NGR-14-005-181

AERONOMY REPORT NO. 84

A ROCKET-BORNE DATA-MANIPULATION EXPERIMENT USING A MICROPROCESSOR

by
L. L. Davis
L. G. Smith
H. D. Voss

April 1, 1979

Library of Congress ISSN 0568-0581

Aeronomy Laboratory
Department of Electrical Engineering
University of Illinois
Urbana, Illinois

Supported by
National Aeronautics and Space Administration

CITATION POLICY

UILU-ENG-79 2503

A E R O N O M Y   R E P O R T

N O. 84

A ROCKET-BORNE DATA-MANIPULATION EXPERIMENT USING A MICROPROCESSOR

by

L. L. Davis
L. G. Smith
H. D. Voss

April 1, 1979

Supported by
National Aeronautics
and Space Administration
Grant NGR-14-005-181

Aeronomy Laboratory
Department of Electrical Engineering
University of Illinois
Urbana, Illinois

ABSTRACT

This report describes the development of a data-manipulation experiment using a Z-80 microprocessor. The instrumentation is included in the payloads of two Nike Apache sounding rockets used in an investigation of energetic particle fluxes. The data from an array of solid-state detectors and an electrostatic analyzer is processed to give the energy spectrum as a function of pitch angle.

The experiment performed well in its first flight test: Nike Apache 14.543 was launched from Wallops Island at 2315 EST on 19 June 1978. The system has been designed to be easily adaptable to other data-manipulation requirements and some suggestions for further development are included.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

This report describes in detail the development of a rocket-borne data-manipulation experiment designed to process data from an array of solid-state detectors and an electrostatic analyzer into a form giving the energy spectrum as a function of pitch angle. The information provided by the system will aid in further study of energetic particles in the *E* region of the ionosphere. Previous phases of the investigation have been described in reports by *Voss and Smith* [1974, 1977] and in papers by *Smith et al.* [1974], *Geller et al.* [1975] and *Smith and Voss* [1976].

The energetic particle experiments in the rocket payloads are described in Chapter 2. These provide the input to the data-manipulation experiment.

A general description of the data-manipulation experiment is given in Chapter 3. The Z-80 microcomputer, the nucleus of the system, is described together with the devices used to interface the detectors to it. The hardware configuration is shown in block diagram form. The input and output algorithms used in the software are shown and a description is given of the format in which data are output from the system.

The hardware and software are described in detail in Chapters 4 and 5, respectively. The four printed circuit boards that contain the data-manipulation experiment are described. Timing diagrams of the interface circuits are presented and an explanation is given of the timing logic. The software is shown in an instruction-level flow chart, and the effects of various parameters in the software are described. A feature of the design is the ability of the experiment to accept a range of rocket spin rates.

Chapter 6 contains a description of the equipment for developing the software; the MOSTEK software development board (SDB-80) was used. The chapter also includes details on the design and operation of the EPROM programmer.

In Chapter 7 the performance of the data-manipulation experiment during the flight of Nike Apache 14.543 is discussed. Some actual data from the flights are presented to show that the system fulfilled its design goal.

In Chapter 8 suggestions for future work are given. Some improvements to the present system in terms of both hardware and software are included. Possible uses of microprocessors in other experiments and in the telemetry system are suggested in a block diagram form.

Table 1.1 contains a glossary of abbreviations used in the text and figures of this report.

Table 1.1

Abbreviations

A/D	analog-to-digital
CPU	central processing unit
CTC	counter-timer circuit*
DIP	dual in-line package
EPROM	erasable programmable read-only memory
EPS	energetic particle spectrometer
ESA	electrostatic analyzer
FIFO	first-in first-out memory
IEI	interrupt enable input
I/O	input-output
LSI	large-scale integration
MOS	metal-oxide semiconductor
MSD	magnetometer-signal digitizer
PHA	pulse-height analyzer
PIO	parallel input-output circuit*
RAM	random access memory

*Circuits used in the Z-80 microcomputer

2. ENERGETIC PARTICLE EXPERIMENTS

2.1 *Introduction*

The data-manipulation experiment in the rocket payload receives data from an energetic particle spectrometer (EPS) and an electrostatic analyzer (ESA). The function of these two types of particle detectors is to provide a broad-band spectral analysis of energetic particles (including electrons and protons) which are believed to be important as an ionization source in the nighttime *E* region. The energetic particle spectrometer is used for the higher energies and the electrostatic analyzer for the lower energies.

The energetic particle spectrometer and electrostatic analyzer experiments prepared for Nike Apache payloads 14.542 and 14.543 are described in this chapter. The objective here is to explain the origin and nature of the signals which are the input to the data-manipulation experiment.

It should be noted that the payloads each include, in addition to the data-manipulation experiment described in this report, two other systems for processing data from the particle detectors. One is a relatively simple system which generates a staircase waveform; this is particularly valuable for rapid evaluation of the performance of the detectors. The system has been used in previous payloads and is described in *Voss and Smith* [1974].

The other system uses a multiplexer followed by a sample-and-hold circuit. This was developed for the payloads of Nike Apaches 14.542 and 14.543. It is described in detail in *Leung et al.* [1979].

The three systems involve some redundancy both for protection against instrumentation malfunction and to allow intercomparison of their performance. Only those circuits which are relevant to the microcomputer data-manipulation experiment will be described in the following sections.

2.2 *Energetic Particle Spectrometer*

The energetic particle spectrometer (EPS) comprises a group of six solid-state detectors in each of the payloads with each detector selected for a particular objective. The detectors are oriented in several directions relative to the payload spin axis and as a group will resolve the energy spectrum between 12 and 400 keV. The data obtained from the detectors can be used to identify the particles as electrons, protons or heavier nuclei.

The solid-state detectors are so-called surface barrier devices with a depletion depth of 300 μm and a sensitive area of 50 mm^2. They are manufactured by Ortec Inc. A surface-barrier detector is a reverse-biased diode fabricated by depositing a thin layer of gold or aluminum on n- or p-type silicon, respectively, as shown in Figure 2.1. The metal also serves to attenuate light.

When an energetic particle enters into the detector, it loses its kinetic energy through lattice interactions. These particles supply energy to the lattice electrons and lift them from the valence band into the nearly empty conduction band. The high electric field across the depletion region causes the electron-hole pairs created by the ionization event to be quickly swept out of the depletion region yielding a current pulse proportional to the energy of the incident particle. The particle loses some energy in the thin metal surface layer (dead zone) so that the thickness of the layer determines the low energy limit of the detector. The resolution (full-width at half-maximum) is typically 7 keV at room temperature.

Current pulses produced by the detectors are at low levels and must be amplified. The pulse is also shaped to minimize the noise content. A block diagram of the amplifying and shaping circuits is shown in Figure 2.2. The shaping circuits employ a triple-integration and single-differentiation scheme to improve the noise characteristics of the pulse. The output of the pulse shaper then goes to the data-manipulation experiment.

Table 2.1 shows the specifications of the six detectors in each of the payloads. The detectors have different energy ranges to aid in the identification of the energetic particles: detector 3 has a broom magnet to screen out low energy electrons while detector 2, because of its thicker surface, screens out low energy protons; detector 6 detects both protons and electrons. If the output pulses from detectors 2 and 6 are similar while that from detector 3 is different, then the detected particles are electrons; if the output pulses from detectors 3 and 6 are similar while that from detector 2 is different, the detected particles are protons or heavier nuclei. Note that detectors 4 and 5 on the payload of Nike Apache 14.542 have aluminum surface layers while detectors 4 and 5 on the payload of Nike Apache 14.543 have gold surface layers. This is done because the payloads were designed to be launched at different times of the day into correspondingly different atmospheric brightness conditions (i.e., dusk and midnight).

More information on the solid-state detectors is given in *Voss and Smith* [1974, 1977]. The EPS experiment for Nike Apache payloads 14.542 and 14.543 is described in more detail in a report being prepared by K. L. Fries, L. G. Smith, and H. D. Voss.

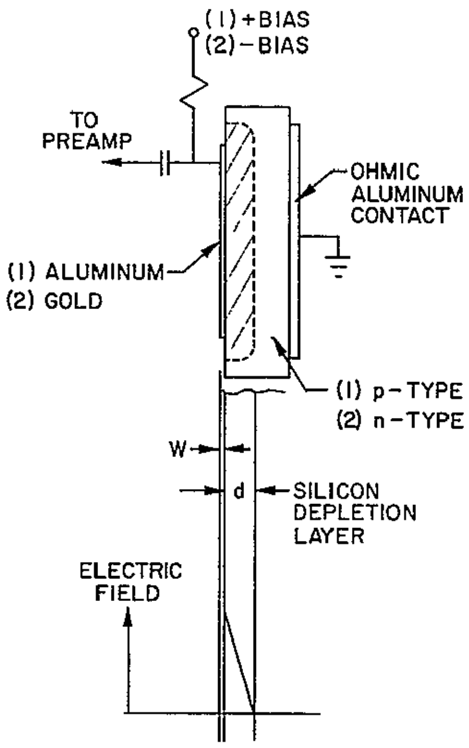


Figure 2.1 Surface barrier detector. Note the differences between the detectors with surface layers of (1) aluminum and (2) gold.

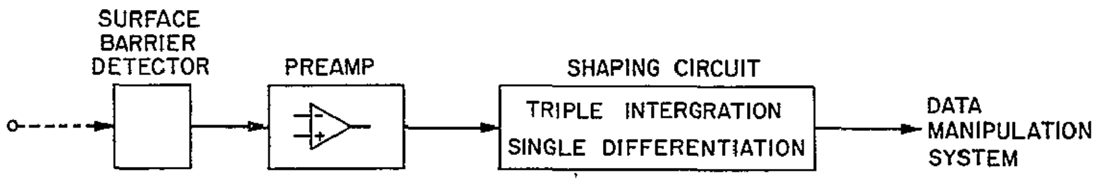


Figure 2.2 Major components of the solid-state detector instrumentation.

Table 2.1

Important characteristics of the six solid-state detectors in the payloads of Nike Apaches 14.542 and 14.543.

Designation‡	Geometrical Factor (cm^2 ster)	Front Surface Material (μg cm^{-2})**	Look Angle from Spin Axis (deg)	Energy Range (keV**) Electrons	Energy Range (keV**) Protons	Energy Channels	Pitch Angle Sectors	Comments
EPS 1 U	0.05	40-Au	90	>10	>15	12	15	High Resolution Detectors Energetic Particle Identification
EPS 1 D	0.05	40-Au	90	>150	>15	12	15	Electron Broom Magnet CWEF*
LPS 2 U	0.05	100-Al	90	>12	>55	11	15	For Energetic Particle Identification, CWEF
EPS 3 U	0.05	40-Al(Au)	45	>10	>25 (15)	12	15	Precipitated Flux
EPS 3 D	0.05	40-Al(Au)	135	>10	>25 (15)	12	15	Backscattered Flux
EPS 1 UD	0.05	40-Al	90	>10	>25	12	15	CWEF and Ion Mass Determination

* Comparison with earlier flights

** Parentheses for Nike Apache 14.543 only

‡ U indicates ascent and D indicates descent

2.3 *Electrostatic Analyzer*

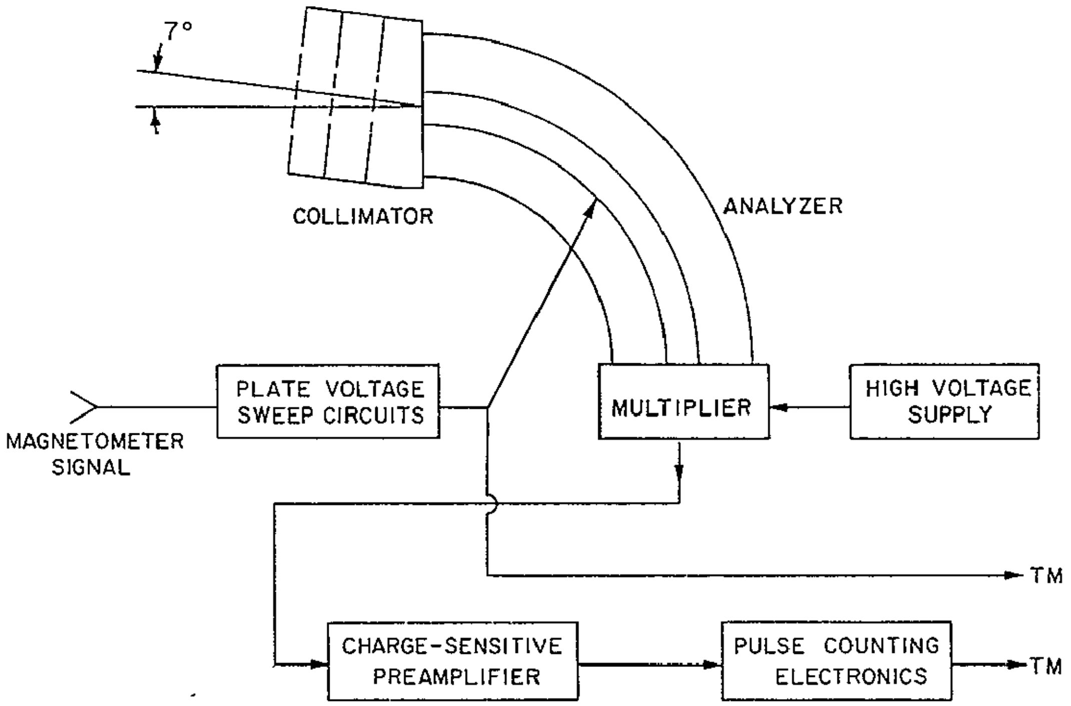
An ESA is included in the payloads to measure particle flux in the 0.5 to 10 keV range with a resolution of less than 1 keV. An ESA essentially consists of two cylindrical parallel plates followed by a detector. A voltage is applied to the plates to deflect particles into the detector. The plate voltage determines the energy of the particle which can successfully traverse the analyzer section.

A block diagram of the ESA is shown in Figure 2.3. The plate voltage is stepped logarithmically through six levels and then is allowed to decay exponentially to zero. The voltage sequence alternates between positive polarity and then negative polarity and is symmetrical about zero. This allows the ESA to scan for electrons and protons as well as positive ions. A fraction of the output of the sweep circuit is also telemetered to ground so that the actual plate voltage can be continuously monitored. The voltage sequence is synchronized to the rocket spin such that one step corresponds to one rocket rotation and the decay to two rocket rotations. The total sequence thus repeats after 16 rotations (i.e., six positive-going steps and decay plus six negative-going steps and decay).

The detector is an electron multiplier, Johnston Model MM1-5NG. An incoming particle strikes the first of the copper-beryllium dynodes and in so doing generates secondary electrons. An applied potential causes these electrons to strike succeeding dynodes thereby generating more electrons. After 20 stages the electrons strike an anode and the charge pulse is collected at the anode.

The signal from the detector is processed in the payload in two independent ways. In the first a charge preamplifier in the ESA is used to drive a staircase generator which gives a direct method of measuring the count rate. This does not involve the data-manipulation experiment.

The second method of processing the ESA data extracts considerably more information: it gives pulse-height analysis of the detector output (which allows energetic positive ions to be identified) and it provides pitch-angle information. For this method the detector is used with the preamplifier and charge shaping circuits of the type developed for the solid-state detectors (see Figure 2.2). The signal is then processed in the data-manipulation experiment in exactly the same way as the signals from the solid-state detectors. The ESA is described in detail in *Pozzi et al.* [1979].



7°
COLLIMATOR
ANALYZER
MAGNETOMETER SIGNAL
PLATE VOLTAGE SWEEP CIRCUITS
MULTIPLIER
HIGH VOLTAGE SUPPLY
TM
CHARGE-SENSITIVE PREAMPLIFIER
PULSE COUNTING ELECTRONICS
TM

Figure 2.3 Block diagram of the electrostatic analyzer. The acceptance angle of the collimator is exaggerated.

3. GENERAL DESCRIPTION OF THE EXPERIMENT

This chapter presents a general description of the data-manipulation experiment in terms of the hardware and software involved. The information content of the output is also explained. A block diagram of the complete system is shown in Figure 3.1.

3.1 *Z-80 Microcomputer*

The Z-80 microcomputer family of components (available from Zilog and Mostek) has been used to implement the experiment. A microcomputer consists of three main parts: a CPU (the microprocessor), memory, and interface circuits to peripheral devices. All of these main parts are MOS LSI devices and their small size allows a powerful system to be constructed in a small physical space.

The Z-80 CPU uses 8-bit data words and 16-bit addresses (i.e., a 2^{16} or 65,536 byte address space). It contains eighteen 8-bit registers and four 16-bit registers. The registers include two sets of six general purpose registers that may be used individually as 8-bit registers or in pairs as 16-bit registers. There are also two sets of accumulator and flag registers. There are six special purpose registers such as a program counter and a stack pointer. The Z-80 CPU can execute 158 instructions; these include special I/O instructions which allow the CPU to address 256 I/O devices in addition to the memory addresses. The CPU is contained in a 40-pin dual-in-line package (DIP).

The Z-80 Parallel I/O Circuit (PIO) is a programmable device that provides two 8-bit parallel interfaces between peripheral devices and the CPU. Each of the two interfaces also has two 'handshake' lines so that the peripheral devices can control when data is input or output. The PIO can be programmed to operate in different modes depending on the application. For instance it can be programmed to be used as an input port that generates an interrupt to the CPU when one of the handshake lines is toggled; or interrupt capability and the handshake lines can be disabled so that data is input or output under software control. The PIO is contained in a 40-pin DIP.

The Z-80 counter-timer circuit (CTC) is a programmable four-channel device that provides counting timing functions for the CPU. It can also be programmed to act as an interval timer. In this mode a time constant is loaded into a down counter in the CTC by the CPU and the system clock is then used to decrement the counter; when it reaches zero, an interrupt is generated. Other modes allow external circuits to decrement the down counter or to be driven by

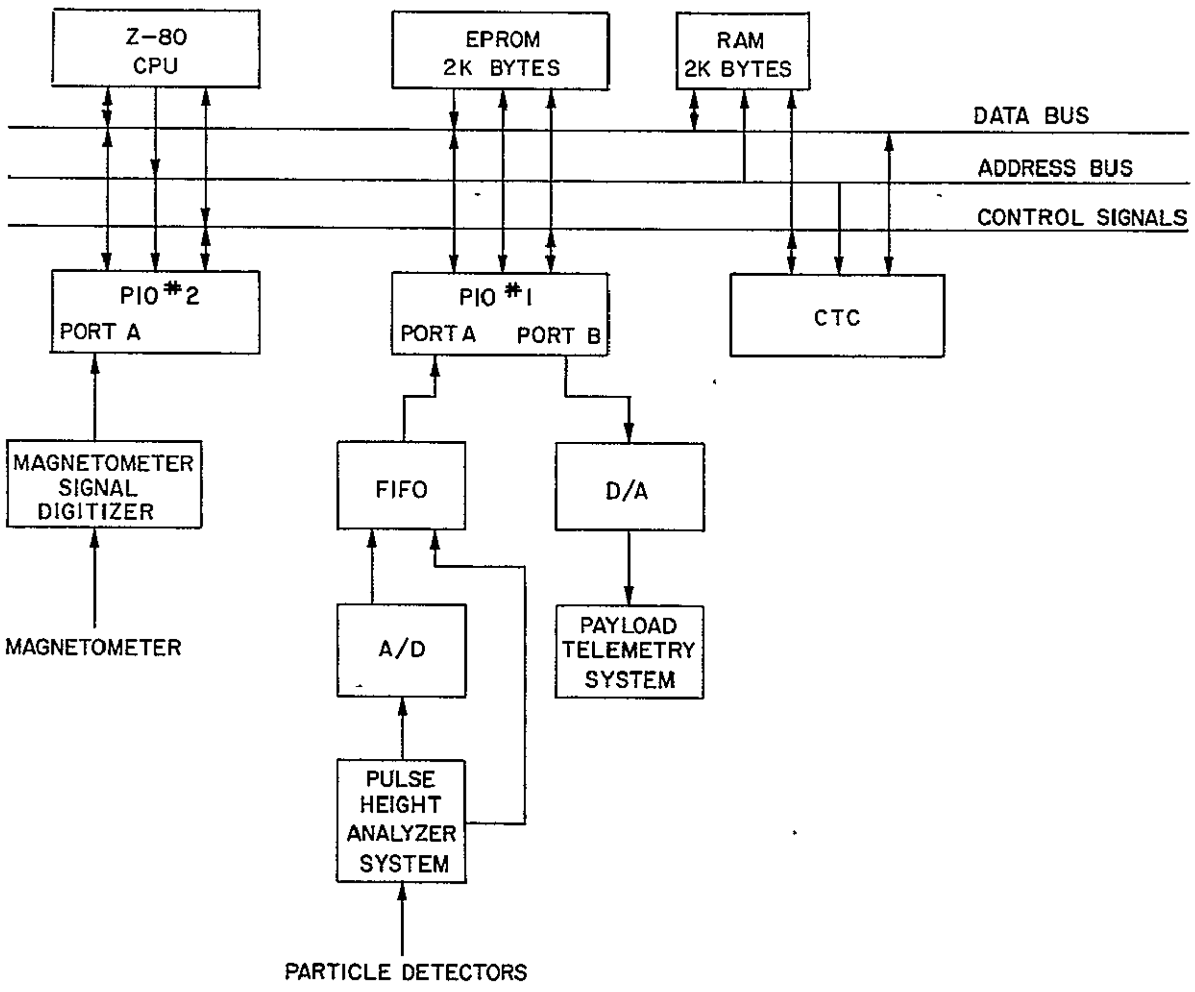


Figure 3.1 The data-manipulation system.

a divided system clock. The CTC is packaged in a 28-pin DIP.

