MESSICK, STEVEN LEE

A DIGITAL DATA COLLECTION SYSTEM FOR GEOMAGNETIC PULSATION STUDIES

UNIVERSITY OF ALASKA M.S. 1983

A DIGITAL DATA COLLECTION SYSTEM FOR GEOMAGNETIC PULSATION STUDIES

A

THESIS

Presented to the Faculty of the University of Alaska

in Partial Fulfillment of the Requirements

for the Degree of

MASTER OF SCIENCE

By

Steven L. Messick, B.S.E.E., B.A.

Fairbanks, Alaska

September 1983

A DIGITAL DATA COLLECTION SYSTEM FOR

GEOMAGNETIC PULSATION STUDIES

RECOMMENDED:

Chairman, Advisory Committee

Program Head

APPROVED:

Vice Chancellor for Research and Advanced Study

Sept. 2, 1983.

Date

Abstract

Geomagnetic pulsations are minute variations in the ultra-low frequency spectrum of the earth's magnetic field. In order to develop models to explain these phenomena we have first to record data of pulsation events. A flexible, microprocessor controlled data acquisition device suitable for operation at remote sites has been developed. This instrument demonstrates the feasibility of integrating a microprocessor controller with traditional geophysical sensors to create a conceptually simple, yet powerful, data recording device.

Much information can be extracted from the data thus recorded. Geomagnetic pulsations are subdivided into several frequency bands; a digital computer routine has been developed which determines the signal power present in each band as a function of time. Pulsations signals tend to be elliptically polarized; another routine has been developed to calculate several polarization parameters, such as ellipticity and handedness, from these data.

Table of Contents

List of Figures

List of Tables

Acknowledgments

A project such as the design and construction of the data collecting units described herein could not have been completed without the assistance of many individuals. I wish to thank the technical staff of the Geophysical Institute, University of Alaska, Fairbanks for their gracious support. I especially wish to acknowledge the efforts of Rudy Domke and Joe Knox who did much of the physical construction of the instruments. B. David Spell's expertise in digital electronics was invaluable. Brett Delana braved the cold and wind of the arctic to install the units at Cape Parry and Mould Bay. I also wish to thank the innumerable others who made comments and suggestions during this project.

For their continued encouragement and the direction of my graduate studies I thank my graduate advisory committee. My committee chairman, Dr. John Olson, spent many hours in consultation with me, for which I am deeply indebted.

I thank Ray Duncan of Laboratory Microsystems for allowing me to reprint the copyrighted FORTH nucleaus listed in

Appendix B.

My parents deserve the greatest thanks of all. They have always encouraged my interest in science, and in general learning. For that, and bearing up under the burden of raising me and my siblings, I am very grateful.

This project was funded by the National Science Foundation under grant number ATM81-11475.

Chapter 1

Introduction

This thesis describes an instrument designed to collect data generated by geomagnetic pulsation events and presents some techniques useful in the analysis of these data. This chapter provides a review of the geomagnetic spectrum and some of the issues involved in geophysical data collection. In chapter 2 the description of the data collection instrument begins with a discussion of the physical construction of the machine. The method of controlling the data acquisition is covered in chapter 3. The data returned is reviewed in chapter 4 and the last chapter offers some suggestions for improvement. For completeness, the appendices include schematics and program listings.

1.1 Geomagnetic Pulsations

Pulsations of the earth's magnetic field were first reported by Stewart (1861). Early investigations were

conducted with insensitive (by today's standards) equipment. As a result, there were not a large number of pulsation studies until the advent of induction coil magnetometers and the advocation of magnetohydrodynamic theory. Increasingly sensitive instruments and better explanations of the observations, coupled with the cooperative attitude of the International Geophysical Year, were responsible for a large increase in the amount of pulsations research (Saito, 1969). More recently, the introduction of digital data recording techniques has led to more informative data sets. Employment of computerized time series analysis algorithms has allowed much more information to be extracted from the data (Collier, 1983).

The range of frequencies in which geomagnetic events occur is rather broad, although much more restricted than the general electromagnetic spectrum. Jacobs, et al. (1964), divided the geomagnetic spectrum into a series of bands within which various types of geomagnetic pulsations were known to occur. These bands cover the range of pulsation periods from 0.2 seconds to 600 seconds. The series of continuous pulsation bands, as described by Jacobs, et al., is listed in Table 1. Pc1 pulsations, with a minimum period of 0.2 seconds, are the highest frequency events with which we are concerned. We want to record data within the continuous range of frequencies from Pc1 down to the lower

Table 1. Classification of geomagnetic pulsations.

Type		Period range (sec.)	Frequency range (mHz)
Continuous Pulsations	Pc1	0.2 - 5	5000 - 200
	Pc2	5 - 10	200 - 100
	Pc3	10 - 45	100 - 22.2
	Pc4	45 - 150	22.2 - 6.7
	Pc5	150 - 600	6.7 - 1.7
Irregular Pulsations	Pi1	1 - 40	1000 - 25
	Pi2	40 - 150	25 - 6.7

edge of the Pc5 band (600 seconds). Note, however, that the power contained within a particular event is strongly dependent upon the frequency content of the event. Figure 1, from Orr (1973), graphically depicts this dependence. We see that the higher frequency events tend to have lower energy content than lower frequency events. An excellent review of geomagnetic pulsations is given by Saito (1969).

Briefly, geomagnetic pulsations are ultra low frequency (ULF) band, semi-periodic variations in the earth's magnetic field. These events, detectable by ground-based magnetometers, fall into two major categories. Irregular pulsations (designated Pi) often occur at the beginning part of a magnetospheric substorm and tend not to have a well defined period. Continuous pulsations (designated Pc) are much more structured in their time variations (Jacobs, 1970).

Current theory for the formation of the long-period Pc events (Pc2-Pc5) holds that the local phenomena begin with a buffeting of the earth's magnetosphere by the solar wind (Southwood, 1974; Chen and Hasegawa, 1974). The Kelvin-Helmholtz instability generates wave packets at the magnetospheric boundary which couple to Alfven waves in the magnetosphere. These Alfven waves propagate along a resonant magnetic field line to the ionosphere.

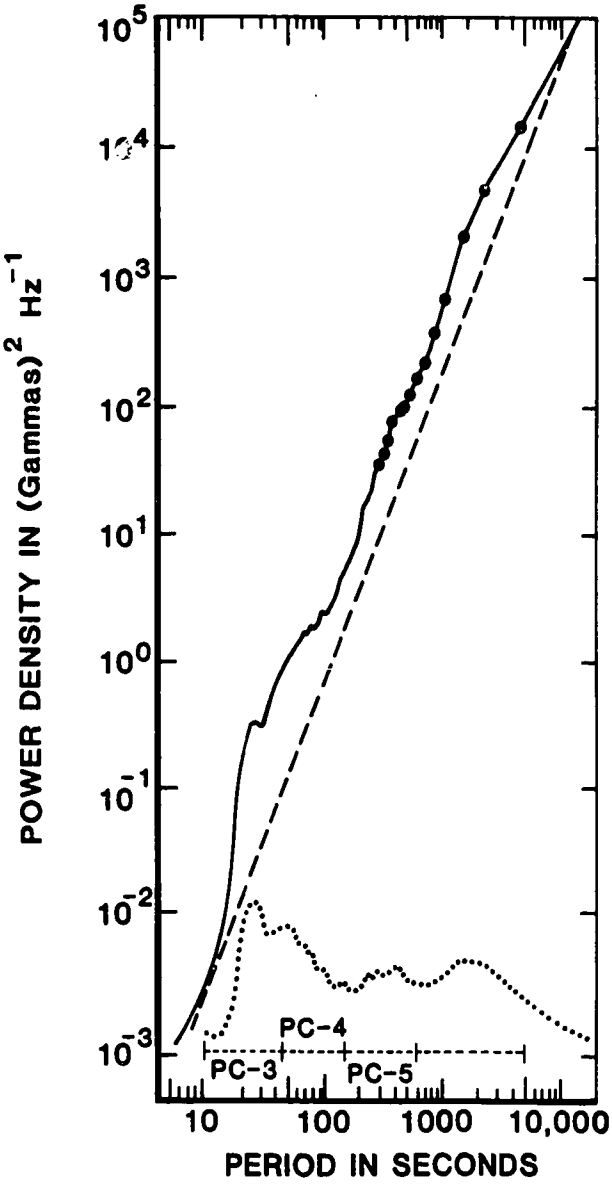


Figure 1. Power spectral density of geomagnetic variations in the total field (solid line). The dotted curve is a flattened spectrum obtained by removing a 7.1 dB/octave slope (dashed line) from the natural spectrum. From Orr, 1973.

Interactions with the electric fields in the ionosphere set up a secondary magnetic field which is detectable via ground based sensors (Hughes and Southwood, 1976).

The shorter period Pcl events, on the other hand, are apparently not connected to the solar wind. Rather, the ion cyclotron instability generates a circularly polarized electromagnetic wave by resonance between the wave and a stream of charged particles running in the direction of propagation of the wave. This wave packet is thought to bounce between conjugate points along a magnetic field line (Jacobs, 1970). The reflection from mirror points in the upper ionosphere results in the loss of enough electromagnetic energy to be detected by ground based instruments.

Pil and Pi2 pulsations are both associated with the onset of magnetospheric substorms. Short period irregular pulsations (Pil) appear to be caused by the fluctuations of current systems in the ionosphere. Changes in the ionospheric conductivity from bombardment by magnetospheric particles cause abrupt shifts in the current flow which, in turn, changes the magnetic field. Pi2 pulsations appear to be related to an impulse response of the magnetosphere to sudden changes in the magnetospheric convection rate. These changes produce an influx of wave and particle energy from

the magnetotail into the ionosphere. The transient response of the magnetospheric and ionospheric current systems causes the magnetic effects detected as Pi2 pulsations (Jacobs, 1971). Kan, et al., (1982) have given evidence that the Pi2 pulsation can be thought of as the successive reflection of the initial impulse between the ionosphere and the source region in the magnetotail.

Through analysis of the pulsation data collected by passive sensors it is possible to check the accuracy and consistency of theoretical models of the earth's magnetosphere and ionosphere; these analyses may also serve as a diagnostic of the plasmas and fields through which the waves propagate (Nishida, 1978).

1.2 Data Collection

Geomagnetic pulsations are detectable as variations in the strength of the earth's magnetic field at a particular location, or set of locations. This indicates that standard fluxgate and induction coil magnetometers are the sensors to use when recording pulsations data. Fluxgate magnetometers measure the magnetic field strength and produce a voltage level proportional to it. Induction coil magnetometers produce a voltage proportional to the change in the field

strength with respect to time. Both of these voltages are first amplified to an easily measurable level before being presented as the output of the instrument. These voltages are then recorded, in either analog or digital format, for analysis.

Magnetometer stations have been installed in many locations around the world, in both the northern and southern hemispheres. Occasionally, a single station is installed to measure local characteristics of the magnetic field; more commonly a chain of stations is installed to provide coherent data over a broad geographical zone. An example of such a chain is the Alberta Magnetometer Chain (Samson, et al, 1971). This chain of stations was located along a constant geomagnetic longitude; much information concerning the latitudinal characteristics of geomagnetic pulsations was gathered while it was in operation.

Our research effort is directed towards performing both latitudinal and longitudinal studies of pulsations. To meet this goal we need to have available data from a wide range of geomagnetic latitudes and longitudes. These data must also be coherent, i. e., have accurate and consistent timing information, so that the correlation of events at several stations may be investigated. The instrument described herein was designed to provide the data required for these

studies.

1.3 Data Recording

Geomagnetic pulsations appear to occur most abundantly near the auroral zone, at high latitudes. The sparse population in this area mandates the use of remotely located sensors. It would be possible to have the data sent to the analysis site via telemetry but this technique is usually impractical for a small research effort with a limited budget. Telemetry equipment is costly; rarely does the need for instant analysis of pulsation data justify its expense. It is more realistic to employ self-contained data logging units located at remote weather stations or other scientific sites and have the data routinely sent back for analysis. This is the method we decided to employ.

When collecting geophysical data, or any other type of data, one first has to decide upon the format of the records. Most data collection devices now use digital recording techniques although analog recording is useful in some circumstances. We chose to use a digital format partially for ease of analysis; the data do not have to be preprocessed in any way to be useful in digital computer analysis programs. Digital recording also has other

important benefits. The recorded signal is linear; it does not suffer the nonlinearity due to hysteresis which analog magnetic tape units impart. Digital recording also allows one to record a large dynamic range. The range is dependent only on the number of bits recorded for each datum.

Another concern of note in our research is the need for versatility. Some locations from which we decided to collect data already had available sensors which we could use, others required the installation of new sensors. As a consequence, more data channels were available at some locations than at others. We needed to be able to adjust the number of channels sampled and the rate of sampling for each site independently. Commercial data logging units are available which meet these criteria but they are rather expensive. They do not, however, include the capability for on-line analysis of the data; for example, they do not offer the ability to filter the data received to prevent aliasing. The maximum scan rate of most commercial units is lower than that which we may decide to use in the future and these units also use an inefficient, but convenient-to-access, technique for storing the data. For these reasons we decided to construct our own microprocessor controlled data acquisition units. By varying the program which controls the microprocessor we are able to adjust and control all these parameters.

Chapter 2

Physical Design Constraints and Construction

2.1 Introduction

This chapter discusses the issues involved in the design and selection of the hardware components for the data logging units. Specifically, I will describe the environmental parameters which needed to be considered, the selection of "off-the-shelf" components and the design of custom-made ones, and provide an overview of the hardware system.

2.2 Environmental Parameters

There is a common misconception that everything in the arctic is permanently frozen. I say this is a misconception because many people live in the far north, and surely they are not frozen. Actually, much of the equipment intended for installation in polar regions does not need to be

designed to work in extreme temperatures. The data logging units (field units) discussed herein were installed in established stations within heated buildings. However, the low humidity associated with arctic winters can cause problems. Often, static electricity will develop on equipment that is installed in a very dry environment. This static electricity could cause bursts of noise or short circuits within the equipment which may be harmful to some of the more delicate components. Having the equipment well grounded helps; so does placing it in a little-used area so that static caused by movement within the room does not develop. In extreme situations humidifiers or negative ion generators may need to be installed with the equipment.

2.3 Operational Parameters

The initial system proposal calls for four stations located in a diamond shape with College, Alaska as the southern tip. Mould Bay, N.W.T. is the northern tip of the array and the eastern and western boundaries are defined by Cape Parry, N.W.T. and Barrow, Alaska. Table 2 lists the geomagnetic locations of our field sites.

The remote locations of these stations call for equipment requiring a minimum of maintenance. Most stations are

Table 2. Station Locations.

Station	Geomagnetic Latitude	Geomagnetic Longitude	L Value
College, Ak.	64.72N	256.98E	5.44
Cape Parry, NWT	73.82	270.49	14.20
Mould Bay, NWT	79.25	256.41	38.79
Barrow, Ak.	68.63	241.52	8.10

located at arctic weather stations where technically trained people may not be present. To reduce the chances of operator error we decided to use cartridge tape for recording the data. Cartridge tapes require only that someone pull out one cartridge and insert another in its place. Power failure is a common occurrence at these remote locations. We opted for microprocessor control of the field units to allow for automatic recovery after power failure with a minimum data loss. Of course, the operator will have to restore the power. No other maintenance should be needed.

Accurate timing of the data is mandatory since multi-station correlation of events is to be done. A common, accurate time base is available via the National Bureau of Standards GEOS time relay satellite. The timing signal from the satellite is received at each station, converted into a NASA standard 20-bit slow time code, and recorded as one channel of data. The signal is received by a Kinemetrics timer (Kinemetrics, 1981). The 20-bit time code produced by the timer is slowed from one time frame per minute to one time frame per hour by logic boards designed and fabricated at the Geophysical Institute. This slow code ensures that the time code will be properly recorded even at very slow sampling rates. See the discussion below for information on the determination of the sample rate.

2.4 Signal Characteristics

Geomagnetic pulsations fall in the period range from 0.2 to 600 seconds. A 10 Hz sample rate would be required to sample the data sufficiently fast to record the entire spectrum. However, due to the nature of the higher frequency Pcl signals, the sampling rate can be reduced, thus saving considerable data storage space. Most of the interesting polarization information contained in a Pcl event is carried as slow modulations of the ~0.5 Hz carrier wave. Analog preprocessing of Pcl band signals can eliminate the higher frequency carrier and provide estimates of the more slowly varying characteristics. ULF Spectrum Channel Cards (SCC's) have been designed which respond to these slow modulations. With the aid of the SCC's we are able to use a 0.1 Hz sample rate, thus saving a significant amount of data storage space. A detailed description of the SCC's is contained in the report by Olson (1982); a brief description of the cards is provided here.

The SCC's are a hardware realization of a common time-series analysis algorithm. The output of each card is a continuous estimate of the bivariate spectral matrix, **S**, formed from the x- and y-components of the magnetometer

signal (the x- and y-components are referred to as H and D components, respectively). When working with a time series $x_i(t)$ we estimate the spectral matrix as follows. Let $X_i(\omega)$ be the Fourier transform of the time series $x_i(t)$

$$X_i(\omega) = \frac{1}{T} \int_{-T/2}^{T/2} x_i(t) \exp[-i\omega t]\, dt$$

and let $|\mathbf{X}\rangle$ be the column vector of these transforms for N input signals

$$|\mathbf{X}\rangle = (X_1(\omega), X_2(\omega), \ldots, X_N(\omega))^T$$

where superscript T represents the transform operation. Then an estimate of the spectral matrix **S** is the frequency band average of the outer product of $|\mathbf{X}\rangle$ with its transpose $\langle\mathbf{X}|$

$$\mathbf{S} = E\{|\mathbf{X}\rangle\langle\mathbf{X}|\}$$

where E represents the averaging operation (Olson and Samson, 1980). By using a sliding window and examining only a portion of the time series we can calculate the individual elements of **S** as functions of time.

The four signals produced by each SCC provide a continuous estimate of the components of **S**, averaged over 20 seconds, from the H and D induction magnetometer signals at a particular frequency. We do not, at present, record the Z

component of the induction magnetometer. Therefore, our estimated spectral matrix is only two dimensional. If we let the H magnetometer component be represented by $H_o\exp[-i(\omega t+\phi_1)]$ and the D component by $D_o\exp[-i(\omega t+\phi_2)]$ then the spectral matrix composed from these signals, adapted from the derivation of Fowler, et al. (1967) is given by

$$\mathbf{S} = \begin{vmatrix} H_o^2 & H_oD_o\exp[-i(\phi_1-\phi_2)] \\ H_oD_o\exp[i(\phi_1-\phi_2)] & D_o^2 \end{vmatrix}$$

Inspection of the spectral matrix terms allows us to collect data on Pc1 band pulsations without having to employ a high sampling rate. The diagonal terms provide measures of the power in each channel; therefore, Trace(**S**) gives us a continuous estimate of the power in the signal. Signal coherency information is provided in the off-diagonal terms: the real portion of the complex element S_{12} is an estimate of the "in phase" component of the cross spectral power; the imaginary component of S_{12} is the "out of phase" component of the power common to the H and D signals. If the spectral matrix is recorded at a number of frequencies all of the original polarization information, such as ellipticity and handedness, is still available for later analysis at a considerable savings of data storage space. Currently SCC's centered at 0.3 Hz and 1.0 Hz are in use at the College and Cape Parry sites.

2.5 Component Selection

To ensure versatility and minimize operator intervention we elected to use a microprocessor to control the acquisition and storage of the data. Of course, this meant we also needed an analog-to-digital (A/D) converter to transform the data to digital format. Most of these items are available as standard off-the-shelf components and there are a large number of similar products to choose from. Our requirements indicated, most of all, the need for reliable equipment. It is prohibitively expensive to make site maintenance trips when the field sites are located in remote locations, serviced only by chartered aircraft. But at the same time we needed to keep the cost of the entire unit to a minimum so we had to find the most cost-effective components available. Due to the increasing popularity of home computer systems many manufacturers are producing inexpensive, high quality components to work with the IEEE standard 696 interface buss.[1] We determined that a selection of components to operate with the IEEE-696 protocol met our requirements of reliability and price.

