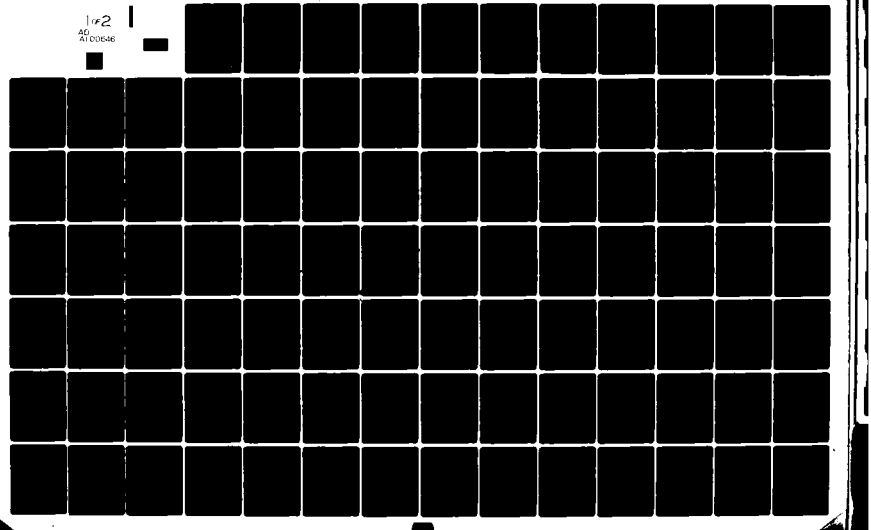
AD-A100 646 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 9/2
AIRCREW INFLIGHT PHYSIOLOGICAL DATA ACQUISITION SYSTEM,(U)
JUN 80 K L MOORE
UNCLASSIFIED AFIT/GE/EE/80-6 NL



AFIT/GE/EE/80-6

AIRCREW INFLIGHT PHYSIOLOGICAL DATA ACQUISITION SYSTEM

THESIS

AFIT/GE/EE/80-6 Kenneth L. Moore
Capt USAF

14
AFIT/GE/EE/80-6

AIRCREW INFLIGHT PHYSIOLOGICAL DATA ACQUISITION 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 Kenneth L. Moore
Capt USAF

Graduate Electrical Engineering

11 June 1980

Approved for public release; distribution unlimited

Preface

This project describes the design and simulation of a totally solid-state, self-contained data acquisition system. The system is designed to collect and store physiological and environmental data of aircrew members performing actual missions. The Rockwell System-65 minicomputer, augmented with two megabits of magnetic bubble memory, was used for operational software development and system simulation.

Many thanks go to the School of Aerospace Medicine at Brooks AFB for their invaluable assistance in obtaining hardware and for sponsoring the project. My thanks also go to Mr. Bob Durham, Mr. Dan Zamba, and Mr. Orville Wright of AFIT for their aid with laboratory simulation. For the guidance and assistance from my advisors, Dr. Ross, Dr. Kabrisky, and Dr. Lamont, I am sincerely thankful. Finally, for hours of editing and typing, and for her constant encouragement and understanding, I am deeply grateful to my wife, Becky.

Kenneth L. Moore

Contents

List of Figures

List of Tables

List of Abbreviations

A/D	Analog-to-digital
BMC	Bubble Memory Controller
C	Minimum reportable amount of change
CMOS	Complementary metal oxide semiconductor
CPD	Coil Predriver
CPG	Current Pulse Generator
CSM	Continuous Storage Method
D	Calculated difference of $\frac{\text{(Last value saved) - (this value)}}{C}$
DCSM	Delta Continuous Storage Method
DT	Drive Transistor for MBM Coil
EOC	End-of-conversion signal
EPROM	Erasable, programmable, read-only-memory
FSA	Formatter/Sense Amplifier
Gx	Acceleration in x direction
Gy	Acceleration in y direction
Gz	Acceleration in z direction
IFPDAS	Inflight Physiological Data Acquisition System
IRQ	Interrupt request
LSI	Large scale integration
MBM	Magnetic bubble memory
MVCSM	Modified Variable Change Storage Method
NMOS	N-channel metal oxide semiconductor
PIA	Peripheral Interface Adaptor

RAM	Random access memory
ROM	Read-only-memory
R/W	Read/write signal
SAH	Sample-and-hold
SAM	United States Air Force School of Aerospace Medicine, Brooks AFB, Texas
VCSM	Variable Change Storage Method
VIA	Versatile Interface Adaptor
1K	Equivalent to 1024
Ø2	Phase 2 signal

Abstract

A design is presented for a self-contained, man-mounted data acquisition system to sample and store 12 environmental and physiological parameters. The design consists of one-megabyte of nonvolatile magnetic bubble memory storage, 16 analog input channels, and four digital input channels, and is controlled by a 6502 microcomputer. Operational software was designed and simulation conducted on a Rockwell System-65 minicomputer augmented with two-megabits of magnetic bubble memory. Two types of data storage methods are examined--continuous (or pulse code modulation), and three variations of delta pulse code modulation for reduction of data storage.

Nonuniform sampling rates (or sampling jitter) caused by simultaneous sampling requests were investigated, and ways to reduce or eliminate the occurrence of jitter are also presented.

AIRCREW INFLIGHT PHYSIOLOGICAL DATA ACQUISITION SYSTEM

I Introduction

As aircraft capabilities increase, so do the physiological stresses placed on the crew. These stresses, such as low temperature, reduced oxygen and pressure, and artificial gravity, evoke certain unwanted, possibly hazardous, physiological responses. The United States Air Force School of Aerospace Medicine (SAM) has a program underway to collect and analyze data on these physiological responses during actual flight. The objectives of this program have been to evaluate the effectiveness of life support equipment and systems, determine the oxygen generation and storage requirements for various types of missions, accumulate a data base from which design criteria for new breathing systems and environmental control systems can be developed, and assess the physiological cost of flying operations (Ref 1).

As indicated by the title, this investigation addresses the data collection and storage portion of the physiological response analysis problem. The device which collects the data is called the Inflight Physiological Data Acquisition System, or IFPDAS. The current IFPDAS operated by SAM has undergone several improvements, but still uses

a cassette tape recorder as the mass storage device. As with all mechanical devices, its performance is degraded during high-G maneuvering. The existing system also requires that analog signals be converted several times before getting to the final digital state for analysis. The original analog signals are recorded on cassette tape in a pulse duration modulation format. It is changed back to analog when read from the cassette tape, and finally converted to an eight-bit digital representation. The current system is controlled by discrete logic, with each channel being sampled 32 times per second. There is no capability to change this basic sampling rate for slowly-varying signals. Faster sampling rates are obtained in increments of 32 by applying the input signal to multiple channels. Also, due to discrete logic control, the current IFPDAS has no capability for data reduction or preprocessing. Due to these inherent limitations in the existing system, there is keen interest in developing a highly flexible microprocessor-controlled IFPDAS utilizing no mechanical devices.

Background

System Requirements. SAM personnel identified several initial requirements. First, the following 12 parameters must be collected:

a. triaxial acceleration (Gx, Gy, Gz)

b. cabin pressure

c. anti-G suit pressure

d. mask pressure

e. inspired flow rate

f. inspired oxygen concentration

g. expired flow rate

h. expired oxygen concentration

i. body temperature

j. heart rate

Eleven of the above parameters are provided by sensors which generate analog signals in the range of zero to five volts. The remaining parameter, heart rate, is provided both as an analog signal and as an eight-bit digital word. The sensors used to monitor physiological parameters are noninvasive (not surgically implanted) and have an accuracy no better than 1% of full range. Therefore, 1% was set as a guideline for IFPDAS accuracy. It is further required that all parameters be time tagged so that a physiological response parameter, such as increased heart rate, can be correlated to an input parameter change, such as increased Gz acceleration.

The system must be self-contained and fit into a survival vest. These requirements imply that the system be battery powered and not larger than 2x5x9 inches. Lastly, the system must be capable of operating for at least four hours.

Previous Work. Two previous AFIT theses investigations in this area have been done. The first, by Jolda and

Wanzek (Ref 2), proposed a microprocessor-controlled system with a magnetic bubble memory (MBM) as the mass storage device. Several sensors were interfaced to an Intel 8080 microprocessor test system to demonstrate the feasibility of implementing a completely solid-state IFPDAS. The data storage algorithm used to reduce the amount of data stored averaged each signal over a 10-second period.

The second thesis investigation, by Hill (Ref 3), looked at the 12 input parameters. Their rates of change were examined and sampling rates necessary to accurately reproduce the parameters' signals were proposed. Several data storage formats were suggested and implemented on the Intel 8080 test system. To varying degrees, these storage formats traded parameter accuracy for reduced storage. A general design of the IFPDAS was proposed, and the power and space requirements for that system were specified. For the general design proposed, an implicit assumption was made that the list of 12 parameters was complete. The overall effect of this assumption was that the system was designed to the 12 parameters with no capability for expansion.

Discussions with SAM personnel indicate that, as the analysis continues, additional parameters will be identified. This fact is evident from discussions concerning their interest in various real-time preprocessing techniques of such signals as electrocardiograms. They also imply that some parameters currently recorded might be omitted in future tests, while other parameters, such as body

temperature, might be collected for several locations of interest. In short, SAM cannot, at this time, specify a complete list of parameters or the parameter mix that will be used. It is therefore impossible to design a digital IFPDAS to a set of input parameters whose number, type (analog or digital), and storage rate are not, as yet, known. Because of the limitation on space, the requirement for battery power, and the existing level of MBM technology, it is also not feasible to "over design" the system in anticipation of future needs!

Problem Statement

The purpose of this effort was to design and simulate a solid-state, self-contained, microprocessor-controlled IFPDAS. As MBM technology advances, the IFPDAS should be able to increase its capability with only minor hardware changes and little or no software changes. The objectives of the simulation were to demonstrate relationships between the parameter characteristics (number, type, and sampling rate) and each of the following:

a. amount of hardware

b. size of mass storage

c. power

d. package volume

For a given level of technology, these relationships allow the realistic determination of conditions under which a solid-state IFPDAS could function successfully.

Scope and Assumptions

Because of time constraints, this effort was limited to the collection and storage aspects for the problem of physiological response analysis. Within this guideline, the following assumptions were made to further define the scope of the investigation:

a. sensor outputs correctly represent the quantities measured

b. analog signals are in the range of zero to five volts

c. digital parameter data are represented by an eight-bit byte

Approach

For analysis the system was divided into two parts, the controller hardware and the storage hardware. The controller hardware was defined as that hardware required to collect, manipulate, and store data at the correct sampling rates. The storage hardware was defined as that hardware used solely for mass storage or the control of mass storage. Note that by this definition the random access memory (RAM), used to buffer data to the MBM, was considered as part of the storage hardware.

The first step was to define a controller hardware configuration. The basic design constraints were to provide a path for the flow of data from the inputs to the storage device at a sufficient rate--a "sufficient rate"

being defined as that required to process and store at least the original 12 parameters. The next step was to simulate the IFPDAS controller software for the controller hardware structure. The Rockwell System-65 minicomputer was the host machine for this study. A survey of available MBM was made, and the interface structure of the most promising was added to the simulation program. The storage size of the MBM, as well as that of the RAM buffer, were provided as variable inputs to be set upon program initialization. Different data storage techniques which indicated a good potential for storage reduction were examined. These were also added to the simulation program. Finally, several simulations were conducted. In each simulation one of the following was varied:

a. number of input parameters
b. sampling rates
c. data storage methods
d. size of RAM buffer
e. size of bubble storage

Sequence of Presentation

Chapter II is concerned with system hardware. First, the configuration for the controller hardware is examined. Next, the storage hardware is analyzed in light of what is currently available and what should be available in the near future. Lastly, the simulation hardware is examined and compared with the controller and storage hardware.

Chapter III deals with the simulation software and its operation. Storage methods to reduce the amount of data stored are discussed, along with accuracy and errors associated with each. This chapter also examines other nonhardware related issues. These include a discussion of sampling rates to insure signal reproducibility, methods of handling storage error, and the effect of sampling delays due to multiple simultaneous sampling requests.

Results and recommendations are presented in Chapter IV.

II Hardware

For discussion purposes, this chapter is divided into three parts: the controller hardware, the storage hardware, and the simulation hardware. The first section describes how and why the controller hardware structure was minimized. A prototype design is presented using current state-of-the-art devices. The subsequent section discusses the storage hardware, centering on the selection of an appropriate MBM. The simulation hardware is discussed in the final section.

Controller Hardware

The controller hardware was defined as that hardware required to collect, manipulate, and store the incoming data. One characteristic of the controller hardware was that the amount of hardware required was not a direct function of the number of input parameters and mission length, as was the case with the storage hardware, but was dependent on the functions that it performed. In keeping with the power and space limitations discussed earlier, a definition of a minimum controller hardware configuration was needed. Defining the minimum controller hardware had the added benefit of maximizing the physical space allotted for the MBM storage hardware. In order to minimize the controller hardware, the functions of the controller hardware were considered.

As shown in Figure 1, three functions associated with the controller hardware were identified. Interfacing the system to both analog and digital input signals was one function, labeled as the Channel Interface. Control of the interface hardware, manipulation of data for preprocessing or storage reduction, and control of data flow to mass storage were grouped as the second function, called System Control. The Mission Run Clock function was identified to provide a continuous time readout in relation to the start of the test to allow the incoming signals to be correlated in time. Having identified the functions performed by the control hardware, it was necessary to identify the hardware to perform those functions. Specific device recommendations, given in Table 1, were predicated on meeting the functional requirements of the minimum controller with currently available hardware at minimum power. Because of its extremely low power consumption and moderately fast operation, complementary metal oxide semiconductor (CMOS) devices were recommended when available.

The system requirements that came to bear on the selection of the Analog Channel Interface were to service at least the 12 original analog signals, remain near the 1% error guideline, and have low power consumption. The 16-channel ADC0817 analog data acquisition chip was selected for this function (Ref 4). The chip contains a 16-to-1 analog multiplexer, a successive approximation analog-to-digital (A/D) converter, and a tri-state output

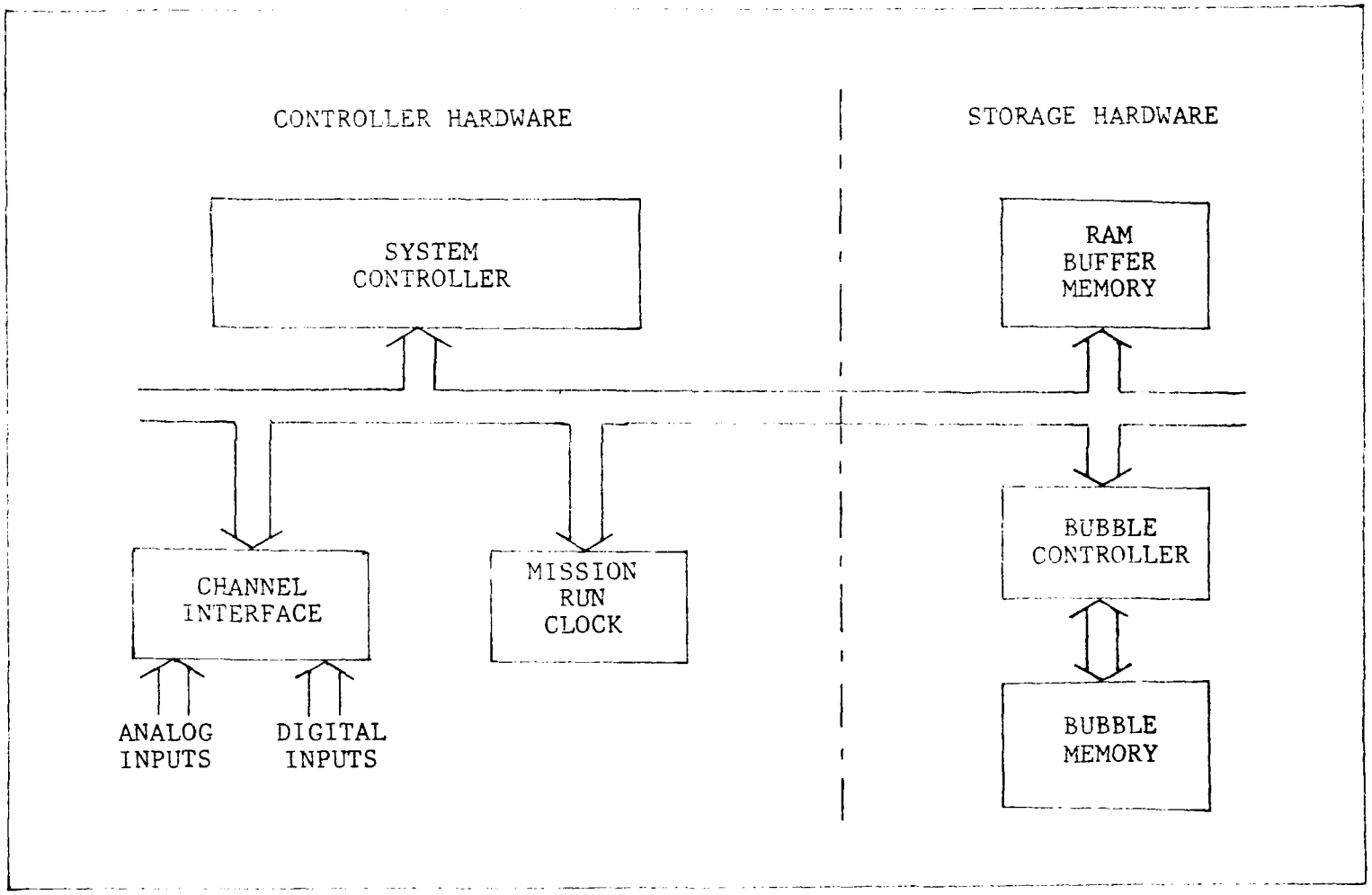


Fig. 1. IFPDAS Functional Block Diagram

TABLE I

EPDAS HARDWARE POWER REQUIREMENTS

Device	Quantity	Part Number	Max Power (milliwatts) @ 5 volts	Max Power (milliwatts) @ 12 volts	Mission Power (watt-hour) @ 5 volts	Mission Power (watt-hour) @ 12 volts
Microcomputer	1	R6502	250	---	1	---
EPROM	1	TI2564	400	---	1.6	---
RAM	2	HM6116	400	---	1.6	---
DATA BUS DRIVERS	2	CDP1857	.0005	---	.002	---
ADDRESS BUS DRIVERS	2	CD4050	.0006	---	.0024	---
GATES	1	HD74C04	*	---	*	---
	1	HD54C10	*	---	*	---
ADDRESS DECODE	1	MM54C154	*	---	*	---
DIGITAL PORTS	2	CDP1851	15	---	.06	---
ANALOG PORTS	1	ADC0817	15	---	.06	---
TIMER	1	M6840	550	---	2.2	---

TABLE I--Continued

Device	Quantity	Part Number	Max Power (milliwatts) @ 5 volts	@ 12 volts	Mission Power (watt-hour) @ 5 volts	@ 12 volts
**INTEL BUBBLE						
-BMC	1	7220	*** 250	---	*** .02	---
-CPC	8	7230	1280	3840	.1024	.3072
-FSA	8	7242	4800	2880	.384	.2304
-CPD	8	7250	---	7200	---	.576
-DT	16	7254	---	28,000	---	2.24
-MBM	8	7110	---	10,400	---	.832
TOTALS	64		7960	51,720	7.0308	4.1856

* - less than 100 nanowatts

** - 2% Duty Cycle

*** - Estimated

latch. Multiplexing the analog inputs to a single A/D converter, rather than requring an A/D converter for each signal, reduces cost, power, and space. Under microcomputer control, the analog multiplexer can access any one of the 16 single-ended analog channels. The selected channel's signal is fed through the A/D converter to produce an eight-bit byte. The total conversion time (from start-conversion to end-of-conversion flags) is 100 microseconds, resulting in a maximum rate of 10,000 conversions per second. This is well beyond the 144 samples per second required by the currently identified parameters (see Table II).

The ADC0817 performs a linear, ratiometric conversion with a total error of less than $\pm\frac{1}{2}$ of the least significant bit. This translates into a maximum conversion error of less than 0.2% for an eight-bit byte, which compares quite favorably with the 1% error guideline. In line with the low power consumption requirement, the ADC0817 is a CMOS device and consumed 15 milliwatts of power from a single five volt supply.

The ADC0817 does not contain a sample-and-hold (SAH), but one can be added externally. In deciding whether or not to use a SAH, it was necessary to examine the sampling error without the SAH.

In Figure 2 the aperture time, t_a, refers to the time uncertainty (or time window) in making a measurement. If the signal being measured changes during that time, an amplitude uncertainty, or error, results. It should be

TABLE II

PARAMETER SAMPLING RATE

Parameter	Sampling Rate (samples per second)
Inspired Flow Rate	20
Expired Flow Rate	20
Inspired Oxygen Partial Pressure	20
Expired Oxygen Partial Pressure	20
Heart Rate	8
Body Pressure	2
Mask Pressure	20
Cabin Pressure	2
G-Suit Pressure	8
Vertical Acceleration	8
Lateral Acceleration	8
Longitudinal Acceleration	8
	144 Total

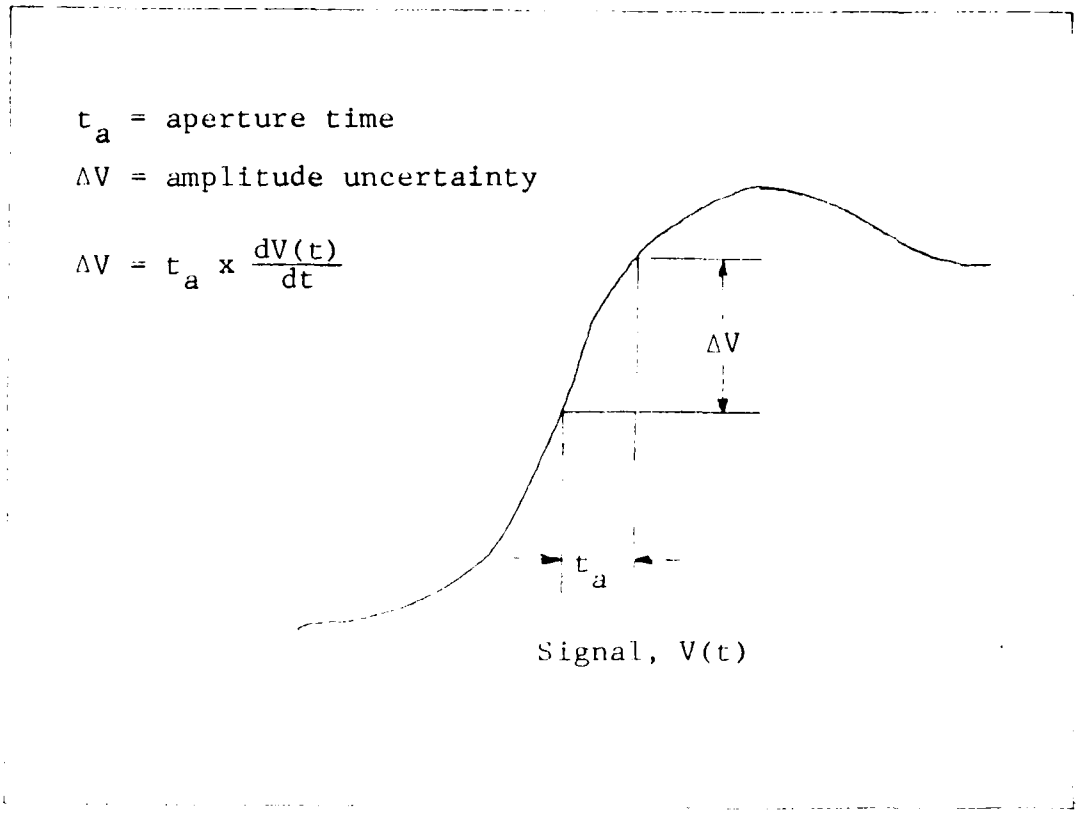


Fig. 2. Amplitude Uncertainty and Aperture Time

noted that at some point within the aperture time the signal amplitude corresponds exactly to the code word produced by the A/D converter. Therefore, the amplitude uncertainty, ΔV, represents the maximum error due to signal change. For the maximum rate of change identified in the current list of parameters (Ref 3:10), the maximum amplitude uncertainty corresponds to an error of less than 8% of the least significant bit. Therefore, the possible accuracy improvement is insignificant and the SAH is not required.

The Digital Channel Interface requires an eight-bit port for each digital input, capable of handshaking with

both the system controller and the data source. After each data collection mission, these ports could be programmed to dump the contents of MBM to a mass storage device such as digital tape, or, via a modem, transmit the data to the main computer for immediate analysis. Device selection was based mainly on power consumption. The CDP1851 contains two programmable digital ports, with handshaking control lines for each. The device requires a single five-volt supply and consumes approximately 7.5 milliwatts of power (Ref 5:97-111).

The System Control tasks were grouped for a microcomputer realization. The microcomputer selected was the Rockwell R6502. The selection was based upon four system development needs. The first was to have a microcomputer which was fast enough for current and near-term system realizations but could, with little or no design changes, meet future needs. Experimentally, the maximum sampling rate was 2380 samples per second for a single analog channel and 129 samples per second for each of 16 analog channels (total samples per second of 2064). The R6502 specified in Table I is a one-megahertz microcomputer capable of meeting foreseeable mission requirements. However, with little or no redesign, the two-megahertz version could be used to increase system response. Next, the microcomputer must have file-oriented instructions. While the R6502 microcomputer is not specifically file-oriented, it does have a straightforward instruction set with several

addressing modes, which make data file manipulations relatively easy. Most important, the R6502 has a microcomputer development system geared toward development of MBM systems. This development system (called the Rockwell System-65) was used for software simulation and is discussed later in this chapter. Although CMOS microcomputers such as the CPD1851 were available, none had the system development support hardware and software required for IFPDAS development. The R6502 consumes 250 milliwatts versus 7.5 milliwatts for the CPD1851.

The 8Kx8 erasable, programmable, read-only-memory (EPROM) (Ref 6) specified in Table I allows for the existing simulation program (approximately 4.5K), plus room for future preprocessing subroutines, without need for redesign. Because it is erasable and field programmable, initial development costs, as well as future software modification costs, will be minimized.

The Mission Run Clock function requires a programmable 16-bit counter to divide the one-megahertz system clock down to the basic sampling interval rate. The Mission Run Clock then counts the number of basic sampling intervals during the four-hour mission. Twenty-four bits are required for a one-millisecond sampling interval. The Mission Run Clock can be realized in software or (if available) in hardware. Low power consumption was the primary selection criterion for the programmable counters. However, at the time of this writing, no appropriate CMOS devices are

available. The M6840 NMOS device with three programmable counters was selected (Ref 7).

Storage Hardware

The actual size of bubble storage required depended on several variables:

a. number of input parameters

b. sampling rates

c. storage reduction methods used

d. amount of storage overhead required

e. mission length

As discussed in Chapter I, the number, rate, and type of input parameters have not been determined. Therefore, one simulation objective was to realistically determine the amount of hardware required for a given set of the above variables. The first step toward simulating the storage hardware was to determine its structure by examining the functions it performed.

The storage hardware realized three functions (see Figure 1); the RAM buffer memory, the bubble controller, and the MBM with its associated drive circuitry. The RAM buffer memory was required for two reasons. It allowed data from a particular channel (analog or digital) to be grouped in a predefined block size. Each block was then labeled with, among other information, the channel number. This reduced the amount of MBM storage overhead by eliminating the need to channel tag each piece of data. The RAM

buffer memory also allowed the bubble memory to be completely powered down when not used, thereby reducing the total power required by the storage hardware.