In the system described here the PIO's and the CTC are programmed as follows. One port of PIO #1 (Port A) is programmed as an interrupting input port with the handshake lines enabled. Port B is programmed as a non-interrupting output port and the handshake lines are not used. Port A of PIO #2 is programmed as a non-interrupting input port and the handshake lines are again not used. The CTC is programmed to interrupt the CPU approximately every 1.5 ms.

An Intel 2716 EPROM is used to contain the non-volatile program for the rocket. It must be programmed by a special programmer (described in Chapter 6) before it is placed in the circuit. Its contents cannot be changed by the CPU after it is in the circuit although an ultra-violet light can be used to erase it if the program is to be changed. It can contain a 2 K byte program and is packaged in a 24-pin DIP.

The RAM portion of the memory consists of four Intel 2114 memory chips (also available from EMM SEMI). The 2114 has 1 K × 4 bits and is packaged in an 18-pin DIP.

3.2 *Peripheral Devices*

When a particle is sensed by one of the detectors, the pulse that is generated (as described in the previous chapter) goes to a pulse-height analyzer (PHA). It latches onto the peak of the pulse and holds it until a reset signal is received. The peak of the highest level pulse is held if more than one is received before it is reset.

Four detectors are used in the PHA system although six particle detectors and the electrostatic analyzers are accommodated during a flight. An apogee switch is used to switch two of the particle detectors out and the electrostatic analyzer and another particle detector in at apogee. Thus the upleg and downleg records of a rocket flight produced by two of the PHA's are for different devices.

The four PHA outputs are sequentially multiplexed to the analog input of the A/D. Every 50 μs (actually 51.2 μs) the multiplexer switches to the next PHA, thus each PHA is reset every 200 μs. Two bits from the PHA system are included with each PHA output going to the A/D to identify which PHA produced it.

The A/D converts the 0 to 10 V analog signal from the PHA system to an 8-bit digital signal although only the upper 4 bits are used. An 8-bit converter was used due to the unavailability of a suitable 4-bit converter. The A/D used in the system is a Datel ADC-EH8B which converts the signal by the successive approximation method. The conversion time is approximately 2.4 μs.

The first-in first-out memory (FIFO) is used to store data in a queue while the CPU is servicing an interrupt from the CTC (the CTC has a higher priority than PIO #1). Nine-bit words are accepted at the input and automatically shifted toward the output and are removed at any rate in the same sequence in which they were entered. Up to 40 9-bit words can be contained i the FIFO, but only 6 of the 9 bits are used in this system (4 bits from the A/D, 2 identifying bits from the PHA system). The three unused inputs are tied to ground. The FIFO allows data to be taken at a faster rate from the PHA system without losing data when the CPU cannot be interrupted by the PIO.

The D/A converts the system output from PIO #2 to an analog signal varying between 0 and 5 V. The D/A used in the system, a Micro-Networks MN3020, converts an 8-bit word to an analog signal between 0 and 10 V. Since the telemetry circuitry requires a signal between 0 and 5 V, the most significant bit of the D/A is not used and is tied to ground.

The magnetometer signal digitizer (MSD) converts the magnetometer signal to a 4-bit digital signal that represents the rocket's spin position. The magnetometer signal is a sine wave whose frequency equals the spin rate of the rocket. The conversion is done by incrementing a 4-bit counter at a constant rate (100 Hz) and resetting it when the magnetometer signal crosses a zero reference point in the downward direction. Therefore, the counter is reset once per revolution of the rocket.

The circuit of the MSD is described in *Leung et al.* [1979]. It can be noted here that when the rocket is not spinning as during testing and before launch, the 4-bit counter resets itself, allowing the operation of the experiments to be observed.

The rotation is divided into equal sectors only for certain values of rocket spin rate: e.g., 6.67 Hz will give exactly 15 sectors (i.e., 100/6.67); 6.25 Hz will give exactly 16 sectors; and 5.88 Hz will give exactly 17 sectors. In general, because the rocket spin rate is pre-set only within a limited range, there will not be an integer number of sectors.

In the following section, for simplicity, the data manipulation will be described for the special case of a spin rate of 6.25 Hz. Later, in Section 5.2.3, the effect of other values of spin rate will be considered in detail.

3.3 *Data Manipulation*

As indicated above, the rocket rotation is divided into 16 discrete sectors (for a spin rate of 6.25 Hz). This is shown in Figure 3.2(a). The 4-bit number provided by the MSD is used to generate an address for a memory sector that corresponds to a rotation sector. The offset within each memory sector is provided by the data from the FIFO. The entire memory address is formed by concatenating the 4-bits from the MSD with the data from the FIFO, as shown in Figure 3.2(b).

The lower 1 K bytes of the RAM are divided into 16 sectors corresponding to the rocket rotation sectors as shown in Figure 3.2(c). Each of these 16 sectors is further divided into four areas corresponding to the PHA's. Each of these areas contains 16 bytes that correspond to 16 energy levels of the particles. Figure 3.2(d) shows which parts of the 10-bit address correspond to the rotation sectors, PHA's and energy levels.

When data is input to Port A of PIO #1, an interrupt is generated by the PIO. The CPU services the interrupt by inputting the data. An address is then generated as stated above and the location pointed to is incremented. A flowchart of the input routine alogrithm is shown in Figure 3.3.

The CTC generates an interrupt every 1.536 ms which causes the CPU to go to an output routine. Data outputs are done in synchronism with rocket rotation. One memory sector (corresponding to a rotation sector) is output during 1 + 1/16 revolution of the rocket. Data starts being output when the rocket has rotated to the rotation sector directly after the one corresponding to the memory sector to be output. For example, data from sector #1 starts being output when the rocket has rotated to sector #2 (referring to Figures 3.2(a) and 3.2(c)). Note that this results in the data in each sector being accumulated over 17 revolutions (i.e., 16 + 16 × 1/16). Since the duration of each sector is 10 ms (the period of the 100 Hz oscillator), it follows that the output sequence represents the particle data (in each sector) accumulated for a period of 170 ms (i.e., 17 × 10 ms) less the reset time of the PHA (42.5 ms). The output sequence repeats at intervals of 2.72 s (i.e., 17 × 160 ms) for this spin rate (6.25 Hz).

The output routine algorithm as shown in Figure 3.4 outputs a byte of data when the output flag has been set. If the flag has not been set, the routine checks to see if the rocket has rotated to within the next sector. If it has, the output flag is set and the routine outputs the first byte in the sector.

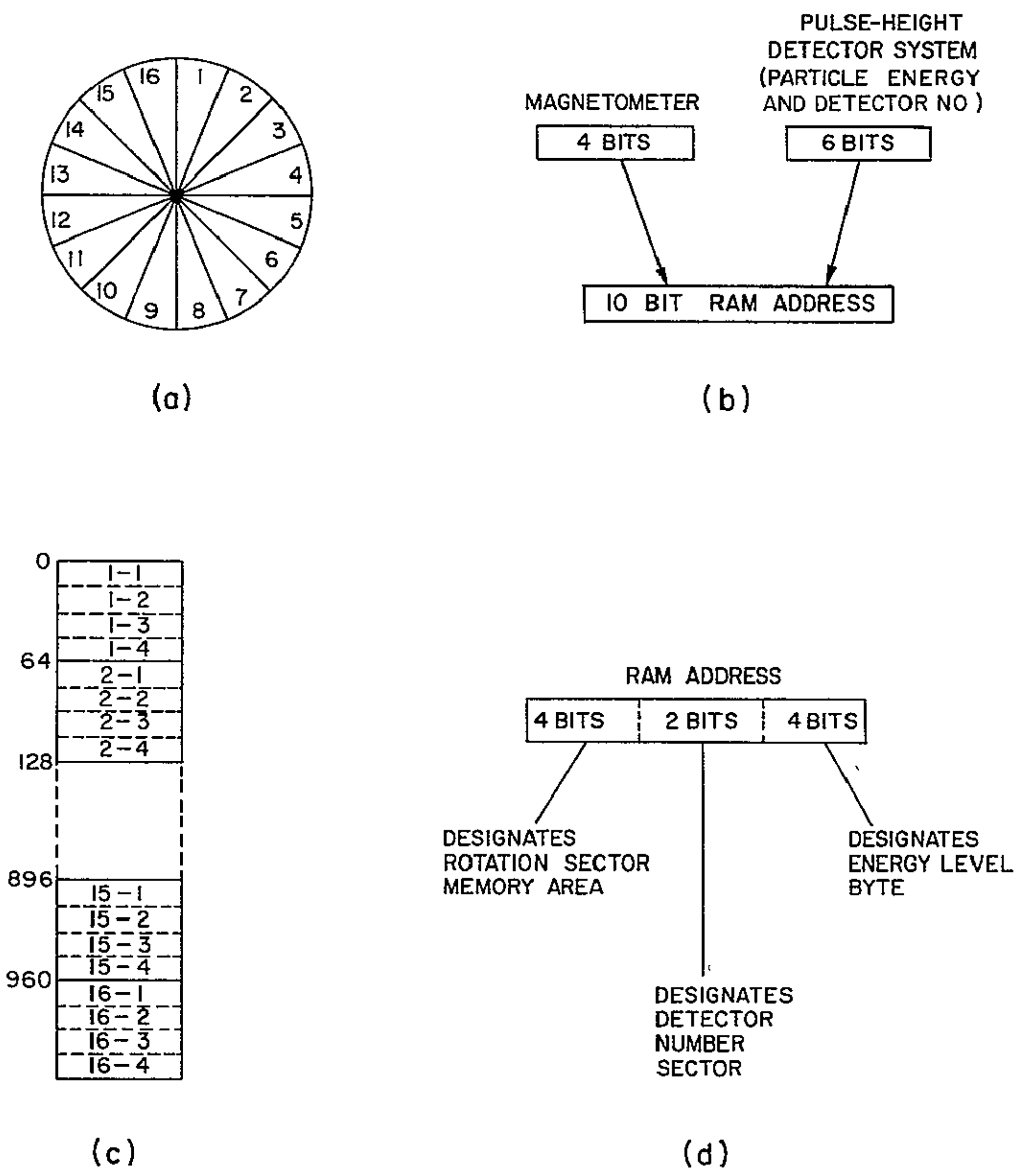


Figure 3.2 (a) Rocket rotation sectors; (b) Address formed from the two data words; (c) Memory (d) RAM address.

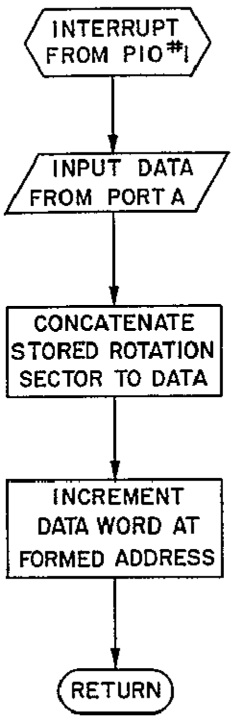

INTERRUPT FROM PIO #1
INPUT DATA FROM PORT A
CONCATENATE STORED ROTATION SECTOR TO DATA
INCREMENT DATA WORD AT FORMED ADDRESS
RETURN

Figure 3.3 Input routine algorithm.

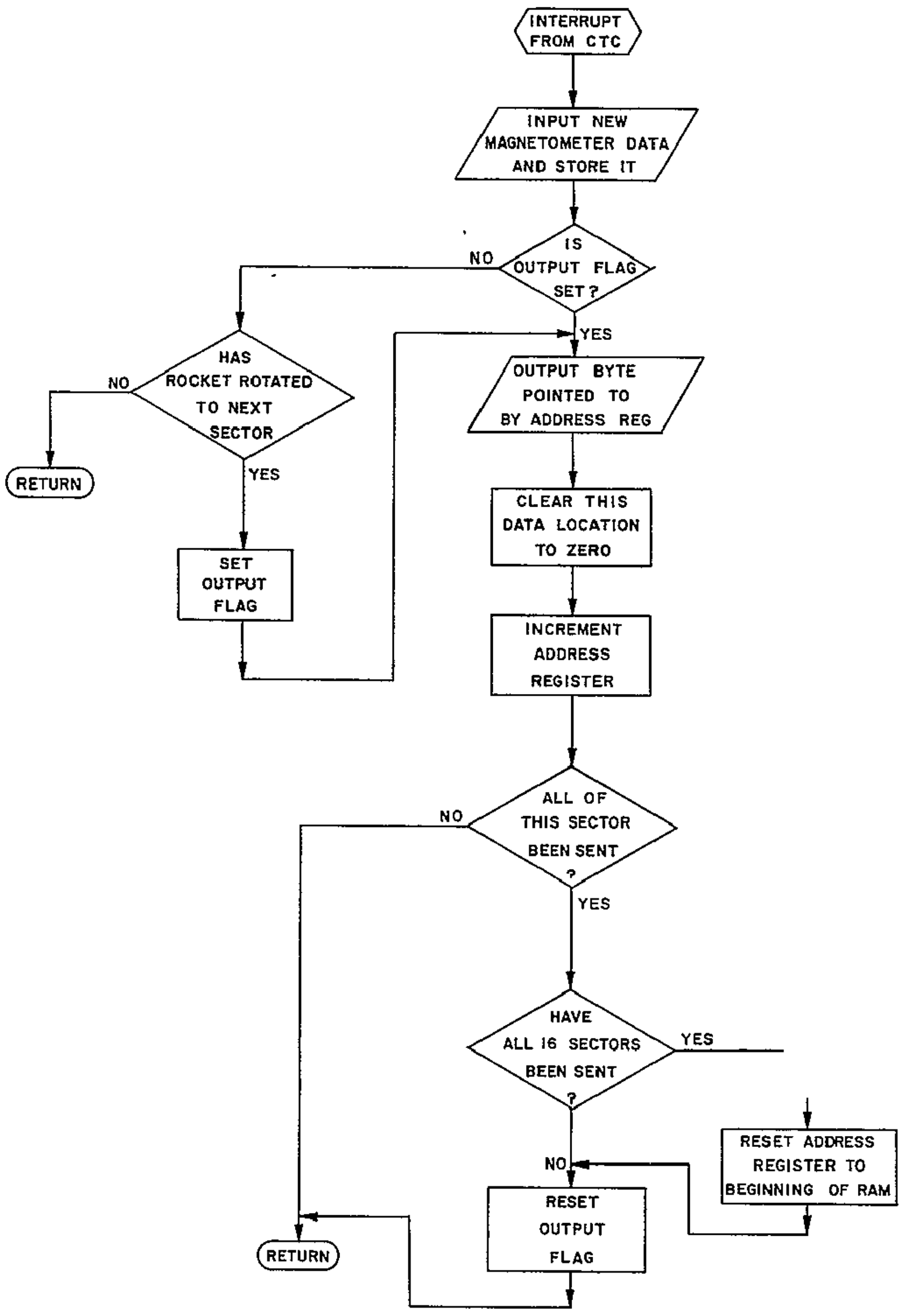

INTERRUPT FROM CTC
INPUT NEW MAGNETOMETER DATA AND STORE IT
IS OUTPUT FLAG SET?
NO
YES
HAS ROCKET ROTATED TO NEXT SECTOR
NO
RETURN
YES
SET OUTPUT FLAG
OUTPUT BYTE POINTED TO BY ADDRESS REG
CLEAR THIS DATA LOCATION TO ZERO
INCREMENT ADDRESS REGISTER
ALL OF THIS SECTOR BEEN SENT ?
NO
YES
HAVE ALL 16 SECTORS BEEN SENT ?
YES
NO
RESET ADDRESS REGISTER TO BEGINNING OF RAM
RESET OUTPUT FLAG
RETURN

Figure 3.4 Output routine algorithm.

Otherwise the routine just updates the stored magnetometer data and the routine is exited.

An address register (two of the 8-bit general purpose registers) is used to point to the byte to be output. After the data at a RAM location has been output to the D/A, the location is cleared. The address register is then incremented and checked to see if all of the sector has been sent. If it has, the output flag is reset. If all of the 16 memory sectors have been sent (1 K bytes), the address register is reset to the beginning address of the RAM.

An example of the output from the system is shown in Figure 3.5. Note that the data from each detector has a 5 V peak at the beginning. Although this peak should by definition correspond to the number of zeroes (no particles or particles below the sensitivity of the detectors) from each detector, it is automatically set to the maximum voltage by the CPU since the number of zeroes would be large anyway. The peaks are also useful in that they separate the detectors allowing the data to be read easily.

Note also the signal labeled "marker" at the beginning of the data for sector #1. This signal is included to identify which sector is being output. It varies between 0 and approximately 5 V in 16 discrete steps. For instance the marker for sector #0 is 0 V, the marker for sector #7 is approximately 2.5 V and the marker for sector #15 is approximately 5 V.

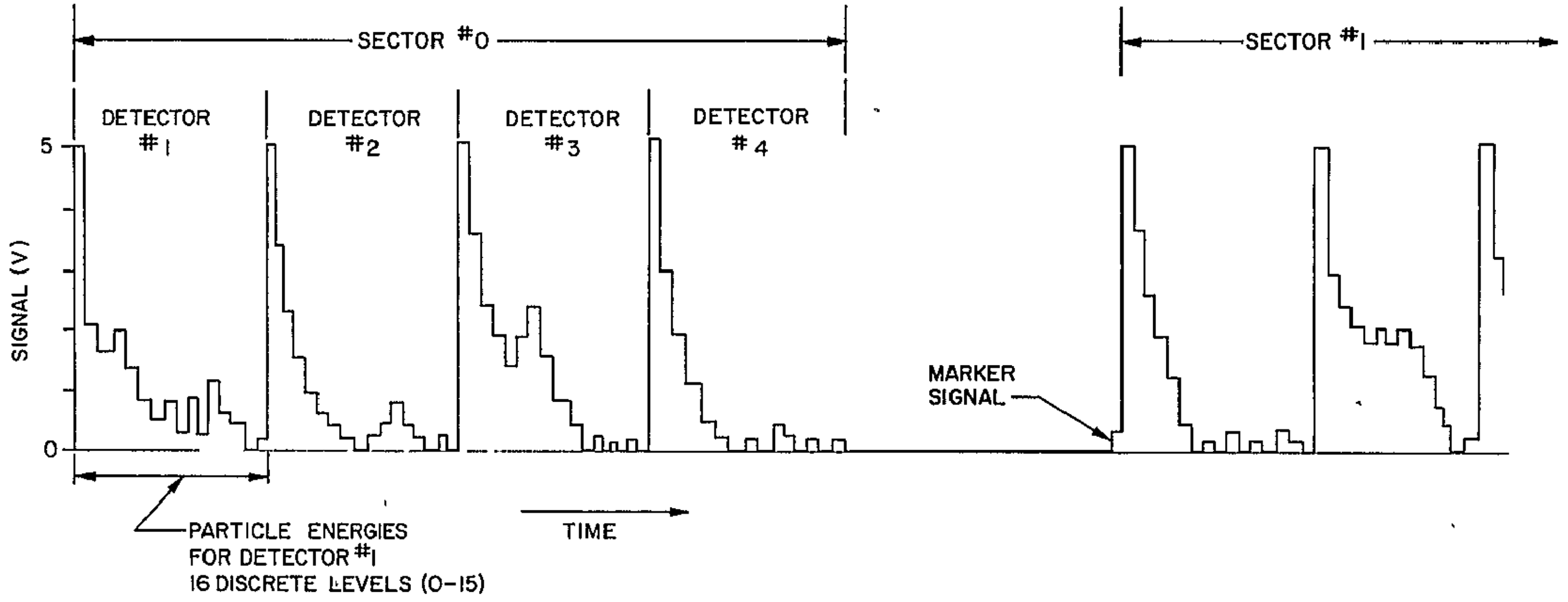


Figure 3.5 Example of output from D/A. The ordinate is the telemetered signal (in volts) and is proportional to number of units (5V ≡ 32 particles) in the accumulation period.

4. HARDWARE

The designs of the microcomputer, the A/D-FIFO interface, and the D/A interface are discussed in this chapter. The microcomputer and the A/D and D/A converters are contained on four printed circuit boards as shown in Figure 4.1. The circuit board for PIO #2 also contains the MSD and the PHA. A photograph of this board is shown in *Leung et al.* [1979]. Figure 4.2 shows the location of the system on the payload.

Some control signals used in the system are defined in Table 4.1. The detailed timing diagrams of these signals are not presented here as they are not necessary for an understanding of the system operation, but they are available in the MOSTEK data books in the reference list.

