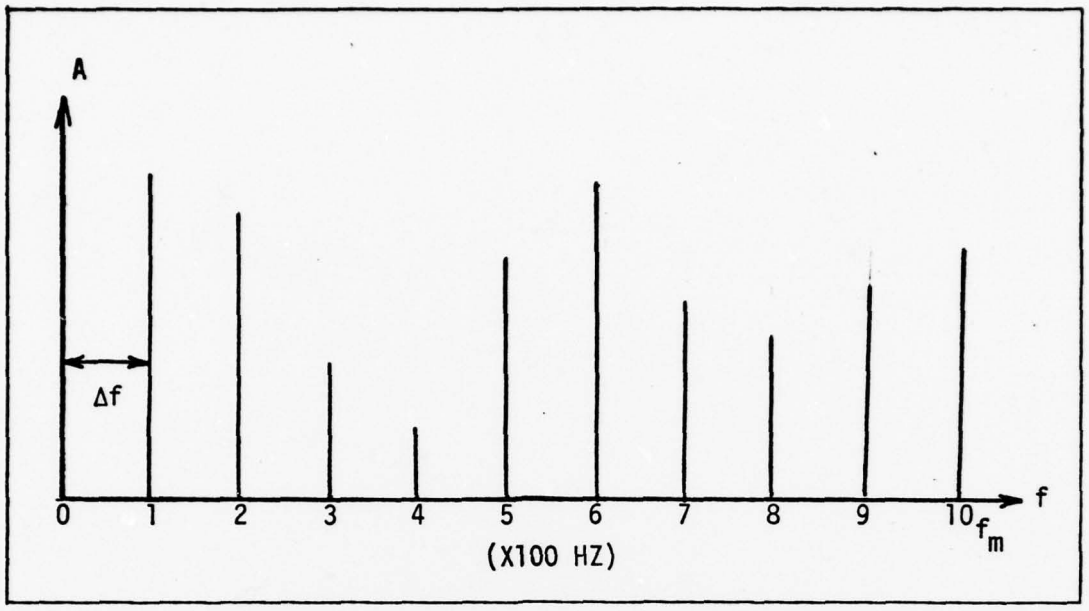


Figure 37. Frequency Domain

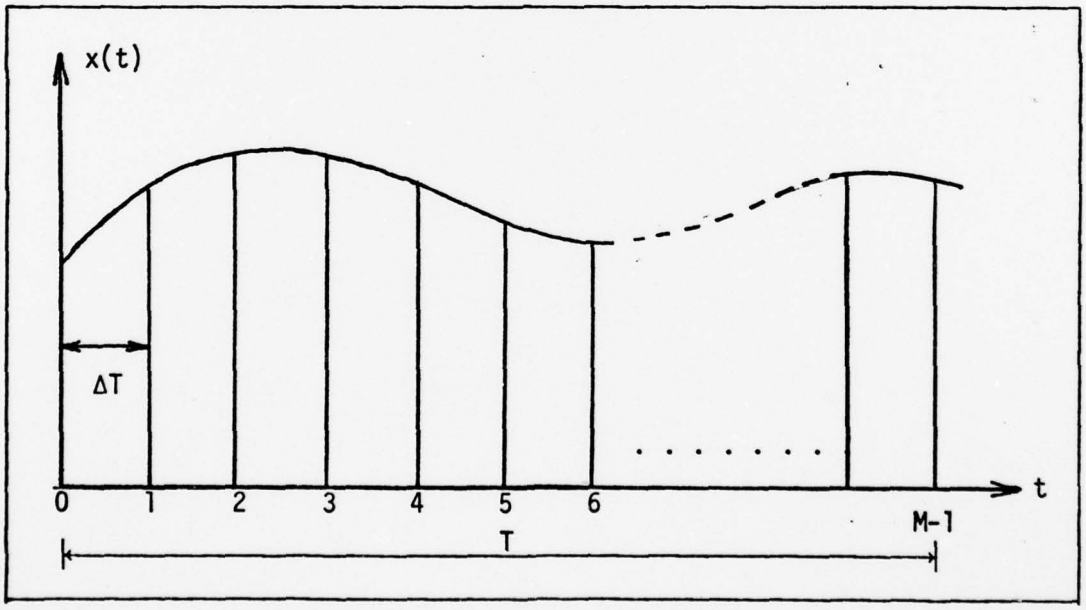


Figure 38. Time Domain

$$F(n) = x_n + jy_n \tag{13}$$

the power spectral density can be calculated using Eq (14):

$$A(n) = (x_n)^2 + (y_n)^2 \tag{14}$$

However, the nature of the input signal is random (Ref 13) and to get a good representation of the power spectral density a mean value for the P set of spectral density points is taken (Ref 2:195). This is expressed mathematically in Eq (15):

$$\tilde{A}(n) = \frac{1}{P} \sum_{k=0}^{P-1} A(n), \quad n = 0, 1, \ldots M-1 \tag{15}$$
$$K$$

where subscript k denotes a unique set of M spectral density values, and $\tilde{A}$ is the mean power spectrual density.

Once the mean power spectral density is computed, the damage factor at each frequency component can be evaluated using Eq (16) (Ref 13):

$$DF(n) = \frac{k[\bar{A}(n)]^{3.2}}{(f_n)^{5.4}}, \quad n = 0, 1 \ldots M-1 \tag{16}$$

where

$$\bar{A}(n) = \sqrt{\tilde{A}(n)} \tag{17}$$

Here f_n is the frequency at which the damage factor is being calculated. From Eqs (16) and (17) the following relationship is derived:

$$DF(n) = \frac{k\left[\{\tilde{A}(n)\}^{\frac{1}{2}}\right]^{3.2}}{(f_n)^{5.4}}, \quad n = 0, 1 \ldots M-1 \tag{18}$$

or

$$DF(n) = \frac{k[\tilde{A}(n)]^{1.6}}{(f_n)^{5.4}}, \quad n = 0, 1 \ldots M-1 \tag{19}$$

Equation (19) eliminates the need to compute the root mean square (RMS) of the average power spectral density. The CDF, which is a function of frequency and temperature, is obtained by continuous summation of DF values over a predetermined interval. Equation (20) gives the required relationship:

$$CDF(t_i, m) = \sum_{j=1}^{R} DF_j(m), \quad m = 0, 1 \ldots M-1 \tag{20}$$

Here, t_i represents the ith temperature interval for which the CDF is computed, and subscript j indicates the number of DF value sets that have been summed together.

2. Analog Temperature Signal. The temperature signal is sampled every time a set of 20 data points of the vibration signal is processed and the power spectral density has been computed. Since P set of power spectral density values are summed [Eq (15)], there are P temperature values that can be summed up to give an average temperature value $\overline{T}$. The average temperature value would be available when the DF is computed and the final output array (CDF versus frequency and temperature) is to be updated. The following equation gives the mathematical relationship for $\overline{T}$ (average temperature):

$$\overline{T} = \frac{1}{P} \sum_{k=0}^{P-1} T(k) \tag{21}$$

This concludes the mathematical analysis of the data acquisition and signal processing functions of MIBDAPS. The software modules to implement different functional activities will now be developed.

Implementation of Software Modules

The software modules which will be used to perform different functional activities are developed in subsequent paragraphs.

Fourier Transform Module. The Fourier transform is a method to convert the signal spectrum from the time domain to a spectrum in the frequency domain. Since MIBDAPS suses digitized data for processing, the DFT algorithm can be used to transform the digitized vibration signal from the time domain to the frequency domain. Equation (7) gives the mathematical relationship to convert a digitized signal to a frequency domain. A computer program to perform the summation given in Eq (7) can be written, but it turns out that this can be a very slow and time-consuming process if the equation is implemented as given (Ref 6:267). The reason for this inefficiency is that the transformation of N-data points requires N^2 complex multiplications and N(N-1) complex additions (Ref 3:151). To speed up the DFT implementation on the computer, Cooley and Tukey (Ref 3:151) developed a fast Fourier transform (FFT) algorithm, which for $N = 2^x$ points reduces the complex multiplication to Nx/2 and the complex addition to Nx (Ref 3:151).

Appendix C gives the theoretical development of the base 2 FFT algorithms. It was determined earlier that 20 points of DFT are required to get the required frequency spectrum of the vibration signal. N = 20 does not satisfy the relationship $N = 2^x$ where x is an integer; therefore, a base 2 FFT algorithm would not be used. FFT algorithms for arbitrary factors can also be written (Ref 3:184). An FFT algorithm for 20-point DFT was written, but it was found that 125 complex multiplications and 108 complex additions are required. On the other hand, a 32-point FFT algorithm requires 80 complex multiplications and 160 complex additions,

so it was decided to use a 32-point FFT algorithm. This would give a frequency spectrum from 0 HZ to 1500 HZ, and by picking up the first 11 frequency components, the desired spectrum for MIBDAPS could be obtained. Figure 39 shows the relationship between the time domain signal and the resultant frequency domain signal. The sampling frequency F should now be 3200 HZ [Ref Eq (5)], and the sampling period ΔT becomes equal to 0·3125 msec.

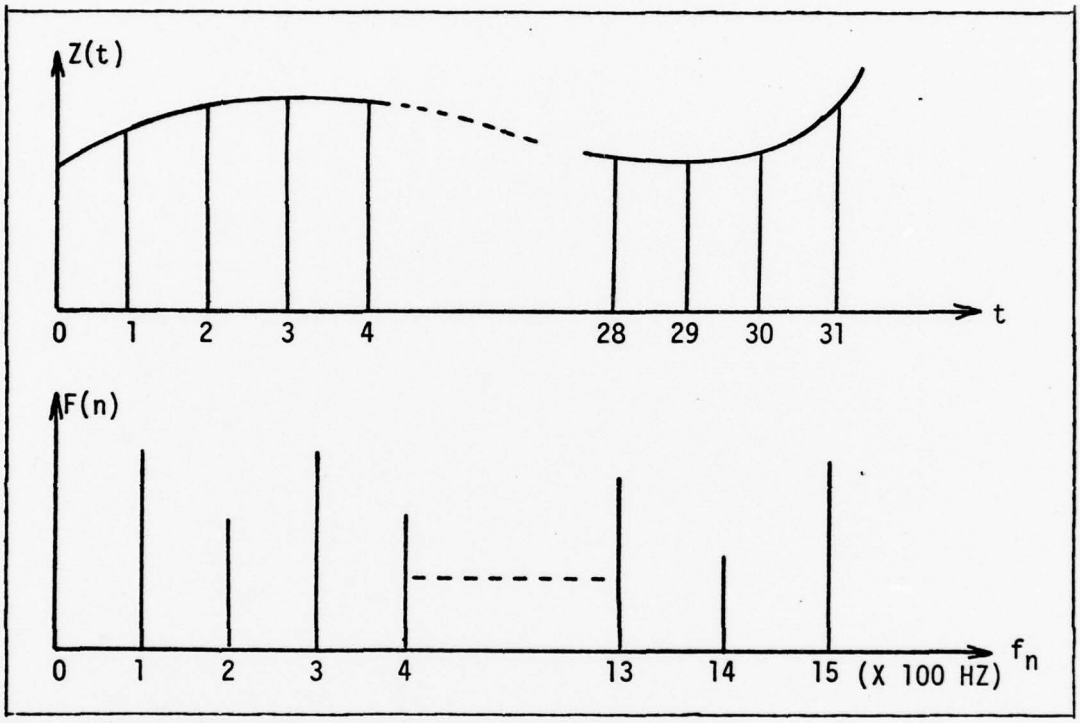


Figure 39. Time and Frequency Domain Signals for 32 Samples

The FFT algorithm developed in Appendix C can handle both real and complex input data. However, the input to MIBDAPS is real data, and computation time would be halved if 2N points of real data are treated as N points of complex data. This would enable the use of an N-point FFT algorithm. The theoretical background for transforming 2N points of real data using an N-point FFT is given in Appendix D; this algorithm was eventually

used for implementation of the Fourier transform module. The description of the algorithm is given in the following paragraphs.

1. Algorithm to Implement Fourier Transform Module. Figure 40 shows the flow chart of the algorithm that was implemented. The input data array is assumed to have 32 real data points in contiguous memory locations. Since the FFT algorithm described in Appendix D would be implemented, it is assumed that the input array has 16 complex data points (all odd points are treated as real points and all even points are treated as complex points). The bit-inversion subroutine is called first so that the data elements are bit-reversed before being processed by the FFT subroutine. This is done to ensure that the output of the FFT subroutine would be in the correct order. The FFT computation is performed on the bit-inversed array; the resultant array would be 16 complex Fourier transformed points. To get the correct transformed array for real data points, the post-processing computation (Ref Appendix D) is performed; this would give the 16 complex values corresponding to frequencies from 0 HZ to 1600 HZ. Although a 32-point real data array should result in 32 complex point Fourier transformed arrays, the algorithm being implemented only gives 16 complex Fourier transformed points. This discrepancy is because the other 16 complex transformed points are the complex conjugates of these points and are obtainable easily. However, these points are not required for further processing in MIBDAPS and, therefore, are not computed.

2. Description of Different Subroutines.

a. Bit-Inversion Subroutine. The flow diagram for the bit-inversion (BIV) subroutine is given in Figure 41. The starting address of the input data array is stored in pointer DATA. The five LSB of the

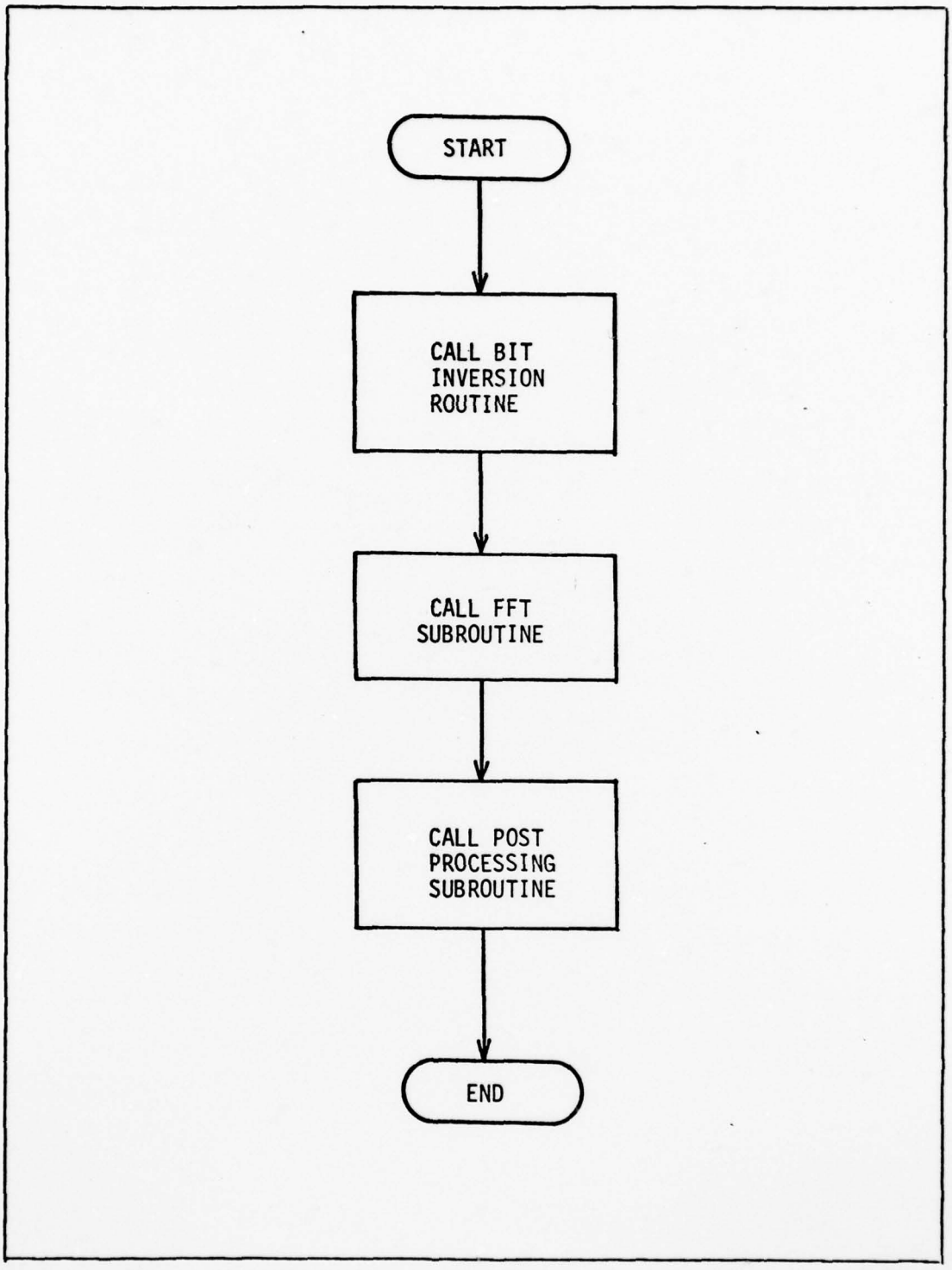


Figure 40. Flow Diagram from Fourier Transform Module

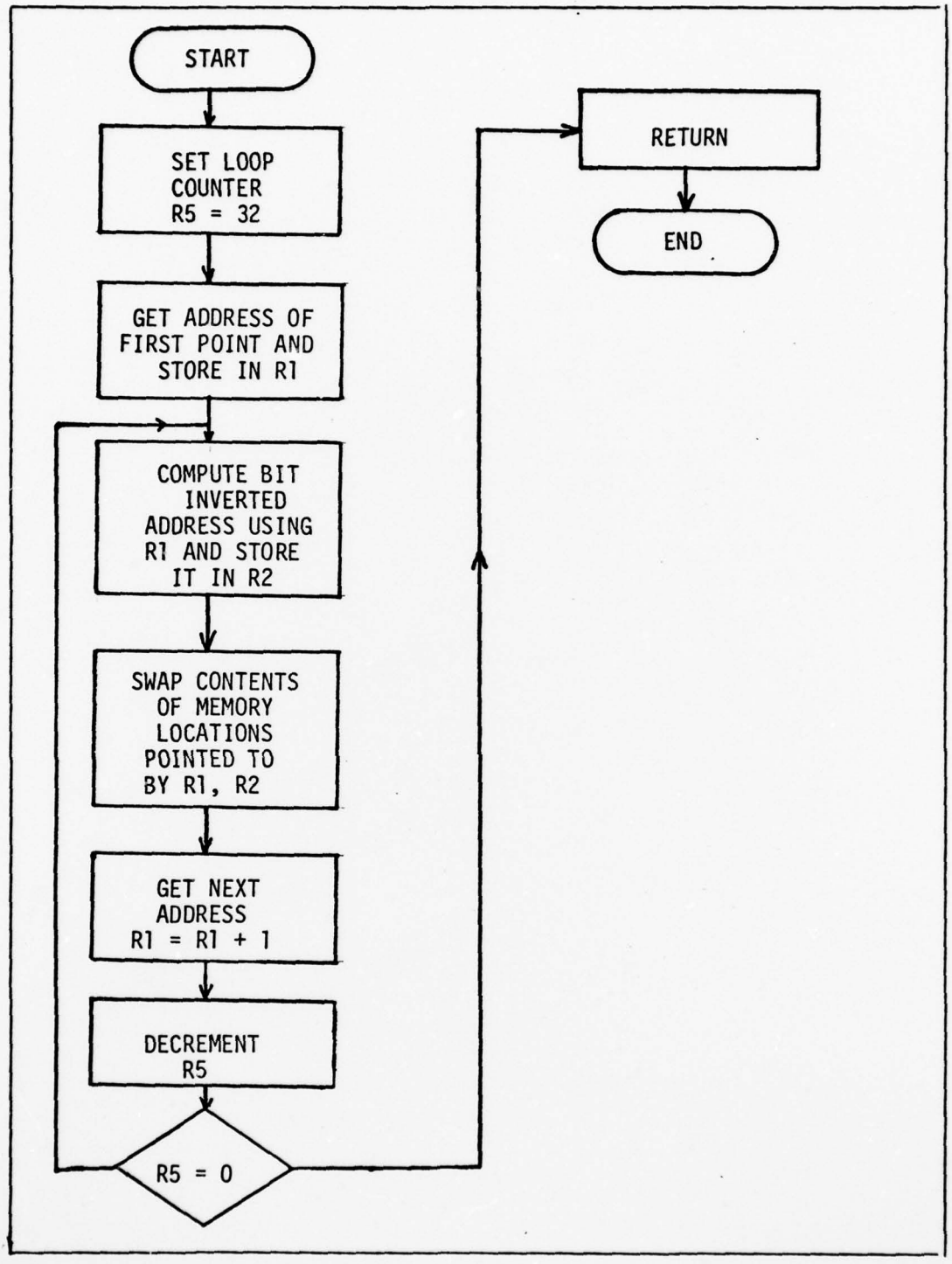


Figure 41. Flow Diagram for Bit-Inversion Subroutine

address for any data elements are used to compute the bit-inverted address. These two memory location contents are then swapped. This process is repeated for all data elements, and the resulting array is in bit-reversed order.

b. FFT Subroutine. The transformation of input data is carried out in four steps (or passes). The manipulation of data within this subroutine follows the scheme given in the signal flow graph shown in Figure C-2, Appendix C. During the first pass, there are no complex multiplications (Ref Table C-I, Appendix C); therefore, the following equations need to be implemented:

$$R(m)' = R(m) + R(n) \tag{22}$$

$$I(m)' = I(m) + I(n) \tag{23}$$

$$R(n)' = R(m) - R(n) \tag{24}$$

$$I(n)' = I(m) - I(n) \tag{25}$$

The letter within the parenthesis signifies a pair of points which are used in the calculation.

The second, third, and fourth pass require complex multiplications because of the W_N^{nk} factor. The equations implemented in these passes are:

$$X(m) = X(m) + X(n)W^y \tag{26}$$

$$X(n) = X(m) + X(n)W^z \tag{27}$$

Here

$$X(m) = R(m) + jI(m)$$

$$X(n) = R(n) + jI(n)$$

$$W^y = \exp\left(\frac{-2\pi in}{N}\right) = W^z$$

Also, the exponents of W, y and z, are N/2 apart, such that the following relationship holds:

$$W^y = -W^{N/2+y} = -W^z \tag{28}$$

Equations (26) and (27) can be rewritten using Eq (28).

$$X(m) = X(m) + X(n)W^y \tag{29}$$

$$X(n) = X(m) - X(n)W^y \tag{30}$$

or

$$R(m) + jI(m) = R(m) + jI(m) + [R(n) + jI(n)](Cosy + jSiny) \tag{31}$$

$$R(n) + jI(n) = R(m) + jI(m) - [R(n) + jI(n)](Cosy + jSiny) \tag{32}$$

Collecting terms and rearranging Eqs (31) and (32), the following relationship evolves:

$$R(m)' = R(n)Cosy + I(n)Siny + R(m) \tag{33}$$

$$R(n)' = -R(n)Cosy - I(n)Siny + R(m) \tag{34}$$

$$I(m)' = -R(n)Siny + I(n)Cosy + I(m) \tag{35}$$

$$I(n)' = R(n)Siny - I(n)Cosy + I(m) \tag{36}$$

Equations (33), (34), (35), and (36) are finally used in computation during the second, third, and fourth pass. The flow diagram for the FFT subroutine is given in Figure 42. The flow diagram shows a call to subroutine SCALE; this subroutine is used to ensure that the data array is scaled to avoid any arithmetic overflow during computations in any pass. The flow diagram for the subroutine SCALE is shown in Figure 43. Before the subroutine is called, it is assumed that the starting address is loaded in Register R0, and the number of points in the array is stored in Register R1.

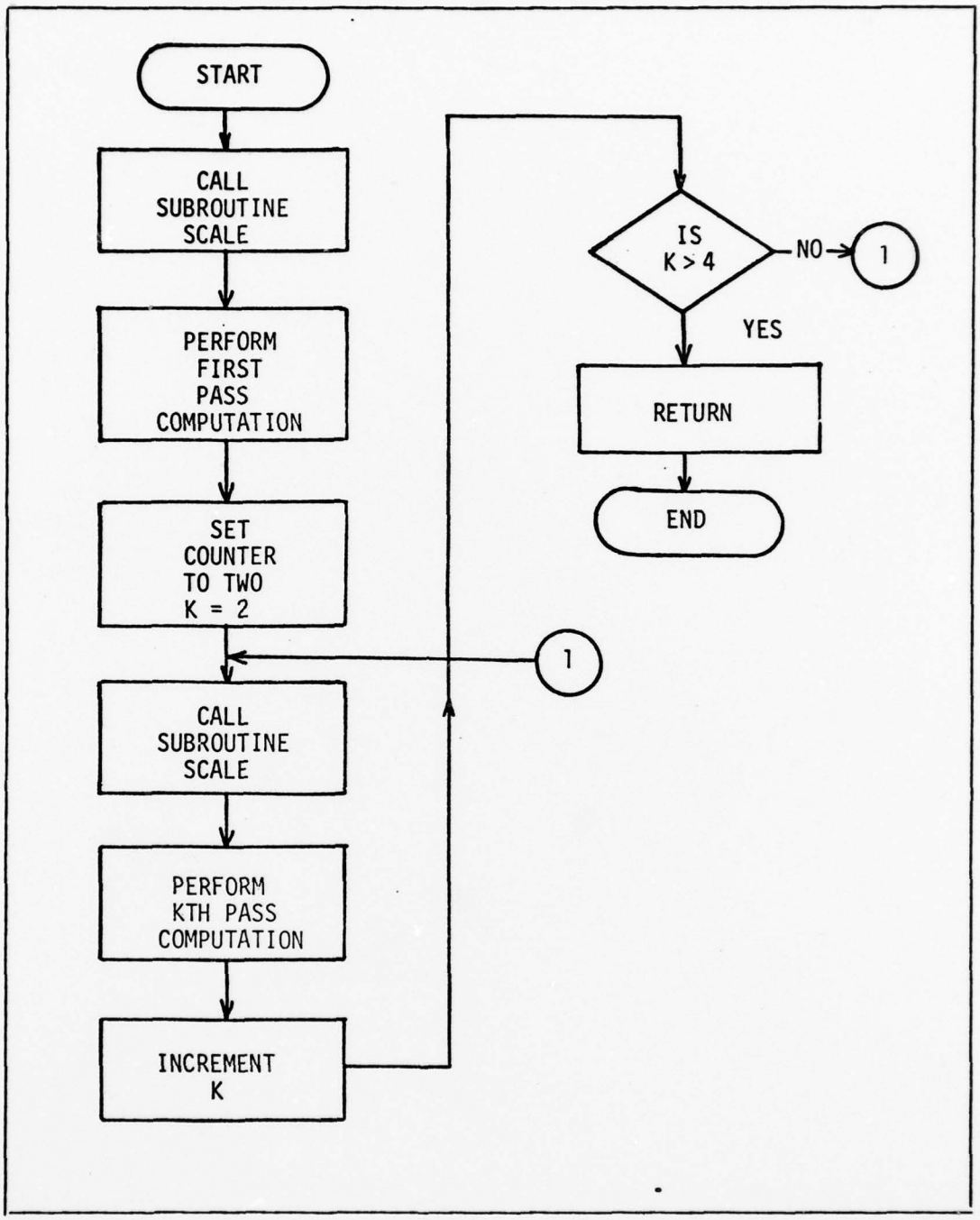


Figure 42. Flow Diagram for FFT Subroutine

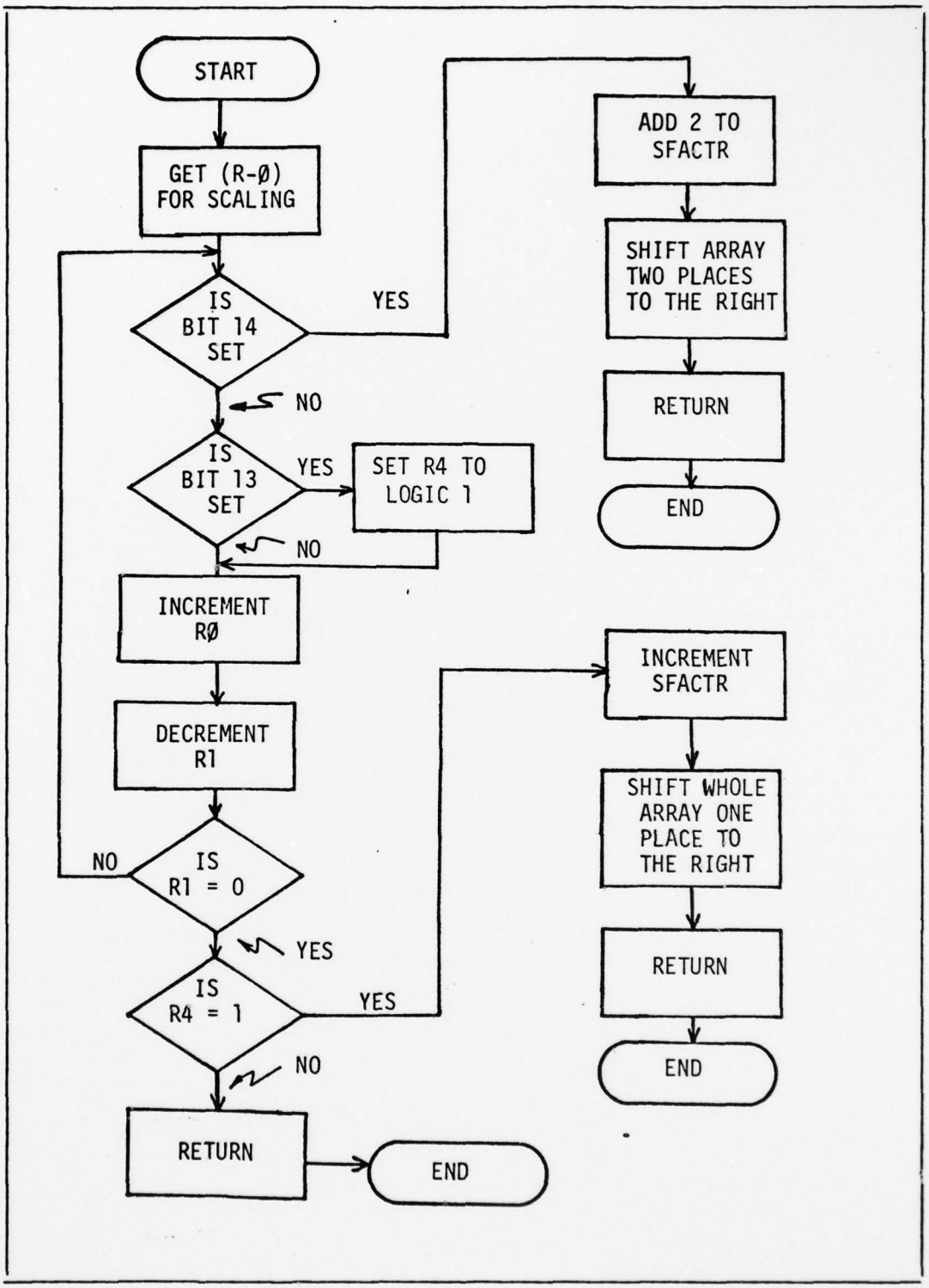


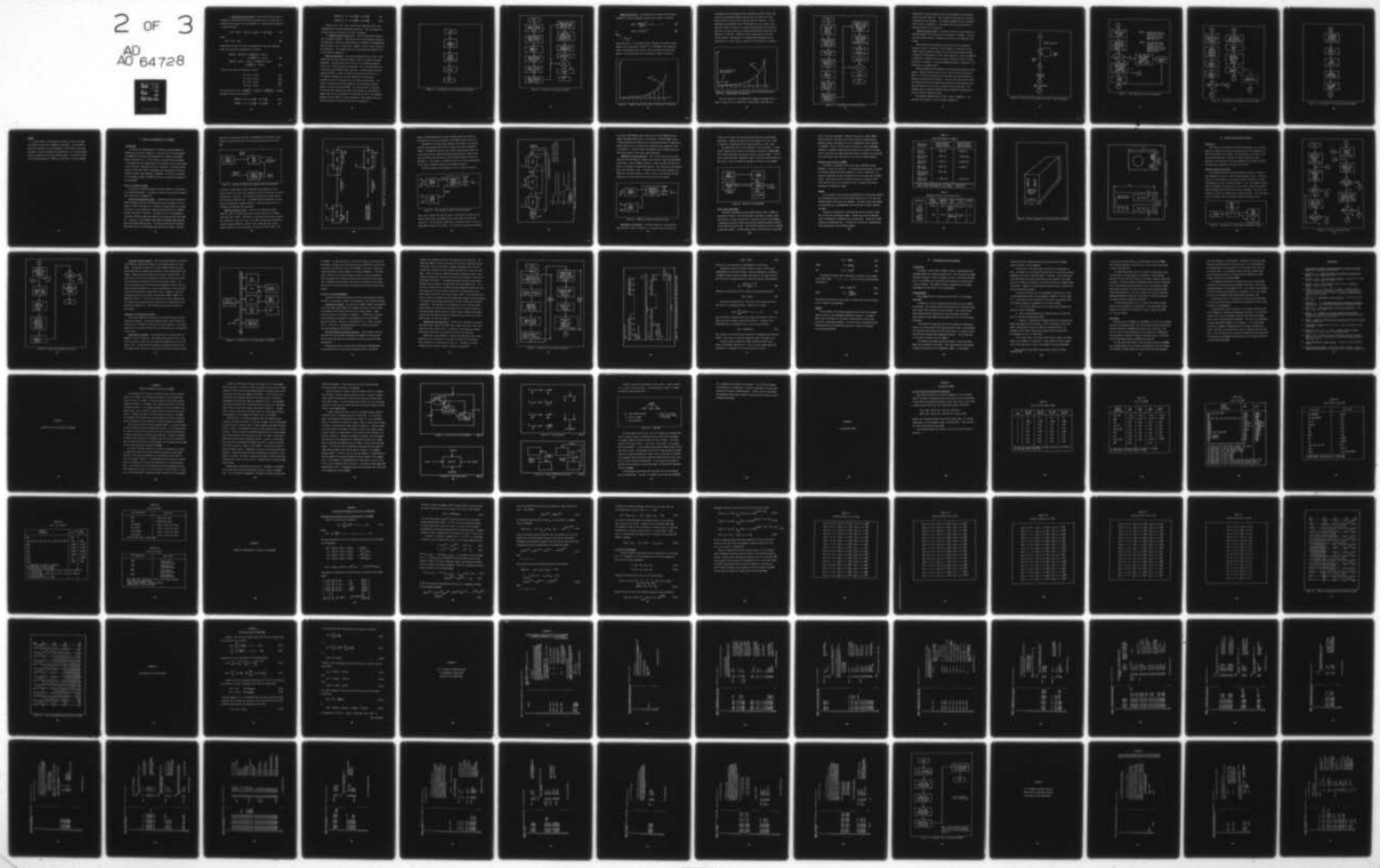
Figure 43. Flow Diagram for Subroutine SCALE

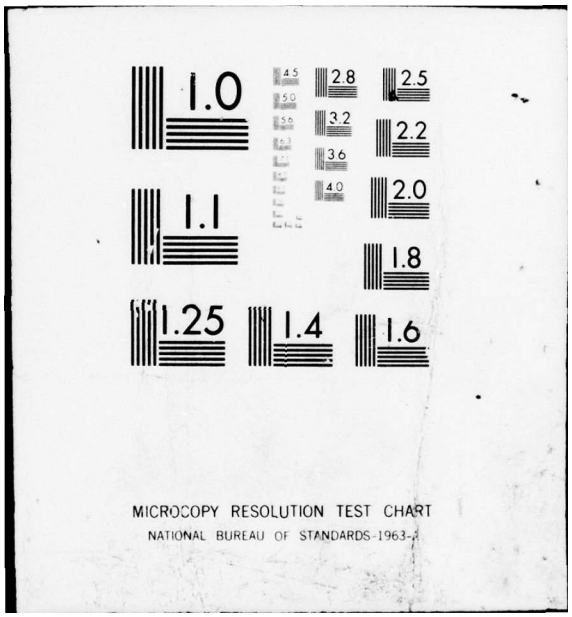
AD-A064 728 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 9/2
MICROPROCESSOR BASED DATA ACQUISITION AND PROCESSING SYSTEM.(U)
DEC 78 S IFTEKHAR

UNCLASSIFIED AFIT/GE/EE/78-29 NL

2 OF 3

AD
AD A064728




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

c. Post-Processing Subroutine. After the FFT of input data is completed, the post-processing of the resulting array is carried out to get the correct output array (Ref Appendix D). Equation (D-8), Appendix D, is rewritten below:

$$X(n) = \tfrac{1}{2}[V(n) + V^*(N-n)] + \tfrac{1}{2j}[V(n) - V^*(N-n)]W_{2N}^{n} \qquad (37)$$

where

$$V(n) = Y(n) + jZ(n) \qquad (38)$$

Substituting Eq (38) in Eq (37) and separating the real and imaginary parts, the following relationships are derived.

$$\mathrm{Re}[X(n)] = \tfrac{1}{2}[Y(N-n)] + \cos(\tfrac{\pi n}{N})[Z(n) + Z(N-n)] - \sin(\tfrac{\pi n}{N})[Y(n) - Y(N-n)] \qquad (39)$$

$$\mathrm{Im}[X(n)] = \tfrac{1}{2}[Z(n) - Z(N-n)] - \cos(\tfrac{\pi n}{N})[Y(n)-Y(N-n)] - \sin(\tfrac{\pi n}{N})[Z(n) + Z(N-n)] \qquad (40)$$

If Rp, Ip, Rn, and Im are defined as follows

$$Rp = Y(n) + Y(N-n) \qquad (41)$$

$$Ip = Z(n) + Z(N-n) \qquad (42)$$

$$Rm = Y(n) - Y(N-n) \qquad (43)$$

$$Im = Z(n) - Z(N-n) \qquad (44)$$

and using the fact that $\sin\left[\frac{(N-n)\pi}{N}\right] = \sin(\frac{n\pi}{N})$ and $\cos\left[\frac{(N-n)\pi}{n}\right] = -\cos(\frac{n\pi}{n})$, Eqs (39) and (40) can be rewritten:

$$\mathrm{Re}[X(n)] = Rp + Ip\cos(\tfrac{n\pi}{N}) - Rm\sin(\tfrac{n\pi}{N}) \qquad (45)$$

$$\mathrm{Im}[X(n)] = Im - Ip\sin(\tfrac{n\pi}{N}) - Rm\cos(\tfrac{n\pi}{N}) \qquad (46)$$

$$\text{Re}[X(N-n)] = Rp - Ip\ \text{Cos}\left(\frac{n\pi}{N}\right) - Rm\ \text{Sin}\left(\frac{n\pi}{N}\right) \quad (47)$$

$$\text{Im}[X(N-n)] = Im - Ip\ \text{Sin}\left(\frac{n\pi}{N}\right) + Rm\ \text{Cos}\left(\frac{n\pi}{N}\right) \quad (48)$$

Equations (45), (46), (47), and (48) were eventually used in the implementation of the post-processing subroutine. The flow diagram for the post-processing subroutine is given in Figure 44.

3. Assembly Language Code for FFT. The FFT algorithm was coded in LSI-11 assembly language. A deviation from the flow diagram of Figure 42 was made and all other subroutines (SCALE, BIT INVERSION, POST-PROCESSING) were included in the FFT subroutine. Appendix E gives a source listing of FFT subroutines. The complete analysis of the subroutine execution time is also given in Appendix E.

Power Spectrum Module. The Fourier transformed array, which is the output from the Fourier transform module, contains 16 complex frequency components. The power spectrum for the first 11 frequency components (from 0 HZ to 1000 HZ) needs to be computed. This is accomplished using a subroutine called PRSPEC (Power Spectrum). The flow diagram from the subroutine PRSPEC is shown in Figure 45, which shows that the first 11 frequency components of the power spectrum are calculated by squaring the real and imaginary parts and summing them together. The output is added to the previous elements of an array whose starting address is stored at pointer SPCTRA. This feature helps in computing the average power spectrum which would then be used for computing the damage factor at each frequency of interest [Ref Eq (15)]. The assembly language code for PRSPEC is given in Appendix F; this appendix also contains the analysis of execution time for this subroutine.