RAM Buffer Sizing. In selecting the RAM buffer required, several conflicting criteria were considered:

a. minimization of total power for RAM buffer and MBM
b. IFPDAS software data structure requirements
c. reduction of percentage of block header overhead
d. reduction of block manipulations due to sampling errors
e. packaging requirements

To address the first criterion, a test was conducted which simulated the effect of powering the MBM down when not in use. The objective was to determine the relationship between the amount of RAM buffer and the total power required by the storage hardware (RAM buffer and MBM). A single channel was sampled at 156 samples per second, which was slightly above the total rate specified for the original 12 parameters. The program halted after a predefined number of channel blocks were written to the Rockwell MBM. The percent of time the MBM was powered up was recorded for RAM buffer sizes from 1K to 5K in increments of 1K. A variation of the test was also run to determine the effect of sampling multiple channels. For this test, 12 channels were sampled, but the total sampling rate for the channels was kept at 156 samples per second. In all

cases, the percent of time the MBM was on was constant at approximately 1.7%. Consequently, at slow sampling rates typical for the IFPDAS, the percent of MBM "on" time was independent of the amount of RAM buffer memory. Therefore, to minimize the total storage hardware power required that only the RAM buffer power be minimized.

At program initialization each active channel was allocated a block of RAM buffer, where each block was of equal size. When a particular channel's block was full, a new block of RAM buffer was allocated and the full block was so flagged. This sequence required that the RAM buffer contain at least one more block of data than active channels. Since all channels may be active, the RAM buffer must have at least 21 blocks of data; 16 analog channels, four digital channels, and one extra. If the RAM only contained one more block than the number of active channels, the MBM was required to be on continuously.

The overhead associated with each block of data consisted of the block header. The header was made up of the channel identification (1 byte), the block start time (2 bytes), and the first value (1 byte), making the block header four bytes long. To keep the MBM overhead to 5% or less, the blocks must be at least 82 bytes long, requiring the 21 block RAM buffer to be at least 1722 bytes long.

The term "range error," sometimes referred to as "slope overload," describes the inability to represent difference values by a specified (reduced) number of bits.

(The reasons for saving difference values instead of the values themselves are discussed in Chapter III.) When a range error occurs, the remaining data in the block in which the range error occurred will be incorrect and must be corrected in some manner. Obviously, the smaller the block size, the less correction, on the average, must be done to correct for the range error occurrence. (Methods of handling range errors are also discussed in Chapter III.)

The most important criterion for RAM buffer size selection was the IFPDAS packaging requirement, which dictates high density, low power devices. The amount of RAM necessary for at least 21 blocks of buffer, plus that required for IFPDAS software, was slightly over 2K of RAM. Table I specifies two HM6116 static CMOS RAM chips. These 2Kx8 chips have the highest density at the lowest power currently available. The extra memory will allow for future preprocessing capabilities, as well as allowing the block size to be adjusted according to mission needs.

Magnetic Bubble Memory. There were three MBM devices available to choose from for a near-term IFPDAS realization. The Texas Instruments 96-kilobit MBM was eliminated because of its low package density and lack of support hardware. The Rockwell 256-kilobit MBM (used in the simulation hardware) was also eliminated. Its relatively low package density, as well as lack of special-purpose LSI control chips, precluded meeting the power and volume requirements (Refs 8 and 9). The Intel Magnetics one-megabit MBM and

its support electronics was chosen as the most promising for a near-term IFPDAS realization (Ref 10). The most appealing aspects of the Intel bubble were its relatively high package density and special-purpose support electronics, both of which greatly reduced the physical space and power required by the storage hardware. The support electronics consisted of a bubble memory controller chip capable of controlling eight bubble memory devices, and five other chips which supply the drive and timing signals to the bubble device. To provide the capability to test higher density devices, while keeping the results tied to a currently available device, only the structure of the Intel bubble was simulated. The amount of bubble memory, as well as RAM buffer memory, were varied in the simulation from run-to-run.

Using the sampling rates shown in Table II, the amount of MBM required was 2,073,600 eight-bit bytes for a four-hour mission. The Delta Continuous Storage Method (discussed in Chapter III) reduced by half the amount of storage required. (The other techniques discussed in Chapter III had greater potential for data reduction, but only this storage technique guaranteed a reduction by half.) This implied that at least 1,036,800 eight-bit bytes were required to insure sufficient storage for a four-hour mission.

For a near-term realization, eight Intel MBM devices, providing 1,048,576 eight-bit bytes, are required. Figure 3

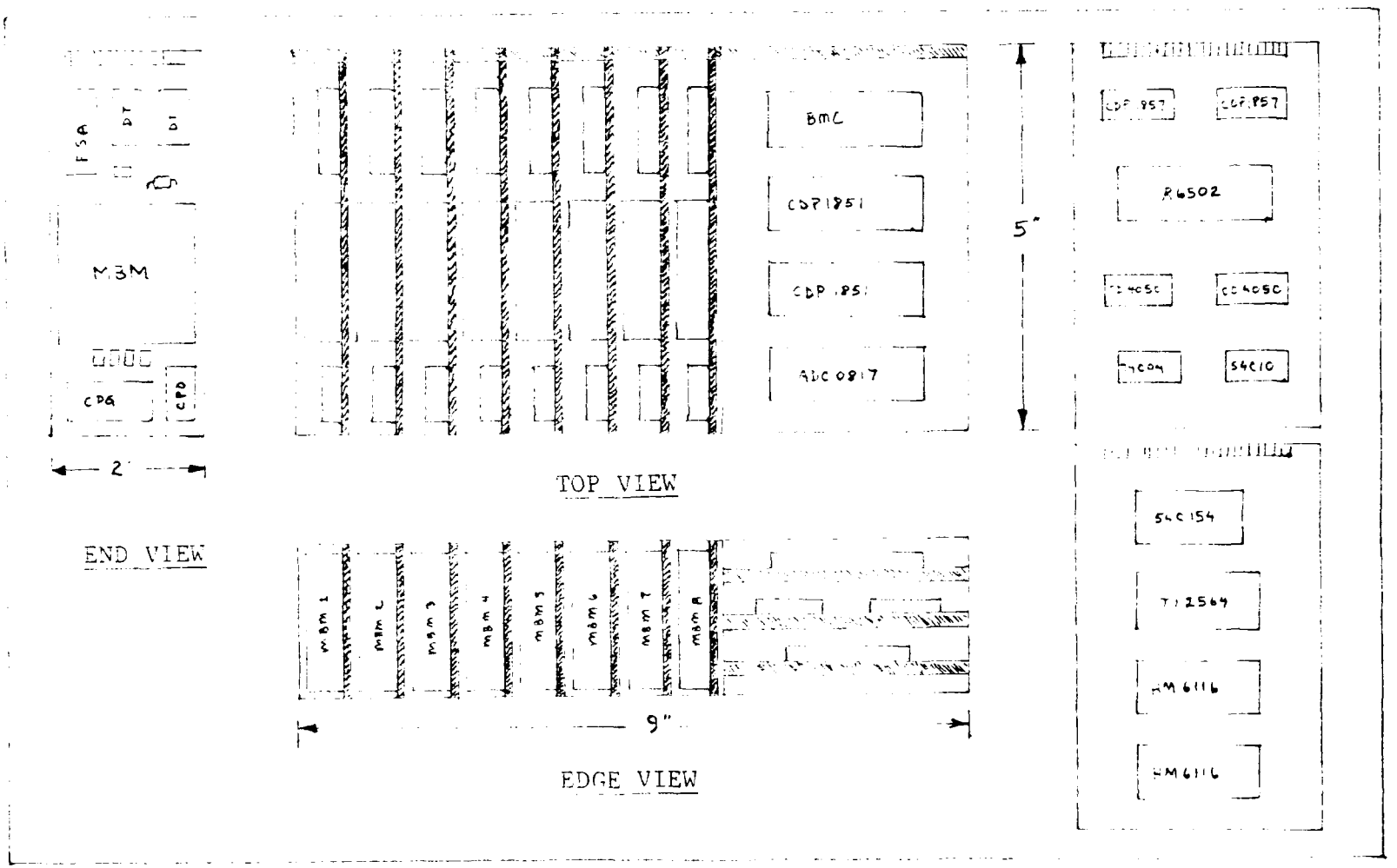


Fig. 3. IFPDAS Layout--One-half Scale

is a one-half scale drawing of the proposed IFPDAS layout. Because of the obvious crowded conditions, the IFPDAS power would be supplied by another module. The crowded conditions make routing of the address, data, and control busses difficult, necessitating the use of multilayered printed-circuit boards. Also, the existing Intel printed-circuit board is too large and requires a redesign.

It is evident from the above discussion that the capability of the IFPDAS is limited by the density of the MBM currently obtainable. Recent experimental and theoretical results by Bell Labs (Ref 11) promise a quadrupling of the storage density at a bit rate per device of one million bits per second or greater, as compared to the 50-100 thousand bits per second of existing devices. Bubble movement was derived from patterned conducting sheets instead of orthogonal field coils. This had the added benefits of reducing the power required by the MBM device, simplifying the control circuitry, and further reducing the physical space required. Also, the device required a single five-volt source rather than the five- and twelve-volt sources currently required. This would eliminate the need for multiple power sources in the IFPDAS.

Simulation Hardware

The objective of the simulation hardware was to duplicate both the control hardware and the storage hardware structures as closely as possible to simulate IFPDAS

operations. The simulation hardware was chosen to meet structural and functional requirements and to be readily available. The remainder of this section discusses the specific hardware used for the simulation. The interconnections of the simulation hardware are shown via the block diagram in Figure 4.

Rockwell System-65. The heart of the simulation hardware was the Rockwell System-65 minicomputer (Ref 12). It performed the System Control function of the controller hardware. The System-65's MBM subsystem (Ref 13) also enabled it to perform all functions associated with the storage hardware. The MBM subsystem consists of a MBM controller board and up to 16 MBM boards (two MBM boards were used in the simulation). Each MBM board has four 256-kilobit devices along with drive circuitry. The structure of the System-65 matched the structure chosen for the minimum configuration IFPDAS.

The System-65 was specifically designed to aid in the development of microcomputer software systems. Its development support includes:

a. a ROM resident interactive system monitor
b. a ROM resident assembly language compiler
c. a ROM resident debut routine
d. a higher order language compiler (PL-65)
e. two mini-floppy disks and support software
f. hardware in-circuit emulator

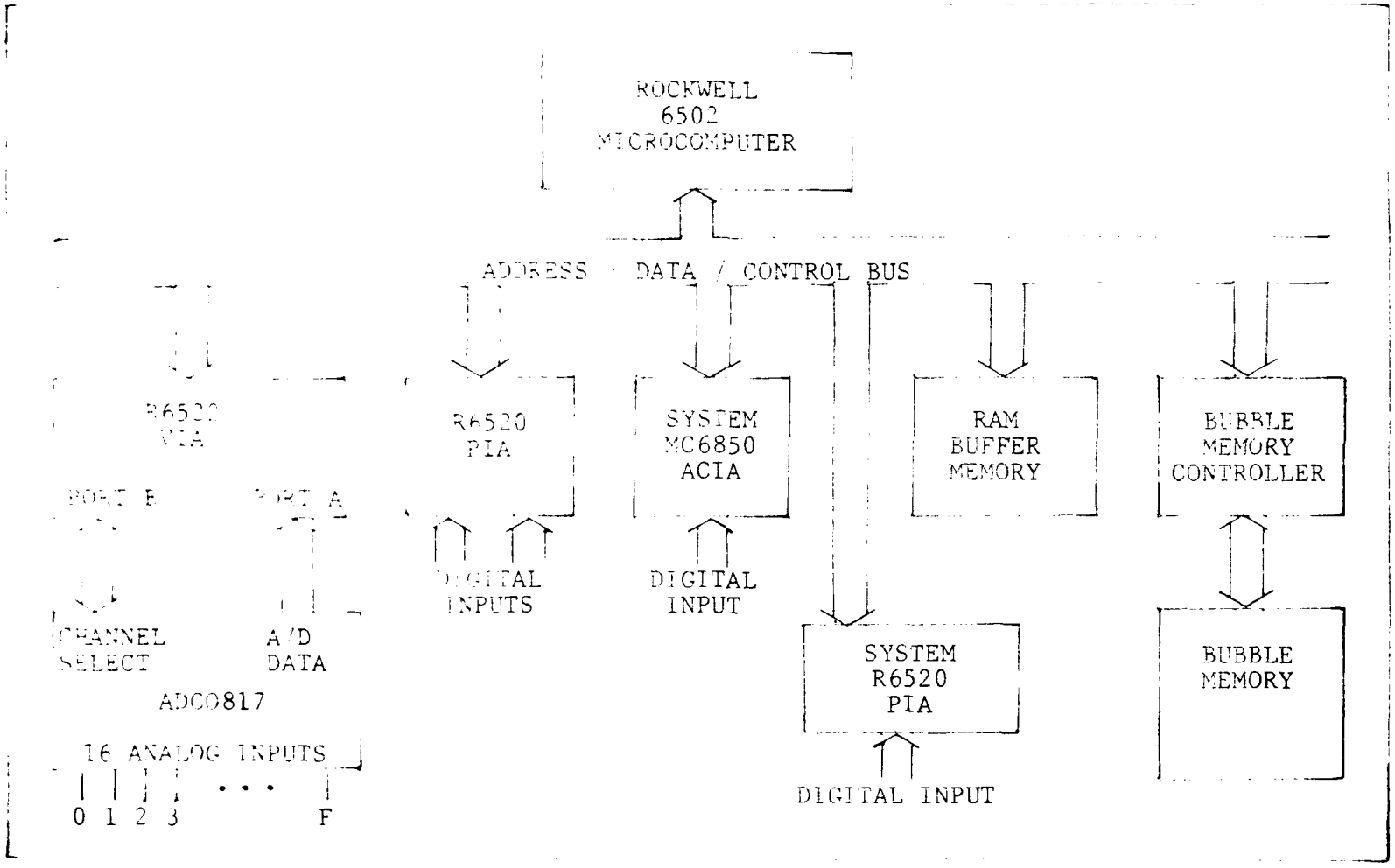


Fig. 4. Simulation Hardware Block Diagram

g. an EPROM programmer

h. parallel and serial terminal support from 110 to 9600 baud

The simulation software developed on the System-65 could be easily transitioned to a R6502 microcomputer-based IFPDAS prototype. This, along with its existing MBM capability, made the System-65 ideally suited as the simulation host machine.

Data Acquisition Hardware Board. The remaining functions, analog and digital channel interface and the Mission Run Clock, were simulated on a data acquisition hardware board. While the board was functionally equivalent to that specified for the minimum configuration IFPDAS hardware, it contained some nonessential hardware. See Appendix B for details of the Data Acquisition Hardware board.

The analog channel interface function was performed by two 40-pin chips; the ADC0817 data acquisition chip described earlier for the IFPDAS prototype, and the R6522 Versatile Interface Adaptor (VIA) (Ref 14: Sec 6). The R6522 VIA has two peripheral ports, each with two control lines, which provided an interface between the System-65 and the ADC0817 data acquisition chip.

The R6522 VIA also has two independent 16-bit interval timers which were used to provide a programmable Mission Run Clock. The first timer was programmed to provide a pulse at the basic sampling interval, while the second timer counted the number of pulses to provide 16 bits of

the 24 bits required for the Mission Run Clock. The remaining eight bits were realized by incrementing a memory location whenever the Mission Run Clock counter overflowed. The R6522 VIA also contains a serial input/output eight-bit shift register which might be useful during IFPDAS prototyping.

Two digital channels were provided by the MC6820 Peripheral Interface Adaptor (PIA). Both of the PIA's parallel ports have programmable control lines for handshaking with the external device as well as an interrupt signal to the microcomputer. The System-65 has two additional digital ports which could be used for simulation--the serial port to which the system terminal is attached, and the parallel printer port (Ref 12); however, neither was used.

The data acquisition hardware board also has a M6840. The M6840 contains three independent 16-bit, programmable interval timers. This chip was added as a tool for simulation.

III Software

This chapter deals with the IFPDAS controller simulation software. First, the simulation software design is discussed and a detailed description given. Next, an analysis of the sampling rate to insure signal reproducibility, followed by a discussion of the data reduction storage methods used, is given. Lastly, the analog sampling delays due to simultaneous request and the possible sampling jitter they cause are considered. The worst-case jitter is closely examined and a method to reduce the occurrence of jitter is presented.

Design Method

The simulation software was designed in a top-down (sometimes called structured programming) manner. Myers defined structured programming as "the attitude of writing code with the intent of communicating with people instead of machines" (Ref 15:130). While he did not give a more precise definition, he did define five "acceptable" programming constructs which produce readable code. These five constructs, shown in Figure 5, were used extensively in the software design. Other structured programming "do's" and "don'ts" that were used as design guidelines are as follows:

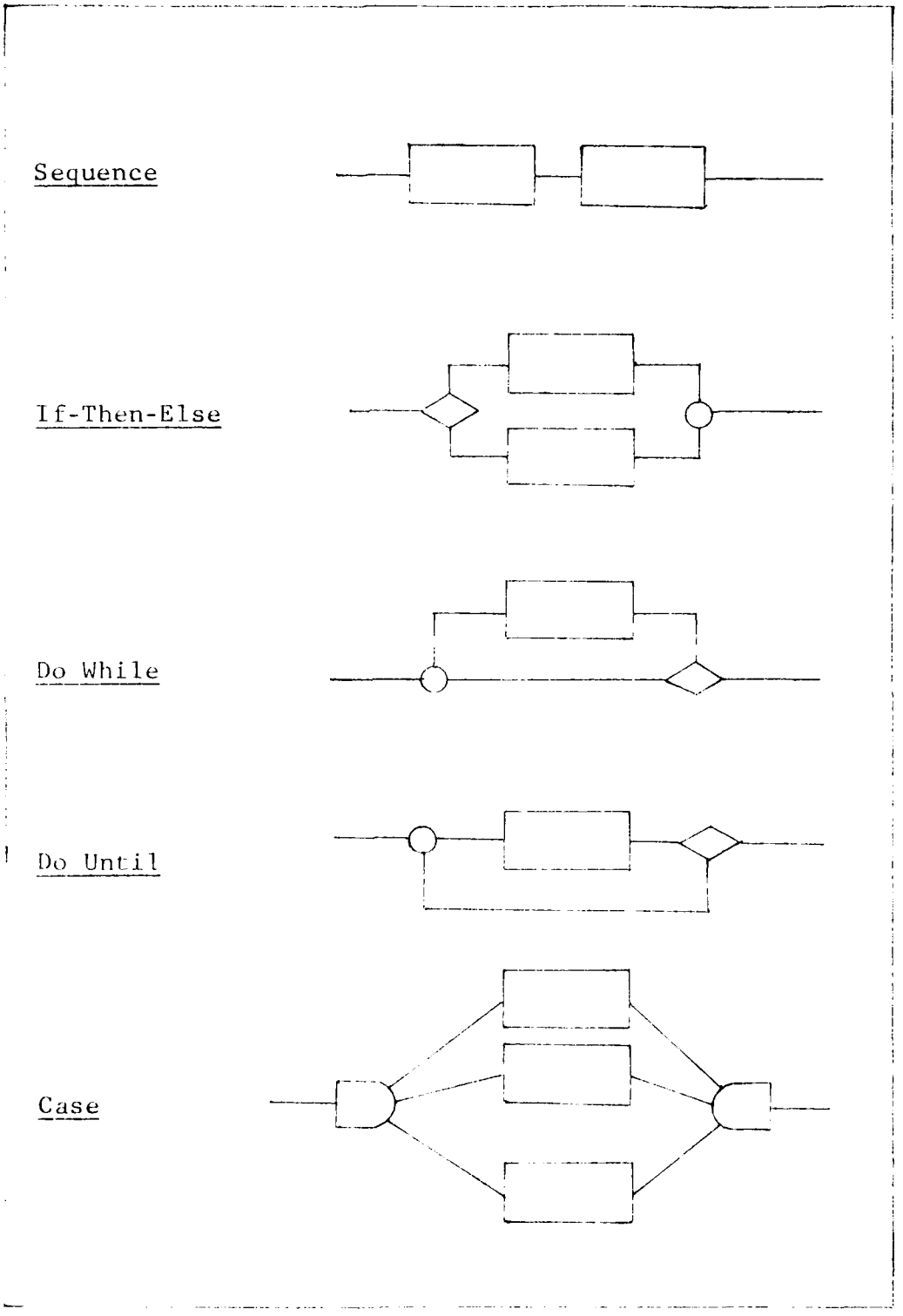


Fig. 5. Structured Programming Constructs

a. software modules performed a single function

b. module independence was maximized

c. module coupling was minimized

d. module size was small

e. modules were predictable (had no memory of previous calls that modified the execution)

Certain guidelines of "pure" structured programming were, however, violated. Specifically, because of the real-time aspects of manipulating certain array variables, the use of global array variables was necessary for those cases. The use of global variables resulted in some data handling subroutines being, to a large degree, dependent on the data structure chosen, another violation of "pure" structured programming practices. In both cases, the degree of program complexity and obscurity was significantly reduced, and it was determined that this departure from the structured programming approach was warranted.

Software Design

The software was designed in two parts: the preflight and postflight software, and the real-time mission run software. The preflight software allowed the operator to initialize the particular mission scenario for:

a. signal parameter characteristics

b. basic sampling interval

c. RAM buffer size

d. MBM size

e. mission end time

The postflight software allowed the contents of the MBM to be dumped in a graphic format to a specified device/port. For both preflight and postflight software, existing System-65 input/output routines were used as required. This allowed more attention to be focused on the mission run software.

The mission run software duplicated, as closely as possible, the real-time operation of the minimum configuration IFPDAS. Exact duplication was not possible due to simulation overhead calculations such as the amount of time the bubble was powered up/down. The simulation overhead was kept to a minimum and did not appreciably affect system performance. The program listing is given in Appendix A.

Software Description

The mission run software was designed using the interrupt capability of the R6502 microcomputer. When an interrupt request (IRQ) signal was detected by the microcomputer, the Main program was halted and the interrupt handler polled the possible requesting devices in the order given:

a. Mission Run Clock overflow

b. Basic System Clock timeout

c. A/D conversion complete

The order in which the requesting devices were polled dictated the relative priority of each device. This interrupt polling method was chosen over a hardware-vectored interrupt method to keep the control hardware minimized. With the relatively slow sampling rates of the original 12 parameters, the interrupt polling method proved more than adequate.

The Main Program. The Main program continually monitored the status of the RAM and MBM. When it was determined that the available RAM buffer memory was at or below a predefined level (usually 20%), the Main program powered up the MBM and evoked a subroutine to flush the data from RAM buffer to the MBM. All full blocks associated with the fastest channel were flushed and the pointers were updated before the next-fastest channel was considered. When a particular block of RAM buffer was flushed, it was returned to a stack of available memory for subsequent use. When all active channels had been flushed to the MBM, the Main program again checked the amount of available RAM buffer memory before powering down the MBM and starting the sequence again. Powering up and down of the MBM was simulated because that feature was not available on the System-65.

Basic System Clock Interrupt. The Basic System Clock was a 16-bit programmable timer which provided the basic, elemental time increments. Each channel's sampling interval could then be programmed as an integer multiple (1-255)

of this basic time increment. The input to the Basic System Clock was the one-megahertz microcomputer clock.

An interrupt occurred each time the Basic System Clock counted the predefined number of one-megahertz pulses. The interrupt handler then checked each channel (fastest channels first) to see which should be sampled. When it was determined that a channel should be sampled, the A/D conversion was initiated, and the mission run time for that sample was saved. If the A/D converter was busy, a flag was set to indicate the channel needed to be sampled. When all channels were checked, program control was returned to the Main program.

Mission Run Clock Interrupt. The purpose of the Mission Run Clock was to provide a count of the number of elemental time increments throughout the entire mission. For this simulation, the Mission Run Clock was realized as a 16-bit hardware counter and a memory location to store the number of clock overflows. This resulted in a 24-bit Mission Run Clock.

An interrupt was generated when the 16-bit hardware counter overflowed. The Mission Run Clock handler then incremented the overflow memory location and checked to see if the allowed simulation time had elapsed. If the allowed simulation time had elapsed, the simulation was halted; otherwise program control was returned to the Main program.

A/D Conversion Complete Interrupt. An interrupt was generated by the A/D converter upon conversion completion.

The End-of-Conversion handler first saved the value just converted and then checked to see if any other channels (starting with the fastest) were flagged as needing to be sampled. If a channel was so flagged, conversion for that channel was initiated and its flag cleared. The handler then determined, according to the particular storage method, if the converted value just saved should be kept. If the data was to be kept, it was formated as dictated by the storage method for that channel, and placed in a block of RAM buffer designated for that channel. If the placement of the data filled the block, then another block was allocated from the list of available RAM buffer memory, and channel header information written on the block. Control was then returned to the Main program.

Channel Service Request Interrupt. The channel service request provided an alternate means for sampling data. Instead of sampling the data at predefined intervals, the channel was only sampled upon request. This method was used exclusively for the digital channels during the simulation, but could be used for analog channels. Likewise, the digital channels could be automatically sampled at predefined intervals, as was done with the analog channels.

The storage method used for a channel service request was the continuous method; however, variations of any of the storage methods discussed later in this chapter could be used under appropriate conditions.

Channel Sampling

This section deals with several aspects of channel sampling. First, the sampling rate to insure signal reproduction is discussed. The storage methods used in the simulation are then described. Lastly, the three storage reduction methods are compared to the Continuous Storage Method, and the benefits and drawbacks of each are examined.

Signal Reproduction. The Shannon Sampling Theorem defines the sampling rate that assures the complete recovery of a band-limited signal (after appropriate filtering). This theorem can be stated as follows:

> If a continuous, band-limited signal contains no frequency components higher than f_c, then the original signal can be recovered without distortion if it is sampled at a rate of at least $2f_c$ samples per second. (Ref 16)

This concept is illustrated in Figure 6. The frequency spectrum of the signal being sampled is repeated at the sampling frequency.

If the sampling frequency, f_s, is at least twice the signal's cutoff frequency, no "frequency folding" occurs. In reproducing the original signal, frequency folding causes distortion. The effect of an inadequate sampling rate produces a phenomena called aliasing, in which the signal appears to vary at a much slower frequency (called the alias frequency). This effect is shown in Figure 7 for a sinusoidal input.