4.1 *CPU-Memory Board*

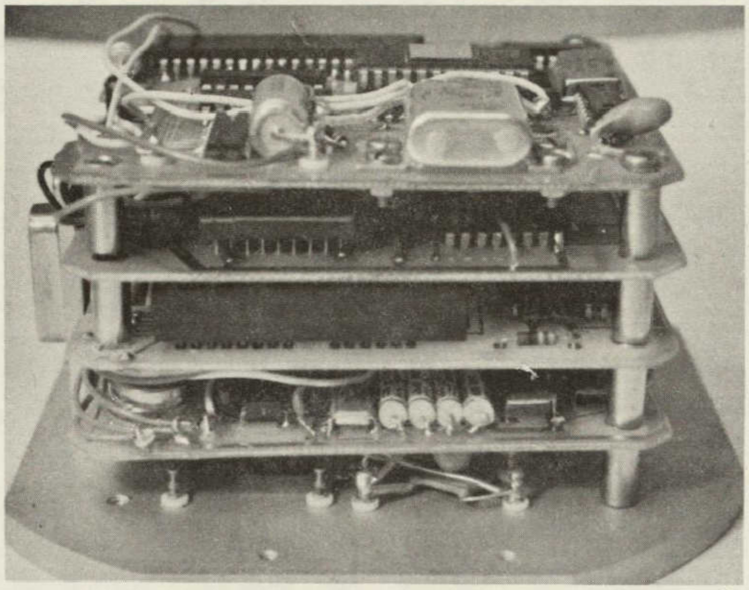
The CPU-memory board contains the Z-80 CPU, EPROM and RAM memory, reset circuitry and the system clock oscillator as shown in Figure 4.3. A flexible ribbon cable (Ansley #FSN-21A-20) from connector #1 provides the necessary address, data, and control lines for the operation of the other three boards in the system.

A detailed schematic of the CPU-memory board is shown in Appendix I.1.

4.1.1 *Memory address decoding.* A one-of-eight decoder (Intel 8205 or T.I. 74LS138) is used to enable the RAM and EPROM memories. A diagram of the decoder circuit pin connections is shown in Figure 4.4(a). Address bits A11, A12, and A13 from the Z-80 CPU define which output of the decoder will go low. The $\overline{\text{RD}}$, $\overline{\text{WR}}$, and $\overline{\text{MREQ}}$ control lines are used as enabling inputs to the decoder. If $\overline{\text{RD}}$ or $\overline{\text{WR}}$ goes low and, in addition, $\overline{\text{MREQ}}$ goes low, the decoder is enabled and one of the eight outputs specified by the address lines will go low. Only three of the eight outputs are used because only 4 K bytes of memory are in this system.

This particular decoder circuit was used because only one external gate was needed to generate the three select signals (the abbreviation for a select signal is $\overline{\text{CS}}$ where the line indicates active low) and because it will allow easy system expansion in the future with a minimum of changes on the board (i.e., only additional address lines and the other $\overline{\text{CS}}$ lines need be brought out to add another memory board). The resulting memory address map is shown in Figure 4.4(c). The 1 K gap between the RAM segments exists because the decoder generates $\overline{\text{CS}}$'s for 2 K memory segments and the RAM chips are 1 K segments.

(a)



(b)

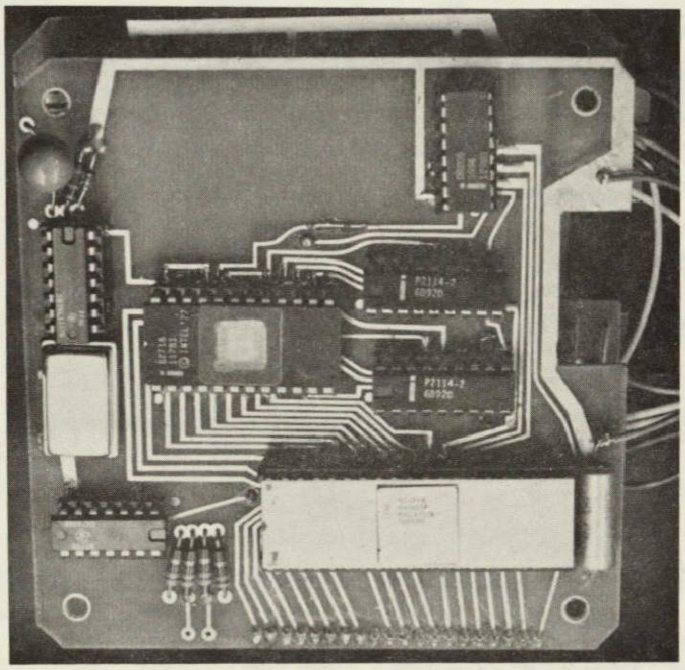
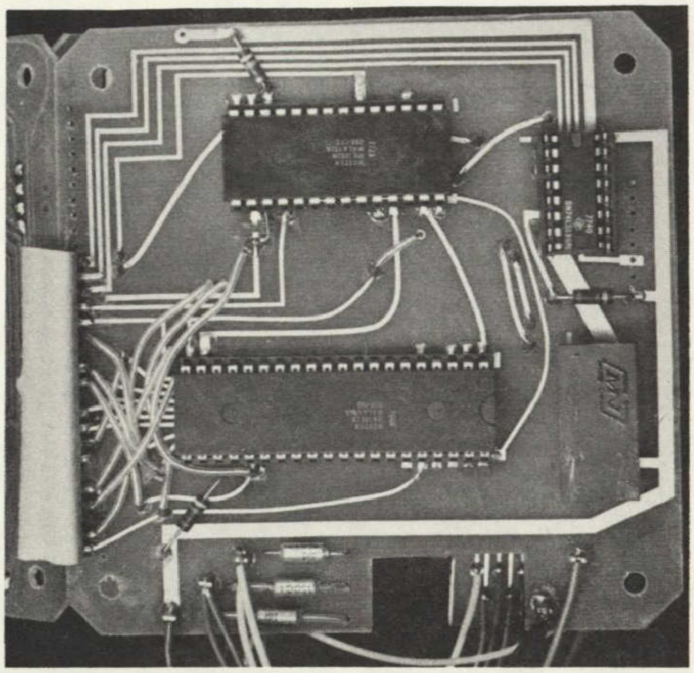


Figure 4.1 Data-manipulation system circuit boards: (a) boards connected for installing in payload, (b) CPU-memory board.

(c)



(d)

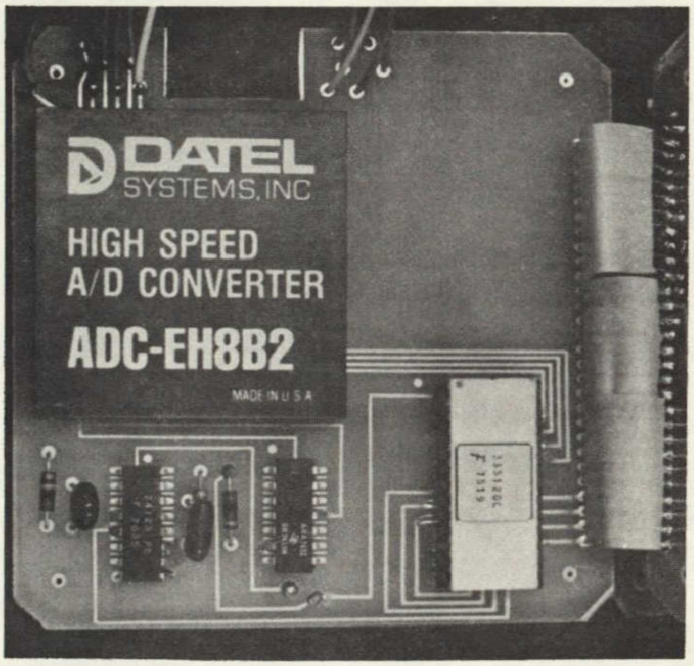


Figure 4.1 Data-manipulation system circuit boards (continued). (c) PIO #1-CTC board, and (d) A/D FIFO board.

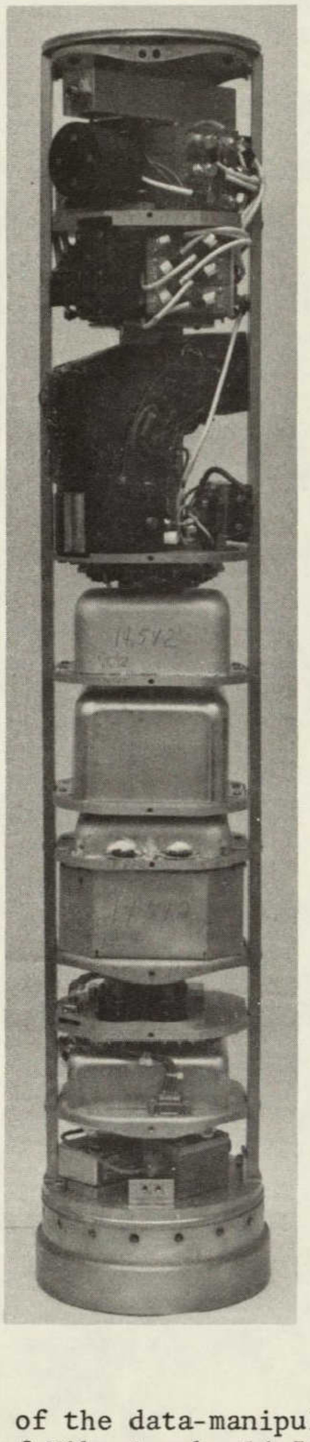


Figure 4.2 Location of the data-manipulation system in the payload of Nike Apache 14.542.

Table 4.1

Z-80 Control signals.

$\overline{\mathrm{M1}}$: An output signal from the CPU that indicates the current machine cycle is the instruction fetch cycle. $\overline{\mathrm{M1}}$ also occurs with $\overline{\mathrm{I/OREQ}}$ to indicate an interrupt acknowledge cycle, and it indicates to I/O devices when to place data on the data bus.

$\overline{\mathrm{MREQ}}$: The memory request signal is an output signal from the CPU that indicates the address bus holds a valid address for a memory read or memory write operation.

$\overline{\mathrm{I/OREQ}}$: The I/O request signal is an output signal from the CPU that indicates the lower half of the address bus holds a valid I/O address for an I/O read or write operation.

$\overline{\mathrm{RD}}$: The read signal is an output signal from the CPU that indicates data should be placed on the data bus by memory or an I/O device.

$\overline{\mathrm{WR}}$: The write signal is an output signal from the CPU that indicates that the CPU data bus holds valid data to be stored in the addressed memory or an I/O device.

$\overline{\mathrm{INT}}$: The interrupt request signal is generated by I/O devices. The request will be honored at the end of the current instruction by the CPU if software has enabled an internal interrupt enable flip-flop.

$\overline{\mathrm{NMI}}$: The non-maskable interrupt request signal is also generated by I/O devices. The request will always be honored at the end of the current instruction.

$\overline{\mathrm{RST}}$: Reset signal

Φ: System clock

NOTE: The lines above the signals mean they are active low.

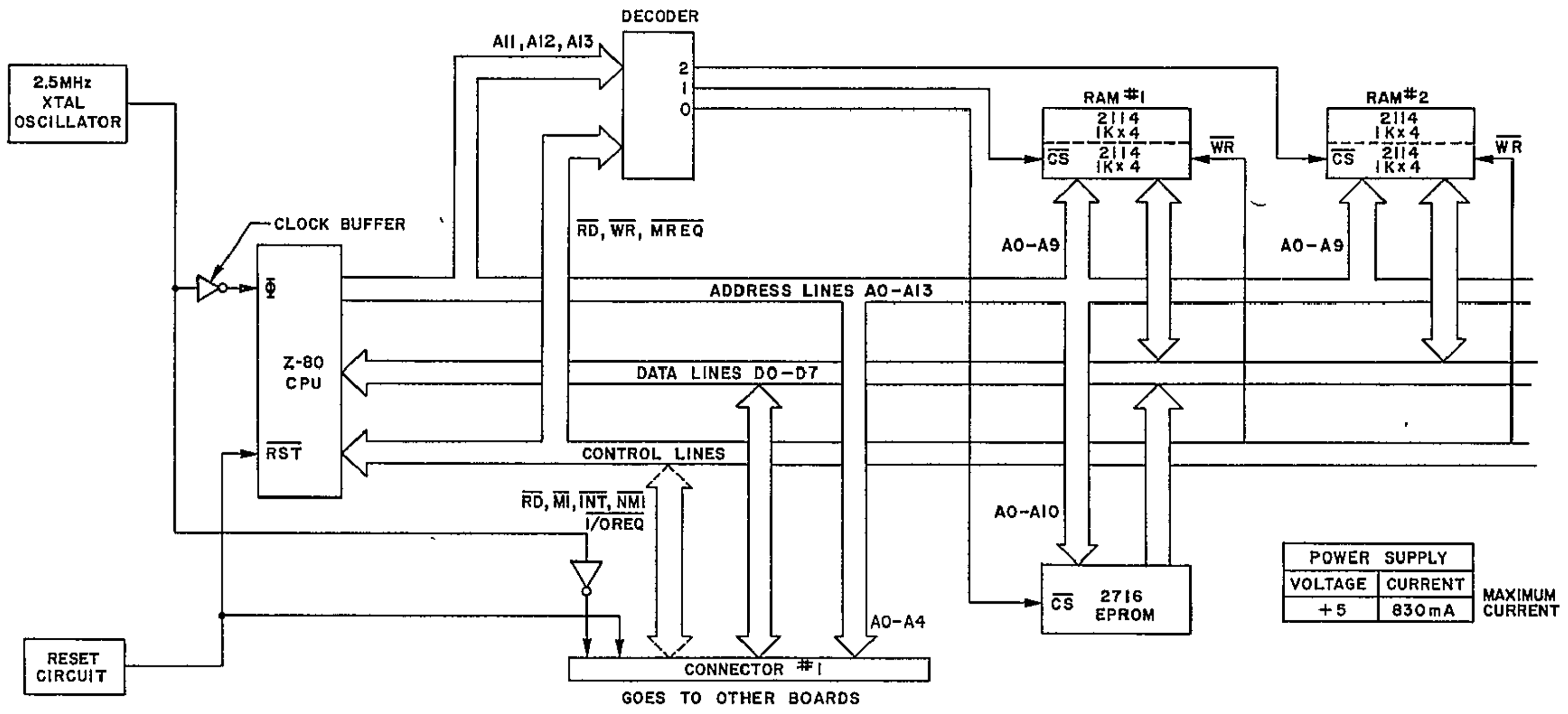
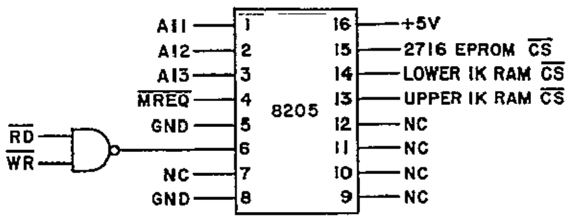


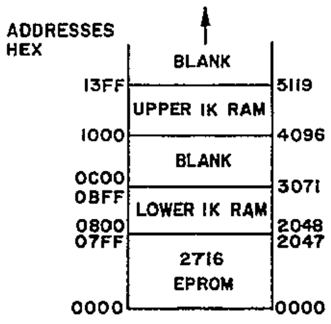
Figure 4.3 The CPU-Memory board. An arrow on one end of a line indicates a single direction signal; arrows on both ends indicate a bidirectional signal; dotted arrows indicate some lines go in one direction and some the other direction.



(a)

					8205 PIN NUMBERS		
A11	A12	A13	$\overline{MREQ}$	$\overline{RD}\,\overline{WR}$	15	14	13
L	L	L	L	H	L	H	H
L	L	H	L	H	H	L	H
L	H	H	L	H	H	H	L
H	X	X	L	H	H	H	H
X	X	X	H	X	H	H	H
X	X	X	X	L	H	H	H

(b)



(c)

Figure 4.4 Address decoding: (a) Connection of the 8205 one-of-eight decoder; (b) Truth table (X = don't-care state); (c) Memory addresses.

4.1.2 *Reset circuit.* The Z-80 CPU and CTC have a reset input that cannot be brought high until the data, address and control lines have stabilized after power-on. The reset input can also be used to restart the CPU and CTC in their initial states after power-on by bringing it low for a short period, but this feature was not needed in the data manipulation experiment since it is a dedicated system (i.e., the software is not changed). When the reset line goes high, the CPU starts at address zero and the CTC is in its initial state (i.e., it is not yet programmed for operation). The PIO does not have a reset input but it is in its initial state automatically after a power-on.

The reset circuit itself is simple as shown in Figure 4.5. When power is applied, the 68 μf capacitor starts charging through the 10 kΩ resistor. When the Schmitt trigger input of the first NAND gate reaches approximately 1.6 V, it switches and an instant later the reset line goes high. If for some reason power is interrupted to the system, the capacitor discharges rapidly through the diode. The reset line goes high approximately 77 ms after power is applied [i.e., $-(10^4)(68 \times 10^{-6})\ln(1.6/5)$].

4.1.3 *Clock.* A Motorola K1115A 2.5 MHz crystal oscillator provides the system clock signal. It was used because of its small size, slightly larger than a 14-pin DIP, and because it provides a TTL compatible output. As shown in Figure 4.6 a 7404 with 330 Ω pull-up resistors on the outputs is used to buffer the clock signal. The 330 Ω pull-up resistors are required in the Z-80 components' specifications. The K1115A oscillator is specified to tolerate a minimum shock of 100 G's for 0.1 milliseconds with three shocks in each plane. This specification is adequate for a rocket flight: the oscillator performed well both in pre-flight tests and in actual flight of Nike Apache 14.543.

4.1.4 *Read-write control.* As shown in Figure 4.3 the 2114 RAM circuits receive the $\overline{\text{WR}}$ control signal. It controls whether data is input or output: a high level causes data to be output by the RAM's; a low level causes data to be input (the $\overline{\text{CS}}$ lines must be low also).

4.2 *PIO #1 and CTC Board*

This printed circuit board contains PIO #1, the CTC and the D/A as shown in Figure 4.7. A detailed schematic of the board is shown in Appendix I.2.

The 8205 (or 74LS138) decoder on this board is used to select which I/O port is enabled. PIO #1 has I/O addresses 1 through 3, the CTC has 4 through 7, and PIO #2 has 8 through 11. The other outputs of the decoder are brought out on connector #2 for future expansion of the I/O capability.

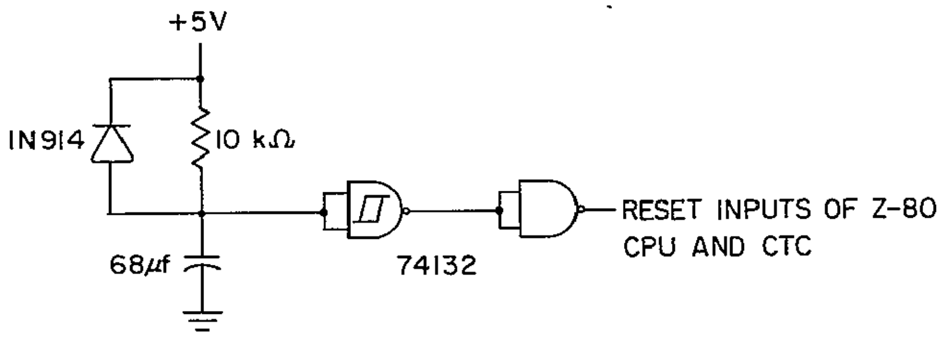


Figure 4.5 Reset circuit.

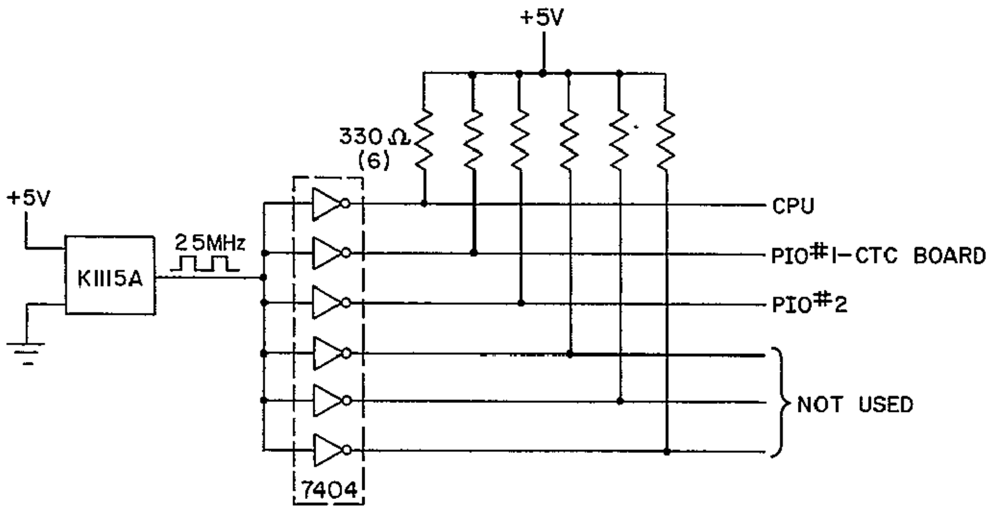


Figure 4.6 System clock.