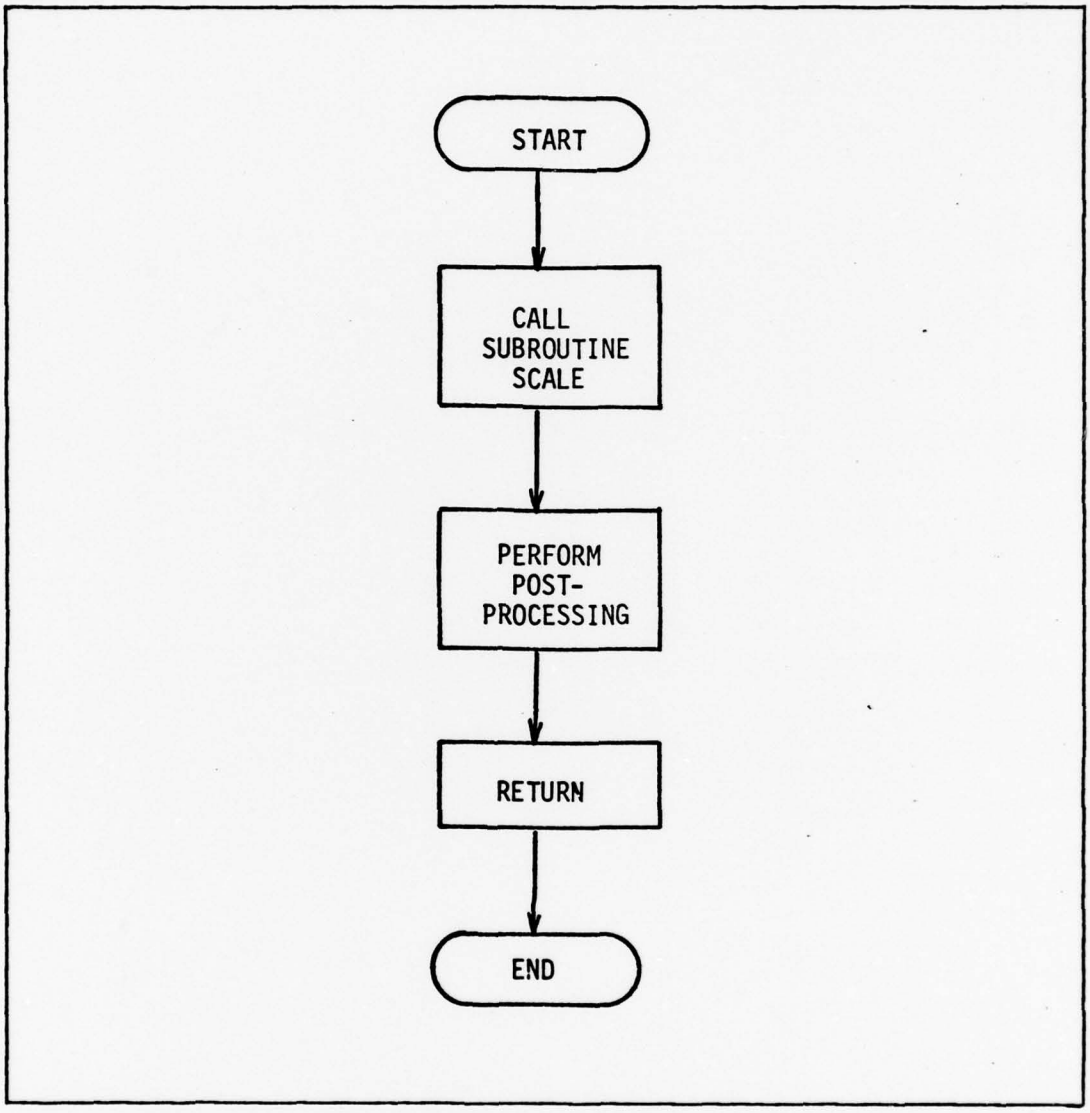


Figure 44. Flow Diagram for Post-Processing Subroutine

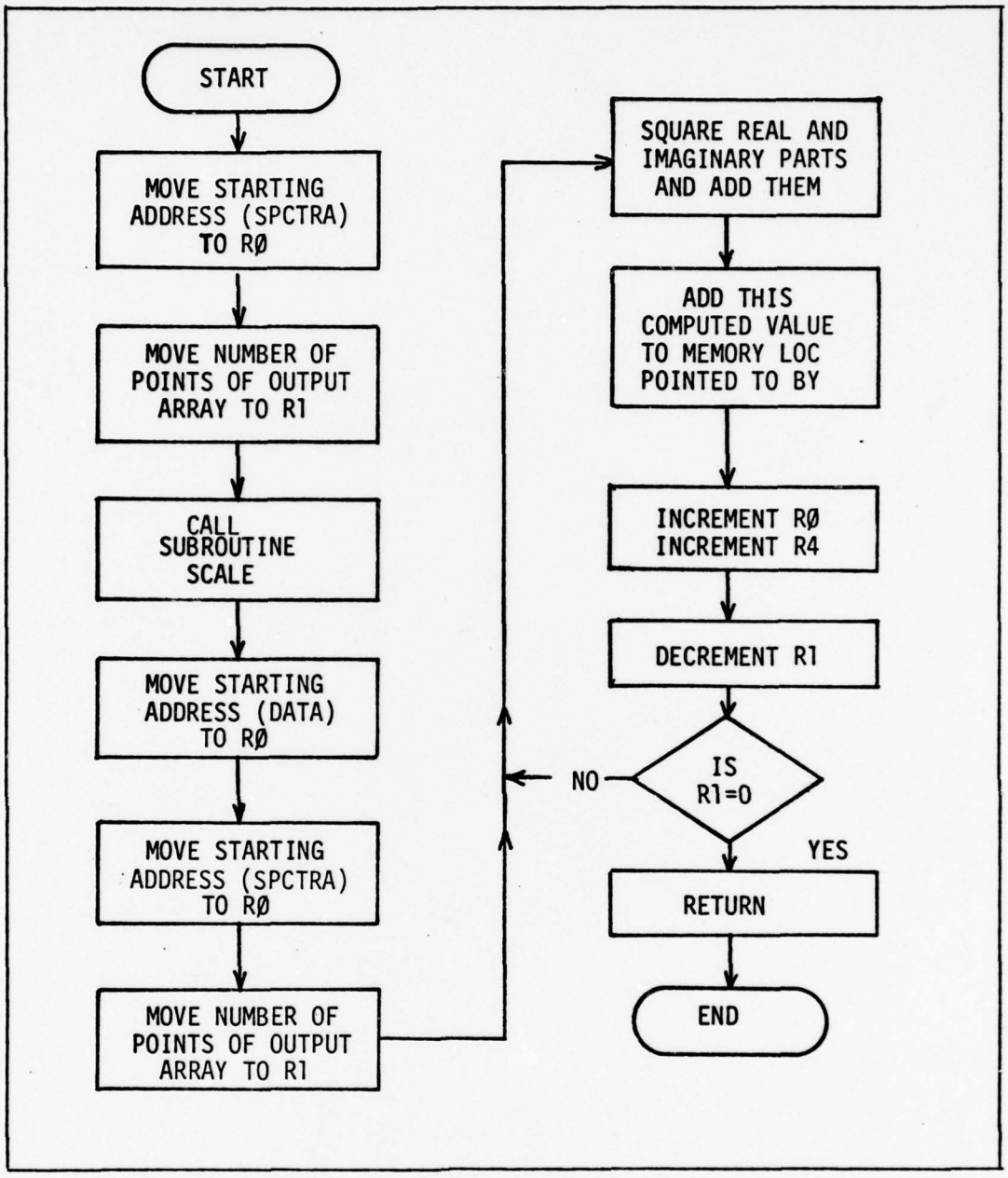


Figure 45. Flow Chart for Subroutine PRSPEC

Damage Factor Module. The subroutine to calculate the DF at each frequency of interest implements Equation (19), which is rewritten:

$$DF(n) = \frac{K[\tilde{A}(n)]^{1.6}}{f_{(n)}^{5.4}}, \quad n = 0, 1, \ldots 10 \tag{49}$$

or

$$DF(n) = K'[\tilde{A}(n)]^{1.6} \tag{50}$$

where

$$K' = K/(f_n)^{5.4}$$

Since the factor K' is constant for each frequency, the function implemented in this subroutine is $[\tilde{A}(n)]^{1.6}$. To implement this exponential factor, the expected values of $\tilde{A}(n)$, which are going to be less than equal to 1.0, were divided into 10 intervals as shown in Figure 46.

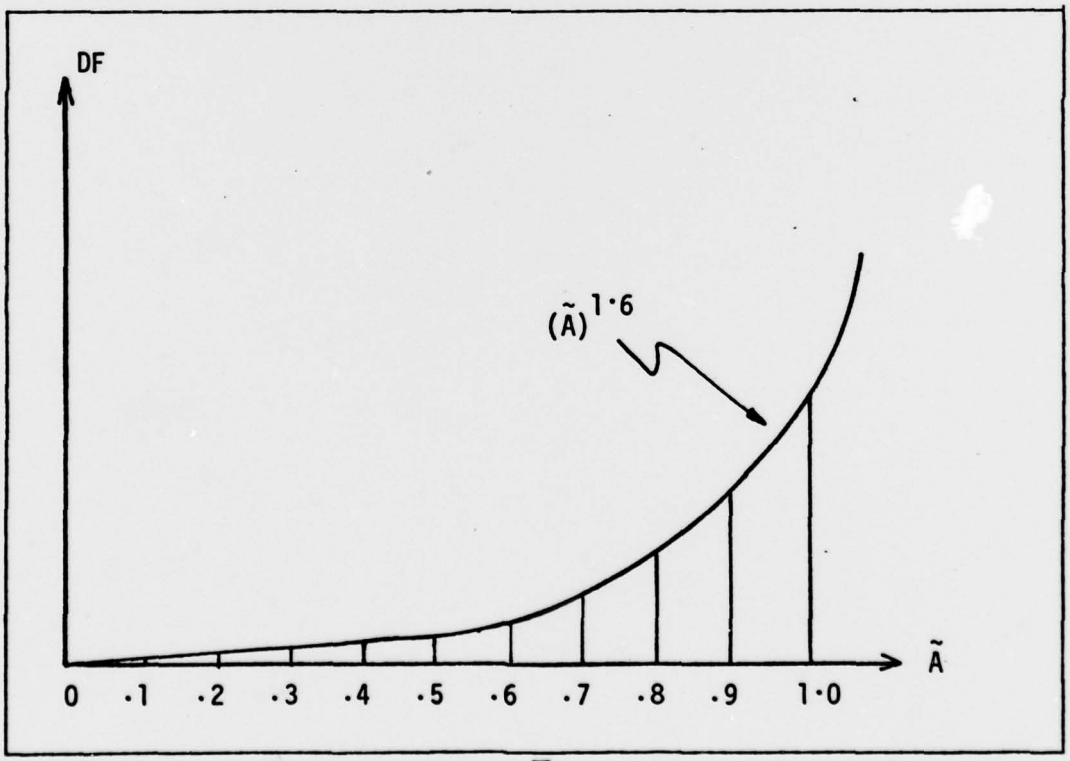


Figure 46. Damage Factor Versus $\bar{A}$ Curve Divided Into 10 Intervals

A polynomial curve fit program PLSCF, available as an AFIT Library subroutine on the CDC 6600 computer (Ref 15:A-5), was used to fit polynomial curves to each of the 10 intervals shown in Figure 46. It was experimentally determined that if the ordinate (DF) was divided into 8 intervals, shown in Figure 47, a second-order polynomial gave the least RMS error for these intervals as compared with RMS errors for other combinations of intervals. Appendix K gives a description of the curve fitting program. The equation for a second-order polynomial and the coefficients for the 8 intervals (Figure 47) are also given in Appendix K.

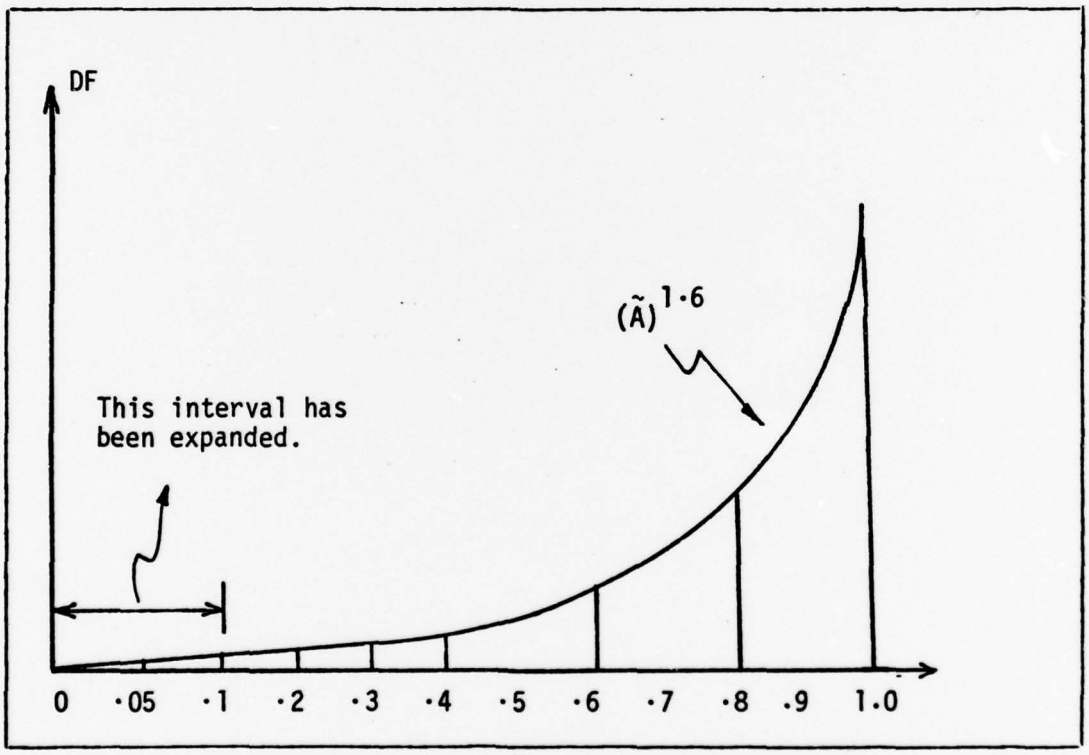


Figure 47. Damage Factor Versus $\bar{A}$ Curve with the Selected Intervals for Curve Fitting Program (8 Intervals)

The flow diagram for the subroutine to compute the damage factor is given in Figure 48; this subroutine is called POLYN. Note that the

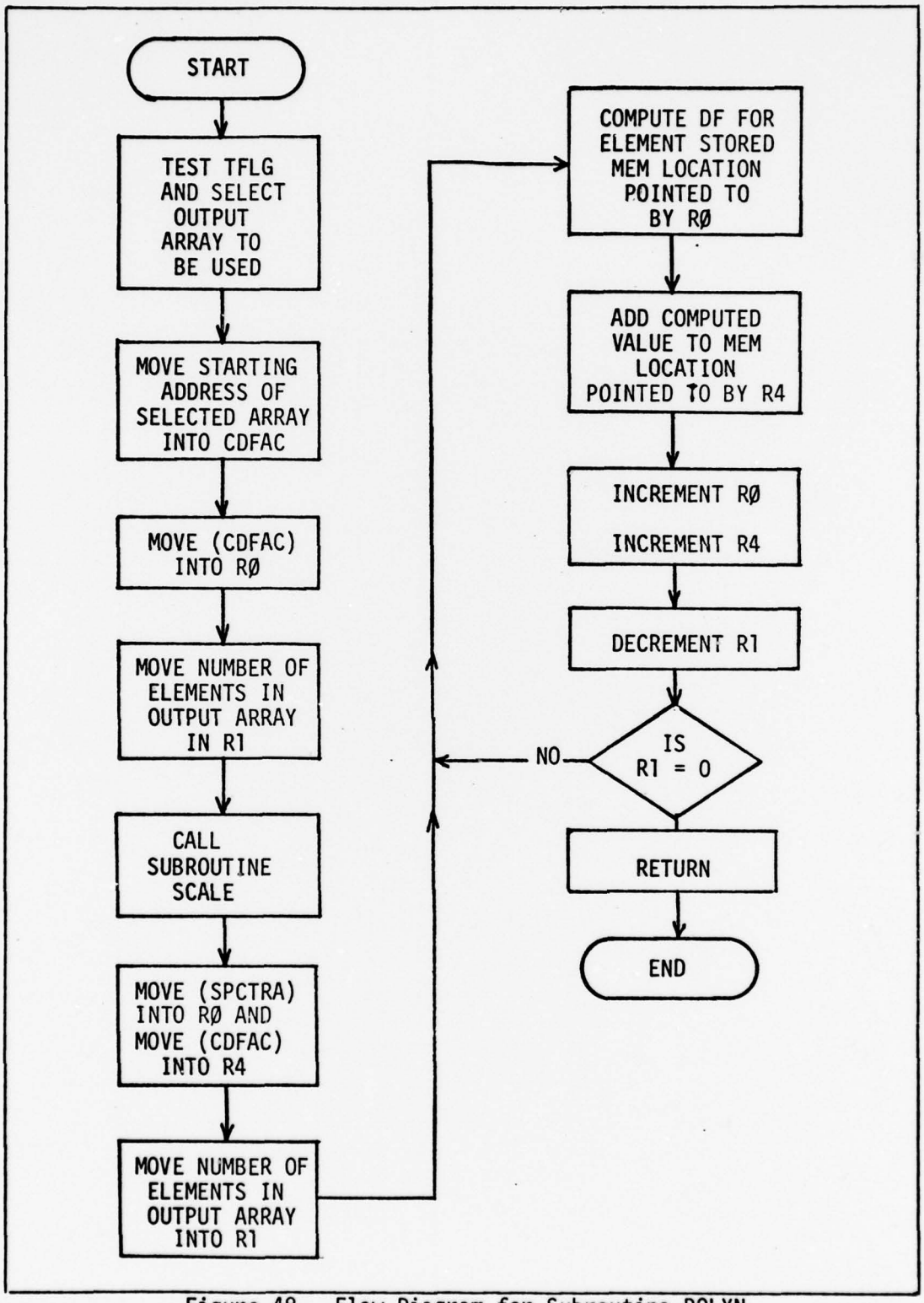


Figure 48. Flow Diagram for Subroutine POLYN

resulting DF values are added to the existing elements of the selected output array (Ref Chap III). This is done to get the CDF as a function of temperature and frequency. The assembly language code for subroutine POLYN is given in Appendix G. This appendix also gives an analysis of execution time for this subroutine.

Control Executive (EXEC). The Control Executive is the systems program that handles the execution of all the modules in MIBDAPS. The control and synchronization of different activities is also handled by this program.

The state transition diagram that shows the initial switching on sequence is given in Figure 49. Once the power is switched ON to the system, the microcomputer executes its own interval microcode and jumps to location $(17300)_8$ where if finds a WAIT instruction. Once the POWER OK or POWER FAIL interrupt occurs, the service routine for either of these interrupts is executed. In case the POWER ON sequence is successful, the execution of the system starts.

Figure 50 shows the flow diagram for the continuous execution of the system. The acquisition of data is done using the interrupt feature of ADV11-A. Therefore, the data acquisition occurs while the processing of previously acquired data is going on. The real time clock would be used for synchronization for the entire data acquisition and processing cycle, but this feature has not yet been included in the flow diagram. The "dashed" block in Figure 50 (page 94) would accomplish the synchronization of different processing cycles.

The assembly language code for EXEC is given in Appendix H. The execution time analysis is also included in Appendix H.

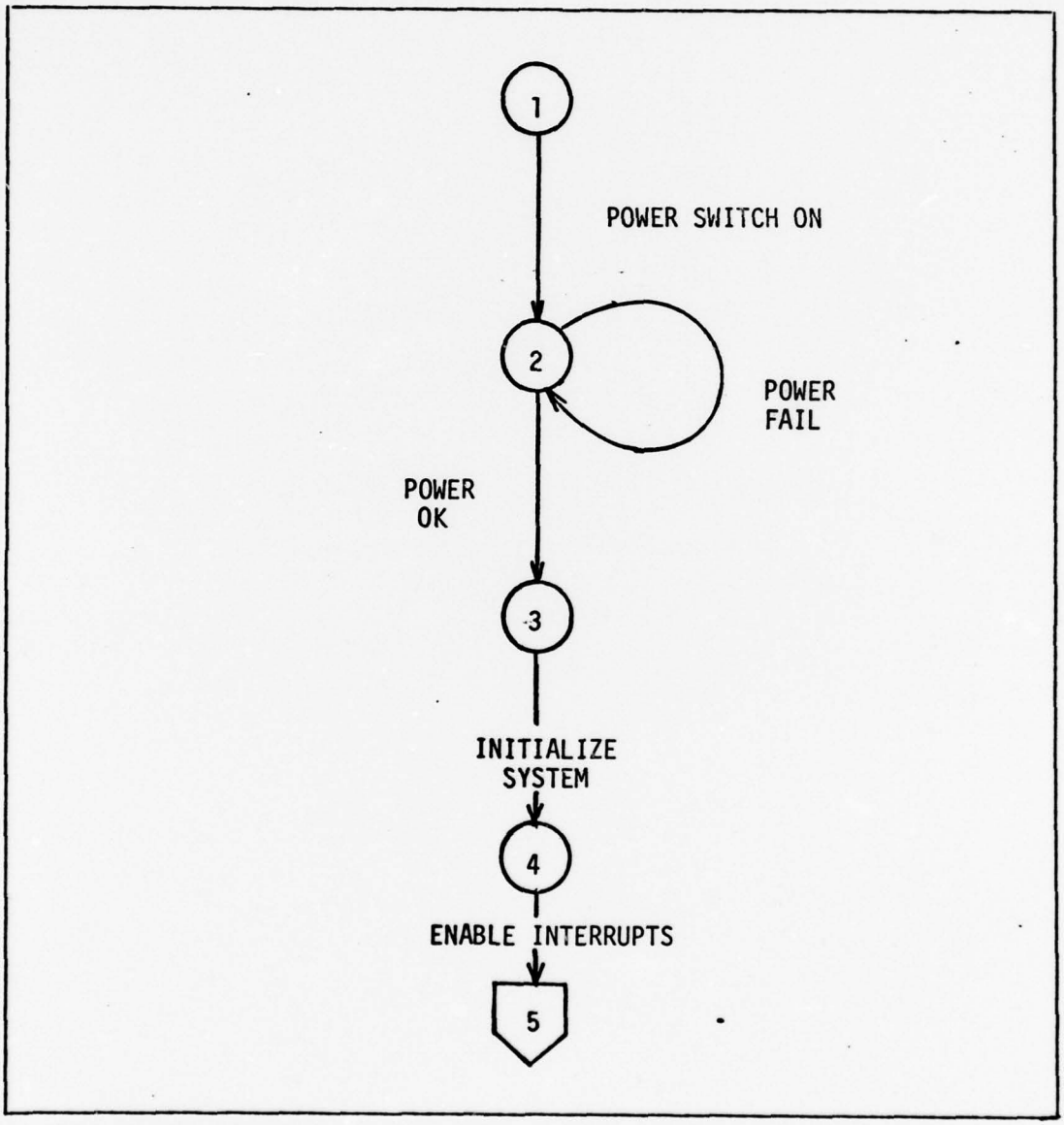


Figure 49. State Transition Diagram for Initial Power ON Sequence

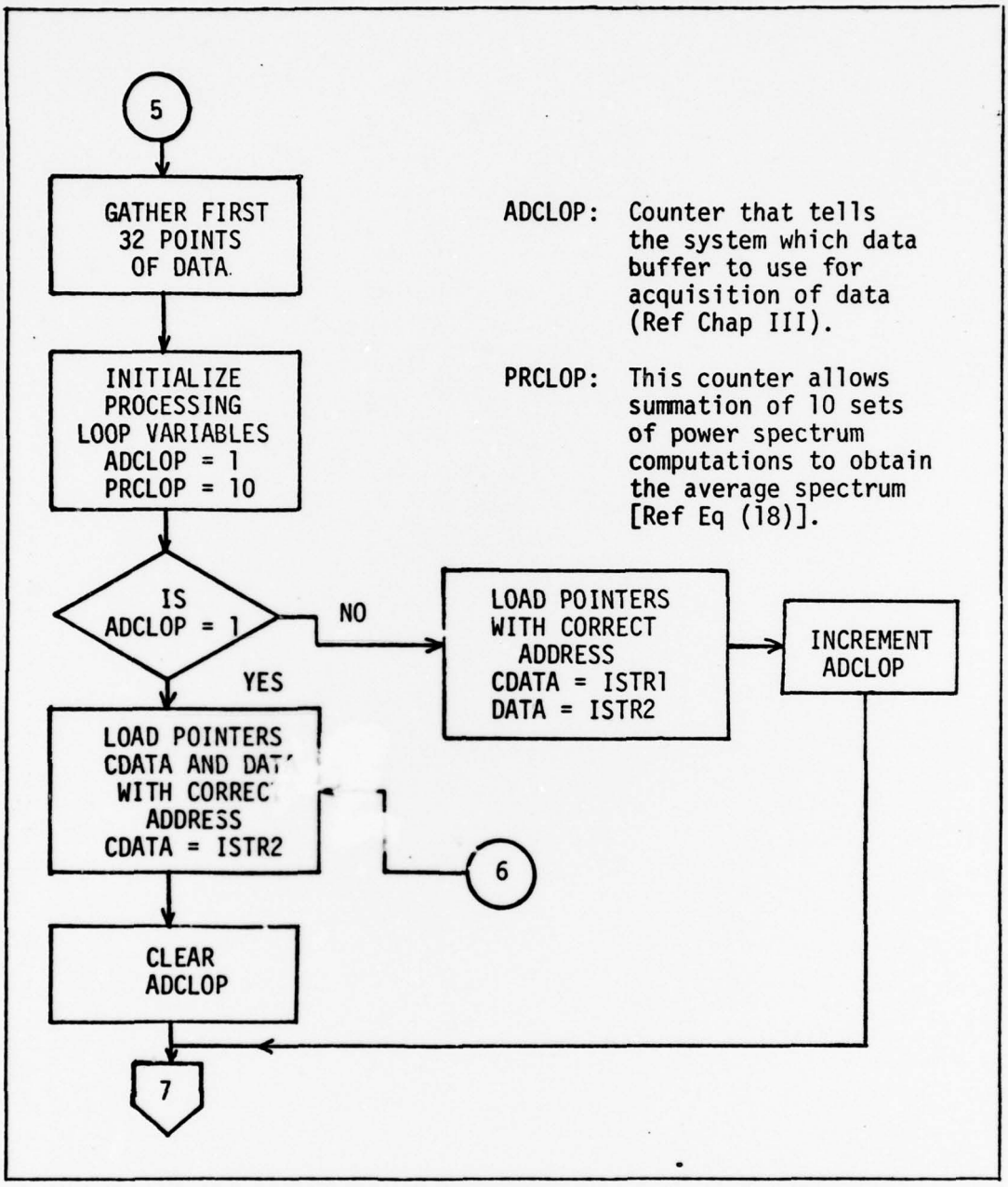


Figure 50. Flow Diagram for Continuous Execution

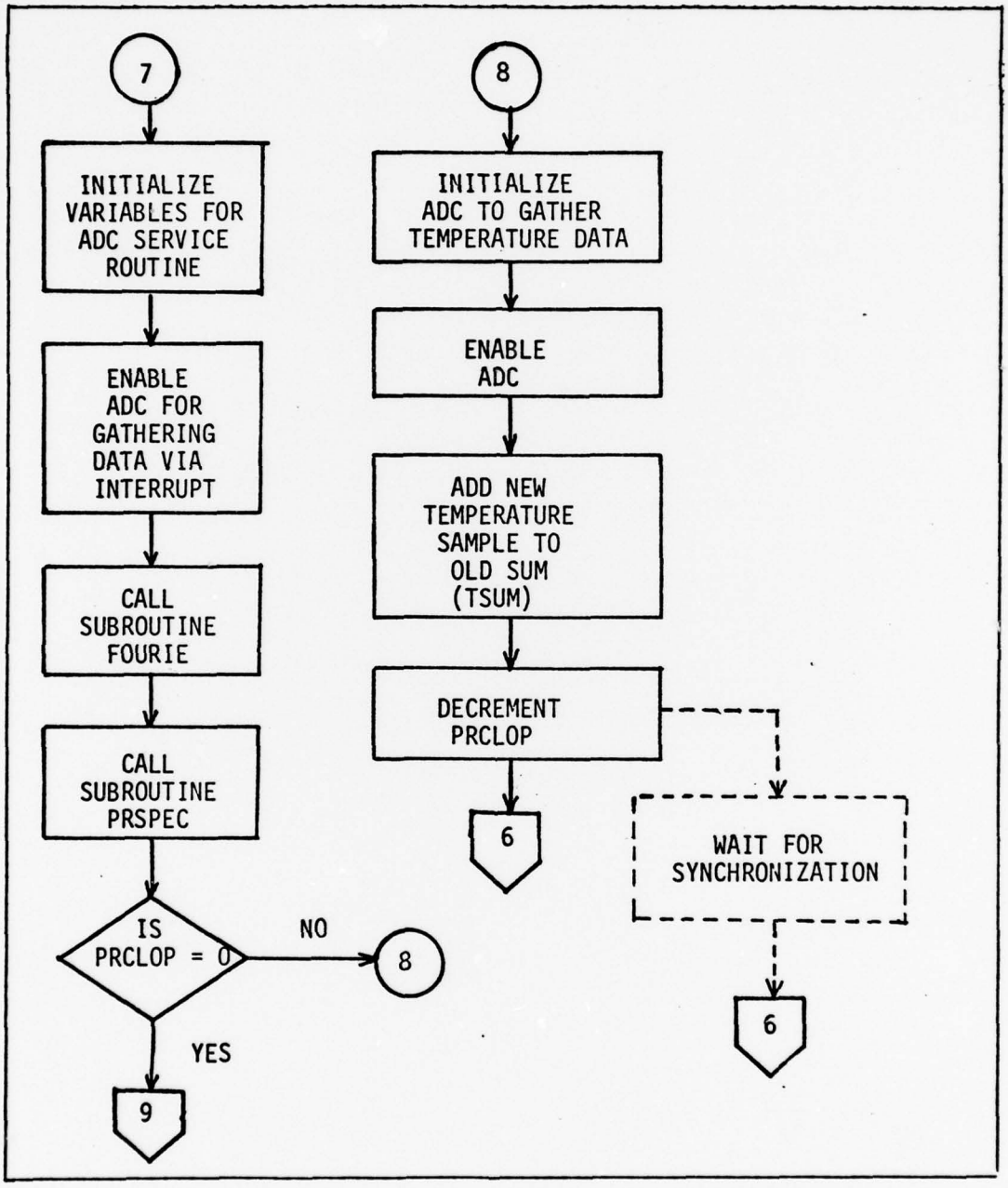


Figure 50. Flow Diagram for Continuous Execution (cont)

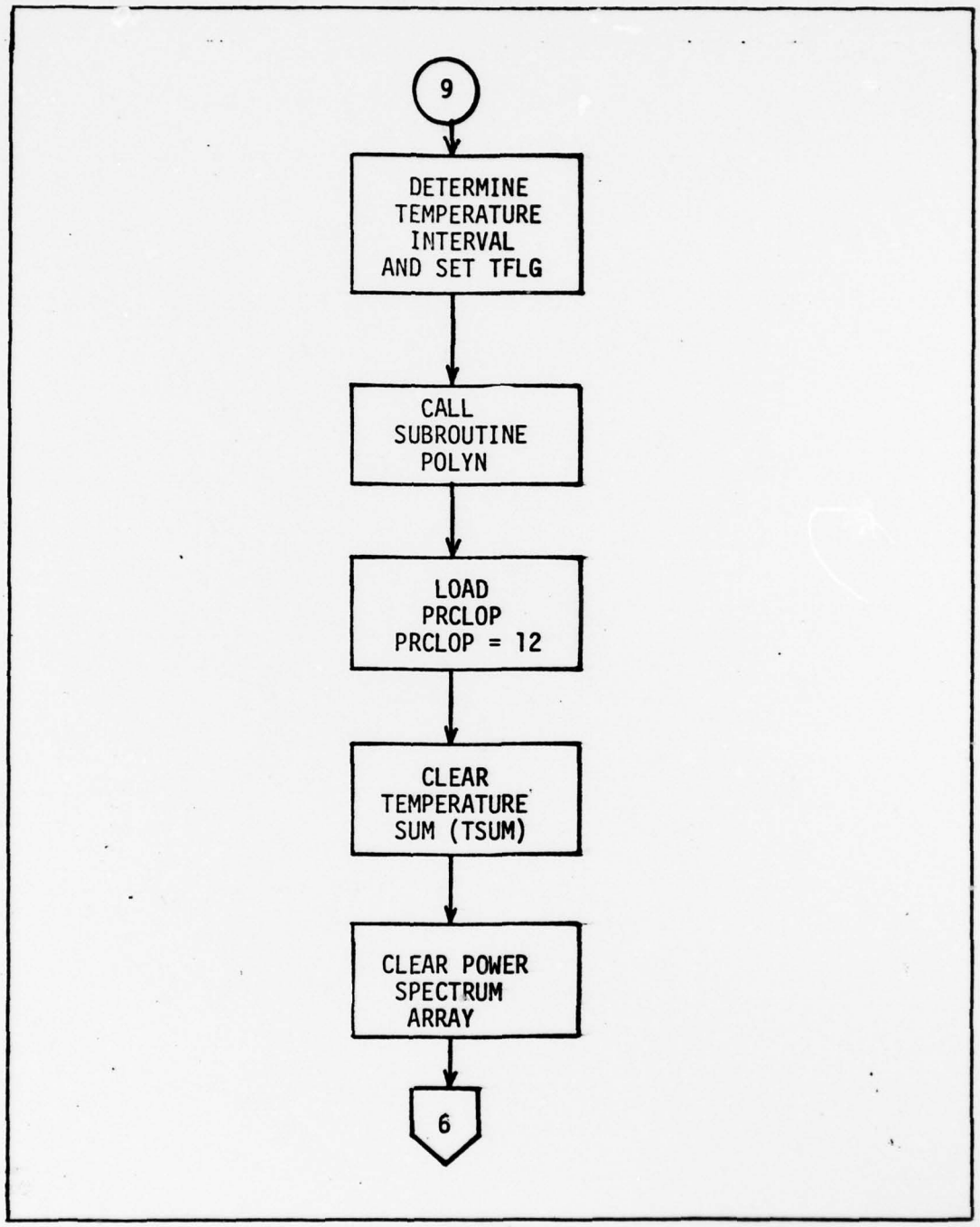


Figure 50. Flow Diagram for Continuous Execution (cont)

Summary

In this chapter, the mathematical analysis of data acquisition and processing functions of MIBDAPS was presented. The theoretical background necessary for the development of each software module was discussed. This was followed by the generation of a flow diagram and assembly language code for each software module. The design of hardware realizable modules of MIBDAPS is undertaken in the next chapter.

V. Design and Implementation of Hardware

Introduction

The design and implementation of software realizable modules of MIBDAPS was presented in Chapter IV. The next step in the development of MIBDAPS was the design and implementation of hardware realizable modules (Ref Chap III). In this chapter, the general design approach is discussed and the implementation of the designed circuits using standard off-the-shelf integrated circuits is explained. At the end of this chapter, the DC power supply requirements are discussed and the use of standard DC power supply modules is suggested. The author's conception of the prototype module of MIBDAPS along with its physical dimensions is also presented.

Design of Hardware Circuits

The design of different hardware realizable modules is described in the following paragraphs. The design is based on the functional requirements spelled out in Chapter III.

System Clock Generation Circuit. The data acquisition function of MIBDAPS required that the vibration analog signal be sampled at a frequency of 3200 HZ (Ref Chap IV). Therefore, to gather data at this sampling rate, it was necessary to trigger the ADC (ADV11-A) by a 3200 HZ external clock (Ref 11:335-338). The same clock was also required to synchronize different processing loops during system execution (Ref Chap IV). Figure 51 is the schematic that shows the hardware configuration where the system clock was used. It should be noted that the output of the clock is fed to the RTC before being applied to the ADC. This has

been done to ensure that the clock is conditioned by the Schmitt trigger circuit on the RTC board (Ref 11:211-212) before being applied to the ADC.

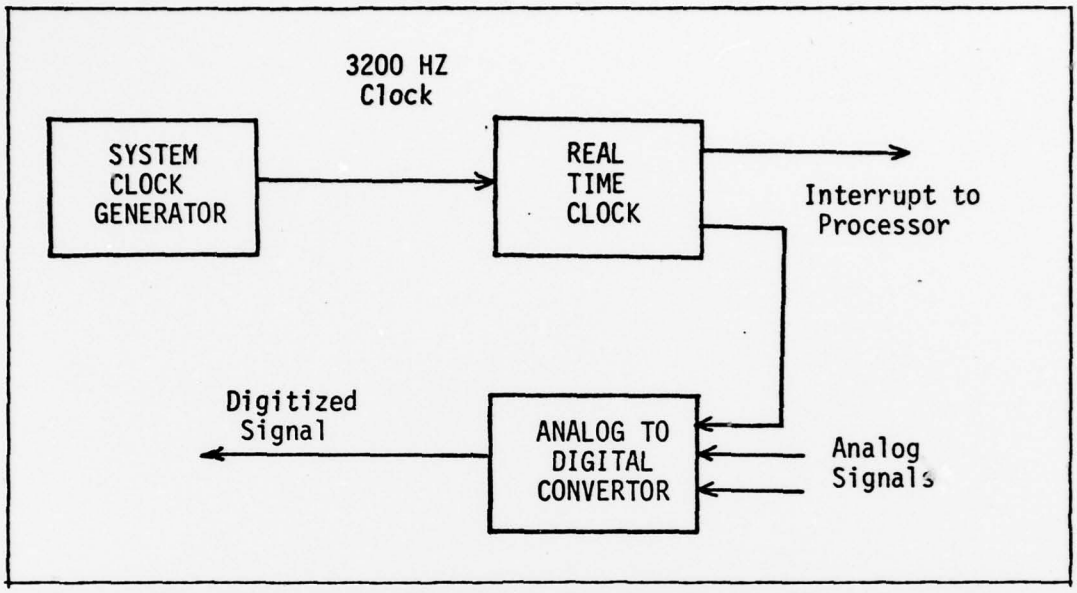


Figure 51. Hardware Configuration Showing System Clock Generator

To design a stable 3200 HZ clock, SN74LS424 clock generator IC and a 64K HZ crystal were used as the reference oscillator (the choice of crystal frequency was arbitrary). This reference frequency was divided down to get the required 3200 HZ clock. Two SN7490 ICs were configured to accomplish the appropriate frequency division. Figure 52 shows the circuit diagram of the designed clock generator.

POWER ON Interrupt Circuit. The LSI-11 bus foundation module (DRV11-P), which is manufactured by DEC, can be used for user defined interfaces (Ref 11:169). This module is supplied with the logic necessary for interfacing to the LSI-11 bus. This logic includes bus transceivers, a device address comparator, protocol logic, interrupt logic, vector address comparator, and bus receivers and invertors (Ref 11:169). The

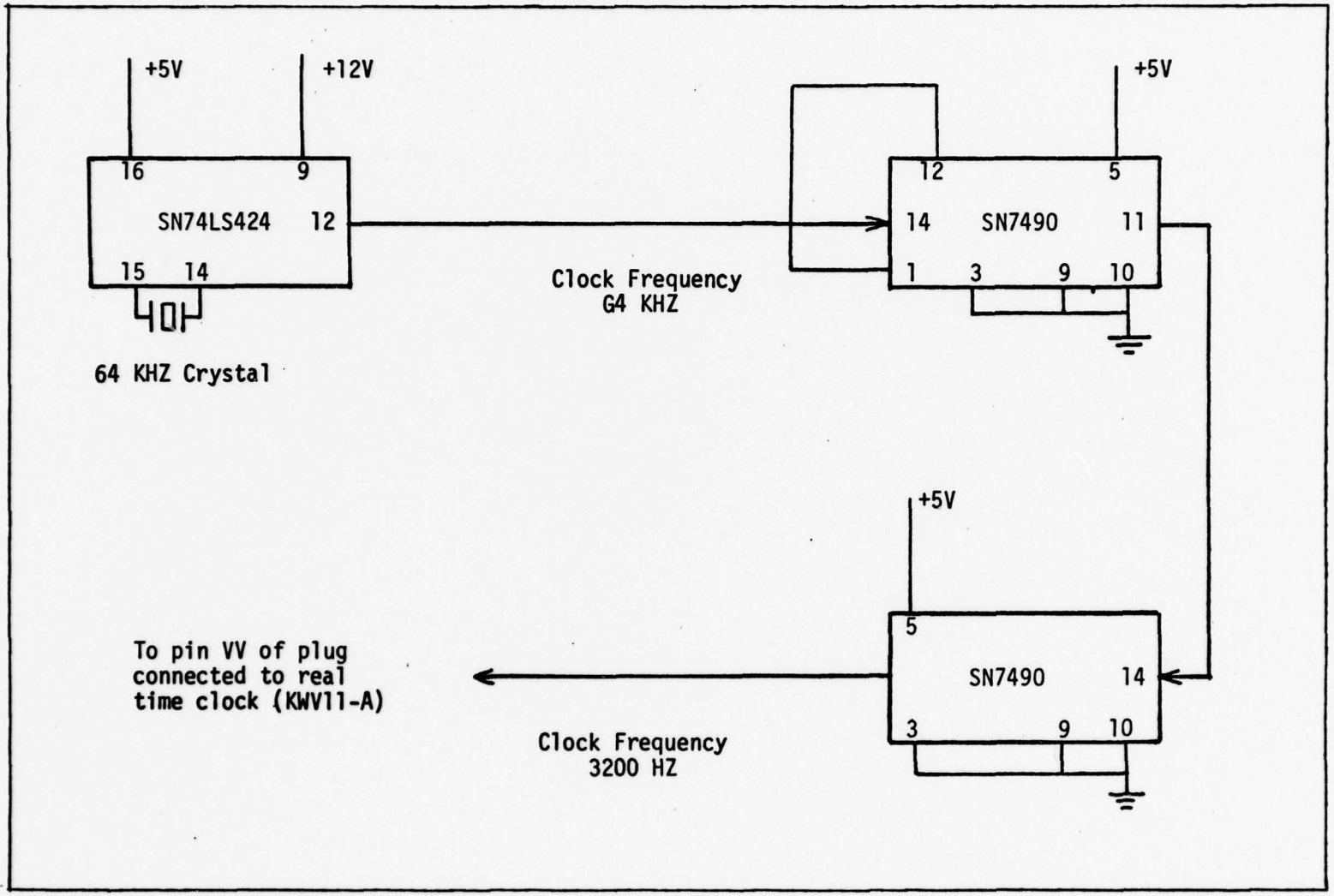


Figure 52. Circuit for System Clock Generator

device interrupt generator and vectors address generation feature of this module were used in the design of the POWER ON interrupt circuit.

The POWER OK interrupt occurs whenever the DC power to the system has been switched ON and +5V, +12V DC supplies are within acceptable limits. The POWER ON transition and a signal from the power sense circuit would be used to generate the requisite sequence of signals necessary to activate the interrupt and vector address generator feature on the DRV11-P. (The reader is referred to Ref 11:177 for an understanding of the logic requirement placed on the requesting device.)

The block diagram of the designed circuit is given in Figure 53.