As indicated in Figure 6, recreation of the original signal required an ideal low-pass filter, a mathematical

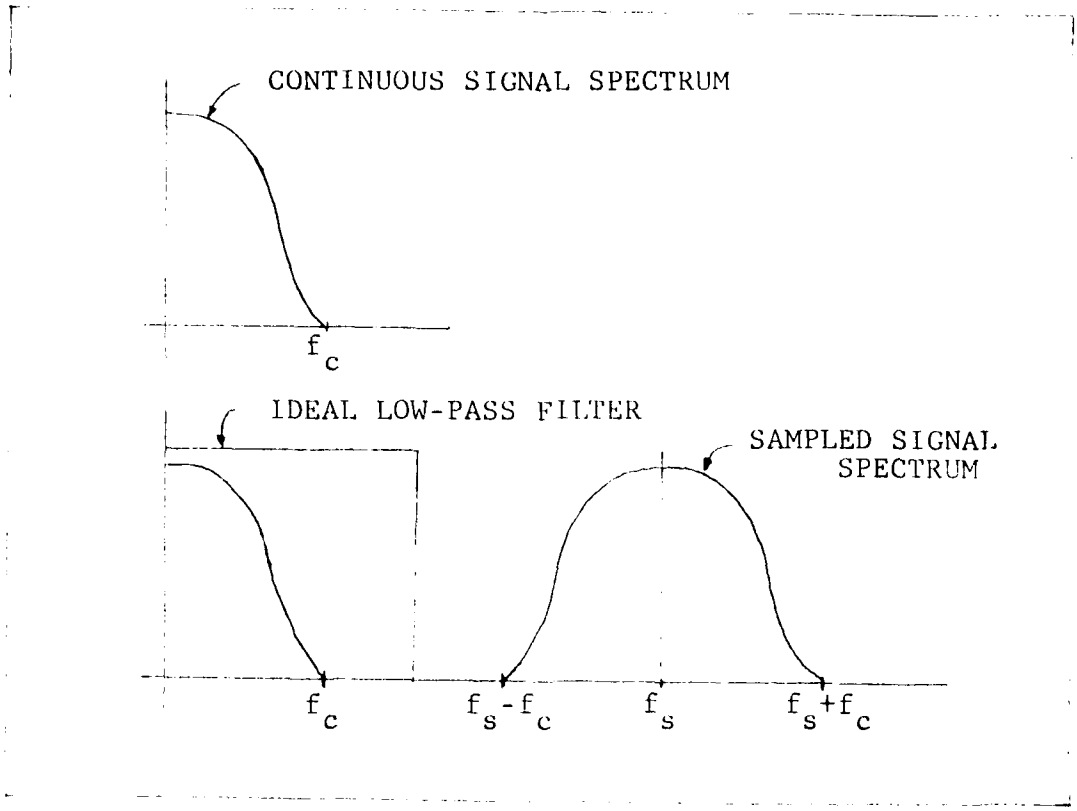


Fig. 6. Frequency Spectra Demonstrating the Shannon Sampling Theorem

fiction. However, the error from a realizable low-pass filter can be made arbitrarily small by increasing the order of the filter. In practice, however, aliasing is reduced by increasing the sampling frequency, f_s. A rule of thumb is to sample six to eight times the signal's highest frequency component.

Description of Storage Methods. Each storage method presented in this section had its own strong and weak points. Each parameter input should be examined and matched to the appropriate storage method according to the guidelines presented.

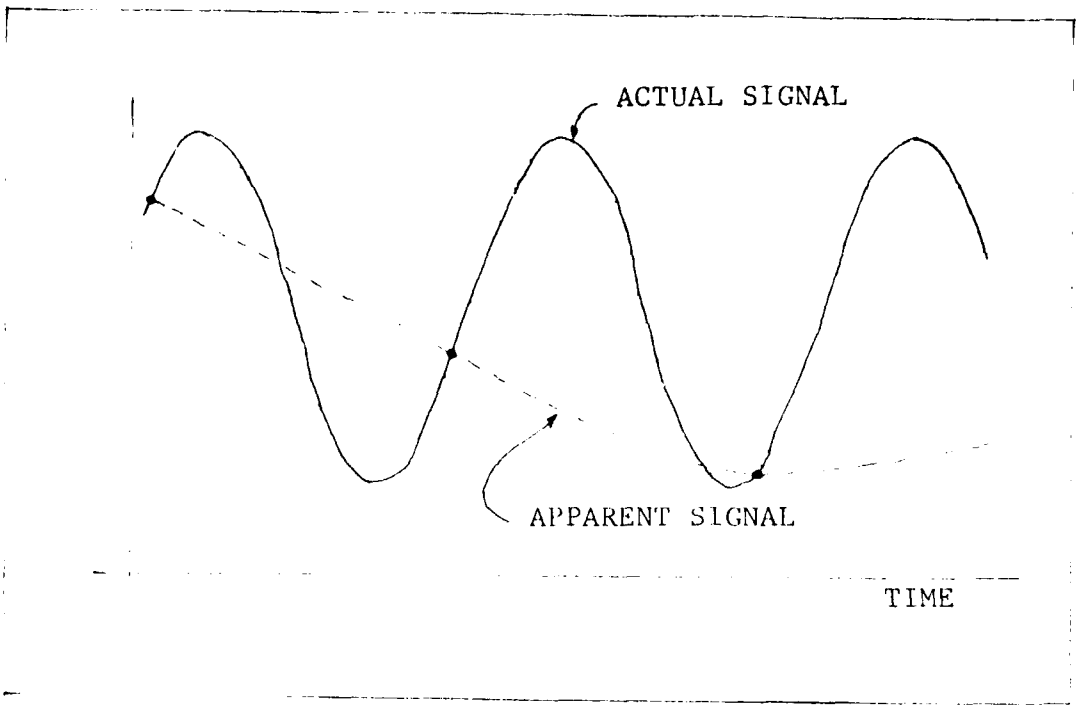


Fig. 7. Aliasing Caused by an Inadequate Sampling Rate

Five storage methods were suggested by Hill to reduce the storage required (Ref 3:15-20,51-60). Of those, the continuous and variable change methods were judged both feasible and within the 1% error guideline. A variation of each method, Delta Continuous and Modified Variable Change, is also presented in the following discussion.

Continuous Storage Method. The Continuous Storage Method (CSM) saved the data value for each sample taken. Because each sample was stored, and the time between samples was known, there was no need to time tag the individual samples. This method added no additional error above that of the A/D converter alone (less than 0.2%) and had the

smallest overall error of the methods examined. Although this method had the highest accuracy of the methods examined, it also lacked potential for storage reduction. Table III shows the storage required for a four-hour mission. The CSM should be used for signals which require maximum accuracy.

TABLE III

FOUR-HOUR STORAGE REQUIRED FOR CONTINUOUS STORAGE METHOD

Samples per Second	Storage Required (eight-bit bytes)
20	288,000
8	115,200
4	57,600
2	28,800

Delta Continuous Storage Method. The Delta Continuous Storage Method (DCSM) differed from the CSM in that the sign plus two's complement difference between the current value and the previously stored value, rather than the current value itself, was stored. The difference was represented in a four-bit, sign plus two's complement format, as shown in Figure 8. This method reduced the storage to half that required by the continuous method. This storage reduction was not without cost. Storing the difference, rather than the value, required the difference be in the range of possible four-bit, sign plus two's complement

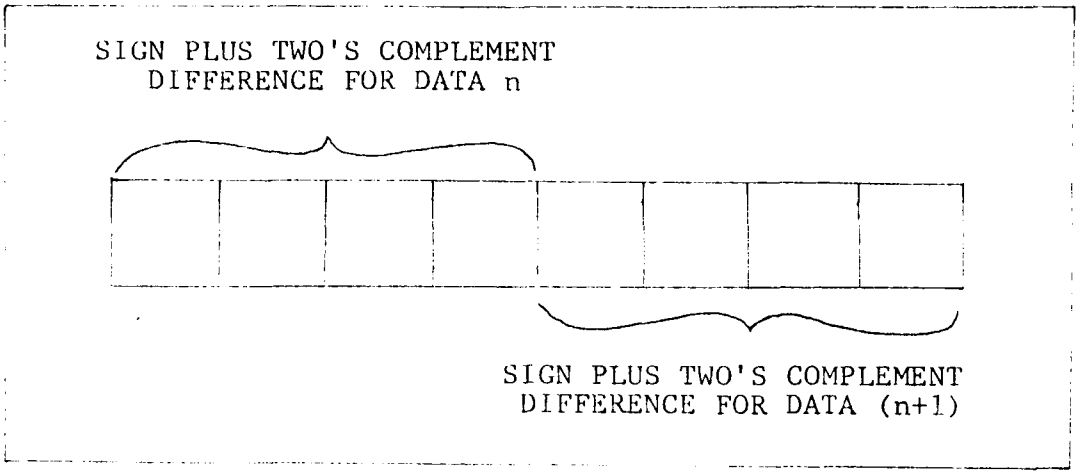


Fig. 8. Delta Continuous Storage Method Format

numbers (-8 to +7). A difference value larger than this resulted in a range error and caused the subsequent data within a block of data to be incorrect. (By storing the first value in the block header, the rippling effect was limited to a single block and did not carry over to the next block.) The accuracy associated with this method depended on the value chosen as the minimum reportable amount of change, C. The current difference value, D_n, was calculated to the nearest whole number as

$$D_n = \frac{\text{(last value saved)} - \text{(this value)}}{C}$$

where

(this value) = value just obtained from A/D conversion

(last value saved) = $C \times D_{n-1}$

At a given sampling rate, a larger C value reduced the amount of storage required and reduced the probability of having a range error, but did so at the expense of accuracy. Table IV shows the relationship between a given C value and the maximum error possible due to A/D conversion and storage for this and the remaining methods used.

TABLE IV

RELATION BETWEEN MINIMUM REPORTABLE CHANGE AND THE ERROR ASSOCIATED

Minimum Reportable Change, C	Maximum Error Due to A/D Conversion and Storage
1	0.7%
2	1.17%
3	1.56%
4	1.95%
5	2.34%

Variable Change Storage Method. The Variable Change Storage Method (VCSM), like the DCSM, stored the difference between the current value and the previously stored value, rather than the value itself. The accuracy associated with the VCSM also depended on the C value selected, and is shown in Table IV. The VCSM stored data only when the current value differed from the previously stored value by at least C. This required each difference value to be time tagged with the number of elapsed sampling intervals since the previously stored sample. Figure 9 shows the format

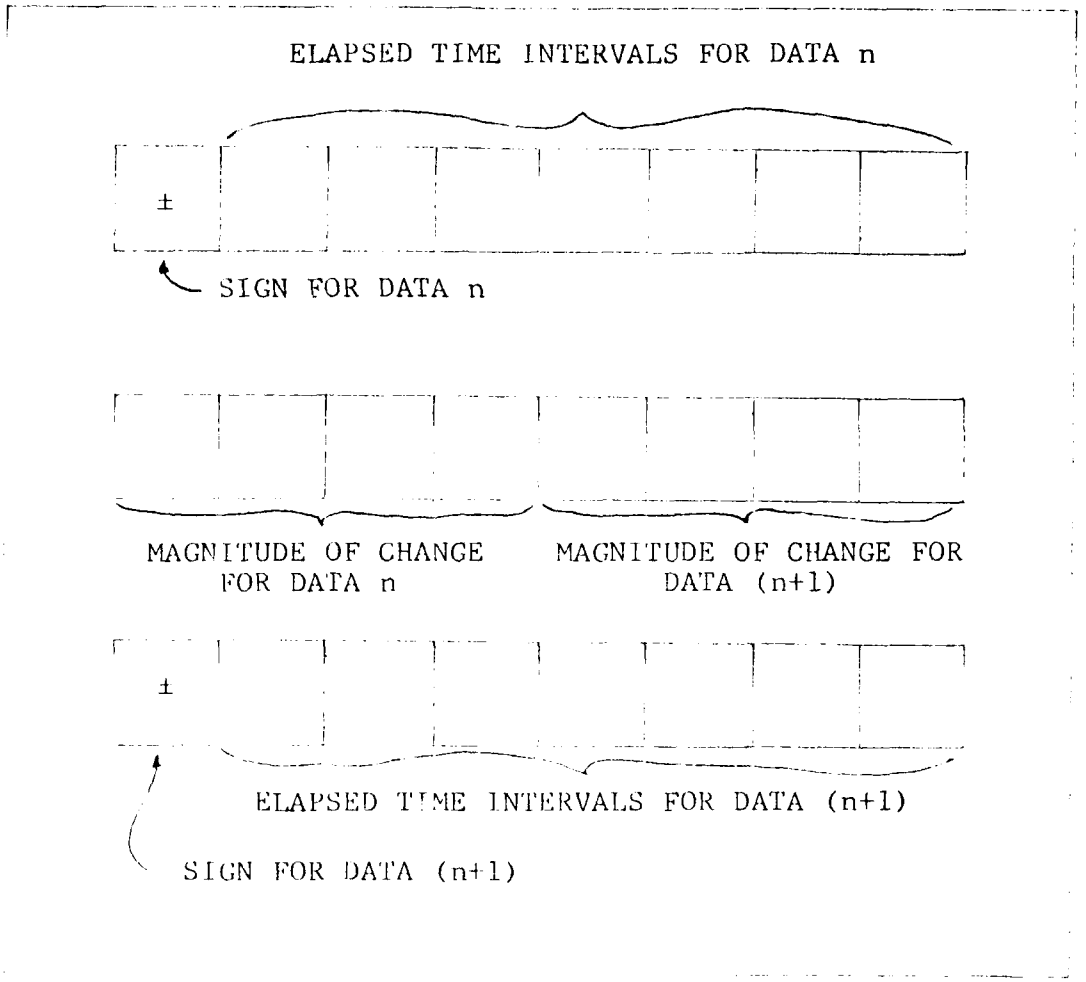


Fig. 9. Variable Change Storage Method Format

used for the VCSM. If the input signal did not change by at least C before the time tag overflowed (128 sampling intervals), a "no change" value was saved. Therefore, the time correlation from sample to sample was maintained. Table V represents the storage required for different sampling rates in terms of the maximum and minimum number of eight-bit bytes required. The numbers represent the data generated by one input signal during a four-hour mission. The maximum storage was required when every sample taken

TABLE V

FOUR-HOUR STORAGE REQUIREMENTS FOR VARIABLE CHANGE STORAGE METHOD

Rate (Samples per Second)	Storage (eight-bit bytes) Minimum	Maximum
20	3375	432,000
8	1350	172,800
4	675	86,400
2	337.5	43,200

differed from the previous by at least C. The minimum storage was required when the input signal was stored only as the time tag overflowed. The minimum values indicated the VCSM's potential for storage reduction. The maximum values indicated the storage penalty possible. Like the DCSM, this method required the difference value to be within the allowable range (-15 to +15).

Modified Variable Change Storage Method. The Modified Variable Change Storage Method (MVCSM) used the basic data structure of the VCSM. As seen in Figure 10, the MVCSM saved one data entry (time tag and difference value) in a single eight-bit byte. As shown in Table VI, this method had a potential for storage reduction, although less than that of the VCSM. However, for the worst-case condition where every sample taken was stored, the storage penalty was no worse than that of the CSM.

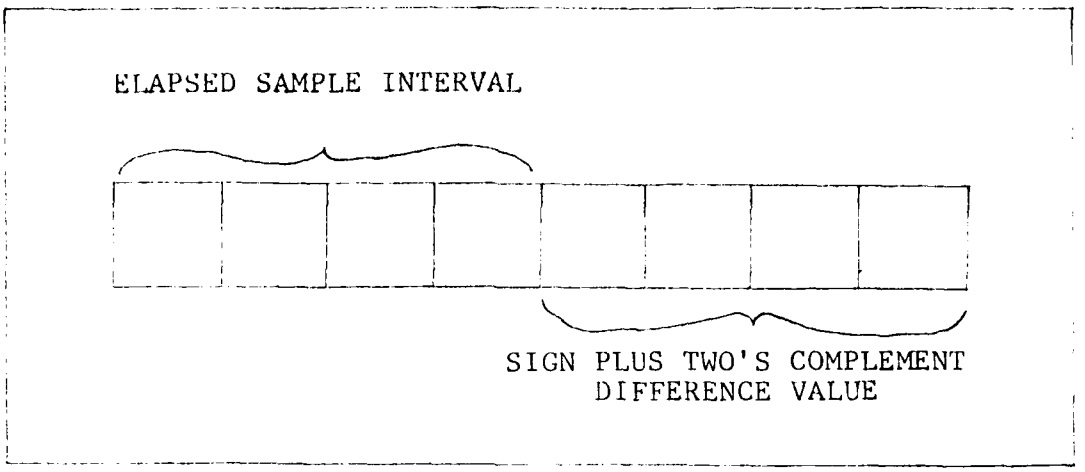


Fig. 10. Modified Variable Change Storage Method Format

TABLE VI

FOUR-HOUR STORAGE REQUIREMENTS FOR
MODIFIED VARIABLE CHANGE STORAGE METHOD

Rate (Samples per Second)	Storage (eight-bit bytes) Minimum	Maximum
20	18,000	288,000
8	7,200	115,200
4	3,600	57,600
2	1,800	28,800

As with the previous method that stored differences, this method required the difference to be in the allowable range (-8 to +7 in this case). The time tag overflowed and a "no change" entry was saved after 16 sampling periods had elapsed. Also, like the previous difference methods, the error was dependent on the value of C chosen.

Storage Method Comparison. A graphical representation of the amount of storage required by each method is shown in Figure 11. To account for various sampling rates or mission lengths, the data is normalized to the Continuous Storage Method. For input signals which change by a small amount, the Continuous Storage Method requires the full eight bits to convey as little as one bit of additional information. For storage-bound applications such as this, alternative storage methods were needed for these types of signals.

The last three storage methods discussed above were variations of what the literature called delta pulse code modulation (Ref 17:218). The appeal of these storage types was that, by storing the difference instead of the signal itself, fewer bits could be used to convey essentially the same information.

The drawback to these methods was that, as soon as the number of bits was defined, an allowable range of variation from one sample to the next was also defined. These methods should be used only for signals which usually do not vary by more than the method's allowable range. As long as the

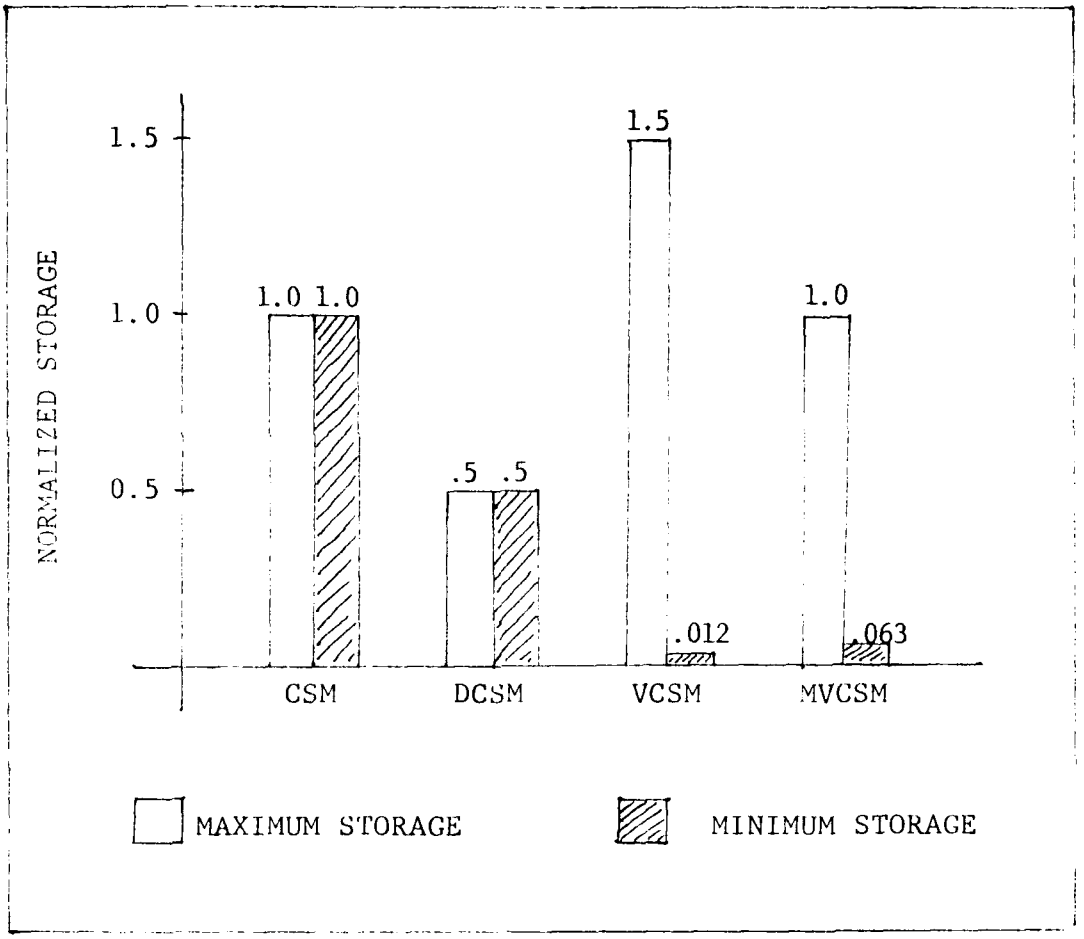


Fig. 11. Normalized Storage Versus Storage Required

signal variations from sample to sample were less than or equal to the allowable range, these methods worked admirably. However, when a difference storage method was used for input signals, which consistently varied by more than the allowable range, one of two courses had to be taken: either the sampling rate was increased, thereby requiring more storage memory, or the minimum reportable change, C, was increased.

Increasing the sampling rate to eliminate range errors can cause the resulting storage to be more than that required

by the Continuous Storage Method. In a test case, in which a zero to five-volt sinusoidal test signal was sampled, all difference storage methods required significantly more storage than the Continuous Storage Method because the sampling rate was increased to eliminate range errors.

In many cases the occurrence of a small percentage of range errors might be acceptable, when compared to the sampling rate required to insure that no range errors occur. Whether a small percentage of range errors is knowingly allowed or not, the software should recognize and handle range errors to prevent erroneous data from being stored.

In the simulation conducted, the occurrence of a range error caused the software to stop the simulation with a "range error" message. The following paragraphs discuss possible methods of handling range errors.

One way to handle range errors would be to discard the entire block of data containing the error. The smaller the block, there will be less data lost. This approach would be justified if the amount of storage memory were marginal, the probability of a range error were small, and the occurrence of small gaps in the data would not invalidate the entire test.

If preservation of all data were necessary, then the program could zero-fill the remaining portion of the block and start a new block. Again, the smaller the block, the less zero-filled MBM storage there will be. This approach

should be used only when the probability for range error is small, to prevent excessive zero-filled storage.

Another approach would be to store the maximum difference possible until the signal could again be correctly represented. This approach produces a "signal tracking error" whenever range errors occur. If the signal tracking error were acceptable, this method would be desirable from a storage point of view.

Lastly, the sampling of a particular channel could be adaptively adjusted throughout the mission. For instance, a predefined number of range errors would cause the minimum reportable change or the sampling rate to be increased. Similarly, repeated storage of "no change" would cause the sampling rate or the minimum reportable change to be decreased. If required, the sampling method could also be adaptively changed to match signal to storage method. While this approach is beneficial in many respects, it would require added storage overhead (block header information) to indicate sampling rate, minimum reportable change, and storage method.

The second course to prevent range errors, increasing the minimum reportable change, C, can reduce the amount of storage required, but does so at the expense of sampling accuracy (see Table IV). For the DCSM, increasing C allowed the sampling rate to be reduced, thereby decreasing the amount of data stored. For both the VCSM and MVCSM,

increasing C also resulted in more "no changes" between samples, further reducing the amount of storage required.

A qualitative test was conducted to determine the tradeoffs between sampling accuracy and storage requirement. A zero to five volt sinusoidal test signal was sampled using the three difference methods discussed. The test was repeated at several signal frequencies. Although a signal of this type should be, in reality, sampled continuously to achieve minimum storage, the test did provide an indication of the tradeoff between accuracy and storage.

Sampling accuracy was varied by varying the minimum reportable change, C. The sampling rates were set at the minimum rate possible that produced no range errors. Therefore, every sample taken was stored and the sampling rate equaled the storage rate. The storage reduction percentage proved to be virtually independent of the signal frequency. Table VII shows the storage percentage reduction for the difference methods tested.

Sampling Delays

To insure reducibility it is important to correlate the samples from a given channel with time. For ease of programming, as well as analysis, the time between samples should be equal. When sampling multiple channels, each with a different sampling rate, it is difficult to maintain equal time between all samples for all channels.

TABLE VII

DATA REDUCTION FOR INCREASED MINIMUM REPORTABLE CHANGE, C

Method	Minimum Reportable Change, C			
	2	3	4	5
Delta Change Storage Method	63%	75%	83%	86%
Variable Change Storage Method	57%	69%	77%	82%
Modified Variable Change Storage Method	63%	75%	83%	86%

A collision was defined as the occurrence of multiple channel sampling requests. For analog channels the collisions caused the nth channel's samples to be delayed in time by the amount $(n-1)(T_c+T_s)$, where T_s was the time to determine the next channel and start its conversion, and T_c was the A/D conversion time. For digital channels the delay was $(d-1)(T_d)$, where d was the number of digital channels and T_d the time to determine the next channel and read its value.

Sampling "jitter" was defined as a fluctuation in the sampling interval. Two sources of jitter were observed. The first was a one to five microsecond jitter due to the random nature of interrupt request. This amount of jitter was insignificant when compared to the slow varying signals being sampled. The second source of jitter was the

inconsistent occurrence of collisions. For this case, the jitter was equal to the delay due to the collision.

As shown in Figure 12, no sampling jitter was observed when collisions occurred consistently at each sample interval. In Figure 13, the jitter for this case was eliminated by sampling the faster channels first. Figure 14 shows that no jitter occurs for the case where successive channel sampling intervals were integer multiples of their predecessors. Figure 15 shows that jitter occurred, however, when all channel intervals were not integer multiples of each other.

For the two-channel case, the repetition interval for channels with sampling intervals n_1T and n_2T was $(n_1n_2)T$, where T was the basic system time interval, $n_1<n_2$, and n_2 was not an integer multiple of n_1. A theoretical worst-case jitter of approximately $19(T_s+T_c)$ would occur if all 20 channels (analog and digital) had sampling intervals that were not integer multiples of each other. For an observed (T_s+T_c) of approximately 300 microseconds, the theoretical worst-case delay is approximately 5.7 milliseconds. Sampling intervals which are not multiples of each other imply that they be prime numbers (say P_0 through P_{19}) times the basic sampling interval, T. The repetition interval for such a jitter would be $(P_1 \cdot P_2 \cdot P_3 \cdots P_{19})T$.