---

1. Also known as the S-100 buss.

There was also a different set of requirements which had to be met. The hardware had to be sophisticated enough to handle the software which was envisioned to run with it. For instance, we wanted to be able to sample up to 16 different analog channels simultaneously. Further, communication with a tape drive was mandatory and the ability to query the processor through a computer terminal was deemed very desirable. All these requirements were met by a combination of the Sierra Data Sciences (SDS) Slave Processor (Sierra Data Sciences, 1982) and the Dual Systems Analog-to-Digital Converter (Dual Systems Control Corp., 1981). The SDS processor includes 2 serial ports (for tape drive and terminal), 16 kilobytes[2] of non-volatile programmable memory, and 48 kilobytes of read/write memory. The Dual A/D converter is capable of sampling up to 32 different channels at a maximum rate of 40,000 samples per second. Samples are recorded with 12 bit resolution over a 10 volt range. These specifications leave ample room for expansion.

Our control program requires over five kilobytes of non-volatile memory for storage. SDS markets two versions of their processor board: a Master which is intended for controlling multi-processor systems, and a Slave. These

---

2. One kilobyte is 2^{10} 8 bit bytes.

units are very similar but there is an important difference. The Master includes provision for only 4 kilobytes of non-volatile memory. We therefore had to use the Slave version of the SDS processor board as the Master does not include adequate program storage space. However, the slave was designed to interface directly to the master without benefit of the electrical buss being used for communication by the rest of the digital system (i.e. the A/D converter). This necessitated construction of a custom-designed board to provide the proper interface between the processor and the system buss. The schematic for this simple interface is given in Appendix A.

2.6 Hardware System

A logical block diagram of the field unit is provided in Figure 2. It separates naturally into two parts: one for data acquisition and another for data storage. The data are acquired by standard geophysical sensors, i.e. riometers, fluxgate and induction coil magnetometers. The riometer and fluxgate magnetometer signals are transmitted directly to the storage subsystem. Induction magnetometer signals, too, go directly to storage but also are preprocessed by the SCC's as described above.

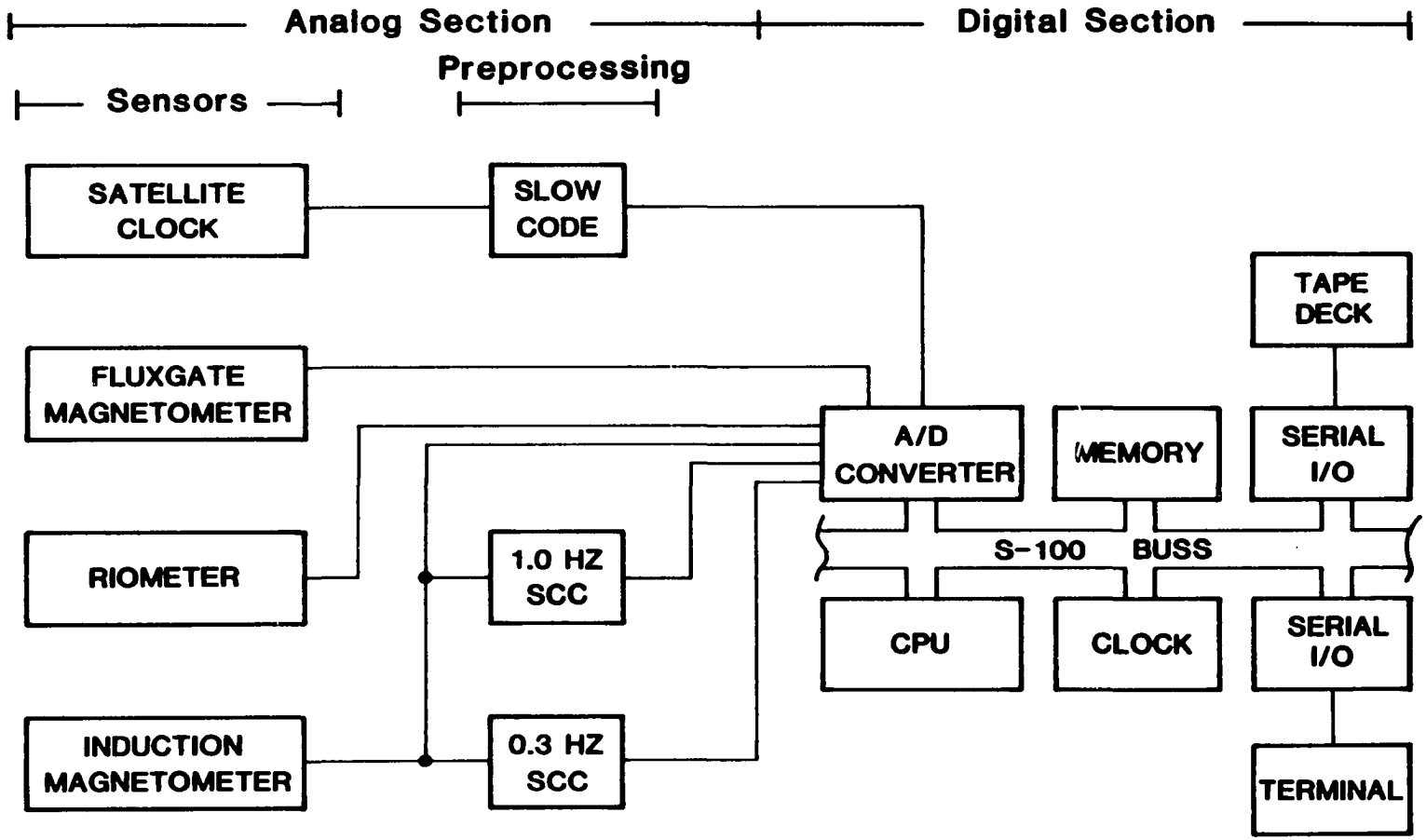


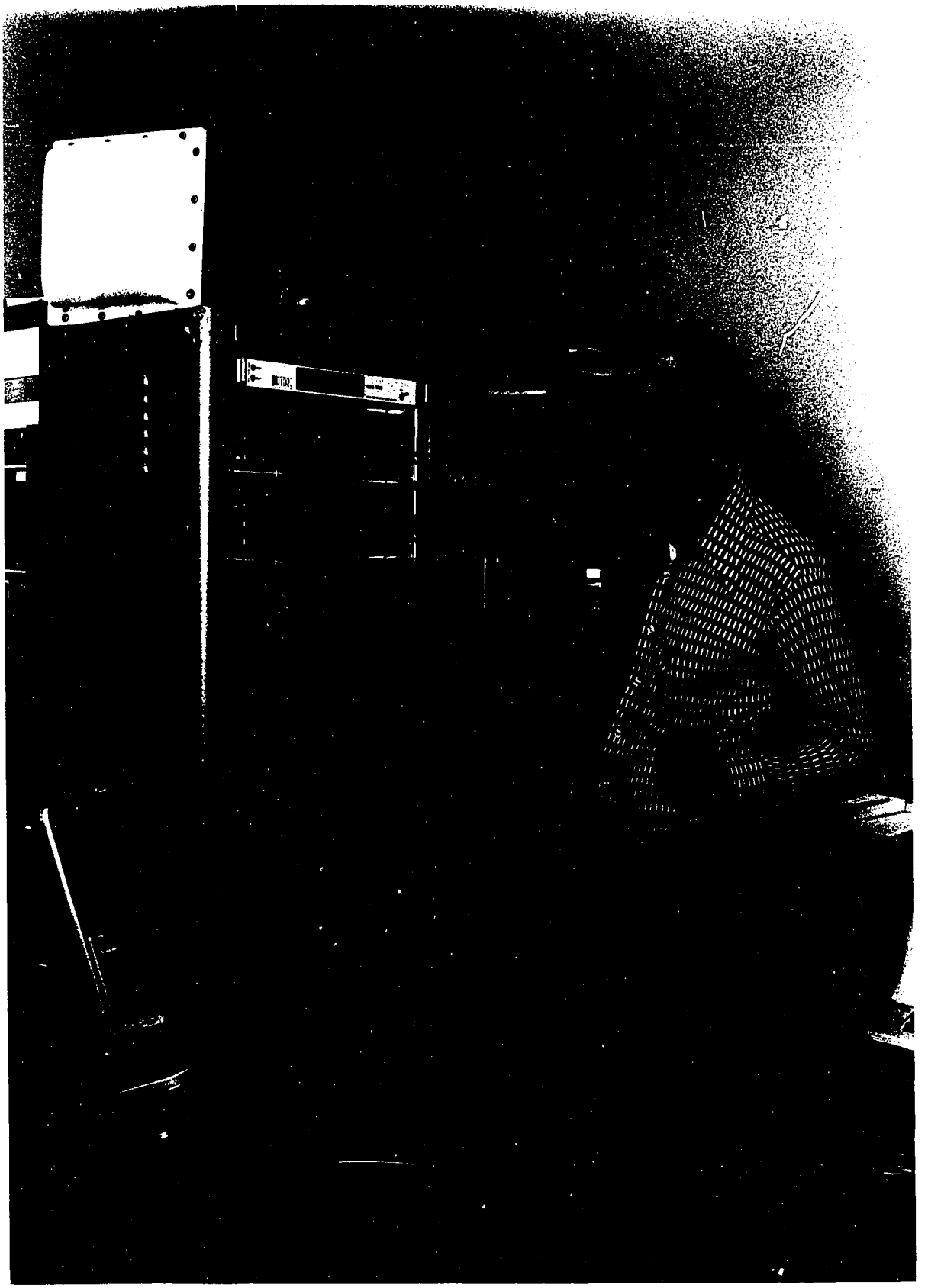
Figure 2. Hardware system block diagram.

All these signals (including the time code) are presented to the storage subsystem in analog format. The A/D converter transforms them to digital signals which are placed on the digital data buss. The CPU is responsible for storing the data temporarily in the internal memory then transferring a block of data to the cartridge tape interface unit for permanent storage on magnetic tape. The CPU receives its control instructions from the internal non-volatile memory. A block diagram of the control program is provided in Chapter 3 where it is discussed in detail.

During testing in our laboratory we were unable to detect any noise added to the signals by the hardware system. The signals did include noise generated by the sensors; however, when a pure sine wave was directly applied, bypassing the sensors, no noise was observed in the recorded signal. We conclude that the noise added by the hardware system is less than one digital level (2.44 millivolts). The noise added by the sensors will depend upon the individual characteristics of each sensor.

During the final check-out phase of construction the photograph of Figure 3 was taken. This photograph shows the instrument which was later installed at Cape Parry. For convenience in construction and transportation all components were mounted in a standard 19 inch wide equipment

Figure 3. Cape Parry Field Unit. Technician Rudy Domke is performing final checkout.



rack. The satellite time clock is mounted at the top of the rack. The two open panels below it contain the 20 bit time code generator and the two SCC's. The "front panel" for the instrument is located directly below. It has thumbwheel switches to select the number of channels to sample and the station location to record with the data, and an analog voltmeter with a channel selector switch for use in monitoring the analog signals being recorded. The next panel hides the cartridge tape deck power supply and logic boards; the tape drive itself is mounted directly below that. The final compartment holds the enclosure for the digital subsystem. It contains a separate power supply, the S-100 buss connectors, the A/D converter, the processor board, and the custom-built interface board. Power for the unit is filtered by a regulator to reduce the effect of voltage transients on the equipment. The antenna for the time clock, sitting on top of the rack in the picture, is mounted separately in the field.

Chapter 3

Software Development

This chapter examines software design issues. Program execution speed, development time, extensibility, and size all affect the selection of the computer language to be used. These each are discussed in detail. A review of the language selected, FORTH, follows. The chapter concludes with a discussion of the software used to control the data collection devices.

3.1 Execution Speed

Collecting geophysical data often requires a very fast sample-and-record operation. The minimum rate at which data are sampled is directly dependent upon the frequency of the event being observed. This rate, referred to as the Nyquist sampling rate, is determined by Nyquist's theorem to be twice the maximum frequency content of the signal. To reconstruct an event with a maximum frequency of one hertz

requires a sampling rate of at least two hertz.

It is undesirable to sample at too high a rate. A certain amount of memory space is required to store data samples. If sampling is performed at an arbitrarily fast rate then there is the possibility of exhausting the memory capacity. Most instruments have only a limited amount of memory space available. The sampling could easily occur so rapidly that all memory would be used up before a useful number of event periods had been recorded.

As mentioned in Chapter 2 we chose to sample the data channels once every ten seconds. This does not mean that for a system with ten channels one sample will be recorded from every tenth channel each second. This technique could be used but it introduces unnecessary complexity into the subsequent analysis of the data. The complexity arises because each channel would then have a phase shift proportional to the time difference from the base time, which would have to be considered during multichannel analyses. Instead, it is desirable to keep the channel rate as high as possible; restated, we want to record the data from each channel in as nearly the same instant as possible. The frame rate, or rate at which the data from a group of channels is recorded, is then set to the desired sample rate. In our system events with a period on the

order of 10-100 seconds are recorded 100 microseconds apart resulting in an error of less than one part in 10^5. This error is small enough to be neglected. Even though the data may be sampled at a low rate it is still necessary to sample multiple channels rapidly.

Thus, since the channel rate of our A/D converter is under software control, the language selected for implementation of the control program must be capable of rapid execution. Most compilers available for microprocessor-based systems produce code which will execute sufficiently fast for data collection purposes but do not support convenient development tools. The fastest programs are, of course, written in assembler language. However, coding in assembler language has another set of problems, discussed below. The language selected in this effort, FORTH, is capable of executing programs at a sufficiently fast rate without incurring the problems associated with assembler language. FORTH will be discussed further in Section 3.5.

3.2 Program Development Time

Often, during the course of a project, the need arises for a small, special purpose program which will be used either very few times or occasionally but by very few people. It

does not make sense to spend large amounts of time on the development of programs of this nature. Rather, it is preferable to have a set of software tools which can be quickly combined and extended to produce the proper program. Most high-level languages offer this capability to some extent. However, most assembler languages do not. This means that assembler language is not a good choice for developing new systems in a hurry. FORTH wholeheartedly embodies the philosophy of building on existing modules. As a result, program development time is reduced to a minimum.

3.3 Program Extensibility

Another aspect of software development to consider is that of program extensibility. Program extensibility refers to the ease of including additional capability within the framework of an existing program. For example, most programs written in a research environment initially perform the basic tasks which were recognized at the time the program was written. Additional tasks may subsequently be recognized and added to the code. More complex tasks may also be included after the success of the basic tasks has been demonstrated. This can become complex when working with assembler language programs. (Although it need not

be.) It can also become complex if a high-level language is used. However, it is generally easier to maintain a program written in a high-level language than the same program written in assembler language. FORTH programs tend to be easily extensible due to the manner in which FORTH code is written.

3.4 Program Size

When coding for a small system, program size may be a crucial factor in the selection of the language to be used. If large amounts of data are to be buffered in limited memory before being sent to secondary storage, the size of the program controlling the collection needs to be minimized. In this way a maximum amount of memory is available for buffering data. High-level languages generally do not produce code which requires the smallest amount of memory possible. Properly written assembler code will require comparatively little memory. In some cases, however, FORTH programs can be written which require less memory than do assembler language programs. This is due to FORTH's insistence upon using the same piece of code many times.

3.5 FORTH

After consideration of the selection criteria (execution speed, development time, extensibility, and program size) in terms of the programming systems available for microprocessors the language FORTH was chosen for implementation of the control program. FORTH is a small but flexible language designed for process control applications. Typical FORTH programs execute approximately an order of magnitude slower than the same program written in assembler language. However, this speed is still roughly 10 times faster than that of most other interpreted languages running in a microprocessor environment. It is slower than many compiled high-level languages but FORTH has another advantage here, too. It is possible, and very easy, to write the time critical portions of the program in assembler language without giving up any of the other advantages FORTH has.

FORTH programs are written by literally extending the basic FORTH system. New functions are written in terms of previously defined functions, or in assembler language. This makes for an easily extendible program. It also lends itself to rapid program development in a top-down,

structured manner. Although new functions must be defined in terms of old, during the program design phase new functions may be written in terms of undefined functions which are themselves defined later. After the functions have been entered into the FORTH system they are compiled into a pseudo-code which is very compact. Each function call requires only one computer word of memory space. The entire FORTH program is composed only of function calls plus the basic FORTH system so programs are memory efficient. However, there is a drawback to this technique: the basic FORTH system must be included in every program, no matter what size. In other words, there is no concept of linking previously compiled program segments in FORTH. This is not a disadvantage for programs of a moderate size and it can be a real advantage for very large programs. Thus, it is possible to have a FORTH program which requires less memory space than a comparable assembler language program.

FORTH has another advantage over most other languages. The basic system includes an interpreter which can be used to debug the code after it has been installed in an instrument (provided one can connect a computer terminal to the instrument). Having the ability to query the instrument is of primary importance during program development but may also be of value during installation and field testing.

3.6 Control Program

The software we have developed, which controls the acquisition of data, runs in a stand-alone environment within the instrument. It is stored in erasable, programmable, read-only memory (EPROM) and requires 8305 bytes of storage. Two-thirds of this space is taken by the basic FORTH system and approximately one-fourth of that is not required by the control program. The control program also requires almost 18 kilobytes of dynamic random access memory (RAM) to store a number of variables, some stack space plus two data buffers, each eight kilobytes in length.

The two data buffers are used to implement a double buffering algorithm. Double buffering involves an active buffer, within which new data are saved, and a passive buffer, which holds a previously collected block of data. The passive buffer is available for on-line analysis for as long a time as is required to fill the active buffer. When the active buffer is filled, it becomes the passive buffer and the other buffer becomes active. This scheme also reduces program complexity.

The program is written almost entirely in FORTH, with a small amount of assembler code included. It is logically divided into four parts: some code to initialize the hardware and software, a monitor, a sampling routine, and a routine for transmitting a full buffer of data to secondary storage on magnetic tape. The initialization code is called when the instrument is powered up. It ensures that all the hardware devices are configured correctly and that the software variables are set to their initial values. Note that this minimizes the problems caused by power failures: configuring the hardware includes setting the position of the tape to just after the last data record. This is accomplished by writing a file mark to tape after each record is written then spacing backwards over the mark so that it will be written over when the next record is written. If, however, a power failure occurs the end of the data recorded on tape is marked by the file mark. This file mark is searched for during initialization. Of course, unused tapes must have first had a file mark written on them for initialization to proceed. The initialization code also reads the values of the thumbwheel switches on the front panel to determine the number of channels to record and the station location to be saved with each record. The monitor and sampler are asynchronous processes. The monitor continuously checks the status of the active data buffer:

when the active buffer is full it swaps the active and passive buffers and initiates the transmission of data in the passive buffer to secondary storage. The active buffer is filled by the sampler. Periodically, the system clock interrupts the monitor and causes the sampling routine to execute. It is responsible for collecting the data from the real world and storing it in the active data buffer. Figure 4 summarizes the control program in a block diagram.

The first version of the control program included no code to control aliasing. Samples were taken every 10 seconds and the raw data were saved in the data buffer. In the second version, however, anti-aliasing code was included. Samples are still recorded every 10 seconds but now a triangular window over 25 points is used to produce one datum. The windowing algorithm is covered more fully in Chapter 4.

The astute reader may be be concerned at this point that the timing mechanism will introduce errors due to clock drift or instability. This problem is eliminated by including the slow time code (see Chapter 2) as the first channel of the data record. The instrument's clock receives the time relayed from a satellite in geosynchronous orbit. This time is accurate to within a few milliseconds, the error being introduced by the signal propagation time.