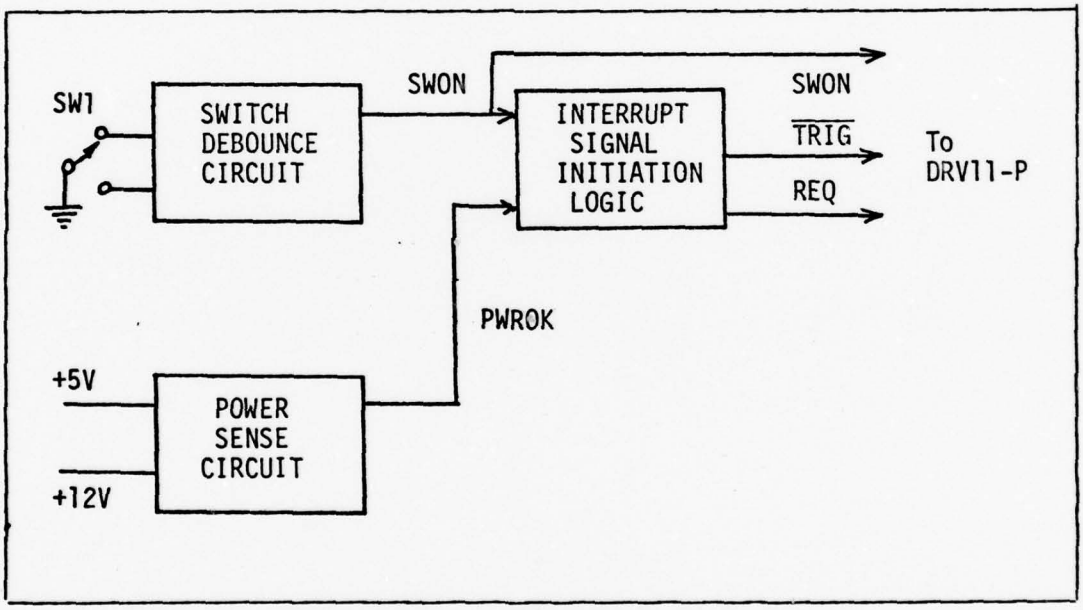


Figure 53. Block Diagram of PWROK Interrupt Generator

This circuit assumes that the +5V supply is available to itself and the DRV11-P module before the POWER ON switch (SW1) makes the OFF-to-ON transition. This assumption would create a problem if there is only one power ON/OFF switch in the system. The solution to overcome this problem

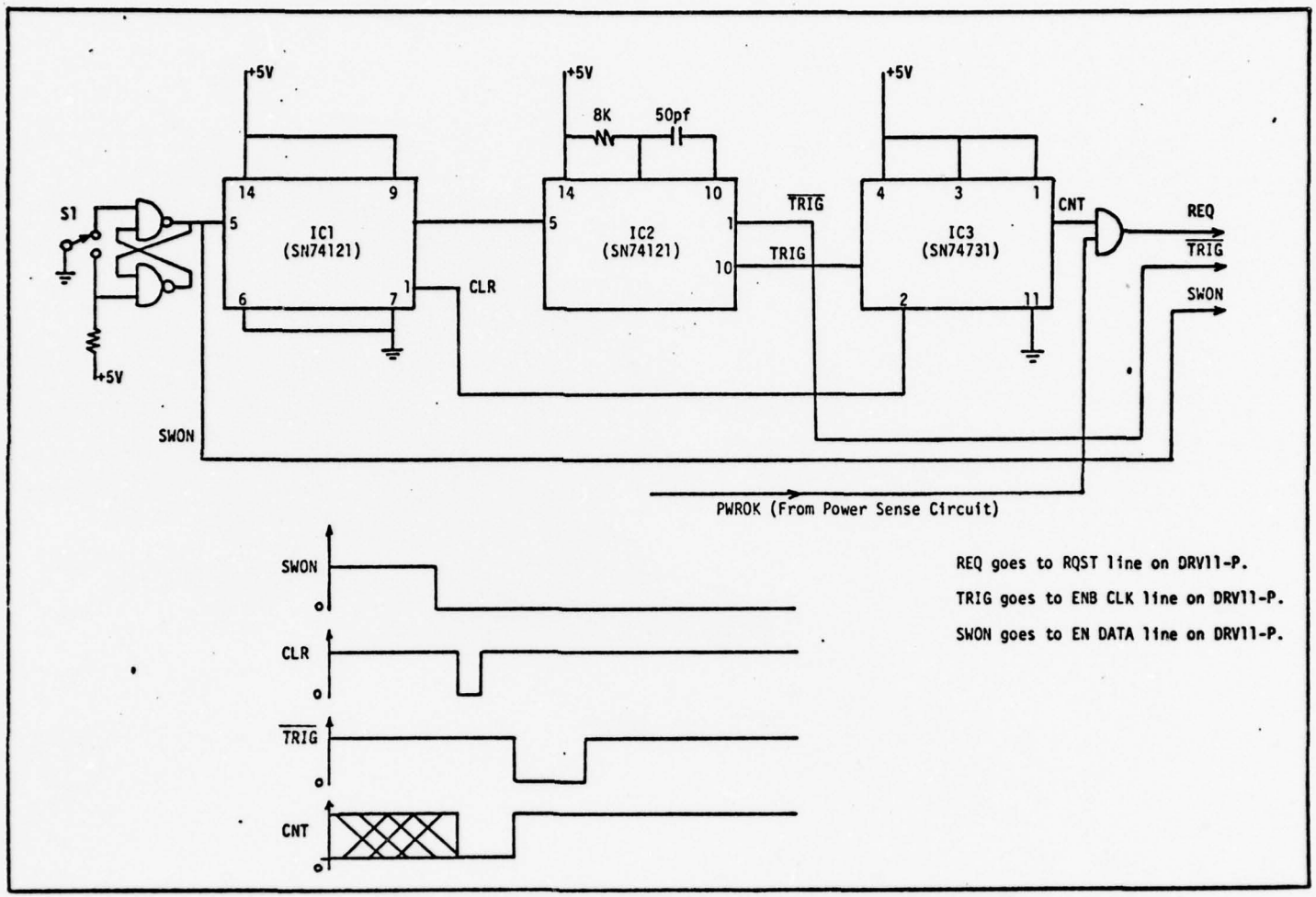


Figure 54. Circuit Diagram for POWER OK Interrupt Generator

is to have a RUN/STANDBY switch, which is put to the STANDBY position before the POWER ON/ORR switch is switched ON. The RUN/STANDBY switch is then switched to the RUN position, which would generate the appropriate signal sequence for the interrupt and interrupt vector generation logic on the DRV11-P. Figure 54 shows the designed circuit diagram along with the timing diagram of the logic signals generated.

POWER FAIL Interrupt Generator. This circuit would be used to generate an interrupt whenever a power failure occurs or when the RUN/STANDBY switch makes a RUN-to-STANDBY transition. Figure 55 shows the block diagram of the hardware configuration required. This circuit was not designed due to the shortage of time. The DRV11-P could be used to generate the POWER FAIL interrupt; however, a logic circuit to initiate the interrupt request and interrupt vector cycles would have to be designed.

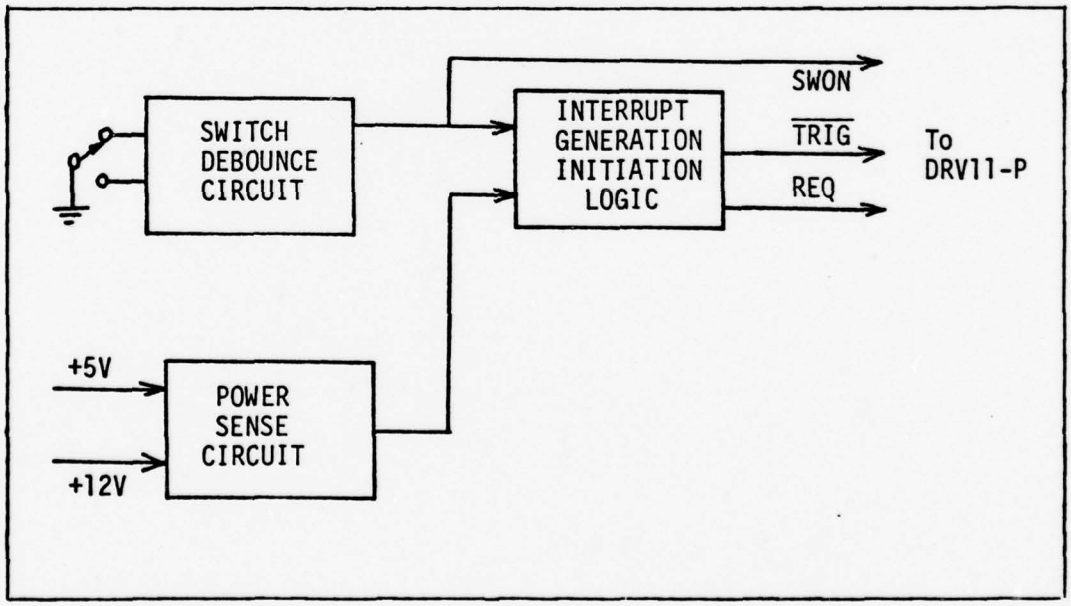


Figure 55. POWER FAIL Interrupt Generation Logic

Operational Time Recorder. The block diagram for the operational time recorder is given in Figure 56. The system clock is fed to this

circuit, and it keeps track of the time for which the system remains operational. Whenever the operational time reaches the 15-hour limit, an interrupt is generated and the system executives a "halt" cycle.

The operational time is also displayed on the front panel. In case either a power failure occurs or the system is switched to STANDBY mode, the current operational time would be stored in the core memory. This is done to obtain the total operational time for which the system monitors a test piece in case the system was operated in different time segments.

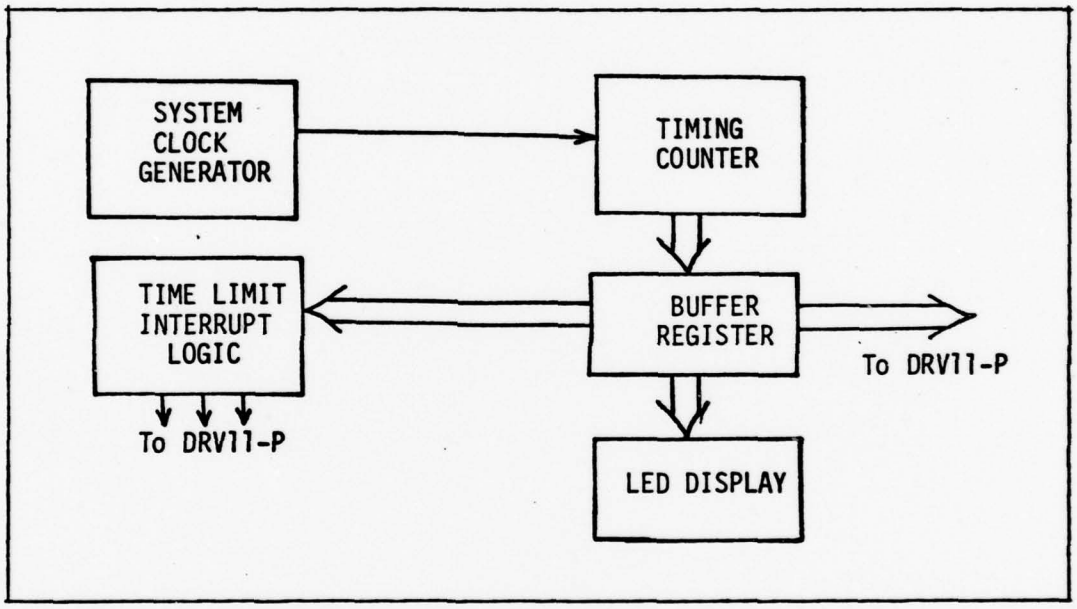


Figure 56. Operational Time Recorder

Power Supply Requirement

The power requirement for each hardware module used in MIBDAPS is tabulated in Table IV. The +5V DC supply is required to source a maximum current of 16.63A, and the +12V is required to source 2.21A maximum. A comparative analysis of the +5V and +12V DC supplies available from different manufacturers was done. The following characteristics were compared during this analysis: maximum output current, load regulation, ripple and

noise, case size, and weight. Based on this analysis, model numbers HE 252 and HE 212, the +5V DC and +12V DC supplies, manufactured by Computer Products Incorporated, are the recommended DC power supplies for MIBDAPS. Table V lists the salient features of these recommended supplies; it should be noted that the maximum rated current for the supplies is more than the required maximum current for MIBDAPS (see Table IV). This would enable future additions of the required hardware modules.

Mechanical Description of MIBDAPS

The author's conception of the final shape of MIBDAPS (called MIBDAPS-I) is given in Figure 57. The detailed diagram showing the layout of different subunits and their dimensions is given in Figure 58. It should be noted that the rear view shows an AC inlet socket and an RS-232 outlet. The RS-232 would be provided to connect the system to a TTY, which would be used as an I/O device to get a printout of CDF versus frequency and temperature arrays.

Summary

In this chapter, the design and implementation of two hardware modules was presented along with the block diagram descriptions of the other two hardware modules which need to be designed. The power supply requirements was determined and a recommendation for off-the-shelf DC power supplies was made.

The physical dimensions for the mainframe and the location of subunits within the mainframe were made. Diagrams depicting the physical chape and dimensions of MIBDAPS have also been drawn. Integration of the system was accomplished next. The results of testing the integrated system are discussed in the following chapter.

TABLE IV

Power Requirement for MIBDAPS

Description of Modules	Current Required From +5V Supply (nominal/maximum)	Current Required From +12V Supply (nominal/maximum)
LSI-11M Processor	1·8A/2·4A	0·8A/1·1A
DLV11 Serial Line Unit	1·6A/1·6A	0·18A/0·25A
ADV11-A A/D Convertor	2·0A/2·0A	0·45A/0·45A
MMV11-A 4K Core Memory	3·0A/7·0A	0·6A/0·6A
REV11-C Bootstrap ROM and Terminator	1·0A/1·88A	--
KWV11-A Real Time Clock	1·75A/1·75A	0·01A/0·01A
Total current requirement for +5V supply = 11·15A/16·63A. Total current requirement for +12V supply = 2·04A/2·21A.		

TABLE V

Recommended Power Supply Characteristics

Description	Output Current (maximum)	Load Reg. Maximum (NL-FL)	Ripple and Noise	Case Size (in.) L x W x H	Weight (lbs.)
+5V HE 252 (110V AC output)	20·0A	±0·1%	50MV P-P (13 MV RMS)	6·5x4·5x	3·25
+12V HE 212 (110V AC output)	3·0A	±0·1%	20MV P-P (2MV RMS)	6·5x4·5x 1·8	1·7

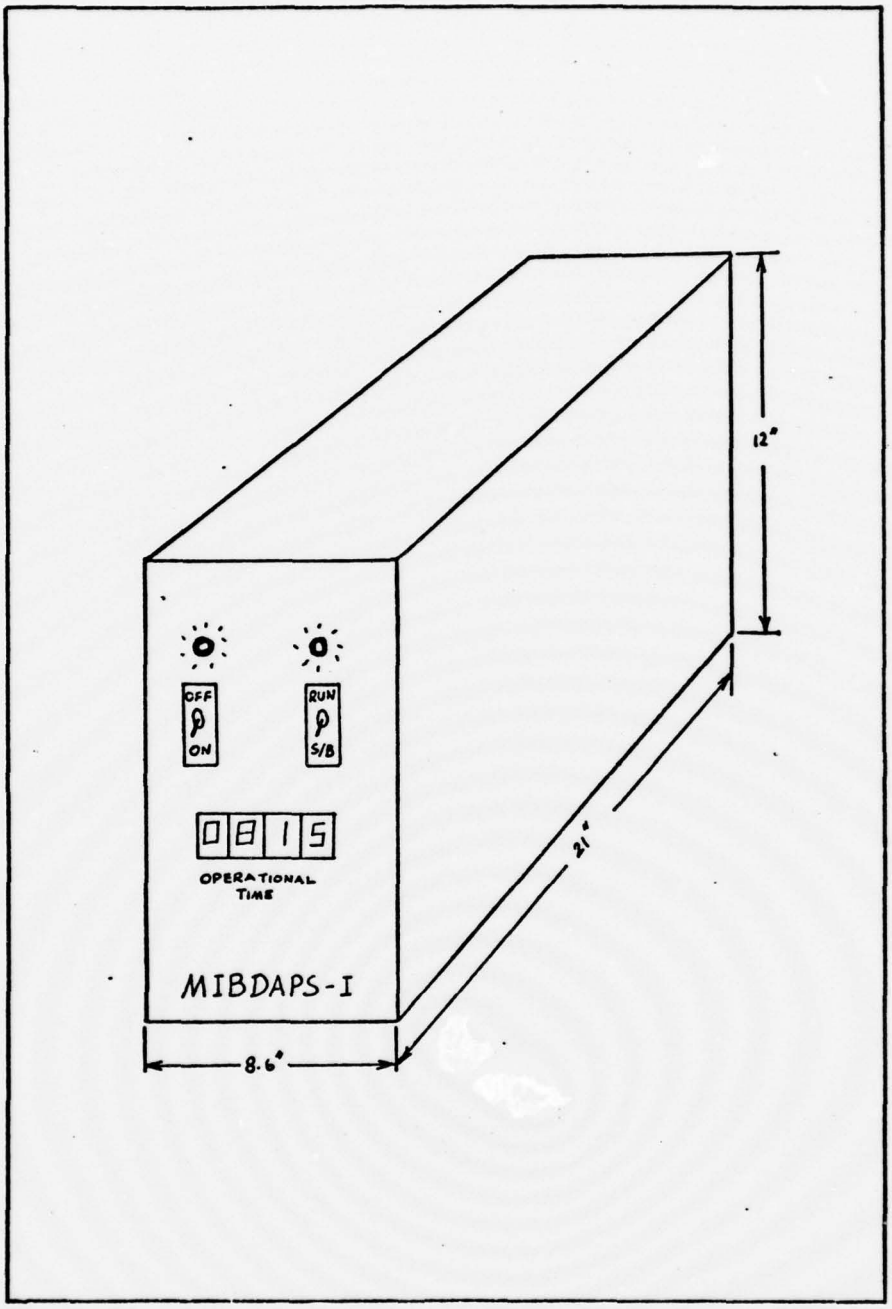


Figure 57. Author's Conception of the Final Shape of MIBDAPS

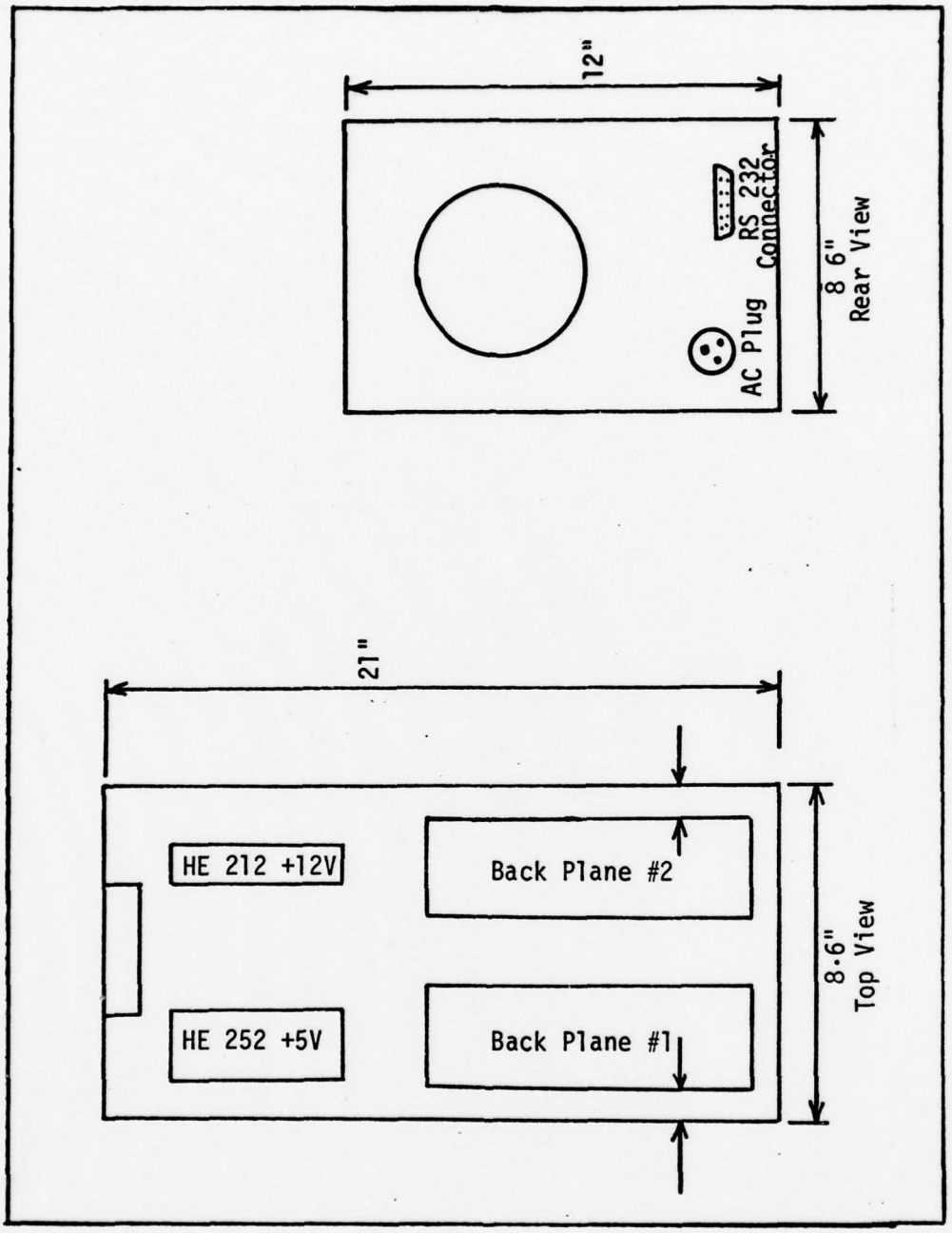


Figure 58. Location of Subassemblies in the Mainframe

VI. Testing and Experimental Results

Introduction

The testing of the software during the development stage, the integration of hardware and software, and the testing of the integrated system are discussed in this chapter. The experimental setup used in both the software development stage and the system integration stage is explained, and the results of the experimentation and software developed for testing are presented at the end of the chapter.

Software Debugging and Testing

The software algorithms that were designed and coded in Chapter IV were debugged and tested on a PDP-11/03 minicomputer system. The configuration of the development system used is given in Figure 59. A three-step approach was taken to develop software. First, the assembly language code was assembled and syntax errors were removed. Second, the logical errors detected during program execution were debugged. And, lastly, the testing of software for computational accuracy was done. The flow diagram given in Figure 60 shows the complete software development cycle.

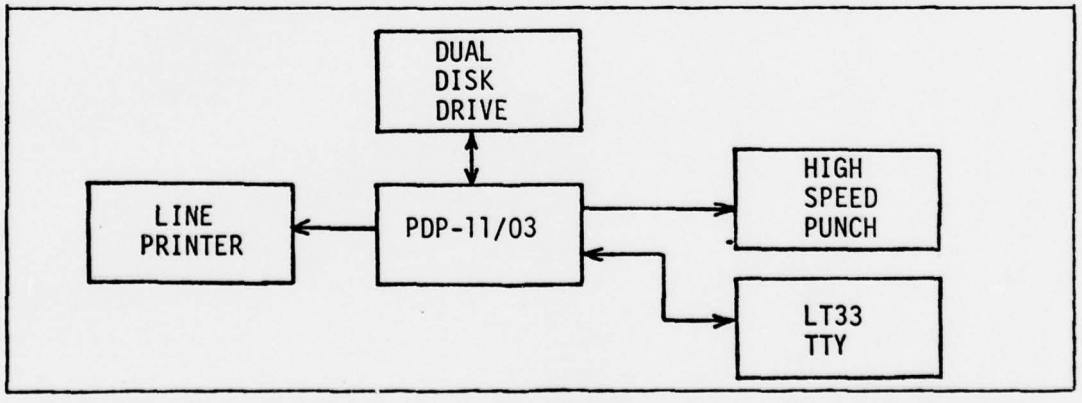


Figure 59. Configuration of the Software Development System

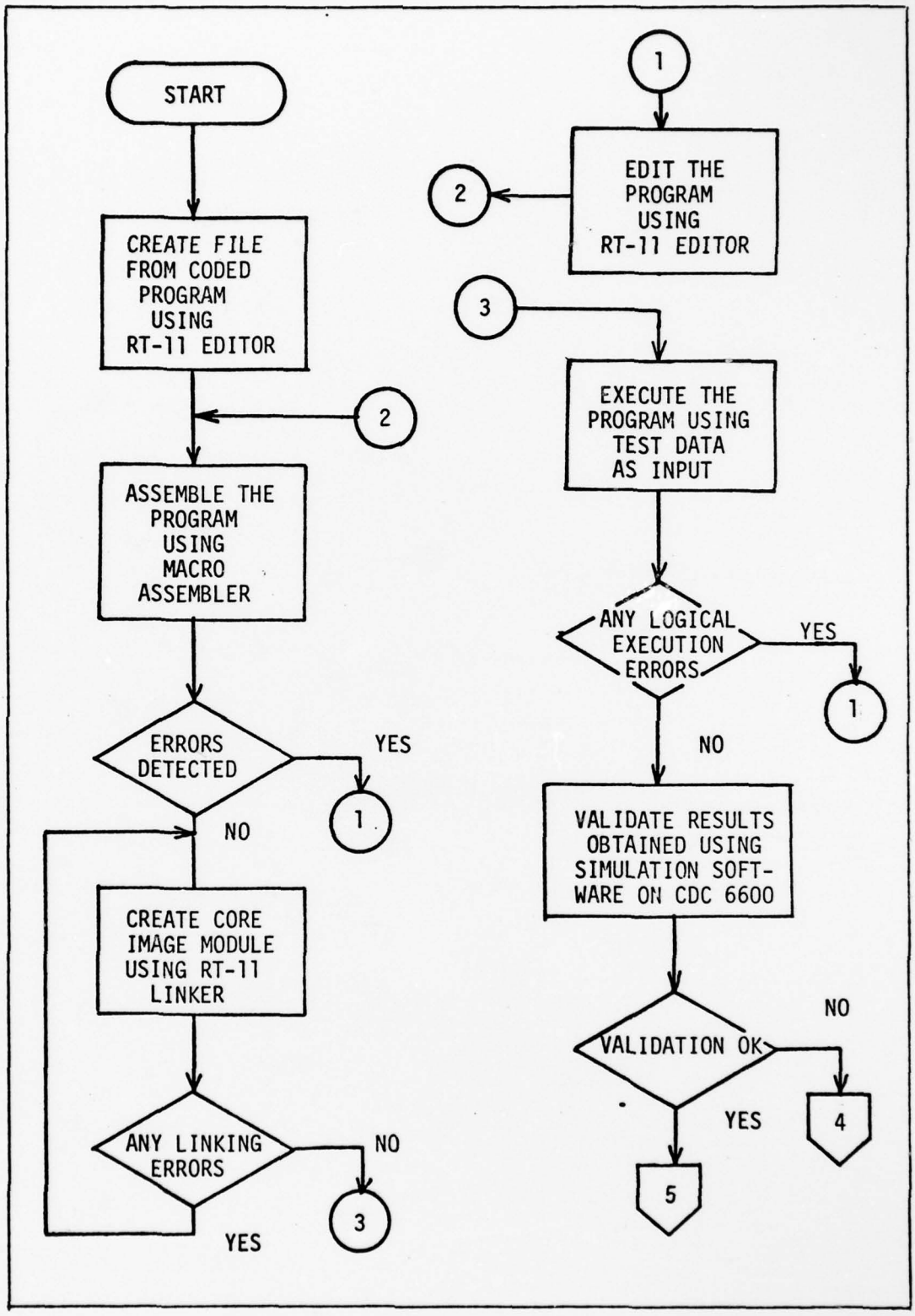


Figure 60. Software Development Cycle

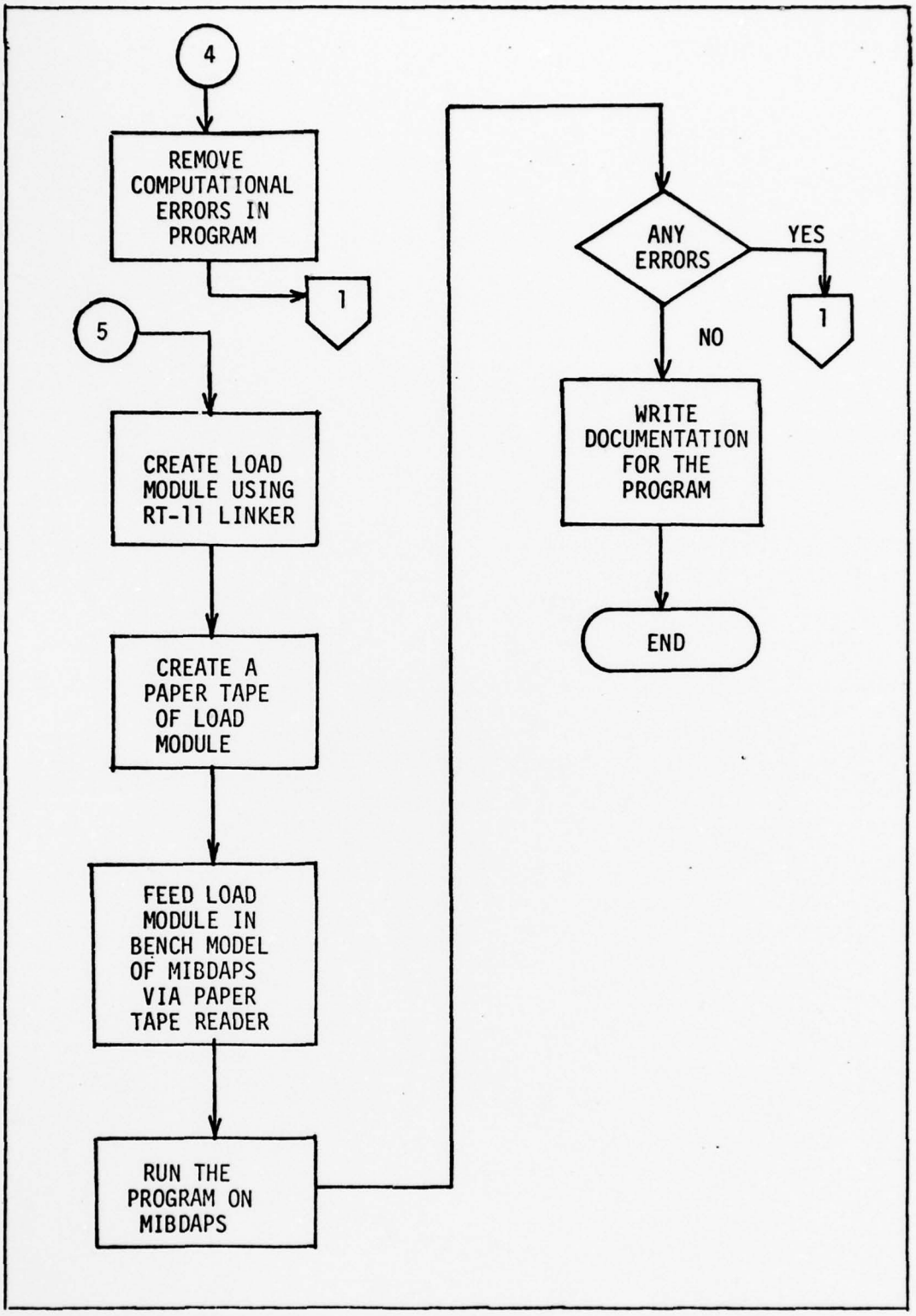


Figure 60. Software Development Cycle (cont)

Testing of Software Modules. After the syntax/semantics and logical error had been removed, the testing of each software module was undertaken. The approach followed was to write FORTRAN simulated programs which duplicated the computations performed by the software module being tested. These simulation programs were executed on the CDC 6600. The input data to the simulation programs was similar in format and value as that used during the execution of the actual software module. The results obtained from the simulation were compared with earlier results from the system software and, in case of discrepancies, the system software code was checked for computational errors. Appendix J gives the simulation program used to check the subroutines FOURIE, PRSPEC, and POLYN (Ref Chap IV). The test data used and the results obtained from the simulation programs and system programs are also tabulated for comparison (Ref Appendix J). These results are discussed at the end of this chapter.

Integration of Software and Hardware

The system integration was done once the software modules had been developed and tested. The system integration phase required setting up a bench model for MIBDAPS and testing the control execution. The timing constraints for both data acquisition and processing were also determined at this stage.

Bench Model for MIBDAPS. The configuration of the bench model for MIBDAPS is given in Figure 61. The function generator was used to simulate the vibration signal, and the square wave generator was used as the system clock. The bench model was tested by running the control executive program which exercised the data acquisition and processing function

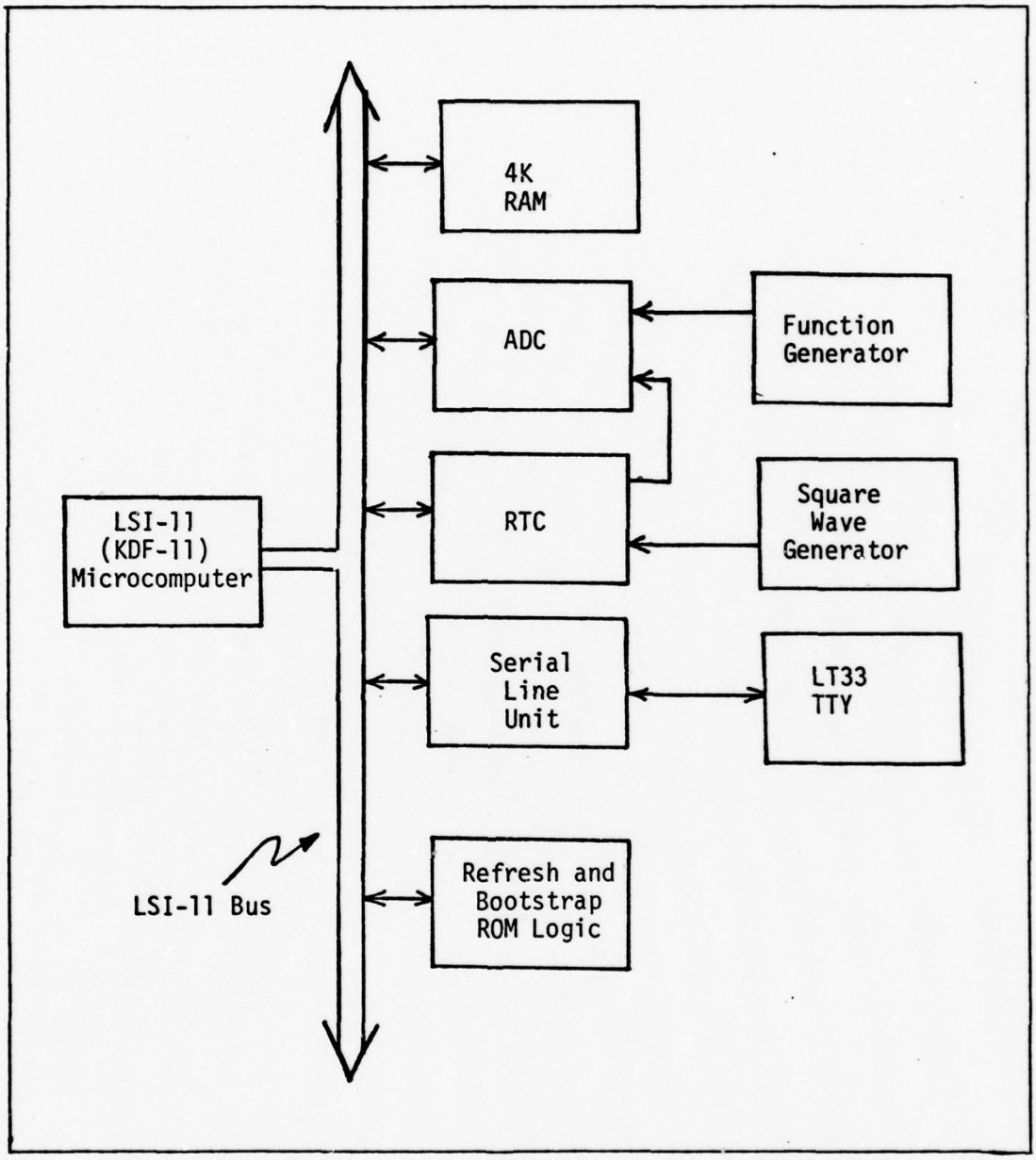


Figure 61. Configuration of the Bench Model for MIBDAPS

of MIBDAPS. To check the results at different stages of acquisition and processing, a decimal print-out subroutine called DECPR was included; this subroutine was called by a macro called DECMAC. The source listing for subroutines DECPR and macro DECMAC is included in Appendix I. The final source listing for the control executive, which is given in Appendix H does not contain this print-out feature because during the actual operation of MIBDAPS the intermediate results are not required to be printed. The results of the control executive operation are discussed later in the chapter.

Results of the Experimentation

The result obtained from testing individual software modules and the operation of the control executive are presented in the following sections.

Software Test Results. The results of software testing are presented as the root means square (RMS) error between the output from the simulation program and the output from the actual software module. These results are given in the tables in Appendix J. The RMS error criterion was used because an estimate of the average error for each software module was desired. The mean of average RMS error is .0387 (see Appendix J). This error is acceptable because it is much less than the specified tolerance of $\pm$10% (Ref 13).

Execution Results for Control Executive. The two aspects that are important from the functional standpoint of MIBDAPS are the execution time for the acquisition and processing loop and the accuracy of time computations.

The execution times for each software module were computed separately (Ref Chap IV), and the total execution time for the control

executive was computed using the flow diagram given in Figure 62. The loop that computes the power spectrum for a new set of data points takes 29·391 msec, and the time taken to compute the damage factor once the average power spectrum has been computed (Ref Chap IV) is equal to 4·596 msec. Figure 63 shows the timing relationship between the actual acquisition and processing time and the desired time as defined in the initial requirement (Ref Chap I). The designed system, therefore, would update the CDF versus frequency and temperature array every 298·48 msec. For a new set of 32 data points, the power spectrum computation takes 29·36 msec; the average over 10 such intervals would yield a good estimate of the power spectral density of the input vibration signal. This estimate would not be in error as compared with the average power spectral density obtained if the power spectrum computed had taken less than 10 msec (as originally stipulated). This statement is true because the input vibration signal can be assumed to be wide-sense stationary (Ref 13), and the estimate of the average power spectral density of such a signal depends on the number of power spectrum samples that are averaged and not on the rate at which these samples are available (Ref 6).

Interpreting the Output Array. The CDF versus frequency and temperature array would contain fractional binary numbers that have to be correlated with actual physical data. An analysis of the scaling introduced during data acquisition and processing of data is now discussed.

Assume that an accelerometer is used as the vibration signal sensor which gives out a voltage of K1 for every 32 ft/sec^2 of acceleration. (This acceleration is referred to as one "g.") Therefore, the input voltage from the vibration signal is given by the equation:

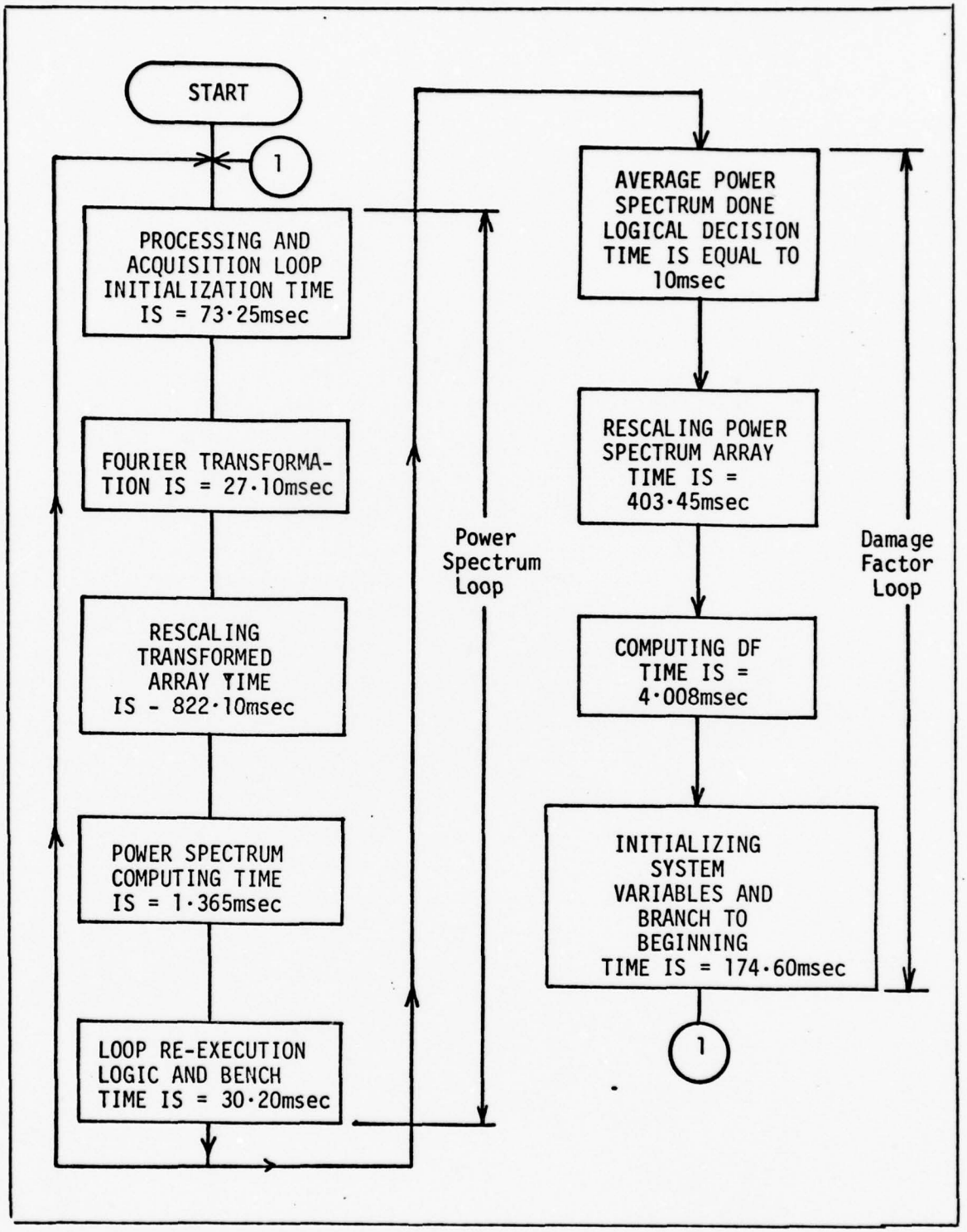


Figure 62. Execution Time for Control Executive

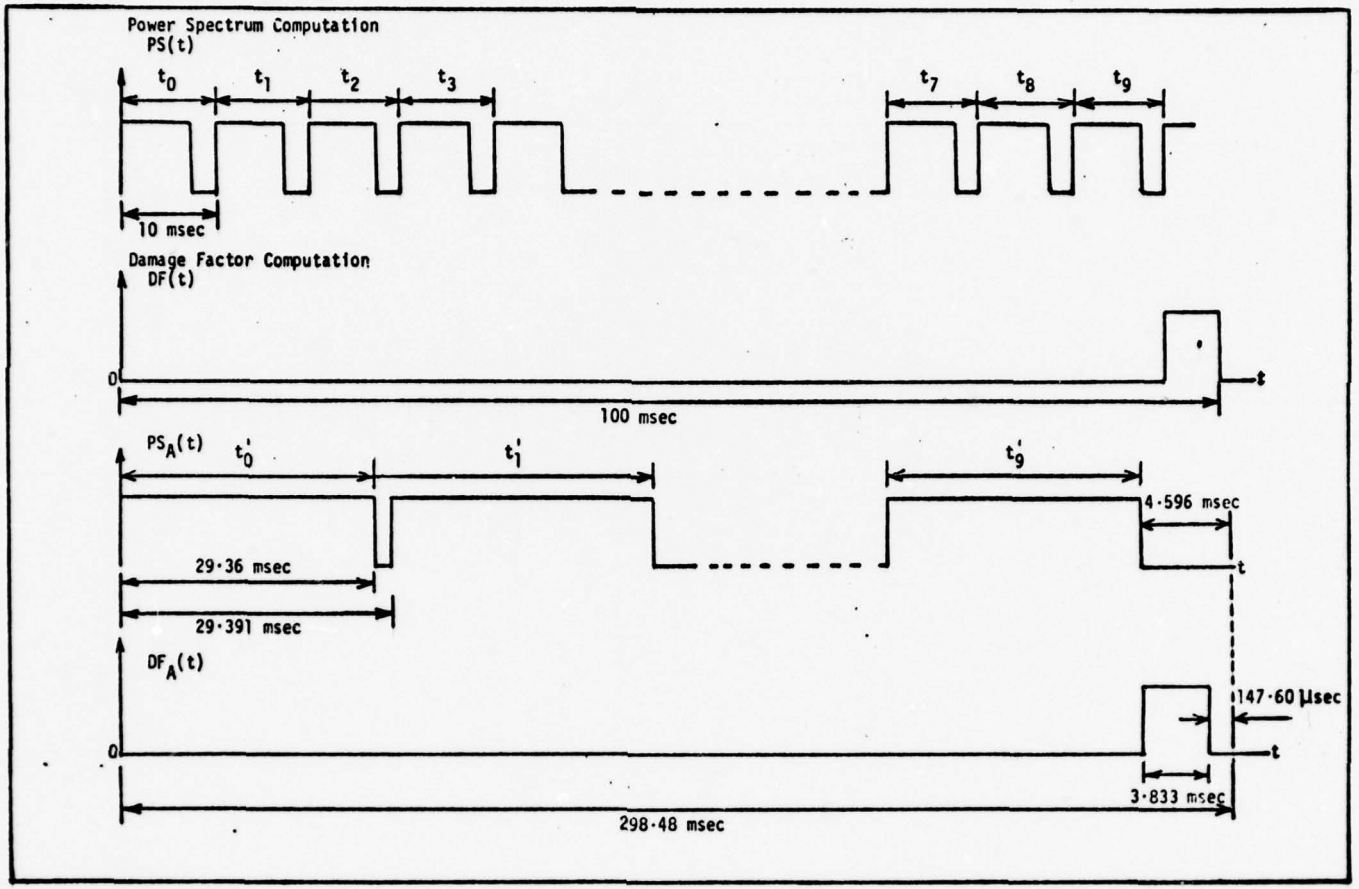


Figure 63. Timing Relationship Between Actual and Desired Execution Times

$$e(k) = K_1 G(k) \quad (51)$$

Here, G(k) is the acceleration encountered by the test piece.