This theoretical worst-case jitter delay is presented here to show that even the worst-case jitter possible is only approximately 11% of a 50 millisecond system sampling

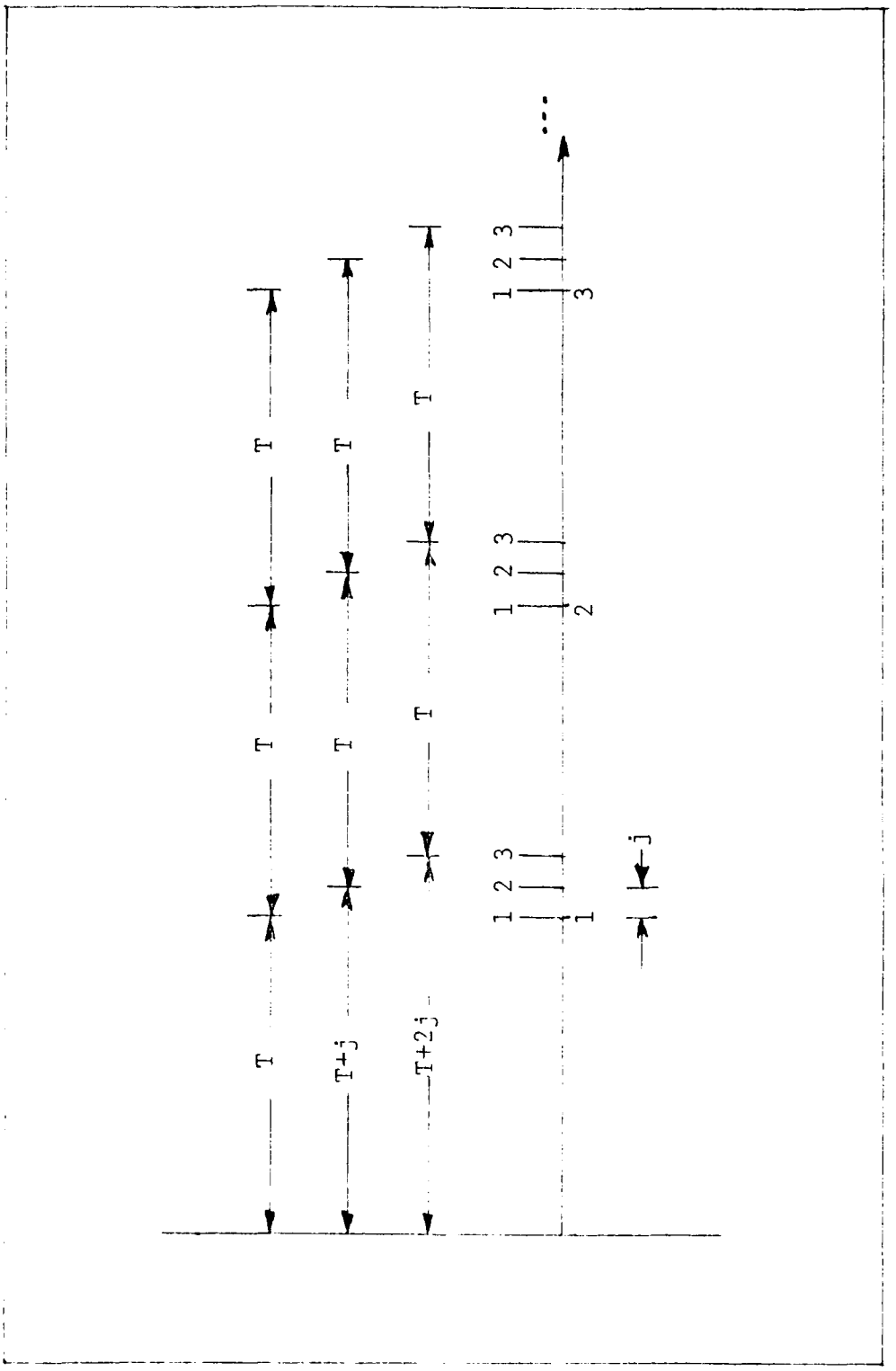


Fig. 12. Consistent Collisions Eliminate Jitter

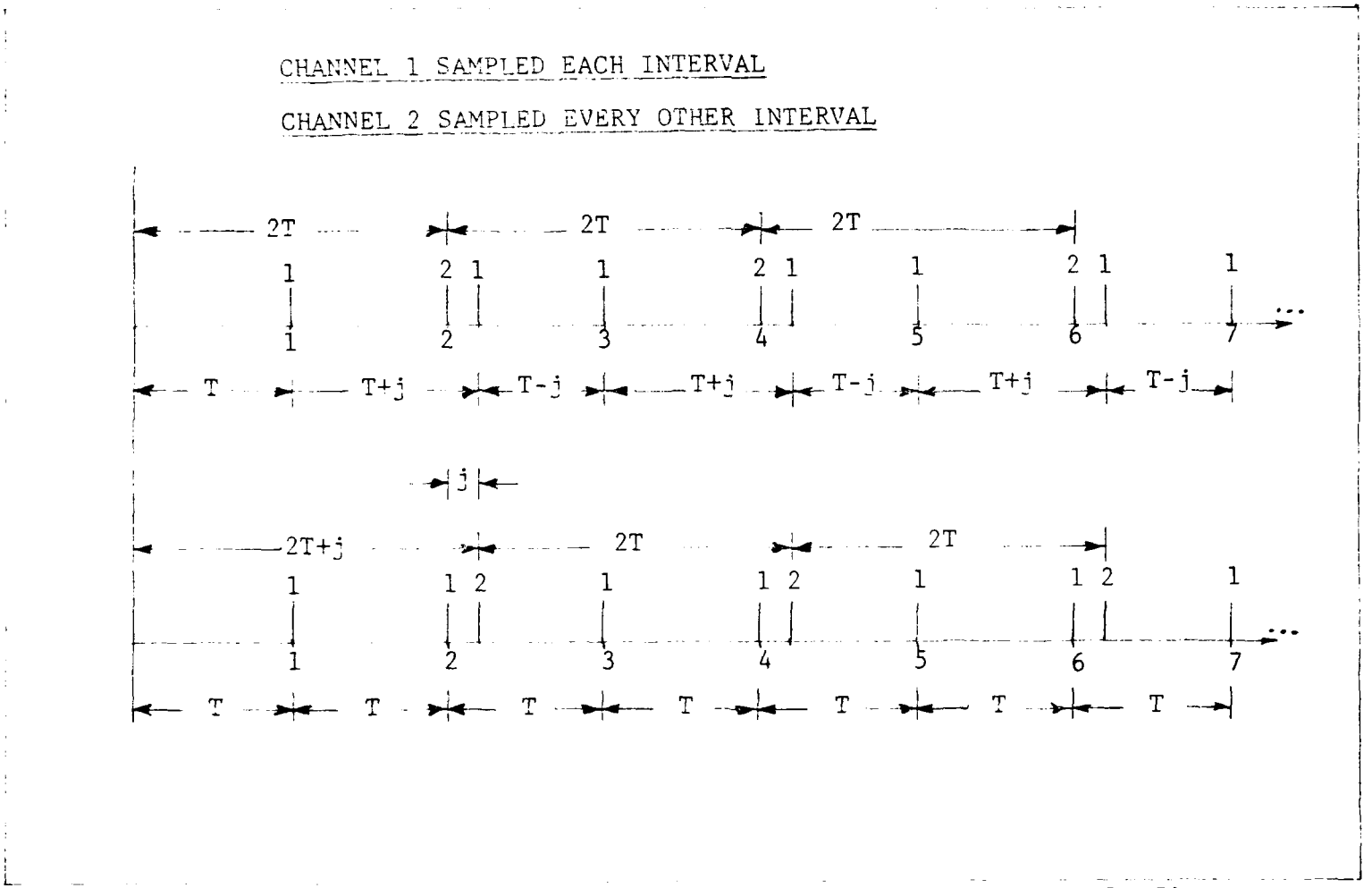


Fig. 13. Jitter Elimination by Sampling the Fastest Channels First

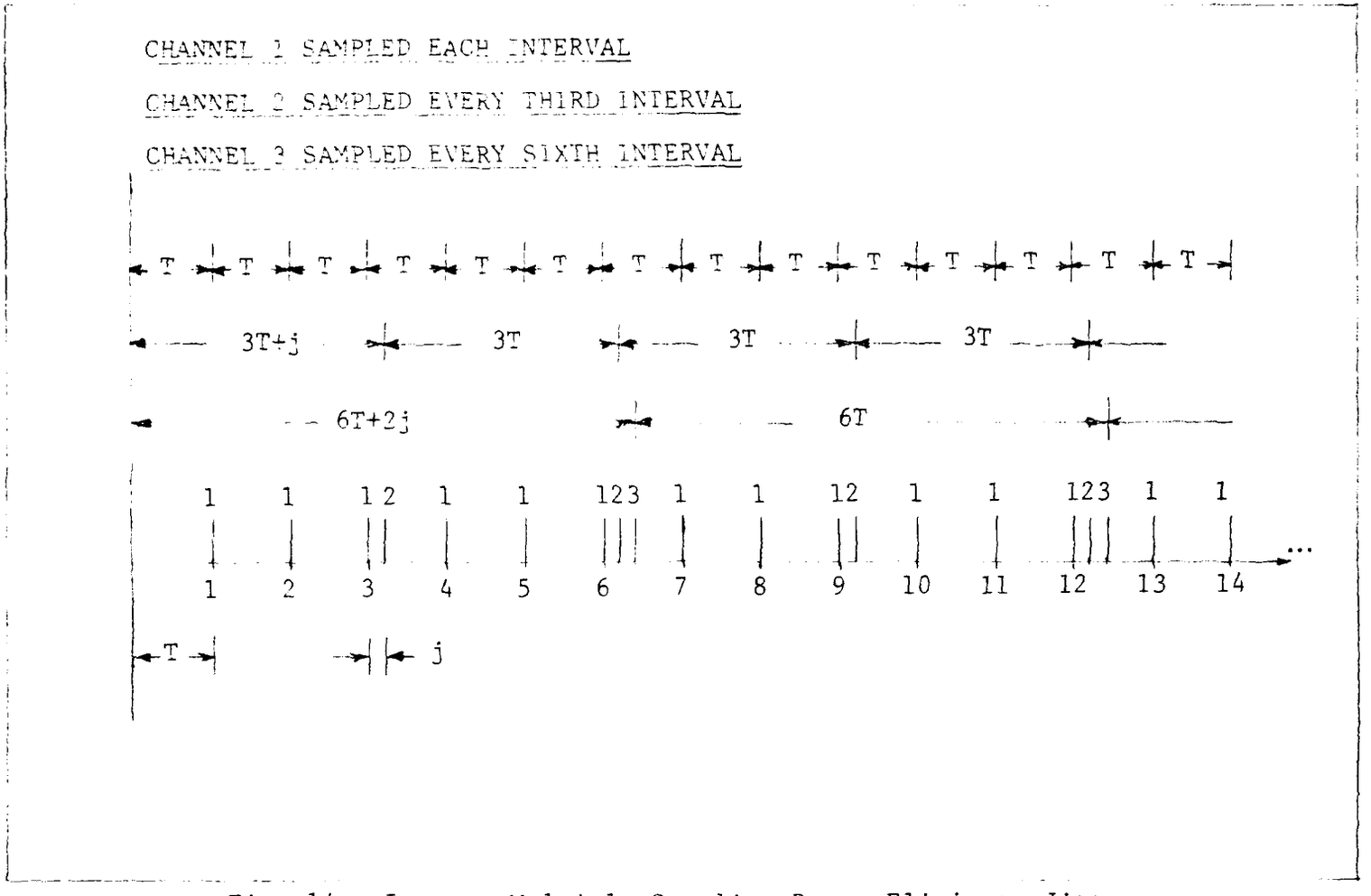


Fig. 14. Integer Multiple Sampling Rates Eliminate Jitter

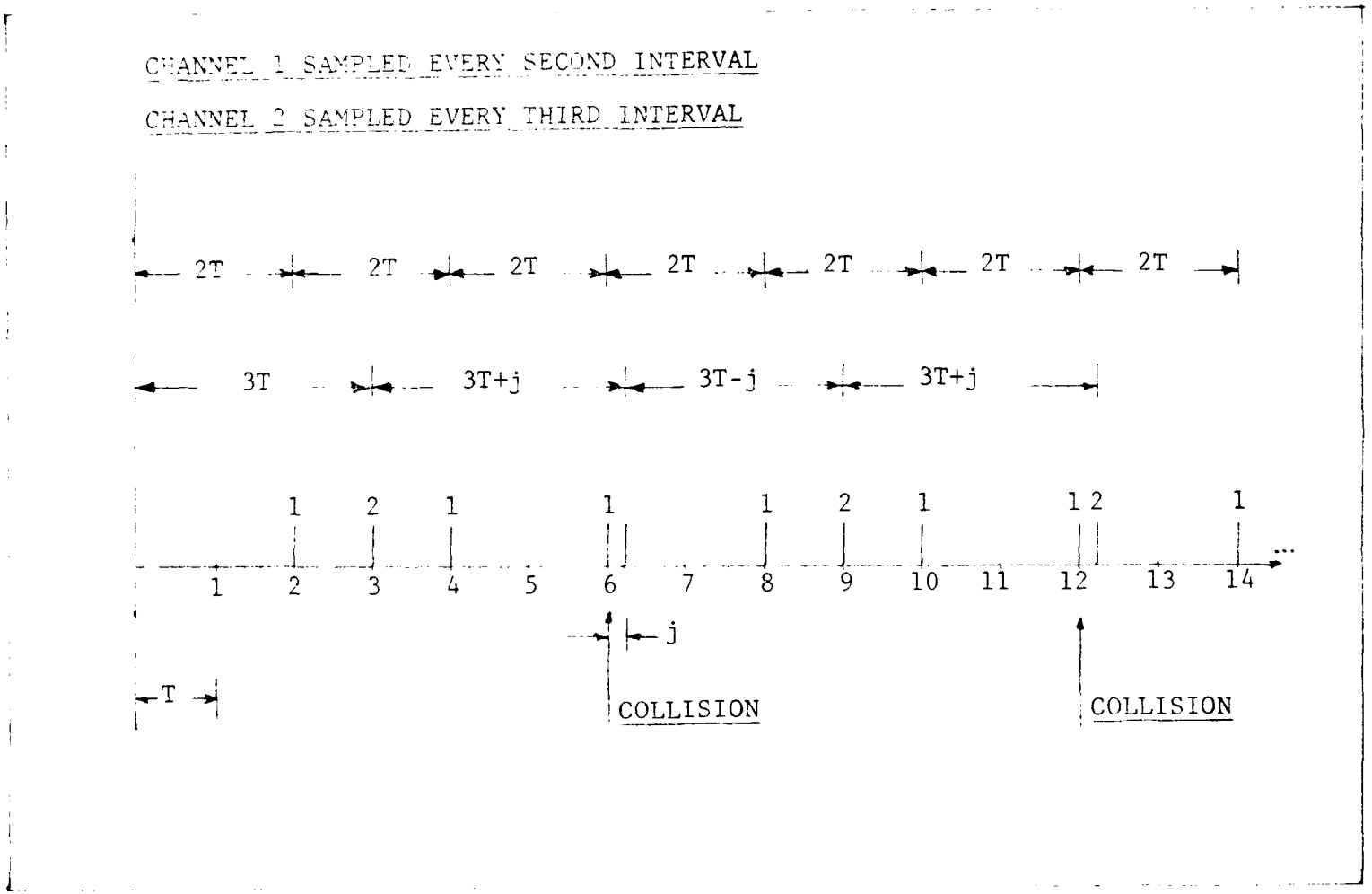


Fig. 15. Jitter Caused by Inconsistent Collisions

interval. With the frequency of the worst-case jitter being $(P_1 \cdot P_2 \cdot P_3 \ldots P_{19}T)$, the mission would run at least $(7.8 \times 10^{24})T$ for the worst-case jitter to occur. For T=50 milliseconds, a worst-case jitter would occur approximately every 1.24×10^{16} years! Therefore, it is very unlikely that the worst-case jitter would ever occur.

For the sampling rates suggested in Table I, assuming fastest channels are sampled first, the worst-case jitter would be ±1.5 milliseconds for the parameters sampled at eight samples per second. This is 1.2% of the .125 second interval, and occurs at every sample.

If the sampling rate were just twice the frequency of the highest signal component, this timing jitter would produce a maximum possible error of 3.7%. The sampling frequency is higher than twice the highest signal; therefore, the error produced from the jitter is less than the 3.7% maximum. If this error were determined to be unacceptable, the sampling rate of the appropriate parameters could be increased from eight samples per second to ten samples per second. The sampling intervals would then be integer multiples of each other and, as previously shown, no jitter would occur.

IV Results and Recommendations

The proposed IFPDAS was designed using current state-of-the-art devices. The design consisted of an eight-bit microcomputer which controlled the flow of data from one of 16 analog or four digital channels to MBM storage. Although the controller design was simplistic, its capability was significantly beyond that required by the IFPDAS. This should allow this basic controller design to be used with future MBM devices at much higher sampling rates.

Operational software was designed and the system simulated on a Rockwell System-65 minicomputer augmented with two-megabits of MBM. This software could be easily transferred to a R6502-based IFPDAS prototype.

It was evident from the start that any design using existing MBM devices would be storage-limited. The controller portion of the IFPDAS was designed with the minimum amount of hardware possible. This, with judicial device selection, insured the minimum amount of power and physical space for the controller hardware and the maximum remaining power and space for MBM storage. Even so, the IFPDAS design using eight, one-megabit Intel MBMs could only support the storage rate of the 12 original parameters. Greater storage rates were possible, but only at the expense of mission duration. This is shown graphically in Figure 16. The

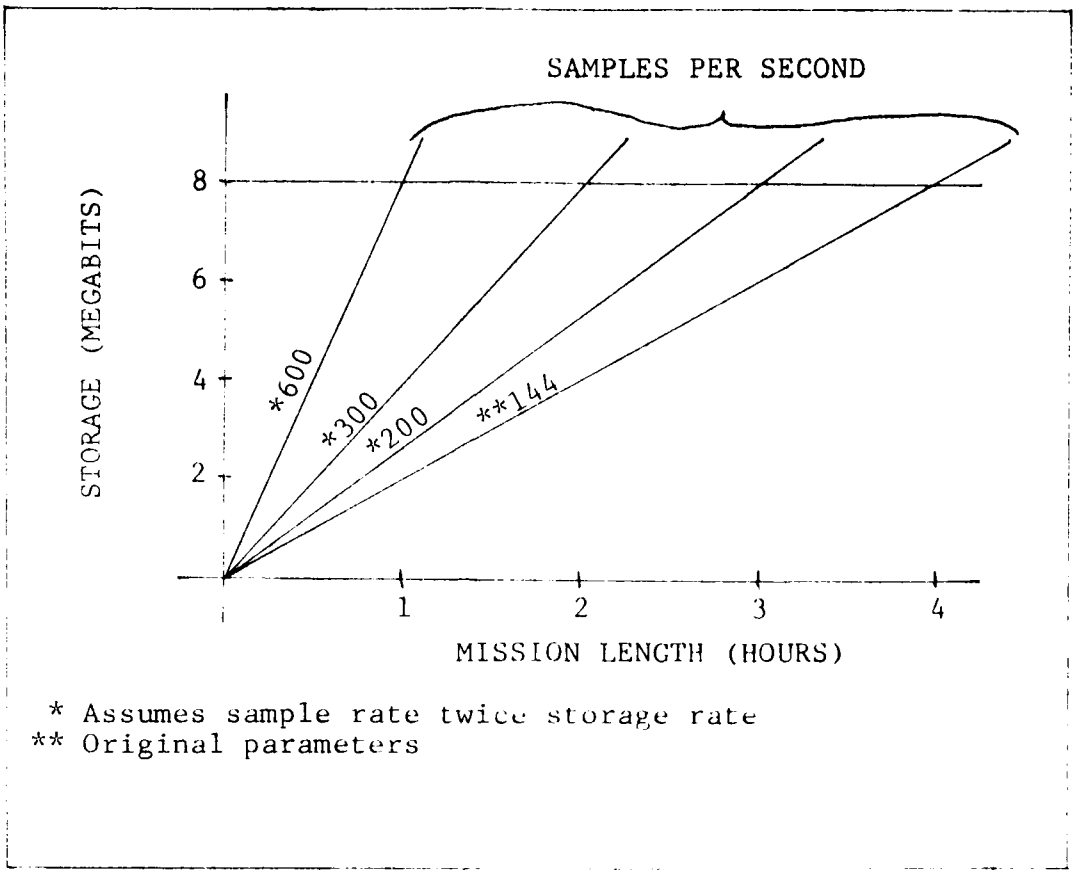


Fig. 16. Mission Length Versus Storage Required

eight-megabit line represents the maximum amount of MBM that will fit into a 2x5x9 inch IPFDAS using existing state-of-the-art devices. As the sample rate (and, therefore, the storage rate) increases, the eight-megabit line is crossed at shorter mission lengths.

Although the device chosen to convert analog signals did not have an internal sample-and-hold, one could be added if required. Because of the slow-varying signals and the relatively fast A/D conversion time, the possible error due to the signal changing while being converted was insignificant when compared to probe accuracy. If signals of

higher frequency should be sampled in the future, the decision to omit the SAH should be reconsidered.

The RAM buffer memory allowed the MBM to be powered down when not in use. At the slow IFPDAS sampling rates, the percent of MBM "on" time was independent of the amount of RAM buffer available. Size selection for RAM (system and buffer) was, therefore, based mainly on power and space requirements.

The four data storage methods used were:

a. Continuous Storage Method

b. Delta Continuous Storage Method

c. Variable Change Storage Method

d. Modified Variable Change Storage Method

The last three reduced the amount of storage by saving the difference between values rather than the values themselves, since the difference could be stored in fewer bits. Also, the last two methods stored differences only if they were larger than a predetermined value.

If the difference was larger than the bits could represent, a range error occurred and the current data, plus all subsequent data in the block, were incorrect. Two ways to prevent range errors were to increase the sampling rate or to increase the minimum reportable change (or minimum difference value). For signals which had wide variations from sample to sample, increasing the sampling rate required, in some cases, more memory storage than the Continuous Storage Method would have required. For this

reason, parameters should be matched to storage methods. Increasing the minimum reportable change significantly reduced the amount of storage memory required, but did so at the expense of data accuracy. Choosing the largest value for the minimum reportable change that can possibly be tolerated is the easiest, most straightforward way to reduce storage.

Several methods are suggested to handle range errors when they occur. If memory storage is at a premium, the block containing the range error can be discarded. If accurate, continuous samples are important, the remaining block can be zero-filled and a new block started. If some error can be tolerated, the maximum difference can be stored until the difference saved again correctly represents the true value. Lastly, sampling can be adaptively adjusted throughout the mission, increasing or decreasing sampling rates or minimum reportable change and changing the storage method used.

Consistent sampling intervals are important for reproducibility, as well as signal analysis. Inconsistent sampling intervals or "jitter" are caused by inconsistent, simultaneous sampling requests. For the worst case possible, the jitter was approximately 11% of the sampling interval. The jitter which can be expected during normal sampling would be less and is not considered significant. However, jitter can be totally eliminated by sampling faster channels first and making each channel's sampling

rate an integer multiple of the previous channel's sampling rate.

It is evident that this basic, simplistic design is very versatile and could be used for other related types of applications. One in particular is the collection of parachute drop data (Ref 18).

It is recommended that an IFPDAS prototype be built using the components specified in Chapter II or their functional equivalents. This construction would identify the layout and interface problems of packaging MBM alluded to by MBM manufacturers (Ref 10:49). Also, since it is reasonable to expect that the next generation MBMs will be bus compatible with existing MBMs, an IFPDAS with increased speed and storage capabilities will be more easily realized.

Bibliography

1. Merrifield, John T., et al. An Inflight Physiological Data Acquisition and Analysis System. Crew Technology Division, USAF School of Aerospace Medicine (USAFSAM), Brooks AFB, Texas.

2. Jolda, Joseph G. and Stephen J. Wanzek. Aircrew Inflight Physiological Data Acquisition System II. MS Thesis. Wright-Patterson AFB, Ohio: School of Engineering, Air Force Institute of Technology, December 1977. (AD A055638)

3. Hill, Robert E. Aircrew Modularized Inflight Data Acquisition System. MS Thesis. Wright-Patterson AFB, Ohio: School of Engineering, Air Force Institute of Technology, December 1978. (AFIT/GE/EE/78-28)

4. ADC0817. National Semiconductor Data Handbook, 1977.

5. RCA COS/MOS Memories, Microprocessors, and Support Systems. Summerville, NJ, 1979.

6. The MOS Memory Data Book. Houston: Texas Instruments, Inc., 1980.

7. The Complete Motorola Microcomputer Data Library. Chap. 1, 197. Phoenix: Motorola Semiconductor Products, Inc., 1978.

8. Rockwell Control Module (RCM650) System 65 Bubble Memory System. Product description. Anaheim: Rockwell International Corp., March 29, 1979.

9. Rockwell Linear Module - RLM6580 - 256Kx4 Bubble Memory Storage. Product specification. Anaheim: Rockwell International Corp., February 8, 1979.

10. Intel Magnetics Bubble Memory Design Handbook. Santa Clara: Intel Magnetics, Inc., 1979.

11. Bobeck, A. H., et al. "Current-Access Magnetic Bubble Circuits." The Bell System Technical Journal. American Telephone and Telegraph Company: 1453-1541 (July-August 1979).

12. Rockwell R6500 Microcomputer System-65 User's Manual. Document No. 29650N35. Anaheim: Rockwell International Corp., Rev. 2, January 1978.

13. Logic Diagram of the RCM650. Bubble memory application note #2, Rev. A. Anaheim: Rockwell International Corp., November 1, 1978.

14. R6500 Microcomputer System Hardware Manual. Rockwell International Corp. USA, revised August 1978.

15. Meyers, Glenford J. Software Reliability Principles and Practices. New York: John Wiley and Sons, Inc., 1976.

16. Simpson, Richard S. and Ronald C. Houts. Fundamentals of Analog and Digital Communications Systems. Boston: Allyn and Bacon, Inc., 1971.

17. Taub, Herbert and Donald L. Schilling. Principles of Communication Systems. New York: McGraw-Hill Book Company, 1971.

18. Richardson, Gary A. Evaluation of an Improved Parachute Test Data Acquisition System. MS Thesis. Wright-Patterson AFB, Ohio: School of Engineering, Air Force Institute of Technology, December 1979. (AFIT/GCS/79-13)

19. Intersil Data Acquisition Handbook. Cupertino, California: Intersil, Inc., 1980.

20. Garnett, Patrick H. Analog Systems for Microprocessors and Minicomputers. Reston, Virginia: Reston Publishing Company, Inc., 1978.

21. Tocci, Ronald J. and Lester P. Laskowski. Microprocessors and Microcomputers Hardware and Software. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1979.

22. Wollensen, Donald L. "CMOS LSI - The Computer Component Process of the 80's." Computer: 59-67 (February 1980).

23. Capece, Raymond P. "Bubble Offerings Start to Balloon." Electronics: 41-42 (March 13, 1980).

24. Chang, Hsu. "Major Activity in Magnetic Bubble Technology." Computer Design: 117-125 (November 1979).

25. Chang, Hsu. Magnetic Bubble Memory Technology. New York: Marcel Dekker, Inc., 1978.

26. Rabiner, L. R. and R. W. Schafer. Digital Processing of Speech Signals. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1978.