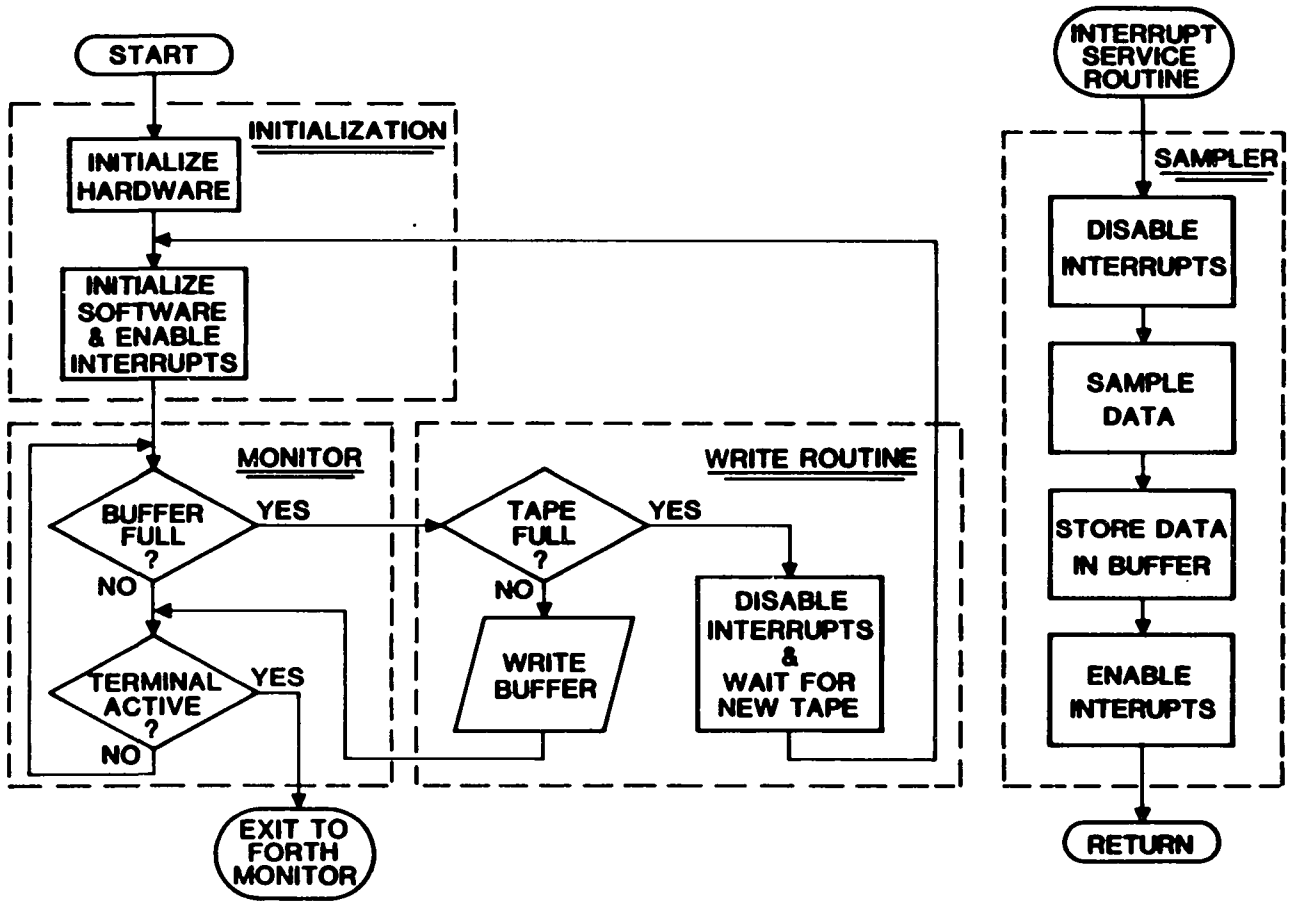


Figure 4. Software logic diagram.

Chapter 4

Data Recovery

4.1 Introduction

The field units for Mould Bay and Cape Parry were constructed and installed in 1982. We have been receiving data tapes continuously since these units were installed. In this chapter I will describe the installation of these field units, the quality and quantity of the data returned, and present some useful analysis techniques.

4.2 Installation

After the field units were assembled and tested at the Geophysical Institute they were shipped to Cape Parry, N.W.T., Canada and Mould Bay, N.W.T., Canada. Both units were equipped to record a fluxgate magnetometer, a riometer, and the time code. The Cape Parry unit was also fitted with 1.0 Hz and 0.3 Hz Spectrum Channel Cards and a two component

(H and D) induction magnetometer; the Mould Bay unit was not. At Mould Bay we were allowed access to the magnetometer and riometer being used for another project so no additional sensors needed to be installed. The Cape Parry site required the induction coil sensors to be installed with the field unit; however, riometer and fluxgate signals were available on-site. Installation was completed in mid-October, 1982. An additional field unit is currently being assembled. It will be installed later this year in Barrow, Alaska.

The units at Cape Parry and Mould Bay are maintained by personnel stationed at the Canadian arctic weather stations located at Cape Parry and Mould Bay. They are responsible for changing data tapes and mailing recorded tapes back to the Geophysical Institute for analysis. In addition, for the first version of the control program the operator had to manually reset the microprocessor after a power failure. Power failure recovery has been automated in the current version of the control program.

4.3 The Data

We have been receiving data tapes from both sites continuously since the units were installed. The data recorded in 1982 are rather sporadic due to the above mentioned problem with the power fail recovery mechanism. After that problem had been identified, and the operator instructions modified accordingly, the data became more continuous. Unfortunately, we have been unable to process the data on the early tapes from Mould Bay. The tape unit developed a problem in encoding the data to be written to tape and our tape drives are not able to decode the information. We are currently seeking assistance from the manufacturer in extracting the data from the Mould Bay tapes. The tape unit was replaced when the revision to the control program was installed and appears to be operating satisfactorily.

We have been able to recover all the information from the Cape Parry tapes and it has been copied to high speed mass storage for computer processing. During the process of copying and reformatting the data files it was discovered that a programming error had occurred. It was the cause of a subtle problem: many of the records stored on tape (but

not all) were missing the first sample of each channel in a block of data. Linear interpolation was used to estimate the values required for each data channel. The time code could not simply be interpolated. Fortunately, it is highly redundant and a few heuristics sufficed to generate the missing value. This error has been corrected.

One problem remains with the Cape Parry installation which we have not corrected. The induction magnetometer signals and, therefore, the Spectrum Channel Card signals experience a periodic pulse overlaid upon the signal. These pulses, identified as emanating from the GOES system transmitter, are 93 seconds in duration and occur 12 minutes apart. The interference signal has a small amplitude but it is prominent in the sensitive induction magnetometer signal. It is carried through the installation's DC power supply; the only way to eliminate this interference would be to use a separate power supply for the magnetometer.

I mentioned in chapter 2 that some arctic (or antarctic) installations have difficulty with static buildup due to the low humidity of these regions. It is noteworthy that our stations have experienced no severe problems in this regard. There is, occasionally, one bit dropped from or added to a data word (these are easily identified in the time code). This may be caused by static but may also be

due to other phenomena (such as tape head misalignment, X-ray bombardment, etc.).

In summary, the quantity of data received through May 1983 is low: approximately 53 percent of the data we could have gathered was lost due to power failures at Cape Parry; none of the data from Mould Bay during this period of time has been recovered. The quality of the data we did receive, however, appears to be high. There is an occasional odd (very high or very low) value in the data, but these are infrequent: less than 1 per 500 samples. Since the replacement of the defective tape deck at Mould Bay the rate of data return from that station has been comparable to that of the Cape Parry station. It is still too early to tell if automating the power failure recovery sequence has had a noticeable effect on the amount of data collected.

4.4 Aliasing, or How to Get Something From Nothing

Suppose your data logger was sampling at a 30 Hz rate and you were only interested in signals of 15 Hz, or lower, frequency. Further, suppose that a 50 Hz sine wave presented itself to your instrument. This signal, which doesn't exist by your criteria, would be sampled just often enough to be recorded as a 10 Hz sine wave. This phenomenon,

known as aliasing, is a very real problem when taking data with broad band sensors. There are two solutions. Obviously, the sampling rate could be increased so that the highest frequency signal present would be sampled at least twice every period. This would increase the amount of space required to store the data and could result in the need for a data logger which sampled at an impossibly high rate. Often, a more practical solution is to employ an antialiasing filter.

4.5 Antialiasing

Antialiasing refers to the process of reducing the problems caused by the undersampling of a signal. An antialiasing filter is a low pass filter which has a cutoff frequency equal to the maximum frequency to be recorded. We know from Fourier analysis that any continuous signal can be represented as a series of sines and/or cosines of harmonics of the lowest frequency contained in the signal. A low pass filter works by reducing the amplitude of the higher frequency terms in this series to zero, in the ideal case. A real filter will reduce the higher order components significantly, but probably not to zero.

The only filtering performed with the first version of the

control program was that provided by the SCC's. The SCC's adequately processed the higher frequency signals and eliminated the possibility of aliasing in their respective channels. For those channels which were not filtered by the SCC's, though, aliasing was a potential problem. We hoped, based on the observations made by Orr (1973), that any high frequency signals would be low enough in amplitude to cause no problem and that these signals would be very infrequent.

4.6 An Antialiasing Filter

We are currently more prepared to handle the vagaries of the natural spectrum. The major code modification between versions one and two (besides the bug fixes) was the inclusion of an antialiasing filter. We used a standard low pass, non-recursive digital filter implemented as part of the FORTH code which collects data samples. If we let y_k represent the k-th datum produced by the filter then the filter is the weighted average given by the function

$$y_k = \frac{1}{A_o} \sum_{i=-N/2}^{N/2} a_i x_i \qquad (a_i = a_{-i})$$

where the x_i are the original data points, the a_i are the weighting coefficients, and A_o is a normalization factor

equal to the sum of the coefficients. The coefficients were chosen to give a triangular window centered about the time at which the sample is considered to have been recorded. In this filter N = 25. To retain a 10 second period between samples and still have 25 data points to average together to produce one sample we needed to increase the sampling rate from once every ten seconds to once every four tenths of a second.

The frequency domain transfer function $H(\omega)$ is represented in Figure 5. $H(\omega)$ is the ratio of the amplitude of the signal produced by the filter to that of the signal presented to the filter as a function of frequency. We can see that higher frequencies are, indeed, much lower in amplitude; signals above the cutoff frequency (0.05 Hz) are reduced in amplitude by a minimum of 14 decibels. $H(\omega)$ is easily derived from the filter function. We transform the filter function to the frequency domain, then an application of the shift theorem gives us

$$\frac{Y(\omega)}{X(\omega)} = \frac{1}{A_o}\sum_{n=N/2}^{N/2} a_n \cos(n\omega T) + \frac{i}{A_o}\sum_{n=N/2}^{N/2} a_n \sin(n\omega T)$$

where T is the period between samples (0.4 seconds) and our transfer function is the magnitude of this ratio

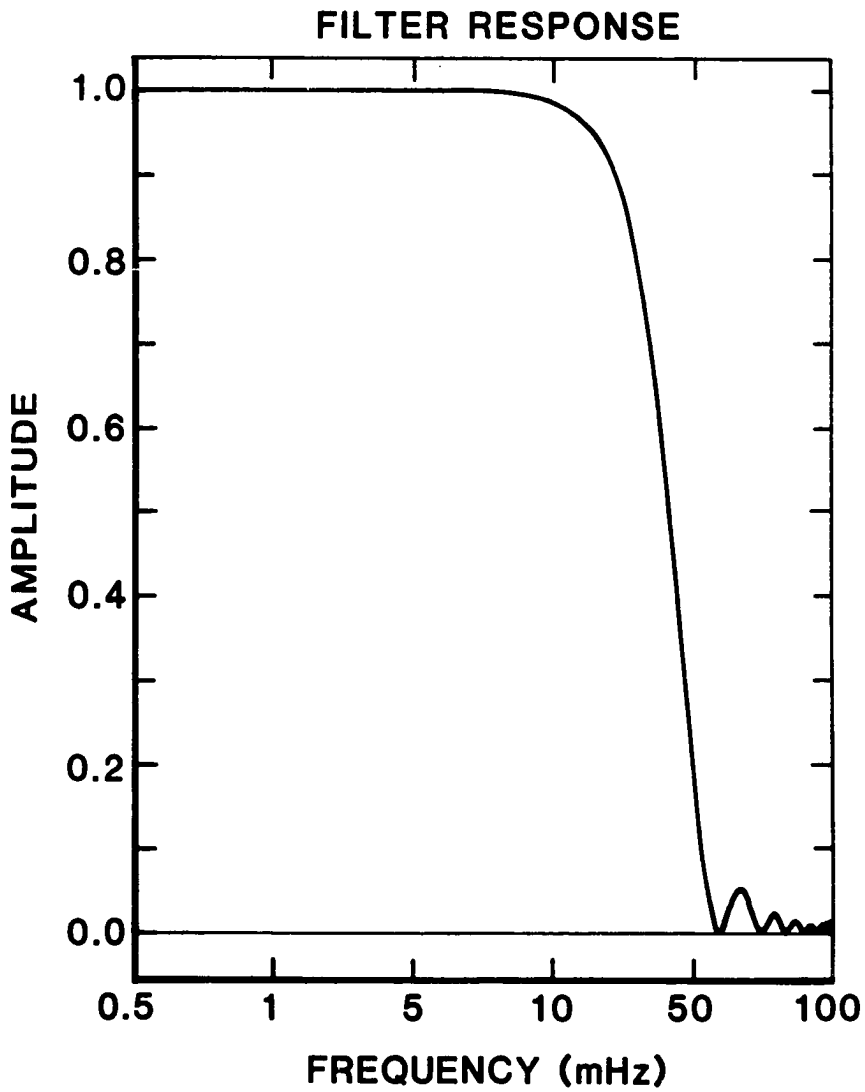


Figure 5. Antialiasing filter frequency response. Amplitude is in arbitrary units.

$$H(\omega) = \left| \frac{Y(\omega)}{X(\omega)} \right|$$

4.7 Pulsation Band Power

While trying to determine which portions of the data set warrant further analysis it is handy to have available tools to manipulate the data and restructure the information contained within them. One such tool which has been developed displays the power in the individual pulsation bands as it varies over a period of time.

As explained in chapter 2 the Spectrum Channel Cards provide a continuous estimate of the spectral matrix **S** for the signals detected by the induction magnetometer within a particular frequency band. The signal power P within that band can then be determined by

$$P = \text{Trace}(\mathbf{S})$$

This provides us with estimates of the power at 1.0 Hz and 0.3 Hz as functions of time.

Data from a typical day is shown in Figure 6. The top eight traces are the output of the 1.0 and 0.3 Hz SCC's. H, D, C, and Q refer to H and D magnetometer components and real and imaginary components of the off-diagonal terms of **S**, respectively. HL and DL are the induction magnetometer

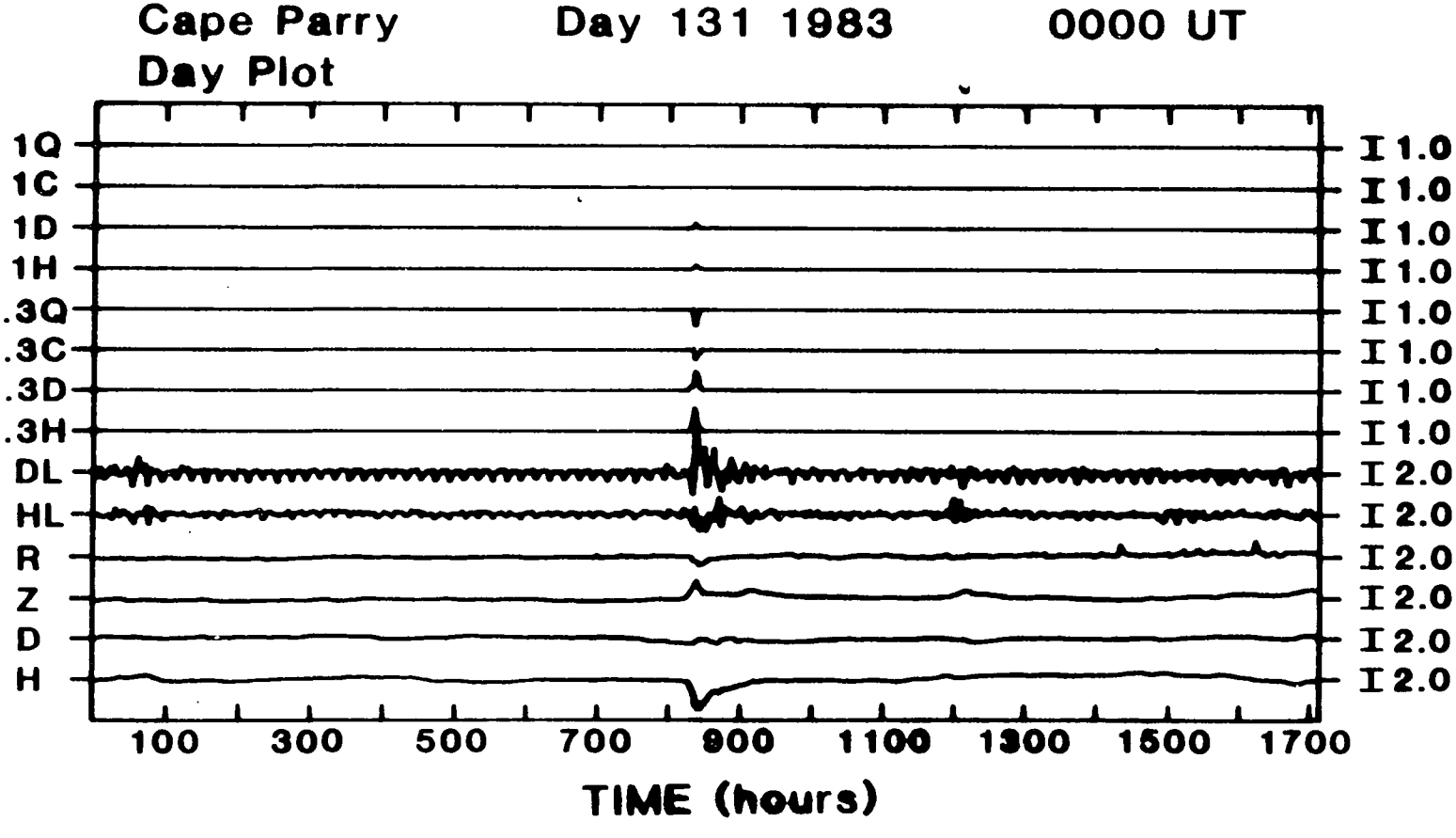


Figure 6. Data from a typical day at Cape Parry. Amplitude is in arbitrary units.

signals, R the riometer signal, and H, D, and Z are the three components of the fluxgate magnetometer.[3] Its pulsation band power is plotted in Figure 7. The frequency bands correspond to those given in Table 1, except there is no entry on the graph for the Pc2 band (the annotation on the left of each trace refers to the center frequency of each band, in milli-Hertz). The signal at 0820 UT is of a Pi emission. The power spectrum follows the general trends given in Figure 1: the lower frequency bands contain much more power than do the higher frequency bands. The plots represent the time domain convolution of a rectangular window one half hour in width with the data. To perform the convolution the window was shifted by five minutes after the estimates of the power in the Pc bands were computed, then the next estimates were found. This results in an inevitable broadening of the band power in the vicinity of the pulsation signal. The power in each band increases at about 0800 UT but the signal does not actually begin until 0815 UT. These plots represent the relative changes in power with respect to some base level. The 12 minute interference recorded in the induction coil magnetometer signals contribute to the power but, since this interference is continuous, its contribution to the power of the signal is

---

3. Z is defined to be positive in the downward direction, H is positive southward, D is positive westward.

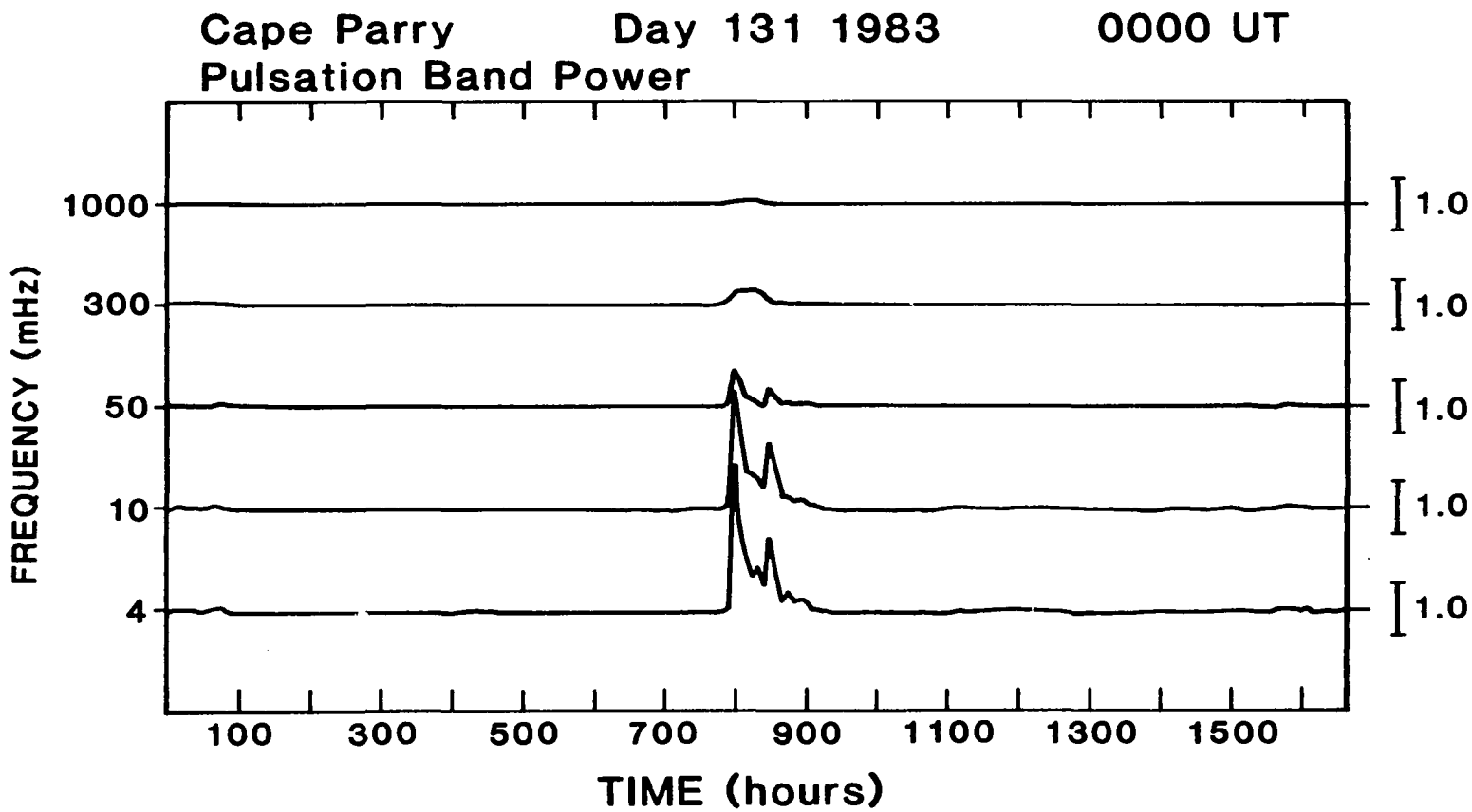


Figure 7. Pulsation band power of the data in Figure 6. Frequencies represent the center frequency of the continuous pulsation bands of Table 1. Amplitude is in arbitrary units.

removed when the base level is subtracted.