The analog-to-digital convertor, ADV11-A, gives a 12-bit binary representation of the input voltage. Since the computation carried out in MIBDAPS assumes fractional numbers (Ref Chap IV), an input voltage of one volt would be represented by the fraction K_2 which is defined below.

$$K_2 = \frac{(007777)_8 \times 2^{-15})}{5 \cdot 12} \quad (52)$$

Therefore, each data point x(n) is given by the following equation:

$$x(k) = K_2 e(k) \quad (53)$$

The Fourier transformation of these data points yields the equivalent value in the frequency domain. Using Eq (6), we get

$$F(n) = \sum_{K=0}^{N-1} x(k) W^{nk}, \quad n = 0, 1, \ldots N-1 \quad (54)$$

since the Fourier transformation module (subroutine FOURIE) scales the data array to avoid memory overflow (Ref Chap IV). Therefore, the resultant Fourier transformed array is given by the following:

$$Z(n) = (SFACTR)F(n) \quad (55)$$

Here, SFACTR is the value by which the Fourier transformed array has been scaled. (This value in subroutine FOURIE has been fixed equal to 3.)

The power spectrum subroutine PRSPEC introduces another scale factor called CUMSF1 (Ref Chap IV). Also, since an average power spectrum density is computed, Eq (15) can be modified to give:

$$\tilde{A}'(n) = K3\bar{A}(n) \tag{56}$$

where $$K3 = CUMSF1_{/P} \tag{57}$$

and $$A(n) = |Z(n)|^2 \tag{58}$$

The cumulative damage factor computation introduced a scale factor called CDSFi (here i = 1, 2, ...11), and Eq (19) can be modified and rewritten as:

$$CDF(n) = K4[\tilde{A}']^{1.6} \tag{59}$$

where $$K4 = \frac{CDSFi}{[f(n)]^{5.4}} \tag{60}$$

The equations derived here can be used to interpret the output array CDF versus frequency and temperature.

Summary

In this chapter, the testing procedures used to check the software modules as well as the integrated system were discussed. The experimental results obtained for individual software modules and also for the overall system were presented. The scale factors introduced by each software module and the equations necessary to interpret the output array were also derived.

VII. Recommendations and Conclusions

Introduction

At present, a bench model of MIBDAPS exists; the development of a prototype model will require additional work. This effort would include developing software to test the system's accuracy, configuring the hardware in a mainframe, and incorporating the system error detection and recovery features. This chapter presents recommendations for future development and conclusions of the investigation.

Recommendations

The recommendations for future work are listed in the following paragraphs:

- Software to test the performance and accuracy of results should be developed. This software should include provisions to generate input test signals which duplicate real world signals. The software should gather memory overflow statistics for varying operating intervals. These statistics would enable the determination of the fail-safe operating period.

- The software to output the CDF versus frequency and temperature arrays, at the conclusion of the system operation, needs to be developed. This software should either print the output arrays in a tabular form or plot CDF versus frequency and temperature contours. (The availability of a plotter is essential for the latter.)

- At present, the control executive includes a latent code which needs to be activated in the future. This latent code has been written to handle the acquisition of a temperature signal. It would also

compute the average temperature during the interval when the power spectral density is being computed.

- The ADC used in the bench model can handle 16 single-ended inputs. At present, only two inputs are being used for acquiring the analog temperature and analog vibration signals. Therefore, similar data from seven additional test pieces can be acquired and processed. The possibility of increasing the number of test pieces monitored by MIBDAPS should be explored. Changes in existing control executive would be required for handling the additional data.

- The need to minimize the existing MIBDAPS configuration is vital to those applications where size and weight of MIBDAPS is critical (Ref 13). In case this venue is explored for developing the prototype, then the following changes are suggested to the existing system:

a. Use of static RAM in place of core memory would decrease the weight and power requirement.

b. Use of custom-designed dual channel ADC would eliminate the need for the DEC manufactured ADC board.

c. Design of a dedicated FFT hardwired module would decrease the memory requirements for the software. The FFT integrated circuit chip R5601*, manufactured by Reticon, would be a candidate device. If designed, the FFT module should be bus-compatible with the LSI-11M microcomputer and act as a DMA device.

- The DC power sense circuit which would sense DC supply variations needs to be designed. At switch ON, if the DC supply is within limits, this circuit would send a signal to the POWER OK interrupt generator.

*Ref Preliminary Data Sheet: Quad Chipped Transversal Fitter, 27 March 1978.

In the event of power failure, this module would signal the POWER FAIL interrupt generator circuit which, in turn, would interrupt the processor (Ref Chap IV).

- An operational time recorder is required to keep track of the time for which the system remains in operation. There should be provisions to move the current value in the buffer register (Figure 56) to memory. This would enable the permanent record of total operational time. There should also be provisions to move the previously stored operational time back to the buffer register and recording the additional operational time. This would enable the system to record the aggregate operational time even if the system is required to monitor the same test piece at different intervals.

- The DC power backup for use in case of a main power failure should also be designed for the prototype. The backup power should maintain the system for the duration it takes to execute the power fail service routine.

Conclusions

The bench model of MIBDAPS as implemented is the first step towards developing a prototype system. The initial design and implementation of different functions of MIBDAPS were successfully carried out. The choice of the LSI-11M microcomputer as the processor for MIBDAPS and the development of software to acquire and process data fulfilled the initial requirements placed on MIBDAPS (Ref Chap II).

The time constraints placed on the processing function of MIBDAPS were exceeded because of the software implementation of the FFT module. The individual software modules have been tested and found accurate

within the bounds of a 16-bit machine. Therefore, the testing of the integrated system would be a fairly easy task. The incorporation of the recommendation discussed earlier in this chapter would result in a prototype model. This prototype system would then accomplish the acquisition and processing of requisite data for damping the layer breakout design as envisioned by AFFDL.

The computation accuracy of the data acquisition and processing functions of the integrated system (MIBDAPS bench model) could not be determined because of the shortage of time. But because the performance of the individual software modules was acceptable, it is a fair assumption that the integrated system would perform accurately, too.

The designed system in its present form shows the practicability of using the state-of-the-art technology to accomplish the task of data acquisition and processing.

The application of such a system is not limited to monitoring aircraft hardware for vibration and temperature data only. It can be configured to act as an environmental recorder where additional analog signals need to be monitored and processed. This could be accomplished by making appropriate changes in the software. In its present form, the system is capable of computing the FFT and power spectrum of a signal; therefore, this feature can be used for spectrum analysis functions. This investigation has successfully demonstrated the feasibility of developing a microcomputer based system from the conceptual phase to a working bench-type model.

Bibliography

1. An Introduction to SADT, Structured Analysis and Design Technique, Softech Inc., November 1976 (9022-78R).

2. Beauchamp, K. B. Signal Processing Using Analog and Digital Techniques. John Wiley and Sons, 1973.

3. Brigham, E. Oran. The Fast Fourier Transform. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1974.

4. Bursky, D. et al. "Microcomputer Selection Guide," Electronic Design, 11: 66-78 (May 24, 1978).

5. Cooley, J. W., P. W. Lewis, and P. D. Welch. "The Fast Fourier Transform Algorithm: Programming Consideration in the Calculation of Sine, Cosine and Laplace Transform," Journal of Sound and Vibration, 12: 315-337 (1970).

6. Cooper, J. W. Minicomputer in the Laboratory. John Wiley and Sons, Inc., 1977.

7. Eckhouse, Jr., R. H. Minicomputer Systems Organization and Programming (PDP-11). Englewood Cliffs, New Jersey: Prentice-Hall, 1975.

8. Hratek, E. R. A User's Handbook of D/A and A/D Convertors. John Wiley & Sons, Inc., 1976.

9. Maneely, J. R. Design of a Laboratory Data Acquisition System. MS thesis presented at the Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio (78-4).

10. M2-8932. Operation and Maintenance Instruction Manual for LSI-11M CPU. Norden Systems, Inc.

11. Microcomputer Handbook (second edition). Digital Equipment Corporation, 1977.

12. Oppenheim, A. V. and R. W. Schafer. Digital Signal Processing. Englewood Cliffs, New Jersey: Prentice-Hall, 1975.

13. Rogers, Lynn., personal interviews, March-November 1978, Aerospace Engineer, Air Force Flight Dynamics Laboratory, Wright-Patterson Air Force Base, Ohio.

14. Structured Analysis: Reader's Guide. Softech Inc, May 1975 (9022-73.2)

15. Subprogram Library Guide. Aeronautical Systems Division, Computer Center, Wright-Patterson Air Force Base, Ohio. Brochure, January 1978.

APPENDIX A

STRUCTURED ANALYSIS AND DESIGN TECHNIQUE

APPENDIX A

Structured Analysis and Design Technique

This appendix (Ref 9) gives a short description of how structured analysis (SA) models are constructed and explains the SA diagram conventions used in this paper. It must be noted that the format used to present the models in this paper is not standard according to the rules developed by SofTech. The changes were made to present the models in a manner which is more familiar to readers who have no experience with SA models. Although the format is not that used by SofTech, the diagrams of the models are organized and related according to SofTech procedures, and the conventions used to construct individual diagrams are standard.

The structured analysis design technique (SADT) is a general purpose top-down, modular technique for modeling functions. The functions may be as varied as farming or manufacturing, but SA was developed primarily as a software requirements definition and design tool. Although a complete SA model actually consists of two models, one for activities and one for data, this paper employs only activity models so the conventions described here are those which apply to activity models.

An SA activity model consists of a series of diagrams which present in progressively more detail the activities necessary to perform some function. Each diagram represents a self-contained activity which is part of the overall function. A diagram shows how its activity is decomposed into subactivities and how the subactivities are related to each other. The subactivities in each diagram may then be decomposed on separate diagrams which leads to a tree structure of several levels. At the top is one diagram which represents the whole function, and at the bottom

Figure A-1 shows how an SA model would appear if all the diagrams were on one page. Of course, in real SA diagrams only one level of decomposition is shown, but the figure demonstrates the top-down nature of SA and the way activities are grouped into modules. In the figure, as in real models, one large box represents the whole function, and that is decomposed into successive levels of related activities. The decomposition process continues until the desired amount of detail has been developed, which may require more levels than shown in Figure A-1. Another thing to note is that while the figure shows only three subactivities in each decomposition, any number from three to six is acceptable.

From Figure A-1, it should be apparent that SA diagrams are constructed with boxes and arrows. In an activity model, each box represents an activity and is called a "node." Arrows represent data where the word "data" is used in a very general sense to include anything that is not an activity. Figure A-2 shows the different meanings given to arrows depending on which side of a box they enter or leave. An input is data that is modified by the activity to produce an output. A control is data which may or may not be converted into output, but which in some way restricts the activity (starts or stops it, for example). Every box must have at least one control arrow. A mechanism is a person or thing which acts as a processor. Mechanism arrows are often omitted when the processor is the same for all nodes. No limit is placed on the number of arrows which may interface with a side of a box, but it is common practice to group related types of data.

Between boxes, arrows may split and join. In general, all branches of an arrow contain the same data unless a branch is given a separate label. This convention is summarized in Figure A-3, which also gives two

forms of OR branches. The OR branches are used to show that data follows one path or the other, but not both.

When two nodes are related so that the output of each is a control for the other, a special two-way arrow may be used. Figure A-4 shows a mutual control situation with a two-way arrow and the equivalent form with normal allows. An arrow showing mutual control has two labels separated by a slash; the first label identifies data going forward, and the second is the feedback data.

A special numbering system is used to distinguish between nodes at different levels and between nodes at the same level. In an activity model, node numbers are prefixed with the letter "A." For preliminary nodes, A is follows by a dash and a number. Node A-1 may be used to show the model in relationship to other functions (Figure 3). Node A-Ø serves as a cover sheet for the model; the node is simply a box showing inputs, outputs, controls, and mechanisms for the function which the model is to describe (Figure 4). Decomposition begins in node AØ. Note in Figure 5 that each box of the decomposition is numbered; the boxes on all decomposition diagrams are numbered, and this number is used to form the node number. For the activities subordinated to node AØ, the node number is simply the box number on AØ, Process Data in Figure 15, for example, becomes node A3. From this level on, the node number is a combination of the node number of the parent diagram and the box number of the subordinate. As an example, the decomposition of Process Data is given in Figure 26. Box 2, Compute Fourier Transform, is assigned the node number A32. Subordinates of A32, if diagrammed, would have the numbers A321, A322, and so on through the last box number.

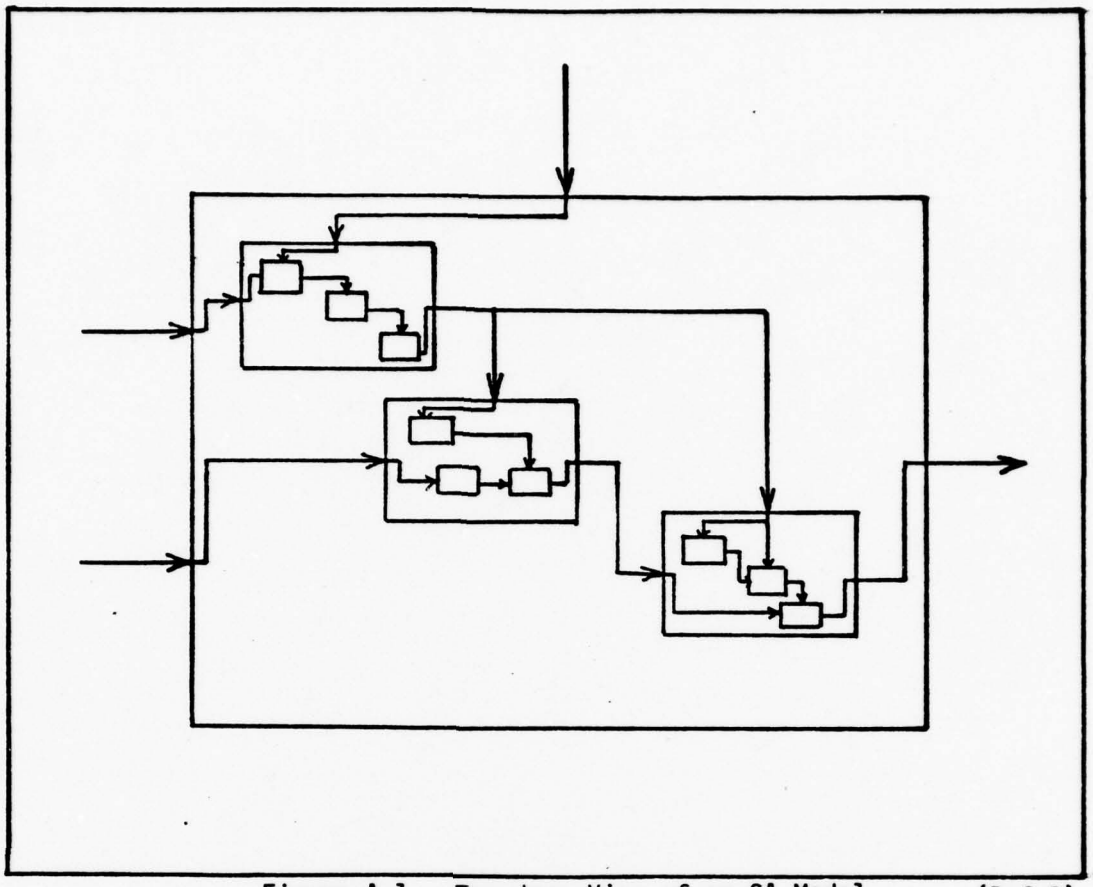


Figure A-1. Top-down View of an SA Model (Ref 9)

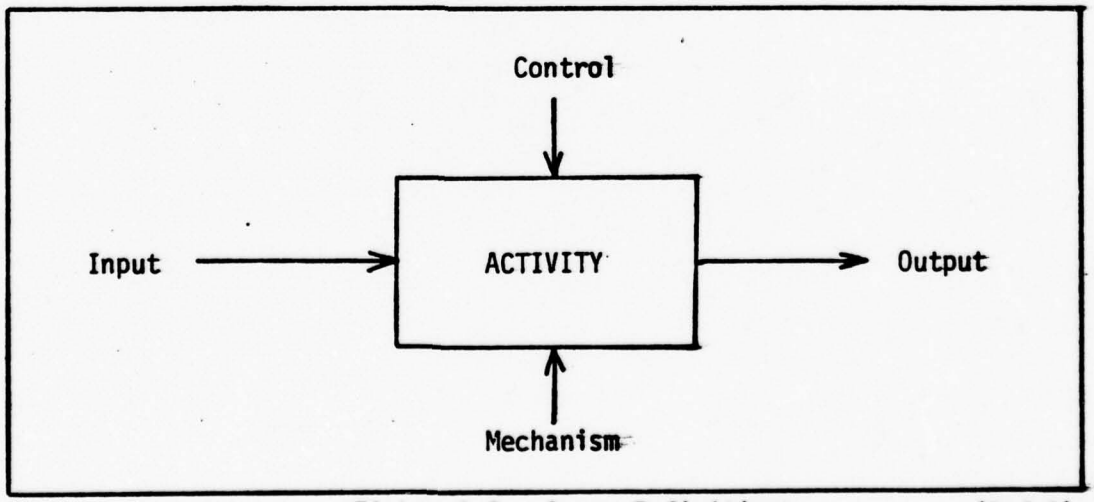


Figure A-2. Arrow Definitions (Ref 9)

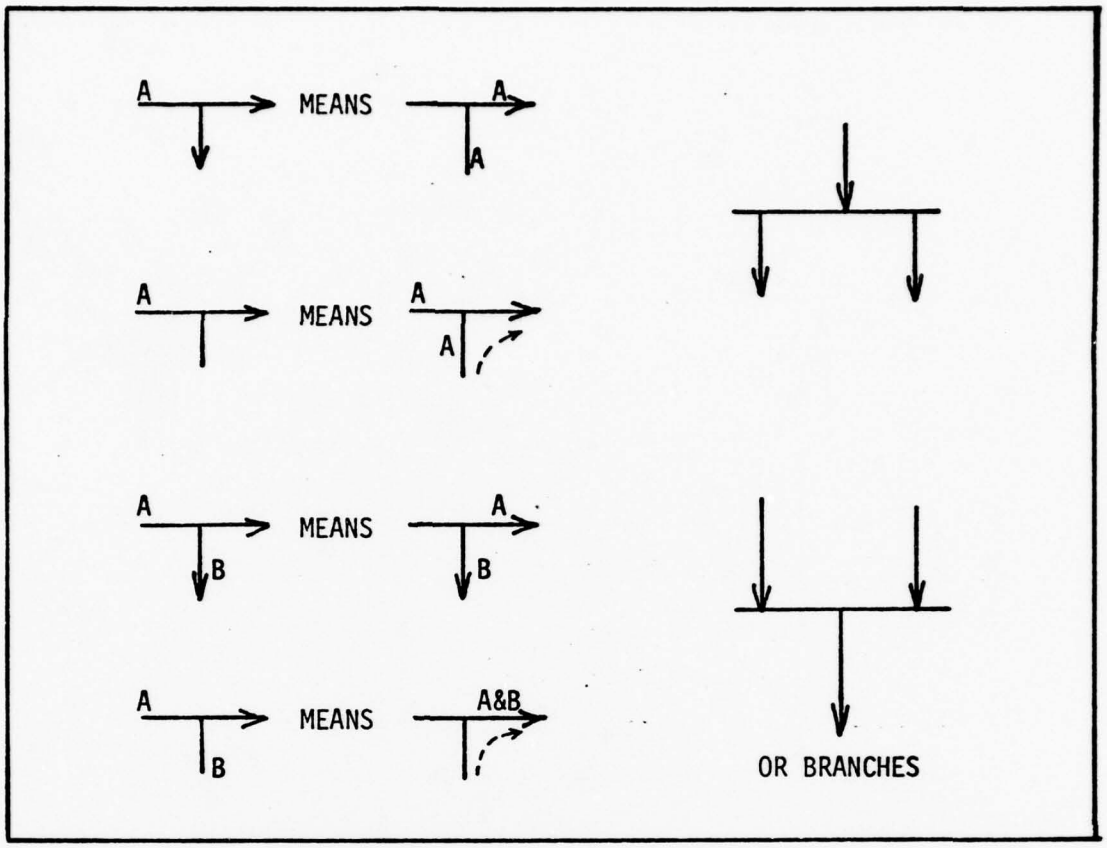


Figure A-3. Arrow Branches (Ref 9)

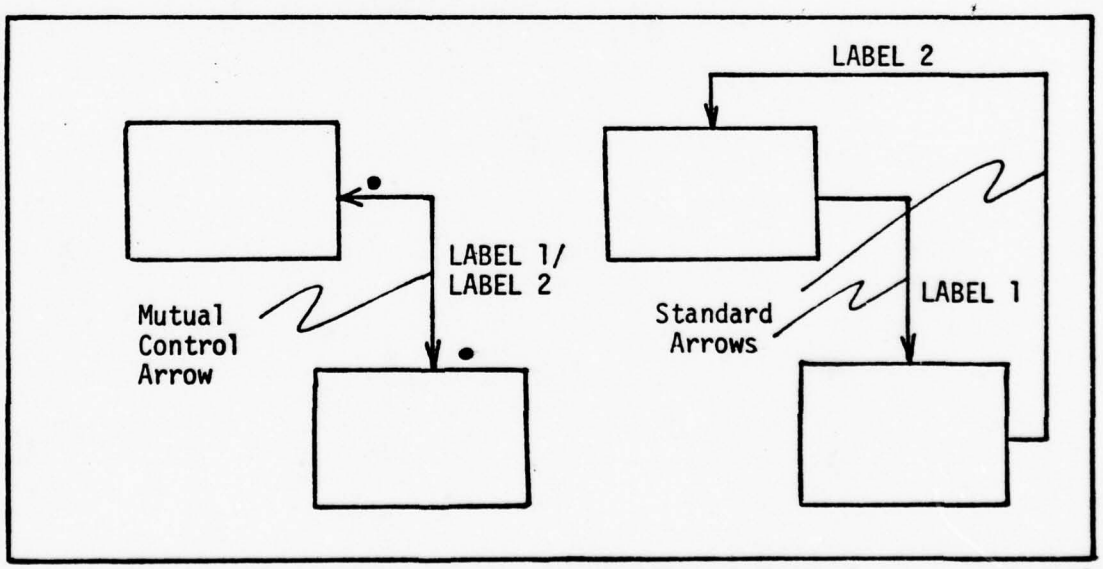


Figure A-4. Arrows Showing Mutual Control (Ref 9)

A special code called an ICOM code (input, control, output, mechanism) is used to identify arrows. The code contains a number, a letter, and another number (Figure A-5).

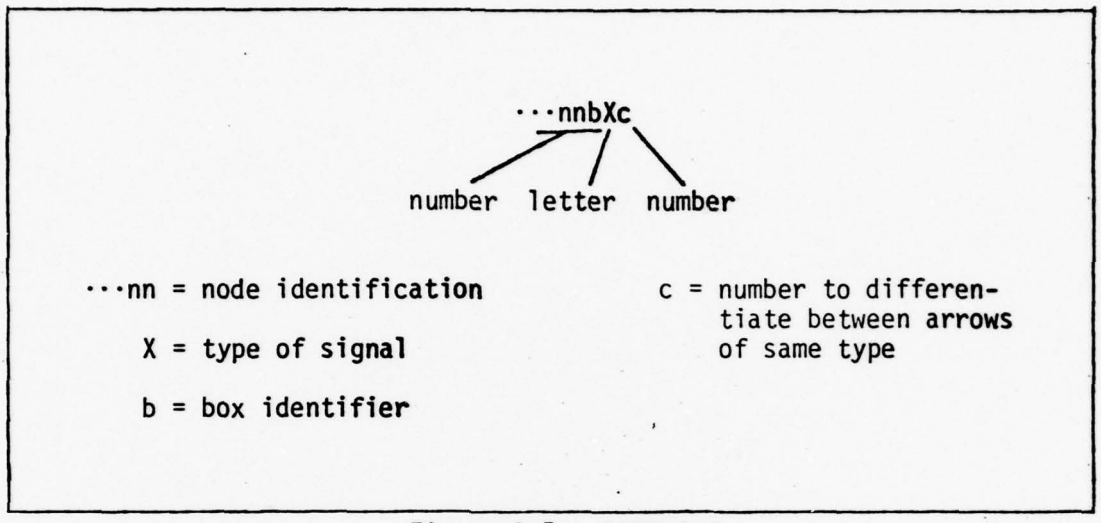


Figure A-5. ICOM Code

The first number has two parts: the digit immediately preceeding the letter is that of source or destination box, and the digits preceeding this number identify the node on which the box is located. The letter refers to the type of arrow: I for input, C for control, O for output, and M for mechanism. The last number distinguishes between arrows of the same type on a box. One exception to the rule is node AØ where the ICOM code has no number preceeding the letter; this has been done to differentiate between signals within the system and signals which are input, output, or control from the outside world. If any signal that appears on node AØ is also present on a subordinate node, the identifying ICOM code remains unchanged.

A few important points about the text describing each SA diagram must be included here. The text is intended to point out the highlights

of a diagram and not repeat all the details. As an aid to following the discussion, the ICOM code is included in parenthesis following any reference to specific diagram features. Finally, the text describing the diagrams in this paper includes I/O specifications, which are not a standard SA practice.

APPENDIX B

INSTRUCTION TIMING

APPENDIX B

Instruction Timing

LSI-11M Instruction Execution Time (Ref 10)

The execution time for an instruction depends on the instruction itself, the modes of addressing used, and the type of memory referenced. In most cases, the instruction execution time is the sum of a basic time, a source address (SRC) time, and a destination address (DST) time.

Instr Time = Basic Time + SRC Time + DST Time

(Basic Time = Fetch Time + Decode Time + Execute Time)

Some of the instructions require only some of these times. All timing information is in microseconds, unless otherwise noted. Times are typical; processing timing can vary ±20%.

The following talbes give the basic time for the LSI-11M instruction set.

TABLE B-I

Source and Destination Times

Mode	SRC Time (Word)	SRC Time (Byte)	DST Time (Word)	DST Time (Byte)
0	Ø	Ø	Ø	Ø
1	1.Ø5µs	1.Ø5µs	1.3Øµs	1.3Øµs
2	1.55	1.8Ø	1.55	1.8Ø
3	2.55	2.55	2.55	2.55
4	1.8Ø	1.8Ø*	1.8Ø	1.8Ø*
5	3.35	3.35	3.35	3.35
6	2.85	2.85	2.85	2.85
7	4.35	4.35	4.35	4.35

*If R6 or R7 is used with Mode 4 and Byte operation, add Ø.25µs to SRC time and Ø.25µs to DST time.

TABLE B-II

Basic Time (DOP)

Double Operand Instruction	SMØ DMO	SM1-7 DMØ	SMØ DM1-7	SM1-7 DM1-7
MOV, ADD	1.75µs	2.55µs	4.55µs	4.55µs
MOVB	2.25(1)	4.Ø5(1)	4.55(2)	5.55(2)
SUB	2.8Ø	2.8Ø	4.8Ø	4.8Ø
CMP, BIT	3.Ø5	3.Ø5	3.Ø5	3.Ø5
CMPB, BITB	2.55	3.55	3.8Ø	2.8Ø
BIC	2.8Ø	2.8Ø	2.8Ø	2.8Ø
BIS	3.Ø5	3.Ø5	4.3Ø	4.3Ø
BICB, BISB	2.8Ø	3.8Ø	5.55(2)	6.55(2)
XOR	2.3Ø	-	4.Ø5	-

(1) Add 0.25µs if effective SRC byte operand is negative.
(2) Add 0.25µs if DST byte address is odd.

TABLE B-III

Basic Time (SOP)

Single Operand Instruction	DMØ	DM1-7
CLR, INC, DEC, SWAB	2.Ø5µs	4.Ø5µs
ADC	2.3Ø	3.Ø5
SBC	2.3Ø(5)	3.Ø5(7)
COM, NEG, ROL, ASL, ROR, ASR, CLRB, SXT	2.3Ø	4.3Ø
TST	2.3Ø	2.3Ø
INCB	2.55(1)	5.Ø5(3)
ADCB	2.3Ø(5)(9)	3.8Ø(7)(10)
DECB	2.55(2)	5.Ø5(3)
SBCB	2.3Ø(5)(11)	3.8Ø(8)(10)
COMB	2.3Ø	5.3Ø(3)
NEGB	2.8Ø(2)	5.3Ø(3)
ROLB, ASLB, RORB, ASRB	3.55	5.55(4)
TSTB	3.Ø5	3.Ø5
MFPS (1Ø67DD)	3.Ø5	4.3Ø(3)
MTPS (1Ø64SS)*	3.55	4.Ø5
JMP	1Ø.44	2.5Ø
JRS	1Ø.54	5.ØØ

*For MTPS, use byte DST times, not SRC times.

(1) Add Ø.25µs if register byte operand = 377 octal.
(2) Add Ø.25µs if register byte operand = Ø.
(3) Add Ø.25µs if DST byte address is odd.
(4) Add Ø.25µs if DST byte address is even.
(5) Add Ø.25µs if C-bit is set.
(6) Add 1.5Øµs if C-bit is set.
(7) Add 1.75µs if C-bit is set.
(8) Add 2.ØØµs if C-bit is set.
(9) Add Ø.25µs if C-bit is set and register byte operand = 377_8.
(10) Add Ø.25µs if C-bit is set and DST byte address is odd.
(11) Add Ø.25µs if C-bit is set and register byte operand = Ø.

TABLE B-IV

Miscellaneous Instruction

Instruction	Basic Time
SOB (BRANCH)	3.Ø5µs
SOB (NO BRANCH)	2.55
SET CC	2.Ø5
CLEAR CC	2.3Ø
NOP	2.3Ø
RTS	3.3Ø
MARK	4.1Ø
RTI	5.6Ø(1)
RTT	5.6Ø(2)
TRAP, EMT, IOT, BPT	1Ø.15
WAIT	2.75(1)
RESET	3.25+1ØµsINIT+9Øµs

(1) Add Ø.5Øµs if NEW PS has bit 4 (T-bit) set.
(2) Add Ø.25µs if NEW PS has bit 4 (T-bit) set.

TABLE B-V

Basic Time (Branch)

Branch Instruction	Branch(1)	No Branch
BR	2.8Ø	-
BNE, BEQ, BPL, BMI, BVC, BVS, BCC/BLO, BCS/BHOS	3.Ø5	2.ØØ
BGE	3.3Ø(2)	2.25(2)
BLT	3.3Ø(3)	2.25(3)
BGT	3.8Ø(2)	2.ØØ(4)
BLE	3.Ø5(5)	2.75(3)
BHI	3.55	2.ØØ(6)
BLOS	3.Ø5(6)	2.5Ø

(1) Add Ø.25µs if offset is negative.
(2) Add Ø.25µs if N-bit is set.
(3) Add Ø.25µs if V-bit is clear.
(4) If Z-bit clear: add Ø.75µs if N-bit is clear; add 1.ØØµs if N-bit is set.
(5) If Z-bit clear: add Ø.75µs if V-bit is set; add 1.ØØµs if V-bit is clear.
(6) Add Ø.Ø5µs if Z-bit is clear.

TABLE B-VI

Basic Time (EIS)

EIS Instruction*	Basic Time
MUL	16.70μs worst case
DIV	23.40μs worst case
ASH (RIGHT)	5.80μs + Ø.25μs per shift
ASH (LEFT)	5.55μs + Ø.75μs per shift
ASHC (RIGHT)	6.60μs + Ø.25μs per shift
ASHC (LEFT)	6.35μs + Ø.75μs per shift

*Use word DST times, not SRC times.

TABLE B-VII

Basic Time (FIS)*

FIS Instruction	Basic Time
FADD	2Ø.15 typical 22.8Ø worst case
FSUB	2Ø.4Ø typical 23.ØØ worst case
FMUL	5Ø.25 typical 69.4Ø worst case
FDIV	69.4Ø typical 7Ø.1Ø worst case

*Inst Time (FIS) = Basic Time + Shift Time for Binary Points + Shift Time for normalization.
Binary Point Alignment: Ø.25μs per shift.
Normalization: Ø.75μs per shift.

APPENDIX C

THEORETICAL DEVELOPMENT OF A BASE-2 FFT ALGORITHM

APPENDIX C

Theoretical Development of a Base-2 FFT Algorithm

Development of the Cooley-Tukey FFT Algorithm (Ref 2:257-265)

The DFT of N-data can be written as:

$$X(n) = \sum_{k=0}^{N-1} x(k)W^{nk}, \quad n = 0, 1, \ldots, N-1 \qquad \text{(C-1)}$$

where

$$W^{nk} = \exp \frac{-2\pi ink}{N}, \quad n = 0, 1, \ldots N-1, \; k = 0, 1, \ldots N-1$$

The direct evaluation of Eq (C-1) requires the solution of the following set of equations.

$$\begin{aligned}
X(0)^1 &= W^0x(0) + W^0x(1) + W^0x(2) \ldots + W^0x(N-1) \\
X(1) &= W^0x(0) + W^1x(1) + W^2x(w) \ldots + W^{N-1}x(N-1) \\
X(2) &= W^0x(0) + W^2x(1) + W^4x(2) \ldots + W^{2N-2}x(N-1) \\
X(3) &= W^0x(0) + W^3x(1) + W^6x(2) \ldots + W^{3N-3}x(N-1) \\
&\vdots \\
X(N-1) &= W^0x(0) + W^{N-1}x(1) + W^{2N-2}x(2) \ldots W^{(N-1)N-(N-1)}x(N-1)
\end{aligned}$$

The above set of equations can also be written in the matrix form given below:

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ \vdots \\ X(N-1) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & \ldots & W^0 \\ W^0 & W^1 & W^2 & \ldots & W^{N-1} \\ W^0 & W^2 & W^4 & \ldots & W^{2N-2} \\ W^0 & W^3 & W^6 & \ldots & W^{3N-3} \\ & & & & \\ W & W^{N-1} & W^{2N-2} & \ldots & W^{(N-1)N-(N-1)} \end{bmatrix} \begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ \\ X(N-1) \end{bmatrix}$$

Therefore, in order to compute a DFT of N-data points, we have to solve the matrix given by Eq (C-2), all terms of this matrix being complex:

$$[X(n)] = [W^{nk}][x(k)] \qquad \text{(C-2)}$$

This direct evaluation requires N^2 complex multiplications and N(N-1) complex additions (Ref 3:151). The FFT algorithm developed by Cooley and Tukey (Ref 3:151) reduces the number of complex multiplications and additions, and computer execution of this algorithm results in faster computation time as compared with the evaluation of Eq (C-1) (Ref 3:151).

In case N is limited to a power of 2 (i.e., $N = 2^p$), it is possible to express k and n in terms of the index p, as a binary weighted series:

$$k = k_{p-1}2^{p-1} + k_{p-2}2^{p-2} + \ldots k_1 2^1 + k_0 \qquad \text{(C-3)}$$

$$n = n_{p-1}2^{p-1} + n_{p-2}2^{p-2} + \ldots n_1 2^1 + n_0 \qquad \text{(C-4)}$$

Here k_{p-1}, k_{p-2}, ... k_0 and n_{p-1}, n_{p-2}, ... n_0 can take a value of either Ø or 1. In this way, we can express all the N possible values of the indices in terms of a binary number and, hence, facilitate the consideration of storage in the digital computer. Using this convention, Eq (C-1) can be rewritten in the following form:

$$X(n_{p-1}, n_{p-2}, \ldots n_0) = \sum_{k_0=0,} \sum_{k_1=0,} \ldots \sum_{k_{p-1}=0} x(k_{p-1}, k_{p-2}, \ldots k_0)\, x\, W^{n(k_{p-1}2^{p-1} + k_{p-2}2^{p-2} + \ldots k_0)} \qquad \text{(C-5)}$$

If the first term of the exponential for Eq (C-3) is expanded, Eq (C-6) is the resulting equation:

$$W^{nk_{p-1}2^{p-1}} = W^{n_{p-1}2^{p-1}k_{p-1}2^{p-1}} + W^{n_{p-2}2^{p-2}k_{p-1}2^{p-1}} + \ldots W^{n_1 2^1 k_{p-1}2^{p-1}} + W^{n_o k_{p-1}2^{p-1}} \qquad \text{(C-6)}$$

This can be simplified to give Eq (C-7) because all other terms go to unity. This enables

$$W^{nk_{p-1}2^{p-1}} = W^{n_0 k_{p-1} 2^{p-1}} \tag{C-7}$$

the innermost sume of Eq (C-5) over k_{p-1} to be written as a shorter Fourier transform:

$$X_1(n_0, k_{p-2} \cdots k_0) = \sum_{k_{p-1}=0} x_1(k_{p-1}, k_{p-2}, \cdots k_0) W^{n_0 k_{p-1} 2^{p-1}} \tag{C-8}$$

Unlike the complete Fourier transform, this sum consists of a set of N numbers only, each calculated from two of the original data points. Subsequent sums proceeding outwards in Eq (C-5) can be calculated using a generalized recursive expression for the exponential term:

$$W^{nk_{p-q}2^{p-q}} = W^{(n_{q-1}2^{q-1} + n_{q-2}2^{q-2} + \cdots n_0) k_{p-q} 2^{p-q}} \tag{C-9}$$

here

$$q = 1, 2, 3 \cdots p$$

The successive sums are evaluated according to the equation

$$X_q(n_0, n_1 \cdots n_{q-1}, k_{p-q-1}, k_{p-q-2} \cdots k_0) =$$

$$\sum_{k_{p-q}=0} x_{q-1}(n_0, n_1 \cdots n_{q-2}, k_{p-q} \cdots k_0) \; x$$

$$W^{(n_{q-1}2^{q-1} + n_{q-2}2^{q-2} + \cdots n_0) k_{p-q} 2^{p-q}} \tag{C-10}$$

here

$$q = 1, 2, 3 \cdots p$$

To apply this recursive formulae, the initial set of data, x(k), is first made equal to $x_0(k)$ so that $q = 1$, thus

$$x(k) = x(k_{p-1}, k_{p-2} \cdots k_0) = x_0(k_{p-1}, k_{p-2} \cdots k_0) \qquad \text{(C-11)}$$

This leads to the derivation of succeeding arrays in x_q so that the final array will be in X(n). Equation (C-12) given below shows that the elements of X(n) will be calculated in incorrect order, so some sort of shuffling is required to get the result in correct order. (This point is clarified when the algorithm for a 16-point DFT using this FFT method is worked.)

$$X(n_{p-1}, n_{p-2} \cdots n_0) = x_p(n_0, \cdots n_{p-2}, n_{p-1}) \qquad \text{(C-12)}$$

A 16-Point FFT Algorithm

In case the number of data points to be transferred is 16, the value of P = 4. Therefore, k,n can be expressed as 4-bit binary numbers as given in the following equations:

$$k = 8k_3 + 4k_2 + 2k_1 + k_0 \qquad \text{(C-13)}$$

$$n = 8n_3 + 4n_2 + 2n_1 + n_0 \qquad \text{(C-14)}$$

Using Eqs (C-13) and (C-14), Eq (C-5) can be written as

$$X(n_3, n_2, n_1, n_0) = \sum_{k_0=0}^{1} \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} \sum_{k_3=0}^{1} x(k_3, k_2, k_1, k_0) \times W^{n(8k_3 + 4k_2 + 2k_1 + k_0)} \qquad \text{(C-15)}$$

Using Eqs (C-7) and (C-8), the innermost summation can be written as

$$x_1(n_0, k_2, k_1, k_0) = \sum_{k_3=0} x_0(k_3, k_2, k_1, k_0) W^{8k_3 n_0} \qquad \text{(C-16)}$$

Subsequent summations can be written using Eqs (C-9) and (C-10):

$$x_2(n_0, n_1, k_1, k_0) = \sum_{k_2=0} x_1(n_0, k_2, k_1, k_0) W^{4k_2(2n_1 + n_0)} \quad \text{(C-17)}$$

$$x_3(n_0, n_1, n_2, k_0) = \sum_{k_1=0} x_2(n_0, n_1, k_1, k_0) W^{2k_1(4n_2 + 2n_1 + n_0)} \quad \text{(C-18)}$$

$$x_4(n_0, n_1, n_2, n_3) = \sum_{k_0=0} x_3(n_0, n_1, n_2, k_0) W^{k_0(8n_3 + 4n_2 + 2n_1 + n_0)} \quad \text{(C-19)}$$

$$X(n_3, n_2, n_1, n_0) = x_4(n_0, n_1, n_2, n_3) \quad \text{(C-20)}$$

The set of equations that result when summations of Eqs (C-16), (C-17), (C-18), (C-19), and (C-20) are expanded is given in tables C-I, C-II, C-III, C-IV, and C-V, respectively.

Table C-V shows that the final array for X(n) is in bit-reversed order; therefore, to get the result in order, a post-shuffling needs to be done. Another way to overcome this problem is to bit invert the input array $[x_0(k)]$, and the output will automatically be in order (Ref 3:178). The signal flow graph for the 16-point FFT without bit inverting the data is shown in Figure C-1, and Figure C-2 shows the signal flow graph for data which has been bit inverted before being transformed.