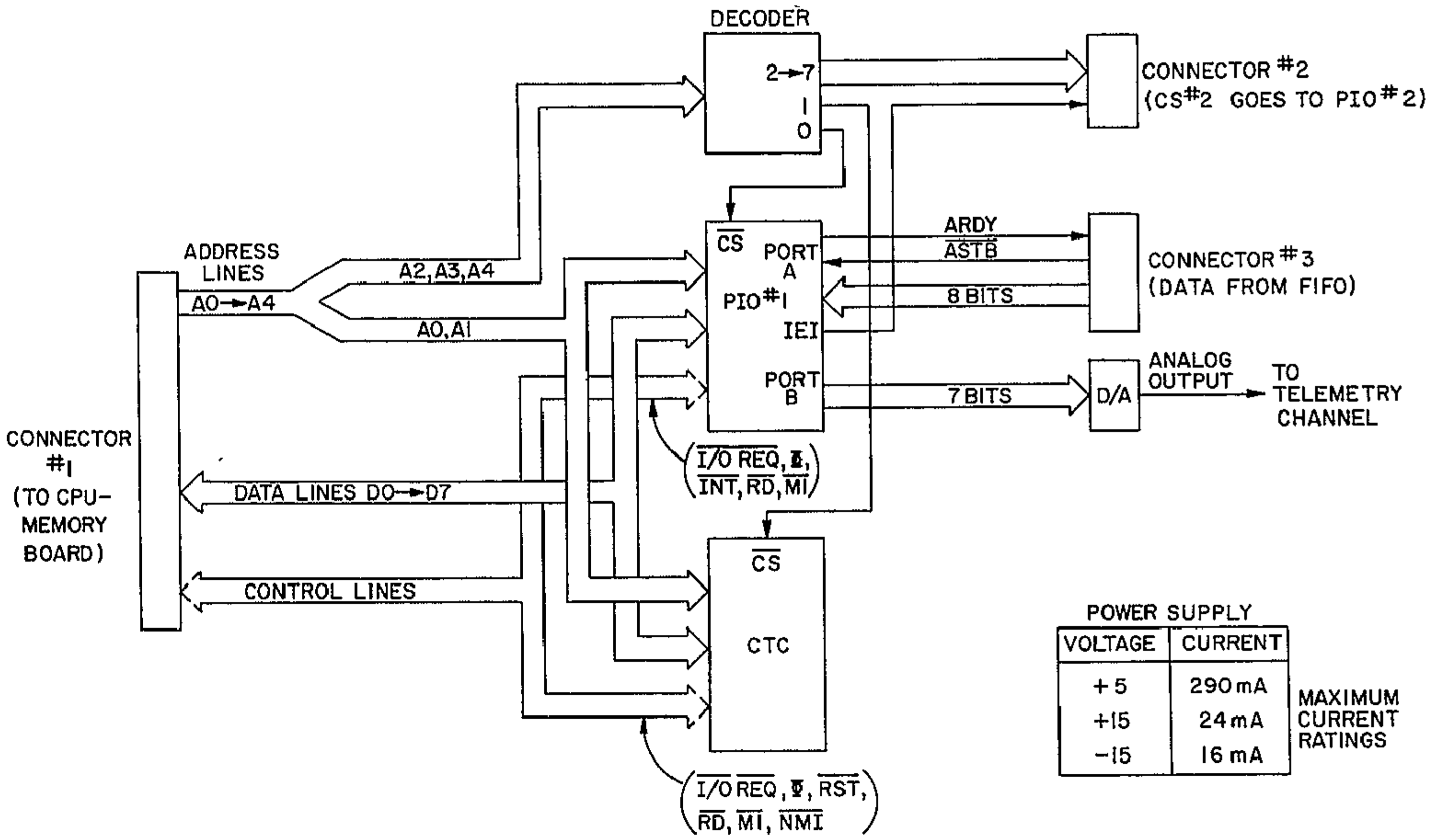


Figure 4.7 The PIO#1-CTC board. The arrows have the same meaning as in Figure 4.3.

The interrupt enable input (IEI) signal from PIO #1 is brought out on connector #2 so that other I/O ports added in the future can generate maskable interrupts. This signal would be used in the daisy-chain interrupt mode (i.e., Mode 2) described in the MOSTEK Z-80 literature in the reference list. This signal is not used in this system, however, since the Mode 1 interrupt option is used, as described in the following chapter.

Even though there are only two ports, the PIO's require four addresses because each port has control logic that must be addressed individually, as shown in Figure 4.8. This logic is programmed to control whether a port is an input or an output port, the handshake lines, and the generation of interrupts.

The two handshake lines, RDY and $\overline{\text{STB}}$, control the data input to a port and the generation of an interrupt to the CPU when the port is programmed to operate in the appropriate mode (i.e., Mode 1, consult the PIO data book). In the data manipulation system the handshake lines are used only for Port A of PIO #1.

The timing diagrams for ARDY and $\overline{\text{ASTB}}$ are shown in Figure 4.9(a) where the "A" indicates they are for Port A. Data is loaded into the Port A input register by applying a pulse of 150 ns (or greater) to the $\overline{\text{ASTB}}$ pin (normally high, pulsed low). ARDY indicates when the input register is empty and is ready to accept an input (normally low, active high). When data is loaded into the input register, an interrupt is generated by the PIO by bringing the $\overline{\text{INT}}$ line low. When the CPU has read the input register contents, ARDY returns high. When the PIO is initialized after power-on, Port A must be read once to bring ARDY high even though there is no data in the input register.

As stated in the previous chapter the CTC generates an interrupt when an internal down-counter reaches zero. The CTC in this system is connected to the $\overline{\text{NMI}}$ line of the CPU which is used to generate non-maskable interrupts. Thus it has a higher priority than PIO #1 or any I/O port that would be added in the future.

4.3 *A/D and FIFO Board*

This board contains the A/D, the FIFO, and associated timing logic as shown in Figure 4.10. A detailed schematic of the board is shown in Appendix I.3. Data is input to this board from the PHA system and is output to Port A of PIO #1.

The A/D starts converting an analog signal to its digital representation when a pulse of 100 ns minimum width is applied to its start-convert pin as

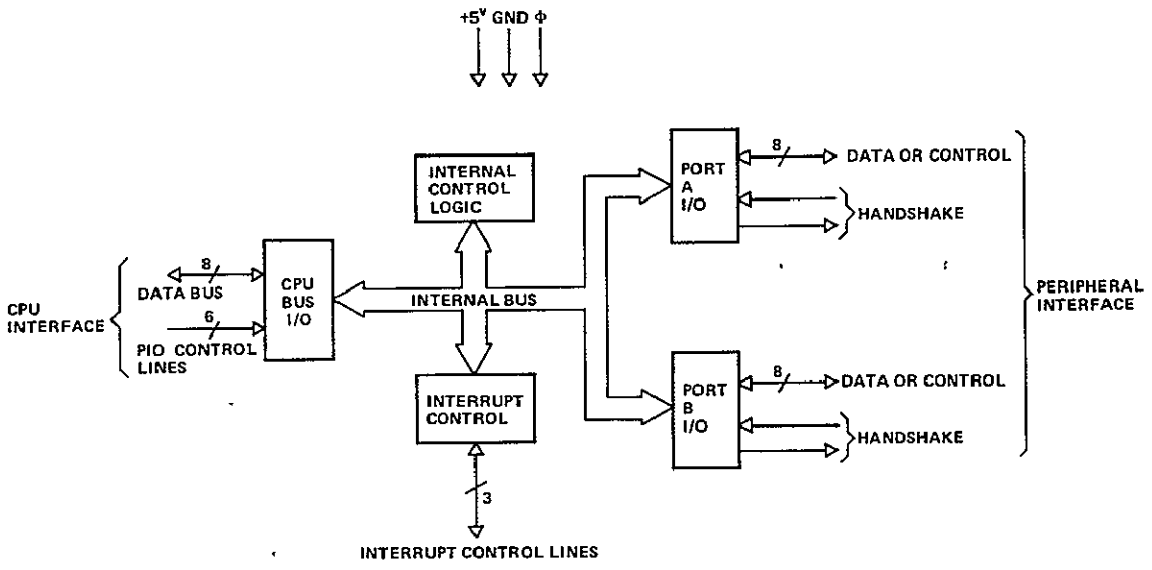


Figure 4.8 PIO block diagram (MOSTEK Z80 Manual).

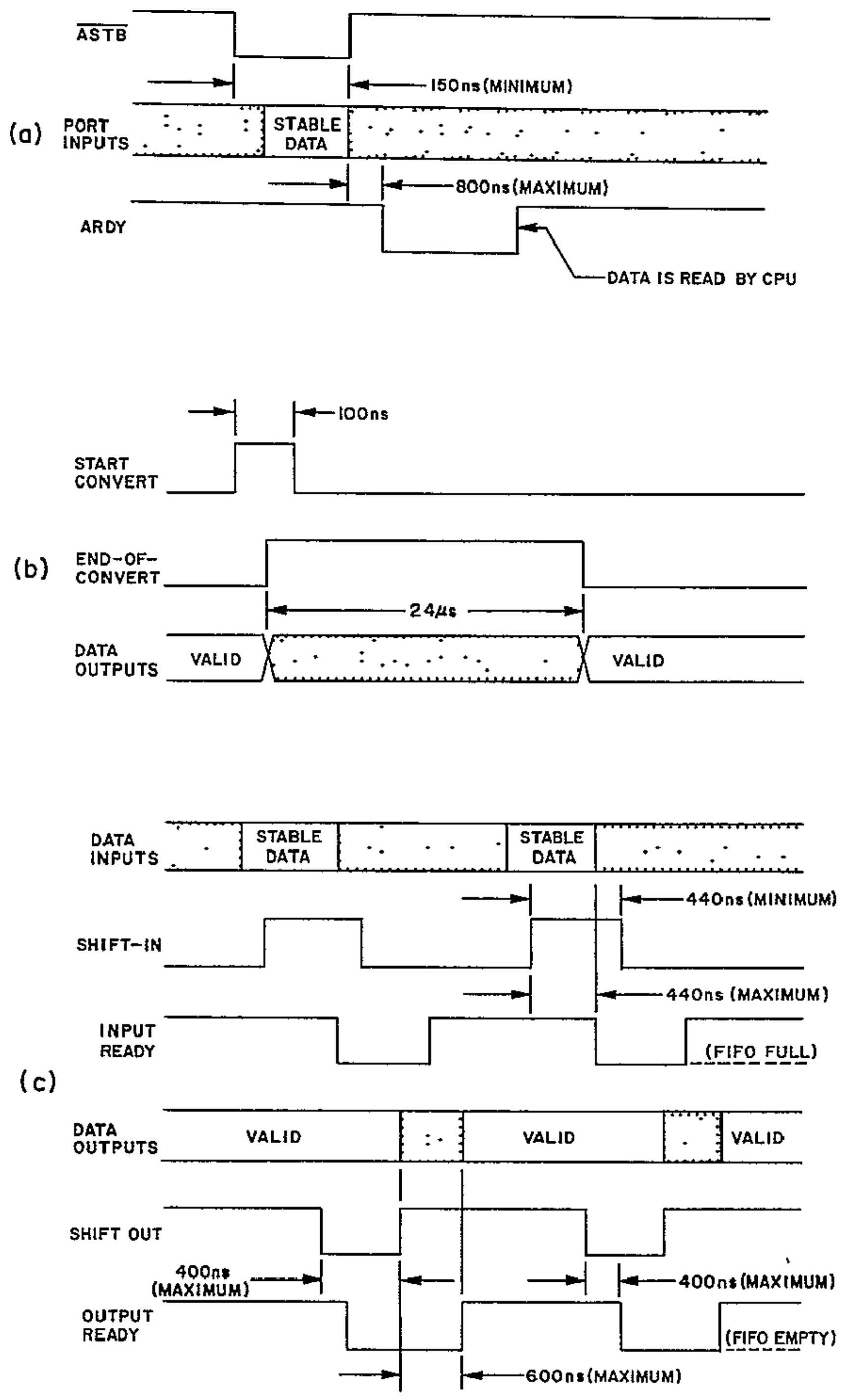
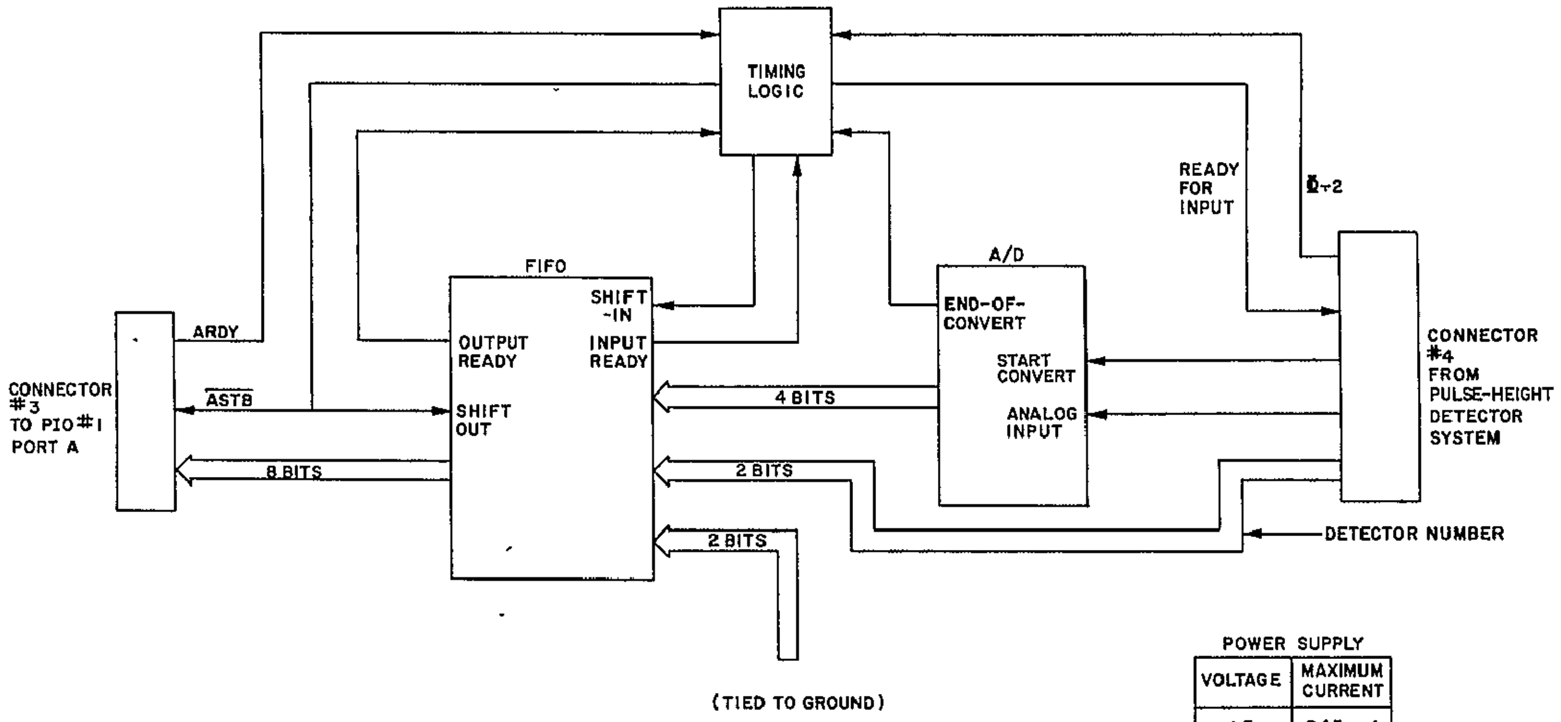


Figure 4.9 Timing diagrams: (a) PIO#1 Port A handshake line timing; (b) A/D control signal timing; (c) FIFO handshake line timing.



POWER SUPPLY

VOLTAGE	MAXIMUM CURRENT
+5	245 mA
+15	25 mA
-15	35 mA

Figure 4.10 The A/D-FIFO board. The arrows have the same meaning as in Figure 4.3.

shown in Figure 4.9(b). When the conversion is finished, the end-of-convert line goes low until another conversion is started.

The FIFO can be regarded as nine 40-bit serial shift registers. Data is input to the first register and trickles through until a location containing data or the 40th location is encountered. The FIFO has handshake signals similar to the PIO signals which control data input and output.

The timing diagram of the FIFO handshake signals is shown in Figure 4.9(c). The input-ready signal indicates when data may be entered into the first register. Data is entered by bringing the shift-in line high. This causes the input-ready line to go low but data will stay in the first register until the shift-in input goes low. The data then propagates to the second location providing it is empty, and the input-ready line goes high again. The input-ready line will stay low if the FIFO is full. Once data has been shifted to the second location, it automatically continues until other data or the last register is encountered as stated above. The output-ready line goes high when the data reaches the last location. Data is shifted-out when the shift-out line is brought low and returns high. Data is not shifted from the 39th to the 40th location until the shift-out line goes back high, therefore, the output data is stable until then. The output ready line goes back high when the data has been shifted unless the FIFO is empty.

The end-of-convert signal from the A/D is used to enter data into the FIFO. When the end-of-convert signal makes a high-to-low transition, it triggers a 74123 monostable multivibrator as shown in Figure 4.11. The multivibrator then generates a 475 ns pulse on the shift-in line of the FIFO.

The A/D end-of-convert signal and the FIFO input-ready line are combined to generate a status signal to indicate when the board is ready for an input, but it is not used by the PHA system. Since data is input at relatively long intervals of 50 μs the status signal is not needed. It may be noted that the ready-for-input signal, if used, would have to be used judiciously since it will momentarily go low up to 1.025 μs before another start-convert can be applied, as noted on the diagram. It momentarily goes low because the FIFO input-ready signal does not go low for up to 440 ns after the shift-in input is brought high.

Data is strobed into PIO #1 using: the output-ready signal from the FIFO; ARDY of the PIO; and the system clock divided by eight (i.e., 0.3125 MHz). These three signals are input to a NAND gate whose output goes to another

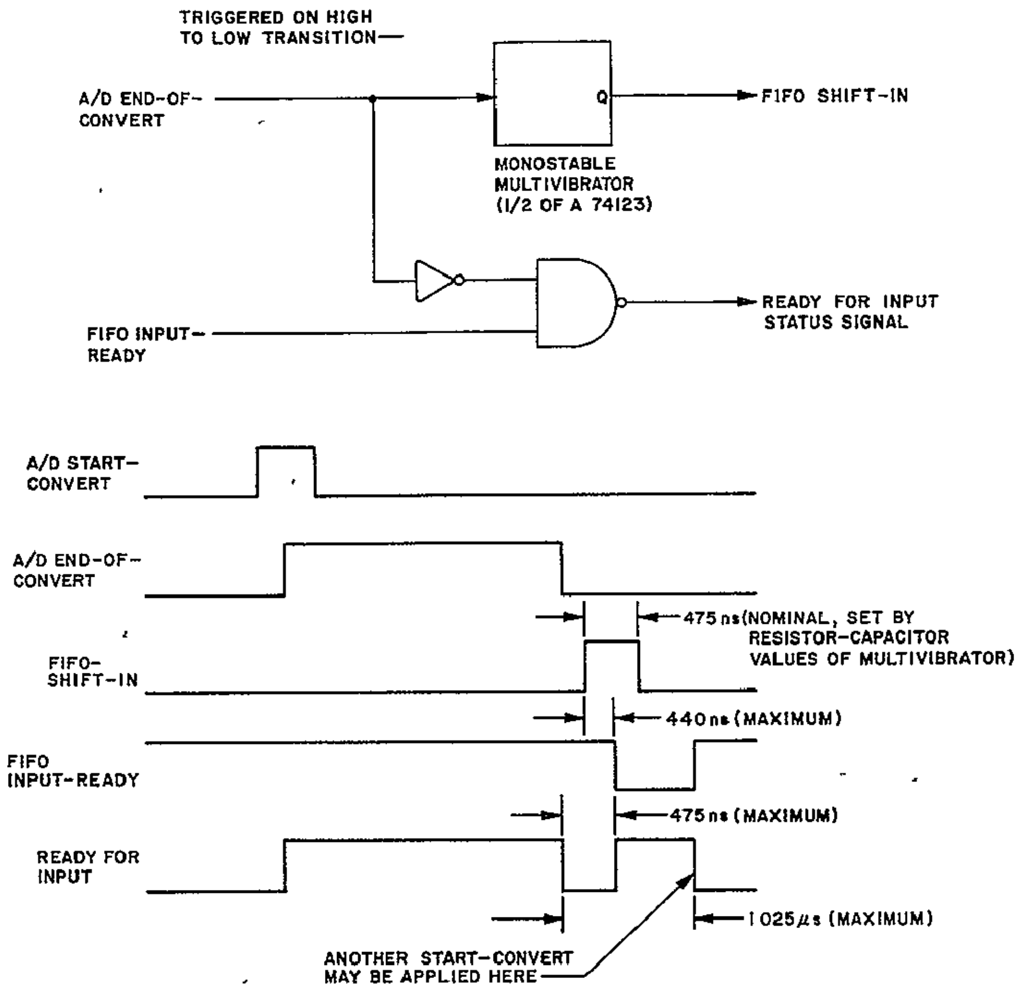

TRIGGERED ON HIGH TO LOW TRANSITION
A/D END-OF-CONVERT
Q
FIFO SHIFT-IN
MONOSTABLE MULTIVIBRATOR (1/2 OF A 74123)
READY FOR INPUT STATUS SIGNAL
FIFO INPUT-READY
A/D START-CONVERT
A/D END-OF-CONVERT
475 ns (NOMINAL, SET BY RESISTOR-CAPACITOR VALUES OF MULTIVIBRATOR)
FIFO-SHIFT-IN
440 ns (MAXIMUM)
FIFO INPUT-READY
475 ns (MAXIMUM)
READY FOR INPUT
1025 µs (MAXIMUM)
ANOTHER START-CONVERT MAY BE APPLIED HERE

Figure 4.11 The circuit and timing diagram for the input to the FIFO and the ready-for-input status signal.

monostable multivibrator as shown in Figure 4.2 (the multivibrator is the other half of the 74123 used above). When the divided clock signal makes a high-to-low transition while the other two NAND gate inputs are high, the monostable will be triggered and the $\overline{\text{ASTB}}$ line of the PIO will receive a 580 ns pulse. This signal is also used to drive the shift-out line of the FIFO. The divided clock signal is provided by the PHA system.