4.8 Polarization Parameters

Another tool, more useful in the detailed study of particular events, calculates the polarization parameters of a signal based on the magnetometer recordings for a specific period of time. Since these parameters depend upon the estimates of the spectral matrix, the output of the Spectrum Channel Cards is found most useful here.

Signals can be divided into two classes: those which are completely polarized (the polarization parameters are independent of time) and those which are completely unpolarized. Geomagnetic pulsations generally consist of both a polarized component and an unpolarized one (Fowler, et al., 1967). Figure 8 gives a schematic representation of the polarization ellipse of an arbitrary elliptically polarized signal. The general theory for working with partially polarized waves and its application to geomagnetic pulsation analysis is given by Olson and Samson (1980). Given an estimate of the spectral matrix (either from the Spectrum Channel Cards or via the equations in Chapter 2) the following estimates of the polarization parameters may be found. The relative phase of the signal, ϕ, is derived

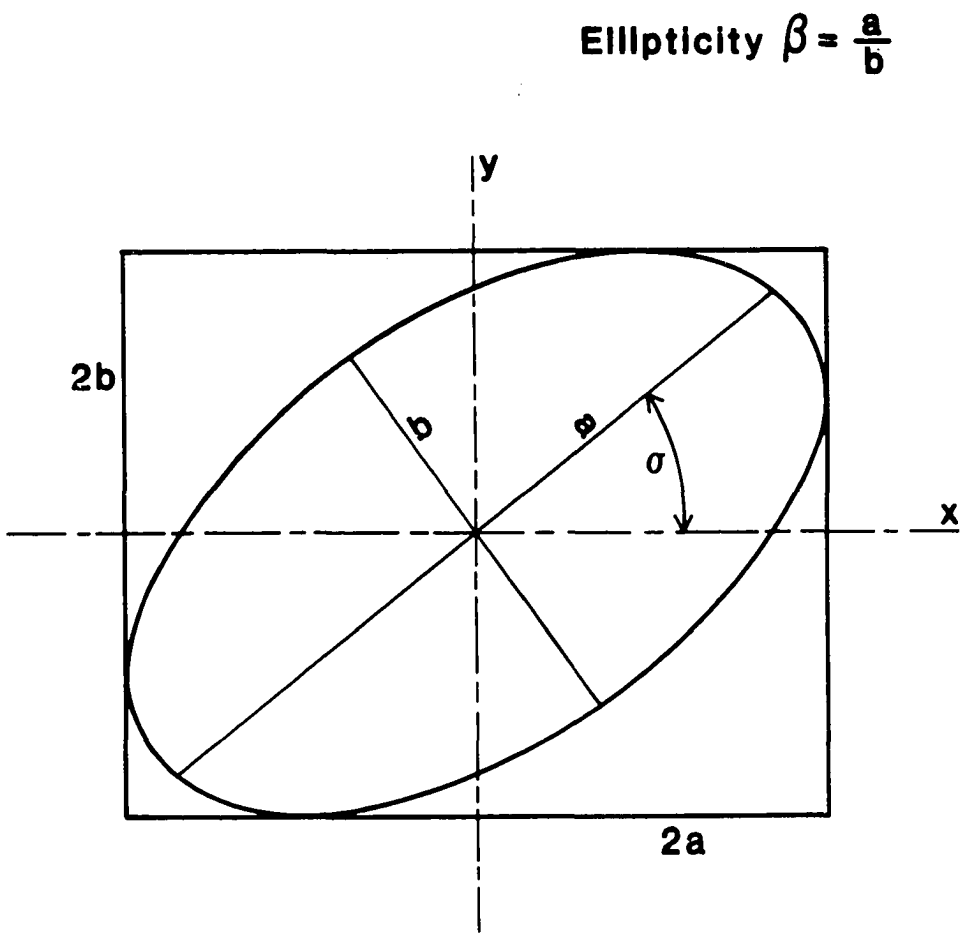


Figure 8. Schematic polarization ellipse for an arbitrary elliptically polarized wave depicting ellipticity and orientation.

from

$$\tan(\phi) = \frac{\mathrm{Re}(S_{12})}{\mathrm{Im}(S_{12})}$$

The direction of polarization (clockwise or counterclockwise) is determined by the sign of S_{12}, positive indicating clockwise polarization. The ellipticity, or ratio of major and minor axis of the ellipse, $\tan\beta$, is found from

$$\sin(2\beta) = \frac{2S_{11}S_{22}}{S_{11}^2 + S_{22}^2} \sin(\phi)$$

The orientation, σ, or angle between the major axis of the polarization ellipse and the X axis is derived from

$$\tan(2\sigma) = \frac{2S_{11}S_{22}}{(S_{11}^2 - S_{22}^2)} \cos(\phi)$$

These parameters yield physical insight into the nature of the pulsation event being studied. For instance, Southwood (1974) has shown that the direction of propagation of a resonant pulsation wave can be determined from the direction of polarization. A wave with clockwise polarization (looking downward) would propagate eastward south of the resonance in the northern hemisphere; north of the resonance the wave would propagate westward. The reverse holds for counterclockwise polarized waves (Southwood, 1974). Olson and Samson (1979) have shown that the relative position of a

Pc4 or Pc5 resonance (whether north or south of a sensor) can be determined using only data from a single station. Once the position is found the direction of propagation is determined by the direction of polarization.

Figure 9 illustrates the results of the above computations. The original data is that given in Figure 7 between the hours 0700-1000 UT. The phase trace (PHA) represents the phase of the signal (as recorded by the detectors) as the signal passes by the detectors. The trace labeled ELL depicts the ellipticity of the wave as a function of time. The handedness (HND), or direction of polarization, is just the sign of the ellipticity and shows whether the signal is right-hand, left-hand, or linearly polarized. A positive value indicates a right-hand, or clockwise (looking down), polarized signal. A signal with counterclockwise polarization has a negative handedness value. The fourth plot (TRA) is Trace(S) and represents the power of the signal as described above. The last plot (ORI) is the orientation of the major axis of the signal's polarization ellipse with respect to the X (or H, in this case) axis.

As the trace of the spectral matrix is related to the amount of power present in a signal we can use this plot to determine when the signal was strongest, or most

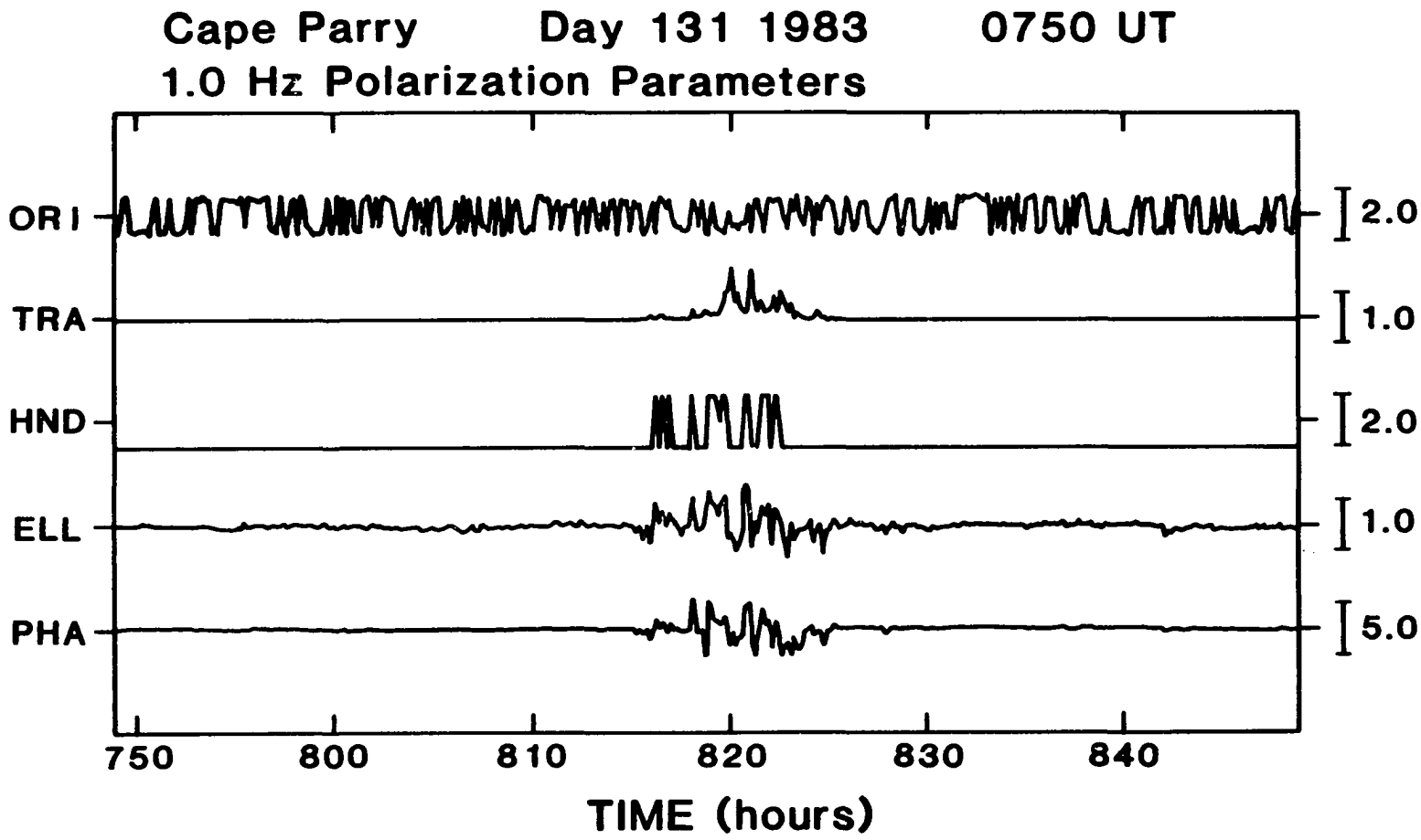


Figure 9. Polarization parameters of the data in Figure 6. Amplitude is in arbitrary units.

interesting. For instance, in Figure 9 there are two periods of time when the signal strength was much higher than usual. These fall on either side of 0820 UT. Immediately prior to the first strong period (which was at about 0819 UT) we see that the signal was rather incoherent: phase, ellipticity, and handedness all vary widely. As the signal strength increases, though, there is an abrupt shift from clockwise to counterclockwise polarization (as determined from the handedness plot) and the magnitude of the ellipticity decreases. (Note that in the figure ellipticity has both positive and negative values.) This suggests that the signal became more nearly linearly polarized. During this period the phase shifted from a small positive angle to a small negative angle; the orientation, which began as a negative angle, jumped to zero at the time when the signal power was at its maximum, then shifted back to a negative angle. Similar observations could be made for the signal peak at 0821 UT.

It should be noted that the availability of these polarization parameters will allow much more information to be extracted from pulsations data. Previous investigations had to rely on analog polarization analysis, such as hodograms, which were much less precise and did not yield as much information. It remains yet to be determined precisely what physical information may be extracted from digital data

using these polarization parameters. We are, however, confident that the SCC's will prove useful in analysis of high frequency pulsation events.

Chapter 5

Conclusions and Extensions

This thesis has presented an overview of an instrument which is in use collecting geomagnetic pulsations data, and a review of some of the data obtained with it. It demonstrates the practicality of integrating microprocessor controllers with traditional ground-based geophysical detectors. We are now in the process of cataloging the data and determining which events warrant further analysis. The signal analysis techniques presented in the previous chapter are valuable aids in this task.

The interactive capabilities of FORTH can be used to simplify the diagnosis of problems in the field. Attaching a computer terminal to the unit activates the FORTH interpreter. This allows check-out of the digital system and can be used to confirm that the sensors and tape deck are all operating properly. During installation, proper configuration can be determined quickly and easily. It would even be possible, with some extra programming effort beforehand, to make modifications to the control program in

the field. Those portions of the program which might need modification would have to be written such that their critical portions were resident in read/write memory rather than in non-volatile memory. See the discussion of vectored execution in Brodie (1981) for details.

There are a number of ways in which this basic design could be extended. It may be desirable to record higher frequency events from certain detectors while allowing others to continue recording events of a lower frequency while keeping magnetic tape consumption to a minimum. In this case the sampler would be programmed to execute at a rate equivalent to the greatest common multiple of the individual sample rates. Then, it would check each channel to see how often it is to be sampled and record samples only at the appropriate times. For instance, if channel one were to be sampled three times per second and channel two twice per second then the sampler would run six times per second. Channel one would be recorded every other time the sampler ran and channel two every third time.

Another extension which would be useful would be the inclusion of additional data preprocessing in the control program. During the design of the system whenever a choice of hardware was being made consideration was given to making the system as extensible as possible. There is sufficient

memory (both RAM and EPROM) left over that some analysis could be performed in the field. Also, at the present rate of data collection there is a minimum of 45 minutes required to fill a buffer, some of which could be used to perform analyses. Since the monitor and sampler run asynchronously the time when the sampler is not executing could be used by an expanded monitor which also would perform some analysis of previously collected data. In the first version of the control program the sampler required about 2 minutes of processor time to record a full buffer of data; the current sampler (not completely optimized) needs over 6 minutes. The time difference between this and the total time required for filling the data buffer would be the amount of time available for use by an analysis routine. If on-line analysis is to be performed it may be possible to use a sampler which does not include the antialiasing filter, thereby increasing the time available for analysis.

Event detection is an obvious application of this instrument using on-line analysis. We could, for instance, look for some particular correlation between two (or more) channels and record data only when we have a positive correlation. If the data just prior to an event were needed the double buffering scheme could be extended to a triple buffering technique. With three data buffers (one active and two passive) one passive buffer would be used for

analysis while the other would save the previously analyzed data. This would provide a minimum of 8 kilobytes of data prior to an event for later study.

This instrument could also be used in fields other than geomagnetic pulsation recording. By rewriting the sampling loop in assembler language and eliminating the antialiasing code the sample rate could easily be increased to one kilohertz for 10 channels. Due to hardware limitations the maximum rate would be one channel at 40 kilohertz. This is, however, quite rapid enough for many applications.

Moreover, the data collecting unit could be reconfigured to be more portable. The basic data logger is the section of Figure 2 under the "Digital Section" label. Physically, it consists of the front panel, tape deck, and microprocessor cabinet; with the satellite time clock, 26 inches of vertical space in a standard equipment rack are required for mounting (see Figure 3). These components could be housed in a smaller equipment cabinet and transported as needed to the field sites.

Appendix A

SDS Slave Processor to S-100 Buss Interface

This appendix provides electrical details of the logic board constructed to interface between the S-100 buss and the Sierra Data Sciences X-Buss on their Slave Processor. Understanding of the technical documentation for the CPU board and the A/D converter is assumed (Dual Systems Control, 1981; Sierra Data Sciences, 1982). Familiarity with the signals defined by the IEEE 696 standard (S-100) is helpful (see Libes and Gartez, 1981).

Figure 10 is the schematic diagram for the address and data buffer. U1 simply provides additional buffering for the address lines. The buffering of these lines provided by the CPU board is inadequate when other S-100 boards are added to the buss. Likewise U2 and U3 buffer the data buss between the X-Buss (J1) and the S-100 Buss (J2). In addition, these tri-state buffers are only enabled during

I/O operations. DOENA* and DIENA*[1] are normally high. When a read operation is requested, DIENA* goes low and enables U2, the input data buffer. DOENA* functions similarly for write operations. Address line A7 is used to disable buffering for internal (to the CPU board) I/O operations. All I/O ports resident on the CPU board have address greater than or equal to 80 hex. Therefore, when a read (or write) is requested with an address less than 80 hex it is referring to an I/O port which is not on the CPU board. OR-ing A7[2] and DIENA* (or DOENA*) ensures that the data buffer is only enabled for external I/O operations.

Generation of the signals required for operation of the A/D converter (and I/O requests) is provided by the top schematic of Figure 11. U4 buffers both the signals taken as input from the X-Buss (J1) and those produced as output to the S-100 Buss (J2). pWR* is a strobe which signals that the data output buss contains valid data and is the same signal on both busses. sOUT* is a status signal on the S-100 Buss which signifies that the CPU is writing data to the output data buss. It is high when both pWR* and IORQ* are low. sINP* functions in a similar manner for read operations. It

---

1. The "*" is used in place of overlining for those names which denote active-low signals.

2. Which is high for an address of 80 hex or above.

is generated when RD* and IORQ* and low. pDBIN* is the general S-100 read strobe. This signals a device that the CPU is ready to read data from it. It is high when the CPU is fetching an opcode from memory (e.g. IORQ* and M1* are low), or if it is reading from an input port (e.g. RD* is low). These signals are all that the A/D board requires for operation. None of the other buss signals are generated; if another board is added to the system, or the specific model of A/D board is changed, its schematic will have to be carefully examined to ensure that all required signals are present on the buss. If it requires control or status signals other than those generated in Figure 12 then additional circuitry will have to be added to supply those signals.

Two circuits, labelled "Power Supply" and "Power-On Reset," are at the bottom of Figure 12. The power supply provides a regulated five volt voltage source for use by components on the interface board. The power-on reset circuitry activates the signals RESET* and POC* for as long as it takes C5 to collect a charge each time the power is turned on. This provides for the resetting of all devices in the system and the automatic initialization of the control program.