TABLE C-I

Expansion Results of Eq (C-16)

$$x_1(0, 0, 0, 0) = x_0(0, 0, 0, 0)W^0 + x_0(1, 0, 0, 0)W^0$$
$$x_1(0, 0, 0, 1) = x_0(0, 0, 0, 1)W^0 + x_0(1, 0, 0, 1)W^0$$
$$x_1(0, 0, 1, 0) = x_0(0, 0, 1, 0)W^0 + x_0(1, 0, 1, 0)W^0$$
$$x_1(0, 0, 1, 1) = x_0(0, 0, 1, 1)W^0 + x_0(1, 0, 1, 1)W^0$$
$$x_1(0, 1, 0, 0) = x_0(0, 1, 0, 0)W^0 + x_0(1, 1, 0, 0)W^0$$
$$x_1(0, 1, 0, 1) = x_0(0, 1, 0, 1)W^0 + x_0(1, 1, 0, 1)W^0$$
$$x_1(0, 1, 1, 0) = x_0(0, 1, 1, 0)W^0 + x_0(1, 1, 1, 0)W^0$$
$$x_1(0, 1, 1, 1) = x_0(0, 1, 1, 1)W^0 + x_0(1, 1, 1, 1)W^0$$
$$x_1(1, 0, 0, 0) = x_0(0, 0, 0, 0)W^0 + x_0(1, 0, 0, 0)W^8$$
$$x_1(1, 0, 0, 1) = x_0(0, 0, 0, 1)W^0 + x_0(1, 0, 0, 1)W^8$$
$$x_1(1, 0, 1, 0) = x_0(0, 0, 1, 0)W^0 + x_0(1, 0, 1, 0)W^8$$
$$x_1(1, 0, 1, 1) = x_0(0, 0, 1, 1)W^0 + x_0(1, 0, 1, 1)W^8$$
$$x_1(1, 1, 0, 0) = x_0(0, 1, 0, 0)W^0 + x_0(1, 1, 0, 0)W^8$$
$$x_1(1, 1, 0, 1) = x_0(0, 1, 0, 1)W^0 + x_0(1, 1, 0, 1)W^8$$
$$x_1(1, 1, 1, 0) = x_0(0, 1, 1, 0)W^0 + x_0(1, 1, 1, 0)W^8$$
$$x_1(1, 1, 1, 1) = x_0(0, 1, 1, 1)W^0 + x_0(1, 1, 1, 1)W^8$$

TABLE C-II

Expansion Results of Eq (C-17)

$$x_2(0,\ 0,\ 0,\ 0) = x_1(0,\ 0,\ 0,\ 0)W^0 + x_1(0,\ 1,\ 0,\ 0)W^0$$
$$x_2(0,\ 0,\ 0,\ 1) = x_1(0,\ 0,\ 0,\ 1)W^0 + x_1(0,\ 1,\ 0,\ 1)W^0$$
$$x_2(0,\ 0,\ 1,\ 0) = x_1(0,\ 0,\ 1,\ 0)W^0 + x_1(0,\ 1,\ 1,\ 0)W^0$$
$$x_2(0,\ 0,\ 1,\ 1) = x_1(0,\ 0,\ 1,\ 1)W^0 + x_1(0,\ 1,\ 1,\ 1)W^0$$
$$x_2(0,\ 1,\ 0,\ 0) = x_1(0,\ 0,\ 0,\ 0)W^0 + x_1(0,\ 1,\ 0,\ 0)W^8$$
$$x_2(0,\ 1,\ 0,\ 1) = x_1(0,\ 0,\ 0,\ 1)W^0 + x_1(0,\ 1,\ 0,\ 1)W^8$$
$$x_2(0,\ 1,\ 1,\ 0) = x_1(0,\ 0,\ 1,\ 0)W^0 + x_1(0,\ 1,\ 1,\ 0)W^8$$
$$x_2(0,\ 1,\ 1,\ 1) = x_1(0,\ 0,\ 1,\ 1)W^0 + x_1(0,\ 1,\ 1,\ 1)W^8$$
$$x_2(1,\ 0,\ 0,\ 0) = x_1(1,\ 0,\ 0,\ 0)W^0 + x_1(1,\ 1,\ 0,\ 0)W^4$$
$$x_2(1,\ 0,\ 0,\ 1) = x_1(1,\ 0,\ 0,\ 1)W^0 + x_1(1,\ 1,\ 0,\ 1)W^4$$
$$x_2(1,\ 0,\ 1,\ 0) = x_1(1,\ 0,\ 1,\ 0)W^0 + x_1(1,\ 1,\ 1,\ 0)W^4$$
$$x_2W1,\ 0,\ 1,\ 1) = x_1(1,\ 0,\ 1,\ 1)W^0 + x_1(1,\ 1,\ 1,\ 1)W^4$$
$$x_2(1,\ 1,\ 0,\ 0) = x_1(1,\ 0,\ 0,\ 0)W^0 + x_1(1,\ 1,\ 0,\ 0)W^{12}$$
$$x_2(1,\ 1,\ 0,\ 1) = x_1(1,\ 0,\ 0,\ 1)W^0 + x_1(1,\ 1,\ 0,\ 1)W^{12}$$
$$x_2(1,\ 1,\ 1,\ 0) = x_1(1,\ 0,\ 1,\ 0)W^0 + x_1(1,\ 1,\ 1,\ 0)W^{12}$$
$$x_2(1,\ 1,\ 1,\ 1) = x_1(1,\ 0,\ 1,\ 1)W^0 + x_1(1,\ 1,\ 1,\ 1)W^{12}$$

TABLE C-III

Expansion Result of Eq (C-18)

$$x_3(0,0,0,0) = x_2(0,0,0,0)W^0 + x_2(0,0,1,0)W^0$$
$$x_3(0,0,0,1) = x_2(0,0,0,1)W^0 + x_2(0,0,1,1)W^0$$
$$x_3(0,0,1,0) = x_2(0,0,0,0)W^0 + x_2(0,0,1,0)W^8$$
$$x_3(0,0,1,1) = x_2(0,0,0,1)W^0 + x_2(0,0,1,1)W^8$$
$$x_3(0,1,0,0) = x_2(0,1,0,0)W^0 + x_2(0,1,1,0)W^4$$
$$x_3(0,1,0,1) = x_2(0,1,0,1)W^0 + x_2(0,1,1,1)W^4$$
$$x_3(0,1,1,0) = x_2(0,1,0,0)W^0 + x_2(0,1,1,0)W^{12}$$
$$x_3(0,1,1,1) = x_2(0,1,0,1)W^0 + x_2(0,1,1,1)W^{12}$$
$$x_3(1,0,0,0) = x_2(1,0,0,0)W^0 + x_2(1,0,1,0)W^2$$
$$x_3(1,0,0,1) = x_2(1,0,0,1)W^0 + x_2(1,0,1,1)W^2$$
$$x_3(1,0,1,0) = x_2(1,0,0,0)W^0 + x_2(1,0,1,0)W^{10}$$
$$x_3(1,0,1,1) = x_2(1,0,0,1)W^0 + x_2(1,0,1,1)W^{10}$$
$$x_3(1,1,0,0) = x_2(1,1,0,0)W^0 + x_2(1,1,1,0)W^6$$
$$x_3(1,1,0,1) = x_2(1,1,0,1)W^0 + x_2(1,1,1,1)W^6$$
$$x_3(1,1,1,0) = x_2(1,1,0,0)W^0 + x_2(1,1,1,0)W^{14}$$
$$x_3(1,1,1,1) = x_2(1,1,0,1)W^0 + x_2(1,1,1,1)W^{14}$$

TABLE C-IV

Expansion Result of Eq (C-19)

$$x_4(0, 0, 0, 0) = x_3(0, 0, 0, 0)W^0 + x_3(0, 0, 0, 1)W^0$$
$$x_4(0, 0, 0, 1) = x_3(0, 0, 0, 0)W^0 + x_3(0, 0, 0, 1)W^8$$
$$x_4(0, 0, 1, 0) = x_3(0, 0, 1, 0)W^0 + x_3(0, 0, 1, 1)W^4$$
$$x_4(0, 0, 1, 1) = x_3(0, 0, 1, 0)W^0 + x_3(0, 0, 1, 1)W^{12}$$
$$x_4(0, 1, 0, 0) = x_3(0, 1, 0, 0)W^0 + x_3(0, 1, 0, 1)W^2$$
$$x_4(0, 1, 0, 1) = x_3(0, 1, 0, 0)W^0 + x_3(0, 1, 0, 1)W^{10}$$
$$x_4(0, 1, 1, 0) = x_3(0, 1, 1, 0)W^0 + x_3(0, 1, 1, 1)W^6$$
$$x_4(0, 1, 1, 1) = x_3(0, 1, 1, 0)W^0 + x_3(0, 1, 1, 1)W^{14}$$
$$x_4(1, 0, 0, 0) = x_3(1, 0, 0, 0)W^0 + x_3(1, 0, 0, 1)W^1$$
$$x_4(1, 0, 0, 1) = x_3(1, 0, 0, 0)W^0 + x_3(1, 0, 0, 1)W^9$$
$$x_4(1, 0, 1, 0) = x_3(1, 0, 1, 0)W^0 + x_3(1, 0, 1, 1)W^5$$
$$x_4(1, 0, 1, 1) = x_3(1, 0, 1, 0)W^0 + x_3(1, 0, 1, 1)W^{13}$$
$$x_4(1, 1, 0, 0) = x_3(1, 1, 0, 0)W^0 + x_3(1, 1, 0, 1)W^3$$
$$x_4(1, 1, 0, 1) = x_3(1, 1, 0, 0)W^0 + x_3(1, 1, 1, 1)W^{11}$$
$$x_4(1, 1, 1, 0) = x_3(1, 1, 1, 0)W^0 + x_3(1, 1, 1, 1)W^7$$
$$x_4(1, 1, 1, 1) = x_3(1, 1, 1, 0)W^0 + x_3(1, 1, 1, 1)W^{15}$$

TABLE C-V

Expansion Result of Eq (C-20)

$$x(0, 0, 0, 0) = x_4(0, 0, 0, 0)$$
$$x(0, 0, 0, 1) = x_4(1, 0, 0, 0)$$
$$x(0, 0, 1, 0) = x_4(0, 1, 0, 0)$$
$$x(0, 0, 1, 1) = x_4(1, 1, 0, 0)$$
$$x(0, 1, 0, 0) = x_4(0, 0, 1, 0)$$
$$x(0, 1, 0, 1) = x_4(1, 0, 1, 0)$$
$$x(0, 1, 1, 0) = x_4(0, 1, 1, 0)$$
$$x(0, 1, 1, 1) = x_4(1, 1, 1, 0)$$
$$x(1, 0, 0, 0) = x_4(0, 0, 0, 1)$$
$$x(1, 0, 0, 1) = x_4(1, 0, 0, 1)$$
$$x(1, 0, 1, 0) = x_4(0, 1, 0, 1)$$
$$x(1, 0, 1, 1) = x_4(1, 1, 0, 1)$$
$$x(1, 1, 0, 0) = x_4(0, 0, 1, 1)$$
$$x(1, 1, 0, 1) = x_4(1, 0, 1, 1)$$
$$x(1, 1, 1, 0) = x_4(0, 1, 1, 1)$$
$$x(1, 1, 1, 1) = x_4(1, 1, 1, 1)$$

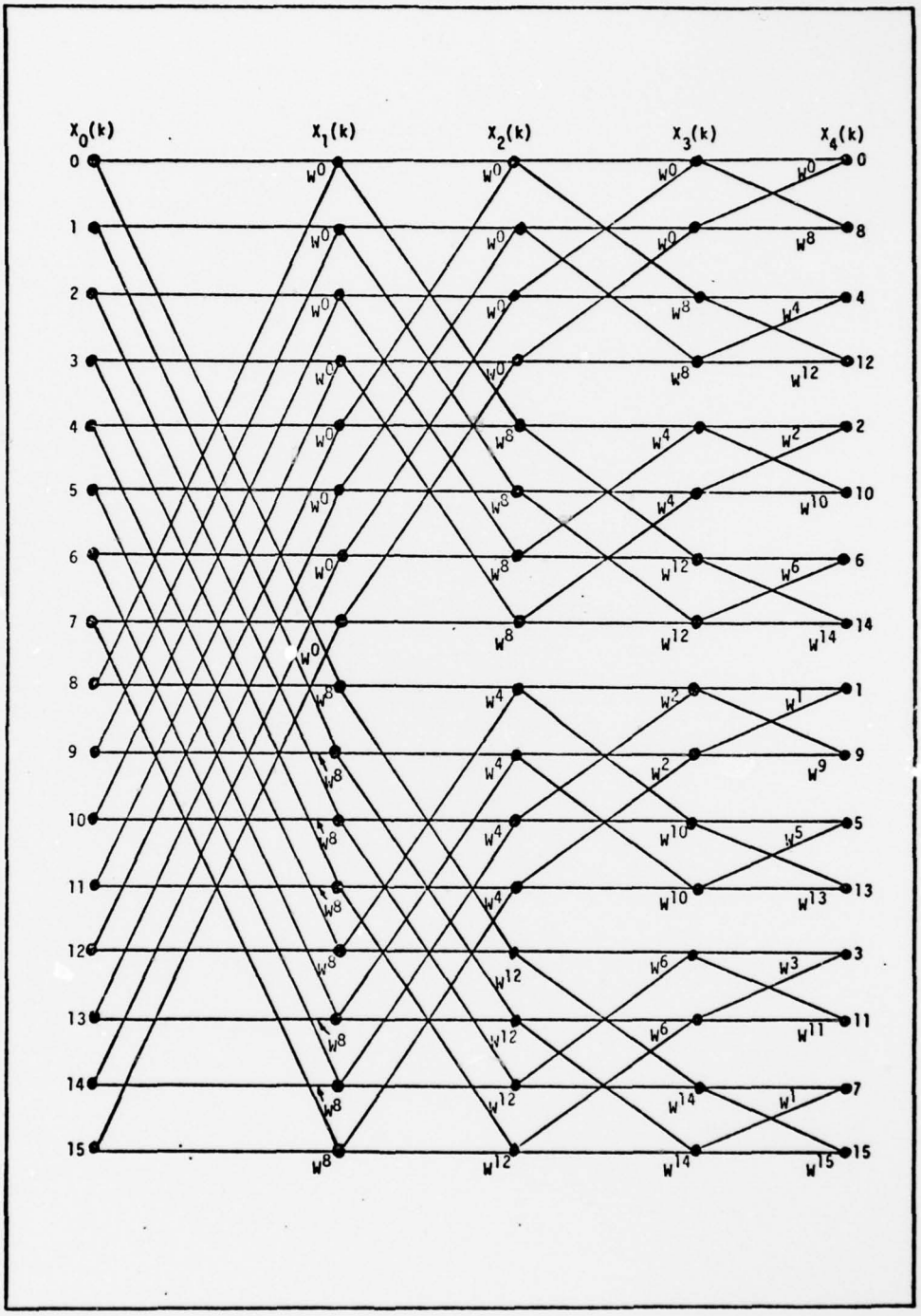


Figure C-1. Signal Flow Graph Without Bit-Inversion of Data

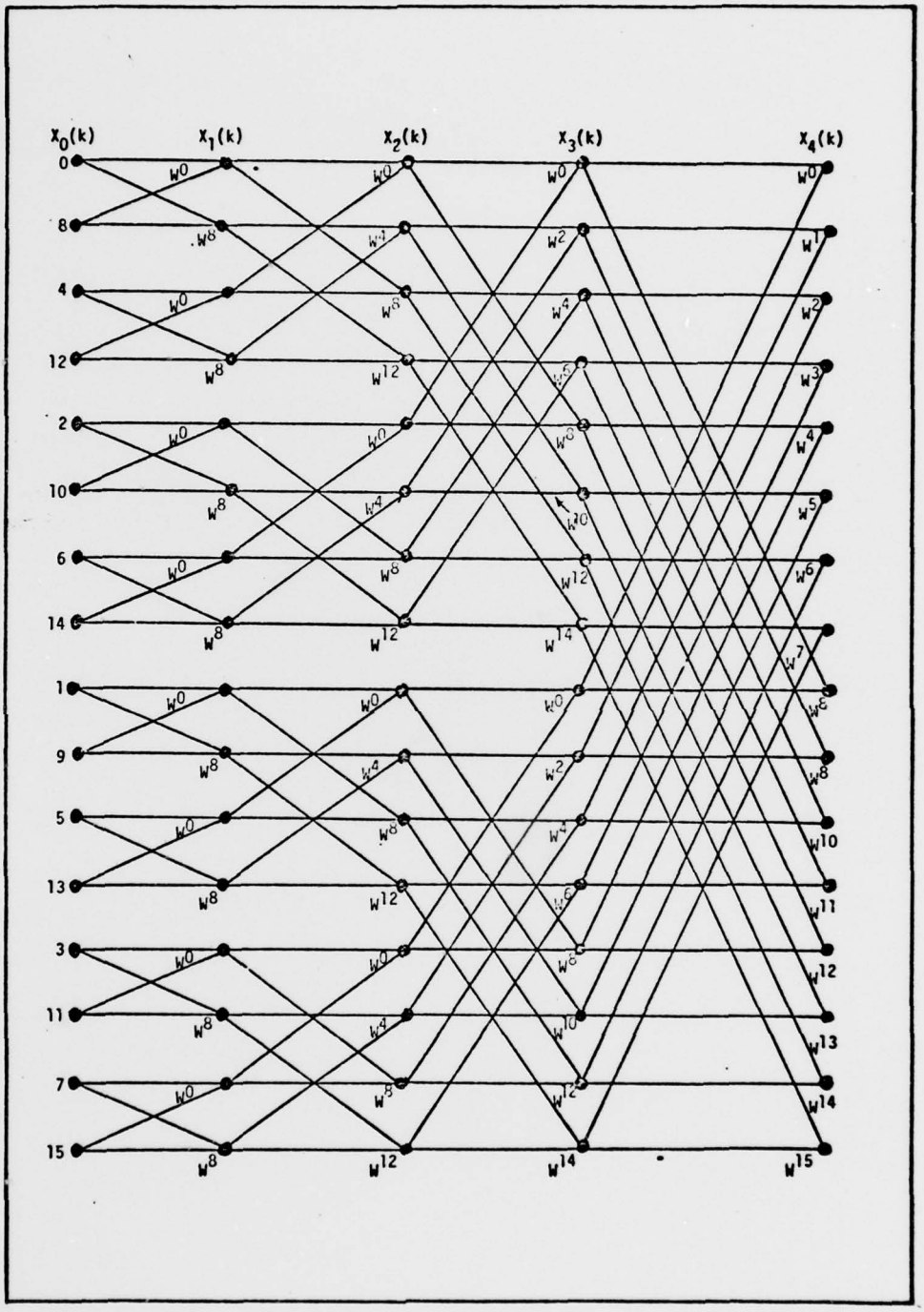


Figure C-2. Signal Flow Graph With Bit-Inversion of Data

APPENDIX D

CALCULATION OF DFT OF REAL DATA

APPENDIX D

Calculation of DFT of Real Data

Consider a real 2N point sequence x(k); the DFT of this sequence can be carried out using Eq (D-1):

$$X(n) = \sum_{k=0}^{2N-1} x(k) W_{2N}^{nk} \quad n = 0, 1, \ldots 2N-1 \tag{D-1}$$

$$W_{2N}^{nk} = \exp \frac{-2\pi nik}{2N}, \ldots n = 0, 1 \ldots 2N-1 \tag{D-2}$$

Equation (D-1) can be rewritten in the following form:

$$X(n) = \sum_{r=0}^{N-1} x(2r) W_{2N}^{2rn} + \sum_{r=0}^{N-1} x(2r+1) W_{2N}^{(2r+1)n} \tag{D-3}$$

$$X(n) = \sum_{r=0}^{N-1} x(2r)\, W_N^{Rn} + W_N^n \left[\sum_{r-0}^{N-1} x(2r+1)\, W_N^{rn} \right] \tag{D-4}$$

Equation (D-4) has resulted by splitting x(k) in (D-1) into even and odd sequences as shown in Equations (D-5) and D-6) respectively.

$$y(r) = x(2r) \qquad \text{even sequence} \tag{D-5}$$

$$z(r) = x(2r+1) \qquad \text{odd sequence} \tag{D-6}$$

A complex sequence v(r) is now defined such that the even points of x(k) [Equation (D-5)] becomes the real part of v(r) and the odd point of x(k) [Equation (D-6)] becomes the imaginary part of v(r).

$$v(r) \equiv y(r) + jz(r) \tag{D-7}$$

An N-point DFT of V(r) yields V(n), which is given in Eq (D-8):

$$V(n) = \sum_{r=0}^{N-1} v(r)W_N^{rn} \tag{D-8}$$

or

$$V(n) = \sum_{r=0}^{N-1} y(r)W_N^{rn} + j\sum_{r=0}^{N-1} z(r)W_N^{rn} \tag{D-9}$$

or

$$V(n) = Y(n) + jZ(n) \tag{D-10}$$

The DFT of real and imaginary parts can be written in terms of V(n) as (Ref 5:319).

$$Y(n) = 1/2[V(n) + V^*(N-n)] \tag{D-11}$$

and

$$Z(n) = 1/2j[V(n) - V^*(N-n)] \tag{D-12}$$

here

$$V^*(N-n) = Y(N-n) - jZ(N-n) \tag{D-13}$$

The result obtained in Eqs (D-11) and (D-12) can be used to rewrite Eq (D-4) as:

$$X(n) = Y(n) + W_{2N}^n Z(n) \tag{D-14}$$

or

$$X(n) = 1/2[V(n) + V^*(N-n)] + 1/2j[V(n) - V^*(N-n)] \tag{D-15}$$

This equation is true for $0 \leqslant n \leqslant N$. [Note that $V(N) = V(0)$].

(Ref 5:320-321)

APPENDIX E

LSI-11 ASSEMBLY LANGUAGE CODE FOR FFT SUBROUTINE (FOURIE) AND EXECUTION TIME COMPUTATION

```
FOURIE --FFT SUBROUTINE RT-11 MACRO VM02-12   30-OCT-78 23:42:04 PAGE 1

                                        .TITLE FOURIE --FFT SUBROUTINE
                                        .SBTTL INTRODUCTION --INTRO: TO FOURIE AND ITS USE.
000000'                                 .CSECT


                                              1. DESCRIPTION:
                                              ---------------
                                                 THIS SUBROUTINE CAN BE USED TO TRANSFORM 32 REAL
                                              DATA POINTS. IT ASSUMES THE DATA EQUIVALENT TO 16
                                              COMPLEX POINTS (THE ODD REAL POINTS ARE CONSIDERED
                                              REAL PARTS AND THE EVEN PARTS ARE CONSIDERED COMPLEX)
                                              AFTER THE TRANSFORM A POST PROCESSING OF DATA GIVES
                                              16 FOURIER TRANSFORMED FREQUENCY COMPONENTS (THE OTH
                                              -ER 16 COMPONENTS ARE THE COMPLEX CONJUGATES OF THESE
                                              VALUES).

                                              2. DEFINITION OF PARAMETERS USED
                                              --------------------------------
                                                 THE FOLLOWING ARE THE DIFFERENT PARAMETERS USED
                                              IN THE FFT SUBROUTINE "FOURIE":

000000  000040                                        N:      40      ;N IS THE # OF REAL DATA
                                                                      POINTS IN OCTAL FORMAT

000002  000020                                        N2:     20      ;N2 IS THE ASSUMED # OF
                                                                      COMPLEX POINTS IN OCTAL.

000004  000004                                        B:      4       ;B IS THE EXPONENT OF 2
                                                                      WHICH SATISFIES THE RELA
                                                                      -TIONSHIP  N2=2**B.

000006  000000                                        DATA:   0       ;DATA IS POINTER THAT WILL
                                                                      CONTAIN THE STARTING ADDRESS
                                                                      OF INPUT REAL POINTS THAT
                                                                      ARE TO BE TRANSFORMED.

                                              3. DATA STORAGE AREA ALLOCATION:
                                              --------------------------------
                                                 TWO DATA BUFFER AREAS ARE USED SO THAT DATA CAN
                                              BE ACQUIRED AND STORED IN THE 2ND BUFFER WHILE PROC
                                              -ESSING IS BEING DONE ON THE 1ST BUFFER.

000010  000000                                        ISTR1:  0
        000112'                                                 .=.+100
000112  000000                                        ISTR2:  0
        000214'                                                 .=.+100

                                              NOTE: EACH BUFFER IS 32 WORDS LONG.
```

```
                                        
                                        
                                        
                                        
                                        
                                        
                                        
                                        
                                        
                                        
                                        
                                        
                                        
                                        
                                        
                                        
                                        
                                        
                                        
                                        
                                        
                                        
                                        
```

4. GLOBAL SYMBOLS:

THE FOLLOWING SYMBOLS OF "FOURIE" ARE USED BY OTHER SUBROUTINES THAT ARE CALLED BY THE EXECUTIVE.

```
.GLOBL FOURIE,DATA,ISTR1,ISTR2,N,N2
```

5. REGISTER DEFINITION:

```
.MCALL .REGDEF
.REGDEF
```

CONTINUED ON NEXT PAGE....

```
                                        .SBTTL FOURIE --DIFFERENT PARTS AND THEIR CODE.
                                ;
                                ;
                                ;
                                ;       1. THE BIT INVERSION ALGORITHM:
                                ;       -------------------------------
                                ;         THE INPUT DATA POINTS ARE BIT INVERTED (OR SHUFF
                                ;       -LED) SO THAT THE FINAL TRANSFORMED ARRAY IS IN CO
                                ;       -RRECT ORDER.
                                ;
                                ;
000214  005067  001234          FOURIE: CLR     SFACTR          ;SET SACLE FACTOR TO 0
000220  005000                          CLR     R0              ;R0 WILL CONTAIN THE
                                ;                               STARTING ADDRESS OF DA
                                ;                               -TA TO BE TRANSFORMED.
000222  005001                          CLR     R1              ;R1 WILL CONTAIN BIT
                                ;                               INVERTED ADDRESS.
000224  016705  177552                  MOV     N2,R5           ;COUNTER TO KEEP TRACK
                                ;                               OF # OF DATA POINTS.
000230  016704  177550          BIV10:  MOV     B,R4            ;BIT SHIFTING COUNTER
000234  005002                          CLR     R2
000236  000241                  BIV20:  CLC                     ;CARRY FLAG IS CLEARED
                                ;                               BECAUSE BITS WILL BE
                                ;                               COPIED INTO C-FF BY
                                ;                               ROTATES, WHICH WILL ACC
                                ;                               -OMPLISH SHIFT FROM R1
                                ;                               INTO R2 OF LSB.
000240  006001                          ROR     R1
000242  006102                          ROL     R2
000244  077404                          SOB     R4,BIV20        ;AT THIS POINT 4 LSB
                                ;                               OF R1 ARE COPIED INTO
                                ;                               IN BIT REVERSED ORDER.
000246  010001                          MOV     R0,R1           ;COPY FOR SWAP
000250  020201                          CMP     R2,R1           ;COMPARE TO PREVENT
                                ;                               DOUBLE SWAP.
000252  002415                          BLT     BIV30
                                ;
000254  006301                          ASL     R1              ;LEFT SHIFT R1 FOR
                                ;                               BYTE ADDRESSING.
000256  006301                          ASL     R1
000260  006302                          ASL     R2
000262  006302                          ASL     R2
000264  066702  177516                  ADD     DATA,R2         ;STARTING ADDRESS
000270  066701  177512                  ADD     DATA,R1         ;DESTINATION ADDRESS
000274  004767  001256                  JSR     PC,SWAP
000300  022122                          CMP     (R1)+,(R2)+     ;ADVANCE TO IMAG PART
000302  004767  001250                  JSR     PC,SWAP         ;SWAP DATA.
                                ;
                                ;
                                ;                       BIT INVERSION CONTINUED...
```

```
                                                ;
                                                ;
                                                ;
000306  005200                          BIV3D:  INC     R0                      ;GO TO NEXT POINT.
000310  010001                                  MOV     R0,R1                   ;COPY R0,R1 FOR INVER
                                                ;                               -ION.
000312  077532                                  SOB     R5,BIV1D
                                                ;
                                                ;               NOTE:
                                                ;               -----
                                                ;                     AT THIS POINT THE BIT INVERSION IS COMPLETE
                                                ;                     AND DATA IS READY TO BE TRANSFORMED.
                                                ;
                                                ;
                                                ;               2. FAST FOURIER TRANSFORMATION (FIRST PASS):
                                                ;               -------------------------------------------
                                                ;                DURING THIS PASS THERE ARE NO MULTIPLICATIONS AND
                                                ;               THE FOLLOWING EQUATIONS ARE IMPLEMENTED:
                                                ;
                                                ;                       R(M)'=R(M)+R(N)
                                                ;                       I(M)'=I(M)+I(N)
                                                ;                       R(N)'=R(M)-R(N)
                                                ;                       I(N)'=I(M)-I(N)
                                                ;
                                                ;
000314  016700  177466                  FPASS:  MOV     DATA,R0                 ;LOAD STARTING ADDRESS
                                                ;                               OF DATA FOR SCALING.
000320  016701  177454                          MOV     N,R1                    ;LOAD # OF POINTS.
000324  004767  001126                          JSR     PC,SCALE                ;ALWAYS SCALE DATA TO
                                                ;                               PREVENT OVERFLOW.
000330  016700  177452                          MOV     DATA,R0
000334  010001                                  MOV     R0,R1                   ;FIRST POINT R(M)
000336  022121                                  CMP     (R1)+,(R1)+             ;2ND POINT R(N)
000340  012705  000010                          MOV     #10,R5                  ;COUNTER=N2/2
                                                ;
000344  011002                          PASS1:  MOV     (R0),R2                 ;SAVE R(M)
000346  061120                                  ADD     (R1),(R0)+              ;R(M)'=R(M)+R(N)
000350  161102                                  SUB     (R1),R2                 ;R(N)'=R(M)-R(N)
000352  010221                                  MOV     R2,(R1)+
000354  011002                                  MOV     (R0),R2
000356  061120                                  ADD     (R1),(R0)+              ;I(M)'=I(M)+I(N)
000360  161102                                  SUB     (R1),R2                 ;I(N)'=I(M)-I(N)
000362  010221                                  MOV     R2,(R1)+
000364  022021                                  CMP     (R0)+,(R1)+
000366  022021                                  CMP     (R0)+,(R1)+             ;GO TO NEXT PAIR
000370  077513                                  SOB     R5,PASS1                ;GO BACK N2/2 TIMES
000372  000412                                  BR      $PASS
                                                ;
                                                ;
                                                ;               NOTE:
                                                ;               -----
                                                ;                     FIRST PASS IS OVER NOW GO TO SECOND PASS.
```

```
                                                    ;
                                                    ;
                                                    ;
                                                    ;
                                                    ;
                                                    ;
                                                    ;
                                                    ;
                                                    ;
                                                    ;
000374  000000                                      ;
                                                    ;
                                                    ;
                                                    ;
                                                    ;
                                                    ;
                                                    ;
                                                    ;
000376  000000                                      ;
                                                    ;
000400  000000                                      ;
                                                    ;
000402  000000                                      ;
                                                    ;
                                                    ;
000404  000000                                      ;
                                                    ;
                                                    ;
000406  000000                                      ;
                                                    ;
                                                    ;
000410  000000                                      ;
                                                    ;
                                                    ;
000412  000000                                      ;
                                                    ;
                                                    ;
000414  000000                                      ;
                                                    ;
                                                    ;
000416  000000                                      ;
                                                    ;
                                                    ;
                                                    ;
                                                    ;
                                                    ;
                                                    ;
```

3. FAST FOURIER TRANSFORM (2ND PASS AND ONWARD)!

THE 2ND,3RD AND4TH PASS USE THE SAME LOOP, HOWEVER IN EACH PASS THE PARAMETERS DEFINED BELOW TAKE DIFFERENT VALUES:

```
DELTAY: 0       ;SIZE OF INCREMENT FOR EXPO
                -NENT OF "W" WHICH SATISF
                -IES THE FOLLOWING RELATIO
                -NSHIP:
                    W=EXP(2*PI*K/N)
                        K=0,1,2.....N-1
                        N= # OF POINTS

CELNUM: 0       ;# OF CELLS IN EACH PASS.

PAIRNM: 0       ;# OF PAIRS IN EACH PASS.

CELDIS: 0       ;DISTANCE BETWEEN PAIR OF POINTS
                ;IN A CELL.

POINT:  0       ;TEMPORARY POINTER TO STARATING
                ADDRESS OF DATA TO TRANSFORMED.

CELCNT: 0       ;CELL COUNTER TO KEEP TRACK
                OF # OF CELLS IN EACH PASS.

PARCNT: 0       ;COUNTER TO KEEP TRACK OF
                # OF PAIRS IN EACH CELL.

SINE:   0       ;TEMPORARY STORAGE TO HOLD
                VAUE FOR SIN(Y).

COS:    0       ;TEMPORARY STORAGE TO HOLD
                VALUE FOR COS(Y).

YN:     0       ;THIS INDICATOR HOLDS THE
                VALUE OF "Y" BEFORE SINE
                COSINE LOOK UP TABLES ARE
                SEARCHED.
```

CONTINUED ON NEXT PAGE.....

```
                                        ;
                                        ;
                                        ;
                                        ;
                                        ;       4. CODE FOR 2ND PASS AND ONWARD:
                                        ;       -------------------------------
                                        ;            THE EQUATIONS THAT ARE IMPLEMENTED IN 2ND,3RD
                                        ;       AND 4TH PASS ARE GIVEN BELOW:
                                        ;
                                        ;               R1'=R1+R2*COS(Y)+I1*SIN(Y)
                                        ;               I1'=I1-R2*SIN(Y)+I2*COS(Y)
                                        ;               R2'=R1-R2*COS(Y)-I2*SIN(Y)
                                        ;               I2'=I1+R2*SIN(Y)-I2*COS(Y)
                                        ;
                                        ;       WHERE Y=2*PI*K/N, N=# OF COMPLEX POINTS, AND
                                        ;       K=0,1,2...N/2-1
                                        ;
                                        ;
                                        ;       NOTE:
                                        ;       -----
                                        ;            3RD AND 4TH PASS START FROM LABEL "NEWPASS"
                                        ;          WHEREAS 2ND PASS STARTS AT LABEL "SPASS".
000420  012767  000004  177746          ;SPASS: MOV     #4,DELTAY
000426  012767  000004  177742          ;       MOV     #4,CELNUM
000434  012767  000002  177736          ;       MOV     #2,PAIRNM
000442  012767  000010  177732          ;       MOV     #10,CELDIS
                                        ;
                                        ;
                                        ;
                                        ;
000450  016700  177332                  ;NEWPAS: MOV    DATA,R0                 ;GET STARTING ADDRESS
                                        ;                                       AND # OF POINTS BEFORE
                                        ;                                       CALLING SUB:"SCALE"
000454  016701  177320                  ;       MOV     N,R1
000460  004767  000772                  ;       JSR     PC,SCALE
                                        ;
000464  016767  177706  177714          ;       MOV     CELNUM,CELCNT           ;SET UP CELL COUNTER.
000472  016767  177310  177704          ;       MOV     DATA,POINT
000500  000404                          ;       BR      NEWCEL
                                        ;
                                        ;
                                        ;
                                        ;
                                        ;            CONTINUED ON NEXT PAGE.....
```

```
                                                        ;
                                                        ;
                                                        ;
                                                        ;       NOTE: EACH NEW CELL WITHIN A PASS STARTS HERE.
                                                        ;       ----  SOME TEMPORARY STORAGE VARIABLES ARE DEFIN
                                                        ;             -ED BELOW:
                                                        ;
000502  000000                                          ;                     I2SINY: 0       ;I2*SIN(Y) IS STORED HERE.
000504  000000                                          ;                     R2SINY: 0       ;R2*COS(Y) IS STORED HERE.
000506  000000                                          ;                     I2COSY: 0       ;I2*COS(Y) IS STORED HERE.
000510  000000                                          ;                     R2COSY: 0       ;R2*COS(Y) IS STORE HERE.
                                                        ;
                                                        ;
                                                        ;
000512  016767  177662  177670                          ;       NEWCEL: MOV     PAIRNM,PARCNT
000520  005067  177672                                  ;               CLR     YN                      ;CLEAR YN
000524  016700  177654                                  ;       NEWC10: MOV     POINT,R0
000530  010001                                          ;               MOV     R0,R1
000532  066701  177644                                  ;               ADD     CELDIS,R1
000536  016702  177654                                  ;               MOV     YN,R2
000542  004767  001060                                  ;               JSR     PC,SNLOK2               ;GET THE VALUE OF
                                                        ;                                               SINE(Y)
                                                        ;
000546  010267  177640                                  ;       FT10:   MOV     R2,SINE                 ;STORE THIS IN "SINE"
000552  062702  040000                                  ;               ADD     #40000,R2
000556  004767  001044                                  ;               JSR     PC,SNLOK2               ;GET VALUE OF COSINE.
000562  010267  177626                                  ;               MOV     R2,COS                  ;STORE THIS IN "COS".
000566  011102                                          ;               MOV     (R1),R2                 ;GET REAL PART R2
000570  070267  177620                                  ;               MUL     COS,R2                  ;R2*COS(Y).
000574  073227  000001                                  ;               ASHC    #1,R2                   ;LEFT SHIFT FOR FRACTIO
                                                        ;                                               -NAL MULTIPLICATION.
000600  010267  177704                                  ;               MOV     R2,R2COSY               ;R2COSY=R2*COS(Y).
000604  011102                                          ;               MOV     (R1),R2                 ;GET REAL PART R2.
000606  070267  177600                                  ;               MUL     SINE,R2                 ;R2*SIN(Y)
000612  073227  000001                                  ;               ASHC    #1,R2
000616  010267  177662                                  ;               MOV     R2,R2SINY               ;R2SINY=R2*SIN(Y).
000622  010104                                          ;               MOV     R1,R4                   ;COPY PRESENT ADDRESS.
000624  005724                                          ;               TST     (R4)+                   ;GET ADDRESS OF IMAG.
000626  011402                                          ;               MOV     (R4),R2                 ;THIS IS VALUE OF I2.
000630  070267  177556                                  ;               MUL     SINE,R2                 ;I2*SIN(Y).
000634  073227  000001                                  ;               ASHC    #1,R2
000640  010267  177636                                  ;               MOV     R2,I2SINY               ;I2SINY=I2*SIN(Y).
000644  011402                                          ;               MOV     (R4),R2
000646  070267  177542                                  ;               MUL     COS,R2                  ;I2*COS(Y).
000652  073227  000001                                  ;               ASHC    #1,R2
000656  010267  177624                                  ;               MOV     R2,I2COSY               ;I2COSY=I2*COS(Y).
                                                        ;
                                                        ;
                                                        ;
                                                        ;
                                                        ;               CONTINUED ON NEXT PAGE.....
```

```
                                                        NOTE:
                                                        -----
                                                             NOW COMBINE TERMS TO GET THE VALUES FOR
                                                        R1',R2',I1' ANDI2'.

000662  011011                                        MOV     (R0),(R1)
000664  066710  177620                                ADD     R2COSY,(R0)             ;(R0)=R1+R2*COS(Y)
000670  066710  177606                                ADD     I2SINY,(R0)             ;R1'=R1+R2*COS(Y)+I2*SIN(Y).
000674  166711  177602                                SUB     I2SINY,(R1)             ;(R1)=R1-I2*SIN(Y).
000700  166711  177604                                SUB     R2COSY,(R1)             ;R2'=R1-I2*SIN(Y)-R2*COS(Y).
000704  010401                                        MOV     R4,R1
000706  005720                                        TST     (R0)+
000710  011011                                        MOV     (R0),(R1)
000712  166710  177566                                SUB     R2SINY,(R0)             ;(R0)=I1-R2*SIN(Y).
000716  066710  177564                                ADD     I2COSY,(R0)             ;I1'=I1=R2*SIN(Y)+I2*COS(Y).
000722  066711  177556                                ADD     R2SINY,(R1)             ;(R1)=I1+R2*SIN(Y).
000726  166711  177554                                SUB     I2COSY,(R1)             ;I2'=I1+R2*SINY-I2*COS(Y).


                                                        NOTE:
                                                        -----
                                                             ONE PAIR OF POINTS HAS BEEN DONE NOW DO
                                                             NEXT PAIR.

000732  062767  000004  177444                          ADD     #4,POINT                ;NEXT ADDRESS IN THE CELL.
000740  066767  177430  177450                          ADD     DELTAY,YN               ;INCREMENT EXPONENT OF W.
000746  005367  177436                                  DEC     PARCNT                  ;ONE PAIR DONE.
000752  001264                                          BNE     NEWC10                  ;DO NEXT PAIR,EXIT IF ALL
                                                                                        PAIRS WITHIN A CELL ARE
                                                                                        DONE.


                                                        NOTE:
                                                        -----
                                                             ONE CELL DONE NOW RESET EXPONENT OF W AND
                                                            PAIR-COUNTER (PARCNT).

000754  066767  177422  177422                          ADD     CELDIS,POINT            ;ADDRESS OF NEXT CELL.
000762  005367  177420                                  DEC     CELCNT                  ;ALL CELL DONE?
000766  001251                                          BNE     NEWCELL                 ;NO,GO BACK.


                                                             CONTINUED ON NEXT PAGE.....
```

```
                                                        NOTE:
                                                        -----
                                                              ONE PASS IS DONE NOW GO ON TO NEXT PASS.
                                                              SET NEW VALUES OF PARAMETERS DEFINED
                                                              IN PARA 3.

000770  006267  177402                                  ASR     CELNUM          ;LESS CELLS THIS TIME.
000774  001420                                          BEQ     POST            ;ALL DONE WHEN VARAIBLE
                                                                                CELNUM=0.
000776  006367  177376                                  ASL     PAIRNM          ;THIS TIME THER ARE
                                                                                MORE PAIRS IN EACH CELL.
001002  006367  177374                                  ASL     CELDIS          ;PAIRS ARE FURTHER APART.
001006  006267  177362                                  ASR     DELTAY          ;LESS INC OF YN PER CELL.
001012  000411                                          BR      POST


                                                        END OF FFT ALGORITHM.
```

```
                                        .SBTTL POST PROCESSING ALGORITHM

                                        ;
                                        ;
                                        ;
                                        ;       1. INTRODUCTION TO POST PROCESSING ALGORITHM:
                                        ;       ---------------------------------------------
                                        ;          POST PROCESSING OF THE TRANSFORMED ARRAY IS DONE
                                        ;       BECAUSE THE INPUT REAL DATA WAS ASSUMED TO BE COMPLEX
                                        ;       AND THE OUTPUT OF THE FFT IS NOT IN CORRECT FORM.
                                        ;          THE EQUATIONS THAT ARE IMPLEMENTED IN THIS PASS
                                        ;       ARE GIVEN BELOW:
                                        ;
                                        ;               AR(M)=RP+IP*COS(T)-RM*SIN(T)
                                        ;               AI(M)=IM-IP*SIN(T)-RM*COS(T)
                                        ;               AR(N-M)=RP-IP*COS(T)+RM*SIN(T)
                                        ;               AI(N-M)=-IM-IP*SIN(T)-RM*COS(T)
                                        ;
                                        ;       WHERE T=PI*M/N, N=16 AND M=0,1,2,...,N/2-1, AND
                                        ;       RP=I(M)+I(N-M), RP=R(M)+R(N-M),IM=I(M)-I(N-M) AND
                                        ;       RM=R(M)-R(N-M).
                                        ;
                                        ;          SOME TEMPORARY VARIABLES THAT ARE USED IN
                                        ;       THIS PASS ARE DEFINED BELOW:
                                        ;
001014  000000                          ;               RP:     0
001016  000000                          ;               IP:     0
001020  000000                          ;               RM:     0
001022  000000                          ;               IM:     0
001024  000000                          ;               ARM:    0       ;ARM=AR(M)
001026  000000                          ;               ARNM:   0       ;ARNM=AR(N-M)
001030  000000                          ;               AIM:    0       ;AIM=AI(M)
001032  000000                          ;               AINM:   0       ;AINM=AI(N-M)
001034  000000                          ;               INX:    0       ;INDEX TO GET SIN,COS VALUES.
                                        ;
                                        ;
                                        ;
                                        ;
                                        ;
                                        ;
                                        ;
                                        ;
                                        ;
                                        ;
                                        ;               CONTINUED ON NEXT PAGE.....
```

```
                                        ;
                                        ;
                                        ;
                                        ;
                                        ;           2. CODE FOR POST PROCESSING :
                                        ;           ----------------------------
                                        ;
                                        ;           NOTE:
                                        ;           -----
                                        ;                SET UP ADDRESS POINTERS AND OTHER RELATED
                                        ;                COUNTERS.
                                        ;
                                        ;
                                        ;
001036  016700  176744                  ;      POST: MOV     DATA,R0                ;SET ADDRESS POINTER
                                        ;                                           AND # OF POINTS BEFORE
                                        ;                                           SCALING DATA ARRAY.
001042  016701  176732                  ;            MOV     N,R1
001046  004767  000404                  ;            JSR     PC,SCALE
001052  016700  176730                  ;            MOV     DATA,R0
001056  012701  000010                  ;            MOV     #10,R1                 ;SET COUNTER TO DECIMAL 8.
001062  012767  000001  177744          ;            MOV     #1,INX                 ;INCREMENT FOR SINE POINTER.
001070  012704  000074                  ;            MOV     #74,R4                 ;OFFSET FOR ADDRESS OF
                                        ;                                           POINT.
001074  066704  176706                  ;            ADD     DATA,R4
                                        ;
                                        ;
                                        ;           NOTE:
                                        ;           -----
                                        ;                THE POST PROCESSING OF FIRST POINT IS DONE
                                        ;                SEPERATELY BECAUSE WE DO NOT PHYSICALLY GE
                                        ;                -NERATE THE (N+1)ST POINT.
                                        ;
                                        ;
001100  016002  000002                  ;      POST1: MOV     2(R0),R2               ;GET IMAG PART OF 1ST
                                        ;                                           POINT.
001104  060210                          ;            ADD     R2,(R0)                ;ADD IT TO REAL PART.
001106  006320                          ;            ASL     (R0)+                  ;DOUBLE THIS RESULT TO
                                        ;                                           MAINTAIN SCALE.
001110  005020                          ;            CLR     (R0)+                  ;CLEAR THE LOCATION WH
                                        ;                                           -ERE THE IMAG WAS STORED.
                                        ;
                                        ;
                                        ;           NOTE:
                                        ;           -----
                                        ;                NOW THE REST OF THE POINTS ARE PROCESSED,SO
                                        ;                GET RM,IM,RP AND IP FOR ALL EXCEPT THE FIRST
                                        ;                POINT.
                                        ;
                                        ;
001112  016702  177716                  ;      POST5: MOV     INX,R2
001116  004767  000600                  ;            JSR     PC,SNLOK1              ;GET THE VALUE OF SINE.
                                        ;
                                        ;
                                        ;
                                        ;                CONTINUED ON NEXT PAGE.....
```

```
                                                ;
                                                ;
                                                ;
                                                ;
                                                ;
001122  010267  177264                          MOV     R2,SINE                 ;SAVE VALUE OF SINE.
001126  016702  177702                          MOV     INX,R2
001132  062702  040000                          ADD     #40000,R2               ;GET VALUE OF COSINE.
001136  004767  000560                          JSR     PC,SNLOK1
001142  010267  177246                  POST7:  MOV     R2,COS                  ;STORE VALUE OF COSINE.
001146  011467  177642                          MOV     (R4),RP                 ;R(N-M)
001152  011467  177642                          MOV     (R4),RM                 ;RM=R(N-M)
001156  005467  177636                          NEG     RM                      ;RM=-R(N-M)
001162  016467  000002  177626                  MOV     2(R4),IP                ;IP=I(N-M)
001170  016467  000002  177624                  MOV     2(R4),IM                ;IM=I(N-M)
001176  005467  177620                          NEG     IM                      ;IM=-I(N-M)
001202  061067  177606                          ADD     (R0),RP                 ;RP=R(M)+R(N-M)
001206  061067  177606                          ADD     (R0),RM                 ;RM=R(M)-R(N-M)
001212  066067  000002  177576                  ADD     2(R0),IP                ;IP=I(M)+I(N-M)
001220  066067  000002  177574                  ADD     2(R0),IM                ;IM=I(M)-I(N-M)
001226  016767  177562  177570          POST10: MOV     RP,ARM
001234  016767  177562  177566                  MOV     IM,AIM                  ;ARM=AR(M) AND AIM=AI(M).
001242  016767  177546  177556                  MOV     RP,ARNM
001250  016767  177546  177554                  MOV     IM,AINM                 ;ARNM=AR(N-M) AND
                                                                                AINM=AI(N-M).
001256  005467  177550                          NEG     AINM                    ;AINM=-AI(N-M).
001262  016702  177530                          MOV     IP,R2
001266  070267  177122                          MUL     COS,R2                  ;IP*COS(T)
001272  073227  000001                          ASHC    #1,R2
001276  060267  177522                  POST20: ADD     R2,ARM                  ;ARM=A(M)+IP*COS(T)
001302  160267  177520                          SUB     R2,ARNM                 ;ARNM=AR(N-M)-IP*COS(T)
001306  016702  177504                  POST25: MOV     IP,R2
001312  070267  177074                          MUL     SINE,R2                 ;IP*SIN(T)
001316  073227  000001                          ASHC    #1,R2
001322  160267  177502                          SUB     R2,AIM                  ;AIM=AI(M)-IP*SIN(T)
001326  160267  177500                          SUB     R2,AINM                 ;AINM=AI(N-M)-IP*SIN(T).
001332  016702  177462                          MOV     RM,R2
001336  070267  177050                          MUL     SINE,R2                 ;RM*SIN(T)
001342  073227  000001                          ASHC    #1,R2
001346  160267  177452                          SUB     R2,ARM                  ;ARM=AR(M)+IP*COS(T)-RM*SIN(T)
001352  060267  177450                          ADD     R2,ARNM                 ;ARNM=AR(N-M)-IP*COS(T);RM*SIN(T)
001356  016702  177436                          MOV     RM,R2
001362  070267  177026                          MUL     COS,R2                  ;RM*COS(T)
001366  073227  000001                          ASHC    #1,R2
001372  160267  177432                  POST50: SUB     R2,AIM                  ;AIM=AI(M)-IP*SIN(T)-RM*COS(T9
001376  160267  177430                          SUB     R2,AINM                 ;AINM=AI(N-M)-IP*SIN(T)-RM*COS(T)
                                                ;
                                                ;
                                                ;
                                                ;
                                                ;
```