27. Spieger, Phil. "Getting Started in Bubble Memories." Digital Design: 46-50 (November 1979).

Appendix A

IFPDAS PROGRAM LISTING

```
LINE # LOC     CODE        LINE

       0000                ;********** PAGE ZERO VERIABLES  **********
       0000                        *=$0006

       0006    FF          NCHNLS .BYT $FF         ;NUMBER OF ACTIVE CHANNELS - 1
       0007    FF          NPORTS .BYT $FF         ;NUMBER OF ACTIVE PORTS - 1

       0008    00          ADBUSY .BYT $0          ;A/D BQSY FLAG, 0-3 = INDEX, 7 = BUSY
       0009                BUBNDX *=*+1            ;CURRENT BUFFER TO BUBBLE INDEX POINTER
       000A                CURBFA *=*+2            ;CURRENT BUFFER ADDRESS
       000C    00          BBUSY  .BYT $0          ;BUBBLE BUSY FLAG, 0-3 = INDEX, 4 = CHAN/PORT
       000D                ; 6 = BUBBLE POWERED FLAG, 7 = BUBBLE BUSY
       000D                NLINK  *=*+1            ;NUMBER OF BLOCKS IN LINK
       000E                TLINK  *=*+2            ;'TOP OF LINK' POINTER
       0010                NEWBLK *=*+2            ;ADDRESS OF BLOCK JUST DE-LINKED
       0012                OLDBLK *=*+2            ;ADDRESS OF BLOCK TO BE LINKED

       0014                CHANLS *=*+16           ;LIST ACTIVE CHAN, 0-3 = CHAN, 4-5 = METHOD,
       0024                                  ; 6 = BUBBLE REQUEST,  7 = A/D REQUEST
       0024                COUNTV *=*+16           ;VARIABLE TIMER COUNTER
       0034                POINTR *=*+2            ;USED IN 'SAVE' FOR INDIRECT ADD.
       0036                CBKPTR *=*+16           ;CHANNEL BLOCK POINTER
       0046                KEPNDX *=*+1            ;SAVE INDEX VALUE
       0047                VDIFF  *=*+1            ;A/D - THIS VALUE/LAST VALUE DIFFERENCE
       0048                SDIFF  *=*+1
       0049                MDIFF  *=*+1
       004A                COUNTT *=*+16           ; # TIME INTERVALS SINCE VALUE SAVED
       005A                CBFADD *=*+32           ;CHANNEL BUFFAR ADDRESS
       007A                CCHBIK *=*+32           ;CHANNEL BUFFER INPUT POINTERS
       009A                LSTVAL *=*+16           ;LAST A/D VALUE
       00AA                LSTTIM *=*+32           ;LAST A/D TIME
       00CA                THSVAL *=*+16           ;THIS VALUE OF A/D CONVERSION
       00DA                THSTIM *=*+32           ;TIME OF THIS A/D CONVERSION
       00FA                PORTS  *=*+4            ;LIST OF ACTIVE PORTS - BIT 7 = NEED BUBBLE SERV

       00FE                ACURCY *=*+1


       00FF                ;******** INTERVAL TIMER ADDRESS DEFINITIONS ********

       00FF                TCTL2  = $AFE9          ;WRITE CONTROL REG2
       00FF                ;READ STATUS REG
       00FF                TCTL13 = $AFE8          ;CTL2 BIT      --> WRITE CTL3
       00FF                ;              CTL2 BIT 0 = [illegible] TE CTL1
       00FF                ;              READ --> NC [illegible]
       00FF                T1CNTR = $AFEA          ;RE[illegible]MER 1 COUNTER
       00FF                ;              WRITE MSB BUFFER REG
       00FF                T1LCHW = $AFEB          ;WRITE TIMER 1 LATCH
       00FF                ;                READ LSB BUFFER REG
       00FF                T2CNTR = $AFEC          ;READ TIMER 2 COUNTER
       00FF                ;              WRITE MSB BUFFER REG
       00FF                T2LCHW = $AFAD          ;WRITE TIMER 2 LATCH
       00FF                ;              READ LSB BUFFER REG
       00FF                T3CNTR = $AFEE          ;READ TIMER 3 COUNTER
       00FF                ;                WRITE MSB BUFFER REG
       00FF                T3LCHW = $AFEF          ;WRITE TIMER 3 LATCH
```

```
LINE # LOC     CODE       LINE

0056  00FF              ;                READ LSB BUFFER REG


0058  00FF              ;******  VIA ADDRESS DEFINITIONS  ******

0060  00FF              PORTA  = $AFF1          ;DEFINE A/D DATA ADDRESS
0061  00FF              DDRA   = $AFF3          ;PORT A DATA DIRECTION REGISTER
0062  00FF              AUXCIL = $AFFB          ;AUXCILARY CONTROL REGISTER

0064  00FF              PORTB  = $AFF0          ;ADDRESS OF CHANNEL SELECT (0-F)
0065  00FF              DDRB   = $AFF2          ;PORT B DATA DIRECTION REGISTER

0067  00FF              T1LL   = $AFF6          ; R/W TIMER 1 LOW-LATCH
0068  00FF              T1HL   = $AFF7          ; R-TIMER 1 HIGH LATCH
0069  00FF              ;W--TIMER 1 HIGH LATCH - RESET IRQ FLAG
0070  00FF              T1LC   = $AFF4          ; R--TIMER 1 LOW COUNTER - RESET IRQ FLAG
0071  00FF              ; W--TIMER 1 LOW LATCH
0072  00FF              T1HC   = $AFF5          ; R--TIMER 1 HIGH COUNTER
0073  00FF              ; W--TIMER 1 HIGH COUNTER
0074  00FF              ;    TIMER 1 HIGH LATCH
0075  00FF              ;    TIMER 1 LOW LATCH --> LOW COUNTER
0076  00FF              ;    START NEW TIME INTERVAL - RESET IRQ FLAG

0078  00FF              T2LL   = $AFF8          ; R--TIMER 2 LOW COUNTER - RESET IRQ FLAG
0079  00FF              ; W--TIMER 2 LOW LATCH
0080  00FF              T2HC   = $AFF9          ; R--TIMER 2 HIGH COUNTER
0081  00FF              ; W-- TIMER HIGH COUNTER
0082  00FF              ;     TIMER 2 LOW LATCH --> TIMER 2 LOW COUNTER
0083  00FF              ;     TIMER 2 RESTART COUNT

0085  00FF              SHFTRG = $AFEA          ;SHIFT REGISTER ADDRESS
0086  00FF              PCR    = $AFEC          ;PERIPHERAL CONTROL REGISTER
0087  00FF              IFR    = $AFED          ;INTERRUPT FLAG REGISTER
0088  00FF              IER    = $AFEE          ;INTERRUPT ENABLE REGISTER

0090  00FF              ;******  MONITOR LINKS  ******

0092  00FF              ACIA   = $C000
0093  00FF              MSGADR = $C606
0094  00FF              MONITR = $C9E9
0095  00FF              BLANK  = $D0AF
0096  00FF              CRLOW  = $D0F1
0097  00FF              RKEP   = $D139
0098  00FF              OUTPUT = $D2C1
0099  00FF              REDOUT = $D2B0
0100  00FF              HEX    = $D306
0101  00FF              NUMA   = $D2C6
0102  00FF              READ   =$D11C
0103  00FF              LEFT   =$D350
0104  00FF              RCHEK  = $D1BC          ;READ & SEE IF KEYBOARD HIT


0107  00FF              PAIN   = $B7AF          ;SET UP PA BUBBLE PORT AS INPUTS
0108  00FF              PAOUT  = $B7AB          ;SET PA BUBBLE PORT AS OUTPUTS
```

```
LINE # LOC     CODE      LINE

0109  00FF               PBIN   = $B7C6      ;SET PB BUBBLE PORT AS INPUTS
0110  00FF               PBOUT  = $B7C2      ;SET PB BUBBLE PORT AS OUTPUTS
0111  00FF               WAITB  = $B73D      ;WAIT FOR SYSTEM BUBBLE NOT BUSY
0112  00FF               SEND   = $B7DC      ;SEND A COMMAND TO SYSTEM BUBBLE AND WAIT FOR A
0113  00FF               PBIO   = $B7C8      ;B PORT INPUT OR OUTPUT ACCORDING TO 'X'
0114  00FF               CRA    = $B801      ;SYSTEM BUBBLE CONTROL REGISTER A
0115  00FF               PA     = $B800      ;BUBBLE PORT A
0116  00FF               PB     = $B802      ;BUBBLE PORT B

0118  00FF               BUFI   = $BC00      ;INPUT BUFFAR OF 256 LOCATIONS
0119  00FF               BUFO   = $BC00      ;OUTPUT BUFFER OF 256 LOCATIONS
```

```
LINE # LOC      CODE        LINE

0121  00FF                ;********** THIS IS THE START OF THE PROGRAM **********
0122  00FF                ;********** PROGRAM STARTS WITH KEY6        **********

0124  00FF                      *=$C411
0125  C411  4C 00 02            JMP RESET       ;SET KEY6 TO START PROGRAM

0127  C414                      *=$200
0128  0200  78           RESET  SEI             ;INITIALIZE SYSTEM AFTER RESET
0129  0201  D8                  CLD
0130  0202  A2 FF               LDX #$FF
0131  0204  9A                  TXS             ;SET UP SYSTEM STACK
0132  0205  20 DD 05            JSR INIT        ;DO SYSTEM INITIALIZATION
0133  0208  20 EC 05            JSR SLFTST      ;DO SYSTEM SELFTEST

0135  020B  A9 00        SETUP  LDA #<MSG1      ; INITIALIZES RUN PARAMETERS OR DUMPS BUBBL
0136  020D  20 ED 05            JSR MSGOUT      ;OUTPUT MSG1 TO CRT
0137  0210  20 FC 05            JSR GETVAL      ;GET A CHERACTER FROM THE CRT
0138  0213  C9 44               CMP #'D
0139  0215  D0 03               BNE SET2
0140  0217  4C CD 05            JMP DUMP
0141  021A  C9 49        SET2   CMP #'I
0142  021C  D0 ED               BNE SETUP       ;IF NOT 'D' OR 'I' THEN ASK AGAIN
0143  021E  20 41 06            JSR BUBLE0      ;ZERO BUFFER INDEX
0144  0221  20 03 06      SET3  JSR CRLF        ;OUTPUT A CARRAGE/RETURN TO CRT

0146  0224  A9 1C        CHNPRM LDA #<MSG3      ;'ENTER'
0147  0226  20 ED 05            JSR MSGOUT      ; 'CHAN     RATE     METHOD'
0148  0229  A9 22               LDA #<MSG3A
0149  022B  20 ED 05            JSR MSGOUT
0150  022E  20 03 06            JSR CRLF
0151  0231  20 FC 05      CHN1  JSR GETVAL      ;GET ACTIVE CHANNEL VALUE
0152  0234  C9 0D               CMP #$0D        ;TEST FOR CR
0153  0236  F0 24               BEQ PRTPRM      ;BRANCH TO GET PORT PARAMETER
0154  0238  20 0D 06            JSR TOHEX       ;CHANGE 'A' TO A HEX VALUE
0155  023B  29 0F               AND #$0F
0156  023D  AA                  TAX             ;SET UP INDEX 'X'
0157  023E  A9 09               LDA #9
0158  0240  20 11 06            JSR SPACES      ;OUTPUT 'A' SPACES TO CRT
0159  0243  20 1F 06            JSR CHEX2       ;READ IN FROM CRT TWO CHARACTERS --> HEX IN 'A'
0160  0246  9D 9D 11            STA CRATE,X     ;SAVE THIS RATE VALUE
0161  0249  A9 0A               LDA #10
0162  024B  20 11 06            JSR SPACES
0163  024E  20 FC 05            JSR GETVAL
0164  0251  29 07               AND #7          ;FORCE METHOD INTO RANGE
0165  0253  9D 7D 11            STA STORGE,X    ;GET AND SAVE CHANNEL SAMPLING METHOD
0166  0256  20 03 06            JSR CRLF
0167  0259  4C 31 02            JMP CHN1        ;GET ANOTHER CHANNEL PARAMETER SET

0169  025C  A9 1C        PRTPRM LDA #<MSG3
0170  025E  20 ED 05            JSR MSGOUT      ;'ENTER'
0171  0261  A9 3C               LDA #<MSG4
0172  0263  20 ED 05            JSR MSGOUT      ; 'ACTIVE PORT     RATE'
```

```
LINE # LOC     CODE        LINE

0173  0266  20 03 06          JSR CRLF
0174  0269  20 FC 05    PRT1  JSR GETVAL
0175  026C  C9 0D             CMP #$0D
0176  026E  F0 14             BEQ MOK1
0177  0270  29 03             AND #$3
0178  0272  AA                TAX             ;SCALE PORT INDEX VALUE
0179  0273  A9 0A             LDA #10
0180  0275  20 11 06          JSR SPACES
0181  0278  20 1F 06          JSR CHEX2       ;READ CRT TWICE --> HEX 'A'
0182  027B  9D 59 11          STA PORTBF,X    ;SAVE PORT RATE
0183  027E  20 03 06          JSR CRLF
0184  0281  4C 69 02          JMP PRT1        ;GO GET ANOTHER PORT PARAMETER SET


0186  0284  20 03 06    MOK1  JSR CRLF
0187  0287  20 00 05          JSR ACTIV       ;SORT ACTIVE ANALOG & DIGITAL CHANNELS
0188  028A  A9 99             LDA #<MSG8
0189  028C  20 ED 05          JSR MSGOUT
0190  028F  A9 22             LDA #<MSG3A
0191  0291  20 ED 05          JSR MSGOUT
0192  0294  A6 06             LDX NCHNLS
0193  0296  30 23             BMI MOK6
0194  0298  20 03 06    MOK2  JSR CRLF        ;LIST ACTIVE CHAN'S
0195  029B  B5 14             LDA CHANLS,X
0196  029D  29 0F             AND #$0F
0197  029F  20 3F 06          JSR OUTHEX
0198  02A2  A9 08             LDA #8
0199  02A4  20 11 06          JSR SPACES
0200  02A7  BD BC 10          LDA COUNTP,X
0201  02AA  20 3F 06          JSR OUTHEX
0202  02AD  A9 08             LDA #8
0203  02AF  20 11 06          JSR SPACES
0204  02B2  BD 6D 11          LDA METHOD,X
0205  02B5  20 3F 06          JSR OUTHEX
0206  02B8  CA                DEX
0207  02B9  10 DD             BPL MOK2

0209  02BB  20 03 06    MOK6  JSR CRLF        ;LIST ACTIVE PORTS
0210  02BE  A9 99             LDA #<MSG8
0211  02C0  20 ED 05          JSR MSGOUT
0212  02C3  A9 3C             LDA #<MSG4
0213  02C5  20 ED 05          JSR MSGOUT
0214  02C8  A6 07             LDX NPORTS
0215  02CA  30 18             BMI MOK5
0216  02CC  20 03 06    MOK3  JSR CRLF
0217  02CF  B5 FA             LDA PORTS,X
0218  02D1  29 03             AND #3          ;FORCE INTO RANGE
0219  02D3  20 3F 06          JSR OUTHEX
0220  02D6  A9 08             LDA #8
0221  02D8  20 11 06          JSR SPACES
0222  02DB  BD 59 11          LDA PRATE,X
0223  02DE  20 3F 06          JSR OUTHEX
0224  02E1  CA                DEX
0225  02E2  10 E8             BPL MOK3
```

```
LINE # LOC      CODE        LINE

0227  02E4  A9 4B       MOK5   LDA #<MSG5
0228  02E6  20 03 06            JSR CRLF
0229  02E9  20 ED 05            JSR MSGOUT      ;'VERIFY-V   CHANGE-C    OK-K'
0230  02EC  20 FC 05            JSR GETVAL
0231  02EF  C9 56               CMP #'V
0232  02F1  F0 91               BEQ MOK1
0233  02F3  C9 43               CMP #'C
0234  02F5  D0 03               BNE MOK4
0235  02F7  4C 21 02            JMP SET3
0236  02FA  C9 4B       MOK4   CMP #'K
0237  02FC  D0 E6               BNE MOK5

0239  02FE  A9 6F               LDA #<MSG6
0240  0300  20 ED 05            JSR MSGOUT      ;'WAIT - BUBBLE INITIALIZATION'
0241  0303  20 70 06            JSR WRBHDR      ;WRITE HEADER INFORMATION TO BUBBLE
0242  0306  A9 82               LDA #<MSG7      ;'INITIALIZATION COMPLETE - POWER DOWN SYSTEM'
0243  0308  20 ED 05            JSR MSGOUT
0244  030B  20 DC 01    MOK7    JSR READ        ;WAIT FOR 'SPACE' BEFORE MISSION
0245  030E  C9 20               CMP #$20
0246  0310  D0 F9               BNE MOK7
0247  0312  4C 15 03            JMP MISSN
```

```
LINE #  LOC   CODE       LINE

0250  0315  20 71 06   MISSN  JSR RDBHDR      ;READ BUBBLE HEADER FOR RUN PARAMETERS
0251  0318  20 B0 06          JSR RUNIT       ;INITIALIZE POINTERS, COUNTERS, TABLES, ETC.
0252  031B  A6 06             LDX NCHNLS      ;GET CHANNEL INDEX
0253  031D  30 4B      MISSN1 BMI MISSN2
0254  031F  8E 34 11          STX SAVEX
0255  0322  A9 00             LDA #0
0256  0324  95 36             STA CBKPTR,X    ;ZERO ANALOG CHAN OFFSET
0257  0326  95 4A             STA COUNT,X     ;ZERO DELTA TIME COUNT

0259  0328  20 72 06          JSR TIMERS
0260  032B  B5 14             LDA CHANLS,X    ;READ 1ST VALUE OF ANALOG CHANNEL 'X'
0261  032D  49 FF             EOR #$FF
0262  032F  8D B0 AF          STA PORTB       ;START A/D CONVERSION
0263  0332  A9 02      MISSN5 LDA #2
0264  0334  2C B0 AF          BIT IFR
0265  0337  D0 F9             BNE MISSN5
0266  0339  AD B1 AF          LDA PORTA
0267  033C  49 FF             EOR #$FF
0268  033E  95 CA             STA THISVAL,X   ;SET CURRENT VALUE
0269  0340  95 9A             STA LSTVAL,X

0271  0342  BD BC 10          LDA COUNTP,X
0272  0345  95 24             STA COUNTV,X    ;SET TIMER COUNTERS
0273  0347  20 A4 0A          JSR BLKALC      ;ALLOCATE 1ST BLOCK THIS ANALOG CHAN
0274  034A  8A                TXA
0275  034B  0A                ASL A
0276  034C  AA                TAX
0277  034D  A5 10             LDA NEWBLK
0278  034F  95 5A             STA CBFADD,X    ;SET UP 1ST BLOCK POINTERS
0279  0351  95 7A             STA CCBLK,X
0280  0353  A5 11             LDA NEWBLK+1
0281  0355  95 5B             STA CBFADD+1,X
0282  0357  95 7B             STA CCBLK+1,X
0283  0359  A9 00             LDA #0
0284  035B  95 DA             STA THISTIM,X   ;ZERO START TIME
0285  035D  95 DB             STA THISTIM+1,X
0286  035F  EE 34 11          INC SAVEX
0287  0362  20 63 0A          JSR WRHDR       ;WRITE CHANNEL HEADER TO RAM BUFFER AREA 'X'
0288  0365  94 36             STY CBKPTR,X    ;MODIFY INDEX POINTER
0289  0367  CA                DEX
0290  0368  10 B3             BPL MISSN1

0292  036A  A6 07      MISSN2 LDX NPORTS
0293  036C  30 28      MISSN3 BMI MISSN4
0294  036E  A9 00             LDA #0
0295  0370  9D 51 11          STA PBKPTR,X    ;ZERO DIGITAL POINTER OFFSET
0296  0373  20 A4 0A          JSR BLKALC      ;GET 1ST BLOCK
0297  0376  8A                TXA
0298  0377  0A                ASL A
0299  0378  A8                TAY
0300  0379  A5 10             LDA NEWBLK      ;SET UT BLOCK POINTERS
0301  037B  99 49 11          STA CPBLK,Y
0302  037E  99 41 11          STA PBFADD,Y
0303  0381  A5 11             LDA NEWBLK+1
```

```
LINE # LOC     CODE       LINE

0304  0383  99 4A 11          STA CPBLK+1,Y
0305  0386  99 42 11          STA PBFADD+1,Y
0306  0389  8A                TXA
0307  038A  09 10             ORA #$10
0308  038C  8D 34 11          STA SAVEX
0309  038F  20 63 0A          JSR WRHDR       ;WRITE PORT HEADER TO RAM BUFFER AREA 'X'
0310  0392  98                TYA
0311  0393  9D 51 11          STA PBKPTR,X
0312  0396  CA                DEX
0313  0397  10 D3             BPL MISSN3

0315  0399  A9 00        MISSN4 LDA #0
0316  039B  85 0C              STA BUSY
0317  039D  20 72 06          JSR TIMERS      ;START TIMER FOR MISSION RUN
0318  03A0  58                CLI
```

```
LINE # LOC     CODE        LINE

0320  03A1                ;******** MAIN LOOP OF PROGRAM ********

0322  03A1  A9 20     MAIN  LDA #%00100000  ;CHECK FOR BUFFER 80% FULL
0323  03A3  24 0C           BIT BBUSY
0324  03A5  D0 0A           BNE MAIN6
0325  03A7  30 F8           BMI MAIN        ;BUSY?
0326  03A9  50 F6           BVC MAIN        ;POWERED?
0327  03AB  20 51 07        JSR BUURDN      ;YES
0328  03AE  4C A1 03        JMP MAIN        ;NO

0330  03B1  50 04     MAIN6 BVC MAIN1       ;BRANCH BUBBLE$NOT POWERED
0331  03B3  30 EC           BMI MAIN        ;BRANCH BUBBLE BUSY
0332  03B5  10 03           BPL MAIN2       ;ALWAYS SKIP NEXT INSTRUCTION
0333  03B7  20 9D 07  MAIN1 JSR BUPUP      ;GO POWER UP THE BUBBLE

0335  03BA  A9 E0     MAIN2 LDA #%11100000  ;SET POWERED, BUSY, & 80%
0336  03BC  85 0C           STA BBUSY
0337  03BE  A6 06           LDX NCHNLS      ;POINTERS TO TOP OF THIS LILKED LIST
0338  03C0  A5 0C     MAIN3 LDA BBUSY       ;UPDATE BBUSY FLAG
0339  03C2  29 E0           AND #$E0
0340  03C4  85 0C           STA BBUSY
0341  03C6  8A              TXA
0342  03C7  30 17           BMI MAIN4
0343  03C9  05 0C           ORA BBUSY
0344  03CB  85 0C           STA BBUSY
0345  03CD  8A              TXA
0346  03CE  0A              ASL A
0347  03CF  A8              TAY
0348  03D0  B9 5A 00        LDA CBFADD,Y
0349  03D3  85 0A           STA CURBFA
0350  03D5  B9 5B 00        LDA CBFADD+1,Y
0351  03D8  85 0B           STA CURBFA+1

0353  03DA  20 E0 0A        JSR BUBBLD      ;PUT ANALOG CHAN LINKED LIST IN BUBBLE

0355  03DD  CA              DEX
0356  03DE  10 E0           BPL MAIN3

0358  03E0  A6 07     MAIN4 LDX NPORTS      ;PUT PORT LINKED LIST IN BUBBLE
0359  03E2  A5 0C     MAIN5 LDA BBUSY
0360  03E4  29 E0           AND #$E0
0361  03E6  85 0C           STA BBUSY
0362  03E8  8A              TXA
0363  03E9  30 19           BMI MAIN7
0364  03EB  09 10           ORA #$10
0365  03ED  05 0C           ORA BBUSY
0366  03EF  85 0C           STA BBUSY
0367  03F1  8A              TXA
0368  03F2  0A              ASL A
0369  03F3  A8              TAY
0370  03F4  B9 41 11        LDA PBFADD,Y
0371  03F7  85 0A           STA CURBFA
0372  03F9  B9 42 11        LDA PBFADD+1,Y
0373  03FC  85 0B           STA CURBFA+1
```

```
LINE # LOC     CODE       LINE

0375  03FE  20 F0 0A          JSR BUBBLE      ;PUT PORT LINKED LIST IN BUBBLE

0377  0401  CA                DEX
0378  0402  10 DE             BPL MAIN5

0380  0404  A5 0C       MAIN7 LDA BBUSY       ;RESET BUBLLE BUSY FLAG
0381  0406  29 60             AND #%01100000
0382  0408  85 0C             STA BBUSY

0384  040A  AD 00 C0          LDA ACIA
0385  040D  29 01             AND #1
0386  040F  F0 90             BEQ MAIN
0387  0411  78                SEI
0388  0412  AD 01 C0          LDA ACIA+1
0389  0415  C9 1B             CMP #$1B
0390  0417  D0 03             BNE MAIN8
0391  0419  4C F0 C9          JMP MONITR
0392  041C                MAIN8                 ;INHIBIT INTERRUPTS
0393  041C  A9 1C             LDA #<MSG3
0394  041E  20 ED 05          JSR MSGOUT
0395  0421  AD 00 C0       MAIN9 LDA ACIA
0396  0424  29 01             AND #1
0397  0426  F0 F9             BEQ MAIN9
0398  0428  20 03 06          JSR CHLF
0399  042B  58                CLI             ;ENABLE INTERRUPTS
0400  042C  4C A1 03          JMP MAIN
0401  042F                ;     ESCAPE --> GO TO MONITOR
0402  042F                ;     SPACE --> STOP/START MISSION
```

```
LINE # LOC     CODE        LINE

0404  042F                ITIRQ                ;6840 INTERVAL TIMER IRQ HANDLER
0405  042F  AD E9 AF             LDA TCTL2      ;READ STATUS
0406  0432  49 FF                EOR #$FF
0407  0434  10 04                BPL ITIRQ1     ;NO IRQ HERE
0408  0436  6A                   ROR A
0409  0437  6A                   ROR A          ;CHECK TIMER 2
0410  0438  B0 02                BCS ITIMR2

0412  043A  68          ITIRQ1 PLA              ;RESTORE 'A'
0413  043B  40                   RTI



0415  043C  A9 01        ITIMR2 LDA #1          ;40 WORD TIMER IRQ
0416  043E  8D 3A 11             STA C40FLG
0417  0441  49 FF                EOR #$FF
0418  0443  8D E8 AF             STA TCTL13
0419  0446  68                   PLA
0420  0447  40                   RTI



0422  0448                DIGIRQ               ;THE DIGITAL IRQ HANDLER GOES HERE.
0423  0448                                ;IT WILL CHECK WHICH DIGITAL CHANNEL
0424  0448                                ;CAUSED THE IRQ, READ, AND SAVE THE DATA

0426  0448  4C 2F 04             JMP ITIRQ      ;SEE IF INTERVAL TIMER IRQ
```

```
LINE # LOC      CODE        LINE

0428  044B                ;.*** REFERENCES MUST CONSIDER THAT DATA BUS IS INVERTED *

0430  044B  48            VIA1RQ PHA              ;INTERRUPT FROM THE VERSATILE INTERFACE ADAPTOR (
0431  044C  A9 20                 LDA #%00100000   ;SAVE 'A' THEN CHECK WHICH CAUSED INTERRUPT
0432  044E  2C FD AF              BIT IFR
0433  0451  D0 21                 BNE VIA1         ; BRANCH IF NOT TIMER 1
0434  0453  49 FF                 EOR #$FF
0435  0455  8D FD AF              STA IFR
0436  0458  A9 FF                 LDA #$FF
0437  045A  49 FF                 EOR #$FF
0438  045C  8D F9 AF              STA T2HC         ;ENABLE COUNT
0439  045F  EE FB 10              INC CLOCK
0440  0462  AD FB 10              LDA CLOCK        ;CHECK FOR STOP-TIME
0441  0465  CD FA 10              CMP STOPTM
0442  0468  90 08                 BCC VIA4
0443  046A  A9 0D                 LDA #<MSG14      ;'STOP-TIME'
0444  046C  8D F9 10              STA ERRPTR
0445  046F  4C 83 09              JMP ERRMSG

0447  0472  68            VIA4   PLA
0448  0473  40                   RTI

0450  0474  50 0A         VIA1   BVC TIME1        ; BRANCH IF TIMER1
0451  0476  A9 02         VIA2   LDA #%00000010
0452  0478  2C FD AF              BIT IFR
0453  047B  F0 45                 BEQ DOCIRQ       ;BRANCH IF END OF CONVERSION IRQ
0454  047D  4C 48 04       VIA3   JMP DIGIRQ       ;SEE IF DIGITAL IRQ



0456  0480  A9 40         TIME1  LDA #%01000000
0457  0482  49 FF                 EOR #$FF
0458  0484  8D FD AF              STA IFR          ;CLEAR TIMER1 IRQ FLAG
0459  0487  98                   TYA              ;SAVE 'Y' & 'X'
0460  0488  48                   PHA
0461  0489  8A                   TXA
0462  048A  48                   PHA

0464  048B  A6 06                LDX NCHNLS       ;SET UP INDEX 'X'
0465  048D  30 07                BMI TCNT5
0466  048F  D6 24         TCNT   DEC COUNTV,X
0467  0491  F0 09                BEQ TCNT2
0468  0493  CA            TCNT1  DEX
0469  0494  10 F9                BPL TCNT
0470  0496  68            TCNT5  PLA              ;RESTORE 'X', 'Y', & 'A'
0471  0497  AA                   TAX
0472  0498  68                   PLA
0473  0499  A8                   TAY
0474  049A  68                   PLA
0475  049B  40                   RTI


0477  049C  24 08         TCNT2  BIT ADBUSY
0478  049E  10 0D                BPL TCNT3        ;GO START A/D CONVERSION
0479  04A0  A9 80                LDA #%10000000
```

```
LINE # LOC     CODE        LINE

0480  04A2  15 14          ORA CHANLS,X     ;SET NEED A/C CONVERSION FLAG THIS CHANNEL
0481  04A4  95 14          STA CHANLS,X
0482  04A6  BD FC 10  TCNT4 LDA COUNTP,X
0483  04A9  95 24          STA COUNTV,X     ;RESET TIMER COUNTER THMS CHANNEL
0484  04AB  D0 E6          BNE TCNT1

0486  04AD  8A        TCNT3 TXA
0487  04AE  09 80          ORA #%10000000
0488  04B0  85 08          STA ADBUSY       ;SET BUSY FLAG
0489  04B2  B5 14          LDA CHANLS,X     ;START A/D THIS CHANNEL
0490  04B4  49 FF          EOR #$FF
0491  04B6  8D E0 AF       STA PORTB        ;START A/D CONVERSION
0492  04B9  20 EB 07       JSR RDTIME       ;SAVE THE TIME OF CONVERSION FOR THIS CHANNEL
0493  04BC  20 06 08       JSR RDTIM1
0494  04BF  4C A6 04       JMP TCNT4
```

```
LINE # LOC     CODE       LINE

0496  04C2                ;***** END OF A/D CONVERSION INTERRUPT *****

0498  04C2  98            EOCIRQ TYA              ;SAVE 'Y' AND 'X'
0499  04C3  48                   PHA
0500  04C4  8A                   TXA
0501  04C5  48                   PHA
0502  04C6  A5 08                LDA ADBUSY
0503  04C8  85 46                STA KEPNDX       ;SAVE FOR LATER USE
0504  04CA  29 0F                AND #$0F
0505  04CC  AA                   TAX              ;GET A/D CONVERTION INDEX 'X'
0506  04CD  AD F1 AF             LDA PORTA        ;GET A/D VALUE
0507  04D0  49 FF                EOR #$FF
0508  04D2  95 CA                STA THSVAL,X
0509  04D4  A6 06                LDX NCHNLS
0510  04D6  B5 14          EOC1  LDA CHANLS,X     ;SEE IF ANY CHANNELS NEED A/D SERVICE
0511  04D8  30 09                BMI EOC2         ;GO START A/D CONVERSION
0512  04DA  CA                   DEX
0513  04DB  10 F9                BPL EOC1

0515  04DD  A9 00                LDA #0
0516  04DF  85 08                STA ADBUSY       ;CLEAR A/D BUSY
0517  04E1  F0 14                BEQ EOC3         ;ALWAYS - GO SERVICE DATA

0519  04E3  29 7F          EOC2  AND #$7F         ;CLEAR A/D REQUEST FLAG
0520  04E5  95 14                STA CHANLS,X
0521  04E7  49 FF                EOR #$FF
0522  04E9  8D F0 AF             STA PORTB        ;START A/D CONVERSION
0523  04EC  20 EB 07             JSR RDTIME       ;SAVE THIS CONVERSION'S SAMPLE TIME
0524  04EF  20 06 08             JSR RDTIM1

0526  04F2  8A                   TXA              ;SET UP ADBUSY FLAG
0527  04F3  09 80                ORA #%10000000
0528  04F5  85 08                STA ADBUSY
0529  04F7  20 1F 08       EOC3  JSR KEEP         ;DETERMINE IF TO KEEP THIS DATA - USES 'KEPNDX
0530  04FA  68                   PLA
0531  04FB  AA                   TAX              ;RESTORE 'X' , 'Y' , AND 'A'
0532  04FC  68                   PLA
0533  04FD  A8                   TAY
0534  04FE  68                   PLA
0535  04FF  40                   RTI
```

```
LINE #  LOC   CODE        LINE

0537  0500                ;********** SUBROUTINES **********

0539  0500                ACTIV                    ;SORT ACTIVE ANALOG & DIGITAL CHANS

0541  0500  A0 00                 LDY #0           ;MAKE LIST OF ACTIVE CHANNELS
0542  0502  A2 0F                 LDX #15
0543  0504  BD 5D 11     ACTIV4  LDA CRATE,X
0544  0507  F0 0E                 BEQ ACTIV5
0545  0509  99 FC 10              STA COUNTP,Y     ;SAVE COUNT
0546  050C  BD 7D 11              LDA STORGE,X
0547  050F  99 6D 11              STA METHOD,Y
0548  0512  8A                    TXA
0549  0513  99 14 00              STA CHANLS,Y
0550  0516  C8                    INY
0551  0517  CA            ACTIV5  DEX
0552  0518  10 EA                 BPL ACTIV4       ;BRANCH IF NOT DONE
0553  051A  88                    DEY
0554  051B  84 06                 STY NCHNLS       ;SAVE NUMBER OF ACTIVE CHANNELS

0556  051D  A2 03                 LDX #3           ;MAKE A LIST OF ACTIVE PORTS
0557  051F  A0 00                 LDY #0
0558  0521  BD 59 11     ACTIV6  LDA PORTBF,X
0559  0524  F0 08                 BEQ ACTIV7       ;BRANCH IF PORT NOT ACTIVE
0560  0526  99 55 11              STA PRATE,Y
0561  0529  8A                    TXA
0562  052A  99 EA 00              STA PORTS,Y      ;SAVE PORT INDEX
0563  052D  C8                    INY
0564  052E  CA            ACTIV7  DEX
0565  052F  10 F0                 BPL ACTIV6       ;BRANCH IN NOT DONE
0566  0531  88                    DEY
0567  0532  84 07                 STY NPORTS       ;SAVE NUMBER OF ACTIVE PORTS

0569  0534  A9 00           ACTIV1  LDA #0          ;SORT ACTIVE CHANNELS
0570  0536  8D E7 10               STA ATEMP        ;RESET EXCHANGE FLAG
0571  0539  A6 06                 LDX NCHNLS
0572  053B  A4 06                 LDY NCHNLS
0573  053D  88                    DEY
0574  053E  30 49                 BMI ACTIV9
0575  0540  B9 FC 10     ACTIV2  LDA COUNTP,X     ;ARRANGE ACTIVE CHANNELS BY RATE
0576  0543  D9 FC 10              CMP COUNTP,Y
0577  0546  90 38                 BCC ACTIV3       ;BRANCH IF LESS THAN OR EQUAL TO
0578  0548  F0 36                 BEQ ACTIV3

0580  054A  8D E7 10              STA ATEMP
0581  054D  B9 FC 10              LDA COUNTP,Y
0582  0550  9D FC 10              STA COUNTP,X     ;EXCLANGE X & X-1
0583  0553  AD E7 10              LDA ATEMP
0584  0556  99 FC 10              STA COUNTP,Y

0586  0559  BD 6D 11              LDA METHOD,X
0587  055C  8D E7 10              STA ATEMP
0588  055F  B9 6D 11              LDA METHOD,Y
0589  0562  9D 6D 11              STA METHOD,X
0590  0565  AD E7 10              LDA ATEMP
0591  0568  99 6D 11              STA METHOD,Y
```