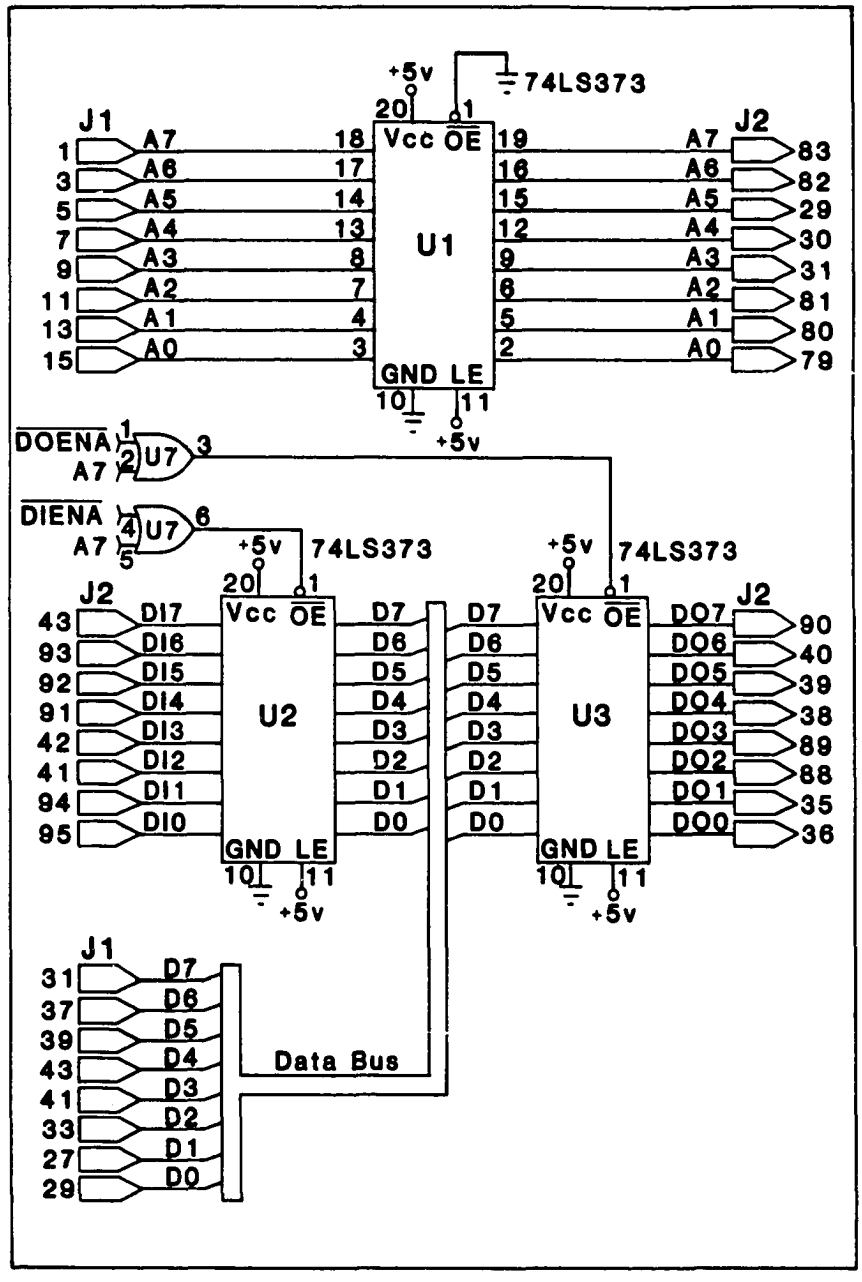


Figure 10. Schematic diagram of the Interface Board. Part A: buffer circuitry.

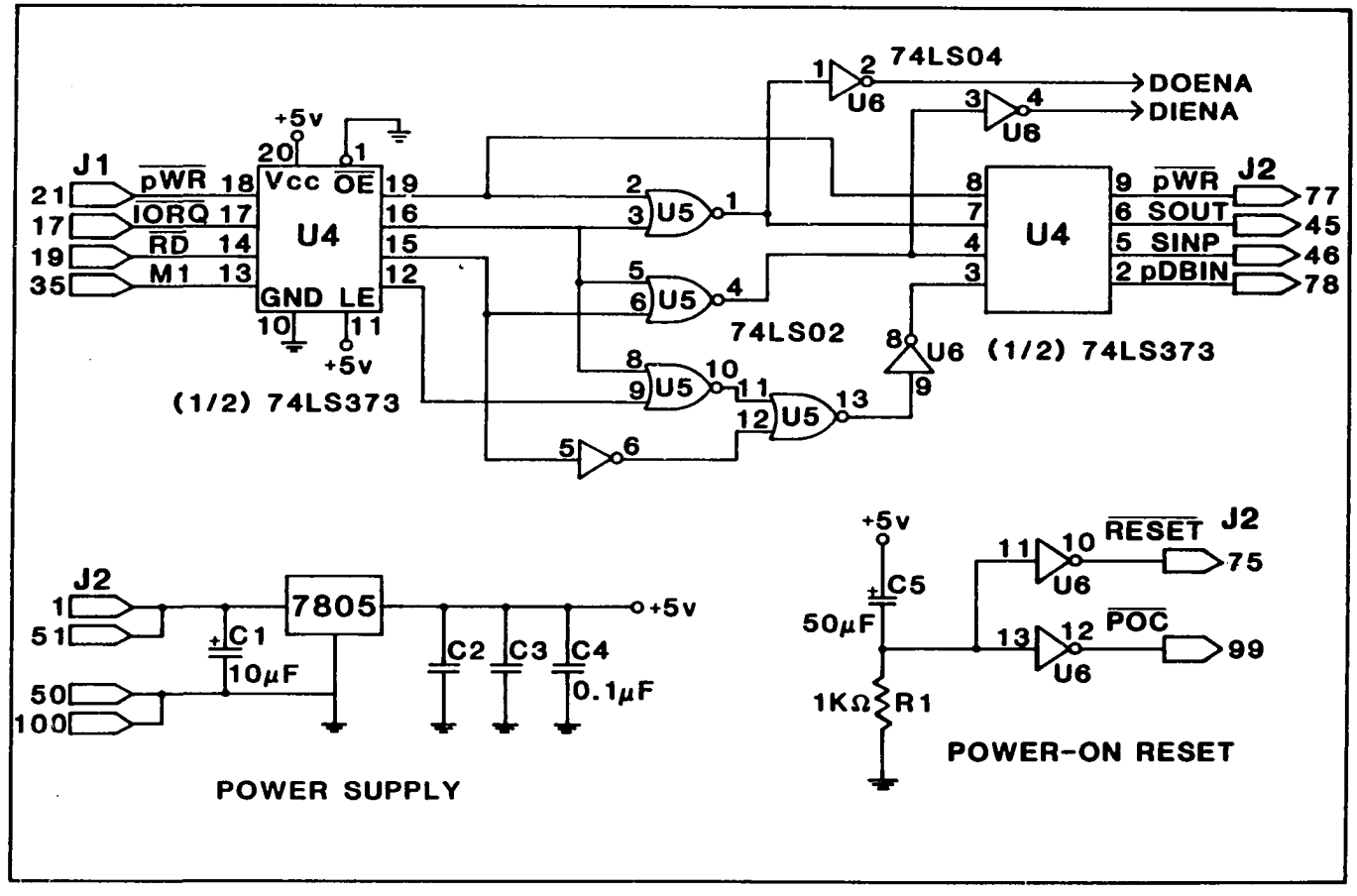


Figure 11. Schematic diagram of the Interface Board. Part B: buss signal transformation circuitry.

Table 3. Parts List for Interface Board.

Symbol	Component
U1, U2, U3, U4	74LS373
U5	74LS02
U6	74LS04
U7	74LS32
C1	10 microfarad capacitor
C2,C3,C4	0.1 microfarad capacitor
C5	50 microfarad capacitor
R1	1000 ohm resistor
J1	50 pin connector (to X-Buss)
J2	100 pin connector (to S-100 Buss)

Appendix B

Control Program Listing

This appendix provides detailed documentation of the control program. Familiarity with FORTH is assumed. Also, full understanding of the action of many of the FORTH words requires knowledge of the material contained in the technical documentation for the A/D converter, the cartridge tape drive, and the CPU board (Dual Systems Control, 1981; Alloy Engineering, 1980; Sierra Data Sciences, 1982).

B.1 Glossary

The following pages contain a listing of all FORTH words which make up the control program. The words which belong to the basic FORTH system are documented elsewhere (Laboratory Microsystems, 1982; Nautilus Systems, 1981; Derick and Baker, 1982). As is standard in FORTH documentation, each entry begins with the name of the word followed by the stack contents, both before and after

execution (the stack contents are the input to and the output from each word). A dashed line separates the input parameters on the left from the output parameters on the right. Beginning on the next line is a short English description of what each word does.

The following abbreviations are used to designate entities in the stack notation.

b	8 bit byte
d	4 byte double precision number
n	2 byte number
tf	truth value (zero is false)

#SAMP! b ---
Store number of samples being collected in proper place in header of data buffer.

#SAMP@ --- b
Return the number of samples being collected.

0-CHK-INIT --- n
Return the address of the location where the address of the checksum word for channel 0 is located.

0-PTR-INIT --- n
Return the address of the location where the beginning address of buffer 0 is saved.

1C&R b --- n
Send tape drive a 1 byte command and return Drive/Interface Status Word (DSW).

1-CHK-INIT --- n
Similar to 0-CHK-INIT, but for channel 1.

1-PTR-INIT --- n
Related to 0-PTR-INIT.

3C&R b b b --- n
Send tape drive a 3 byte command and return DSW.

A-CLOCK ---
Initialize baud rate generator for SIO A.

A-SIO ---
Initialize tape drive serial I/O port.

A/D-CH b ---
Tell A/D board which channel to sample next.

A/D-DATA --- n
Return 12 bits of A/D data.

A/D-GO ---
Start measurement of next sample on A/D board.

A/D-RDY ---
Return when next A/D sample is ready for reading. Not used.

A/D-READ --- d
Read current channel on A/D board, multiply by current weighting coefieient, and store temporarily.

B-CLOCK ---
Initialize baud rate generator for SIO B.

B-SIO ---
Initialize terminal serial I/O port.

BK-FMK ---
Backspace over last file mark.

BLINK-SELECT ---
Cause SELECT light on tape drive front panel to blink.

BUF! n ---
Save a data value and increment buffer pointer.

BUF* --- n
Returns the address of the word which keeps track of which buffer (0 or 1) is active.

BUF-FULL? --- tf
Return true if the current buffer cannot hold another set of data samples.

CARTRIDGE-IN? --- tf
Return true if a cartridge is inserted in drive.

CHA! b ---
Send b to the uart control port for the tape drive.

CHA-INIT ---
Setup tape drive communications.

CHB! b ---
Send b to the uart control port for the terminal.

CHB-INIT ---
Setup terminal communications.

CHKSUM n ---
Update checksum.

CHKSUM! n ---
Store new value of checksum.

CHKSUM# --- n
Checksum address. Returns the address of the word in the active buffer which contains the checksum.

CHKSUM@ --- n
Return current value of checksum.

CMD b ---
Send a command byte to tape drive.

CNT --- n
Returns the address of the word which holds the sample count.

CTC-INIT ---
Initialize interrupt timer and ISR.

DESELECT ---
Deselect tape drive. Used with RESELECT to cause SELECT light on tape drive front panel to blink.

DEV-INIT ---
Initialize programmable hardware.

DI ---
Disable interrupts.

DSW --- n
Return Drive Status Word composed of Interface Status (low byte) and Drive Status (hi byte) as returned from tape drive.

DUMP n1 n2 ---
Display contents of n2 memory locations starting with n1 rounded to next lower 16 byte boundary. Provided with 8086 FORTH. For debugging.

EI ---
Enable interrupts.

EISR ---
Not a FORTH word. Interrupt Service Routine (ISR) entry point. Transfers control to the FORTH ISR word.

EOT? --- tf
Return true if at physical end of tape.

FIND-EOD ---
Locate end-of-data after power failure. Power-up always causes rewind of tape drive. EOD is signaled by a file mark.

GET-STATUS --- n
Return current drive status in DSW format.

GREEN-ON ---
Turn on green light on front panel and leave it on.

INIT-TAPE ---
Set tape drive options. Of primary concern is the buffer length, specified by INI-MA.

IS-OK? n --- tf
Return true if last tape command executed without error.

ISR ---
Interruppt Service Routine control. Take another sample from each channel. If it is time to produce a filtered datum, then do it.

ISR1 ---
Not a true FORTH word. Label used by EISR to

transfer control to the FORTH interpreter.

ISREXIT ---
Restore the state of the machine after servicing an interrupt. Return from ISR.

KILL-TIME ---
Main time wasting loop. Reset stacks then do status check. If buffer is full, write it. If the terminal becomes active, go back to the FORTH interpreter. If the tape has been removed, set up for reinitialization.

NEXT-TRACK ---
Select next track and rewind tape. If at end of tape, wait for operator to replace tape.

NOT-OK? n --- tf
Return true if any error bits are on in DSW.

ON b ---
Turn the light(s) specified by the top nybble in b on.

ORANGE-BLINK ---
Cause amber light on front panel to blink.

OVER-RUN ---
Handle tape full errors. Wait for new tape, initialize drive and variables, then transfer to KILL-TIME.

PIO-INIT ---
Initialize front panel switches parallel I/O ports.

PIOA! b ---
Send a byte to program front panel switches parallel I/O port.

PIOA@ --- b
Get a byte from the front panel switches. (Number of channels)

PIOB! b ---
Send a byte to program front panel location selection switch parallel I/O port.

PIOB@ --- b
Get a nybble from front panel switch. (Location)

PISR ---
Program Z-80 interrupt service mechanism.

PTR --- n
THE buffer pointer. Returns the address of the next word in the active buffer to store a datum in.

READ ---
Read 8kb from tape to memory starting at location 6000 hex. For debugging.

RED-BLINK ---
Cause red light on front panel to blink.

RESELECT ---
Reselect tape drive. Used with DESELECT to cause SELECT light on tape drive front panel to blink.

REWIND ---
Rewind tape.

RW-SKIP-R --- n
Rewind then skip forward one record. Used to set status bits in DSW when determining where end-of-data is after power failure. Returns DSW.

S-RATE! b ---
Store sample rate (seconds per sample) in each channel's sample rate byte.

SAMPLE ---
Get and save a datum from each channel of the A/D board which has an instrument connected to it. Data are placed in a temporary buffer prior to being filtered.

SAVE ---
Form the weighted average of the previously saved data values for each channel and store them in the active data buffer then update checksum. This word, with A/D-READ, implements the antialiasing filter.

SEND-D ---
Send last filled buffer to tape.

SISR ---
Initialize ISR vector and hardware options.

SITE! n ---
Store site code in proper place in header of data buffer.

SKIP n ---
Skip forward (n positive) or backward (n negative) over n records on the tape. For debugging.

SKIP-FIL ---
Forward space over file mark.

SWAP-BUF ---
Toggle buffer selector prior to writing full buffer.

TAPE-ON? ---
Check tape cartridge and drive. Blink SELECT if no cartridge installed. Blink orange if tape is SAFE. Blink red if tape is full. In other words, make sure the tape unit is ready to write data to tape.

TMP --- n
Return the address of the first byte of the temporary data buffer.

TMP! d ---
Save a double precision value in the temporary buffer.

TP! b ---
Send a byte to the tape drive.

TP@ --- b
Get a byte from the tape drive.

TPR? ---
Return when tape drive port has a byte to be read.

TPX? ---
Return when tape drive port is ready to send a byte to drive.

UABORT ---
ABORT, user version. Control transfers here after FORTH has done its own initialization. Initialize the programmable devices, the tape drive, and variables, then transfer to KILL-TIME.

VALS --- n
Return the address of the first byte of the array

which contains the weighting coeficients for the antialiasing filter.

VAR-INIT ---
Initialize header for both data buffers.

W-PTR --- n
Returns the address of the beginning of the passive buffer.

WAIT5 ---
Waste about 1/2 second.

WR-CMD ---
Send Write command to tape.

WR-EOF ---
Write end-of-file (2 file marks) on tape.

WR-FMK ---
Write file mark on tape.

WRA! b b ---
Send a pair of control bytes to serial channel A (tape drive).

WRB! b b ---
Send a pair of control bytes to serial channel B (termianl).

WRITE-BUF ---
Swap buffers, write the full one to tape followed by a file mark, and backspace over the file mark.

WRITE-ENABLED? --- tf
Return true if cartridge is write enabled (not SAFE).

WRITE-IT ---
Send a buffer to the tape drive, being careful about things like end of track and end of tape.