CONTINUED ON NEXT PAGE.....

```
                                        ;
                                        ;
                                        ;
                                        ;
001402  016710  177416        POST55: MOV     ARM,(R0)
001406  016714  177414                MOV     ARNM,(R4)
001412  016760  177412  000002        MOV     AIM,2(R0)
001420  016764  177406  000002        MOV     AINM,2(R4)
                                        ;
                                        ;
                                        ;               NOTE:
                                        ;               -----
                                        ;                    ONE PAIR OF POINTS HAS BEEN DONE NOW
                                        ;                    GO ON TO NEXT PAIR OF POINTS.
                                        ;
                                        ;
001426  005267  177402                  INC     INX                     ;INCREMENT SINE INDEX
001432  022020                          CMP     (R0)+,(R0)+             ;INC R0 BY 2
001434  024444                          CMP     -(R4),-(R4)             ;DEC R4 BY 2
001436  005301                          DEC     R1                      ;R1<-R1-1
001440  001402                          BEQ     POSEXT                  ;IF R1=0 EXIT
001442  000167  177444                  JMP     POST5
001446  005367  000002        POSEXT: DEC     $FACTR
001452  000207                          RTS     PC
                                        ;
                                        ;
                                        ;
                                        ;
                                        ;
                                        ;
                                        ;
                                        ;
                                        ;
                                        ;               END OF POST PROCESSING ALGORITHM.
```

```
                                        .SBTTL SUBROUTINE SCALE
                                ;
                                ;
                                ;
                                ;
                                ;       1. INTRODUCTION TO SUBROUTINE "SCALE":
                                ;       ---------------------------------------
                                ;          THIS SUBROUTINE CHECKS BITS 13-14 AND SHIFTS THE
                                ;       WHOLE ARRAY RIGHT BY TWO BITS IF BIT-14 IS SET AND
                                ;       SHIFTS THE ARRAY ONE BIT RIGHT IF BIT-13 IS SET.
                                ;          BEFORE CALLING THIS SUBROUTINE IT SHOULD BE
                                ;       ENSURED THAT THE STARTING ADDRESS OF THE ARRAY
                                ;       IS LOADED IN R0 AND THE LENGHT OF THE ARRAY IS LO
                                ;       -ADED IN R1. THE SUBROUTINE IS CALLED BY GIVING THE
                                ;       COMMAND "JSR  PC,SCALE".
                                ;          THE VARIABLE "SFACTR" WHICH IS DEFINED BELOW
                                ;       KEEPS TRACK OF THE # OF TIMES THE ARRAY HAS BEEN
                                ;       SHIFTED RIGHT BY ONE BIT.
                                ;
                                ;
                                ;
001454  000000                  ;             SFACTR: 0       ;SFACTR IS DEFINED HERE.
                                ;
                                ;
                                ;       2. CODE FOR SUBROUTINE SCALE:
                                ;       -----------------------------
                                ;
001456  010005          SCALE:  MOV     R0,R5           ;GET STARTING ADDRESS
                                ;                       AND STORE IN REG: R5.
001460  010103                  MOV     R1,R3           ;GET ARRAY LENGTH AND
                                ;                       STORE IN REG: R3.
001462  005004                  CLR     R4              ;CLEAR R4 WHICH IS A FLAG
                                ;                       TO KEEP TRACK WHETHER BIT
                                ;                       -14 OR BIT-13 IS SET.
001464  012002          SCAL1:  MOV     (R0)+,R2        ;GET EACH POINT.
001466  100001                  BPL     SCAL2           ;SKIP NEXT INSTRUCTION IF
                                ;                       IF VALUE IS POSITIVE.
001470  005402                  NEG     R2              ;GET ABS VALUE.
001472  032702  040000  SCAL2:  BIT     #40000,R2       ;IS BIT-14 SET?
001476  001011                  BNE     RDUCE2          ;YES,SCALE TWICE.
001500  032702  020000          BIT     #20000,R2       ;IS BIT-13 SET?
001504  001402                  BEQ     SCAL3           ;NO,GET ANOTHER POINT.
001506  012704  000001          MOV     #1,R4           ;YES,SET FLAG AND CHECK
                                ;                       NEXT POINT.
                                ;
                                ;
                                ;
                                ;               CONTINUED ON NEXT PAGE.....
```

```
                                                !
                                                !
                                                !
                                                !
001512  077114                                  !SCAL3: SOB     R1,SCAL1                !ALL POINTS CHECKED?
001514  005704                                  !       TST     R4                      !TEST IF SCALING IS REQUIRED.
001516  001004                                  !       BNE     RDUCE1                  !YES IT IS REQUIRED,
001520  000207                                  !       RTS PC
                                                !
                                                !
                                                !       NOTE:
                                                !       -----
                                                !             SHIFT ENTIRE ARRAY EITHER TWO BITS OR ONE
                                                !             BIT TO THE RIGHT.
                                                !
                                                !
001522  005004                                  !RDUCE2: CLR    R4
001524  005267  177724                          !       INC     SFACTR
001530  005267  177720                          !RDUCE1: INC    SFACTR
001534  005704                                  !       TST     R4
001536  001004                                  !       BNE     RDUCE4
001540  006215                                  !RDUCE3: ASR    (R5)                    !SHIFT EACH POINT TWO
                                                !                                       BITS TO THE RIGHT.
001542  006225                                  !       ASR     (R5)+
001544  077303                                  !       SOB     R3,RDUCE3               !ALL POINTS SCALED?
001546  000207                                  !       RTS     PC                      !YES
001550  006225                                  !RDUCE4: ASR    (R5)+
001552  077302                                  !       SOB     R3,RDUCE4
001554  000207                                  !       RTS     PC
                                                !
                                                !
                                                !
                                                !
                                                !
                                                !
                                                !       END OF SCALING SUBROUTINE.
```

```
                                                .SBTTL  SUBROUTINE "SWAP"
                                ;
                                ;
                                ;
                                ;
                                ;
                                ;               1. SUBROUTINE "SWAP" (INTRODUCTION AND CODE):
                                ;               ---------------------------------------------
                                ;                   THIS SUBROUTINE IS USED TO SWAP THE CONTENTS OF
                                ;               MEMORY LOCATIONS POINTED TO BY REGISTERS R1 AND R2.
                                ;               BEFORE CALLING THIS SUBROUTINE IT SHOULD BE ENSURED
                                ;               THAT THE ADDRESS OF MEMORY LOCATIONS THAT ARE TO SWAP.
                                ;               -PED ARE LOADED IN R1 AND R2.
                                ;
                                ;
                                ;
                                ;
001556  011103                  SWAP:   MOV     (R1),R3
001560  011211                          MOV     (R2),(R1)
001562  010312                          MOV     R3,(R2)
001564  000207                          RTS     PC
                                ;
                                ;
                                ;
                                ;
                                ;
                                ;
                                ;
                                ;
                                ;               END OF SUBROUTINE SWAP.
```

```
                                                        .SBTTL SUBROUTINE "SNLOK2"
                                                ;
                                                ;
                                                ;
                                                ;
                                                ;       1. INTRODUCTION TO "SNLOK2":
                                                ;       ----------------------------
                                                ;          THIS SUBROUTINE IS CALLED FROM FOURIE WHEN THE
                                                ;       ALGORITHM IS EXECUTING 2ND PASS AND ONWARD. THIS SUB
                                                ;       -ROUTINE IS USED TO RETURN VALUES FOR SINE AND COSINE
                                                ;       BY SEARCHING TABLE "SNTAB2".THE ONLY DIFFERENCE BETWEE
                                                ;       -N THIS SUB: AND "SNLOK1" IS THE SINE-COSINE LOOK UP
                                                ;       TABLE. TABLE "SNTAB2" HAS SIXTEEN VALUES STORED,THE FIR
                                                ;       -ST EIGHT VALUES ARE FOR SIN(PI*M/8) AND THE NEXT EIGHT
                                                ;       ARE FOR COS(PI*M/8). IN THIS CASE M=0,,1,2...N/2-1.
                                                ;          THE SINE-COSINE TABLE IS GIVEN BELOW:
                                                ;
001566  000000  030373  055202                  ;       SNTAB2: 000000,030373,055202,073101
001574  073101
001576  077777  073101  055202                  ;               077777,073101,055202,030373
001604  030373
                                                ;
001606  077777  073101  055202                  ;               077777,073101,055202,030373
001614  030373
001616  000000  147404  122575                  ;               000000,147404,122575,104676
001624  104676
                                                ;
                                                ;
                                                ;
                                                ;
                                                ;       2. CODE FOR "SNLOK2":
                                                ;       ---------------------
                                                ;
                                                ;
001626  032702  040000          SNLOK2: BIT     #40000,R2
001632  001001                  BNE     CSLOK2
001634  000404                  BR      LOK2
001636  042702  177770          CSLOK2: BIC     #177770,R2
001642  062702  000010                  ADD     #10,R2
001646  006302                  LOK2:   ASL     R2
001650  016202  001566'                 MOV     SNTAB2(R2),R2
001654  000207                  RTS     PC
                                                ;
                                                ;
                                                ;
                                                ;
                                                ;
                                                ;
                                                ;
                                                ;
                                                ;
                                                ;
                                                ;                       END OF SUBROUTINE "SNLOK2"
```

```
                                                        .SBTTL SUBROUTINE "SNLOK1"
                                                ;
                                                ;
                                                ;
                                                ;
                                                ;
                                                ;       1. INTRODUCTION TO SUBROUTINE "SNLOK1"
                                                ;       -------------------------------------
                                                ;          SUBROUTINE "SNLOK1" IS USED TO LOOK UP THE VALUE
                                                ;       OF SINE AND COSINE TO THE CALLING PROGRAM BY SEARC
                                                ;       -HING A SINE COSINE TABLE "SNTAB1". HITHE CALLING PROGRAM
                                                ;       LOADS R2 WITH A 4-BIT BINARY # CORRESPONDING TO THE DES
                                                ;       -IRED VALUE OF ANGLE . SINCE THIS SUBROUTINE IS USED BY
                                                ;       POST PROCESSING ALGORITHIM ,TABLE "SNTAB1" HAS SINE AND
                                                ;       COSINE  VALUES STORED FOR T=PI*M/16 WHERE M=0,1..N/2 ,
                                                ;       THE SINE COSINE TABLE "SNTAB1" IS GIVEN BELOW,THE FIRST
                                                ;        NINE VALUES ARE FOR SIN(PI*M/16) AND THE NEXT  NINE
                                                ;       VALUES ARE FOR COS(PI*M/16)!
                                                ;
                                                ;
001656  000000  014370  030373                   ;       SNTAB1: 000000,014370,030373
001664  043434  055202  065155                   ;               043434,055202,065155
001672  073101  076612  077777                   ;               073101,076612,077777
                                                ;
001700  077777  076612  073101                   ;               077777,076612,073101
001706  065155  055202  043434                   ;               065155,055202,043434
001714  030373  014370  000000                   ;               030373,014370,000000
                                                ;
                                                ;
                                                ;
                                                ;       2. CODE FOR SUBROUTINE "SNLOK1":
                                                ;       --------------------------------
                                                ;
001722  032702  040000                  SNLOK1: BIT     #40000,R2               ;IS BIT-14 SET?
001726  001001                          LOK1:   BNE     CSLOK1                  ;YES, GET VALUE OF COSINE
001730  000404                                  BR      LOK1
001732  042702  177760                  CSLOK1: BIC     #177760,R2              ;CLEAR BITS 3-15
001736  062702  000011                          ADD     #11,R2                  ;ADD OFFSET FOR COSINE VALUES.
001742  006302                          LOK1:   ASL     R2                      ;SHIFT LEFT FOR BYTE ADDRESSING.
001744  016202  001656'                          MOV     SNTAB1(R2),R2           ;GET VALUE FROM SNTAB1.
001750  000207                                  RTS     PC
                                                ;
                                                ;
                                                ;
        000001'                                 .END
```

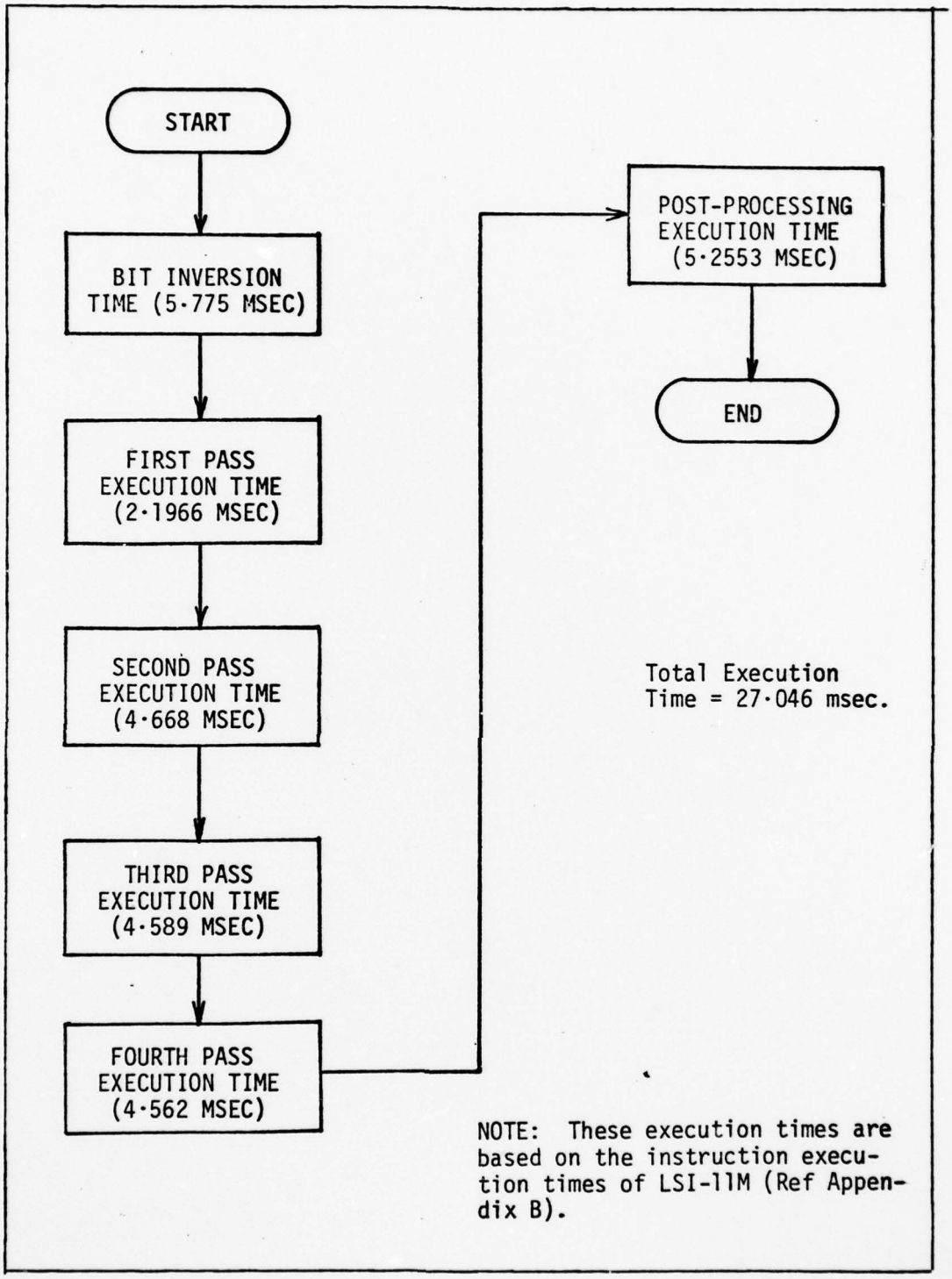


Figure E-1. Execution Time for Subroutine FOURIE

APPENDIX F

LSI-11 ASSEMBLY LANGUAGE CODE FOR POWER SPECTRUM SUBROUTINE (PRSPEC) AND EXECUTION TIME COMPUTATION

```
PRSPEC---POWER SPECTRUM SUBROUT RT-11 MACRO VM02-12    01:28:18 PAGE 1


                                .TITLE PRSPEC---POWER SPECTRUM SUBROUTINE.
                                .SBTTL INTRODUCTION TO "PRSPEC" SUBROUTINE.


                                1. INTRODUCTION TO "PRSPEC" SUBROUTINE:
                                ---------------------------------------
                                   THIS SUBROUTINE CALCULATES THE POWER SPECTRUM OF
                                AN INPUT ARRAY BY SQUARING AND ADDING THE REAL AND
                                IMAGINARY PARTS OF EACH DATA POINT. THE INPUT ARRAY
                                IS ASSUMED TO HAVE REAL AND IMAGINARY PARTS OF EACH
                                DATA POINTS IN CONTIGOUS MEMORY LOCATIONS.
                                   THE INPUT ARRAY IS ASSUMED TO HAVE 32 DATA POINTS,
                                AND IT IS ASSUMED BY THIS SUBROUTINE THAT THE STARTIN
                                -G ADDRESS OF THE INPUT ARRAY IS STORED IN LOCATION
                                "DATA".
                                   THE POWER SPECTRUM OF ONLY THE FIRST 11 DATA POINTS
                                IS CALCULATED,BECAUSE THE FREQUENCY OF INTEREST LIES
                                BETWEEN 0-HZ AND 1000-HZ.
                                THE GLOBAL VARIABLES AND MACRO CALLS ARE DEFINED BELOW:

                                        .GLOBL SCALE,DATA,SPCTRA,SSTORE
                                        .GLOBL PRSPEC,SFACTR,N,CUMSF1


                                        .MCALL  .REGDEF
000000                                  .REGDEF


                                        .CSECT
000000'
```

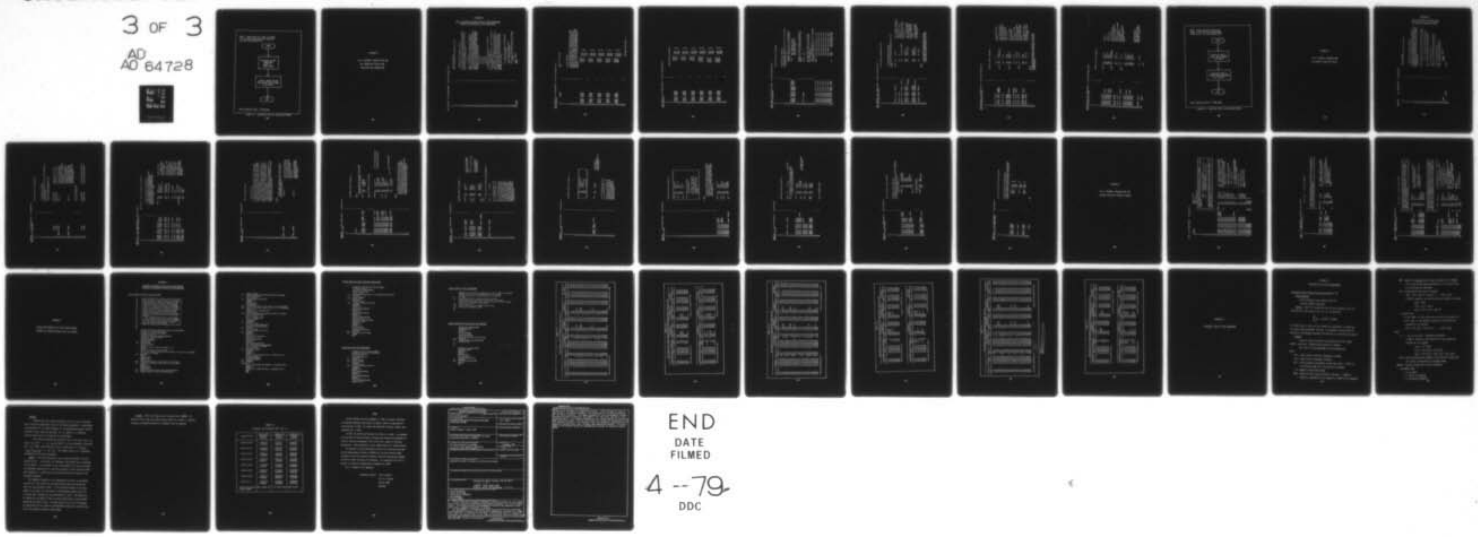
```
                                .SBTTL DATA STRUCTURE AND VARIABLE DEFINITION.



                                2.DATA STRUCTURE AND VARIABLES USED:
                                ------------------------------------
                                   THE VARIABLES USED IN THIS SUBROUTINE ARE DEFINED
                                BELOW:

000000  000000                          CUMSF1: 0       ;CUMMULATIVE SCALE FACTOR
                                                        WHICH KEEPS TRACK OF THE
                                                        NUMBER OF TIMES THE OUTPUT
                                                        ARRAY HAS BEEN SHIFTED ONE
                                                        PLACE TO THE RIGHT.
000002  000000                          RLSQR:  0       ;TEMPORARY VARIABLE THAT
                                                        STORES THE SQUARE OF REAL
                                                        PART OF A DATA POINT.


                                   THE DATA STRUCTURE FOR THE OUTPUT ARRAY IS GIVEN
                                BELOW. THE OUTPUT ARRAY IS CALLED "SPCTRA"
                                AND IS 11 WORDS LONG:

000004  000006'                          SPCTRA: .WORD SSTORE    ;STARTING ADDRESS OF
                                                                THE ARRAY "SPCTRA".
000006  000024                  SSTORE: .REPT 24
                                        .WORD 0
                                        .ENDM
                                .=.+26
000104'
```

```
                                                .SBTTL CODE FOR "PRSPEC".


                                                3. CODE FOR "PRSPEC":
                                                ---------------------
                                                   THE CODE FOR POWER SPECTRUM SUBROUTINE STARTS
                                                BELOW:

000104  016700  177674                          PRSPEC: MOV     SPCTRA,R0               ;LOAD THE STARTING
                                                                                        ADDRESS OF THE
                                                                                        OUTPUT ARRAY.
000110  016767  177664  000000G                         MOV     CUMSF1,SFACTR
000116  012701  000013                                  MOV     #13,R1                  ;MOVE THE # OF POINTS
                                                                                        IN THE OUTPUT ARRAY
                                                                                        TO R1.
000122  004767  000000G                                 JSR     PC,SCALE                ;SCALE THIS ARRAY.
000126  016767  000000G 177644                          MOV     SFACTR,CUMSF1
000134  016700  000000G                                 MOV     DATA,R0                 ;GET STARTING ADDRESS
                                                                                        OF INPUT ARRAY.
000140  016704  177640                                  MOV     SPCTRA,R4               ;GET STARTING ADDRESS
                                                                                        OF THE OUTPUT ARRAY.
000144  012701  000013                                  MOV     #13,R1
000150  012002                                  PRS10:  MOV     (R0)+,R2                ;REAL PART.
000152  070202                                          MUL     R2,R2                   ;SQUARE THE REAL
                                                                                        PART.
000154  073227  000001                                  ASHC    #1,R2
000160  010267  177616                                  MOV     R2,RLSQR                ;STORE THIS VALUE.
000164  012002                                          MOV     (R0)+,R2                ;IMAG PART.
000166  070202                                          MUL     R2,R2                   ;SQUARE THE IMAG PART.
000170  073227  000001                                  ASHC    #1,R2
000174  066702  177602                                  ADD     RLSQR,R2                ;ADD SQUARE OF REAL
                                                                                        AND IMAG PART.
000200  060224                                          ADD     R2,(R4)+                ;ADD THIS VALUE TO
                                                                                        THE PREVIOUS VALUE IN
                                                                                        THE ARRAY.
000202  077116                                          SOB     R1,PRS10                ;ALL POINTS DONE?
                                                                                        NO:GO AND DO MORE.
000204  000207                                          RTS     PC                      ;YES..RETURN.

        000001'                                         .END
```

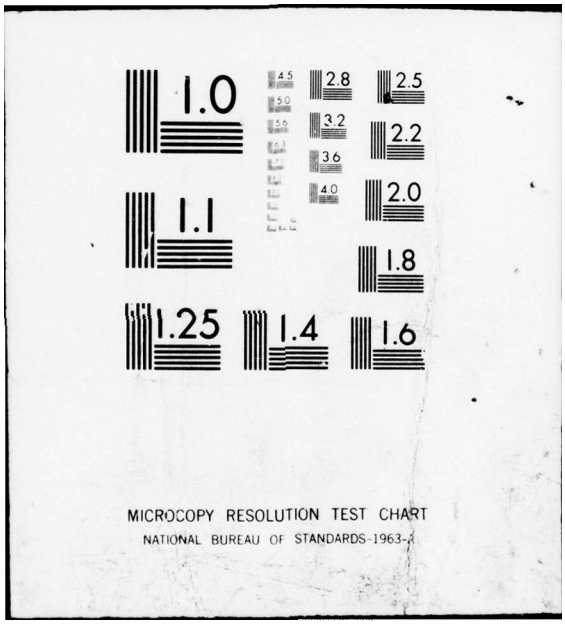
AD-A064 728 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 9/2
MICROPROCESSOR BASED DATA ACQUISITION AND PROCESSING SYSTEM.(U)
DEC 78 S IFTEKHAR