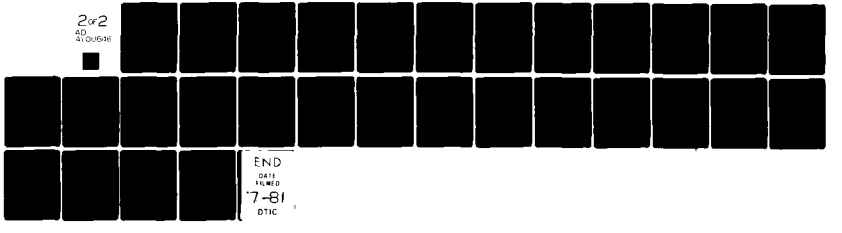
```
LINE # LOC     CODE       LINE

0593  056B  B5 14          LDA CHANLS,X
0594  056D  8D E7 10       STA ATEMP
0595  0570  B9 14 00       LDA CHANLS,Y
0596  0573  95 14          STA CHANLS,X    ;EXCHANGE THESE
0597  0575  AD E7 10       LDA ATEMP
0598  0578  99 14 00       STA CHANLS,Y
0599  057B  A9 FF          LDA #$FF
0600  057D  8D E7 10       STA ATEMP       ;SET FLAG SHOWING EXCHANGE OCCURRED
0601  0580  CA          ACTIV3 DEX
0602  0581  88             DEY
0603  0582  10 BC          BPL ACTIV2      ;BRANCH IF NOT DONE THIS TIME
0604  0584  AD E7 10       LDA ATEMP
0605  0587  D0 AB          BNE ACTIV1      ;BRANCH IF EXCHANGE FLAG SET

0607  0589                ; ***** SORT DIGITAL CHANNELS *****
0608  0589  A9 00       ACTIV9 LDA #0
0609  058B  8D E7 10       STA ATEMP
0610  058E  A6 07          LDX NPORTS
0611  0590  A4 07          LDY NPORTS
0612  0592  88             DEY
0613  0593  30 37          BMI ACTIV8

0615  0595  BD 55 11    ACTIVA LDA PRATE,X
0616  0598  D9 55 11       CMP PRATE,Y
0617  059B  90 26          BCC ACTIVB
0618  059D  F0 24          BEQ ACTIVB
0619  059F  8D E7 10       STA ATEMP
0620  05A2  B9 55 11       LDA PRATE,Y
0621  05A5  9D 55 11       STA PRATE,X
0622  05A8  AD E7 10       LDA ATEMP
0623  05AB  99 55 11       STA PRATE,Y
0624  05AE  B5 FA          LDA PORTS,X
0625  05B0  8D E7 10       STA ATEMP
0626  05B3  B9 FA 00       LDA PORTS,Y
0627  05B6  95 FA          STA PORTS,X
0628  05B8  AD E7 10       LDA ATEMP
0629  05BB  99 FA 00       STA PORTS,Y
0630  05BE  A9 FF          LDA #$FF
0631  05C0  8D E7 10       STA ATEMP
0632  05C3  CA          ACTIVB DEX
0633  05C4  88             DEY
0634  05C5  10 CE          BPL ACTIVA
0635  05C7  AD E7 10       LDA ATEMP
0636  05CA  D0 BD          BNE ACTIV9
0637  05CC  60          ACTIV8 RTS
```

```
LINE # LOC     CODE       LINE

0639  05CD  4C F0 C9    DUMP   JMP MONITR       ;DUMP BUBBLE



0641  05D0               INIT                   ;INITIALIZE THE SYSTEM AFTER RESET
0642  05D0  A9 4B              LDA #<VIAIRQ      ;SET UP IRQ SYSTEM-65 VECTOR
0643  05D2  8D 1D C4           STA $C41D
0644  05D5  A9 04              LDA #>VIAIRQ
0645  05D7  8D 1E C4           STA $C41E

0647  05DA  A9 00              LDA #0            ;STOP ALL TIMER IRQ'S
0648  05DC  49 FF              EOR #$FF
0649  05DE  8D FE AF           STA IER           ;DISABLE VIA IRQ
0650  05E1  A9 01              LDA #1
0651  05E3  49 FF              EOR #$FF
0652  05E5  8D E9 AF           STA TCTL2         ;DISABLE INTERVAL TIMER IRQ
0653  05E8  8D E8 AF           STA TCTL13

0655  05EB  60                 RTS



0657  05EC  60           SLFTST RTS              ;DO SYSTEM SELFTEST


0659  05ED  20 03 06     MSGOUT JSR CRLF         ;OUTPUT A MESSAGE - ZERO PAGE ADDRESS IS IN
0660  05F0  8D 06 C6     MSGOUT1 STA $C606
0661  05F3  A9 10              LDA #>MSG1
0662  05F5  8D 07 C6           STA $C607
0663  05F8  20 39 D4           JSR RKEP          ;MONITOR RKEP SUBROUTINE
0664  05FB  60                 RTS


0666  05FC               GETVAL                 ;GETS A VALUE FROM CRT PUTS IN 'A'
0667  05FC  20 DC D1           JSR READ          ;MONITOR READ SUBROUTINE
0668  05FF  20 C1 D2           JSR OUTPUT
0669  0602  60                 RTS


0671  0603               CRLF                   ;OUTPUTS A CARRIAGE RETURN & LINE FEED TO CRT
0672  0603  48                 PHA
0673  0604  8A                 TXA
0674  0605  48                 PHA
0675  0606  20 F1 D0           JSR CRLOW         ;MONITOR CRLOW SUBROUTINE
0676  0609  68                 PLA
0677  060A  AA                 TAX
0678  060B  68                 PLA
0679  060C  60                 RTS


0681  060D               TOHEX                  ;CHANGES 'A' IN ASKII TO A HEX VALUE IN 'A'
0682  060D  20 06 D3           JSR HEX           ;MONITOR SUBROUTINE
0683  0610  60                 RTS
```

```
LINE # LOC     CODE       LINE

0685  0611                SPACES              ;OUTPUTS 'A' SPACES TO CRT
0686  0611  8D 37 11            STA SCNT
0687  0614  A9 20         SP1   LDA #$20
0688  0616  20 C1 D2            JSR OUTPUT     ;MONITOR SUBROUTINE
0689  0619  CE 37 11            DEC SCNT
0690  061C  D0 F6               BNE SP1
0691  061E  60                  RTS


0693  061F                GHEX2               ;READS TWO CHARACTERS FROM CRT --> HEX IN' 'A'
0694  061F  20 DC D1            JSR READ       ;MONITOR READ SUBROUTINE
0695  0622  20 C1 D2            JSR OUTPUT
0696  0625  20 06 D3            JSR HEX        ;MONITOR HEX SUBROUTINE
0697  0628  20 50 D3            JSR LEFT       ;MONITOR LEFT SUBROUTINE
0698  062B  29 F0               AND #$F0
0699  062D  8D 37 11            STA SCNT
0700  0630  20 DC D1            JSR READ       ;MONITOR READ SUBROUTINE
0701  0633  20 C1 D2            JSR OUTPUT
0702  0636  20 06 D3            JSR HEX        ;MONITOR HEX SUBROUTINE
0703  0639  29 0F               AND #$0F
0704  063B  0D 37 11            ORA SCNT
0705  063E  60                  RTS


0707  063F                OUTHEX              ;OUTPUTS TO CRT THE HEX VALUE OF THE 'A'
0708  063F  20 CE D2            JSR NUMA       ;MONITOR SUBROUTINE
0709  0642  60                  RTS



0711  0643                BUBLE0              ;INITIALIZES BUBBLE & RAM BUFFER
0712  0643                ;SIMPLE ROUTINE TO ZERO BUFFER

0714  0643                LOC   = $6
0715  0643  AD 1F 11            LDA BUFFER
0716  0646  85 06               STA LOC
0717  0648  AD 20 11            LDA BUFFER+1
0718  064B  85 07               STA LOC+1
0719  064D  A2 00               LDX #0

0721  064F                ;ZERO CONSECUTIVE MEMORY LOCATIONS
0722  064F  A9 00         LOOP  LDA #$0
0723  0651  81 06               STA (LOC,X)

0725  0653                ;INCREMENT POINTER "LOC"
0726  0653  A5 06               LDA LOC
0727  0655  18                  CLC
0728  0656  69 01               ADC #1
0729  0658  85 06               STA LOC
0730  065A  A5 07               LDA LOC+1
0731  065C  69 00               ADC #0
0732  065E  85 07               STA LOC+1

0734  0660  A5 06               LDA LOC
0735  0662  C9 00               CMP #0
```

```
LINE # LOC     CODE       LINE

0736  0664  D0 E9            BNE LOOP
0737  0666  A5 07            LDA LOC+1
0738  0668  CD 1E 11          CMP LSTBLK+1
0739  066B  90 E2            BCC LOOP
0740  066D  F0 E0            BEQ LOOP
0741  066F  60               RTS


0743  0670  60          WRBHDR RTS              ;WRITES BUBBLE HEADER INFO - PARAMETERS ETC.
0744  0671  60          RDBHDR RTS              ;READS BUBBLE HEADER INFO & PARAMETERS


0746  0672              TIMERS                ;INITIALIZE FOR A/D AND START TIMERS
0747  0672              ;VIA PORT B INPUT -- PORT A OUTPUT

0749  0672  78                  SEI             ;DISABLE INTERRUPTS
0750  0673  A9 8F               LDA #%10001111   ;LAST 4 BITS ARE A/D CHAN SELECT
0751  0675  49 FF               EOR #$FF        ;INVERT FOR DATA BUS
0752  0677  8D F2 AF            STA DDRB
0753  067A  A9 00               LDA #0
0754  067C  49 FF               EOR #$FF
0755  067E  8D F3 AF            STA DDRA        ;MAKE PORTA INPUT

0757  0681  A9 BD               LDA #%10111101
0758  0683  49 FF               EOR #$FF        ;INVERT FOR DATA BUS
0759  0685  8D FC AF            STA PCR

0761  0688              ; AUXILIARY CONTROL REG INITIALIZE
0762  0688  A9 E3               LDA #%11100011
0763  068A  49 FF               EOR #$FF        ;INVERT FOR DATA BUS
0764  068C  8D FB AF            STA AUXCTL      ;T1 OUT ON PB7, T1=CONTINUOUS
0765  068F              ; T2 CLOCKED  BY PB6, DISABLE SHIFT REG.

0767  068F              ;INTERRUPT ENABLE AND FLAG REGISTER
0768  068F              ;     BIT 0 = CA2      BIT 4 = SHIFT REG
0769  068F              ;         1 = CA1          5 = T2 TIMEOUT
0770  068F              ;         2 = CB2          6 = T1 TIMEOUT
0771  068F              ;         3 = CB1          7 = ANY IRQ

0773  068F  A9 E2               LDA #%11100010
0774  0691  49 FF               EOR #$FF        ;INVERT FOR DATA BUS
0775  0693  8D FD AF            STA IFR         ;CLEAR IRQ FLAGS
0776  0696  8D FE AF            STA IER
0777  0699  A9 FF               LDA #$FF        ;START TIMERS FOR MISSION
0778  069B  49 FF               EOR #$FF
0779  069D  8D F8 AF            STA T2LL
0780  06A0  8D F9 AF            STA T2HC
0781  06A3  AD 3F 11            LDA DELTIM      ;GET SYSTEM DELTA TIME
0782  06A6  8D F6 AF            STA T1LL
0783  06A9  AD 40 11            LDA DELTIM+1
0784  06AC  8D F7 AF            STA T1HC
0785  06AF  60                  RTS
```