The divided clock signal is included because ARDY could possibly stay high longer than the time it takes the output-ready signal to go low and return high as shown in Figure 4.12. Because of this a method was needed to trigger the multivibrator after both signals have been high for a period of time. If only ARDY and output-ready were used, an extra pulse from the multivibrator could possibly occur causing a data sample to be lost. By using the high-to-low transition of the clock to trigger the multivibrator only one pulse will be generated by multivibrator.

On the systems constructed for Nike Apaches 14.542 and 14.543 $\Phi/2$ was used instead of $\Phi/8$ (i.e., Φ is the symbol for the clock signal). This could cause a problem if $\Phi/2$ goes low, high, then back low again after output ready has returned high and before ARDY goes low; or if ARDY goes low while both $\Phi/2$ and output-ready are high. The system using $\Phi/2$ appeared to work well during tests and the rocket flight but on future flights $\Phi/8$ should be used.

The same signal for the PIO $\overline{\text{ASTB}}$ line and the FIFO shift-out line can be used because the data hold time beyond the $\overline{\text{ASTB}}$ low-to-high transition required by the PIO is zero and a finite amount of time is required to shift data from the 39th to the 40th position in the FIFO after the shift-out transition. Since the data on the output pins of the FIFO will not change instantaneously when the transition occurs, the hold time of the FIFO is greater than zero.

In Figure 4.12 note that an extra trigger to the multivibrator could possibly occur during a pulse. This is permissible since it will only cause the pulse to be longer than 580 ns.

4.4 *PIO #2*

As shown in Figure 4.13 PIO #2 is on the board that contains the PHA system and the magnetometer signal digitizer. Four bits of Port A are used to input the magnetometer data, but Port B is not used.

Although only four of the sixteen PIO I/O bits are used, the PIO circuit provided the easiest and most efficient interface to the MSD since all the Z-80 control signals are handled by the PIO's internal logic. In addition the extra

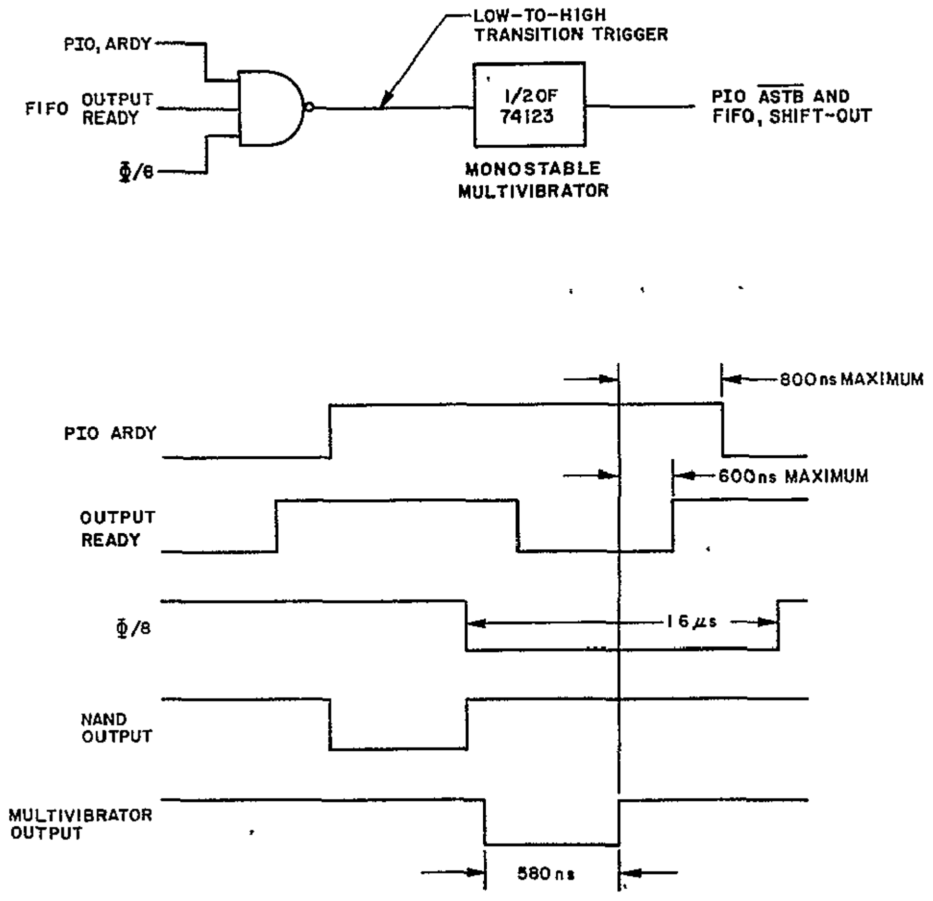


Figure 4.12 The circuit and timing diagram for input to PIO#1 from the FIFO.

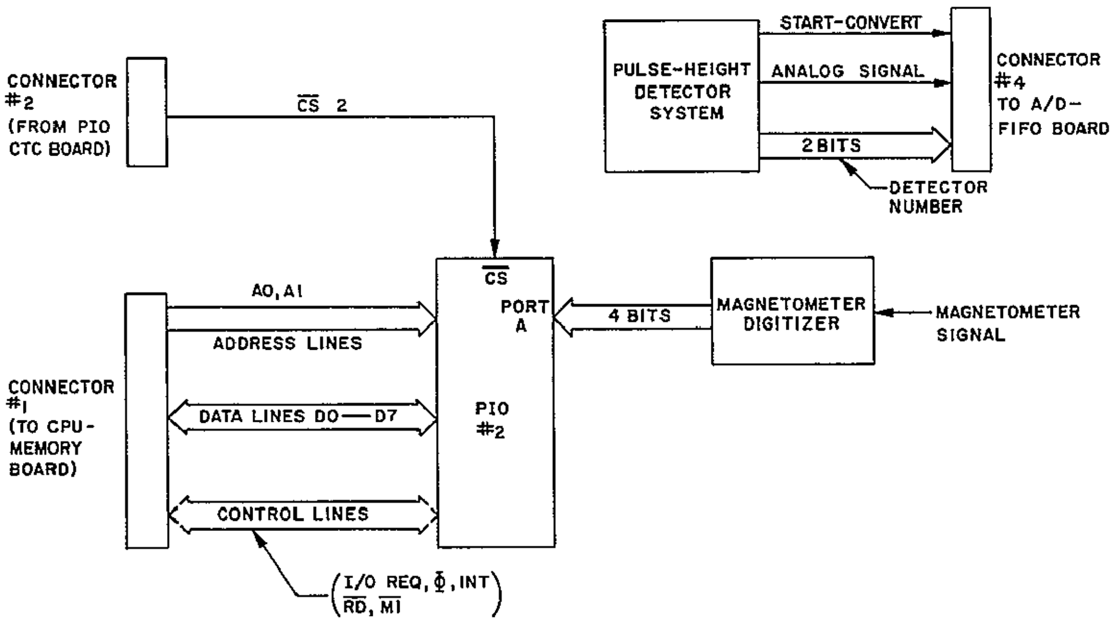


Figure 4.13 The PIO#2 board. The arrows have the same meaning as in Figure 4.3.

I/O bits could be used for a future experiment.

As stated before PIO #2 has addresses 8 through 11 with the $\overline{CS}$ provided by the decoder on the P-IO-CTC board, but only the addresses for Port A are needed (i.e., 8 and 9).

A detailed schematic of this board is given in *Leung et al.* [1979].

5. SOFTWARE

The software written for the data-manipulation experiment is discussed in this chapter. Since some understanding of the Z-80 architecture is needed, a brief description of it is included.

5.1 *Z-80 Architecture*

The Z-80 CPU register configuration is shown in Figure 5.1; the CPU contains two sets of independent accumulator, flag, and general purpose registers. A single exchange instruction enables the CPU to switch between the accumulator and flag pairs while another single instruction is used to switch between the general purpose register sets. This exchange capability is useful (it saves time) when an interrupt is serviced because the contents of the registers do not have to be saved in RAM.

The 8-bit accumulator register is used to hold the results of 8-bit arithmetic and logical operations while the flag register indicates the status of 8 or 16-bit operations, such as indicating whether or not the result of an operation is negative or positive. The 8-bit general purpose registers can be used in pairs for 16-bit operations, specifically the BC, DE and HL pairs in both sets. The HL pairs are used by several instructions for memory addresses.

The program counter holds the 16-bit address of the current top of the stack located in external RAM. The stack is a last-in-first-out memory file that is used to temporarily store data from the registers. The other special purpose registers are not used by the data manipulation experiment software.

The Z-80 CPU has both maskable and non-maskable interrupts. The CPU can be programmed to respond to a maskable interrupt in any one of three possible modes. Mode 1 is used in the present application since there are only two interrupting devices. In this mode a maskable interrupt from PIO #1 causes the CPU to jump to location 0038 H (where H means hexadecimal or base 16). A non-maskable interrupt from the CTC causes the CPU to jump to location 0066 H.

The Port I/O logic of one port of a PIO is composed of six registers with "handshake" control logic as shown in Figure 5.2(a). The 2-bit mode control register is loaded by the CPU to select the desired operating mode. The input register holds data input to the port and the output register holds data that is to be output. The handshake logic controls the transfer of data between the peripheral and the PIO.

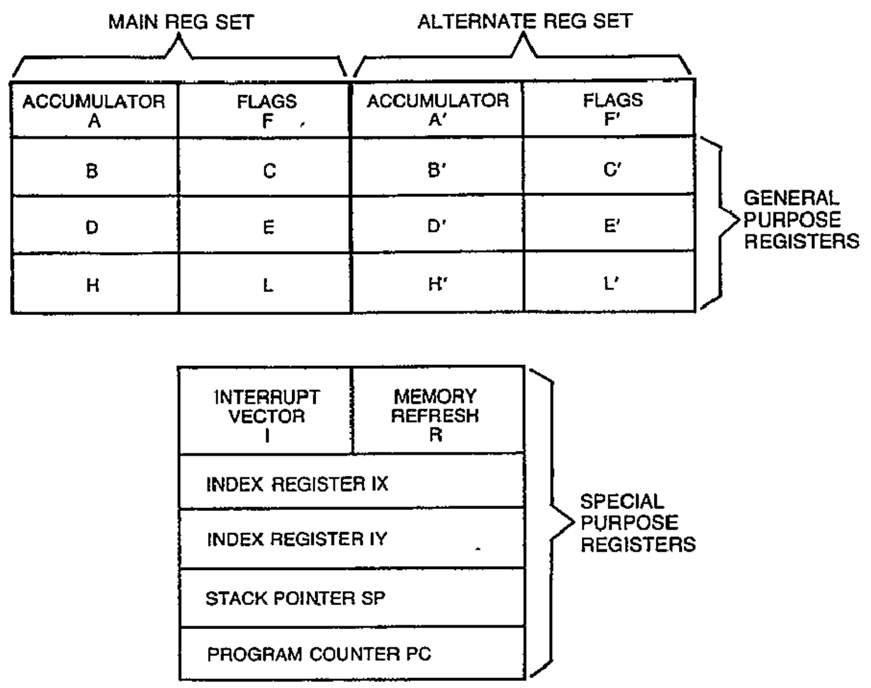
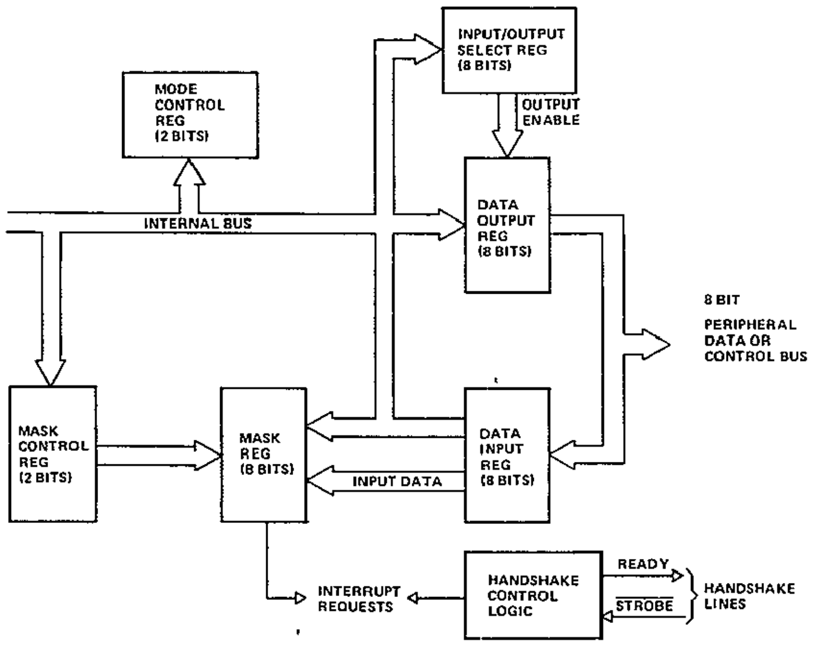


Figure 5.1 Z80-CPU register configuration (MOSTEK Z80 Micro-Reference Manual).

(a)



(b)

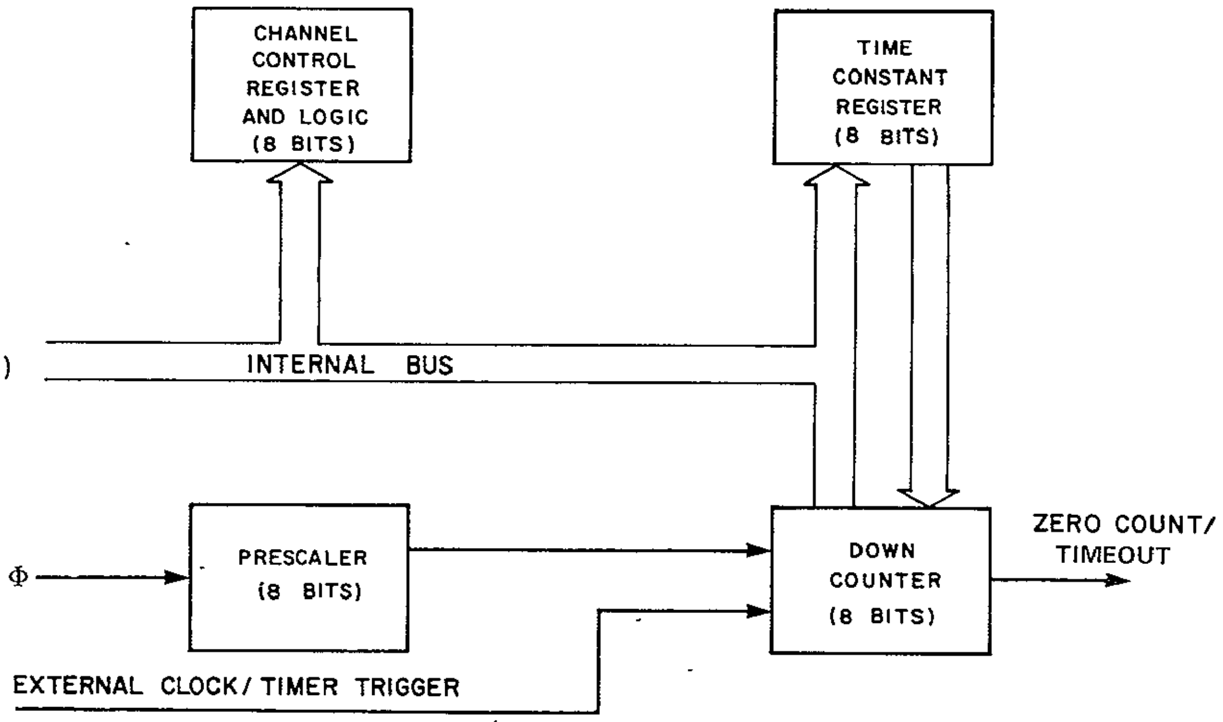

CHANNEL CONTROL REGISTER AND LOGIC (8 BITS)
TIME CONSTANT REGISTER (8 BITS)
INTERNAL BUS
Φ
PRESCALER (8 BITS)
DOWN COUNTER (8 BITS)
ZERO COUNT/ TIMEOUT
EXTERNAL CLOCK/ TIMER TRIGGER

Figure 5.2 Block diagrams of (a) one port of a PIO and (b) one channel of the CTC (MOSTEK Z80 Manual).

The 8-bit I/O select register and the 8-bit mask register are used when the port is programmed to operate in Mode 3. In this mode any of the data lines can be programmed to be an input or output as specified by the select register. The 2-bit mask control register and the mask register are used to control the generation of interrupts in Mode 3, but they are not used in the data manipulation software.

The logic for one channel of a CTC is composed of two registers, two counters and control logic as shown in Figure 5.2(b). The time constant register is loaded by the CPU to initialize and re-load the down counter at a count of zero. The channel control register is loaded by the CPU to select the mode and conditions of channel operation. The prescaler divides the system clock by 16 or 256 for decrementing the down counter. The external clock/timer trigger and the zero count/timeout signals are not used in the data manipulation experiment.

5.2 *Data-Manipulation Software*

The software for the system comprises three main parts: initialization, the input routine, and the output routine. During initialization the CTC and PIO's are programmed for operation, the RAM is cleared and other pre-operation details are taken care of. The input and output routines function as explained previously. These three sections of the software are discussed further below. A listing of the program is given in Appendix II.1.

5.2.1 *Initialization.* The flowchart of this section of the software is shown in Figure 5.3. After power-on the PIO's and the CTC are programmed as shown although the interrupt capability of PIO #1 is not enabled and the CTC time constant is not loaded until just before the "Halt" instruction. This is done to prevent an interrupt from occurring before the initialization is finished. HL' and DC' are loaded with 800 H and zero respectively for use in the output routine. Data from the magnetometer must be shifted right two positions and 800 H must be added as shown in Figure 5.4 before it can be used to designate the rotation sector memory area shown in Figure 3.2(d). The ARDY line of PIO #1 must be raised the first time by reading Port A although the data is not used. After the ARDY line is raised, the CPU executes a "Halt" instruction. When this is done, the CPU enters an inactive state that can only be exited when an interrupt is received. After the interrupt is serviced, the CPU executes the instruction directly following the Halt instruction which is a jump back to it.

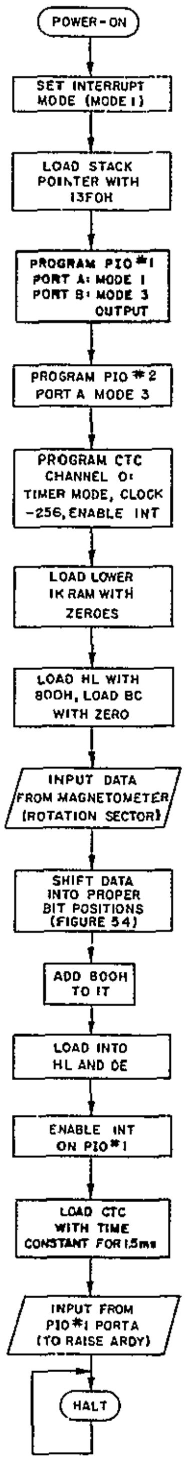

POWER-ON
SET INTERRUPT MODE (MODE 1)
LOAD STACK POINTER WITH 13F0H
PROGRAM PIO #1 PORT A: MODE 1 PORT B: MODE 3 OUTPUT
PROGRAM PIO #2 PORT A MODE 3
PROGRAM CTC CHANNEL 0: TIMER MODE, CLOCK -256, ENABLE INT
LOAD LOWER 1K RAM WITH ZEROES
LOAD HL WITH 800H, LOAD BC WITH ZERO
INPUT DATA FROM MAGNETOMETER (ROTATION SECTOR)
SHIFT DATA INTO PROPER BIT POSITIONS (FIGURE 5.4)
ADD 800H TO IT
LOAD INTO HL AND DE
ENABLE INT ON PIO #1
LOAD CTC WITH TIME CONSTANT FOR 1.5ms
INPUT FROM PIO #1 PORTA (TO RAISE ARDY)
HALT

Figure 5.3 Flow chart of the initialization section.

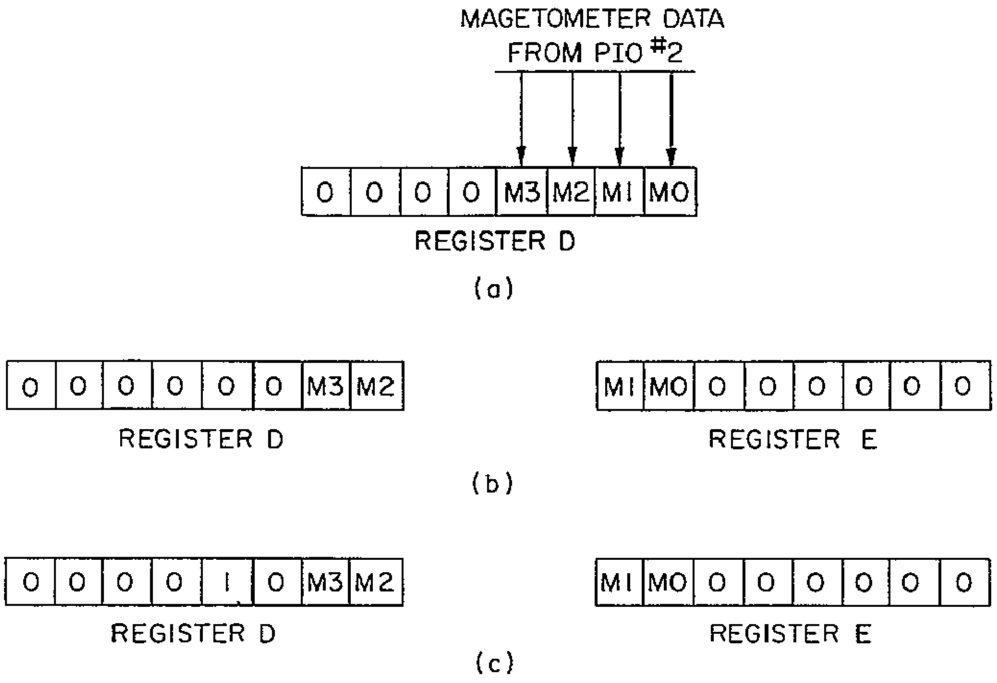


Figure 5.4 Magnetometer data bit positioning: (a) Data bits are put into register D; (b) shifted right two places into top two bits of E, (c) 800H is then added.