UNCLASSIFIED AFIT/GE/EE/78-29 NL

3 OF 3
AD
AD 64728



END
DATE
FILMED
4 -79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

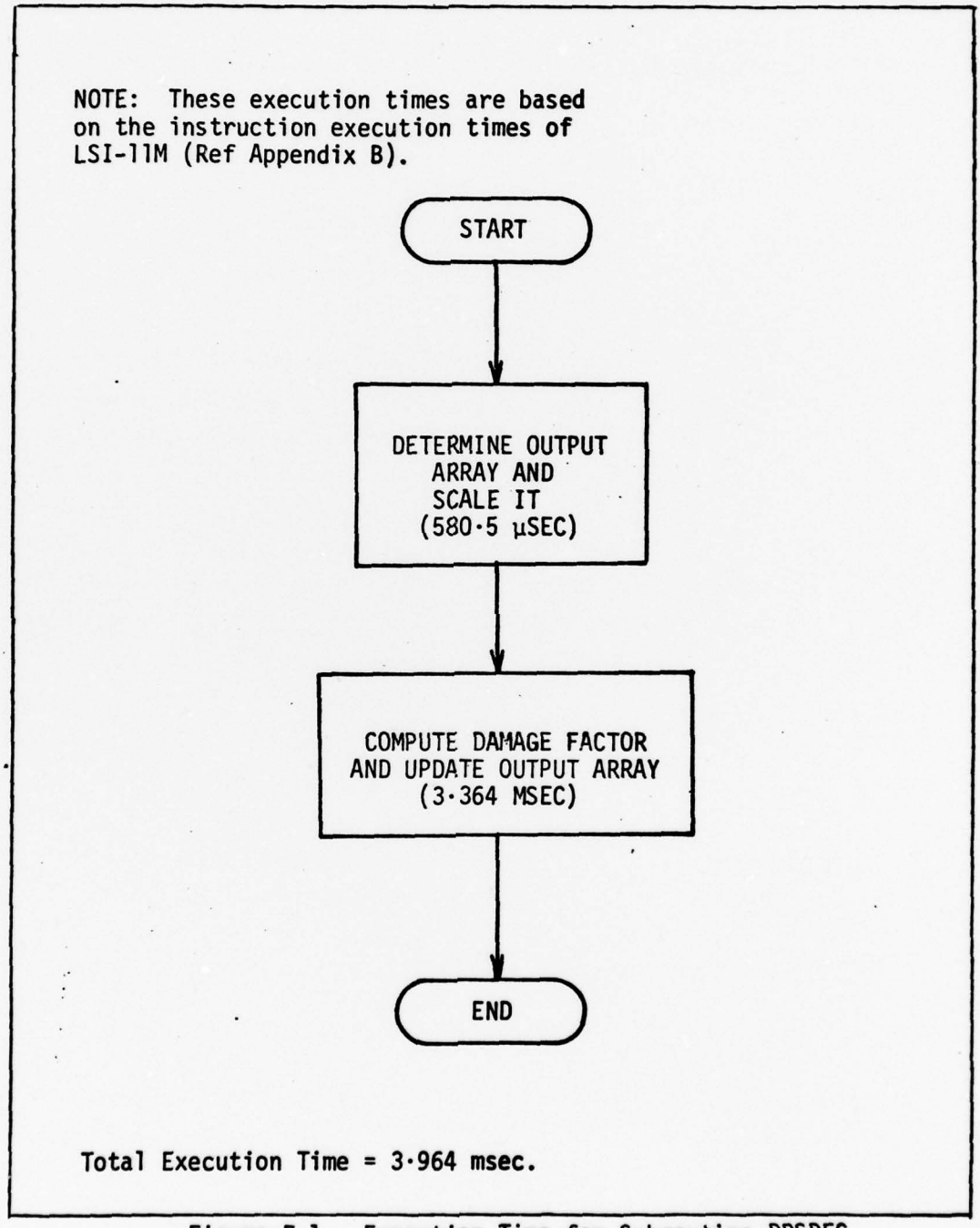


Figure F-1. Execution Time for Subroutine PRSPEC

APPENDIX G

LSI-11 ASSEMBLY LANGUAGE CODE FOR CDF SUBROUTINE (POLYN) AND EXECUTION TIME COMPUTATION

```
POLY CURVE FIT  RT-11 MACRO VM02-12   30-OCT-78 23:58:05 PAGE 1

                .TITLE POLY CURVE FIT
                .SBTTL INTRODUCTION TO SUBROUTINE "POLYN".



                1. INTRODUCTION TO SUBROUTINE "POLYN":
                ---------------------------------------
                   THIS SUBROUTINE IS USED TO APPROXIMATE THE FUCTION
                THAT IS USED TO CALCULATE THE DAMAGE FACTOR(DF) AT
                THE FREQUENCY OF INTEREST. THE DF CAN BE CALCULATED
                USING THE FOLLOWING RELATIONSHIP:

                        DF=(X)**1.6
                WHERE X IS POWER DENSITY VALUE AT THE FREQUENCY OF
                INTEREST. THE VALUE OF X LIES BETWEEN 0.0 AND 1.0
                BECAUSE THE SYSTEM WORKS WITH FRACTIONAL NUMBERS.
                   A POLYNOMIAL CURVE FIT PROGRAM WAS USED TO APPO/ROX
                -IMATE THE DF CURVE, AND IT WAS DETERMINED THAT THE
                CURVE SHOULD BE DIVIDED INTO FOLLOWING INTERVALS.EACH
                INTERVAL CAN THEN BE APPROXIMATED BY A SECOND DEGREE
                POLYNOMIAL WITH MINIMUM RMS ERROR:
                        0.0 TO 0.1
                        0.1 TO 0.2
                        0.2 TO 0.3
                        0.3 TO 0.4
                        0.4 TO 0.6
                        0.6 TO 0.8
                        0.8 TO 1.0
                   THE POLYNOMIAL EQUATION USED TO APPROXIMATE
                DF FUNCTION IS GIVEN BELOW:
                        Y=A*X**2+B*X+C
                FOR EACH INTERVAL DEFINED ABOVE THERE IS A UNIQUE SET
                OF COEFFICIENTS A,B,C.
                   THESE COEFF: ARE STORED AT LOCATION STARTING AT
                LABEL "COEFF:". THE FIRST VALUE FOR EACH INTERVAL IS
                USED TO TEST FOR THE INTERVAL IN WHICH X LIES.
                   ALSO IF THE VALUE FOR X IS LESS THAN 0.04 THEN
                ANOTHER SET OF COEFF: A,B,C STORED AT LABEL "CF:"
                ARE USED. THIS HAS BEEN DONE TO MINIMIZE THE ERROR
                FOR VERY SMALL VALUES.


                        NOTE:THE GLOBAL VARIABLES AND SUBROUTINES
                             ARE DEFINED BELOW.

                        .GLOBL SSTORE,TFLG,SCALE,SPCTRA
                        .GLOBL  POLYN,SFACTR,CDFAC

                        NOTE:THE REGISTER DEFINITION MACRO IS DEFINED
                             BELOW:

                        .MCALL  .REGDEF
                        .REGDEF
 000000
```

```
                                                .SBTTL DATA STRUCTURE.
                                                .CSECT
                                ;
                                ;
                                ;
                                ;
                                ;               2. DATA STRUCTURE:
                                ;               ------------------
                                ;                  THE DATA STRUCTURE FOR THE FINAL ARRAY,CDF VERSUS
                                ;               FREQUENCY AND TEMERATURE,IS GIVEN BELOW.THERE ARE 11
                                ;               ARRAYS CORRESPONDING TO 11 TEMERATURE INTERVALS OF
                                ;               INTEREST. THE FIRST ELEMENT IN EACH ARRAY GIVES THE
                                ;               VALUE FOR TEMPRATURE INTERVAL,THE 2ND VALUE GIVES THE
                                ;               SCALE FACTOR FOR THAT ARRAY,AND THE REMAINING 11 ELEM
                                ;               -ENTS ARE THE CDF VALUES FOR THE FREQUENCY RANGE OF
                                ;               INTEREST:
                                ;
                                ;                       ARRAY0
000000  000000                  ;               TEMP0:  .WORD   0
000002  000000                  ;               DFSF0:  .WORD   0
000004  000024                  ;               CSTR0:  .REPT   24
                                ;                       .WORD   0
                                ;                       .ENDM
                                ;
                                ;                       ARRAY1
000054  000001                  ;               TEMP1:  .WORD   1
000056  000000                  ;               DFSF1:  .WORD   0
000060  000024                  ;               CSTR1:  .REPT   24
                                ;                       .WORD   0
                                ;                       .ENDM
                                ;
                                ;                       ARRAY2
000130  000000                  ;               TEMP2:  .WORD   0
000132  000002                  ;               DFSF2:  .WORD   2
000134  000024                  ;               CSTR2:  .REPT   24
                                ;                       .WORD   0
                                ;                       .ENDM
                                ;
                                ;                       ARRAY3
000204  000003                  ;               TEMP3:  .WORD   3
000206  000000                  ;               DFSF3:  .WORD   0
000210  000024                  ;               CSTR3:  .REPT   24
                                ;                       .WORD   0
                                ;                       .ENDM
                                ;
                                ;                       ARRAY4
000260  000004                  ;               TEMP4:  .WORD   4
000262  000000                  ;               DFSF4:  .WORD   0
000264  000024                  ;               CSTR4:  .REPT   24
                                ;                       .WORD   0
                                ;                       .ENDM
                                ;
                                ;
                                ;
                                ;                       CONTINUED ON NEXT PAGE.....
```

```
                                                        DATA STRUCTURE CONTINUED.....

                                                              ARRAY5
000334  000005                                        TEMP5:  .WORD   5
000336  000000                                        DFSF5:  .WORD   0
000340  000024                                        CSTR5:  .REPT   24
                                                              .WORD   0
                                                              .ENDM

                                                              ARRAY6
000410  000006                                        TEMP6:  .WORD   6
000412  000000                                        DFSF6:  .WORD   0
000414  000024                                        CSTR6:  .REPT   24
                                                              .WORD   0
                                                              .ENDM

                                                              ARRAY7
000464  000007                                        TEMP7:  .WORD   7
000466  000000                                        DFSF7:  .WORD   0
000470  000024                                        CSTR7:  .REPT   24
                                                              .WORD   0
                                                              .ENDM

                                                            ARRAY10
000540  000010                                        TEMP10: .WORD   10
000542  000000                                        DFSF10: .WORD   0
000544  000024                                        CSTR10: .REPT   24
                                                              .WORD   0
                                                              .ENDM

                                                            ARRAY11
000614  000011                                        TEMP11: .WORD   11
000616  000000                                        DFSF11: .WORD   0
000620  000024                                        CSTR11: .REPT   24
                                                              .WORD   0
                                                              .ENDM

                                                            ARRAY12
000670  000012                                        TEMP12: .WORD   12
000672  000000                                        DFSF12: .WORD   0
000674  000024                                        CSTR12: .REPT   24
                                                              .WORD   0
                                                              .ENDM
```

```
                                                        .SBTTL DEFINITION OF VARIABLES
                                                ;
                                                ;
                                                ;
                                                ;
                                                ;       3. DEFINITION OF VARAIBLES:
                                                ;       ---------------------------
                                                ;          THE VARAIABLES AND COEFFICIENTS USED IN THIS SUROU
                                                ;       -TINE ARE DEFINED BELOW:
                                                ;
                                                ;              VARIABLE "CDFAC" IS USED AS AN POINTER THAT
                                                ;              CONTAINS THE ADDRESS OF THE OUTPUT ARRAY.
                                                ;
000744  000004' 000060' 000134'                  ;              CDFAC:  .WORD CSTR0,CSTR1,CSTR2
000752  000210' 000264' 000340'                  ;                      .WORD CSTR3,CSTR4,CSTR5
000760  000414' 000470' 000544'                  ;                      .WORD CSTR6,CSTR7,CSTR10
000766  000620' 000674'                          ;                      .WORD CSTR11,CSTR12
                                                ;
                                                ;
                                                ;              VARIABLE "CSTSAV" IS USED AS A TEMPORARY
                                                ;              STORAGE FOR THE CALCULATED ADRESS OF OUTPUT
                                                ;              ARRAY.
                                                ;
000772  000000                                  ;              CSTSAV: .WORD   0
                                                ;
                                                ;
                                                ;              THE COEFFICIENTS USED IN THE POLYNOMIAL
                                                ;              ARE GIVEN BELOW. THEY ARE STORED IN THE
                                                ;              FOLLOWING MANNER:
                                                ;                      .WORD FLAG,C,B,A
                                                ;              "FLAG" IS THE VALUE WHICH IS USED TO DETERMINE
                                                ;              THE INTERVAL IN WHICH X LIES.
                                                ;
000774  006314  062746  003136                  ;              COEFF:  .WORD 006314,062746,003136,177765
001002  177765
001004  014631  040405  006470                  ;                      .WORD 014631,040405,006470,177641
001012  177641
001014  023146  032444  010766                  ;                      .WORD 023146,032444,010766,177477
001022  177477
001024  031463  027205  012763                  ;                      .WORD 031463,027205,012763,177214
001032  177214
001034  046314  024157  015416                  ;                      .WORD 046314,024157,015416,176571
001042  176571
001044  063146  021533  020441                  ;                      .WORD 063146,021533,020441,175646
001052  175646
001054  017774  023205  174575                  ;                      .WORD       017774,023205,174575
001062  044601  000726  177776                  ;              CF:     .WORD       044601,000726,177776
```

```
                                                        .SBTTL CODE FOR SUBROUTINE "POLYN"
                                                ;
                                                ;
                                                ;
                                                ;
                                                ;
                                                ;       4. CODE FOR SUBROUTINE "POLYN":
                                                ;       -------------------------------
                                                ;          THE CODE FOR POLYNOMIAL SUBROUTINE IS GIVEN
                                                ;       BELOW:
                                                ;
001070  016701  000000G                         ;       POLYN:  MOV     TFLG,R1                 ;GET THE VALUE OF TEMP
                                                ;                                               FLAG TO SELECT APPROPRIATE
                                                ;                                               OUTPUT ARRAY.
001074  006301                                  ;               ASL     R1                      ;SHIFT LEFT FOR BYTE
                                                ;                                               ADDRESSING.
001076  016167  000744' 177666                  ;               MOV     CDFAC(R1),CSTSAV        ;STORE THE CALCULATED
                                                ;                                               ADDRESS OF OUTPUT ARRAY
                                                ;                                               IN "SCTSAV".
001104  016700  177662                          ;               MOV     CSTSAV,R0
001110  012701  000013                          ;               MOV     #13,R1                  ;THE STARTING ADDRESS
001114  016067  177776  000000G                 ;               MOV     -2(R0),SFACTR           ;AND THE # OF POINTS
                                                ;                                               AND THE SCALE FACTOR
                                                ;                                               OF OUTPUT ARRAY ARE
                                                ;                                               PASSED TO SUBROUTINE
001122  004767  000000G                         ;               JSR     PC,SCALE                ;"SCALE.
001126  016700  177640                          ;               MOV     CSTSAV,R0
001132  016760  000000G 177776                  ;               MOV     SFACTR,-2(R0)           ;THE VALUE OF SFACTR
                                                ;                                               RETURNED BY "SCALE"
                                                ;                                               IS STORED BACK.
001140  010067  000074                          ;               MOV     R0,PLY35+2
                                                ;
                                                ;       NOTE:
                                                ;       -----
                                                ;               AT THIS STAGE THE OUTPUT ARRAY HAS BEEN
                                                ;               SCALED AND ITS ADDRESS PASSED TO THE CODE
                                                ;               WHICH STORES THE CALCULATED VALUE OF CDF.
```

```
                                                ;
                                                ;
                                                ;
                                                ;
                                                ;           CODE FOR "POLYN" CONTINUED.......
                                                ;
                                                ;
                                                ;
                                                ;
001144  005005                                  ;
                                                ;                   CLR     R5              ;CLEAR DISPLACEMENT
                                                ;                                           REGISTER.
001146  012701  000013                          ;                   MOV     #13,R1
001152  016500  000000G                         ;           PLY10:  MOV     SSTORE(R5),R0   ;TEST THE FIRST POINT
                                                ;                                           POINT IN THE INPUT
                                                ;                                           ARRAY FOR A ZERO VALUE.
001156  001002                                  ;                   BNE     PLY15           ;BRANCH TO PLY5 IF
                                                ;                                           FIRST DATA POINT IS
                                                ;                                           IS NOT EQUAL TO  0.
001160  010002                                  ;                   MOV     R0,R2           ;MOVE 0 TO R2.
001162  000425                                  ;                   BR      PLY35
001164  022700  002436                          ;           PLY15:  CMP     #2436,R0        ;IS R0 LE TO 0.04.
001170  002027                                  ;                   BGE     SMALL           ;YES,GO AND USE
                                                ;                                           DIFFERENT SET OF
                                                ;                                           COEFF:
001172  012703  000006                          ;                   MOV     #6,R3           ;NO,CHECK FOR OTHER
                                                ;                                           INTERVALS.
001176  012704  000774'                         ;                   MOV     #COEFF,R4       ;LOAD STARTING ADDRESS
                                                ;                                           FOR COEFF INTO R4.
001202  020024                                  ;           PLY20:  CMP     R0,(R4)+        ;COMPARE PRESENT DATA
                                                ;                                           POINT TO SELECT PROPER
                                                ;                                           INTERVAL.
001204  003403                                  ;                   BLE     PLY30
001206  062704  000006                          ;                   ADD     #6,R4           ;GET ADDRESS FOR NEXT
                                                ;                                           INTERVAL.
001212  077305                                  ;                   SOB     R3,PLY20        ;ALL INTERVALS CHACKED?
                                                ;                                           IF NOT GO BACK.
                                                ;
                                                ;
                                                ;           NOTE:
                                                ;           -----
                                                ;           AT THIS POINT R5 CONTAINS DISPLACEMENT INTO SSTORE
                                                ;           AND CSTRO, AND R4 POINTS TO FIRST COEFF IN THE SEL
                                                ;           -ECTED INTERVAL.
```

```
                                                ;
                                                ;
                                                ;
                                                ;           CODE FOR "POLYN" CONTINUED.....
                                                ;
                                                ;
                                                ;
                                                ;
                                                ;
001214  012402                          PLY30:  MOV     (R4)+,R2        ;R2=C   *
001216  070200                                  MUL     R0,R2           ;R2=X*C
001220  073227  000001                          ASHC    #1,R2
001224  062402                                  ADD     (R4)+,R2        ;R2=B+X*C
001226  070200                                  MUL     R0,R2           ;R2=X*(B+X*C)
001230  073227  000001                          ASHC    #1,R2
001234  062402                                  ADD     (R4)+,R2        ;R2=A+X*(B+X*C)
001236  060265  000004'                  PLY35:  ADD     R2,CSTRD(R5)    ;ADD THIS CALCULATED
                                                ;                       VALUE TO PREVIOUS VALUE
                                                ;                       VALUE IN THE ARRAY.
001242  005725                                  TST     (R5)+           ;R5=R5+2
001244  077136                                  SOB     R1,PLY10        ;ALL DATA POINTS FINISHED?
                                                ;                       IF NOT GO BACK.
001246  000207                                  RTS     PC              ;YES, RETURN FROM
                                                ;                       SUBROUTINE.
001250  012704  001062'                 SMALL:  MOV     #CF,R4          ;MOVE STARTING ADDRSEE
                                                ;                       OF COEFF FOR VERY
                                                ;                       SMALL X.
001254  012402                                  MOV     (R4)+,R2
001256  070200                                  MUL     R0,R2
001260  073227  000001                          ASHC    #1,R2
001264  062402                                  ADD     (R4)+,R2
001266  070200                                  MUL     R0,R2
001270  073227  000001                          ASHC    #1,R2
001274  061402                                  ADD     (R4),R2
001276  006302                                  ASL     R2              ;THE CODE HERE IS
                                                ;                       SAME AS IN PLY30
                                                ;                       EXECPT THAT THE FINAL
                                                ;                       FINAL RESULT IS  HALVED.
                                                ;                       THEREFORE WE DOUBLE IT. .
001300  000756                                  BR      PLY35
                                                ;
                                                ;
        000001'                                 .END
```

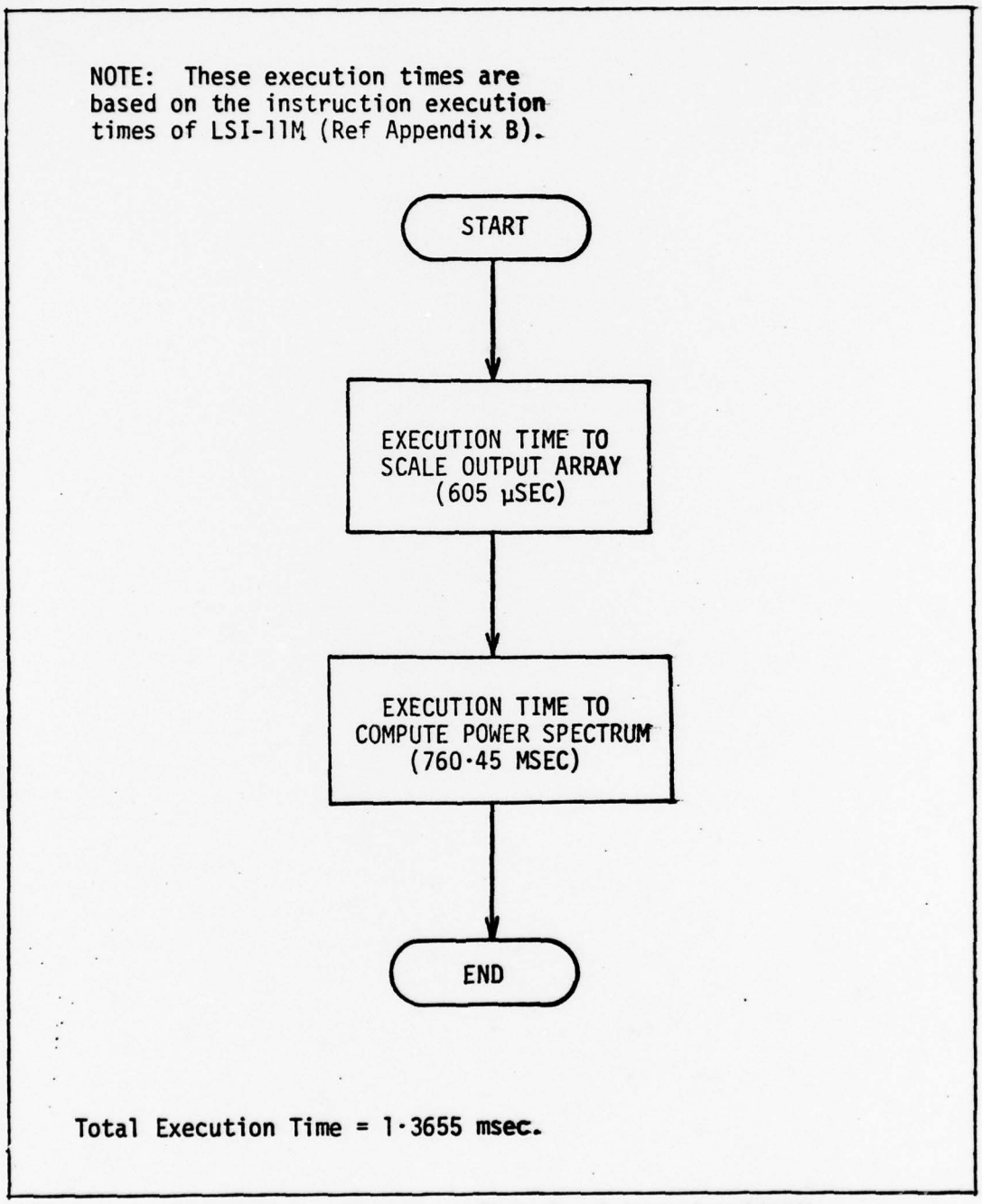


Figure G-1. Execution Time for Subroutine POLYN

APPENDIX H

LSI-11 ASSEMBLY LANGUAGE CODE

FOR CONTROL EXECUTIVE (EXEC)

```
EXECUTIVE       RT-11 MACRO VM02-12     00:06:08 PAGE 1

                                        .TITLE EXECUTIVE
                                        .SBTTL INTRODUCTION TO CONTROL EXECUTIVE
                                ;
                                ;
                                ;
                                ;
                                ;       1. INTRODUCTION TO CONTROL EXECUTIVE:
                                ;       -------------------------------------
                                ;          THIS PART OF THE CONTROL EXECUTIVE GETS CONTROL OF
                                ;       THE SYSTEM ONCE POWER HAS BEEN SWITCHED ON AND THE
                                ;       RUN/STANDBY SWITCH HAS BEEN PLACED IN RUN POSITION. THE
                                ;       EXECUTIVE WILL RELINQUISH CONTROL IF EITHER THE POWER
                                ;       HAS BEEN SWITCHED OFF OR THE SYSTEM HAS BEEN SWITCHED
                                ;       TO STANDBY MODE.
                                ;
                                ;
                                ;       NOTE: THE GLOBL DATA STORAGE AREAS ARE DEFINED BELOW:
                                ;       ----------------------------------------------------
                                ;
                                ;               .GLOBL  DATA,ISTR1,ISTR2,CDATA,SPCTRA,TFLG
                                ;
                                ;       NOTE: THE GLOBAL SUBROUTINES ARE DEFINED BELOW:
                                ;       -----------------------------------------------
                                ;
                                ;       .GLOBL  FOURIE,PRSPEC,POLYN
                                ;
                                ;       NOTE: THE GLOBAL VARIABLES ARE DEFINRD BELOW:
                                ;       ---------------------------------------------
                                ;
                                ;       .GLOBL  SFACTR,CUMSF1,TSUM,TFLG
                                ;
                                ;       NOTE: THE REGISTER ARE DEFINED BELOW:
                                ;       -------------------------------------
                                ;
                                ;               .MCALL  .REGDEF .
                                ;               .REGDEF
000000                          ;
                                ;       .CSECT
                                ;
                                ;
```

```
                                        .SBTTL DEFINITION OF DEVICES USED
                                ;
                                ;
                                ;
                                ;
                                ;
                                ;       2. DEFINITION OF DEVICES:
                                ;       -------------------------
                                ;          THE DEVICES THAT ARE CALLED IN BY THE EXECUTIVE
                                ;       ARE DEFINED HERE.
                                ;
                                ;        ANALOG TO DIGITAL CONVERTOR (ADC)
                                ;       ----------------------------------
                                ;
                                ;
                                ;
                                ;       ADCSR=170400            ;ADC CONTROL AND STATUS
                                ;                               ;REGISTER.
                                ;       ADBFR=170402            ;ADC DATA BUFFER REGISTER
                                ;
000000  000000                  ;       ADFLG:  0               ;ADC FLAG WILL BE USED
                                ;                               ;TO INDICATE THAT 32 DATA
                                ;                               ;POINTS HAVE BEEN READ IN
                                ;                               ;AND STORED AT THE ALLOCATED
                                ;                               ;STORAGE AREA.
                                ;                               ; THIS FLAG IS CLEARED BY
                                ;                               ;THE EXECUTIVE BEFORE ADC
                                ;                               ;INTERRUPT IS ENABLED
000002  000000                  ;       ADCNT:  0               ;THIS COUNTER KEEPS TRACK
                                ;                               ;OF NUMBER OF POINTS TO BE
                                ;                               ;CONVERTED AND STORED. IT
                                ;                               ;IS LOADED BY THE EXEC BEFO
                                ;                               ;-RE ADC IS ENABLED.
                                ;
                                ;        REAL TIME CLOCK (RTC)
                                ;       --------------------
                                ;
                                ;
                                ;
                                ;       RTCSR=170420            ;RTC CONTROL AND STATUS REG:
                                ;                               ;
                                ;       RTBFR=170422            ;RTC DATA BUFFER REGISTER
                                ;
```

```
                                        .SBTTL FPASS --FIRST PASS THROHGH THE EXECUTIVE
                                ;
                                ;
                                ;
                                ;               3. FIRST PASS THROUGH THE EXECUTIVE:
                                ;               ------------------------------------
                                ;                  FIRST TIME THROUGH THERE ARE NO DATA POINTS
                                ;               AVAILABLE THEREFORE THE PROCESSOR REMAINS
                                ;               IDLE AND ADC GATHERS 32 DATA POINTS. THE
                                ;               DATA POINTS ARE STILL BEING CONVERTED AND
                                ;               STORED VIA INTERRUPT SCHEME.
                                ;
                                ;

000004  005067  177770          EXEC:   CLR     ADFLG                   ;CLEAR ADC FLAG
000010  012767  000520' 000476          MOV     #ISTR1,CDATA
000016  012767  000040  177756          MOV     #40,ADCNT               ;SET COUNTER TO 32.
000024  012737  000500' 000440          MOV     #RTCSRV,@#440           ;SET ADDRESS OF RTC
                                                                        ;SEVICE ROUTINE AT
                                                                        ;VECTOR ADDRESS
000032  012737  000100  000442          MOV     #100,@#442
000040  012767  177740  170422'          MOV     #177740,RTBFR           ;SET RTC BUFFER WITH
                                ;                                       REQUIRED COUNT.
000046  012737  000426' 000400          MOV     #ADCSRV,@#400           ;SET ADDRESS OF ADC
                                                                        ;SERVICE ROUTINE AT
                                                                        ;VECTOR ADDRESS
000054  012737  000100  000402          MOV     #100,@#402              ;SET PROCESSOR PRIO
                                                                        ;--RITY TO 0
000062  106427  000100          MTPS    #100                    ;SET PROCESSOR PRIO
                                ;                                       -RITY TO 0
000066  012767  007520  170400'          MOV     #7520,ADCSR             ;ENABLE ADC
000074  000001                  START2: WAIT                            ;WAIT FOR INTERRUPTS.
000076  032767  000001  177674          BIT     #1,ADFLG                ;HAVE 32 POINTS BEEN
                                                                        ;CONVERTED?
000104  001773                          BEQ     START2                  ;NO. WAIT
000106  000402                          BR      CONT1
```

```
                                                .SBTTL PROCESSING CONTROL
                                ;
                                ;
                                ;
                                ;
                                ;               4. PROCESSING CONTROL:
                                ;               ----------------------
                                ;                  EACH SET OF 32 DATA POINTS ARE FIRST FOURIE TRANS
                                ;               -FORMED USING SUBROUTINE "FOURIE". THIS IS FOLLOWED
                                ;               BY COMPUTATION OF POWER SPECTRAL DENSITY.THIS PROCESS
                                ;               IS REPEATED FOR TEN SETS OF DATA POINTS AND AN AVERAGE
                                ;               POWER SPECTRUM OBTAINED.NEXT THE DAMAGE FACTOR IS CAL
                                ;               -CULATED AT EACCH FREQUENCY.
                                ;                  EVERY TIME THE POWER SPECTRUM IS CALCULATED THE
                                ;               TEMPERATURE SIGNAL IS SAMPLED AND ACQUIRED. THIS ACQUI
                                ;               -RED VALUE IS SUMMED IN A  STORAGE VARIABLE "TSUM".
                                ;               THEREFORE AFTER THE AVERAGE POWER SPECTRUM IS OBTAINED
                                ;               ( AFTER 10 LOOP EXECUTIONS) THE AVERAGE TEMPERATURE
                                ;               VALUE IS STORED IN "TSUM"
                                ;                  EVERY TIME THE PROCESSING LOOP IS STARTED THE ADC'IS
                                ;               ENABLED AND NEXT 32 DATA POINTS ARE GATHERED VIA
                                ;               INTERRUPT SCHEME. THE REAL TIME CLOCK IS USED TO SYNCHRO
                                ;               -NIZE DIFFERENT PROCESSING LOOPS.
                                ;
                                ;
                                ;
                                ;               NOTE:
                                ;               -----
                                ;                     BEFORE THESE POINTS ARE PROCESSED THE PROCESS
                                ;                  LOOP COUNTERS (PRCLOP AND ADCLOP) ARE DEFINED
                                ;                  AND INITIALIZED BELOW:
                                ;
000110  000000                                      ADCLOP: 0               ;THIS COUNTER IS USED TO
                                                                            ;TELL THE ADC WHICH BUFFER
                                                                            ;STORAGE SHOULD BE USED TO
                                                                            ;STORE CONVARTED DATA. IT
                                                                            ;IS ALSO USED TO DETERMINE
                                                                            ;WHICH 32 POINTS NEED
                                                                            ;PROCESSING.
                                ;
000112  000000                                      PRCLOP: 0               ;THIS COUNTER IS USED TO
                                                                            ;DETERMINE HOW MANY POWER
                                                                            ;SPECTRUM SETS HAVE TO BE
                                                                            ;ADDED TOGETHER.
```

```
                                ;
                                ;
                                ;
                                ;
                                ;              PROCESSING CONTROL CONTINUED.......
                                ;
                                ;
                                ;
                                ;
                                ;
000114                          ;              NOTE:
                                ;              -----
                                ;                   THE LOOP COUNTERS ARE INITIALIZED BEFORE
                                ;                   PROCESSING IS INITIALIZED.
                                ;
000114  012767  000001  177766  ;              CONT1:  MOV     #1,ADCLOP
000122  012767  000010  177762  ;                      MOV     #10,PRCLOP
000130  000400                  ;                      BR      PRCESS
                                ;
                                ;
                                ;              NOTE: PROCESSING LOOP STARTS NOW.
                                ;              -----
                                ;
                                ;
                                ;
000132  032767  000001  177750  ;              PRCESS: BIT     #1,ADCLOP               ;IS IT FIRST DATA BUFFER?
000140  001411                  ;                      BEQ     PRC2
000142  012767  000714' 000344  ;                      MOV     #ISTR2,CDATA            ;YES,NOW USE 2DN BUFFER
000150  012767  000520' 000340  ;                      MOV     #ISTR1,DATA
000156  005067  177726          ;                      CLR     ADCLOP
000162  000410                  ;                      BR      PRC3
000164  012767  000520' 000322  ;              PRC2:   MOV     #ISTR1,CDATA
000172  012767  000714' 000316  ;                      MOV     #ISTR2,DATA
000200  005267  177704          ;                      INC     ADCLOP
000204  012767  000040  177570  ;              PRC3:   MOV     #40,ADCNT
000212  005067  177562          ;                      CLR     ADFLG
000216  106427  000100          ;                      MTPS    #100                    ;SET PROCESSOR PRIO
                                ;                                              -RITY TO 0
000222  012767  007520  170400' ;                      MOV     #7520,ADCSR
000230  004767  000000G         ;                      JSR     PC,FOURIE
                                ;
                                ;
                                ;              NOTE:
                                ;              -----
                                ;                   AT THIS POINT THE VALUE OF SFACTR WILL SHOW
                                ;                   THE NUMBER BITS THE FOURIER TRANSFORMED ARRAY
                                ;                   HAS BEEN SHIFTED 1-BIT TO THE RIGHT. TO HAVE A
                                ;                   FIXED VALUE OF SCALE FACTOR THE TRANSFORMED AR
                                ;                   -RAY IS RESCALED SO THAT SFACTR IS WQUAL TO
                                ;                   3. THIS VALUE IS CHOSEN AFTER EXPERIMENTATION AND
                                ;                   IT CAN BE CHANGED IF FUTURE DATA WARRANTS A
                                ;                   DIFFERENT VALUE FOR SCALE FACTOR (SFACTR).
                                ;
                                ;
                                ;
```

```
                                                        PROCESSING CONTROL CONTINUED.....

000234  016701  000000G                       RESCLE: MOV     SFACTR,R1
000240  162701  000003                               SUB     #3,R1                  ;STORE THE DIFFERENCE
                                                                                    BETWEEN SFACTR AND 3
                                                                                    IN R1.
000244  001410                                       BEQ     PR20
000246  012703  000040                       PR10:   MOV     #40,R3                 ;GET # OF POINTS IN R3.
000252  016700  000240                               MOV     DATA,R0
000256  011004                               PR15:   MOV     (R0),R4
000260  072401                                       ASH     R1,R4                  ;SHIFT R4 RIGHT OR
                                                                                    LEFT DEPENDING ON
                                                                                    THE VALUE OF R1.
000262  010420                                       MOV     R4,(R0)+
000264  077304                                       SOB     R3,PR15
000266  012767  000003  000000G               PR20:   MOV     #3,SFACTR

                                                        NOTE:
                                                        -----
                                                             AT THIS STAGE THE SFACTR IS ALWAYS GOING TO 3.

000274  004767  000000G                       PR21:   JSR     PC,PRSPEC

                                                        ****************************
                                                        *       IMPORTANT NOTE     *
                                                        ****************************

                                                        AFTER THE POWER SPECTRUM OF
                                                        THE FIRST 11 FREQUENCY COMPO
                                                        -NENTS HAS BEEN CALCULATED
                                                        THE ACQUISITION OF TEMPERATU
                                                        -RE SIGNAL IS CARRIED OUT.
                                                        THE ACQUIRED VALUE IS STORED
                                                        IN "TSTORE" AND LATER ADDED
                                                        TO VARIABLE "TSUM".THUS AN
                                                        AVERAGE TEMPT: VALUE WOULD
                                                        BE AVAILABLE BEFORE DF IS CA
                                                        -LCULATED. THE CODE GIVEN ON
                                                        NEXT PAGE HAS NOT YET BEEN
                                                        IMPLEMENTED BUT IT CAN BE AC
                                                        -TIVATED IN FUTURE.
```

```
. EXECUTIVE       RT-11 MACRO VM02-12    00:06:08 PAGE 7
PROCESSING CONTROL

                                ;
                                ;
                                ;
                                ;               PROCESSING CONTROL CONTINUED.....
                                ;
                                ;
                                ;
                                ;
                                ;
                                ;
                                ;               ****************************************
                                ;               *       MOV     #TSTORE,CDATA          *
                                ;               *       MOV     #1,ADCNT               *
                                ;               *       CLR     ADFLG                  *
                                ;               *       MOV     #7520,ADCSR            *
                                ;               *       WAIT                           *
                                ;               *       ADD     TSTORE,TSUM            *
                                ;               ****************************************
                                ;
                                ;
                                ;
                                ;
000300  005767  177606          ;               PR22:   TST     PRCLOP                 ;IS PRCLOP=0
000304  001042                  ;                       BNE     PR34                   ;IF NOT GO BACK
                                ;                                                      COMPUTE ANOTHER SET OF
                                ;                                                      POWER SPECTRUM.
                                ;
                                ;
                                ;               ****************************
                                ;               *       IMPORTANT NOTE     *
                                ;               ****************************
                                ;
                                ;               BEFORE DF IS CALCULATED THE
                                ;               THE TEMPERATURE INTERVAL IS
                                ;               DETERMINED AND "TFLG" IS LO
                                ;               -ADED WITH THE APPROPRIATE
                                ;               VALUE. THE CODE GIVEN ON
                                ;               NEXT PAGE IS INCLUDED FOR
                                ;               FUTURE ACTIVATION.
                                ;
```

```
                                        ;
                                        ;
                                        ;              PROCESSING CONTROL CONTINUED......
                                        ;
                                        ;
                                        ;              ****************************************
                                        ;              * INTL: MOV     #INTVL,R1             *
                                        ;              *       MOV     #13,R2                *
                                        ;              *       CLR     TFLG                  *
                                        ;              * IN3:  CMP     TSUM,(R1)+            *
                                        ;              *       BLE.    IN5                   *
                                        ;              *       INC     TFLG                  *
                                        ;              *       SOB     R2,IN3                *
                                        ;              * IN5:  BR      PR25                  *
                                        ;              *                                     *
                                        ;              *                                     *
                                        ;              * INTVL: 0000,1111,2222,3333           *
                                        ;              *        4444,5555,6666,7777          *
                                        ;              *        8888,9999,10000              *
                                        ;              *                                     *
                                        ;              *       (THE ABOVE 11 VALUES ARE      *
                                        ;              *       FICTITIOUS. THEY HAVE TO      *
                                        ;              *       BE FILLED UP WITH THE OC      *
                                        ;              *       -TAL EQUIVALENT OF THE        *
                                        ;              *       BOUNDARY VALUE FOR EACH       *
                                        ;              *       INTERVAL).                    *
                                        ;              ****************************************
                                        ;
                                        ;
                                        ;
                                        ;              NOTE:
                                        ;              -----
                                        ;                   THE VALUE STORED IN "CUMSF1" TELLS THE AMOUNT
                                        ;                   BY WHICH THE ARRAY "SPCTRA" HAS BEEN SCALED.
                                        ;                   EXPERIMENTLY IT HAS BEEN DETERMINED THAT IF
                                        ;                   "CUMSF1" IS FIXED EQUAL TO 4,NO SIGNIFICANT
                                        ;                   LOSS IN ACCURACY OCCURS.THE CODE THAT FOLLOWS
                                        ;                   ACCOMPLISHES THIS.
                                        ;
                                        ;
                                        ;
000306  016701  000000G                 ;              RSCL2: MOV     CUMSF1,R1
000312  162701  000004                  ;                     SUB     #4,R1
000316  001410                          ;                     BEQ     PR23
000320  012703  000013                  ;              PR24:  MOV     #13,R3
000324  016700  000000G                 ;                     MOV     SPCTRA,R0
000330  011004                          ;              RSCLP: MOV     (R0),R4
000332  072401                          ;                     ASH     R1,R4
000334  010420                          ;                     MOV     R4,(R0)+
000336  077304                          ;                     SOB     R3,RSCLP
000340  012767  000004  000000G         ;              PR23:  MOV     #4,CUMSF1
```

```
EXECUTIVE       RT-11 MACRO VM02-12    00:06:08 PAGE 9
PROCESSING CONTROL


                                                ;
                                                ;
                                                ;
                                                ;              PROCESSING CONTRON CONTINUED.......
                                                ;
                                                ;
                                                ;
                                                ;
 000346  004767  000000G                        ;              PR25:   JSR     PC,POLYN                ;COMPUTE DF
                                                ;
                                                ;              NOTE: ONCE THE DF IS COMPUTED THE WHOLE LOOP
                                                ;              ----- IS REPEATED AGAIN. THE FOLLOWING CLEARS
                                                ;                    THE REQUISITE VARIABLES AND ARRAYS.
                                                ;
                                                ;
 000352  005067  000534                         ;              PCLR:   CLR     TSUM
 000356  005067  000526                         ;                      CLR     TSTORE
 000362  005067  000000G                        ;                      CLR     CUMSF1
 000366  012767  000012  177516                 ;                      MOV     #12,PRCLOP
 000374  016700  000000G                        ;              CPSA:   MOV     SPCTRA,R0               ;THE POWER SPECTRUM
                                                ;                                                      IS CLEARED NOW.
 000400  012701  000013                         ;                      MOV     #13,R1
 000404  005020                                 ;              PR30:   CLR     (R0)+
 000406  077102                                 ;                      SOB     R1,PR30
 000410  000402                                 ;                      BR      PR35
                                                ;
 000412  005367  177474                         ;              PR34:   DEC     PRCLOP
 000416  005067  000000G                        ;              PR35:   CLR     SFACTR
 000422  000167  177504                         ;                      JMP     PRCESS
                                                ;
                                                ;
                                                ;
                                                ;
                                                ;              END OF PRCESSING CONTROL
```

```
                                                .SBTTL INTERRUPT SERVICE ROUTINES.
                                ;
                                ;
                                ;
                                ;
                                ;               5.INTERRUPT SERVICE ROUTINES:
                                ;               -----------------------------
                                ;                 THE SERVICE ROUTINES TO HANDLE INTERRUPT GENERATED
                                ;               BY THE ADC AND RTC ARE GIVEN BELOW:
                                ;                       ADC SERVICE ROUTINE:
                                ;                       --------------------
000426  016777  170402' 000060  ;               ADCSRV: MOV     ADBFR,@CDATA
000434  162777  004000  000052  ;                       SUB     #4000,@CDATA
000442  005267  000046          ;                       INC     CDATA
000446  005267  000042          ;                       INC     CDATA
000452  005367  177324          ;                       DEC     ADCNT
000456  001007                  ;                       BNE     ADRTI
000460  005267  177314          ;                       INC     ADFLG
000464  042767  000300  170400' ;                       BIC     #300,ADCSR
000472  106427  000200          ;                       MTPS    #200
000476  000002                  ;               ADRTI:  RTI
                                ;
                                ;
                                ;               RTC SERVICE ROUTINE:
                                ;               --------------------
000500  106427  000200          ;               RTCSRV: MTPS    #200
000504  012767  000100  177766  ;                       MOV     #100,RTCSRV
000512  000002                  ;                       RTI
                                ;
```

```
                                                .SBTTL DATA STRUCTURE AND VARIABLES USED      .
                                ;
                                ;
                                ;
                                ;
                                ;               6. DATA STRUCTURE AND VARIABLES USED:
                                ;               -------------------------------------
                                ;                 THE DATA STRUCTURE WHICH IS USED TO STORE ACQUIRED
                                ;               DATA AND DIFFERENT VARIABLES USED ARE GIVEN BELOW:
                                ;
                                                      .
000514  000000                                  CDATA:  .WORD   0
000516  000000                                  DATA:   .WORD   0
000520  000076                                  ISTR1:  .REPT   76
                                                        .WORD   0
                                                        .ENDM
                                ;
000714  000076                                  ISTR2:  .REPT   76
                                                        .WORD   0
                                                        .ENDM
                                ;
001110  000000                                  TSTORE: .WORD   0
001112  000000                                  TSUM:   .WORD   0
                                ;
                                ;
                                ;
        000001'                                 .END
```

APPENDIX I

LSI-11 ASSEMBLY LANGUAGE CODE FOR DECIMAL AND OCTAL PRINTOUT (DECPR)

```
                                        .MCALL  .REGDEF
000000                                  .REGDEF
                                        .ENABL  AMA
                                        .GLOBL  DECPR,TYPE
                                ;*****************************************************************************
                                ;*  DECIMAL VECTOR PRINT SUBROUTINE CALLED BY JSR PC,DECPR.                  *
                                ;*  THIS ROUTINE USES THE FORTRAN STANDARD CALLING SEQUENCE WITH THREE       *
                                ;*  ARGUMENTS:                                                               *
                                ;*               VECTOR -- THE VECTOR TO BE PRINTED;                         *
                                ;*               ADDR   -- THE ADDRESS OF THE FIRST ELEMENT OF THE           *
                                ;*                         VECTOR TO BE PRINTED;                             *
                                ;*               LEN    -- THE NUMBER OF ELEMENTS IN THE VECTOR.             *
                                ;*                                                                           *
                                ;*  FOR EACH ELEMENT OF THE VECTOR, ONE LINE IS PRINTED CONTAINING THE       *
                                ;*  ADDRESS OF THE ELEMENT IN OCTAL FOLLOWED BY THE VALUE OF THE             *
                                ;*  ELEMENT IN DECIMAL.                                                      *
                                ;*****************************************************************************
                                ;
                                ;
                                ;
000000  016537  000002  000142' DECPR: MOV     2(R5),3$        ;3$ <- ACTUAL ADDRESS OF FIRST ELEMENT
000006  017537  000004  000144'       MOV     @4(R5),4$       ;4$ <- ADDRESS TO PRINT FOR 1ST ELEMENT
000014  017537  000006  000146'       MOV     @6(R5),5$       ;5$ <- NUMBER OF ELEMENTS
000022  012705  000140'               MOV     #2$,R5          ;R5 <- ADDR OF ARGUMENT LIST FOR INTPRT
000026  013701  000142'       1$:     MOV     3$,R1           ;FOR EACH ELEMENT OF THE VECTOR, DO
000032  004737  000150'               JSR     PC,OCTPRT       ;   PRINT THE ADDRESS OF THE ELEMENT
000036  112700  000040                MOVB    #40,R0          ;   PRINT A SPACE
000042  004737  000000G               JSR     PC,TYPE
000046  004737  000216'               JSR     PC,INTPRT       ;   PRINT THE VALUE OF THE ELEMENT
000052  112700  000012                MOVB    #12,R0          ;   PRINT A CARRIAGE RETURN
000056  004737  000000G               JSR     PC,TYPE
000062  112700  000015                MOVB    #15,R0          ;   PRINT A LINE FEED
000066  004737  000000G               JSR     PC,TYPE
000072  062737  000002  000142'       ADD     #2,3$           ;   3$ <- ADDRESS OF NEXT ARRAY ELEMENT
000100  005337  000146'               DEC     5$              ;   5$ <- NUMBER OF ELEMENTS REMAINING
000104  001350                        BNE     1$              ;   GO BACK FOR NEXT ELEMENT.
                                ;
000106  012700  000012                MOV     #12,R0          ;PRINT A LINE FEED
000112  004737  000000G               JSR     PC,TYPE
000116  012700  000012                MOV     #12,R0          ;PRINT ANOTHER LINE FEED
000122  004737  000000G               JSR     PC,TYPE
000126  012700  000015                MOV     #15,R0          ;PRINT A CARRIAGE RETURN
000132  004737  000000G               JSR     PC,TYPE         ;RETURN
000136  000207                        RTS     PC
000140  000003              2$:       .WORD   3
000142  000000              3$:       .WORD   0
000144  000000              4$:       .WORD   0
000146  000000              5$:       .WORD   0
```

```
                                        .SBTTL OCTPRT   -- OCTAL PRINT ROUTINE
                                ;
                                ;
                                ;
                                ;
                                ;*************************************************************************
                                ;*  OCTAL PRINT SUBROUTINE CALLED BY JSR PC,OCTPRT.  PRINTS CONTENTS     *
                                ;*  OF R1 AS A SIX DIGIT OCTAL NUMBER.                                   *
                                ;*************************************************************************
                                ;
                                ;
                                ;
                                ;
000150  012702  000006          OCTPRT: MOV     #6,R2           ;R2 <- NUMBER OF DIGITS TO BE PRINTED
000154  012737  000001  000212' 2$:     MOV     #1,2$           ;2$ <- NUMBER OF BITS IN FIRST DIGIT
                                        ;FOR EACH DIGIT (MSD FIRST) DO
000162  005000                  1$:     CLR     R0
000164  073037  000212'                 ASHC    2$,R0           ;   SHIFT 2$ BITS INTO R0 FROM R1
000170  012737  000003  000212'         MOV     #3,2$           ;  (2$ IS 1 FOR 1ST DIGIT, 3 THEREAFTER)
000176  052700  000060                  BIS     #60,R0          ;   CONVERT TO AN ASCII DIGIT
000202  004737  000000G                 JSR     PC,TYPE         ;   PRINT THE DIGIT
000206  077213                          SOB     R2,1$           ;   GO BACK FOR NEXT DIGIT
000210  000207                          RTS     PC              ;RETURN
000212  000001                  2$:     .WORD   1
```

```
                                        .SBTTL INTPRT   -- INTEGER PRINT ROUTINE
                                ;
                                ;
                                ;*********************************************************************
                                ;*  PRINT DECIMAL INTEGER ROUTINE CALLED BY JSR PC,INTPRT.  USES     *
                                ;*  STANDARD CALLING SEQUENCE WITH ONE PARAMETER WHICH CONTAINS THE   *
                                ;*  BINARY INTEGER TO BE CONVERTED AND PRINTED.  THE OUTPUT FORMAT IS *
                                ;*  EQUIVALENT TO FORTRAN (1X,I7).                                    *
                                ;*********************************************************************
                                ;
                                ;
000216  017503  000002          INTPRT: MOV     @2(R5),R3
000222  012701  000005                  MOV     #5,R1           ;R1 <- NUMBER OF DIGITS TO CONVERT
000226  112737  000040  000366'         MOVB    #40,6$          ;SIGN <- BLANK FOR POSITIVE
000234  005703                          TST     R3              ;IF NUMBER IS NEGATIVE, THEN
000236  002004                          BGE     1$
000240  005403                          NEG     R3              ;   MAKE IT POSITIVE
000242  112737  000055  000366'         MOVB    #'-,6$          ;   REMEMBER TO PRINT A MINUS SIGN
000250  012737  020040  000356' 1$:     MOV     #20040,5$       ;BLANK THE BUFFER
000256  012737  020040  000360'         MOV     #20040,5$+2
000264  012737  020040  000362'         MOV     #20040,5$+4
000272  012737  020060  000364'         MOV     #20060,5$+6
                                ;
                                ;
                                ;*********************************************************************
                                ;*  THE FOLLOWING LOOP CONVERTS THE NUMBER TO A CHARACTER STRING ONE  *
                                ;*  DIGIT AT A TIME STARTING WITH THE UNITS DIGIT, THEN THE 10'S DIGIT,*
                                ;*  100'S DIGIT, ETC.  THESE DIGITS ARE CONVERTED TO ASCII AND STORED  *
                                ;*  FROM RIGHT TO LEFT IN A BUFFER.  THE BUFFER IS LATER PRINTED.     *
                                ;*********************************************************************
                                ;
                                ;
000300  005002                  2$:     CLR     R2              ; FOR EACH DIGIT FROM 10**0 TO 10**5, DO
000302  071227  000012                  DIV     #10.,R2         ;   R2 <- NUMBER/10.
                                                                ;   R3 <- LEAST SIGNIFICANT DECIMAL
                                                                ;         DIGIT REMAINING
000306  052703  000060                  BIS     #60,R3          ;   CONVERT THE DIGIT TO ASCII
000312  110361  000360'                 MOVB    R3,5$+2(R1)     ;   PUT IT IN THE BUFFER
000316  010203                          MOV     R2,R3           ;   IF ALL HIGHER ORDER DIGITS = 0,
000320  001402                          BEQ     3$              ;      THEN EXIT LOOP
000322  077112                          SOB     R1,2$           ;   GO BACK FOR NEXT DIGIT.
000324  005201                          INC     R1              ;R1 <- 1 (SOB DECREMENTS ONCE TOO OFTEN)
000326  113761  000366'  000357' 3$:     MOVB    6$,5$+1(R1)     ;BUFFER(R1) <- SIGN CHARACTER
000334  012702  000356'                 MOV     #5$,R2          ;R2 <- ADDRESS OF BUFFER
000340  012701  000010                  MOV     #10,R1          ;R1 <- NUMBER OF CHARACTERS TO PRINT
000344  112200                  4$:     MOVB    (R2)+,R0        ;FOR I = 1 TO NUMBER OF CHARACTERS,
000346  004737  0000006                 JSR     PC,TYPE         ;   PRINT BUFFER(I)
000352  077104                          SOB     R1,4$
000354  000207                          RTS     PC              ;RETURN
000356  000000  000000  000000  5$:     .WORD   0,0,0,0         ;*** BUFFER ***
000364  000000
000366  000000                  6$:     .WORD   0               ;*** SIGN CHARACTER ***
        000001'                         .END
```