B.2 Program Listing

A listing of the entire FORTH program used for control of the data logging instruments follows. The program is written using a version of FORTH by Laboratory Microsystems implemented for the Intel 8086 microprocessor. The complete control program is cross-compiled for the Z-80 processor using the Nautilus Systems FORTH Cross Compiler. Although the entire code is listed, the controller program occupies only screens 57 through 77. Screens 9 through 55 are the FORTH interpreter distributed by the FORTH Interest Group. Screen 56 is an extension provided with the Laboratory Microsystems 8086 FORTH system. Screens 78 and 79 contain some routines found to be useful during the debugging phase; they are retained for use during possible field diagnosis.

```
  Screen 0
 0 ( Stand-alone ROMable Z-80 FORTH source code )
 1 ;S
 2 for use with Nautilus Systems Cross Compiler
 3 ( Z-80 target version )
 4
 5 (c) 1981 by
 6 Ray Duncan
 7 Laboratory Microsystems
 8 4147 Beethoven Street
 9 Los Angeles, CA 90066
10 213-390-9292
11
12 Screens 57 through 77 copyright (c) 1983 by
13 Steve Messick
14 Geophysical Institute, Univ. of Alaska
15 Fairbanks, Alaska 99701

  Screen 1
 0 ( Explanation of load screens )
 1 ;S
 2
 3 ( Cross-compiling:                                              )
 4 ( First edit screen 9 to set origin and memory size.            )
 5 ( From CP/M, type: A>CROSSZ80 ROMZ80.SCR <return>.              )
 6 ( wait for system id.  then type:                               )
 7 ( 7 LOAD  <return>                                              )
 8 ( Target image is left in file IMAGE.COM on current disk.       )
 9 ;S
10
11
12
13
14
15

  Screen 2
 0 ( Screen printing utility SHOW )
 1 ( displays triads on list device )
 2 ( command format:  n1 n2 SHOW  )
 3 0 VARIABLE FF.FLAG
 4 : SHOW
 5   FF.FLAG @ 0=
 6   IF CR ." Does your printer have form feed capability? " KEY
 7   DUP EMIT 89 = IF 2 ELSE 1 ENDIF FF.FLAG ! CR ENDIF
 8   SWAP PRINTER
 9   DO I TRIAD
10      FF.FLAG @ 1 =
11      IF CR CR CR CR CR CR CR ENDIF
12   3 +LOOP
13   CONSOLE
14   ;
15
```

```
   Screen 3
 0
 1
 2
 3
 4
 5
 6
 7
 8
 9
10
11
12
13
14
15

   Screen 4
 0 ( System messages )
 1 empty stack
 2 dictionary full
 3 has incorrect address mode
 4 isn't unique
 5
 6 disc range ?
 7 full stack
 8 disc error !
 9
10
11
12
13 BASE must be DECIMAL
14 missing decimal point
15 Z-80 FORTH                                Laboratory Microsystems

   Screen 5
 0 ( System messages )
 1 compilation only, use in definition
 2 execution only
 3 conditionals not paired
 4 definition not finished
 5 in protected dictionary
 6 use only when loading
 7 off current editing screen
 8 declare vocabulary
 9
10
11
12
13
14
15
```

```
  Screen 6
 0 ( Error messages for Cross Compiler's Z-80 Assembler )
 1 16 bit register not allowed
 2 8 bit register not allowed
 3 address out of range
 4 immediate data value not allowed
 5 missing source register
 6 missing destination register
 7 illegal operation
 8 illegal operand
 9 instruction not implemented
10 illegal destination register
11 illegal source register
12 illegal condition code
13 register mismatch
14 destination address missing
15

  Screen 7
 0 ( Load screen for cross-compilation of Z-80 ROMable system )
 1
 2 DECIMAL
 3
 4 CROSS-COMPILE
 5 ( NO-LOG )
 6 09 79 THRU  ;S
 7
 8 This screen loads the Mark II controller for the field units.
 9
10
11
12
13
14
15

  Screen 8
 0
 1
 2
 3
 4
 5
 6
 7
 8
 9
10
11
12
13
14
15
```

```
  Screen 9
 0 ( ROMable Z-80 FORTH --- Equates )
 1   HEX C000 0 ORG/DB              7 TARGET-WIDTH
 2 ( HEX C000 7000 ORG/IMG  )
 3       0080 0080 ROM/RAM
 4 6000 MEM-END
 5
 6 1 EQU FIGREL     1 EQU FIGVER     0 EQU USRVER     20 EQU ABL
 7 0D EQU ACR       2E EQU ADOT      07 EQU BELL      7F EQU BSIN
 8 8 EQU BSOUT      10 EQU DLE       0A EQU LF        0C EQU FFEED
 9
10 040 EQU US                       0A0 EQU RTS
11 EM US - EQU INIT-R0              INIT-R0 RTS - EQU INIT-S0
12
13  83 EQU STATUS-PORT   001 EQU RDA  ( receive data available )
14  82 EQU DATA-PORT     004 EQU TBE  ( transmit buffer empty )
15 ;S

  Screen 10
 0 ( ROMable Z-80 FORTH --- initialization )
 1 ASSEMBLER
 2 ECLD JP  NOP  NOP  EWRM JP
 3 FORTH
 4 FIGREL C, FIGVER C, USRVER C, 0E C,
 5 HERE LABEL INIT-FORTH 0 ,
 6 BSIN , INIT-R0 , INIT-S0 , INIT-R0 , INIT-S0 , 01F , 0 ,
 7 HERE LABEL INIT-FENCE 0 ,
 8 HERE LABEL INIT-DP 0 ,
 9 HERE LABEL INIT-VOC-LINK 0 , BASE-36 Z80. , , HEX
10 THERE LABEL RPP  INIT-R0 THERE !  2 ALLOT-RAM
11 THERE LABEL UP   INIT-R0 THERE !  2 ALLOT-RAM
12 FORTH ;S
13
14
15

  Screen 11
 0 ( ROMable Z-80 FORTH --- Cold start )
 1 ASSEMBLER
 2 HERE LABEL ECLD
 3
 4         BC, # CLD1     LD
 5         SP, # INIT-S0  LD
 6         IY, # INIT-R0  LD
 7         NEXT JP
 8
 9 HERE LABEL CLD1  ] COLD [
10
11 FORTH ;S
12
13
14
15
```

```
  Screen 12
 0 ( ROMable Z-80 FORTH --- Inner interpreter and warm start )
 1
 2 ASSEMBLER
 3 HERE LABEL EWRM      BC, # WRM1 LD    NEXT JP
 4 HERE LABEL WRM1      ] WARM [
 5
 6 ASSEMBLER
 7 HERE LABEL DPUSH     DE PUSH
 8 HERE LABEL HPUSH     HL PUSH
 9 HERE LABEL NEXT      A, (BC) LD    BC INC    L, A LD
10                      A, (BC) LD    BC INC    H, A LD
11 HERE LABEL NEXT1     E, (HL) LD    HL INC    D, (HL) LD
12                      DE, HL EX     (HL) JP
13 FORTH ;S
14
15


  Screen 13
 0 ( ROMable Z-80 FORTH --- lit execute branch 0branch )
 1 FORTH DEFINITIONS
 2 CODE LIT          A, (BC) LD      BC INC          L, A LD
 3                   A, (BC) LD      BC INC          H, A LD
 4                   HPUSH JP        END-CODE
 5
 6 CODE EXECUTE      HL POP          NEXT1 JP        END-CODE
 7
 8 CODE BRANCH       HERE LABEL BRAN1                H, B LD
 9                   L, C LD         C, (HL) LD      HL INC
10                   B, (HL) LD      NEXT JP         END-CODE
11
12 CODE 0BRANCH     HL POP          A, L LD         A, H OR
13                   Z, BRAN1 JP     BC INC          BC INC
14                   NEXT JP         END-CODE
15 ;S

  Screen 14
 0 ( ROMable Z-80 FORTH --- (loop (do )
 1
 2 CODE (LOOP)       (IY) INC        NZ, XLOO1 JP    1 (IY) INC
 3                   HERE LABEL XLOO1                A, (IY) LD
 4                   A, 2 (IY) SUB   A, 1 (IY) LD    A, 3 (IY) SBC
 5                   M, BRAN1 JP     DE, # 4 LD      IY, DE ADD
 6                   BC INC          BC INC          NEXT JP
 7                   END-CODE
 8
 9 CODE (DO)         DE, # -4 LD     IY, DE ADD      DE POP
10                   (IY), E LD      1 (IY), D LD    DE POP
11                   2 (IY), E LD    3 (IY), D LD    NEXT JP
12                   END-CODE
13 ;S
14
15
```

```
  Screen 15
 0 ( ROMable Z-80 FORTH --- (+loop )
 1
 2 CODE (+LOOP)     DE POP          A, D LD         A, A OR
 3                  M, XPLOO1 JP    L, (IY) LD      H, 1 (IY) LD
 4                  HL, DE ADD      (IY), L LD      1 (IY), H LD
 5                  E, 2 (IY) LD    D, 3 (IY) LD    A, A OR
 6                  HL, DE SBC      M, BRAN1 JP
 7                  HERE LABEL XPLOO0               DE, # 4 LD
 8                  IY, DE ADD      BC INC          BC INC
 9                  NEXT JP
10                  HERE LABEL XPLOO1               L, (IY) LD
11                  H, 1 (IY) LD    HL, DE ADD      (IY), L LD
12                  1 (IY), H LD    E, 2 (IY) LD    D, 3 (IY) LD
13                  DE, HL EX       A, A OR         HL, DE SBC
14                  M, BRAN1 JP     XPLOO0 JR       END-CODE
15 ;S

  Screen 16
 0 ( ROMable Z-80 FORTH --- i j digit )
 1
 2 CODE I           E, (IY) LD      D, 1 (IY) LD    DE PUSH
 3                  NEXT JP         END-CODE
 4
 5 CODE J           E, 4 (IY) LD    D, 5 (IY) LD    DE PUSH
 6                  NEXT JP         END-CODE
 7
 8 CODE DIGIT       HL POP          DE POP          A, E LD
 9                  A, # 30 SUB     M, DIGI2 JP     A, # 0A CP
10                  M, DIGI1 JP     A, # 7 SUB      A, # 0A CP
11                  M, DIGI2 JP     HERE LABEL DIGI1
12                  A, L CP         P, DIGI2 JP     E, A LD
13                  HL, # 1 LD      DPUSH JP
14                  HERE LABEL DIGI2                L, H LD
15                  HPUSH JP        END-CODE                ;S

  Screen 17
 0 ( ROMable Z-80 FORTH --- (find )
 1
 2 CODE (FIND)      DE POP          HERE LABEL PFIN1
 3                  HL POP          HL PUSH         A, (DE) LD
 4                  A, (HL) XOR     A, # 03F AND    NZ, PFIN4 JP
 5                  HERE LABEL PFIN2                HL INC
 6                  DE INC          A, (DE) LD      A, (HL) XOR
 7                  A, A ADD        NZ, PFIN3 JP    NCY, PFIN2 JP
 8                  HL, # 5 LD      HL, DE ADD      (SP), HL EX
 9                  HERE LABEL PFIN6                DE DEC
10                  A, (DE) LD      A, A OR         P, PFIN6 JP
11                  E, A LD         D, # 0 LD       HL, # 1 LD
12                  DPUSH JP
13 ;S
14
15
```

```
  Screen 18
 0 ( ROMable Z-80 FORTH --- (find )
 1
 2                HERE LABEL PFIN3                CY, PFIN5 JR
 3                HERE LABEL PFIN4                DE INC
 4                A, (DE) LD      A, A OR         P, PFIN4 JP
 5                HERE LABEL PFIN5                DE INC
 6                DE, HL EX       E, (HL) LD      HL INC
 7                D, (HL) LD      A, D LD         A, E OR
 8                NZ, PFIN1 JR    HL POP          HL, # 0 LD
 9                HPUSH JP
10                END-CODE
11 ;S
12
13
14
15

  Screen 19
 0 ( ROMable Z-80 FORTH --- enclose )
 1
 2 CODE ENCLOSE    DE POP          HL POP          HL PUSH
 3                 A, E LD         D, A LD         E, # -1 LD
 4                 HL DEC
 5                 HERE LABEL ENCL1                HL INC
 6                 E INC           A, (HL) CP      Z, ENCL1 JR
 7                 D, # 0 LD       DE PUSH         D, A LD
 8                 A, (HL) LD      A, A AND        NZ, ENCL2 JR
 9                 D, # 0 LD       E INC           DE PUSH
10                 E DEC           DE PUSH         NEXT JP
11                 HERE LABEL ENCL2
12                 A, D LD         HL INC          E INC
13                 A, (HL) CP      Z, ENCL4 JR     A, (HL) LD
14                 A, A AND        NZ, ENCL2 JR
15 ;S

  Screen 20
 0 ( ROMable Z-80 FORTH --- cmove )
 1
 2                HERE LABEL ENCL3                D, # 0 LD
 3                DE PUSH         DE PUSH         NEXT JP
 4                HERE LABEL ENCL4                D, # 0 LD
 5                DE PUSH         E INC           DE PUSH
 6                NEXT JP
 7                END-CODE
 8
 9 CODE CMOVE      EXX             BC POP          DE POP
10                 HL POP          A, B LD         A, C OR
11                 Z, CMOVE2 JR    LDIR
12                 HERE LABEL CMOVE2
13                 EXX             NEXT JP         END-CODE
14
15
```

```
  Screen 21
 0 ( ROMable Z-80 FORTH --- u* )
 1
 2 CODE U*          DE POP          HL POP          BC PUSH
 3                  B, H LD         A, L LD         MPYX CALL
 4                  HL PUSH         H, A LD         A, B LD
 5                  B, H LD         MPYX CALL       DE POP
 6                  C, D LD         HL, BC ADD      A, # 0 ADC
 7                  D, L LD         L, H LD         H, A LD
 8                  BC POP          DE PUSH         HPUSH JP
 9                  END-CODE
10 ;S
11
12
13
14
15

  Screen 22
 0 ( ROMable Z-80 FORTH --- mpyx )
 1 ASSEMBLER
 2
 3 HERE LABEL MPYX
 4                  HL, # 0 LD      C, # 8 LD
 5                  HERE LABEL MPYX1
 6                  HL, HL ADD      RLA
 7                  NCY, MPYX2 JP   HL, DE ADD      A, # 0 ADC
 8                  HERE LABEL MPYX2
 9                  C DEC           NZ, MPYX1 JP    RET
10
11 FORTH
12 ;S
13
14
15

  Screen 23
 0 ( ROmable Z-80 FORTH --- u/ )
 1
 2 CODE U/          EXX             BC POP          HL POP
 3                  DE POP          A, L LD         A, C SUB
 4                  A, H LD         A, B SBC        CY, USLA1 JP
 5                  HL, # -1 LD     DE, # -1 LD     USLA7 JP
 6                  HERE LABEL USLA1                A, # 10 LD
 7                  HERE LABEL USLA2                HL, HL ADD
 8                  RLA             DE, HL EX       HL, HL ADD
 9                  NCY, USLA3 JR   DE INC          A, A AND
10                  HERE LABEL USLA3                DE, HL EX
11                  RRA             AF PUSH         NCY, USLA4 JR
12                  A, A OR         HL, BC SBC      USLA5 JR
13 ;S
14
15
```

```
  Screen 24
 0 ( ROMable Z-80 FORTH --- u/ )
 1
 2                 HERE LABEL USLA4                A, A OR
 3                 HL, BC SBC      NCY, USLA5 JR   HL, BC ADD
 4                 DE DEC
 5                 HERE LABEL USLA5                DE INC
 6                 HERE LABEL USLA6                AF POP
 7                 A DEC           NZ, USLA2 JR
 8                 HERE LABEL USLA7                HL PUSH
 9                 DE PUSH         EXX             NEXT JP
10                 END-CODE
11 ;S
12
13
14
15

  Screen 25
 0 ( ROMable Z-80 FORTH --- and or xor )
 1
 2 CODE AND        DE POP          HL POP          A, E LD
 3                 A, L AND        L, A LD         A, D LD
 4                 A, H AND        H, A LD         HPUSH JP
 5                 END-CODE
 6
 7 CODE OR         DE POP          HL POP          A, E LD
 8                 A, L OR         L, A LD         A, D LD
 9                 A, H OR         H, A LD         HPUSH JP
10                 END-CODE
11
12 CODE XOR        DE POP          HL POP          A, E LD
13                 A, L XOR        L, A LD         A, D LD
14                 A, H XOR        H, A LD         HPUSH JP
15                 END-CODE        ;S

  Screen 26
 0 ( ROMable Z-80 FORTH --- sp@ sp! rp@ rp! )
 1
 2 CODE SP@        HL, # 0 LD      HL, SP ADD      HPUSH JP
 3                 END-CODE
 4
 5 CODE SP!        HL, UP LD       DE, # 6 LD      HL, DE ADD
 6                 E, (HL) LD      HL INC          D, (HL) LD
 7                 DE, HL EX       SP, HL LD       NEXT JP
 8                 END-CODE
 9
10 CODE RP@        IY PUSH         NEXT JP         END-CODE
11
12 CODE RP!        HL, UP LD       DE, # 8 LD      HL, DE ADD
13                 E, (HL) LD      HL INC          D, (HL) LD
14                 DE PUSH         IY POP          NEXT JP
15                 END-CODE        ;S
```

```
  Screen 27
 0 ( ROMable Z-80 FORTH --- ;s leave >r r> )
 1
 2 CODE ;S          C, (IY) LD      IY INC          B, (IY) LD
 3                  IY INC          NEXT JP         END-CODE
 4
 5 CODE LEAVE       E, (IY) LD      D, 1 (IY) LD    2 (IY), E LD
 6                  3 (IY), D LD    NEXT JP         END-CODE
 7
 8 CODE >R          DE POP          IY DEC          IY DEC
 9                  (IY), E LD      1 (IY), D LD    NEXT JP
10                  END-CODE
11
12 CODE R>         E, (IY) LD      IY INC          D, (IY) LD
13                  IY INC          DE PUSH         NEXT JP
14                  END-CODE
15 ;S

  Screen 28
 0 ( ROMable Z-80 FORTH --- r 0= 0< + )
 1
 2 CODE R           E, (IY) LD      D, 1 (IY) LD    DE PUSH
 3                  NEXT JP         END-CODE
 4
 5 CODE 0=          HL POP          A, L LD         A, H OR
 6                  HL, # 0 LD      NZ, HPUSH JP    HL INC
 7                  HPUSH JP        END-CODE
 8
 9 CODE 0<          HL POP          A, H LD         A, A OR
10                  HL, # 0 LD      P, HPUSH JP     HL INC
11                  HPUSH JP        END-CODE
12
13 CODE +          DE POP          HL POP          HL, DE ADD
14                  HPUSH JP        END-CODE
15 ;S

  Screen 29
 0 ( ROMable Z-80 FORTH --- d+ d- )
 1
 2 CODE D+          EXX             BC POP          DE POP
 3                  HL POP          (SP), HL EX     HL, DE ADD
 4                  DE, HL EX       HL POP          HL, BC ADC
 5                  DE PUSH         HL PUSH         EXX
 6                  NEXT JP         END-CODE
 7
 8 CODE D-          EXX             BC POP          DE POP
 9                  HL POP          (SP), HL EX     A, A OR
10                  HL, DE SBC      DE, HL EX       HL POP
11                  HL, BC SBC      DE PUSH         HL PUSH
12                  EXX             NEXT JP         END-CODE
13 ;S
14
15
```

```
  Screen 30
 0 ( ROMable Z-80 FORTH --- minus dminus over )
 1
 2 CODE MINUS      DE POP          HL, # 0 LD      A, A OR
 3                 HL, DE SBC      HPUSH JP        END-CODE
 4
 5 CODE DMINUS     EXX             DE POP          BC POP
 6                 HL, # 0 LD      A, A OR         HL, BC SBC
 7                 HL PUSH         HL, # 0 LD      HL, DE SBC
 8                 HL PUSH         EXX             NEXT JP
 9                 END-CODE
10
11 CODE OVER       DE POP          HL POP          HL PUSH
12                 DPUSH JP        END-CODE
13 ;S
14
15

  Screen 31
 0 ( ROMable Z-80 FORTH --- drop 2drop swap dup 2dup )
 1
 2 CODE DROP       HL POP          NEXT JP         END-CODE
 3
 4 CODE 2DROP      HL POP          HL POP          NEXT JP
 5                 END-CODE
 6
 7 CODE SWAP       HL POP          (SP), HL EX     HPUSH JP
 8                 END-CODE
 9
10 CODE DUP        HL POP          HL PUSH         HPUSH JP
11                 END-CODE
12
13 CODE 2DUP       HL POP          DE POP          DE PUSH
14                 HL PUSH         DPUSH JP        END-CODE
15 ;S

  Screen 32
 0 ( ROMable Z-80 FORTH --- +! toggle @ )
 1
 2 CODE +!         HL POP          DE POP          A, (HL) LD
 3                 A, E ADD        (HL), A LD      HL INC
 4                 A, (HL) LD      A, D ADC        (HL), A LD
 5                 NEXT JP         END-CODE
 6
 7 CODE TOGGLE     DE POP          HL POP          A, (HL) LD
 8                 A, E XOR        (HL), A LD      NEXT JP
 9                 END-CODE
10
11 CODE @          HL POP          E, (HL) LD      HL INC
12                 D, (HL) LD      DE PUSH         NEXT JP
13                 END-CODE
14 ;S
15
```

```
   Screen 33
 0 ( ROMable Z-80 FORTH --- c@ 2@ ! )
 1
 2 CODE C@          HL POP          L, (HL) LD      H, # 0 LD
 3                  HPUSH JP        END-CODE
 4
 5 CODE 2@          HL POP          DE, # 3 LD      HL, DE ADD
 6                  D, (HL) LD      HL DEC          E, (HL) LD
 7                  DE PUSH         HL DEC          D, (HL) LD
 8                  HL DEC          E, (HL) LD      DE PUSH
 9                  NEXT JP         END-CODE
10
11 CODE !          HL POP          DE POP          (HL), E LD
12                  HL INC          (HL), D LD      NEXT JP
13                  END-CODE
14 ;S
15

   Screen 34
 0 ( ROMable Z-80 FORTH --- c! 2! 1+ 2+ )
 1
 2 CODE C!          HL POP          DE POP          (HL), E LD
 3                  NEXT JP         END-CODE
 4
 5 CODE 2!          HL POP          DE POP          (HL), E LD
 6                  HL INC          (HL), D LD      HL INC
 7                  DE POP          (HL), E LD      HL INC
 8                  (HL), D LD      NEXT JP
 9
10 CODE 1+          HL POP          HL INC          HPUSH JP
11                  END-CODE
12
13 CODE 2+          HL POP          HL INC          HL INC
14                  HPUSH JP        END-CODE
15 ;S

   Screen 35
 0 ( ROMable Z-80 FORTH --- 1- 2- - = )
 1
 2 CODE 1-          HL POP          HL DEC          HPUSH JP
 3                  END-CODE
 4
 5 CODE 2-          HL POP          HL DEC          HL DEC
 6                  HPUSH JP        END-CODE
 7
 8 CODE -           DE POP          HL POP          A, A OR
 9                  HL, DE SBC      HPUSH JP        END-CODE
10
11 CODE =          HL POP          DE POP          A, A XOR
12                  HL, DE SBC      H, A LD         L, A LD
13                  NZ, HPUSH JP    HL INC          HPUSH JP
14                  END-CODE
15 ;S
```

```
  Screen 36
 0 ( ROMable Z-80 FORTH --- < > fill )
 1
 2 CODE <          DE POP          HL POP          A, A OR
 3                 HL, DE SBC      HL, # 1 LD      M, HPUSH JP
 4                 HL DEC          HPUSH JP        END-CODE
 5
 6 CODE >          HL POP          DE POP          A, A OR
 7                 HL, DE SBC      HL, # 1 LD      M, HPUSH JP
 8                 HL DEC          HPUSH JP        END-CODE
 9
10 CODE FILL       EXX             DE POP          BC POP
11                 HL POP          HERE LABEL FILL1
12                 A, B LD         A, C OR         Z, FILL2 JR
13                 (HL), E LD      HL INC          BC DEC
14                 FILL1 JP        HERE LABEL FILL2
15                 EXX             NEXT JP         END-CODE    ;S

  Screen 37
 0 ( ROMable Z-80 FORTH --- p@ p! )
 1
 2 CODE P@         EXX             BC POP          L, (C) IN
 3                 H, # 0 LD       HL PUSH         EXX
 4                 NEXT JP         END-CODE
 5
 6 CODE P!         EXX             BC POP          HL POP
 7                 (C), L OUT      EXX             NEXT JP
 8                 END-CODE
 9
10 ;S
11
12
13
14
15

  Screen 38
 0 ( ROMable Z-80 FORTH --- s= )
 1
 2 CODE S=         EXX             BC POP          HL POP
 3                 DE POP
 4                 HERE LABEL STREQ1
 5                 A, B LD         A, C OR         Z, STREQ2 JR
 6                 A, (DE) LD      CPI             NZ, STREQ3 JR
 7                 DE INC          STREQ1 JP
 8                 HERE LABEL STREQ2
 9                 EXX             HL, # 1 LD      HPUSH JP
10                 HERE LABEL STREQ3
11                 EXX             HL, # 0 LD      HPUSH JP
12 ;S
13
14
15
```

```
  Screen 39
 0 ( ROMable Z-80 FORTH --- rot s->d mon )
 1
 2 CODE ROT        DE POP          HL POP          (SP), HL EX
 3                 DPUSH JP        END-CODE
 4
 5 CODE S->D       DE POP          HL, # 0 LD      A, D LD
 6                 A, # 080 AND    Z, STOD1 JP     HL DEC
 7                 HERE LABEL STOD1
 8                 DPUSH JP        END-CODE
 9
10 CODE MON        0 JP END-CODE
11 ;S
12
13
14
15

  Screen 40
 0 ( ROMable Z-80 FORTH --- constant user : does> )
 1
 2 : CONSTANT    CREATE SMUDGE ,
 3               ;CODE DE INC  DE, HL EX  E, (HL) LD  HL INC
 4               D, (HL) LD  DE PUSH  NEXT JP  END-CODE
 5 : USER        CONSTANT
 6               ;CODE  DE INC  DE, HL EX  E, (HL) LD  D, # 0 LD
 7               HL, UP LD  HL, DE ADD  HPUSH JP   END-CODE
 8 : :           ?EXEC !CSP CURRENT @ CONTEXT ! CREATE [COMPILE] ]
 9               ;CODE   IY DEC  (IY), B LD  IY DEC  (IY), C LD
10               DE INC  C, E LD  B, D LD  NEXT JP   END-CODE
11 : DOES>       R> LATEST PFA !   ;CODE   IY DEC  (IY), B LD
12               IY DEC  (IY), C LD  DE INC  DE, HL EX
13               C, (HL) LD  HL INC  B, (HL) LD  HL INC
14               HPUSH JP   END-CODE
15 ;S

  Screen 41
 0 ( ROMable Z-80 FORTH --- variable & vocabulary )
 1
 2 : VARIABLE              CREATE SMUDGE HERE 2+ , , ;CODE
 3                         DE INC  DE, HL EX  E, (HL) LD  HL INC