5.2.2 *Input routine*. A more detailed flowchart of the input routine than in Figure 3.3 is shown in Figure 5.5. Register pair DE is used to hold the current magnetometer data and is updated by the output routine. The data from the PHA system in A is concatenated with the magnetometer data in DE by simply adding register E to A. Register pair HL is then used to hold the concatenated data which point to the RAM location to be incremented.

The two lowest order bits of each RAM location are not used since they could be lost when the location is output due to noise (i.e., low order accumulations would be difficult to discriminate from zero). Therefore, four must be added to each location so that it is incremented beginning in the third bit position (i.e., $4_{10} = 100_2$).

When a maskable interrupt is serviced by the CPU, an internal flip-flop is automatically reset such that subsequent maskable interrupts will be ignored. This flip-flop must be set when the input routine is exited so that maskable interrupts are again enabled.

The input routine requires between 30 and 31.6 μs for execution (for a 2.50 MHz system clock) providing the CPU does not service a CTC interrupt. 24.8 μs are required for executing the instructions in the routine and from 5.2 to 6.8 μs are required by the CPU to exit the "Halt" state and prepare for servicing the interrupt (i.e., the contents of the program counter are stored in the stack file).

Since the input routine requires 31.6 μs and data is input to the FIFO every 51.2 μs, the FIFO is never allowed to store data past its maximum limit. In fact, no more than four inputs are input to the FIFO during the output routine since the maximum time required for the output routine is about 170 μs.

The time it takes for the system to catch up with the inputs waiting in the FIFO can be calculated as follows. Let T_1 equal the maximum time of the output routine, T_2 equal the time it takes the system to catch up, r_1 equal the input rate to the FIFO, and r_2 the output rate from the FIFO to the data manipulation system.

$$r_1(T_1) + r_1(T_2) \leq r_2(T_2) \qquad (5.1)$$

Now although it would appear that T_1 = 170 μs, the possibility of the CTC interrupt occurring during the input routine must be taken into account. If the CTC interrupt is received directly after the PIO interrupt, the input routine for that interrupt would have to be finished after the output routine

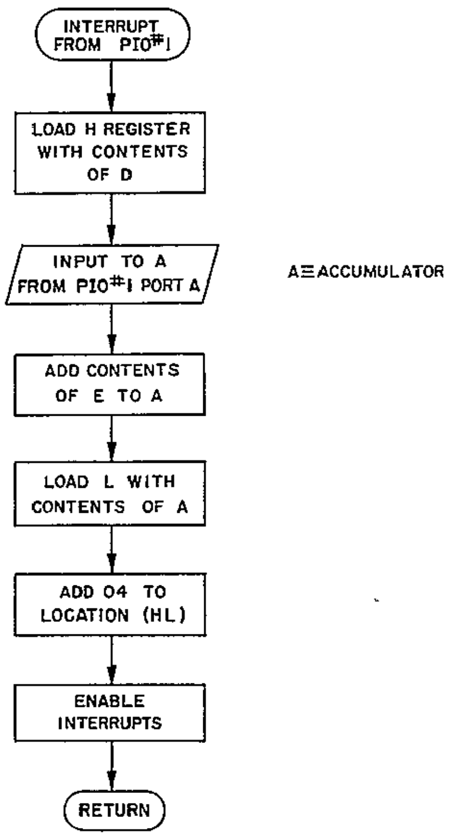


Figure 5.5 Flow chart of the input routine.

since the CTC has a higher priority. In this case the time to complete the input routine should be added on to the maximum time of the output routine. The input routine would, in the worst case, start at the "Load H register with contents of D" instruction as shown in Figure 5.5. In this case T_1 would be approximately equal to 195 μs (i.e., 170 μs + 24.8 μs).

Using equation (5.1) a value for T_2 can now be obtained as follows

$$\frac{1\ \text{input}}{51.2\ \mu\text{s}}\,(195\ \mu\text{s}) + \frac{1\ \text{input}}{51.2\ \mu\text{s}}\,(T_2) \leq \frac{1\ \text{output}}{31.6\ \mu\text{s}}\,(T_2) \tag{5.2}$$

Since it is impossible to get a partial input, we round the first term up to 4. Thus equation (5.2) becomes

$$4 + \frac{T_2}{51.2\ \mu\text{s}} \leq \frac{T_2}{31.6\ \mu\text{s}} \tag{5.3}$$

This gives a value of 330 μs for T_2, but since this is not an integer multiple of 31.6 μs, 347.6 μs should be used.

Since this is the worst case value for the catch-up time, there is no danger of the system getting behind (i.e., 0.3476 ms << 1.536 ms, the CTC interrupt interval). The output routine normally takes much less time than 170 μs, as will be shown in the next section. In most cases no more than two or three inputs would be waiting in the FIFO at the end of the output routine.

5.2.3 *Output routine*. The output routine is the most complex of the three software sections. As will be explained below, this results from a desire to provide a method for outputting data in the event of failure of the MSD and the need to accommodate a range of values of rocket spin rate.

A more detailed flowchart than the one in Figure 3.4 is shown in Figure 5.6. The register sets are exchanged after the magnetometer data in DE is updated. Data for the input routine is contained in one register set while data for the output routine is contained in the alternate set. Note that DE' also contains the updated magnetometer data for use later in the routine. Register C' is used as the output flag, as described in Chapter 3. A 01 H contained in C' corresponds to the flag being set while 00 corresponds to its being reset.

(a) *MSD failure mode*. The software includes a method to output data even if there is a failure of the MSD. In this mode the energy spectrums are obtained but there is no pitch-angle distribution.

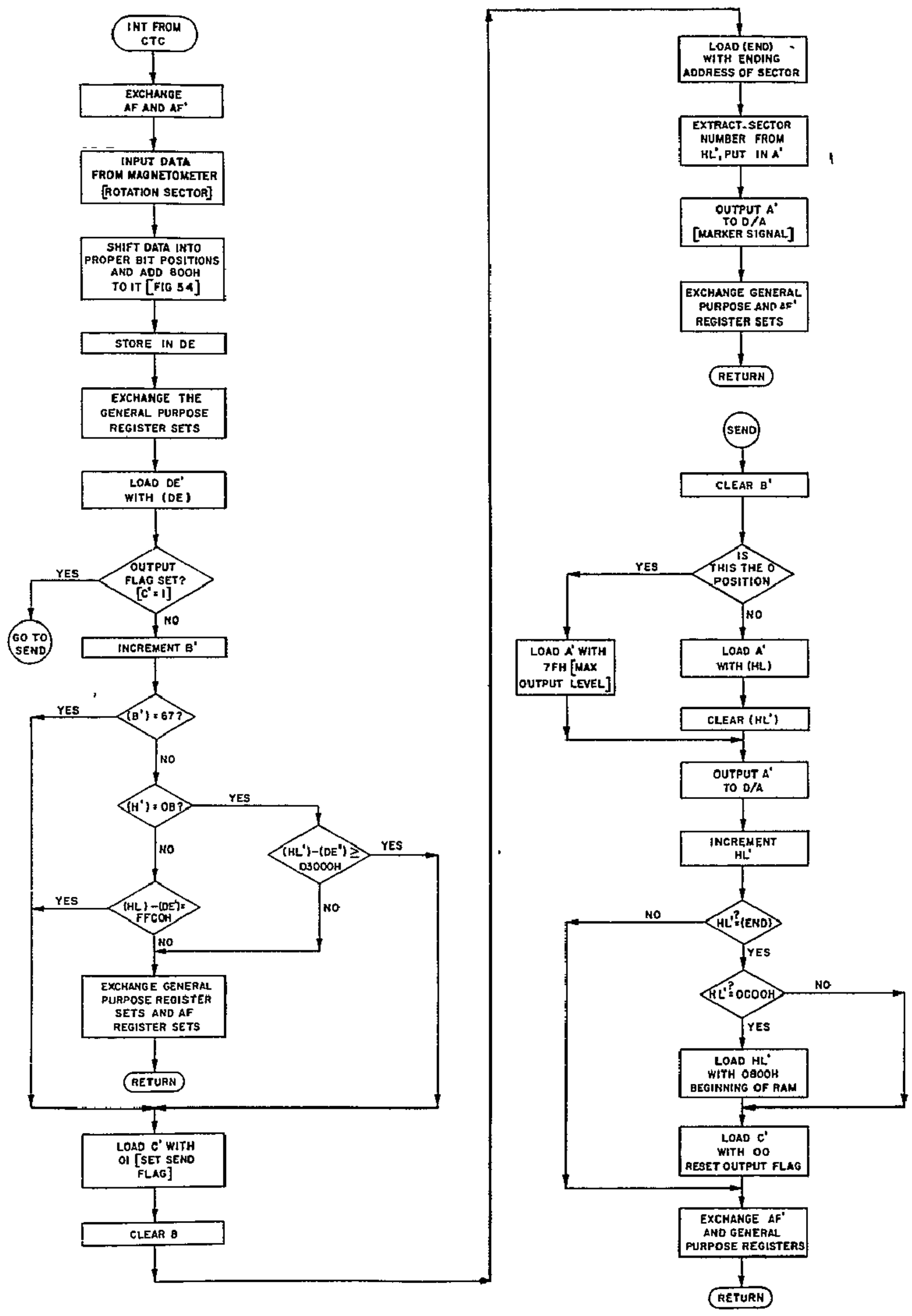

INT FROM CTC
EXCHANGE AF AND AF'
INPUT DATA FROM MAGNETOMETER [ROTATION SECTOR]
SHIFT DATA INTO PROPER BIT POSITIONS AND ADD 800H TO IT [FIG 5.4]
STORE IN DE
EXCHANGE THE GENERAL PURPOSE REGISTER SETS
LOAD DE' WITH (DE)
OUTPUT FLAG SET? [C'=1]
YES
GO TO SEND
NO
INCREMENT B'
(B')=67?
YES
NO
(H')=0B?
YES
(HL')-(DE')≥ 03000H
YES
NO
(HL)-(DE')= FFC0H
YES
NO
NO
EXCHANGE GENERAL PURPOSE REGISTER SETS AND AF REGISTER SETS
RETURN
LOAD C' WITH 01 [SET SEND FLAG]
CLEAR B
LOAD (END) WITH ENDING ADDRESS OF SECTOR
EXTRACT SECTOR NUMBER FROM HL', PUT IN A'
OUTPUT A' TO D/A [MARKER SIGNAL]
EXCHANGE GENERAL PURPOSE AND AF' REGISTER SETS
RETURN
SEND
CLEAR B'
IS THIS THE 0 POSITION
YES
LOAD A' WITH 7FH [MAX OUTPUT LEVEL]
NO
LOAD A' WITH (HL)
CLEAR (HL')
OUTPUT A' TO D/A
INCREMENT HL'
HL' ?= (END)
NO
YES
HL' ?= 0C00H
NO
YES
LOAD HL' WITH 0800H BEGINNING OF RAM
LOAD C' WITH 00 RESET OUTPUT FLAG
EXCHANGE AF' AND GENERAL PURPOSE REGISTERS
RETURN

Figure 5.6 Output section of the software .

Register B' is incremented every time there is a CTC interrupt (i.e., every 1.536 ms) and is reset to zero when data is output. If the output flag is not set by the MSD before B' is incremented to 67, it is set automatically.

The number of times B' is incremented before the output flag is set imposes a lower limit on the spin rate that can be accommodated. The last data output and the first time B' is incremented overlap such that B' would be incremented 67 times between outputs. Since there are 65 outputs (i.e., the 64 data bytes and the marker signal) and B' is incremented 67 times, the largest spin period that can be accommodated is 193 ms (i.e., (65+67) × 1.536 ms - 10 ms). Figure 5.7 illustrates the situation for a spin period of 193 ms (or longer). A spin period of 193 ms corresponds to a spin rate of 5.18 Hz which is considerably slower than the spin rates of past flights (which have been in the range of 6.4 to 7.5 Hz).

The choice of 67 in the above discussion is determined as a compromise between complicating requirements. The output rate in the event of an MSD failure is desired to be as fast as possible, which sets an upper limit on the number of times B' is incremented. It was arbitrarily decided that the interval between outputs should be no more than 10% longer than the time it takes to output a sector. This corresponds to 110 ms and an upper limit of 72 times (i.e., 110/1.536) to increment B'. A 5.5 Hz spin rate (182 ms period) was arbitrarily decided upon as the slowest spin rate anticipated. This gives a lower limit of 60 times (i.e., 182/1.536) to increment B'. Thus the interval created should be greater than 60 but less than 72; 67 was adopted.

(b) *Outputting sectors #12 to #15.* The last four sectors are output when the magnetometer signal digitizer indicates the rocket has rotated to sector #0 (i.e., the 4-bit counter output is zero). Figure 5.8 illustrates the reasons for doing this. When the spin rate is faster than 6.67 Hz (i.e., 150 ms period), data will not be input to one or more of the higher order sectors by the input routine. However, because the software is written assuming there would be sixteen sectors, all sixteen are always output even though the data in the highest order sectors may be zero. Therefore a method to set the output flag for the high order sectors was needed, other than the method used for the low order sectors explained in Chapter 3 (i.e., for the lower order sectors the output flag to output a particular sector is set when the rocket has rotated to the sector directly following it). By setting the output flag for the last four sectors when the 4-bit counter is reset to zero, a spin rate of up to

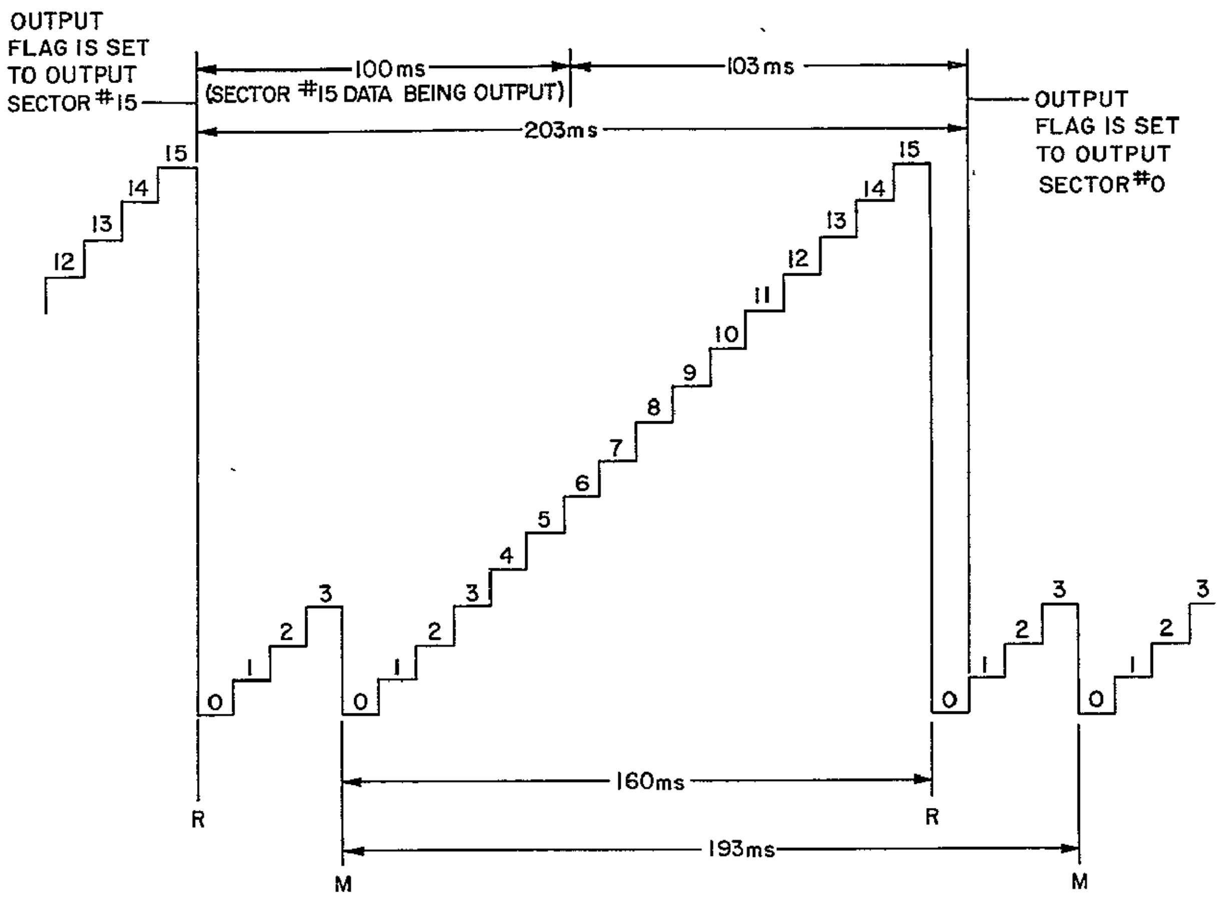

OUTPUT FLAG IS SET TO OUTPUT SECTOR #15
100 ms
(SECTOR #15 DATA BEING OUTPUT)
103 ms
203ms
OUTPUT FLAG IS SET TO OUTPUT SECTOR #0
12
13
14
15
0
1
2
3
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
0
1
2
3
0
1
2
3
160ms
193ms
R
R
M
M

Figure 5.7 Here and in the following figure the output of the MSD is represented as an analog signal to show the relationship between the rocket spin period and the counter cycle time. The case shown here is for a 5.18 Hz spin rate (i.e., 193 ms period). 10 ms must be added to the spin cycle time to get the interval between the times when the output flag is set between succeeding sections. Since the counter is incremented at 100 Hz, there are 20 increments between the times the counter is reset (the last count has a shorter interval, 3 ms). This diagram also represents the situation for a spin period greater than 193 ms and for no magnetometer signal. R indicates the internal reset of the 4-bit counter and M the reset of the counter by the magnetometer.

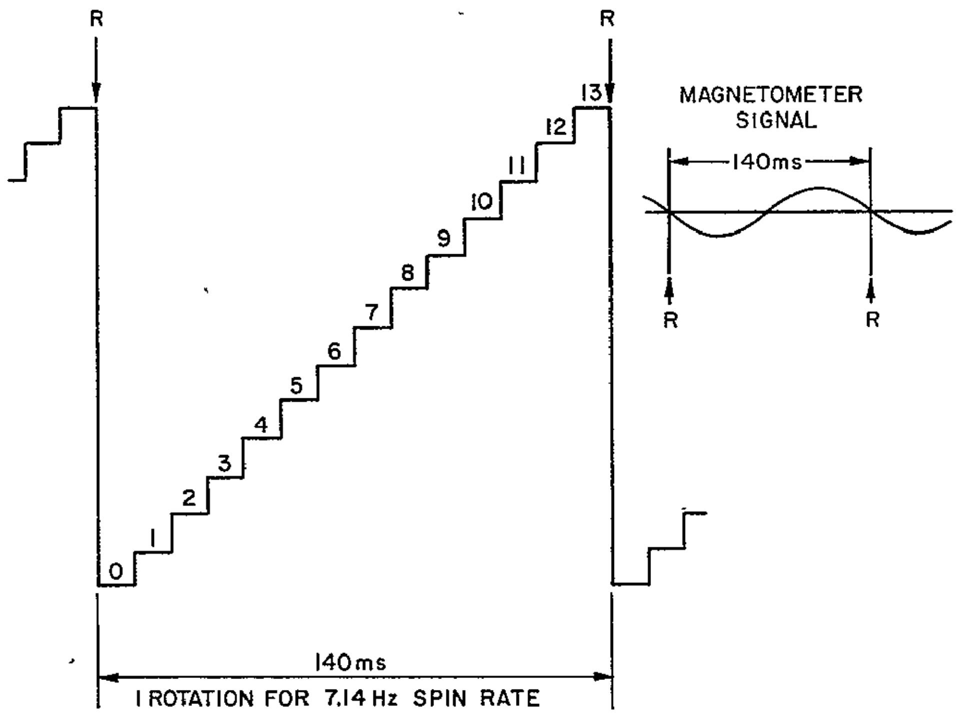


Figure 5.8 7.14 Hz spin rate causes the magnetometer signal digital representation to have 14 levels (refer to Figure 5.7). R indicates reset of 4-bit counter by the magnetometer.

8.33 Hz (i.e., 120 ms period: 12 sectors of input data) can be accommodated without causing the output to get out of synchronism with the rocket rotation.

Register pair HL' is used to contain the address of the next byte to be output, and the difference between the contents of HL' and DE' is used to set the output flag since DE' contains the updated magnetometer data. If the output flag is not set, but the difference between the contents of HL' and DE' is FFC0 H (i.e., the 2's complement of 0040 H), it will be set. For instance if the contents of DE' were 840 H and the contents of HL' were 800 H, the output flag would be set. Since H' contains 0B H when HL' points to a location in the last four sectors, the output flag can be set when the most significant byte of the difference between HL' and DE' equals 03. For example when HL' contains B40 H (i.e., corresponds to sector #13) and DE' contains 800 H (i.e., corresponds to sector #0) the difference is 340 H. By only considering the upper byte the software is simplified and takes less time.