AD-A100 646 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 9/2
AIRCREW INFLIGHT PHYSIOLOGICAL DATA ACQUISITION SYSTEM.(U)
JUN 80 K L MOORE
UNCLASSIFIED AFIT/GE/EE/80-6 NL



```
LINE # LOC     CODE       LINE


0788  06B0                RUNIT                ;INIT. POINTERS, COUNTERS, TABLES, ETC
0789  06B0                ;******** WRITE SYSTEM BUBBLE REGISTERS ********
0790  06B0  20 AF B7            JSR PAIN
0791  06B3  AD 00 B8            LDA PA
0792  06B6  F0 03               BEQ RUNIT1
0793  06B8  4C 68 B0            JMP $B068          ;SYSTEM ERROR ROUTINE
0794  06BB  A9 D4       RUNIT1  LDA #$D4           ;LOAD CANCEL COMMAND
0795  06BD  20 DC B7            JSR SEND

0797  06C0  A9 BF               LDA #$BF           ;LOAD RCM REGISTERS COMMAND
0798  06C2  20 DC B7            JSR SEND
0799  06C5  A0 00               LDY #0
0800  06C7  B9 13 11    RUNIT2  LDA REGSTR,Y
0801  06CA  20 DC B7            JSR SEND
0802  06CD  C8                  INY
0803  06CE  C0 09               CPY #9
0804  06D0  90 F5               BCC RUNIT2

0806  06D2  20 AB B7            JSR PAOUT          ;SET UP FOV WRITE TO BUBBLE
0807  06D5  AD 01 B8            LDA CRA
0808  06D8  09 10               ORA #$10
0809  06DA  8D 01 B8            STA CRA            ;CLEAR BUSY DETECTOR

0811  06DD  20 C9 0B            JSR LNKALL         ;LINK ALL OF BUFFER AREA

0813  06E0  A9 00               LDA #0             ;RESET COUNTERS
0814  06E2  8D 0D 11            STA TOTAL
0815  06E5  8D 0E 11            STA TOTAL+1
0816  06E8  8D 0F 11            STA TOTAL+2

0818  06EB  8D 3D 11            STA C40
0819  06EE  8D FB 10            STA CLOCK
0820  06F1  8D 36 11            STA SAVEY
0821  06F4  8D 33 11            STA SAVEA
0822  06F7  8D 3E 11            STA B256NX
0823  06FA  85 0C               STA BBUSY
0824  06FC  85 08               STA ADBUSY
0825  06FE  8D 1C 11            STA DMAFLG
0826  0701  8D 2B 11            STA NPWRDN
0827  0704  8D 2C 11            STA NPWRDN+1
0828  0707  8D 2D 11            STA NPWRUP
0829  070A  8D 2E 11            STA NPWRUP+1
0830  070D  8D 27 11            STA TPDTIM
0831  0710  8D 28 11            STA TPDTIM+1
0832  0713  8D 29 11            STA TPDTIM
0833  0716  8D 2A 11            STA TPDTIM+1
0834  0719  8D 23 11            STA PUTIM
0835  071C  8D 24 11            STA PUTIM+1
0836  071F  8D 21 11            STA PDTIM
0837  0722  8D 22 11            STA PDTIM+1

0839  0725  A9 00               LDA #<BUF0
0840  0727  85 34               STA POINTR
```

```
LINE # LOC     CODE       LINE

0841  0729  A9 BC          LDA #>BUF0
0842  072B  85 35          STA POINTR+1

0844  072D  A9 FF          LDA #$FF
0845  072F  8D 3A 11       STA C40FLG

0847  0732  AD 3C 11       LDA CNT40+1     ;SET UP INTERVAL TIMER - T2 FOR 40 WORD COUNT
0848  0735  49 FF          EOR #$FF
0849  0737  8D EC AF       STA T2CNTR      ;WRITE MSB
0850  073A  AD 3B 11       LDA CNT40
0851  073D  49 FF          EOR #$FF
0852  073F  8D AD AF       STA T2LCHW      ;WRITE TIMER 2 LATCH
0853  0742  A9 E3          LDA #%11100011  ;CONTROL REG2 - ADDRESS CNTL REG 1, CLOCK T2
0854  0744  49 FF          EOR #$FF        ;T2 = 16 BITS, T2 = 1-SHOT, T2 OUTPUT ENABLED
0855  0746  8D E9 AF       STA TCTL2
0856  0749  A9 01          LDA #1          ;DISABLE ALL INTERVAL TIMER IRQ'S
0857  074B  49 FF          EOR #$FF
0858  074D  8D E8 AF       STA TCTL13
0859  0750  60             RTS


0861  0751                 BPWRDN                  ;POWER DOWN THE BUBBLE TO SAVE ENERGY
0862  0751                 ;                          AND CALCULATE TIME BUBBLE UP
0863  0751  48                      PHA
0864  0752  A9 00                   LDA #0
0865  0754  85 0C                   STA BBUSY
0866  0756  78                      SEI
0867  0757  20 EB 07                JSR RDTIME
0868  075A  AD 32 11                LDA SAVAA+1
0869  075D  8D 22 11                STA PDTIM+1
0870  0760  AD 31 11                LDA SAVAA
0871  0763  8D 21 11                STA PDTIM
0872  0766  58                      CLI
0873  0767  38                      SEC
0874  0768  ED 23 11                SBC PUTIM
0875  076B  8D 25 11                STA TIMDIF
0876  076E  AD 22 11                LDA PDTIM+1
0877  0771  ED 24 11                SBC PUTIM+1
0878  0774  8D 26 11                STA TIMDIF+1
0879  0777  18                      CLC
0880  0778  AD 25 11                LDA TIMDIF
0881  077B  6D 27 11                ADC TPUTIM
0882  077E  8D 27 11                STA TPUTIM
0883  0781  AD 26 11                LDA TIMDIF+1
0884  0784  6D 28 11                ADC TPUTIM+1
0885  0787  8D 28 11                STA TPUTIM+1
0886  078A  18                      CLC
0887  078B  A9 01                   LDA #1
0888  078D  6D 2B 11                ADC NPWRDN
0889  0790  8D 2B 11                STA NPWRDN
0890  0793  A9 00                   LDA #0
0891  0795  6D 2C 11                ADC NPWRDN+1
0892  0798  8D 2C 11                STA NPWRDN+1
0893  079B  68                      PLA
0894  079C  60                      RTS
```

```
LINE # LOC     CODE       LINE

0896  079D                BPWRUP                ;POWER UP THE BUBBLE
0897  079D                ;                   AND CALCULATE TIME DOWN
0898  079D  48                       PHA
0899  079E  A9 C0                    LDA #%11000000  ;SET POWER & BUSY FLAG
0900  07A0  05 0C                    ORA BBUSY
0901  07A2  85 0C                    STA BBUSY
0902  07A4  78                       SEI
0903  07A5  20 EB 07                 JSR RDTIME
0904  07A8  AD 32 11                 LDA SAVAA+1
0905  07AB  8D 24 11                 STA PUTIM+1
0906  07AE  AD 31 11                 LDA SAVAA
0907  07B1  8D 23 11                 STA PUTIM
0908  07B4  58                       CLI
0909  07B5  38                       SEC
0910  07B6  ED 21 11                 SBC PDTIM
0911  07B9  8D 25 11                 STA TIMDIF
0912  07BC  AD 24 11                 LDA PUTIM+1
0913  07BF  ED 22 11                 SBC PUTIM+1
0914  07C2  8D 26 11                 STA TIMDIF+1
0915  07C5  18                       CLC
0916  07C6  AD 25 11                 LDA TIMDIF
0917  07C9  6D 29 11                 ADC TPDTIM
0918  07CC  8D 29 11                 STA TPDTIM
0919  07CF  AD 2A 11                 LDA TPDTIM+1
0920  07D2  6D 26 11                 ADC TIMDIF+1
0921  07D5  8D 2A 11                 STA TPDTIM+1
0922  07D8  18                       CLC
0923  07D9  A9 01                    LDA #1
0924  07DB  6D 2D 11                 ADC NPWRUP
0925  07DE  8D 2D 11                 STA NPWRUP
0926  07E1  A9 00                    LDA #0
0927  07E3  6D 2E 11                 ADC NPWRUP+1
0928  07E6  8D 2E 11                 STA NPWRUP+1
0929  07E9  68                       PLA
0930  07EA  60                       RTS


0932  07EB                RDTIME                ;SAVE TIMER VALUES THIS CONVERSION - USE 'X'
0933  07EB  AD F8 AF                 LDA T2LL       ;MASTER TIMER LOW
0934  07EE  AC F9 AF                 LDY T2HC
0935  07F1  49 FF                    EOR #$FF       ;INVERT FOR BUSS
0936  07F3  49 FF                    EOR #$FF       ;MAKE 2'S COMPLEMENT
0937  07F5  18                       CLC
0938  07F6  69 01                    ADC #1
0939  07F8  8D 31 11                 STA SAVAA
0940  07FB  98                       TYA
0941  07FC  49 FF                    EOR #$FF
0942  07FE  49 FF                    EOR #$FF
0943  0800  69 00                    ADC #0
0944  0802  8D 32 11                 STA SAVAA+1
0945  0805  60                       RTS

0947  0806  8A             RDTIM1 TXA
0948  0807  0A                       ASL A
```

```
LINE #  LOC    CODE        LINE

0949  0808  A8              TAY
0950  0809  AD 31 11        LDA SAVAA
0951  080C  99 DA 00        STA THSTIM,Y
0952  080F  AD 32 11        LDA SAVAA+1
0953  0812  99 DB 00        STA THSTIM+1,Y
0954  0815  F6 4A           INC COUNIT,X    ;INCREMENT DELTA TIME COUNTER
0955  0817  60              RTS
```

```
LINE # LOC     CODE       LINE

0957  0818                KEEP                    ;DETERMINES IF VALUE SAVED OR THROWN AWAY
0958  0818  A5 46               LDA KEPNDX        ;CHECK METHOD OF STORAGE

0960  081A                ;BITS 2 1 0    STORAGE METHOD

0962  081A                ;      0 0 0    MODIFIED VAR. CHANGE—1 LSB
0963  081A                ;      0 0 1    MODIFIED VAR. CHANGE—2 LSB
0964  081A                ;      0 1 0    DELTA CONTINUOUS—1 LSB
0965  081A                ;      0 1 1    DELTA CONTINUOUS—2 LSB
0966  081A                ;      1 0 0    VAR. CHANGE—1 LSB
0967  081A                ;      1 0 1    VAR. CHANGE—2 LSB
0968  081A                ;      1 1 0    DO NOT SAVE THIS CHANNEL
0969  081A                ;      1 1 1    CONTINUOUS

0971  081A  29 0F                AND #$0F
0972  081C  AA                   TAX
0973  081D  BD 6D 11              LDA METHOD,X
0974  0820  29 07                AND #%00000111
0975  0822  F0 22                BEQ MVC1
0976  0824  C9 01                CMP #1
0977  0826  F0 22                BEQ MVC2
0978  0828  C9 02                CMP #2
0979  082A  F0 6A                BEQ DCM1
0980  082C  C9 03                CMP #3
0981  082E  F0 6A                BEQ DCM2
0982  0830  C9 07                CMP #7
0983  0832  D0 03                BNE *+5
0984  0834  4C 7E 09             JMP CNTINUS
0985  0837  C9 04                CMP #4
0986  0839  D0 03                BNE *+5
0987  083B  4C F0 08             JMP VCM1
0988  083E  C9 05                CMP #5
0989  0840  D0 03                BNE *+5
0990  0842  4C F4 08             JMP VCM2
0991  0845  60                   RTS


0993  0846  A9 01          MVC1  LDA #1            ;MODIFIED VARIABLE CHANGE STORAGE METHOD
0994  0848  D0 02                BNE MVC
0995  084A  A9 02          MVC2  LDA #2
0996  084C  85 FE          MVC   STA ACURCY

0998  084E  20 9E 09             JSR DIFF          ;CALCULATE DIFFERENCE AND MAGNITUDE

1000  0851  A5 49                LDA MDIFF
1001  0853  F0 3A                BEQ KEEP7         ;SEE IF TIMER OVERFLOWED

1003  0855  C9 08                CMP #8
1004  0857  90 0E                BCC MVC3          ;CHECK MAGNITUDE OUT OF RANGE
1005  0859  C9 09                CMP #9
1006  085B  90 03                BCC *+5
1007  085D  4C 76 09             JMP KEEP9
1008  0860  A5 47                LDA VDIFF
1009  0862  30 03                BMI *+5
1010  0864  4C 76 09             JMP KEEP9
```

```
LINE # LOC     CODE       LINE

1012  0867  B5 4A       MVC3  LDA COUNIT,X    ;CALCULATE AND OUTPUT VALUE
1013  0869  0A                 ASL A
1014  086A  0A                 ASL A
1015  086B  0A                 ASL A
1016  086C  0A                 ASL A
1017  086D  95 4A              STA COUNIT,X
1018  086F  A5 47              LDA VDIFF
1019  0871  29 0F              AND #$0F
1020  0873  15 4A              ORA COUNIT,X


1022  0875  20 C6 09    KEEP6  JSR SAVE        ;GO SAVE 'A' INTO BUFFER 'X'
1023  0878  8A                 TXA             ;UPDATE LSTVAL $ LSTTIM
1024  0879  0A                 ASL A
1025  087A  A8                 TAY
1026  087B  B9 DA 00           LDA THSTIM,Y
1027  087E  99 AA 00           STA LSTTIM,Y
1028  0881  B9 DB 00           LDA THSTIM+1,Y
1029  0884  99 AB 00           STA LSTTIM+1,Y
1030  0887  B5 9A              LDA LSTVAL,X
1031  0889  18                 CLC
1032  088A  65 48              ADC SDIFF
1033  088C  95 9A              STA LSTVAL,X
1034  088E  60                 RTS


1036  088F  B5 4A       KEEP7  LDA COUNIT,X    ;CHECK TIME OVERFLOW
1037  0891  29 0F              AND #$0F
1038  0893  F0 E0              BEQ KEEP6
1039  0895  60                 RTS



1041  0896  A9 01       DCM1   LDA #1          ;DELTA CONTINUOUS STORAGE METHOD
1042  0898  D0 02              BNE DCM
1043  089A  A9 02       DCM2   LDA #2
1044  089C  85 FE       DCM    STA ACURCY
1045  089E  20 9E 09           JSR DIFF        ;GET DIFFERENCE AND MAGNITUDE

1047  08A1  A5 49              LDA MDIFF       ;CHECK FOR OUT OF RANGE ERROR
1048  08A3  C9 08              CMP #8
1049  08A5  90 0E              BCC DCM3
1050  08A7  C9 09              CMP #9
1051  08A9  90 03              BCC *+5
1052  08AB  4C 76 09           JMP KEEP9
1053  08AE  A5 47              LDA VDIFF
1054  08B0  30 03              BMI *+5
1055  08B2  4C 76 09           JMP KEEP9

1057  08B5  B5 14       DCM3   LDA CHANLS,X    ;CALCULATE AND OUTPUT VALUES
1058  08B7  29 40              AND #%01000000
1059  08B9  D0 17              BNE DCM4        ;BRANCH IF WORD FULL
1060  08BB  B5 14              LDA CHANLS,X
1061  08BD  09 40              ORA #%01000000  ;RESET FLAG
1062  08BF  95 14              STA CHANLS,X
```

```
LINE # LOC     CODE       LINE

1064  08C1  A5 47          LDA VDIFF
1065  08C3  29 0F          AND #$0F
1066  08C5  0A             ASL A
1067  08C6  0A             ASL A
1068  08C7  0A             ASL A
1069  08C8  0A             ASL A
1070  08C9  1D 6D 11       ORA METHOD,X
1071  08CC  9D 6D 11       STA METHOD,X
1072  08CF  4C 78 08       JMP KEEP6+3

1074  08D2  55 14       DCM4  EOR CHANLS,X
1075  08D4  95 14             STA CHANLS,X
1076  08D6  BD 6D 11          LDA METHOD,X
1077  08D9  29 F0             AND #$F0
1078  08DB  8D E8 10          STA TEMP
1079  08DE  BD 6D 11          LDA METHOD,X    ;CLEAR OLD DELTA VALUE
1080  08E1  29 0F             AND #$0F
1081  08E3  9D 6D 11          STA METHOD,X
1082  08E6  A5 47             LDA VDIFF
1083  08E8  29 0F             AND #$0F
1084  08EA  0D E8 10          ORA TEMP
1085  08ED  4C 75 08          JMP KEEP6


1087  08F0  A9 01       VCM1  LDA #1          ;VERIABLE CHANGE STORAGE METHOD
1088  08F2  D0 02             BNE VCM
1089  08F4  A9 02       VCM2  LDA #2
1090  08F6  85 FE       VCM   STA ACURCY
1091  08F8  20 9E 09          JSR DIFF        ;CALCULATE DIFFERENCE & ADJUST & MAGNITUDE

1093  08FB  A5 49             LDA MDIFF
1094  08FD  F0 07             BEQ VCM5

1096  08FF  C9 10             CMP #16
1097  0901  90 0A             BCC VCM3        ;CHECK DIFFERENCE OUT OF RANGE
1098  0903  4C 76 09          JMP KEEP9

1100  0906  B5 4A       VCM5  LDA COUNTT,X
1101  0908  29 7F             AND #$7F
1102  090A  F0 01             BEQ VCM3
1103  090C  60                RTS

1105  090D  B5 14       VCM3  LDA CHANLS,X
1106  090F  29 40             AND #%01000000
1107  0911  D0 27             BNE VCM4
1108  0913  B5 14             LDA CHANLS,X
1109  0915  09 40             ORA #%01000000
1110  0917  95 14             STA CHANLS,X
1111  0919  B5 4A             LDA COUNTT,X
1112  091B  9D E9 10          STA KOUNTT,X
1113  091E  A5 47             LDA VDIFF
1114  0920  10 04             BPL VCM8
1115  0922  A9 10             LDA #$10
1116  0924  D0 02             BNE *+4
```

```
LINE # LOC     CODE        LINE

1117  0926  A9 00       VCM8  LDA #0
1118  0928  05 49             ORA MDIFF
1119  092A  0A               ASL A
1120  092B  0A               ASL A
1121  092C  0A               ASL A
1122  092D  1D 6D 11         ORA METHOD,X
1123  0930  9D 6D 11         STA METHOD,X
1124  0933  A9 00             LDA #0
1125  0935  95 4A             STA COUNTT,X
1126  0937  4C 78 08         JMP KEEP6+3

1128  093A  55 14       VCM4  EOR CHANLS,X
1129  093C  95 14             STA CHANLS,X
1130  093E  BD 6D 11         LDA METHOD,X
1131  0941  29 F8             AND #$F8
1132  0943  0A               ASL A
1133  0944  8D E8 10         STA TEMP
1134  0947  BD E9 10         LDA KOUNTT,X
1135  094A  90 02             BCC VCM6
1136  094C  09 80             ORA #$80
1137  094E  20 C6 09    VCM6  JSR SAVE
1138  0951  A5 49             LDA MDIFF
1139  0953  0D E8 10         ORA TEMP
1140  0956  20 C6 09         JSR SAVE
1141  0959  B5 4A             LDA COUNTT,X
1142  095B  24 47             BIT VDIFF
1143  095D  10 02             BPL VCM7
1144  095F  09 80             ORA #$80
1145  0961  20 C6 09    VCM7  JSR SAVE
1146  0964  A9 00             LDA #0
1147  0966  9D E9 10         STA KOUNTT,X
1148  0969  95 4A             STA COUNTT,X

1150  096B  BD 6D 11         LDA METHOD,X     ;RESET SAVED VALUE
1151  096E  29 07             AND #$7
1152  0970  9D 6D 11         STA METHOD,X

1154  0973  4C 78 08         JMP KEEP6+3

1156  0976  A9 A1       KEEP9 LDA #<MSG9       ;DATA RANGE ERROR
1157  0978  8D E9 10         STA ERROTP
1158  097B  4C 83 09         JMP ERRMSG

1160  097E  B5 CA       CNTINUS LDA THISVAL,X  ;SUBROUTINE TO HANDLE CONTINUOUS STORAGE
1161  0980  4C C6 09         JMP SAVE



1163  0983  78         ERRMSG SEI             ;ERROR MESSAGE - UPDATE TIMERS & HALT PROGRAM
1164  0984  A5 0C             LDA BUSY
1165  0986  29 40             AND #$40
1166  0988  F0 07             BEQ IRT
1167  098A  20 51 07         JSR BUWRDN       ;UPDATE POWERED UP TIMES
1168  098D  78               SEI
```

```
LINE # LOC     CODE       LINE

1169  098E  4C 95 09          JMP ER2
1170  0991  20 9D 07    ER1   JSR BPWRUP      ;UPDATE POWERED DOWN TIMES
1171  0994  78                SEI

1173  0995  AD F9 10    ER2   LDA ERRPTR
1174  0998  20 ED 05          JSR MSGOUT
1175  099B  4C F0 C9          JMP MONITR



1177  099E  38          DIFF  SEC             ;CALCULATE DIFFERENCE AND MAGNITUDE VALUES
1178  099F  B5 CA              LDA THSVAL,X
1179  09A1  F5 9A              SBC LSTVAL,X
1180  09A3  85 48              STA SDIFF       ;SAVE DIFFERENCE
1181  09A5  85 47              STA VDIFF
1182  09A7  10 05              BPL DIFF1
1183  09A9  18                 CLC
1184  09AA  49 FF              EOR #$FF
1185  09AC  69 01              ADC #1

1187  09AE  85 49        DIFF1 STA MDIFF       ;SAVE MAGNITUDE OF DIFFERENCE

1189  09B0  A5 FE              LDA ACURCY      ;ADJUST DIFFERENCE & MAGNITUDE
1190  09B2  C9 02              CMP #2
1191  09B4  90 0F              BCC ADIFF1
1192  09B6  24 47              BIT VDIFF       ;SET CARRY IF NEGATIVE
1193  09B8  30 01              BMI *+3
1194  09BA  18                 CLC
1195  09BB  66 47              ROR VDIFF
1196  09BD  46 49              LSR MDIFF
1197  09BF  A5 48              LDA SDIFF
1198  09C1  29 FE              AND #$FE
1199  09C3  85 48              STA SDIFF
1200  09C5  60          ADIFF1 RTS
```

```
LINE # LOC     CODE        LINE

1202  09C6                 SAVE                    ;SUBROUTINE TO SAVE 'A' IN BUFFER BLOCK X
1203  09C6  8E 34 11             STX SAVEX         ;SAVE REGISTERS
1204  09C9  8C 36 11             STY SAVEY
1205  09CC  8D 33 11             STA SAVEA
1206  09CF  A9 10                LDA #$10          ;CHECK FOR DIGITAL CHANNEL
1207  09D1  2C 34 11             BIT SAVEX
1208  09D4  D0 3D                BNE SAVE4         ;BRANCH IF DIGITAL CHANNEL

1210  09D6  8A             SAVE3 TXA               ;GET INDIRECT ANALOG POINTER
1211  09D7  0A                   ASL A
1212  09D8  AA                   TAX
1213  09D9  B4 7A                LDY CCHBLK,X
1214  09DB  84 34                STY POINTR
1215  09DD  B4 7B                LDY CCHBLK+1,X
1216  09DF  84 35                STY POINTR+1
1217  09E1  AE 34 11             LDX SAVEX
1218  09E4  B4 36                LDY CBKPTR,X      ;GET OFFSET
1219  09E6  AD 33 11             LDA SAVEA
1220  09E9  91 34                STA (POINTR),Y    ;SAVE DATA

1222  09EB  C8                   INY               ;CHECK THIS BLOCK OVERFLOWED
1223  09EC  CC 2F 11             CPY UBKSZ1        ;COMPARE TO UNIT BLOCK SIZE
1224  09EF  B0 06                BCS SAVE2         ;BRANCH IF BLOCK OVERFLOW

1226  09F1  94 36          SAVE1 STY CBKPTR,X      ;UPDATE OFFSET POINTER

1228  09F3  AC 36 11             LDY SAVEY         ;RESTORE REGISTERS AND RETURN
1229  09F6  60                   RTS

1231  09F7  20 A4 0A       SAVE2 JSR BLKALC        ;GET ANOTHER BLOCK ALLOCATED
1232  09FA  8A                   TXA
1233  09FB  0A                   ASL A
1234  09FC  AA                   TAX
1235  09FD  A5 10                LDA NEWBLK
1236  09FF  AC 2F 11             LDY UBKSZ1
1237  0A02  91 34                STA (POINTR),Y    ;LINK BLOCKS
1238  0A04  95 7A                STA CCHBLK,X      ;UPDATE CURRENT BLOCK POINTER
1239  0A06  C8                   INY
1240  0A07  A5 11                LDA NEWBLK+1
1241  0A09  91 34                STA (POINTR),Y
1242  0A0B  95 7B                STA CCHBLK+1,X
1243  0A0D  20 63 0A             JSR WRHDR         ;WRITE BLOCK HEADER
1244  0A10  4C F1 09             JMP SAVE1

1246  0A13  8A             SAVE4 TXA               ;THIS IS A DIGITAL CHANNEL
1247  0A14  29 03                AND #$03
1248  0A16  8D 35 11             STA SAVX
1249  0A19  0A                   ASL A
1250  0A1A  AA                   TAX
1251  0A1B  BD 49 11             LDA CPBLK,X
1252  0A1E  85 34                STA POINTR
1253  0A20  BD 4A 11             LDA CPBLK+1,X
1254  0A23  85 35                STA POINTR+1
1255  0A25  AE 35 11             LDX SAVX
1256  0A28  BD 51 11             LDA PBKPTR,X
```

```
LINE # LOC     CODE       LINE

1257  0A2B  A8             TAY
1258  0A2C  AD 33 11       LDA SAVEA
1259  0A2F  91 34          STA (POINTR),Y  ;SAVE DIGITAL CHANNEL DATA
1260  0A31  C8             INY
1261  0A32  CC 2F 11       CPY UBKSZ1      ;CHECK BLOCK OVERFLOW
1262  0A35  B0 0E          BCS SAVE6

1264  0A37  98       SAVE5 TYA
1265  0A38  9D 51 11       STA PBKPTR,X
1266  0A3B  AC 36 11       LDY SAVEY
1267  0A3E  AD 33 11       LDA SAVEA
1268  0A41  AE 34 11       LDX SAVEX
1269  0A44  60             RTS

1271  0A45  20 A4 0A SAVE6 JSR BLKALC
1272  0A48  8A             TXA
1273  0A49  0A             ASL A
1274  0A4A  AA             TAX
1275  0A4B  A5 10          LDA NEWBLK
1276  0A4D  AC 2F 11       LDY UBKSZ1
1277  0A50  91 34          STA (POINTR),Y
1278  0A52  9D 49 11       STA CPBLK,X
1279  0A55  C8             INY
1280  0A56  A5 11          LDA NEWBLK+1
1281  0A58  91 34          STA (POINTR),Y
1282  0A5A  9D 4A 11       STA CPBLK+1,X
1283  0A5D  20 63 0A       JSR WRHDR
1284  0A60  4C 37 0A       JMP SAVE5



1286  0A63  A0 00    WRHDR  LDY #0          ;WRITE BLOCK HEADER DATA
1287  0A65  AD 34 11        LDA SAVEX       ;CHECK ANALOG OR DIGITAL
1288  0A68  29 10           AND #$10
1289  0A6A  F0 11           BEQ WRHDR1

1291  0A6C  AE 35 11        LDX SAVX        ;DIGITAL CHANNEL
1292  0A6F  B5 FA           LDA PORTS,X
1293  0A71  29 03           AND #$3
1294  0A73  09 10           ORA #$10
1295  0A75  91 10           STA (NEWBLK),Y
1296  0A77  09 80           ORA #$80
1297  0A79  91 34           STA (POINTR),Y  ;FLAG LAST BLOCK FULL
1298  0A7B  C8              INY
1299  0A7C  60              RTS

1301  0A7D  AE 34 11 WRHDR1 LDX SAVEX       ;ANALOG CHANNEL
1302  0A80  B5 14           LDA CHANLS,X
1303  0A82  29 0F           AND #$0F
1304  0A84  91 10           STA (NEWBLK),Y
1305  0A86  09 80           ORA #$80
1306  0A88  91 34           STA (POINTR),Y  ;FLAG LAST BLOCK FULL
1307  0A8A  C8              INY
1308  0A8B  8A              TXA
1309  0A8C  0A              ASL A
```

```
LINE # LOC     CODE       LINE

1310  0A8D  AA              TAX
1311  0A8E  B5 DA           LDA THISTIM,X
1312  0A90  91 10           STA (NEWBLK),Y
1313  0A92  B5 DB           LDA THISTIM+1,X
1314  0A94  C8              INY
1315  0A95  91 10           STA (NEWBLK),Y
1316  0A97  AE 34 11        LDX SAVEX
1317  0A9A  B5 CA           LDA THISVAL,X
1318  0A9C  C8              INY
1319  0A9D  91 10           STA (NEWBLK),Y
1320  0A9F  A0 33 11        LDA SAVEA
1321  0AA2  C8              INY
1322  0AA3  60              RTS


1324  0AA4            BLKALC                  ;BLOCK ALLOCATE
1325  0AA4                                    ;DELINK A BLOCK, IF 80% BUFFER FULL SET FLAG

1327  0AA4  C6 0D           DEC NLINK          ;CHECK LINK EMPTY
1328  0AA6  30 28           BMI BLKAL1

1330  0AA8  A5 0E           LDA TLINK          ;ASSIGN A NEW BLOCK
1331  0AAA  85 10           STA NEWBLK
1332  0AAC  A5 0F           LDA TLINK+1
1333  0AAE  85 11           STA NEWBLK+1

1335  0AB0  A0 00           LDY #0             ;DELINK BLOCK
1336  0AB2  B1 10           LDA (NEWBLK),Y
1337  0AB4  85 0E           STA TLINK
1338  0AB6  C8              INY
1339  0AB7  B1 10           LDA (NEWBLK),Y
1340  0AB9  85 0F           STA TLINK+1

1342  0ABB  A4 0D     BLKAL3 LDY NLINK          ;CHECK 80% BUFFER FULL
1343  0ABD  CC 0C 11        CPY N80
1344  0AC0  B0 07           BCS BLKAL2         ;BRANCH IF NOT FULL

1346  0AC2  A5 0C     B80    LDA BBUSY          ;SET BUFFER 80% FULL FLAG
1347  0AC4  09 20           ORA #%00100000
1348  0AC6  85 0C           STA BBUSY
1349  0AC8  60              RTS
1350  0AC9  A5 0C     BLKAL2 LDA BBUSY          ;CLEAR 80% FLAG
1351  0ACB  29 0F           AND #%11011111
1352  0ACD  85 0C           STA BBUSY
1353  0ACF  60              RTS

1355  0AD0  A9 AF     BLKAL1 LDA #<MSG11        ;'RAM DEPLETED'
1356  0AD2  8D F9 10        STA ERRPTR
1357  0AD5  4C 83 09        JMP ERRMSG


1359  0AD8            LINK                    ;ADD A BLOCK TO THE LINK - INCREMENT THE LINK COUNT

1361  0AD8  A0 00           LDY #0
1362  0ADA  A5 0E           LDA TLINK
```

```
LINE # LOC     CODE       LINE

1363  0ADC  91 12            STA (OLDBLK),Y
1364  0ADE  C8               INY
1365  0ADF  A5 0F            LDA TLINK+1
1366  0AE1  91 12            STA (OLDBLK),Y
1367  0AE3  A5 12            LDA OLDBLK
1368  0AE5  85 0E            STA TLINK
1369  0AE7  A5 13            LDA OLDBLK+1
1370  0AE9  85 0F            STA TLINK+1

1372  0AEB  E6 0D            INC NLINK        ;CHECK 80% BUFFER FULL
1373  0AED  D0 0C            BNE BLKAL3

1375  0AEF  60               RTS
```

```
LINE # LOC     CODE       LINE

1377  0AF0                BUBBLE                  ;PUT LINKED LIST, ADDRESSED BY (CURBFA) INTO BUBBLE
1378  0AF0  8E 39 11              STX BSAVX

1380  0AF3  A0 00         BUBLE1 LDY #0
1381  0AF5  84 09                 STY BUBNDX      ;ZERO BUBBLE INDEX
1382  0AF7  B1 0A                 LDA (CURBFA),Y  ;CHECK BLOCK FULL
1383  0AF9  30 28                 BMI BUBLE2      ;BRANCH IF BLOCK FULL
1384  0AFB  A9 10                 LDA #$10        ;CHECK FOR DIGITAL CHANNEL
1385  0AFD  24 0C                 BIT BRUSY
1386  0AFF  F0 11                 BEQ BUBLE7      ;BRANCH IF ANALOG CHANNEL

1388  0B01  AD 39 11              LDA BSAVX
1389  0B04  AA                    TAX             ;UPDATE DIGITAL CHAN POINTERS & RETURN
1390  0B05  0A                    ASL A
1391  0B06  A8                    TAY
1392  0B07  A5 0A                 LDA CURBFA
1393  0B09  99 41 11              STA PBFADD,Y
1394  0B0C  A5 0B                 LDA CURBFA+1
1395  0B0E  99 42 11              STA PBFADD+1,Y
1396  0B11  60                    RTS

1398  0B12                ;UPDATE ANALOG CHAN POINTERS & RETURN
1399  0B12  AD 39 11      BUBLE7 LDA BSAVX        ;NOT FULL - DON'T DUMP
1400  0B15  AA                    TAX             ;RESTORE X
1401  0B16  0A                    ASL A           ;UPDATE POINTERS AND RETURN
1402  0B17  A8                    TAY
1403  0B18  A5 0A                 LDA CURBFA
1404  0B1A  99 5A 00              STA CBFADD,Y
1405  0B1D  A5 0B                 LDA CURBFA+1
1406  0B1F  99 5B 00              STA CBFADD+1,Y
1407  0B22  60                    RTS

1409  0B23  2C 1C 11      BUBLE2 BIT DMAFLG
1410  0B26  10 09                 BPL BUBLE6
1411  0B28  AD 00 B8              LDA PA
1412  0B2B  20 3D B7              JSR WAITB       ;WAIT FOR DMA OVER & CHECK ERROR
1413  0B2E  8D 1C 11              STA DMAFLG
1414  0B31  A4 09         BUBLE6 LDY BUBNDX
1415  0B33  AE 3E 11              LDX B256NX

1417  0B36  B1 0A         BUBLE3 LDA (CURBFA),Y
1418  0B38  9D 00 BC              STA BUF0,X      ;SAVE WORD IN BUBBLE OUTPUT BUFFER
1419  0B3B  EE 3E 11              INC B256NX
1420  0B3E  D0 18                 BNE BUBLE4

1422  0B40                ;START SYSTEM-65 DMA
1423  0B40  20 AB B7              JSR PAUSE
1424  0B43  A9 48                 LDA #$48        ;SEND WRITE COMMAND
1425  0B45  20 DC B7              JSR SEND
1426  0B48  A9 FF                 LDA #$FF
1427  0B4A  8D 1C 11              STA DMAFLG
1428  0B4D  20 DC B7              JSR SEND        ;SEND EXECUTE COMMAND

1430  0B50  AD 01 B8              LDA ORA
1431  0B53  09 10                 ORA #$10
```

```
LINE # LOC     CODE       LINE

1432  0B55  8D 01 B8          STA CRA          ;CLEAR BUSY DETECTOR

1434  0B58  EE 3D 11    BUBLE4 INC C40
1435  0B5B  AD 3D 11           LDA C40
1436  0B5E  C9 28              CMP #40
1437  0B60  90 3F              BCC BUBLE5       ;BRANCH ON LESS THAN


1439  0B62  A9 00              LDA #0
1440  0B64  8D 3D 11           STA C40          ;RESET COUNTER
1441  0B67  18                 CLC
1442  0B68  AD 0D 11           LDA TOTAL        ;CHECK FOR BUBBLE FULL
1443  0B6B  69 28              ADC #40
1444  0B6D  8D 0D 11           STA TOTAL
1445  0B70  A9 00              LDA #0
1446  0B72  6D 0E 11           ADC TOTAL+1
1447  0B75  8D 0E 11           STA TOTAL+1
1448  0B78  A9 00              LDA #0
1449  0B7A  6D 0F 11           ADC TOTAL+2
1450  0B7D  8D 0F 11           STA TOTAL+2

1452  0B80  CD 12 11           CMP FTOTAL+2
1453  0B83  90 1C              BCC BUBLE5
1454  0B85  D0 12              BNE IMSG
1455  0B87  AD 0E 11           LDA TOTAL+1
1456  0B8A  CD 11 11           CMP FTOTAL+1
1457  0B8D  90 12              BCC BUBLE5
1458  0B8F  D0 08              BNE IMSG
1459  0B91  AD 0D 11           LDA TOTAL
1460  0B94  CD 10 11           CMP FTOTAL
1461  0B97  90 08              BCC BUBLE5

1463  0B99  A9 BC        IMSG  LDA #<MSG12      ;'STOP-BUBBLE FULL'
1464  0B9B  8D F9 10           STA ERRPTR
1465  0B9E  4C 83 09           JMP ERRMSG



1467  0BA1  C8          BUBLE5 INY             ;CHECK END OF BUFFER BLOCK X
1468  0BA2  84 09              STY BUBNDX
1469  0BA4  CC 2F 11           CPY UBKSZ1
1470  0BA7  B0 03              BCS BUBLE8
1471  0BA9  4C 23 0B           JMP BUBLE2       ;JUMP ON 'LESS THAN'

1473  0BAC  A5 0A        BUBLE8 LDA CURBFA      ;RETURN BLOCK TO STACK
1474  0BAE  85 12              STA OLDBLK
1475  0BB0  A5 0B              LDA CURBFA+1
1476  0BB2  85 13              STA OLDBLK+1

1478  0BB4  B1 0A              LDA (CURBFA),Y   ;MODIFY BLOCK POINTERS
1479  0BB6  8D 38 11           STA TEMPA
1480  0BB9  C8                 INY
1481  0BBA  B1 0A              LDA (CURBFA),Y
1482  0BBC  85 0B              STA CURBFA+1
1483  0BBE  AD 38 11           LDA TEMPA
```

```
LINE # LOC     CODE       LINE

1484  0BC1  85 0A             STA CURBFA

1486  0BC3  20 D8 0A          JSR LINK        ;LINK THIS BLOCK INTO STACK

1488  0BC6  4C F3 0A          JMP BUBLE1


1490  0BC9                LNKALL              ;LINKS TOGETHER ALL BLOCKS OF UNIT BLOCK SIZE
1491  0BC9                ;                   'UBKSIZ' STRTING WITH 'BUFFER => 'LSTBLK'
1492  0BC9  A9 01             LDA #1          ;LINK FIRST BLOCK
1493  0BCB  85 0D             STA NLINK       ;SET UP BLOCK COUNT
1494  0BCD  AD 1D 11          LDA LSTBLK
1495  0BD0  85 0E             STA TLINK
1496  0BD2  85 12             STA OLDBLK
1497  0BD4  AD 1E 11          LDA LSTBLK+1
1498  0BD7  85 0F             STA TLINK+1
1499  0BD9  85 13             STA OLDBLK+1

1501  0BDB  A0 01             LDY #1          ;SET NULL POINTER
1502  0BDD  A9 00             LDA #0
1503  0BDF  91 12             STA (OLDBLK),Y

1505  0BE1  18            LNKAL1 CLC           ;MODIFY POINTERS
1506  0BE2  A5 12             LDA OLDBLK
1507  0BE4  ED 30 11          SBC UBKSIZ
1508  0BE7  85 12             STA OLDBLK
1509  0BE9  A5 13             LDA OLDBLK+1
1510  0BEB  E9 00             SBC #0
1511  0BED  85 13             STA OLDBLK+1
1512  0BEF  20 D8 0A          JSR LINK        ;LINK NEXT BLOCK
1513  0BF2  A5 13             LDA OLDBLK+1    ;CHECK IF LAST BLOCK
1514  0BF4  CD 20 11          CMP BUFFER+1
1515  0BF7  F0 02             BEQ LNKAL2      ;BRANCH > OR =
1516  0BF9  B0 E6             BCS LNKAL1
1517  0BFB  A5 12        LNKAL2 LDA OLDBLK
1518  0BFD  CD 1F 11          CMP BUFFER
1519  0C00  F0 02             BEQ LNKAL3      ;BRANCH > OR =
1520  0C02  B0 DD             BCS LNKAL1
1521  0C04  60           LNKAL3 RTS
```

```
LINE # LOC      CODE       LINE

1523  0005                ;********** VARIABLES **********
1524  0005                      *=$1000


1526  1000  49 4E         MSG1   .BYT 'INITIALIZE - I'
1527  100E  20 20                .BYT '     DUMP - D ;'
1528  101C  45 4E         MSG3   .BYT 'ENTER;'
1529  1022  43 48         MSG3A  .BYT 'CHAN   PERIODS'
1530  1030  20 20                .BYT '      METHOD;'
1531  103C  50 4F         MSG4   .BYT 'PORT   PERIODS;'
1532  104B  56 45         MSG5   .BYT 'VERIFY - V    CHA'
1533  105C  4E 47                .BYT 'NGE - C    OK - K ;'
1534  106F  57 41         MSG6   .BYT 'WAIT - BUBBLE INIT;'
1535  1082  43 4F         MSG7   .BYT 'COMPLETED '
1536  108C  2D 20                .BYT '- POWER DOWN;'
1537  1099  41 43         MSG8   .BYT 'ACTIVES;'
1538  10A1  41 2F         MSG9   .BYT 'A/D RANGE ERR;'
1539  10AF  52 41         MSG11  .BYT 'RAM DEPLETED;'
1540  10BF  53 54         MSG12  .BYT 'STOP-BUBBLE FULL;'
1541  10CD  53 54         MSG14  .BYT 'STOP-TIME;'
1542  10D7  43 48         MSG15  .BYT 'CHAN      METHOD;'

1544  10E7                ATEMP  *=*+1
1545  10E8                TEMP   *=*+1

1547  10E9                KOUNIT *=*+16
1548  10F9                ERRPTR *=*+1

1550  10FA  01            STOPTM .BYT 1              ;STOP SIMULATION WHEN CLOCK = STOPTM
1551  10FB  00            CLOCK  .BYT 0              ;MASTER CLOCK OVERFLOW

1553  10FC                COUNTP *=*+16              ;PERMANENT TIMER COUNTERS
1554  110C  03            N80    .BYT 3              ;WHEN # BLOCKS <= N80, START BUBBLE
1555  110D                TOTAL  *=*+3               ;RUNNING BUBBLE SUM
1556  1110  00            FTOTAL .BYT $00,$25,$00 ;MAX BUBBLE COUNT - 40
1556  1111  25
1556  1112  00
1557  1113                REGSTR                     ;SYSTEM BUBBLE REGISTERS
1558  1113  00 02         BKADD2 .WOR $0200          ;BLOCK ADDRESS - 1BLOCK=256 8-BIT WORDS
1559  1115  00 BC         BFADD2 .WOR BUF0
1560  1117  00            NBLKS  .BYT $00            ;NUMBER BLOCKS TRANSFERED - 1
1561  1118  00 02         BKADD  .WOR $0200          ;SEE ROCKWELL BUBBLE USER MANUAL FOR INFO
1562  111A  00 BC         BFADD  .WOR BUF0

1564  111C                DMAFLG *=*+1               ;DMA STARTED FLAG
1565  111D  C0 3F         LSTBLK .WOR $3FC0          ;LAST BLOCK IN BUFFER
1566  111F  00 37         BUFFER .WOR $3700          ;BEGINNING OF BUFFER

1568  1121                ;THESE USED TO CALCULATE BUBBLE ON/OFF TIMES
1569  1121                PDTIM  *=*+2               ;CURRENT POWER DOWN TIME
1570  1123                PUTIM  *=*+2               ;CURRENT POWER UP TIME
1571  1125                TIMDIF *=*+2               ;TIME DIFFERENCE
1572  1127                TPUTIM *=*+2               ;TOTAL POWER UP TIME
1573  1129                TPDTIM *=*+2               ;TOTAL POWER DOWN TIME
1574  112B                NPWRDN *=*+2               ;NUMBER TIMES THRU POWER DOWN LOOP
```

```
LINE # LOC     CODE      LINE

1575  112D             NPWRUP *=*+2          ;NUMBER TIMES THRU POWER UP LOOP

1577  112F  3E          UBKSZ1 .BYT 62        ;UNIT BLOCK SIZE - 2
1578  1130  3F          UBKSIZ .BYT 63        ;UNIT BLOCK SIZE - 1
1579  1131             SAVAA  *=*+2
1580  1133             SAVEA  *=*+1
1581  1134             SAVIX  *=*+1
1582  1135             SAVX   *=*+1
1583  1136             SAVEY  *=*+1

1585  1137             SCNT   *=*+1
1586  1138             TEMPA  *=*+1
1587  1139             BSAVX  *=*+1
1588  113A             C40FLG *=*+1          ;40 WORD COUNTER FLAG
1589  113B  20 03      CNT40  .WOR 800
1590  113D             C40    *=*+1          ;40 WORD COUNTER
1591  113E             B256NX *=*+1          ;256 WORD COUNTER

1593  113F  6C FC      DELTIM .WOR $FC6C     ;DELTA TIME = -4000 (MICRO SEC.)
1594  1141             PBFADD *=*+8          ;PORT BLOCK POINTERS
1595  1149             CPBLK  *=*+8          ;CURRENT PORT BLOCK
1596  1151             PBKPTR *=*+4          ;PORT BLOCK POINTERS

1598  1155  00         PRATE  .BYT 0,0,0,0   ;DIGITAL EXPECTED RATE
1598  1156  00
1598  1157  00
1598  1158  00
1599  1159  00         PORTBF .BYT 0,0,0,0   ;PORT BUFFER FILE
1599  115A  00
1599  115B  00
1599  115C  00
1600  115D  00 00      CRATE  .DBY 0,0,0,0,0,0,0,0 ;CHANNEL RATES ON INITIALIZATION
1600  115F  00 00
1600  1161  00 00
1600  1163  00 00
1600  1165  00 00
1600  1167  00 00
1600  1169  00 00
1600  116B  00 00
1601  116D  00 00      METHOD .DBY 0,0,0,0,0,0,0,0 ;CHAN SAMPLING METHOD
1601  116F  00 00
1601  1171  00 00
1601  1173  00 00
1601  1175  00 00
1601  1177  00 00
1601  1179  00 00
1601  117B  00 00
1602  117D             STORGE *=*+16
1603  118D                    .END

ERRORS = 0000 <0000>
```