 4                         D, (HL) LD  DE PUSH  NEXT JP  END-CODE
 5
 6 : VOCABULARY            <BUILDS HERE 4 + ,
 7                         HERE VOC-LINK @ , VOC-LINK !
 8                         A081 , CURRENT @ CFA ,
 9                         DOES> @ 2+ CONTEXT ! ;
10
11 VOCABULARY FORTH IMMEDIATE
12 ;S
13
14
15
```

```
  Screen 42
 0 ( ROMable Z-80 FORTH --- user-definitions )
 1
 2 06 USER S0       08 USER R0              0A USER TIB
 3 0C USER WIDTH    0E USER WARNING         10 USER FENCE
 4 12 USER DP       14 USER VOC-LINK        16 USER BLK
 5 18 USER IN       1A USER OUT             1C USER SCR
 6 20 USER CONTEXT  22 USER CURRENT
 7 24 USER STATE    26 USER BASE            28 USER DPL
 8 3A USER FLD      2C USER CSP             2E USER R#
 9 30 USER HLD
10 ;S
11
12
13
14
15

  Screen 43
 0 ( ROMable Z-80 FORTH --- +origin cfa latest traverse pfa )
 1 : +ORIGIN        ORIGIN + ;
 2 : CFA            2 - ;
 3 : LATEST         CURRENT @ @ ;
 4 : TRAVERSE       SWAP BEGIN OVER + 07F OVER C@
 5                  < UNTIL SWAP DROP ;
 6 : PFA            1 TRAVERSE 5 + ;
 7 : (;CODE)        R> LATEST PFA CFA ! ;
 8 : HERE           DP @ ;
 9 : ALLOT          DP +! ;
10 : ,              HERE ! 2 ALLOT ;
11 : !CSP           SP@ CSP ! ;
12 : HOLD           -1 HLD +! HLD @ C! ;
13 : SMUDGE         LATEST 20 TOGGLE ;
14 ;S
15

  Screen 44
 0 ( ROMable Z-80 FORTH --- ?comp compile literal count type )
 1 : ?COMP          STATE @ 0= 11 ?ERROR ;
 2 : COMPILE        ?COMP R> DUP 2+ >R @ , ;
 3 : LITERAL        STATE @ IF COMPILE LIT , ENDIF ; IMMEDIATE
 4 : DLITERAL       STATE @ IF SWAP [COMPILE] LITERAL
 5                  [COMPILE] LITERAL ENDIF ; IMMEDIATE
 6 : COUNT          DUP 1+ SWAP C@ ;
 7 : -DUP           DUP IF DUP ENDIF ;
 8 : TYPE           -DUP IF OVER + SWAP DO I C@ EMIT LOOP
 9                  ELSE DROP ENDIF ;
10 : (.")           R COUNT DUP 1+ R> + >R TYPE ;
11 : PAD            HERE 44 + ;
12 : #>             DROP DROP HLD @ PAD OVER - ;
13 : SIGN           ROT 0< IF 2D HOLD ENDIF ;
14 ;S
15
```

```
  Screen 45
 0 ( ROMable Z-80 FORTH --- m/mod # #s d+- dabs +- abs m/ /mod )
 1 : M/MOD          >R 0 R U/ R> SWAP >R U/ R> ;
 2 : #              BASE @ M/MOD ROT 9 OVER < IF 7 + ENDIF
 3                  30 + HOLD ;
 4 : #S             BEGIN # 2DUP OR 0= UNTIL ;
 5 : <#             PAD HLD ! ;
 6 : D+-            0< IF DMINUS ENDIF ;
 7 : DABS           DUP D+- ;
 8 : +-             0< IF MINUS ENDIF ;
 9 : ABS            DUP +- ;
10 : M/             OVER >R >R DABS R ABS U/ R> R XOR +-
11                  SWAP R> +- SWAP ;
12 : /MOD           >R S->D R> M/ ;
13 : /              /MOD SWAP DROP ;
14 : MAX            2DUP < IF SWAP ENDIF DROP ;
15 ;S

  Screen 46
 0 ( ROMable Z-80 FORTH --- spaces d.r d. .cpu m* * */mod u. )
 1 : SPACE          BL EMIT ;
 2 : SPACES         0 MAX -DUP IF 0 DO SPACE LOOP ENDIF ;
 3 : D.R            >R SWAP OVER DABS <# #S SIGN #> R>
 4                  OVER - SPACES TYPE ;
 5 : D.             0 D.R SPACE ;
 6 : .CPU           BASE @ 24 BASE ! 22 +ORIGIN 2@ D. BASE ! ;
 7 : .              S->D D. ;
 8 : M*             2DUP XOR >R ABS SWAP ABS U* R> D+- ;
 9 : *              M* DROP ;
10 : */MOD          >R M* R> M/ ;
11 : -TRAILING      DUP 0 DO OVER OVER + 1 - C@ BL - IF LEAVE
12                  ELSE 1 - ENDIF LOOP ;
13 : U.             0 D. ;
14 ;S
15

  Screen 47
 0 ( ROMable Z-80 FORTH --- terminal I/O )
 1 ( port addresses and bit masks are equates in screen 9 )
 2 HEX
 3
 4 : EMIT           BEGIN STATUS-PORT P@ TBE AND UNTIL
 5                  DATA-PORT P! 1 OUT +! ;
 6
 7 : CR             0D EMIT 0A EMIT ;
 8
 9 : ?TERMINAL      STATUS-PORT P@ RDA AND
10                  IF 1 ELSE 0 ENDIF ;
11
12 : KEY            BEGIN ?TERMINAL UNTIL DATA-PORT P@ 07F AND ;
13 ;S
14
15
```

```
  Screen 48
 0 ( ROMable Z-80 FORTH --- message (abort error number ?exec )
 1 : MESSAGE       -DUP IF ."  Message # " . ENDIF ;
 2 : (ABORT)       ABORT ;
 3 : ERROR         WARNING @ 0< IF (ABORT) ENDIF HERE COUNT TYPE
 4                 ."  ? " MESSAGE SP! BLK @ -DUP
 5                 IF IN @ SWAP ENDIF QUIT ;
 6 : ?ERROR        SWAP IF ERROR ELSE DROP ENDIF ;
 7 : (NUMBER)      BEGIN 1+ DUP >R C@ BASE @ DIGIT WHILE SWAP
 8                 BASE @ U* DROP ROT BASE @ U* D+ DPL @ 1+
 9                 IF 1 DPL +! ENDIF R> REPEAT R> ;
10 : NUMBER        0 0 ROT DUP 1+ C@ 2D = DUP >R + -1 BEGIN DPL !
11                 (NUMBER) DUP C@ BL -
12                 WHILE DUP C@ 2E - 0 ?ERROR 0 REPEAT
13                 DROP R> IF DMINUS ENDIF ;
14 : ?EXEC        STATE @ 12 ?ERROR ;
15 ;S

  Screen 49
 0 ( ROMable Z-80 FORTH ---  u< ?stack blanks word -find nfa etc.)
 1 : U<            2DUP XOR 0< IF DROP 0< 0= ELSE - 0< ENDIF ;
 2 : ?STACK        SP@ S0 @ SWAP U< 1 ?ERROR SP@ HERE
 3                 80 + U< 7 ?ERROR ;
 4 : BLANKS        BL FILL ;
 5 : WORD          TIB @ IN @ + SWAP
 6                 ENCLOSE HERE 22 BLANKS IN +! OVER - >R
 7                 R HERE C! + HERE 1+ R> CMOVE ;
 8 : -FIND         BL WORD HERE CONTEXT @ @ (FIND) DUP 0=
 9                 IF DROP HERE LATEST (FIND) ENDIF ;
10 : NFA           5 - -1 TRAVERSE ;
11 : LFA           4 - ;
12 : ID.           PAD 20 5F FILL DUP PFA LFA OVER - PAD SWAP
13                 CMOVE PAD COUNT 1F AND TYPE SPACE ;
14 ;S
15

  Screen 50
 0 ( ROMable Z-80 FORTH --- expect null min create interpret )
 1 : EXPECT OVER + OVER DO KEY DUP 0E +ORIGIN @ = IF DROP DUP I =
 2   DUP R> 2 - + >R IF BELL ELSE BSOUT EMIT BL EMIT BSOUT
 3   ENDIF ELSE DUP 0D = IF LEAVE DROP
 4   BL 0 ELSE DUP ENDIF R C! 0 R 1+ ! ENDIF EMIT LOOP DROP ;
 5 : X             R> DROP ; IMMEDIATE IS-X
 6 : MIN           2DUP > IF SWAP ENDIF DROP ;
 7 : CREATE -FIND IF DROP NFA ID. 4 MESSAGE SPACE ENDIF HERE
 8   DUP C@ WIDTH @ MIN 1+ ALLOT DUP A0 TOGGLE HERE 1 - 80
 9   TOGGLE LATEST , CURRENT @ ! HERE 2+ , ;
10 : INTERPRET BEGIN -FIND IF STATE @ < IF CFA , ELSE CFA EXECUTE
11   ENDIF ?STACK ELSE HERE NUMBER DPL @ 1+ IF [COMPILE] DLITERAL
12   ELSE DROP [COMPILE] LITERAL ENDIF ?STACK ENDIF AGAIN ;
13 ;S
14
15
```

```
  Screen 51
 0 ( ROMable Z-80 FORTH --- query quit definitions decimal etc. )
 1 : QUERY          TIB @ 50 EXPECT 0 IN ! ;
 2 : [              0 STATE ! ; IMMEDIATE
 3 : ]              C0 STATE ! ; IMMEDIATE
 4 : QUIT           0 BLK ! [COMPILE] [ BEGIN CR RP! QUERY
 5                  INTERPRET STATE @ 0= IF ." OK" ENDIF AGAIN ;
 6 : DEFINITIONS    CONTEXT @ CURRENT ! ;
 7 : DECIMAL        0A BASE ! ;
 8 : <BUILDS        0 CONSTANT ;
 9 : ABORT          SP! DECIMAL ?STACK CR .CPU
10                  ." fig-FORTH [stand-alone version]" CR
11                  [COMPILE] FORTH DEFINITIONS QUIT ;
12 ;S
13
14
15

  Screen 52
 0 ( ROMable Z-80 FORTH --- erase ?pairs back begin endif etc. )
 1 ( *** WARNING:  BACK and ENDIF changed from fig-FORTH model )
 2 HEX
 3 : ERASE          0 FILL ;
 4 : ?PAIRS         - 13 ?ERROR ;
 5 : BACK           , ;
 6 : BEGIN          ?COMP HERE 1 ; IMMEDIATE
 7 : ENDIF          ?COMP 2 ?PAIRS HERE SWAP ! ; IMMEDIATE
 8 : THEN           [COMPILE] ENDIF ; IMMEDIATE
 9 : DO             COMPILE (DO) HERE 3 ; IMMEDIATE
10 : LOOP           3 ?PAIRS COMPILE (LOOP) BACK ; IMMEDIATE
11 : COLD           INIT-R0 RAM-START !
12                  INIT-RAM DUP >R 4 + RAM-START 2+ R> @ 2 - CMOVE
13                  12 +ORIGIN UP @ 6 + 10 CMOVE
14                  0C +ORIGIN @ ' FORTH 2+ @ 2+ ! UABORT ;
15 ;S

  Screen 53
 0 ( ROMable Z-80 FORTH --- +loop until end again repeat if etc. )
 1 : +LOOP          3 ?PAIRS COMPILE (+LOOP) BACK ; IMMEDIATE
 2 : UNTIL          1 ?PAIRS COMPILE 0BRANCH BACK ; IMMEDIATE
 3 : END            [COMPILE] UNTIL ; IMMEDIATE
 4 : AGAIN          1 ?PAIRS COMPILE BRANCH BACK ; IMMEDIATE
 5 : REPEAT         >R >R [COMPILE] AGAIN R> R> 2 -
 6                  [COMPILE] ENDIF ; IMMEDIATE
 7 : IF             COMPILE 0BRANCH HERE 0 , 2 ; IMMEDIATE
 8 : ELSE           2 ?PAIRS COMPILE BRANCH HERE 0 , SWAP 2
 9                  [COMPILE] ENDIF 2 ; IMMEDIATE
10 : WHILE          [COMPILE] IF 2+ ; IMMEDIATE
11 : ?CSP           SP@ CSP @ - 14 ?ERROR ;
12 : ;              ?CSP COMPILE ;S SMUDGE [COMPILE] [ ; IMMEDIATE
13 ;S
14
15
```

```
  Screen 54
 0 ( ROMable Z-80 FORTH --- .r hex immediate [compile] ´ ." warm )
 1 : .R            >R S->D R> D.R ;
 2 : WARM          ABORT ;
 3 : ."            22 STATE @ IF COMPILE (.") WORD HERE C@ 1+ ALL
 4                 ELSE WORD HERE COUNT TYPE ENDIF ; IMMEDIATE
 5 : HEX           10 BASE ! ;
 6 : IMMEDIATE     LATEST 40 TOGGLE ;
 7 : (             29 WORD ; IMMEDIATE
 8 : [COMPILE]     -FIND 0= 0 ?ERROR DROP CFA , ; IMMEDIATE
 9 : ´             -FIND 0= 0 ?ERROR DROP
10                 [COMPILE]   LITERAL ; IMMEDIATE
11
12 20 CONSTANT BL
13 40 CONSTANT C/L
14 ;S
15

  Screen 55
 0 ( ROMable Z-80 FORTH --- mod ? forget vlist noop task )
 1 : MOD           /MOD DROP ;
 2 : C,            HERE C! 1 ALLOT ;
 3 : ?             @ . ;
 4 : FORGET        CURRENT @ CONTEXT @ - 18 ?ERROR [COMPILE] ´ DU
 5                 FENCE @ < 15 ?ERROR DUP NFA DP !
 6                 LFA @ CURRENT @ ! ;
 7 : VLIST         C/L OUT ! CONTEXT @ @ BEGIN
 8                 C/L OUT @ - OVER C@ 01F AND 4 + <
 9                 IF CR 0 OUT ! ENDIF
10                 DUP ID. SPACE SPACE PFA LFA @ DUP 0=
11                 ?TERMINAL OR UNTIL DROP ;
12 : NOOP          ;
13 : NOT           0= ;
14 ;S
15

  Screen 56
 0 ( Case statement, by Charles E. Eaker )
 1
 2 : EXIT R> DROP ;  ( EXIT FROM CURRENT WORD )
 3
 4 ( from FORTH DIMENSIONS II/3 page 37  )
 5 : CASE ?COMP CSP @ !CSP 4 ; IMMEDIATE
 6 : OF 4 ?PAIRS COMPILE OVER COMPILE = COMPILE 0BRANCH
 7   HERE 0 , COMPILE DROP 5 ; IMMEDIATE
 8 : ENDOF 5 ?PAIRS COMPILE BRANCH HERE 0 ,
 9   SWAP 2 [COMPILE] ENDIF 4 ; IMMEDIATE
10 : ENDCASE 4 ?PAIRS COMPILE DROP BEGIN SP@
11   CSP @ = 0= WHILE 2 [COMPILE] ENDIF
12   REPEAT CSP ! ; IMMEDIATE
13
14
15
```

```
   Screen 57
 0 ( CONTROLLER -- EQUATES )
 1 0A       EQU     COUNT$VAL     ( SECONDS BETWEEN SAMPLES )
 2
 3 80       EQU     S$B
 4 S$B      EQU     TAPE-DATA        70       EQU     A$B
 5 S$B 1+   EQU     TAPE-STAT        A$B      EQU     AD-STAT
 6 ( S$B 2+  EQU      DATA-PORT )    A$B 1+   EQU     AD-INIT
 7 ( S$B 3 + EQU      STATUS-PORT)   A$B 2+   EQU     AD-LO
 8 S$B 4 + EQU      PIOA-DATA        A$B 3 +  EQU     AD-HI
 9 S$B 5 + EQU      PIOA-CMND
10 S$B 6 + EQU      PIOB-DATA        8000     EQU     BUF0#
11 S$B 7 + EQU      PIOB-CMND        A000     EQU     BUF1#
12 S$B 8 + EQU      CTC0             2000     EQU     BUF-LEN
13 S$B 9 + EQU      CTC1
14 S$B A + EQU      CTC2             6        EQU     CTCOFS
15 S$B B + EQU      CTC3             CTCOFS   EQU     ISRVCTROFS  ;S

   Screen 58
 0 ( CONTROLLER -- EQUATES )
 1
 2 40       EQU     GO-BACK          01       EQU     WR-EN
 3 02       EQU     WRT              02       EQU     REW
 4 03       EQU     WFMK             04       EQU     EOT
 5 07       EQU     BCKF             10       EQU     CR-IN
 6 08       EQU     REPORT-IN
 7 0D       EQU     SKIP-F           70       EQU     NO-LIGHTS
 8 44       EQU     SKIP-RB          E0       EQU     RED-LIGHT
 9                                   D0       EQU     ORANGE-LIGHT
10 3000     EQU     CMD-OK           B0       EQU     GREEN-LIGHT
11 0C       EQU     DRIVE1
12
13 02       EQU     NSOFS            00       EQU     INI-PA
14 80       EQU     ARDY             C0       EQU     INI-MA
15 ;S                          ( NOTE:  INI-MA IS FOR 8K BUFFER )

   Screen 59
 0 ( CONTROLLER -- EQUATES )
 1
 2 40   EQU CH-LIMIT
 3 800  EQU ZOFST
 4 A9   EQU SIGCOEFF
 5 19   EQU SAMP$VAL
 6 6000 EQU RBUF
 7 ;S
 8
 9 Currently set for 2.5 interrupts per second for a window of 25
10 samples per datum.  Change SIGCOEFF, SAMP$VAL, and the stuff
11 after VALS in screen 62 for different sample rates.
12
13
14
15
```

```
  Screen 60
 0 ( CONTROLLER -- INTERRUPT SERVICE ROUTINE ENTRY )
 1 ASSEMBLER
 2
 3 HERE LABEL EISR     DI  ( INTERRUPT SERVICE ROUTINE ENTRY )
 4   AF PUSH  BC PUSH  DE PUSH  HL PUSH  EXX
 5   AF PUSH  BC PUSH  DE PUSH  HL PUSH
 6   IX PUSH  IY PUSH  BC, # ISR1 LD     NEXT JP
 7
 8 HERE LABEL ISR1          ] ISR ISREXIT [
 9
10 FORTH DEFINITIONS
11
12 CODE ISREXIT      IY POP  IX POP  ( RETURN FROM INTERRUPT )
13   HL POP  DE POP  BC POP  AF POP  EXX
14   HL POP  DE POP  BC POP  AF POP  EI   RETI  END-CODE  ;S
15

  Screen 61
 0 ( CONTROLLER -- ISR PROGRAMMING & INITIALIZATION )
 1
 2 ( PROGRAM INTERRUPT SERVICE ROUTINE )
 3 CODE PISR
 4   IM2  A, A SUB  0ED C,(T)  047 C,(T)    ( I, A LD )
 5   NEXT JP        END-CODE
 6
 7 ( ENABLE INTERRUPTS )
 8 CODE   EI        EI         NEXT JP      END-CODE
 9 ( DISABLE INTERRUPTS )
10 CODE   DI        DI         NEXT JP      END-CODE
11
12 ( INITIALIZE INTERRUPT SERVICE ROUTINE )
13 : SISR  PISR  EISR ISRVCTROFS ! ;
14 ;S
15

  Screen 62
 0 ( CONTROLLER -- VARIABLES )
 1
 2 0  VARIABLE  PTR                0  VARIABLE  0-PTR-INIT
 3 0  VARIABLE  W-PTR              0  VARIABLE  1-PTR-INIT
 4 0  VARIABLE  CHKSUM#            0  VARIABLE  0-CHK-INIT
 5 0  VARIABLE  CNT                0  VARIABLE  1-CHK-INIT
 6 0  VARIABLE  BUF*               0  VARIABLE  MA
 7 0  VARIABLE  NTIMES             0  VARIABLE  CT
 8 201 VARIABLE VALS
 9     3 C,(R) 4 C,(R) 5 C,(R) 6 C,(R) 7 C,(R) 8 C,(R) 9 C,(R)
10     A C,(R) B C,(R) C C,(R) D C,(R)
11                     C C,(R) B C,(R) A C,(R) 9 C,(R) 8 C,(R)
12     7 C,(R) 6 C,(R) 5 C,(R) 4 C,(R) 3 C,(R) 2 C,(R) 1 C,(R)
13
14 0  VARIABLE  TMP  CH-LIMIT ALLOT-RAM
15
```

```
  Screen 63
 0 ( CONTROLLER -- DEVICE PRIMITIVES )
 1
 2 : CHA!          TAPE-STAT P! ;    ( PROGRAM TAPE UART )
 3 : CHB!          STATUS-PORT P! ;  ( PROGRAM TERMINAL UART )
 4 : TP@           TAPE-DATA P@ ;    ( READ TAPE PORT )
 5 : TP!           TAPE-DATA P! ;    ( WRITE TAPE PORT)
 6 : PIOA@         PIOA-DATA P@ ;    ( READ NO. CHANNELS )
 7 : PIOA!         PIOA-CMND P! ;    ( NOT USED )
 8 : PIOB@         PIOB-DATA P@ 0F AND ; ( READ STATION ID )
 9 : PIOB!         PIOB-CMND P! ;    ( NOT USED )
10 : ON            PIOB-DATA P! ;    ( TURN A LIGHT ON )
11 : WRA!          CHA!  CHA! ;
12 : WRB!          CHB!  CHB! ;
13 : A-CLOCK       47 CTC0 P!  0C CTC0 P! ;  ( COUNTER, 9600X16 )
14 : B-CLOCK       47 CTC1 P!  C0 CTC1 P! ;  ( COUNTER, 300X32 )
15 ;S                                        ( SET TO 6 FOR 9600B)

  Screen 64
 0 ( CONTROLLER -- DEVICE PROGRAMMING )
 1
 2 : PIO-INIT  CF PIOA!  FF PIOA!  07 PIOA!    ( MODE 3 0A/4B IN )
 3             CF PIOB!  0F PIOB!  07 PIOB! ; ( NO INTS )
 4 : A-SIO     18 CHA!     4C 04 WRA!    ( RESET 16X CLK 2 STP NP )
 5             00 01 WRA!  C1 03 WRA!    ( NO INT R:8 B/C RX EN )
 6             68 05 WRA! ;              ( T:8 B/C TX ENABLE )
 7 : B-SIO     18 CHB!     84 04 WRB!    ( RST 32X CLK 1 STOP NPAR )
 8             00 01 WRB!  00 02 WRB!    ( NO INT 0 INT VCTR )
 9             C1 03 WRB!  68 05 WRB! ; ( RX&TX AS ABOVE )
10 : CTC-INIT  SISR        00 CTC0 P!    ( 0 INT VECTOR )
11             27 CTC2 P!  19 CTC2 P!    ( NO INT P=256 TC=25 )
12             C7 CTC3 P!  FA CTC3 P! ; ( EN INT CNTR TC=250 )
13 : CHA-INIT  A-CLOCK A-SIO ;
14 : CHB-INIT  B-CLOCK B-SIO ;
15 : DEV-INIT  CHA-INIT CHB-INIT CTC-INIT PIO-INIT ;

  Screen 65
 0 ( CONTROLLER -- CARTRIDGE TAPE PRIMITIVES )
 1
 2 : TPX?          BEGIN TAPE-STAT P@ TBE AND UNTIL ;
 3 : TPR?          BEGIN TAPE-STAT P@ RDA AND UNTIL ;
 4 : CMD           TPX? TP! ;
 5 : DSW           TPR? TP@ TPR? TP@ 100 * + 10 0 DO NOOP LOOP ;
 6 : 1C&R          CMD DSW ;
 7 : 3C&R          CMD CMD 1C&R ;
 8 : INIT-TAPE     REPORT-IN INI-PA INI-MA MA ! INI-MA 3C&R DROP ;
 9 : GET-STATUS    REPORT-IN 1C&R ;
10 : CARTRIDGE-IN? GET-STATUS CR-IN AND ;
11 : WRITE-ENABLED? GET-STATUS WR-EN AND ;
12 : REWIND        GO-BACK 1C&R DROP ;
13 : DESELECT      REPORT-IN INI-PA INI-MA DRIVE1 + 3C&R DROP ;
14 : RESELECT      REPORT-IN INI-PA MA @ 3C&R DROP ;
15 : WAIT5         500 0 DO NOOP NOOP NOOP NOOP NOOP LOOP ;  ;S
```

```
   Screen 66
 0 ( SPARE )
 1 ;S
 2
 3
 4
 5
 6
 7
 8
 9
10
11
12
13
14
15

   Screen 67
 0 ( CONTROLLER -- BUFFER PRIMITIVES )
 1
 2 : #SAMP!          [ BUF0# NSOFS + ] LITERAL C! ;
 3 : #SAMP@          [ BUF0# NSOFS + ] LITERAL C@ ;
 4 : SITE!           BUF0# ! ;
 5 : S-RATE!         [ BUF0# NSOFS 1+ + ] LITERAL PTR !  #SAMP@ 0
 6                   DO  COUNT$VAL PTR @ C!  1 PTR +! LOOP ;
 7 : CHKSUM@         CHKSUM# @ @ ;
 8 : CHKSUM!         CHKSUM# @ ! ;
 9 : SWAP-BUF        BUF* @ DUP NOT BUF* !
10                   IF 0-PTR-INIT @ PTR !  0-CHK-INIT @ CHKSUM# !
11                      BUF1# W-PTR !
12                   ELSE 1-PTR-INIT @ PTR ! 1-CHK-INIT @ CHKSUM# !
13                      BUF0# W-PTR !
14                   ENDIF ;
15 ;S

   Screen 68
 0 ( CONTROLLER -- TAPE ROUTINES )
 1
 2 : WR-FMK          WFMK 1C&R DROP ;
 3 : WR-EOF          WR-FMK WR-FMK ;
 4 : BK-FMK          BCKF 1C&R DROP ;
 5 : WR-CMD          MA @ CMD INI-PA CMD WRT CMD ;
 6 : EOT?            GET-STATUS EOT AND ;
 7 : SEND-D          W-PTR @ DUP BUF-LEN + SWAP
 8                   DO I C@ CMD LOOP ;
 9 : NEXT-TRACK      1 MA +! MA @ 3 AND
10                   IF RESELECT REWIND
11                   ELSE INI-MA MA !  OVER-RUN  ENDIF ;
12
13 : ZTMP           TMP #SAMP@ 4 * 0 FILL ;
14 ;S
15
```

```
  Screen 69
 0 ( CONTROLLER -- TAPE ROUTINES )
 1
 2 : NOT-OK?        CMD-OK AND ;
 3 : IS-OK?         NOT-OK? NOT ;
 4 : SKIP-FIL       SKIP-F 1C&R DROP ;
 5 : RW-SKIP-R      SKIP-RB INI-PA MA @ 3C&R ;
 6 : FIND-EOD       REWIND  INI-MA 4 + MA ! 4 0
 7                  DO -1 MA +! RW-SKIP-R IS-OK?
 8                     IF SKIP-FIL LEAVE ENDIF
 9                  LOOP BK-FMK ;
10 : BLINK-SELECT   DESELECT WAIT5 RESELECT WAIT5 ;
11 : ORANGE-BLINK   NO-LIGHTS ON ORANGE-LIGHT ON WAIT5
12                  NO-LIGHTS ON WAIT5 ;
13 : RED-BLINK      NO-LIGHTS ON RED-LIGHT ON WAIT5
14                  NO-LIGHTS ON WAIT5 ;
15 : GREEN-ON       NO-LIGHTS ON GREEN-LIGHT ON ;

  Screen 70
 0 ( CONTROLLER -- TAPE ROUTINES )
 1
 2 : TAPE-ON?       NO-LIGHTS ON   INIT-TAPE
 3                  BEGIN
 4                    BEGIN
 5                      CARTRIDGE-IN? NOT
 6                    WHILE
 7                      BLINK-SELECT
 8                    REPEAT
 9                    WRITE-ENABLED? NOT
10                  WHILE
11                    BEGIN
12                      ORANGE-BLINK  CARTRIDGE-IN? NOT
13                    UNTIL
14                  REPEAT  FIND-EOD  EOT? IF OVER-RUN ENDIF
15                  GREEN-ON ;

  Screen 71
 0 ( CONTROLLER -- VAR-INIT AND BUF-FULL? )
 1
 2 : VAR-INIT       PIOA@ DUP F AND 30 + PAD 2+ C! 10 / F AND
 3                  30 + PAD 1+ C! 2 PAD C! 20 PAD 3 + C!
 4                  PAD NUMBER DROP 1+ #SAMP!
 5                  PIOB@ DUP 0= IF 4F43 ELSE  ( CO )
 6                   1 -  DUP 0= IF 5043 ELSE  ( CP )
 7                   1 -      0= IF 424D ELSE  ( MB )
 8                                 0000 ENDIF ENDIF ENDIF ( UNDEF)
 9                  SITE!  S-RATE!  PTR @ 1+ FFFE AND
10                  DUP  0-CHK-INIT ! DUP CHKSUM# !  DUP 2000 +
11                  1-CHK-INIT ! 2+ DUP DUP 0-PTR-INIT ! PTR !
12                  2000 + 1-PTR-INIT !  0 BUF* !  0 CNT !
13                  BUF0# BUF1# PTR @ BUF0# - CMOVE ZTMP ;
14 : BUF-FULL?     BUF* @ IF BUF1# ELSE BUF0# ENDIF
15                  BUF-LEN + PTR @ - #SAMP@ 2 * SWAP > ;
```

```
  Screen 72
 0 ( SPARE )
 1 ;S
 2
 3
 4
 5
 6
 7
 8
 9
10
11
12
13
14
15

  Screen 73
 0 ( CONTROLLER -- A/D CONVERTER ROUTINES )
 1
 2   CODE A/D-CH   EXX     BC, # AD-STAT LD        HL POP
 3                 (C), L OUT      EXX     NEXT JP    END-CODE
 4   CODE A/D-GO   EXX     BC, # AD-INIT LD        (C), L OUT
 5                 EXX     NEXT JP          END-CODE
 6 ( CODE A/D-RDY  NEXT JP END-CODE                              )
 7   CODE A/D-DATA EXX     BC, # AD-LO LD  L, (C) IN
 8                 BC INC  A, (C) IN       A, # F AND
 9                 H, A LD HL PUSH   EXX   NEXT JP    END-CODE
10 ;S
11
12
13
14
15

  Screen 74
 0 ( CONTROLLER -- A/D CONVERTER ROUTINES )
 1
 2 : BUF!          PTR @ !  2 PTR +! ;
 3 : TMP!          4 * TMP + DUP >R 2@ D+ R> 2! ;
 4 : CHKSUM        CHKSUM@ + CHKSUM! ;
 5 : A/D-READ      ( A/D-RDY ) A/D-DATA VALS CNT @ + C@ U*
 6                 ROT TMP! ;
 7 : SAMPLE        0 A/D-CH  #SAMP@ 1
 8                 DO A/D-GO I A/D-CH I 1- A/D-READ LOOP
 9                 A/D-GO #SAMP@ 1- A/D-READ ;
10 : SAVE          #SAMP@ 0 DO  I 4 * TMP + DUP >R 2@ SIGCOEFF U/
11                 DUP CHKSUM BUF!  DROP
12                 0 0 R> 2! LOOP ;
13 ;S
14
15
```

```
  Screen 75
 0 ( SPARE )
 1 ;S
 2
 3
 4
 5
 6
 7
 8
 9
10
11
12
13
14
15

  Screen 76
 0 ( CONTROLLER -- WRITE BUFFER )
 1
 2 : WRITE-IT       BEGIN WR-CMD SEND-D DSW NOT-OK?
 3                  WHILE EOT? IF NEXT-TRACK ENDIF
 4                  REPEAT ;
 5 : WRITE-BUF      SWAP-BUF WRITE-IT EOT?
 6                  IF WR-EOF NEXT-TRACK
 7                  ELSE WR-FMK BK-FMK ENDIF ;
 8 : KILL-TIME      RP! SP!
 9                  BEGIN
10                     BUF-FULL? IF WRITE-BUF ENDIF
11                     ?TERMINAL IF KEY DROP DI QUIT ENDIF
12                     DI CARTRIDGE-IN? NOT
13                        IF TAPE-ON? ENDIF
14                     EI
15                  AGAIN ;

  Screen 77
 0 ( CONTROLLER -- INTERRUPT DRIVEN DRIVER )
 1
 2 : OVER-RUN       DI REWIND BEGIN CARTRIDGE-IN? WHILE RED-BLINK
 3                  REPEAT TAPE-ON? VAR-INIT EI  KILL-TIME ;
 4
 5 : ISR            SAMPLE  1 CNT +!
 6                  CNT @ SAMP$VAL =
 7                  IF 0 CNT ! SAVE ENDIF ;
 8
 9 : UABORT         WAIT5 WAIT5 WAIT5 WAIT5 ( GIVE TAPE DRIVE TIME)
10                  DEV-INIT ( ABORT   KILL-TIME ; )
11                  SP! DECIMAL [COMPILE] FORTH DEFINITIONS
12                  0 BLK ! [COMPILE] [ RP!
13                  TAPE-ON? VAR-INIT EI KILL-TIME ;
14 ;S
15
```

```
  Screen 78
 0 ( Dump, intrasegment, byte format )              DECIMAL
 1 ( display n memory locations in hex and ASCII, starting )
 2 ( at addr rounded to next lower 16 byte boundary )
 3 : DUMP   ( addr n DUMP ->   )
 4   BASE @ >R HEX CR CR 5 SPACES       ( save current BASE )
 5   16 0 DO I 3 .R LOOP 2 SPACES       ( print titles )
 6   16 0 DO I 0 <# # #> TYPE LOOP CR
 7   OVER + SWAP DUP 15 AND XOR DO      ( round starting address )
 8   CR I 0 4 D.R 1 SPACES              ( print address )
 9   I 16 + I 2DUP
10             DO I C@ SPACE 0 <# # # #> TYPE LOOP   ( hex )
11             2 SPACES
12             DO I C@ DUP 32 < IF DROP 46 ENDIF     ( ASCII )
13                  DUP 126 > IF DROP 46 ENDIF EMIT LOOP
14   16 +LOOP CR R> BASE !  ;            ( restore BASE )
15 ;S

  Screen 79
 0 ( Read a block for debugging )                          HEX
 1
 2 : SKIP DUP 0< IF ABS 86 ELSE 84 ENDIF SWAP 1- MA C@ 3C&R DROP ;
 3 : READ 1 CMD TPR? TP@ TPR? TP@ [ RBUF BUF-LEN + ] LITERAL RBUF
 4         DO TPR? TP@ I C! LOOP 100 * + U. ;
 5 IS-FENCE
 6 FINIS  ;S
 7
 8
 9
10
11
12
13
14
15
```