(c) *Sampling time*. In the simplified description in Chapter 3 it was shown that the accumulation interval for each sector is the time it takes for 17 revolutions of the rocket. This will now be shown to be true for the actual output routine just described.

As shown in Figure 5.9 the interval between when the output flag is set for one sector and when it is set for the succeeding sector is 1 + 1/16 revolution for the first twelve sectors. The interval between when it is set for sector #11 and when it is set for sector #12 is 1 + 4/16 revolution since the flag is not set to output sector #12 until the rocket has rotated to sector #0. The following three sectors also start being output at sector #0 on succeeding revolutions so that the interval between setting the flag is 1 revolution for these sectors. Thus the sum of the sixteen intervals is seventeen revolutions (i.e., $12(1 + 1/16) + (1 + 4/16) + 4 \times 1 = 17$).

The sampling time for most sectors is a constant (170 ms) when the output is synchronized to the spin of the rocket, independent of the spin rate; the exceptions are the highest order sectors for spin rates greater than 6.25 Hz and the lowest order sectors for spin rates less than 6.25 Hz. After the rocket flight the spin rate is accurately known and the sampling time of the sectors can easily be calculated. For example, consider a spin rate of exactly 7 Hz. This is a spin period of 142.86 ms. Thus sectors #0 to #13 would have the full sampling time of 170 ms but sector #14 would have a sampling time of 48.6 ms (i.e., 17×2.86 ms) and sector #15 would have a sampling time of 0

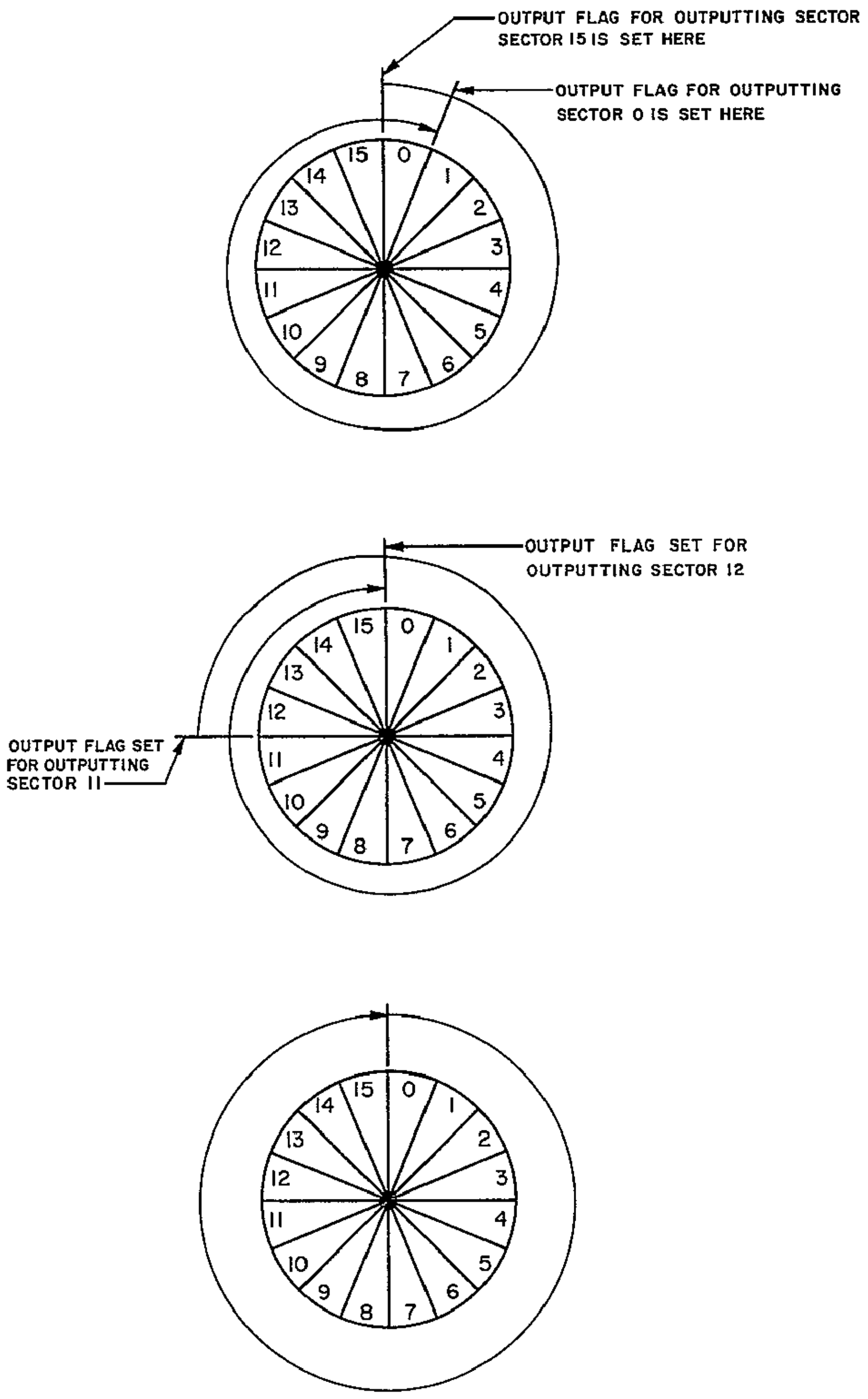


Figure 5.9 The interval between the output flag being set to output consecutive sectors depends on which sector it is being set for. (a) 1 + 1/16 revolutions occur between when the output flag is set to output consecutive sectors except for 12, 13, and 14. (b) The interval between the output flag being set for sectors 11 and 12 is 1 + 4/16 revolutions. (c) The interval between the output flag being set for sectors 12 and 13, sectors 13 and 14, and sectors 14 and 15 is exactly one revolution

(i.e., no data). In practice only the data from sectors #0 to #13 would be used.

Now consider the situation for a spin rate of 5.5 Hz. The spin period of 181.82 ms results in sectors #0 and #1 accumulating samples twice and, therefore, having sampling times of 340 ms. Sector #2 has a sampling time of 201 ms (i.e., 17 × (10 + 1.82) ms). Sectors #3 to #15 have a sampling time of 170 ms. Thus sectors #0 to #2 would contain more samples than the other 13.

As noted in Section 3.2 the input to the data-manipulation experiment is the output of the PHA. The multiplexer in the PHA has a timing sequence, involving the sampling and resetting, which reduces the count rate to 70% of the rate originating at the particle detectors (EPS and ESA) [*Leung et al.*, 1979]. This factor must be taken into account in deriving flux densities from the output data of the data-manipulation experiment.

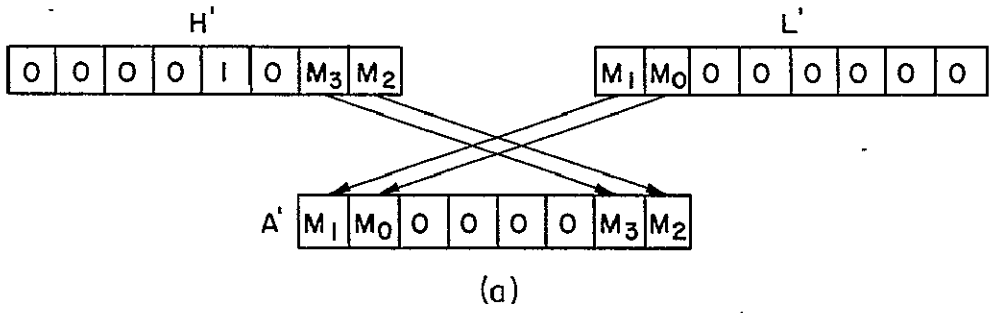
(d) *The marker signal*. The marker signal described in Chapter 3 is obtained by extracting the sector number from the HL' register pair and outputting it immediately after the output flag is set. The lower two bits from the H' register and the upper two bits from L' register are loaded into the A register. The data is then shifted in the A' register such that the most significant bit is in the sixth bit position as shown in Figure 5.10.

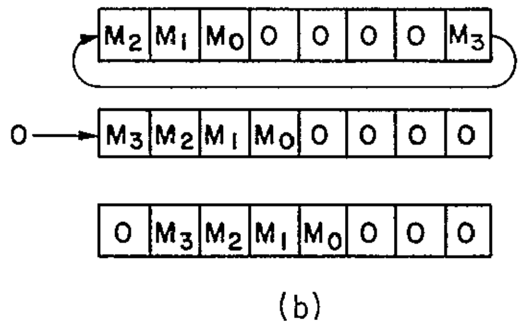
(e) *Other features*. Again referring to the flowchart in Figure 5.6 if the output flag is set when the output routine is entered, the program branches to "SEND". This section of the output routine outputs the sector data to the D/A and resets the output flag when all of a sector has been sent.

"END" is a RAM location that is used to contain the last address of a sector plus one. When HL' has been incremented enough times such that its contents equal the contents of the END and END+1 locations, the output flag is reset.

The high level outputs at zero energy locations that were discussed in Chapter 3 are obtained by outputting 7F H when the four low order bits of L' are all zeroes. The zero energy outputs will be slightly higher than the maximum output of the other fifteen energy levels since the two lowest order bits are ones whereas they are always zero for the other outputs.

Whenever the output routine is exited, the register sets are exchanged. As stated previously the DE register pair now contains the updated magnetometer data for use by the input routine.





M_2 M_1 M_0 0 0 0 0 M_3
0
M_3 M_2 M_1 M_0 0 0 0 0
0 M_3 M_2 M_1 M_0 0 0 0
(b)

Figure 5.10 The marker signal is obtained from HL' and is put in A' for output to D/A. (a) Sector number in HL' is put in A'. (b) Three shifts are then done to put data in proper bit positions for output to the D/A (i.e., two right circular shifts and one right shift with a zero put into the 7th bit).

The "return from a non-maskable interrupt" instruction automatically enables maskable interrupts. Therefore, the interrupt enable instruction does not have to be used (it was used in the input routine).

(f) *Output routine execution time.* The amount of time necessary to execute the output routine depends on the branches taken within it. There are three main possible paths that can be taken: (1) the output flag is set and a byte of data is output to the D/A, (2) the output flag is not set when the routine is entered but it is set during the routine, and (3) the output flag is not set when the routine is entered and it is not set during the routine. Path (1) requires from 99.6 μs to 123.2 μs depending on whether the output flag must be reset or not and whether it is the end of sector 15 or not. Path (2) requires either 130, 160.4 or 168.8 μs depending on whether (a) the flag is set because B' has reached 67, (b) it is set for one of the last four sectors, or (c) it is set for one of the first twelve sectors. Path (3) requires either 112.4 or 104 μs depending on whether the flag test is applied for the first twelve sectors or for the last four.

Path (2) requires the greatest amount of time, but it is the path most seldom taken since the output flag is only set once per revolution. Path (1) will be taken 64 times per revolution since 64 bytes of data are output. The number of times path (3) is taken depends on the spin rate of the rocket but it will be taken fewer times than path (1) unless the spin rate is slower than or equal to 5.3 Hz.

6. THE SDB-80 BOARD AND THE EPROM PROGRAMMER

Software was developed for the rocket system using a MOSTEK SDB-80 board, a teletype, and an EPROM programmer as shown in Figure 6.1. Some of the features of the SDB-80 and the design and operation of the EPROM programmer are discussed in this chapter.

6.1 *The SDB-80 Board*

The SDB-80 is a complete stand-alone microcomputer designed around the Z-80 microprocessor. It contains 10 K bytes of on-board firmware for software development consisting of an assembler/editor and an operating system. Programs can be edited and assembled directly from 16 K bytes of on-board RAM; this is much quicker than using the teletype tape reader and punch for the source and object versions. Two PIO's and a serial port are provided to interface the SDB-80 to peripheral devices. A photograph of the board is shown in Figure 6.2.

The operating system, the DDT-80, has several commands that allow a user to control the operation of the SDB-80. The DDT-80 also provides the software routines for interfacing various peripherals, such as the teletype, to the board. The commands of the DDT-80 are useful when debugging a program on the SDB-80. For example the user can read and modify registers in the CPU, start the execution of a program at a particular address, or dump the contents of a memory segment to a paper tape punch. A particularly useful command allows the user to set a breakpoint at a particular address within a program (i.e., when program execution reaches this location, the contents of the CPU registers are printed out).

The text editor on the SDB-80 is used to generate and modify Z-80 assembly language source programs. There are several editing commands for both line and character editing. Examples of these are a delete-line command and a command that changes a string of characters to another string.

Once the source program is generated using the editor, it is assembled to produce Z-80 object code. The assembler on the SDB-80 is a two-pass assembler, and it supports conditional assemblies, global symbols, relocatable programs and a printed-symbol table.

The SDB-80 was used to generate, run and debug the initial software for the rocket system. The A/D, the D/A, and a simulated magnetometer signal digitizer were interfaced to the SDB-80 using the two on-board PIO's. A CTC on another board was used to generate the non-maskable interrupts since the on-board CTC generated maskable interrupts and had a lower priority than the

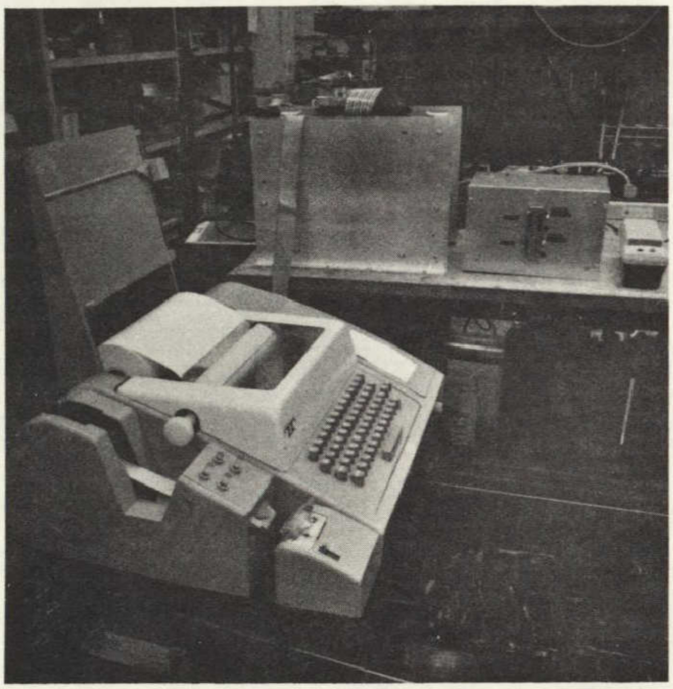


Figure 6.1 The EPROM programmer and the teletype connected to the MOSTEK SDB-80 board. This system was used to develop the data-manipulation system software.

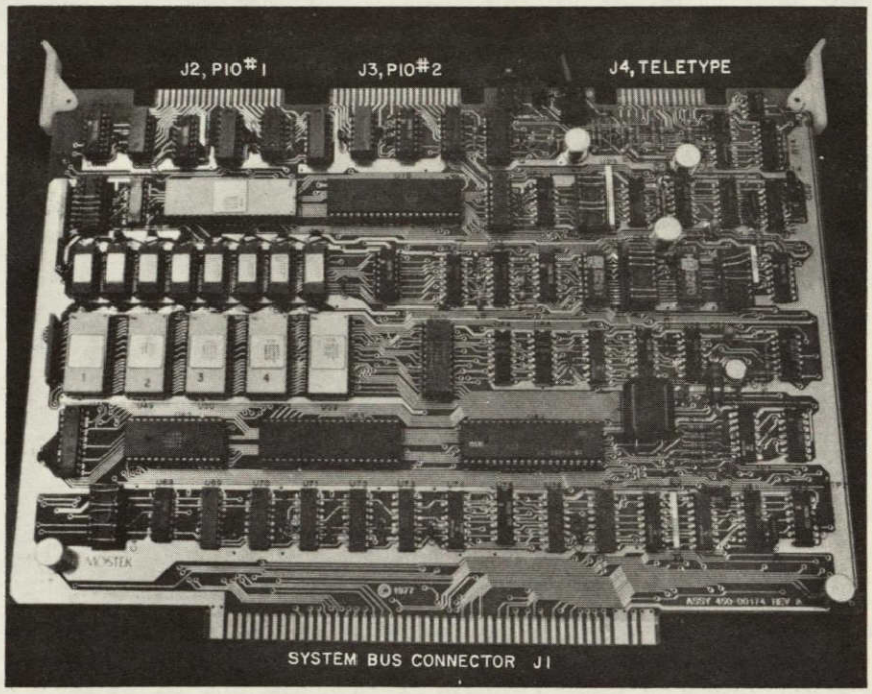


Figure 6.2 MOSTEK SDB-80 Board

PIO's. Once a version of the program was running well on the SDB-80, it was transferred, with some minor modifications, to a 2716 EPROM on the actual data manipulation circuit boards for further debugging. Using the SDB-80 for the initial versions of the software greatly reduced the debugging time.

6.2 *EPROM Programmer*

An EPROM programmer was designed and built to program 2708 and 2716 EPROM's. The EPROM programmer is interfaced to the SDB-80 by PIO #2 (connector J3, Figure 6.2). Port A is used for the 8-bit word to be put on the EPROM, and Port B is used for the programmer control signals.

6.2.1 *Programming waveforms*. The 2716 EPROM is programmed by applying a 50 ms TTL-level pulse to pin 18 (the PD/PGM input) when there is 25 V on pin 21. Figure 6.3(a) shows the 2716 program and verify waveforms. Before the pulse is applied, a data word and its corresponding address must be placed on the data and address lines of the 2716 and held for the duration of the pulse. The $\overline{\text{CS}}$ line must also be held high during the program pulse, but the programmed data may be checked immediately after the 50 ms pulse by lowering the $\overline{\text{CS}}$ line.

To program the 2708 it must first be put in the program mode by bringing the $\overline{\text{CS}}$ line to +12 V and holding it there for the duration of the programming. A +26 V 1 ms pulse is applied to the PGM pin for each data word corresponding to an address generated by the 4040 counter. Programming all 1024 locations sequentially with a 1 ms pulse constitutes one loop and one hundred loops are required to program the 2708 (manufacturer's specifications). The program waveforms for the 2708 are shown in Figure 6.3(b).

6.2.2 *Programmer hardware*. The amount of hardware involved in constructing the programmer was kept to a minimum by implementing all the timing functions with software. A block diagram of the programmer is shown in Figure 6.4 (a detailed schematic is included in Appendix I.4).

The 4040, a CMOS binary counter circuit, is used to generate the addresses for the 2708 or the 2716 when one is being programmed or read. Since the data locations are programmed sequentially, the 4040 is reset and incremented to generate the corresponding addresses. A data word corresponding to the address generated by the 4040 is placed on Port A of PIO #2. Then the PGM line of the 2708 or the PD/PGM line of the 2716 is raised for 1 ms or 50 ms respectively as described in the previous section. The 4040 is then incremented and the next location is programmed.

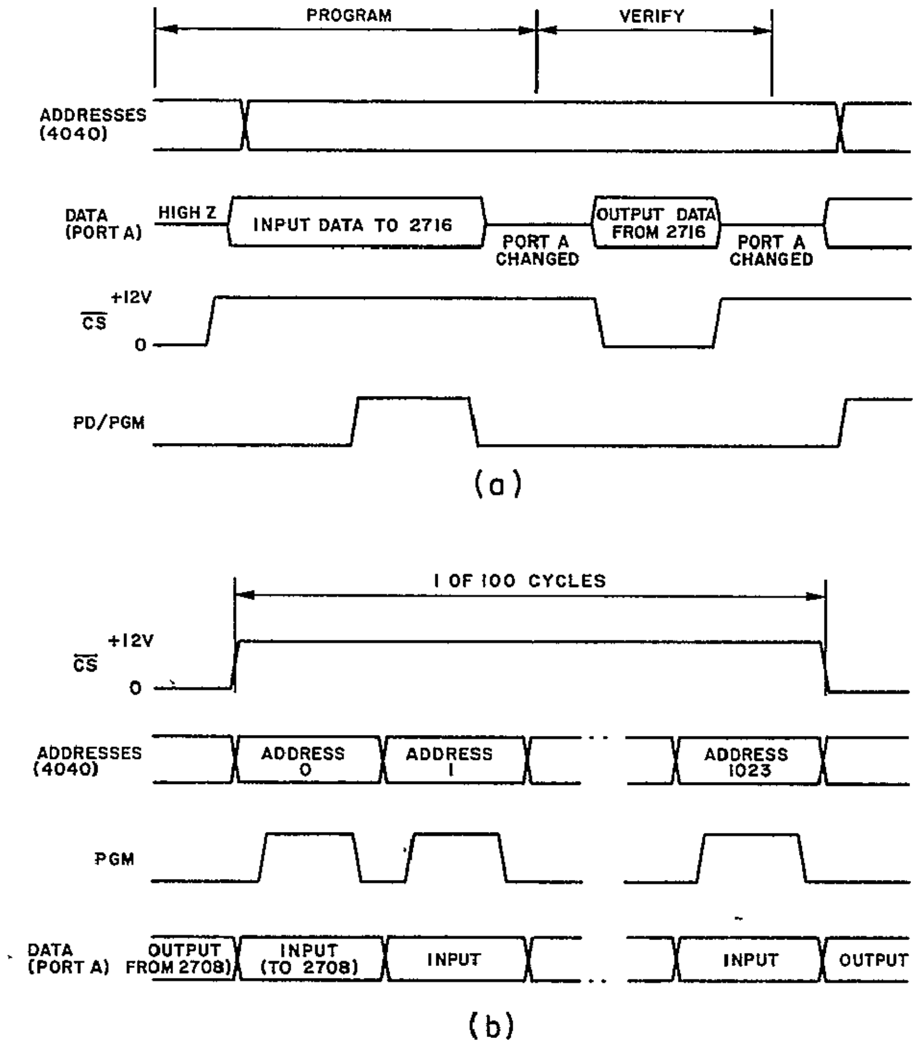


Figure 6.3 Programming waveforms. (a) 2716 program and verify waveforms. (b) 2708 program waveforms.