APPENDIX J

SIMULATION PROGRAM FOR TESTING COMPUTATIONAL ACCURACY OF SOFTWARE MODULES AND TEST RESULTS

APPENDIX J

Simulation Program for Testing Computational Accuracy of Software Modules and Test Results

Source Code for Test Program (TTFT)

```
C     ***************************************************
C     * THIS PROGRAM IS USED TO TEST THE COMPUTATI-      *
C     * ONAL ACCURACY OF DIFFERENT MODULES OF THE        *
C     * MICROPROCESSOR BASED DATA ACQUISITION AND        *
C     * PROCESSING SYSTEM (MIPDAPS). THE PROGRAM         *
C     * ACCEPTS 32 DECIMAL DATA VALUES AND PERFOR-       *
C     * MS THE SAME PROCESSING OF DATA AS MIPDAPS.       *
C     * SINNCE THIS PROGRAM WAS EXECUTED ON A CDC-       *
C     * 6600,WHICH IS A 60-BIT MACHINE THEREFORE         *
C     * THE COMPUTATIONAL RESULTS WOULD BE MORE          *
C     * ACCURATE AS COMPARED WITH THE RESULTS OBT-       *
C     * AINED ON PDP-11/03, WHICH IS A 16-BIT MA-        *
C     * CHINE.*
C     *    THE RESULTS PRINTED OUT BY THIS PROGRA-       *
C     * M ARE IN DECIMAL AS WELL AS IN OCTAL FORM-       *
C     * AT. THIS HAS BEEN DONE FOR EASY COMPARISON       *
C     * WITH THE RESULT FROM ASSEMBLY LANGUAGE           *
C     * SOFTWARE WRITTEN FOR MIPDAPS.                    *
C     ***************************************************
C
C
C
      PROGRAM TFFT(INPUT,OUTPUT,TAPE5=OUTPUT,TAPE6=INPUT)
      DIMENSION IPDATA(32),PRS(12)
      DIMENSION F(32),PF(32),CCF(32)
      DIMENSION CONF(32)
      DIMENSION ERR1(32),ERR2(10),ERR3(10)
      DIMENSION LDATA(32),CF(12)
      DIMENSION X(32)
      COMPLEX RDATA(32)
      WRITE(5,150)
150   FORMAT(1H1)
      DO 444 I=1,29,4
      PRINT*," "
      PRINT*,"  INPUT FOUR DATA VALUES....."
      PRINT*," "
      READ*, X(I),X(I+1),X(I+2),X(I+3)
      X(I)=X(I)*.99999 $ X(I+1)=X(I+1)*.999999 $ X(I+2)=X(I+2)*.999999
      X(I+3)=X(I+3)*.999999
444   CONTINUE
      M=32
      WRITE(5,222)
      PRINT*," "
      WRITE(5,232)
222   FORMAT(10X,30HTABLE BELOW SHOWS 32 REAL DATA)
232   FORMAT(10X,32HPOINTS WHICH ARE FED IN AS INPUT)
      PRINT*," "
      PRINT*," "
      WRITE(5,121)
      WRITE(5,131)
121   FORMAT(3X,11HDATA POINTS,19X,11HDATA POINTS)
131   FORMAT(3X,10HIN DECIMAL,20X,8HIN OCTAL)
```

```
      WRITE (5,242)
242   FORMAT(9X,3HODD,9X,4HEVEN,9X,3HODD,5X,4HEVEN)
      CALL OPRINT(X,M)
      DO 10 I=1,32
10    RDATA(I)=CMPLX(X(I),0.0)
      PRINT*," "
      PRINT*," "
      WRITE(5,252)
      WRITE(5,262)
252   FORMAT(9X,34HTABLE BELOW SHOWS THE COMPLEX DATA)
262   FORMAT(8X,36HARRAY WHICH IS FED TO FFT SUBROUTINE)
      PRINT*," "
      WRITE(5,121) ! WRITE(5,131)
      WRITE(5,272)
272   FORMAT(7X,4HREAL,9X,4HIMAG,9X,4HREAL,5X,4HIMAG)
      CALL OPRINT(RDATA,64)
      N=5  $  XI=1.
      CALL FFT(RDATA,N,XI)
      DO 111 J=1,32
      READ*,F(J)
111   CONTINUE
      DO 276 I=1,16
      CONF(2*I-1)=REAL(RDATA(I))
      CONF(2*I)=AIMAG(RDATA(I))
275   CONTINUE
      DO 277 J=1,32
      ERR1(J)=(CONF(J)-F(J))**2
277   CONTINUE
      PRINT*," "
      PRINT*," "
      CALL OPRINT1(RDATA,32)
      PRINT*," "
      CALL OPRINT1(F,32)
      PRINT*," "
      PRINT*," "
      CALL OPRINT1(ERR1,32)
      CALL PRSPEC(RDATA,PRS,32,ERR2)
      CALL POLY(PRS,CF,11,ERR3)
      DO 777 I=1,32
777   SUM1=SUM1+ERR1(I)
      CONTINUE
      PRINT*," "
      PRINT*," "
      PRINT*," RMS ERROR FOR FFT=" ,(SUM1/32.)**.5
      DO 888 I=1,10
      SUM2=SUM2+ERR2(I)
      SUM3=SUM3+ERR3(I)
888   CONTINUE
      PRINT*," "
PRINT*," "
      PRINT*,"RMS ERROR FOR PWRSPEC= ",(SUM2/10.)**.5
      PRINT*," "
      PRINT*," "
      PRINT*," RMS ERROR FOR CDF= ",(SUM3/10.)**.5
      STOP
      END
```

Source Code for Power Spectrum Subroutine

```
C
      SUBROUTINE PRSPEC(RDATA,PRS,LEN,ERR2)
      DIMENSION PRS(11)
      DIMENSION PF(10),ERR2(10)
      COMPLEX RDATA(LEN)
      DO 10 K=1,11
      PRS(K)=(REAL(RDATA(K)))**2.+(AIMAG(RDATA(K)))**2.
10    CONTINUE
      DO 333 J=1,10
333   READ*, PF(J)
      CONTINUE
      DO 444 J=1,10
444    ERR2(J)=(PRS(J)-PF(J))**2
      CONTINUE
      PRINT*," "
      PRINT*," "
      CALL OPRINT1(PRS,10)
      PRINT*," "
      PRINT*," "
      CALL OPRINT1(PF,10)
      PRINT*," "
      PRINT*," "
      CALL OPRINT1(ERR2,10)
      RETURN $ END
      SUBROUTINE OPRINT1(X,NP)
      DIMENSION X(NP)
      PRINT*," "
      PRINT*," "
      PRINT*," "
      DO 10 I=1,NP
      I1=X(I)*2.**15
      WRITE(5,100) X(I),I1
10    CONTINUE
100   FORMAT(4X,G13.4,4X,O6)
      RETURN
      END
```

Source Code for CDF Subroutine

```
      SUBROUTINE POLY(PRS,CF,NP,ERR3)
      DIMENSION PRS(NP),CF(NP),CCF(10)
      DIMENSION ERR3(10)
      DO 10 K=1,NP
      CF(K)=PRS(K)**1.6
10    CONTINUE
      DO 555 J=1,10
555   READ*, CCF(J)
      CONTINUE
      DO 666 J=1,10
665   ERR3(J)=(CF(J)-CCF(J))**2
      CONTINUE
      PRINT*," "
      PRINT*," "
      CALL OPRINT1(CF,10)
      PRINT*," "
      PRINT*," "
      CALL OPRINT1(CCF,10)
      PRINT*," "
      PRINT*," "
      CALL OPRINT1(ERR3,10)
      RETURN $ END
```

Source Code for FFT Subroutine

```
      SUBROUTINE FFT(X,M,XI) $ COMPLEX X(1),U,W,T $ N=2**M $ NV2=N/2
      NM1=N-1 $ J=1 $ PI=3.1415926535898 $ DO 7 I=1,NM1
      IF(I.GE.J) GO TO 5 $ T=X(J) $ X(J)=X(I) $ X(I)=T
5     K = NV2
5     IF(K.GE.J) GO TO 7 $ J=J-K $ K=K/2 $ GO TO 6
7     J=J+K $ DO 20 L=1,M $ LE=2**L $ LE1=LE/2 $ U=(1.,0.)
      W=CEXP(CMPLX(0.,-XI*PI/LE1)) $ DO 20 J=1,LE1 $ DO 10 I=J,N,LE
      IP=I+LE1 $ T=X(IP)*U $ X(IP)=X(I)-T
10    X(I) = X(I) + T
20    U=U*W $ IF(XI.GT.0.) RETURN $ DO 30 I=1,N
30    X(I)=X(I)/N $ RETURN $ END
C
C
```

Source Code for Printing Out Test Results

```
      SUBROUTINE OPRINT(X,NP)
      DIMENSION X(NP)
      PRINT*," "
      PRINT*," "
      DO 10 I=1,NP,2
      I1=X(I)*2.**15
      I2=X(I+1)*2.**15
      WRITE (5,100) X(I),X(I+1),I1,I2
10    CONTINUE
100   FORMAT(3X,2G13.4,4X,2(3X,O6))
      RETURN $ END
C
C
```

```
      SUBROUTINE OPRINT1(X,NP)
      DIMENSION X(NP)
      PRINT*," "
      PRINT*," "
      PRINT*," "
      DO 10 I=1,NP
      I1=X(I)*2.**15
      WRITE(5,100) X(I),I1
10    CONTINUE
100   FORMAT(4X,G13.4,4X,O6)
      RETURN
      END
```

TABLE J-I

Test Results for FFT Computation (Data Set #1)

	INPUT DATA		OUTPUT FROM SUBROUTINE FOURIE [F(n)]*		DESCALED ARRAY X(n) = SFACTR x F(n)		FFT OUTPUT FROM TEST PROGRAM [Y(n)]		ERROR SQUARE $[Y(n) - X(n)]^2$	
Mem Loc	Value In Decimal	Value In Octal	Value In Decimal	Value In Octal	Value In Decimal	Value In Octal	Value In Decimal	Value In Octal	Value In Decimal	Value In Octal
4172	0.1153E+00	7301	0.3959E+00	31254	.7918	062531	.7922	062546	.1635E-06	000000
4174	0.5356E-01	3333	0.0000E+00	0	0.	000000	0.	000000	0.	000000
4176	0.1094E+00	7001	0.1454E+00	11233	.2908	022470	.2908	022471	.5092E-10	000000
4200	0.9134E-01	5661	0.6821E-01	4273	.1364	010565	.1359	010545	.2507E-05	000000
4202	0.7809E-01	4777	0.2376E+00	17151	.4752	036323	.4756	036337	.1446E-06	000000
4204	-0.2536E-01	176301	-0.1590E+00	165647	-.3180	753513	-.3181	753510	.1291E-07	000000
4206	0.8575E-01	5372	-0.3909E-01	175377	-.7810E-01	772000	-.7835E-01	772770	.6466E-07	000000
4210	-0.7751E-02	177402	-0.5646E-01	174306	-.1129	770614	-.1123	770520	.1639E-07	000000
4212	-0.1240E+00	170042	-0.1348E+00	167300	-.2695	756575	-.2698	756566	.4977E-07	000000
4214	0.1001E+00	6320	0.3842E-01	2353	.7680E-01	004724	.7690E-01	004727	.9710E-08	000000
4216	-0.8002E-01	172702	-0.3128E-01	175777	-.6250E-01	773777	-.5241E-01	774002	.7225E-08	000000
4220	-0.9378E-01	171777	-0.1467E+00	166470	-.2934	755161	-.2938	755145	.1364E-06	000000
4222	0.9982E-01	6307	-0.1513E+00	166241	-.3025	754604	-.3024	754513	.4076E-07	000000
4224	-0.1094E+00	170777	0.1396E+00	10740	.2792	021674	.2792	021674	.1066E-09	000000
4226	0.8841E-01	5521	-0.2086E+00	162514	-.4172	746231	-.4171	746234	.7810E-08	000000
4230	0.1071E+00	6666	-0.3230E+00	153251	-.6460	725517	-.6460	725516	.8946E-09	000000
4232	-0.3107E-01	176006	0.1386E+00	10676	.2772	021573	.2770	021565	.3032E-07	000000
4234	0.5466E-01	3377	-0.2229E+00	161571	-.4458	743360	-.4459	743356	.3667E-08	000000
4236	-0.7178E-01	173320	0.1016E+00	6402	.2032	015002	.2034	015011	.4880E-07	000000
4240	0.7224E-01	4477	0.4092E-01	2475	.8180E-01	005170	.8156E-01	005160	.5719E-07	000000
4242	0.1190E+00	7474	0.9055E-01	5627	.1811	013456	.1815	013472	.1352E-06	000000
4244	0.7571E-01	4661	0.1775E+00	13270	.3550	026560	.3550	026560	.6432E-11	000000
4246	-0.2927E-01	176101	0.2647E+00	20741	.5294	041703	.5294	041704	.2247E-09	000000
4250	-0.9546E-01	171710	-0.1681E+00	165174	-.3362	752367	-.3362	752367	.5230E-10	000000
4252	0.6244E-01	3776	-0.2488E+00	160046	-.4975	740116	-.4973	740106	.5212E-07	000000
4254	0.8575E-01	5372	-0.2912E+00	155273	-.5824	732563	-.5825	732561	.6528E-08	000000
4256	-0.4657E-01	175012	0.1426E+00	11101	.2852	022201	.2854	022210	.4339E-07	000000
4260	-0.7928E-01	172732	0.2236E+00	16236	.4472	034475	.4469	034465	.5930E-07	000000
4262	0.1094E+00	7000	0.1145E+00	7247	.2290	016517	.2293	016531	.9259E-07	000000
4264	0.9375E-01	6000	-0.1453E+00	166547	-.2906	755315	-.2908	755306	.5073E-07	000000
4266	0.9583E-01	6104	0.2098E+00	15333	.4196	032665	.4197	032672	.2073E-07	000000
4270	-0.1115E+00	170671	-0.1205E-01	177165	-.2410E-01	776352	-.2427E-01	776344	.2834E-07	000000

*Scale factor SFACTOR equals 2.0.

The RMS error for FOURIE equals .00022

TABLE J-II
Test Result for Power Spectrum and CDF Computation (Data Set #1)

OUTPUT FROM SUBROUTINE PRSPEC [P(n)]*			DESCALED ARRAY R(n) = CUMSF1 x P(n)		POWER SPECTRUM RESULT FROM TEST PROGRAM [S(n)]		ERROR SQUARE $[S(n) - R(n)]^2$	
Mem Loc	Value In Decimal	Value In Octal	Value In Decimal	Value In Octal	Value In Decimal	Value In Octal	Value In Decimal	Value In Octal
23030	0.3915E-01	2403	.5264	050065	.6276	050124	.1411E-05	000000
23032	0.6439E-02	323	.1030	006457	.1030	006460	.1399E-08	000000
23034	0.2042E-01	1235	.3267	024721	.3274	024747	.4528E-06	000000
23036	0.1160E-02	46	.1850E-01	001136	.1886E-01	001151	.1274E-06	000000
23040	0.4883E-02	240	.7810E-01	004777	.7872E-01	005023	.3818E-06	000000
23042	0.5615E-02	270	.8980E-01	005576	.9020E-01	005613	.1558E-06	000000
23044	0.1056E-01	532	.1689	012636	.1694	012656	.2416E-06	000000
23046	0.3693E-01	2272	.5908	045637	.5913	045660	.2881E-06	000000
23050	0.1721E-01	1064	.2753	021475	.2755	021504	.5521E-07	000000
23052	0.2960E-02	141	.4730E-01	003015	.4803E-01	003045	.5362E-05	000000

*Scale factor CUMSF1 equals 16.

The RMS error for PRSPEC equals .0006

OUTPUT FROM SUBROUTINE POLYN [U(n)]*			DESCALED ARRAY V(n) = U(n) x CUMSF2		OF OUTPUT FROM TEST PROGRAM [V(n)]		ERROR SQUARE $[V(n) - V(n)]^2$	
Mem Loc	Value In Decimal	Value In Octal	Value In Decimal	Value In Octal	Value In Decimal	Value In Octal	Value In Decimal	Value In Octal
23310	0.2747E-02	132	.4639	035541	.4745	036275	.1134E-03	000003
23312	0.6104E-04	2	.1030E-01	000521	.2635E-01	001637	.2576E-03	000010
23314	0.9155E-03	36	.1546	011711	.1675	012661	.1669E-03	000005
23316	-0.1221E-03	177774	-.2060E-01	776534	.1741E-02	000071	.4991E-03	000020
23320	0.0000E+00	0	0.	000000	.1713E-01	001061	.2934E-03	000011
23322	0.6104E-04	2	.1030E-01	000521	.2130E-01	001271	.1209E-03	000003
23324	0.3052E-03	12	.5150E-01	003227	.5637E-01	003570	.4727E-04	000001
23326	0.2441E-02	120	.4122	032302	.4315	033471	.3708E-03	000014
23330	0.6714E-03	26	.1133	007200	.1271	010106	.1915E-03	000006
23332	-0.6104E-04	177776	-.1030E-01	777256	.7771E-02	000376	.3265E-03	000012

*Scale factor CUMSF2 equals 168.89.

The RMS error for POLYN equals .0154

TABLE J-III

Test Results for FFT Computation (Data Set #2)

Mem Loc	INPUT DATA Value In Decimal	INPUT DATA Value In Octal	OUTPUT FROM SUBROUTINE FOURIE [F(n)]* Value In Decimal	OUTPUT FROM SUBROUTINE FOURIE [F(n)]* Value In Octal	DESCALED ARRAY X(n) = SFACTR x F(n) Value In Decimal	DESCALED ARRAY X(n) = SFACTR x F(n) Value In Octal	FFT OUTPUT FROM TEST PROGRAM [Y(n)] Value In Decimal	FFT OUTPUT FROM TEST PROGRAM [Y(n)] Value In Octal	ERROR SQUARE $[Y(n) - X(n)]^2$ Value In Decimal	ERROR SQUARE $[Y(n) - X(n)]^2$ Value In Octal
4172	0.1093E+00	6777	0.2899E-01	1666	.1159	007329	.1348	010500	.3570E-03	000013
4174	-0.1098E+00	170761	0.0000E+00	0	0.	000000	0.	000000	0.	000000
4176	-0.7639E-01	173071	0.2252E-01	1342	.9000E-01	005605	.8650E-01	005422	.1228E-04	000000
4200	0.1111E+00	7072	-0.1263E-01	177142	-.5050E-01	774611	-.3321E-01	775677	.2959E-03	000011
4202	0.1084E+00	6737	0.3854E-01	2357	.1541	011671	.1375	010632	.2743E-03	000010
4204	-0.9833E-01	171552	-0.6100E-01	174061	-.2240	761523	-.2516	757714	.7607E-03	000030
4206	-0.4468E-01	175110	-0.1013E+00	171411	-.4052	746042	-.3952	746550	.9927E-04	000007
4210	0.1048E+00	6552	0.1048E+00	6552	.4192	032650	.4038	031657	.2369E-03	000007
4212	0.9369E-01	5776	0.1535E-01	767	.6140E-01	003733	.7433E-01	004603	.1671E-03	000006
4214	0.1099E+00	7021	0.1743E-01	1073	.6970E-01	004353	.8197E-01	005175	.1505E-03	000004
4216	-0.1094E+00	170777	0.5524E-01	3422	.2209	016106	.2062	015145	.2155E-03	000007
4220	-0.1231E+00	170077	-0.2560E+00	157472	-1.024	676755	-1.015	677040	.9909E-04	000002
4222	-0.1233E+00	170070	-0.9064E-02	177327	-.3620E-01	775535	-.4314E-01	775172	.4817E-04	000001
4224	0.1559E-01	777	0.2274E+00	16433	.9096	072155	.8928	071110	.2807E-03	000011
4226	0.1529E-01	765	0.5081E-01	3201	.2032	015002	.2209	016105	.3121E-03	000012
4230	0.9375E-01	6000	0.8942E-01	5562	.3576	026705	.3539	026513	.1399E-04	000000
4232	0.9821E-01	6222	0.2652E+00	20762	1.060	103656	1.061	103713	.3000E-06	000000
4234	-0.7648E-01	173066	0.1556E+00	11752	.6224	047652	.6404	050767	.3225E-03	000012
4236	-0.1182E+00	170337	0.3433E-01	2145	.1373	010623	.1196	007517	.3123E-03	000012
4240	0.1086E+00	6745	0.1398E+00	10744	.5592	043623	.5552	043417	.1625E-04	000000
4242	0.1225E+00	7655	0.7077E-01	4417	.2930	022071	.2901	022442	.5050E-04	000001
4244	0.7806E-01	4776	0.5545E-01	3431	.2218	016143	.2049	015072	.2857E-03	000011
4246	-0.1096E+00	170770	-0.4193E-01	175242	-.1677	765210	-.1528	766160	.2218E-03	000007
4250	-0.1206E+00	170220	-0.3931E-01	175370	-.1572	765740	-.1475	766436	.9401E-04	000003
4252	-0.1562E-01	177000	0.1993E-01	1215	.7970E-01	005063	.6683E-01	004215	.1555E-03	000005
4254	-0.3125E-01	176000	-0.2713E-01	176207	-.1085	771034	-.9511E-01	771667	.1535E-03	000005
4256	-0.2069E-01	176532	-0.5417E-01	174421	-.2166	762106	-.2264	761403	.9693E-04	000003
4260	0.1133E+00	7200	0.3912E-01	2402	.1564	017004	.1413	011025	.2292E-03	000007
4262	0.1074E+00	6700	0.2908E-01	1671	.1163	007342	.1332	010416	.2872E-03	000011
4264	-0.1106E+00	170731	0.1370E-01	701	.5480E-01	003403	.4767E-01	003032	.5080E-04	000001
4266	-0.9665E-01	171641	0.5658E-01	3476	.2263	016367	.2300	016560	.1369E-04	000000
4270	0.1116E+00	7112	-0.2606E-01	176252	-.1042	771251	-.8707E-01	772332	.2933E-03	000011

*Scale factor SFACTOR equals 4.0.

The RMS error for FOURIE equals .0027.

TABLE J-IV

Test Results for Power Spectrum and CDF Computation (Data Set #2)

OUTPUT FROM SUBROUTINE PRSPEC [P(n)]*			DESCALED ARRAY R(n) = CUMSF1 x P(n)		POWER SPECTRUM RESULT FROM TEST PROGRAM [S(n)]		ERROR SQUARE $[S(n) - R(n)]^2$	
Mem Loc	Value In Decimal	Value In Octal	Value In Decimal	Value In Octal	Value In Decimal	Value In Octal	Value In Decimal	Value In Octal
23030	0.1831E-03	6	.1170E-01	000577	.1817E-01	001123	.4185E-04	000001
23032	0.1576E-03	5	.9000E-03	000035	.8585E-02	000431	.5905E-04	000001
23034	0.1282E-02	52	.8200E-01	005176	.8221E-01	005205	.4425E-07	000000
23036	0.5280E-02	255	.3379	025500	.3193	024335	.3470E-03	000013
23040	0.9155E-04	3	.5000E-03	000020	.1224E-01	000621	.1379E-03	000004
23042	0.1712E-01	1061	1.096	106074	1.072	104466	.5581E-03	000022
23044	0.1291E-01	647	.8262	064700	.7990	063106	.7381E-03	000030
23046	0.2625E-02	126	.1680	012601	.1740	013105	.3610E-04	000001
23050	0.2362E-01	1406	1.512	140574	1.536	142215	.5739E-03	000022
23052	0.5157E-02	251	.3300	025075	.3225	024510	.5591E-04	000001

*Scale factor CUMSF1 equals 64.

The RMS error for PRSPEC equals .0032.

OUTPUT FROM SUBROUTINE POLYN [U(n)]*			DESCALED ARRAY V(n) = U(n) x CUMSF2		DF OUTPUT FROM TEST PROGRAM [W(n)]		ERROR SQUARE $[W(n) - V(n)]^2$	
Mem Loc	Value In Decimal	Value In Octal	Value In Decimal	Value In Octal	Value In Decimal	Value In Octal	Value In Decimal	Value In Octal
23310	-0.1221E-03	177774	-.1895	763676	.1640E-02	000065	.3653E-01	002255
23312	-0.1221E-03	177774	-.1895	763676	.4942E-03	000020	.3510E-01	002236
23314	-0.1221E-03	177774	-.1895	763676	.1836E-01	001131	.4321E-01	002607
23316	0.6104E-04	2	.9470E-01	006037	.1609	012231	.4387E-02	000217
23320	-0.1221E-03	177774	.1895	014101	.8723E-03	000034	.3558E-01	002215
23322	0.6714E-03	26	1.042	102540	1.118	107416	.5719E-02	000273
23324	0.4272E-03	16	.6630	052335	.6984	054545	.1253E-02	000051
23326	-0.6104E-04	177776	-.9470E-01	771740	.6094E-01	003714	.2422E-01	001431
23330	0.1160E-02	46	1.800	163163	1.986	177074	.3453E-01	002153
23332	0.0000E+00	0	0.	000000	.1636	012357	.2675E-01	001554

*Scale factor CUMSF2 equals 1552.09.

The RMS error for POLYN equals .1515.

TABLE J-V

Test Results for FFT Computation (Data Set #3)

Mem Loc	INPUT DATA Value In Decimal	Value In Octal	OUTPUT FROM SUBROUTINE FOURIE [F(n)]* Value In Decimal	Value In Octal	DESCALED ARRAY X(n) = SFACTR x F(n) Value In Decimal	Value In Octal	FFT OUTPUT FROM TEST PROGRAM [Y(n)] Value In Decimal	Value In Octal	ERROR SQUARE [Y(n) - X(n)]^2 Value In Decimal	Value In Octal
4172	-0.1109E+00	170717	-0.2319E+00	161120	-.4638	742242	-.4632	742265	.3532E-06	000000
4174	-0.1194E+00	170271	0.0000E+00	0	0.	000000	0.	000000	0.	000000
4176	-0.3125E-01	176000	-0.1302E+00	167524	-.2604	757253	-.2607	757240	.1089E-06	000000
4200	0.7755E-01	4755	-0.1003E+00	171451	-.2006	763122	-.2012	763075	.4038E-06	000000
4202	0.1068E+00	6655	0.5188E-03	21	.1000E-03	000003	.1033E-02	000041	.8698E-06	000000
4204	0.1158E+00	7321	-0.2823E-01	176143	-.5640E-01	776307	-.5656E-01	774277	.5600E-07	000000
4206	-0.8743E-01	172317	-0.1031E+00	171315	-.2062	762633	-.2063	762626	.2173E-07	000000
4210	-0.7394E-01	173211	-0.1165E+00	170425	-.2330	761036	-.2334	761036	.1936E-06	000000
4212	-0.1208E+00	170211	-0.5333E+00	135676	-1.067	673571	-1.067	673560	.7093E-07	000000
4214	-0.7785E-01	173011	-0.8026E-01	172672	-.1605	765564	-.1608	765551	.1131E-06	000000
4216	0.6665E-01	4210	-0.3154E+00	153642	-.6308	727501	-.6309	727500	.3102E-08	000000
4220	0.1050E+00	6560	0.8075E-01	5126	.1615	012254	.1611	012235	.1889E-06	000000
4222	0.1018E+00	6407	0.3546E+00	26544	.7092	055307	.7093	055313	.1577E-07	000000
4224	-0.7620E-01	173077	0.1705E+00	12723	.3410	025645	.3410	025647	.1860E-08	000000
4226	-0.1172E+00	170377	-0.1853E+00	164107	-.3706	750220	-.3708	750210	.5038E-07	000000
4230	0.1013E+00	6366	-0.1202E-01	177166	-.2404E-01	776354	-.2417E-01	776347	.1762E-07	000000
4232	0.7751E-01	4754	0.5621E-01	3462	.1124	007143	.1125	007145	.3568E-08	000000
4234	-0.7889E-01	172747	0.1501E+00	11466	.3002	023154	.3004	023165	.5238E-07	000000
4236	-0.9576E-01	171676	0.1373E-02	55	.2000E-03	000006	.2896E-02	000136	.7270E-05	000000
4240	0.7886E-01	5030	-0.3140E+00	153720	-.6280	727635	-.6281	727632	.7443E-08	000000
4242	-0.5359E-01	174444	0.1441E+00	11162	.2882	022343	.2881	022342	.2565E-08	000000
4244	0.6250E-01	4000	0.1393E+00	10725	.2786	021651	.2788	021660	.4932E-07	000000
4246	0.6442E-01	4077	-0.6567E-01	173630	-.1313	767461	-.1311	767467	.3305E-07	000000
4250	-0.1161E+00	170444	-0.7465E-01	173162	-.1493	766343	-.1492	766346	.8726E-08	000000
4252	-0.1139E+00	170555	0.2025E+00	14752	.4050	031727	.4052	031734	.2991E-07	000000
4254	-0.6253E-01	173777	0.1321E+00	10352	.2642	020721	.2644	020730	.5167E-07	000000
4256	0.7782E-02	377	0.1547E+00	11715	.3094	023632	.3099	023652	.2428E-06	000000
4260	0.7059E-01	4411	0.2433E+00	17445	.4866	037110	.4864	037101	.4609E-07	000000
4262	-0.9225E-01	172061	-0.9665E-01	171641	-.1933	763501	-.1933	763502	.4195E-09	000000
4264	0.1163E+00	7344	0.2798E-01	1625	.5590E-01	003447	.5576E-01	003443	.1914E-07	000000
4266	-0.1250E+00	170001	-0.1101E+00	170750	-.2202	761720	-.2202	761721	.4728E-09	000000
4270	-0.6308E-01	173755	0.8505E-01	5343	.1701	012705	.1697	012672	.1232E-06	000000

*Scale factor SFACTOR equals 20.

The RMS error for FCURIE equals .00057

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

TABLE J-VI

Test Results for Power Spectrum and CDF Computation (Data Set #3)

OUTPUT FROM SUBROUTINE FRSPEC [P(n)]*			DESCALED ARRAY R(n) = CUMSF1 x P(n)		POWER SPECTRUM RESULT FROM TEST PROGRAM [S(n)]		ERROR SQUARE [S(n) - R(n)]^2	
Mem Loc	Value In Decimal	Value In Octal	Value In Decimal	Value In Octal	Value In Decimal	Value In Octal	Value In Decimal	Value In Octal
23030	0.3357E-02	156	.2148	015576				
23032	0.1648E-02	66	.1054	006575	.2146	015566	.5784E-07	000000
23034	0.3052E-04	1	.1000E-03	000003	.1085	006742	.9481E-05	000000
23036	0.1465E-02	60	.9370E-01	005776	.3211E-02	000151	.9679E-05	000000
23040	0.1816E-01	1123	1.162	112302	.9707E-01	006154	.1139E-04	000000
23042	0.6592E-02	330	.4218	032775	1.164	112400	.3505E-05	000000
23044	0.9644E-02	474	.6172	047400	.4239	033103	.4499E-05	000000
23046	0.2136E-02	106	.1367	010577	.6195	047512	.5102E-05	000000
23050	0.1587E-02	64	.1015	006375	.1381	010655	.1346E-05	000000
23052	0.6134E-02	311	.3924	031072	.1029	006454	.2008E-05	000000
					.3945	031177	.4413E-05	000000

*Scale factor CUMSF1 equals 64.

The RMS error for PRSPEC equals .00228

OUTPUT FROM SUBROUTINE POLYN [U(n)]*			DESCALED ARRAY V(n) = U(n) x CUMSF2		DF OUTPUT FROM TEST PROGRAM [V(n)]		ERROR SQUARE [V(n) - V(n)]^2	
Mem Loc	Value In Decimal	Value In Octal	Value In Decimal	Value In Octal	Value In Decimal	Value In Octal	Value In Decimal	Value In Octal
23310	-0.6104E-04	177776	-.9470E-01	771740	.8521E-01	005350	.3237E-01	002044
23312	-0.1221E-03	177774	-.1895	763676	.2861E-01	001651	.4757E-01	003026
23314	-0.1221E-03	177774	-.1895	763676	.1025E-03	000003	.3595E-01	002231
23316	-0.1221E-03	177774	-.1895	763676	.2395E-01	001420	.4556E-01	002724
23320	0.7324E-03	30	1.137	110577	1.275	121470	.1917E-01	001164
23322	0.6104E-04	2	.9470E-01	006037	.2533	020154	.2516E-01	001470
23324	0.2441E-03	10	.3788	030174	.4648	035574	.7387E-02	000362
23326	-0.6104E-04	177776	-.9470E-01	771740	.4210E-01	002543	.1871E-01	001145
23330	-0.1221E-03	177774	-.1894	763701	.2530E-01	001535	.4653E-01	002764
23332	0.6104E-04	2	.9470E-01	006037	.2258	016346	.1718E-01	001062

*Scale factor CUMSF2 equals 1552.09.

The RMS error for POLYN equals .1719

APPENDIX K

POLYNOMIAL CURVE FITTING SUBROUTINE

APPENDIX K

Polynomial Curve Fitting Subroutine

Polynomial Curve Fitting Subroutine (PLSCF)(Ref 15)

Identification:

PLSCF Polynomial Least Squares Curve Fit

CDC 6600 FORTRAN Subroutine

Purpose. PLSCF will compute the best fitting polynomial p(x) to a discrete set of data points (x_i, y_i), i=1, NP such that

$$\sum_{i=1}^{NP} [y_i - p(x_i)]^2 = \text{minimum}$$

The coefficients of p(x) will be returned for polynomials of degree ≤ 6. For higher degree polynimal fits and, if requested, for polynomials of degree ≤ 6, the Chebyshev polynomial multipliers are returned (see remarks).

Control:

Dimension X(NP),Y(NP),W(NP),C(NC),WORK(N+1)(N+6)/2)) where N=NDEG , and NC=N+1 if NDEG>0,NC=(N+1)(N+2)/2 if NDEG<0

CALL PLSCF (X,Y,W,NP,NDEG,NMAX,C,IN,XD,XO,WORK,IER)

where

X(I) = Input array of distinct independent variables.

Y(I) = Input array of dependent variables.

W(I) = Input weights corresponding to each data point. If W(1) 0, all W(I)>are taken as +1 and need not be defined.

NP = Number of given data points.

NDEG = Degree of least square polynomial requested. If NDEG is negative, polynomials of all degrees 0 to NMAX will be computed.

NMAX = Degree of calculated least square polynomial (0 to |NDEG|). This is the maximum degree polynomial fit for which no loss of significance was indicated.

C(I) = Output coefficient array as follows:

$$p(x) = C(1) + C(2)\ x + C(3)\ x^2 + \ldots + C(N+1)\ x^2$$

If NDEG is negative, the coefficients are returned as follows:

$$P_0(x) = C(1)$$

$$P_1(x) = C(2) + C(3)\ x$$

$$P_2(s) = C(4) + C(5) + C(6)\ x^2$$

IN Input Flag

= 0 and NDEG ≤ 6, the coefficients of p(x) are returned in C.

$\neq$ 0 or NDEG 6, the Chebyshev polynomial multipliers are returned in C as follows:

$$p(x) = C(1)\ T_0(t) + C(2)\ T_1(t) + \ldots + C(N+1)\ T_N(t)$$

where

t = x . XD + X) and $T_i(t)$ = Chebyshev polynomials

If NDEG is negative, the multipliers of each degree are returned in C as follows:

$$p_0(x) = C(1)\ T_0(t)$$

$$p_1(x) = C(2)\ T_0(t) + C(3)\ T_1(t)$$

$$p_2(x) = C(4)\ T_0(t) + C(5)\ T_1(t) + C(6)\ T_2(t)$$

XD,XO = Output multiplicative constant XD and additive constant XO for lineary transformation of argument range.

WORK(I) = Working storage area of ((N+1) (N+6)/2))

IER Output Flag

= 0, no errors

= 1, errors in dimension

= 2, coinciding arguments

Methods.

- Determining least squares polynomial fits to a set of discrete data is generally meaningful only for low degree polynomials. Determining the polynomial fit for higher degrees in a straightforward manner involves solving a linear system of equations AC = B, where A is a positive definite matrix which is frequently ill-conditioned.

- This routine computes the polynomial p(x) in the form $p(x) = c_1 T_0(t) + c_2 T_1(t) + \ldots + c_m T_{m-1}(t)$ where T_i are the Chebyshev polynomials and t is a linear transformation of the original data, $(t = (2x-x_{max} + x_{min}))/x_{max}-x_{min}) = x \cdot XD + XO)$. This method results in a remarkable improvement of the normal equations.

Remarks. With this subroutine, one can obtain polynomial fits for degrees up to 6. In addition, the Chebyshev coefficients may be obtained for any degree. If one wishes to use a high degree fit, one can evaluate the Chebyshev expansion for a specified argument x using subroutine CHEB with argument t = x.XD+XO and the calculated coefficient vector of the Chebyshev expansion.

The Chebyshev expansion of the polynomial p(x) gives a much better indication of the accuracy of the approximation than the coefficient vector of the polynomial itself. If the specified degree of the polynomial is too high, the last terms of the Chebyshev expansion will be uniformly small compared to the coefficients in front. The degree may be reduced by the number of small trailing coefficients without unduly enlarging the overall error. An upper bound for the error introduced by neglecting the last terms of the Chebyshev expansion is given by the sum of the absolute values of these terms.

Storage. PLSCF uses 672_8 words of storage and no COMMON. In addition to the input and output vector arrays X,Y, W and C, a working storage area WORK((N+1)(N=6)/2)), N=|NDEG|, must be supplied.

TABLE K-I

Constants for Polynomial $Ax^2 + Bx + C$

Interval for x	Value of A Decimal (Octal)*	Value of B Decimal (Octal)*	Value of C Decimal (Octal)*
0.00 to 0.04	-.3765E-04 (177776)	.1435E-01 (000726)	.5743E+00 (044601)
0.04 to 0.10	-.3263E-03 (177765)	.4974E-01 (003136)	.7961E+00 (062746)
0.10 to 0.20	-.2890E-02 (177641)	.1033E+00 (006470)	.5080E+00 (040405)
0.20 to 0.30	-.6618E-02 (177477)	.1403E+00 (010766)	.4152E+00 (032444)
0.30 to 0.40	-.1134E-01 (177214)	.1715E+00 (012763)	.3635E+00 (027205)
0.40 to 0.60	-.1973E-01 (176571)	.2114E+00 (015416)	.3159E+00 (024157)
0.60 to 0.80	-.3397E-01 (175646)	.2588E+00 (020441)	.2762E+00 (021533)
0.80 to 1.0	.5085E-01 (174575)	.3010E+00 (023205)	.2499E+00 (017774)

*These octal equivalent values for A, B, and C were used in subroutine POLYN.

Vita

Saleem Iftekhar was born November 11, 1949 at Karachi, Pakistan. He attended Cathedral High School at Lahore, where he completed his matriculation in 1965. He joined the Government College, Lahore, and completed FSc in 1967.

In 1967, he joined the Pakistan Air Force as a cadet. He attended the Pakistan Air Force College of Aeronautical Engineering (PAFCAE) at Karachi, where he graduated in 1971 with a B.E. degree in Avionics Engineering. Upon graduation, he was commissioned as a flying officer.

He served as a radar maintenance officer for three years and was on the undergraduate faculty at PAFCAE for two years before being selected to join the resident Graduate Electrical Engineering program at the Air Force Institute of Technology. He graduated with an M. S. degree in Electrical Engineering on December 15, 1978.

He is a member of Eta Kappa Nu.

Permanent Address: 114-D, Block-2
P.E.C.H. Society
Karachi 2909
Pakistan

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT-GE/EE/78D-29	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) MICROPROCESSOR BASED DATA ACQUISITION AND PROCESSING SYSTEM		5. TYPE OF REPORT & PERIOD COVERED M.S. Thesis 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Saleem Iftekhar, Flt/Lt, PAF		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT-EN) Wright-Patterson AFB, OH 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Structural Mecahnics Division Air Force Flight Dynamics Laboratory Wright-Patterson AFB OH 45433		12. REPORT DATE December 1978 13. NUMBER OF PAGES 237
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Approved for public release; IAW AFR 190-17

JOSEPH P. HIPPS, Major, USAF
Director, Office of Information 1-23-79

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Data Acquisition
Signal Processing
Fast Fourier Transform
Microcomputer
Fatigue Damage

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The Air Force Flight Dynamics Laboratory at Wright-Patterson Air Force Base, Ohio, plans to design a constrained layer damping treatment for aircraft components. This design requires the knowledge of vibration and temperature variations that a component encounters in service.

The development of a microprocessor based data acquisition and processing system (MIBDAPS) to monitor the vibration and temperature variations of a component under test is discussed in this paper. The designed system is capable of acquiring and processing data in real time and generating cumulative damage (cont)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

factor versus frequency and temperature contours. These contours are used in the design of the damping treatment. The system uses an LSI-11M microcomputer as its central processor. The support hardware for this microcomputer includes an analog-to-digital converter, core memory, real time clock, and serial line interface module. The software developed for this system handles the acquisition and processing of data. The real time processing includes the computation of the FFT and power spectrum of the vibration signal and subsequent calculation of the cumulative damage factor versus frequency and temperature contours. The bench model of the designed system was implemented and tested for functional performance.