Literature Cited

Alloy Engineering Company, DEI/DRS-232 Controller Interface Guide, Alloy Engineering Company, 1980.

Brodie, L., Starting FORTH, Prentice-Hall, 1981.

Chen, L. and A. Hasegawa, A Theory of Long-Period Magnetic Pulsations 1. Steady State Excitation of Field Line Resonance, J. Geophys. Res., **79**, 1024, 1974.

Collier, J. L., Morphology of Microbaroms at Windless Bight, Antarctica, and Fairbanks, Alaska, M. S. Thesis, University of Alaska, 1983.

Derick, M., and L. Baker, FORTH Encyclopedia, Mountain View Press, 1982.

Dual Systems Control Corp., 12 Bit Analog Input Module for IEEE-696/S-100 bus Computer Systems Models AIM-12 and AIM-12B User's Manual, Dual Systems Control Corp., 1981.

Fowler, R. A., B. J. Kotick, and R. D. Elliott, Polarization Analysis of Natural and Artificially Induced Geomagnetic Micropulsations, J. Geophys. Res., **72**, 2871, 1967.

Hamming, R. W., Digital Filters, Prentice-Hall, 1977.

Hughes, W. J. and D. J. Southwood, An Illustration of Modification of Geomagnetic Pulsation Structure by the Ionoshpere, J. Geophys. Res., **81**, 3241, 1976.

Jacobs, J. A., Geomagnetic Micropulsations, Springer-Verlag, 1970.

Jacobs, J. A., Y. Kato, S. Matsushita, and V. A. Troitskaya, Classification of Geomagnetic Micropulsations, J. Geophys. Res., **69**, 180, 1964.

Kan, J. R., D. U. Longenecker, and J. V. Olson, A Transient Response Model of Pi2 Pulsations, J. Geophys. Res., **87**, 7483, 1982.

Kinemetrics, Operating and Service Manual Model 468-DC Satellite Synchronized Clock, Kinemetrics, 1981.

Laboratory Microsystems, 8086 FORTH User's Manual, Laboratory Microsystems, 1982.

Libes, S. and M. Garetz, Interfacing to S-100/IEEE 696 Microcomputers, McGraw-Hill, 1981.

Nautilus Systems, A FORTH Cross-Compiler, Nautilus Systems, 1981.

Nishida, A., Geomagnetic Diagnosis of the Magnetosphere, Springer-Verlag, 1978.

Olson, J. V., Fabrication and Implementation of ULF Spectrum Channel Cards, Progress Report No. 1, National Science Foundation Grant ATM81-11475, 1982.

Olson, J. V. and J. C. Samson, On the Detection of the Polarization States of Pc Micropulsations, Geophys. Res. Lett., **6**, 413, 1979.

Olson, J. V. and J. C. Samson, Generalized Power Spectra and the Stokes Vector Representations of Ultralow Frequency Micropulsation States, Can. J. Phys., **58**, 123, 1980.

Orr, D., Magnetic Pulsations Within the Magnetosphere: A Review, J. Atmos. Terr. Phys., **35**, 1, 1973.

Saito, T., Geomagnetic Pulsations, Space Sci. Rev., **10**, 319, 1969.

Samson, J. C., J. A. Jacobs, and G. Rostoker, Latitude Dependent Characteristics of Long Period Geomagnetic Micropulsations, J. Geophys. Res., **76**, 3675, 1971.

Sierra Data Sciences, SBC-100/SBC-100S Product Description Manual, Sierra Data Sciences, Inc., 1982.

Southwood, D. J., Some Features of Field Line Resonances in the Magnetosphere, Planet. Space Sci., **22**, 483, 1974.

Stewart, B., On the Great Magnetic Disturbance which Extended from August 28 to September 7, 1859, as Recorded by Photography at the Kew Observatory, Phil. Trans. Roy. Soc. London, 425, 1861.