SYMBOL TABLE

SYMBOL	VALUE						
ACIA	C000	ACTIV	0500	ACTIV1	0534	ACTIV2	0540
ACTIV3	0580	ACTIV4	0504	ACTIV5	0517	ACTIV6	0521
ACTIV7	052E	ACTIV8	05CC	ACTIV9	0589	ACTIVA	0595
ACTIVB	05C3	ACURCY	00FE	ADBUSY	0008	ADJFF1	09C5
ATEMP	10E7	AUXCTL	AFFB	B256MX	113E	B80	0AC2
BBUSY	000C	BFADD	111A	BFADD2	1115	BKADD	1118
BKADD2	1113	BLANK	D0AF	BLKAL1	0AD0	BLKAL2	0AC9
BLKAL3	0ABB	BLKALC	0AA4	BPWRDN	0751	BPWRUP	079D
BSAVX	1139	BUBBLE	0AF0	BUBLE0	0643	BUBLE1	0AF3
BUBLE2	0B23	BUBLE3	0B36	BUBLE4	0B58	BUBLE5	0BA1
BUBLE6	0B31	BUBLE7	0B12	BUBLE8	0BAC	BUBNDX	0009
BUFFER	111F	BUFI	BC00	BUFO	BC00	C40	113D
C40FLG	113A	CBFADD	005A	CBKPTR	0036	CCHBLK	007A
CHANLS	0014	CHNI	0231	CHNPRM	0224	CLOCK	10FB
CNT40	113B	CNTNUS	097E	COUNTP	10FC	COUNTT	004A
COUNTV	0024	CPBLK	1149	CRA	B801	CRATE	115D
CRLF	0603	CRLOW	D0F1	CURBFA	000A	DCM	089C
DCM1	0896	DCM2	089A	DCM3	0885	DCM4	08D2
DDRA	AFF3	DDRB	AFF2	DELTIM	113F	DIFF	099E
DIFF1	09AE	DIGIRQ	0448	DMAFLG	111C	DUMP	05CD
EOC1	04D6	EOC2	04E3	EOC3	04F7	EOCIRQ	04C2
ER1	0991	ER2	0995	ERRMSG	0983	ERRPTR	10F9
FTOTAL	1110	GETVAL	05FC	GHEX2	061F	HEX	D306
IER	AFFE	IFR	AFFD	IMSG	0B99	INIT	05D0
IT1MR2	043C	IT1RQ	042F	IT1RQ1	043A	KEEP	0818
KEEP6	0875	KEEP7	088F	KEEP9	0976	KEPNDX	0046
KOUNTT	10E9	LEFT	D350	LINK	0AD8	LNKAL1	0BE1
LNKAL2	0BFB	LNKAL3	0C04	LNKALL	0BC9	LOC	0006
LOOP	064F	LSTBLK	111D	LSTTIM	00AA	LSTVAL	009A
MAIN	03A1	MAIN1	03B7	MAIN2	03BA	MAIN3	03C0
MAIN4	03E0	MAIN5	03E2	MAIN6	03B1	MAIN7	0404
MAIN8	041C	MAIN9	0421	MDEFF	0049	METHOD	116D
MISSN	0315	MISSN1	031D	MISSN2	036A	MISSN3	036C
MISSN4	0399	MISSN5	0332	MOK1	0284	MOK2	0298
MOK3	02CC	MOK4	02FA	MOK5	0254	MOK6	02BB
MOK7	030B	MONITR	C9F0	MSG1	1000	MSG11	10AF
MSG12	10BC	MSG14	10CD	MSG15	10D7	MSG3	101C
MSG3A	1022	MSG4	103C	MSG5	1048	MSG6	106F
MSG7	1082	MSG8	1099	MSG9	10A1	MSGADR	C606
MSGOUT1	05F0	MSGOUT	05ED	MVC	084C	MVC1	0846
MVC2	084A	MVC3	0867	N80	110C	NBLKS	1117
NCHNLS	0006	NEWBLK	0010	NLINK	000D	NPORTS	0007
NPWRDN	112B	NPWRUP	1129	NUMA	D2CE	OLDBLK	0012
OLDJEX	063F	OUTPUT	D2C1	PA	B800	PAIN	B7AF
PAOUT	B7AB	PB	B802	PBFADD	1141	PBIN	B7C6
PBIO	B7C8	PBKPTR	1151	PBOUT	B7C2	PCR	AFFC
PDTIM	1121	POINTR	0034	PORTA	AFF1	PORTB	AFF0
PORTBF	1159	PORTS	00FA	PRATE	1155	PRTJ	0269
PRTPRM	025C	PUTIM	1123	RCHEK	D1BC	RDBHDR	0671
RDTIM1	0806	RDTIME	07EB	READ	D1DC	REDOUT	D2B0
REGSTR	1113	RESET	0200	RKEP	D139	RUNIT	06B0
RUNIT1	06BB	RUNIT2	06C7	SAVAA	1131	SAVE	09C6
SAVE1	09F1	SAVE2	09F7	SAVE3	09D6	SAVE4	0A13
SAVE5	0A37	SAVE6	0A45	SAVEA	1133	SAVEX	1134
SAVEY	1136	SAVX	1135	SCNT	1137	SDIFF	0048
SEND	B7DC	SET2	021A	SET3	0221	SETUP	020B

SYMBOL TABLE

SYMBOL	VALUE						
SHFTRG	AFEA	SLFTST	05EC	SP1	0614	SPACES	0611
STOPTM	10FA	STORGE	117D	T1CNTR	AFEA	T1HC	AFF5
T1HL	AFF7	T1LC	AFF4	T1LCHW	AFEB	T1LL	AFF6
T2CNTR	AFEC	T2HC	AFF9	T2LCHW	AFAD	T2LL	AFF8
T3CNTR	AFEE	T3LCHW	AFEF	TCNT	048F	TCNT1	0493
TCNT2	049C	TCNT3	04AD	TCNT4	04A6	TCNT5	0496
TCTL13	AFE8	TCTL2	AFE9	TEMP	10E8	TEMPA	1138
THSTIM	00DA	THSVAL	00CA	TIMDLF	1125	TIME1	0480
TIMERS	0672	TLINK	000E	TOHEX	060D	TOTAL	110D
TPUTIM	1129	TPUTIM	1127	UBKSIZ	1130	UBKSZ1	112F
VCM	08F6	VCM1	08F0	VCM2	08F4	VCM3	090D
VCM4	093A	VCM5	0906	VCM6	094E	VCM7	0961
VCM8	0926	VDIFF	0047	VIA1	0474	VIA2	0476
VIA3	047D	VIA4	0472	VIAIRQ	044B	WAITB	B73D
WRBHDR	0670	WRHDR	0A63	WRHDR1	0A7D		

END OF ASSEMBLY

Appendix B

DATA ACQUISITION BOARD DESCRIPTION

The objective of this appendix is to describe the Data Acquisition Hardware Board and its interface to the Rockwell System-65 minicomputer. The five functional areas, in the order discussed, are:

a. Signal/Buffering

b. Address Decode Circuitry

c. Analog-to-Digital Data Acquisition

d. Interval Timer

e. Digital Data Acquisition

Refer to the wiring diagram (Figure 15) for the following discussions.

Signal/Buffering

Signals from the System-65 are brought on the board via the 86-pin edge connector, labeled as P1 in Figure 17 (Ref 12: Chap 4, 4). Address lines A0 through A15, in addition to read/write (R/W) and Phase 2 (Ø2), are buffered through three 8T97 noninverting, single-direction, hex bus drivers (K1, K2 and K3 in Figure 17). The eight bi-directional data lines, D0 through D7, are buffered through two 8226 bi-directional, quad, inverting (for System-65 compatibility), tri-state bus drivers (K4 and K5 in Figure 17). The tri-state control for the data buffers is taken directly from the buffered R/W line, while chip-select comes from the address decoder.

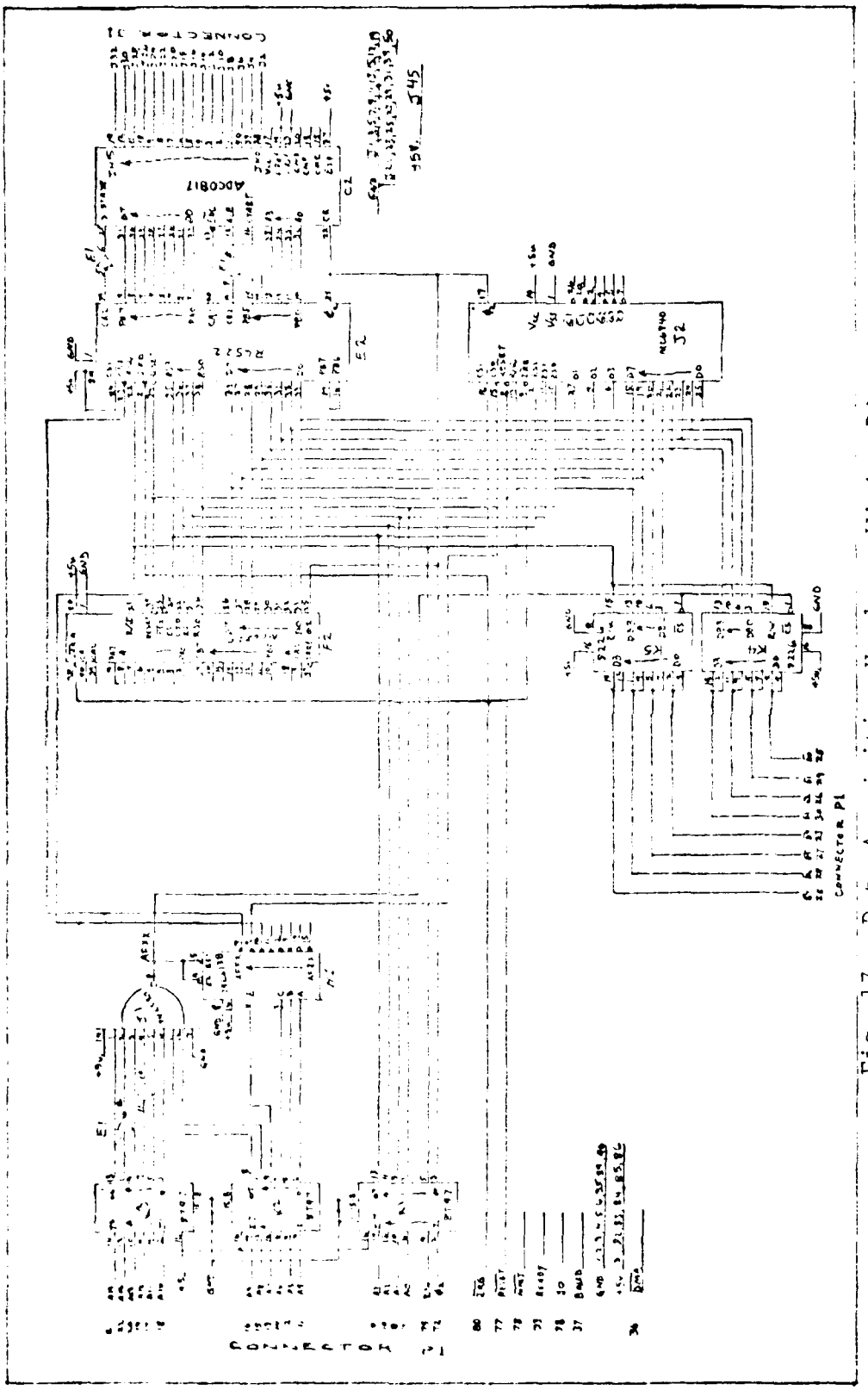


Fig. 17. Data Acquisition Hardware Wiring Diagram

Other signals available from the P1 edge connector are shown in Figure 17. However, only interrupt request (IRQ) and Reset are used at this time. Both IRQ and Reset are active low, nonbuffered, open-collector signals. All IRQ signals on the board are "or" wired, while all Reset signals on the board are "and" wired.

Address Decode Circuitry

The eight most significant bits (A8 through A15) of the buffered address lines, through appropriate inverters, are logically NANDED (7430 eight-input "NAND" gate) together to produce the AFXX signal (see socket J1 in Figure 17). This signal enables the 3-to-8 line decoder (74138) of socket H1. Address line A7 (used as chip enable for H1) and address lines A4 through A6 (used as data inputs to H1), define the outputs of H1 to be AF8X through AFFX. Individual devices use these signals, in addition to the remaining buffered address lines (A0 through A3), to define their unique addresses. The addresses are as follows:

AFF0 through AFFF	VIA
AFE8 through AFEF	INTERVAL TIMER
AFDC through AFDF	PIA

Analog-to-Digital Data Acquisition

This function is obviously the heart of the Data Acquisition Hardware Board. The two LSI, 40-pin chips chosen to implement this function are the ADC0817 analog data acquisition chip and the R6522 Versatile Interface Adaptor (VIA) chip.

The ADC0817 consists of a channel select latch that, through an analog multiplexer, selects one of 16 single-ended analog signals. The selected signal is then passed through a successive approximation analog-to-digital (A/D) converter whose eight-bit output represents a ratio of the full-scale voltage. The chip provides the capability to do signal processing between the multiplexer and the A/D converter input. This allows the addition of a sample-and-hold circuit if necessary.

The R6522 VIA, using peripheral ports A and B, provides the interface between the System-65 and the ADC0817. Each port has two peripheral control lines (CA1, CA2 and CB1, CB2) to do the required handshaking with the ADC0817. The lower four bits of port B are connected to the ADC0817 channel select latch. Port B's lower four bits are programmed as outputs. Control line CB2 is programmed to output a negative-going pulse when port B is written. The pulse on CB2 causes two actions: first, the negative-going edge causes the data on the lower four bits of port B to be latched into the channel select latch and second, the positive-going edge signals the A/D converter to begin conversion of the selected channel.

Port A's data lines are programmed as inputs and are tied directly to the A/D converter outputs. The end-of-conversion (EOC) signal from the A/D converter is wired to port A's control line CA1. EOC causes the data supplied by

the A/D converter to be latched into part A and, if enabled, an IRQ signal to be sent back to the System-65.

The two independent 16-bit timers on the R6522 VIA are wired such that the output of timer 1 is the input to timer 2. This allowed a hardware realization of the Mission Run Clock.

Interval Timer

The M6840 interval timer was used as a simulation tool for timing certain events. It contains three independent 16-bit counters, each capable of being programmed in one of four modes: continuous, single shot, pulse width compare, and frequency compare. Each timer can select as an input either the system Ø2 clock or an externally supplied clock/gate combination. Each timer has an individual output which can act as a programmable pulse timer signal or an individual IRQ signal. The chip also has a combined IRQ signal.

Digital Data Acquisition

Digital Data Acquisition is performed by the MC6820 Peripheral Interface Adaptor (PIA). The PIA is capable of interfacing to two peripherals through two eight-bit parallel ports, each with two control lines for handshaking. The PIA interfaces to the System-65 through the eight-bit data bus, three chip-select lines, two register select lines, two IRQ lines, the R/W line, the enable line, and the reset line. The data bus is tri-stated until the chip-select lines are enabled; the direction of data flow is determined

by R/W. The chip-select lines are enabled by the AFDX signal of the address decode and the buffered A3 and A2 lines. This places the VIA address between AFDC and AFDF. Buffered address lines A0 and A1 are wired to register selects zero and one, respectively, to determine what internal register is to be addressed. The enable line is wired to the buffered Ø2 line and is used to clock data into and out of the PIA. The reset line is "and" wired to the System-65. It is used as a power-on reset and as a master reset during system operation.

Each port can be programmed to act as an input or output. This will allow the ports to interface to digital input parameters during a mission run and interface to a magnetic tape or other mass-storage device to dump the collected data after a mission run.

VITA

Kenneth Lee Moore was born on 5 January 1950, in Yale, Oklahoma. He graduated from high school in Choctaw, Oklahoma in 1968. He attended Oklahoma State University, Stillwater, Oklahoma, and received a Bachelor of Science degree in Electrical Engineering. In May 1973, he entered the Air Force and served as Project Engineer, Electronic Warfare Division, Avionics Laboratory, Wright-Patterson AFB, OH, until 1977, when he became Chief, Hardware Evaluation Branch; 485L, TACC AUTO, Electronic Systems Division, Bergstrom AFB, TX. In July 1979, he entered the School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH. He is a member of Eta Kappa Nu.

Permanent address: Rt. 2, Box 31
Choctaw, Oklahoma
73020

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1 REPORT NUMBER: AFIT/GE/EE/80-6	2 GOVT ACCESSION NO: AD-A100 646
	3 RECIPIENT'S CATALOG NUMBER
4 TITLE (and Subtitle): AIRCREW INFLIGHT PHYSIOLOGICAL DATA ACQUISITION SYSTEM	5 TYPE OF REPORT & PERIOD COVERED: MS Thesis
	6 PERFORMING ORG. REPORT NUMBER
7 AUTHOR(s): Kenneth L. Moore, Capt USAF	8 CONTRACT OR GRANT NUMBER(s)
9 PERFORMING ORGANIZATION NAME AND ADDRESS: Air Force Institute of Technology (AFIT-EN) Wright-Patterson AFB, Ohio 45433	10 PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11 CONTROLLING OFFICE NAME AND ADDRESS: School of Aerospace Medicine Brooks AFB, Texas	12 REPORT DATE: June 1980 13 NUMBER OF PAGES: 123
14 MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15 SECURITY CLASS. (of this report): Unclassified 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

Fredric C. Lynch

FREDRIC C. LYNCH, Major, USAF
Director of Public Affairs

16 JUN 1981

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

Aircrew Physiology Monitor
Magnetic Bubble Memory
Microprocessor
Data Acquisition
Digital Recording Systems

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

A design is presented for a self-contained, man-mounted data acquisition system to sample and store 12 environmental and physiological parameters. The design consists of one-megabyte of nonvolatile magnetic bubble memory storage, 16 analog input channels, and four digital input channels, and is controlled by a 6502 microcomputer. Operational software was designed and simulation conducted on a Rockwell System-65 minicomputer augmented with two-megabits of magnetic bubble memory. Two types of data storage methods are

DD FORM 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Block 20 - Cont'd

examined - continuous (or pulse code modulation), and three variations of delta pulse code modulation for reduction of data storage.

Nonuniform sampling rates (or sampling jitter) caused by simultaneous sampling requests were investigated, and ways to reduce or eliminate the occurrence of jitter are also presented.