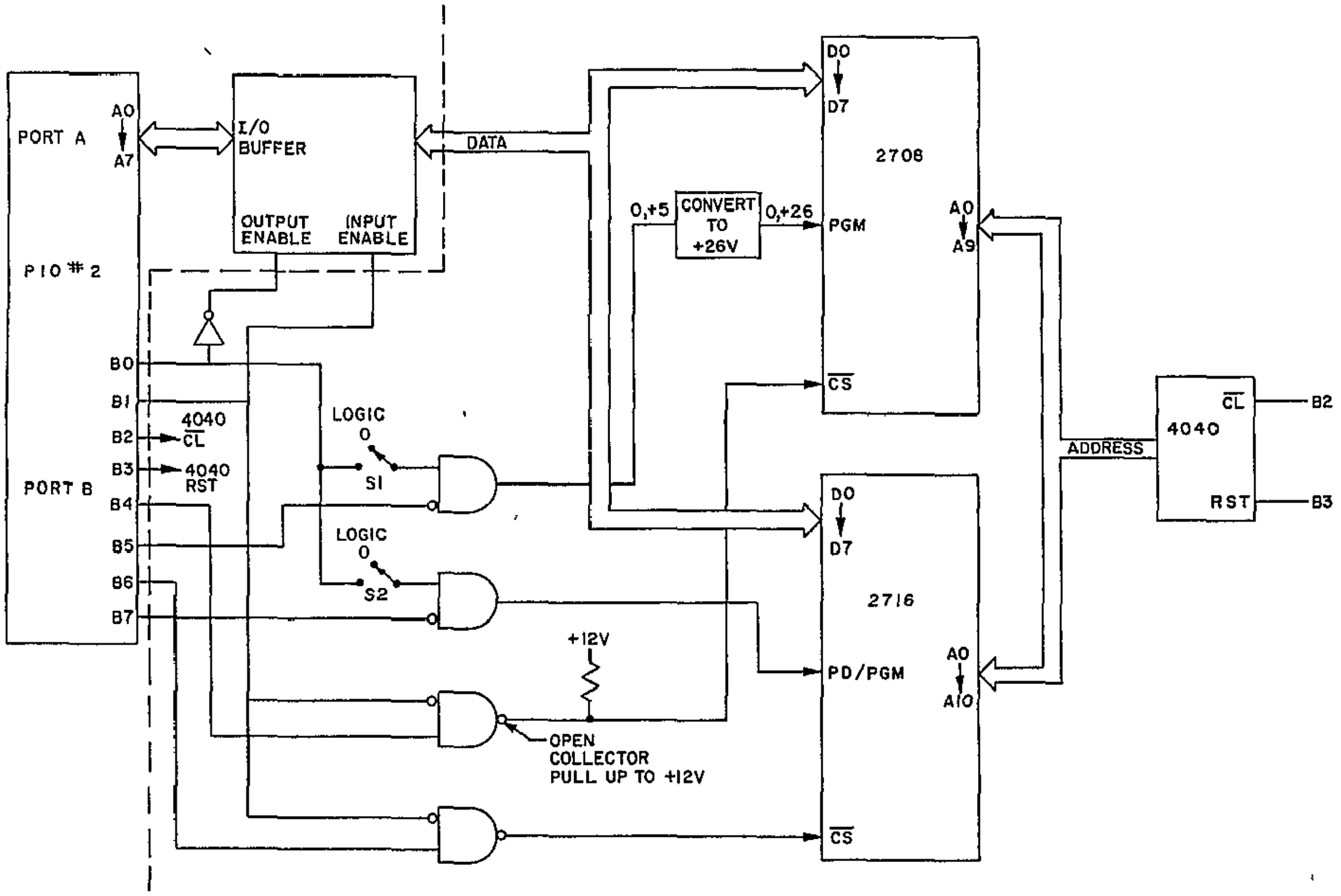


Figure 6.4 2708 and 2716 EPROM programmer. Everything to the left side of the dotted line is on the SDB-80 board.

The AND gates are used to prevent the program pulse lines from going high or the $\overline{CS}$ lines from going low accidentally. Before the programmer software is executed, the output buffers for Port B of PIO #2 on the SDB-80 are high because the Port B lines are in a high impedance state and the Port B buffers are non-inverting TTL gates (i.e., the TTL-gate inputs float high; the buffers are not shown in Figure 6.3). When Port B is initialized, its output lines go low. Since both high and low states are encountered, an exclusive or type logic had to be used to avoid damaging the EPROM or incorrectly programming a location. By inverting one input to an AND gate and not inverting the other only one input state will cause the output to go high, which solves the problem.

The following Boolean equations show the $\overline{CS}$ lines and the program pulse lines of the 2708 and 2716 as functions of the Port B lines. For the 2716 they are:

$$\overline{CS} = \overline{\overline{B1} \cdot B6} = B1 + \overline{B6} \qquad (6.1)$$

$$PD/PGM = B0 \cdot \overline{B7} \qquad (6.2)$$

For the 2708 they are:

$$\overline{CS} = \overline{B4 \cdot \overline{B1}} = B1 + \overline{B4} \qquad (6.3)$$

$$PGM = B0 \cdot \overline{B5} \qquad (6.4)$$

B0 and B1 control the input and output buffers for Port A of the SDB-80 PIO. B0 must be high to enable the output buffer and B1 must be low to enable the input buffer. As shown by the above equations the $\overline{CS}$ lines cannot go low unless the input buffer is enabled, and the program pulse line cannot go high unless the output buffer is enabled.

As shown in Figure 6.4 there is a switch on one of the inputs to each of the program pulse AND gates. They prevent an EPROM from accidentally being programmed by some possible malfunction of the hardware or software on the SDB-80. Before Port B is initialized by the programmer software, the switches should not be switched with an EPROM in one of the sockets since there would be a possibility of a spike being generated and reaching the program pulse inputs. This could result in an incorrectly programmed location or damage to the EPROM.

The timing diagram for the Port B lines is shown in Figure 6.5 B2 is used to increment the 4040 and B3 is used to reset it to zero. The other lines function as explained previously.

6.2.3 *SDB-80 modifications*. The SDB-80 was modified to allow a user to easily operate the programmer and to generate the 50 ms program pulses for the 2716. An off-board 2716 EPROM was added to contain the programmer software

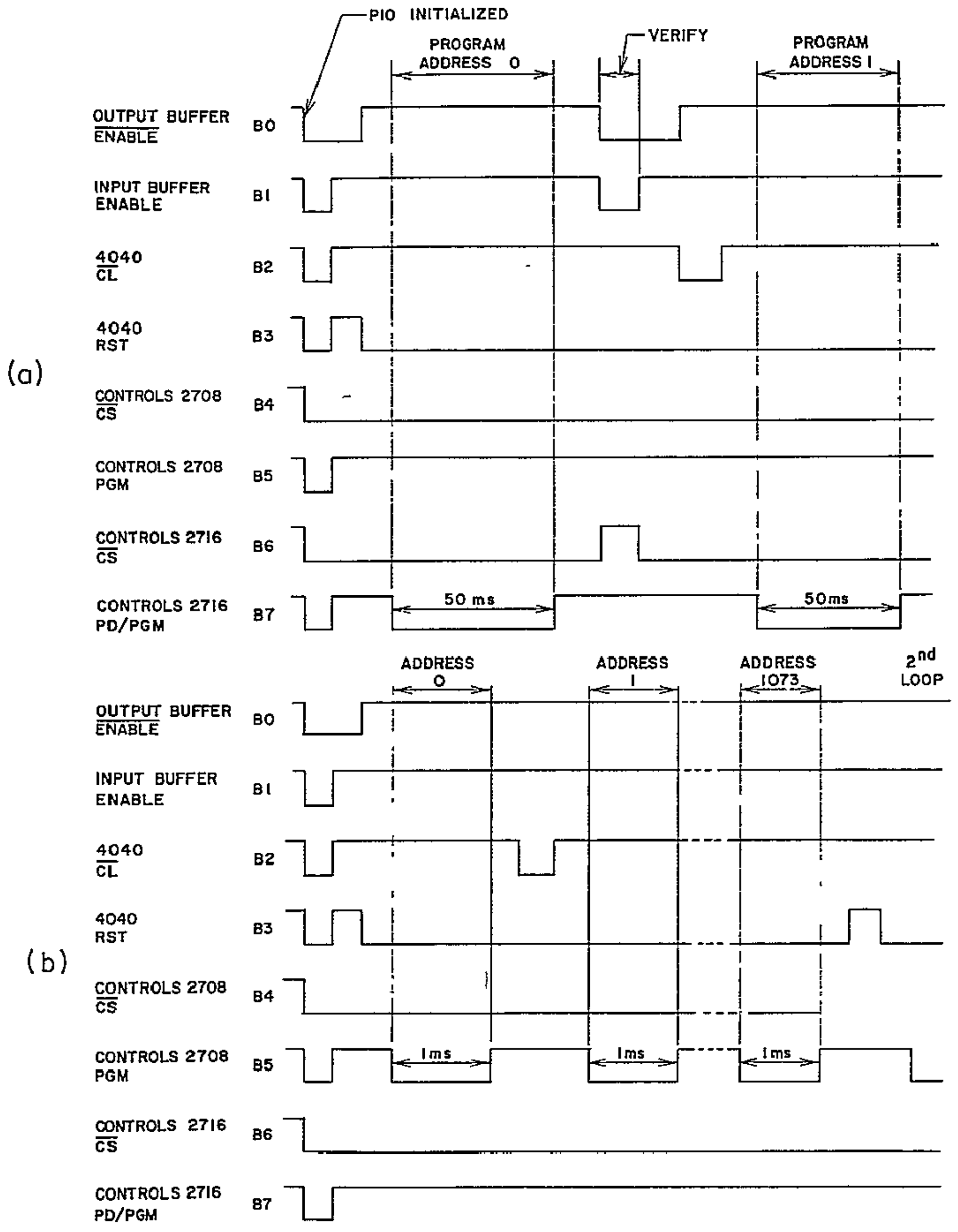


Figure 6.5 Port B timing diagram. (a) 2716 program and verify. (b) 2708 program.

and the on-board CTC had two lines tied together.

As shown in Figure 6.6 two one-of-eight decoders (74LS138) are used to decode the memory address for the off-board 2716. The starting address for the 2716 is 4000 H. This places it directly above the 16 K bytes of the SDB-80 on-board RAM. The EPROM is interfaced to the SDB-80 through connector J1 (the system bus connector, shown in Figure 6.2).

Note that the $\overline{CS}$ signal for the 2716 also goes to the $\overline{DINB}$ signal of connector J1. This enables a data bus input buffer on the SDB-80 when the $\overline{CS}$ line goes low.

The zero count output of channel #2 of the SDB-80 on-board CTC is tied to the timer trigger input of channel #3 (pin 8 to pin 6 of connector J1). This allows the counter of channel #3 to start being decremented when the counter of channel #2 reaches zero. This is done to time the 50 ms program pulse necessary for programming 2716 EPROM's because a single channel of the CTC cannot time that long of interval.

6.2.4 *EPROM programmer software.* The programmer software in the off-board 2716 controls the sequencing required to program the 2708 and 2716 EPROM's. The user is queried to select which operations are performed by the firmware (i.e., he chooses either the 2708 or 2716 to be either read or programmed). A flowchart in Figure 6.7 shows how this is done. The program is listed in Appendix II.2.

The SDB-80 PIO #2 is programmed to be operated in Mode 3 for both port A and port B. Port A is initially defined to be an input port, and port B is defined to be an output port.

After port B is programmed, the 4040 reset line is raised and the I/O buffer to port A is disabled. A message is then output to the user to ask whether a 2708 or a 2716 operation is to be performed. After the user defines which one, he is asked to specify a read or write operation.

The read routines for both the 2708 and 2716 are very similar. The only differences are that the $\overline{CS}$ lines controlled by port B are different and that the 2708 has 1024 locations while the 2716 has 2048 locations. To perform a read operation the corresponding $\overline{CS}$ line is lowered and the input buffer is enabled. The data byte is then input and stored at the location specified by the HL registers. HL and the 4040 are then incremented. If the contents of HL then equal 1024 (for 2708) or 2048 (for the 2716), the routine is exited, otherwise it continues as above.

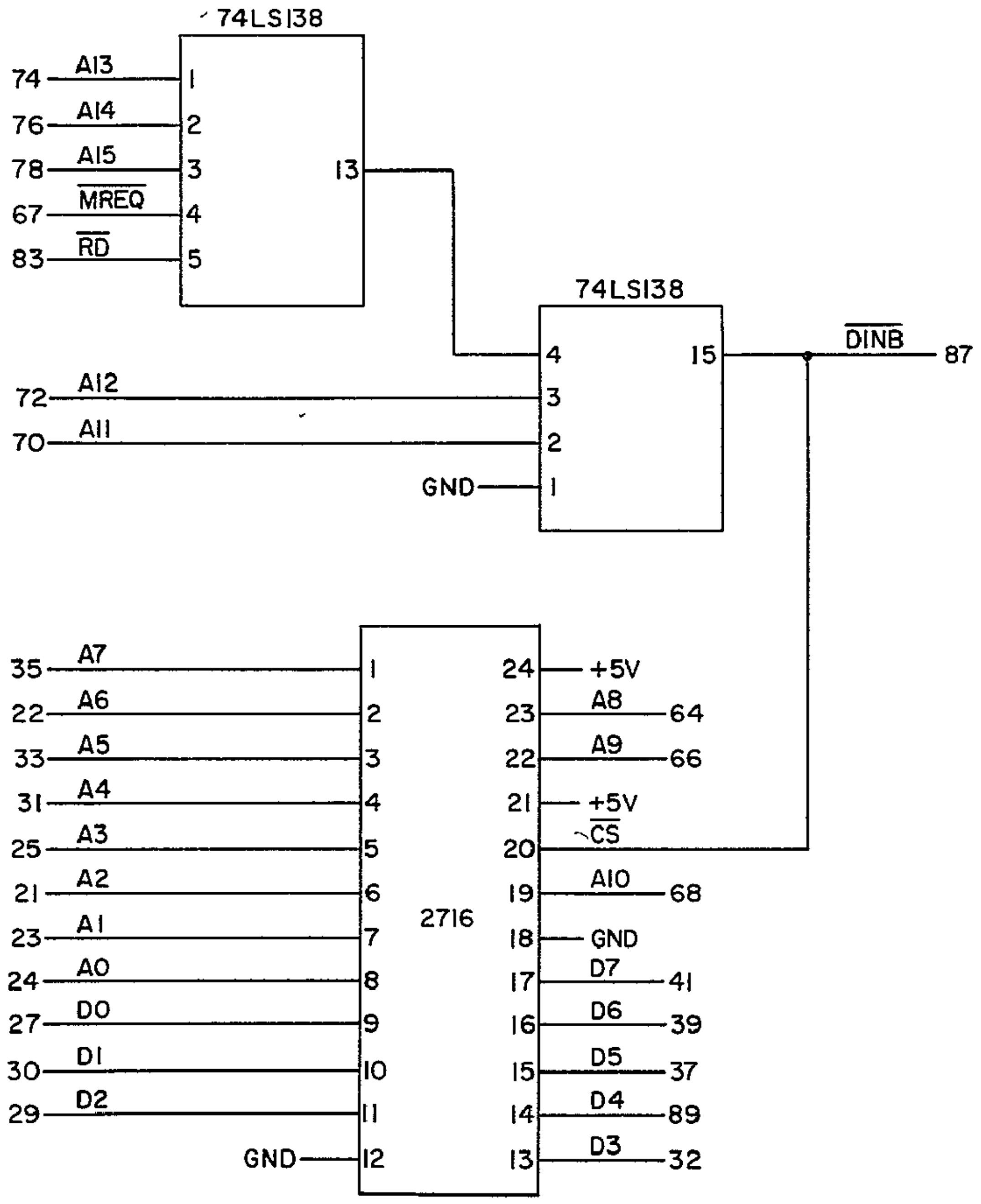


Figure 6.6 Connection of the 2716 EPROM that contains the programmer software.

(a)

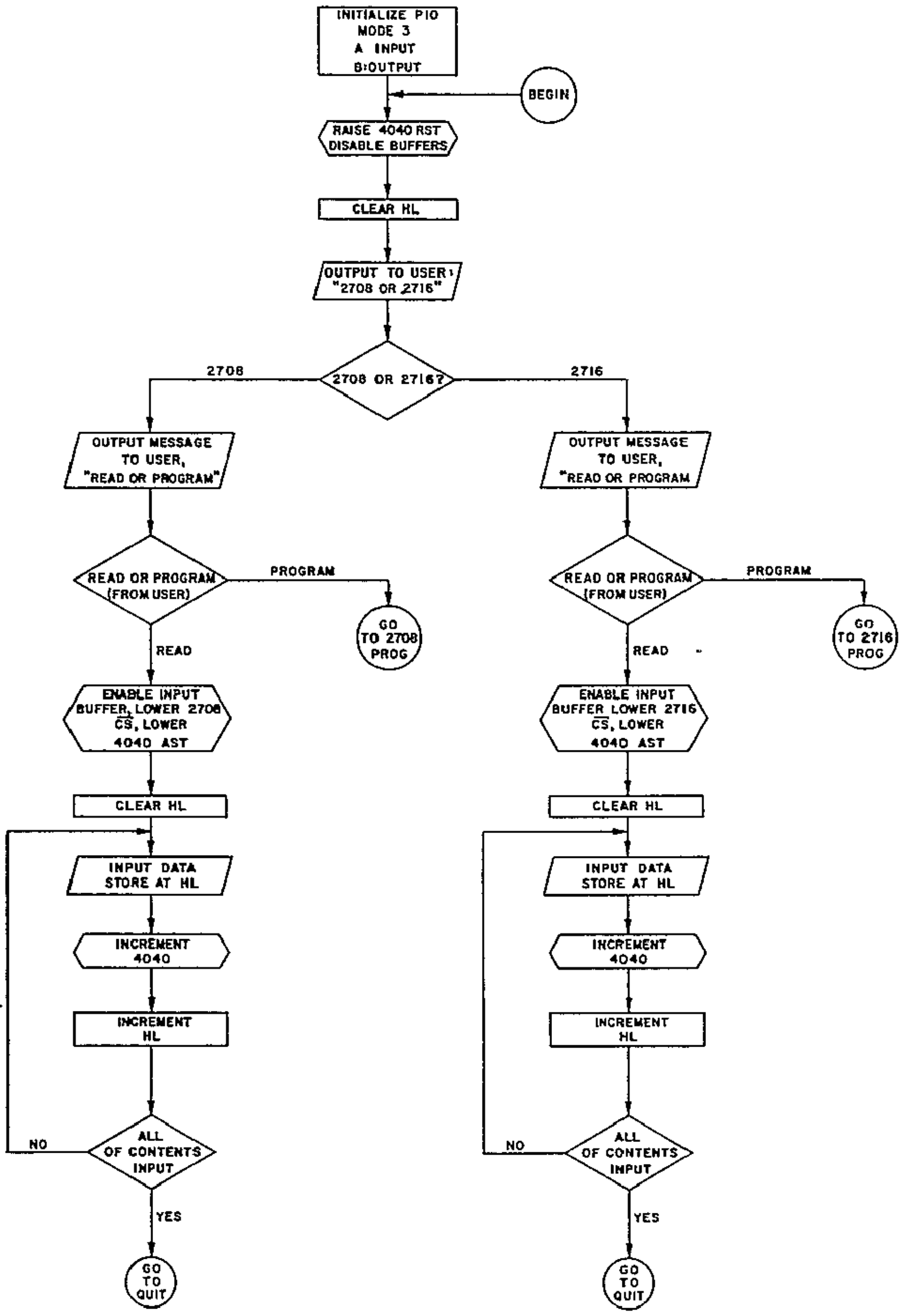

INITIALIZE PIO MODE 3 A INPUT B:OUTPUT
BEGIN
RAISE 4040 RST DISABLE BUFFERS
CLEAR HL
OUTPUT TO USER: "2708 OR 2716"
2708 OR 2716?
2708
2716
OUTPUT MESSAGE TO USER, "READ OR PROGRAM"
READ OR PROGRAM (FROM USER)
PROGRAM
GO TO 2708 PROG
GO TO 2716 PROG
READ
ENABLE INPUT BUFFER, LOWER 2708 CS, LOWER 4040 AST
ENABLE INPUT BUFFER LOWER 2716 CS, LOWER 4040 AST
CLEAR HL
INPUT DATA STORE AT HL
INCREMENT 4040
INCREMENT HL
ALL OF CONTENTS INPUT
NO
YES
GO TO QUIT

Figure 6.7 Flow chart of the EPROM programmer software. (a) The initialization and read routines.

(b)

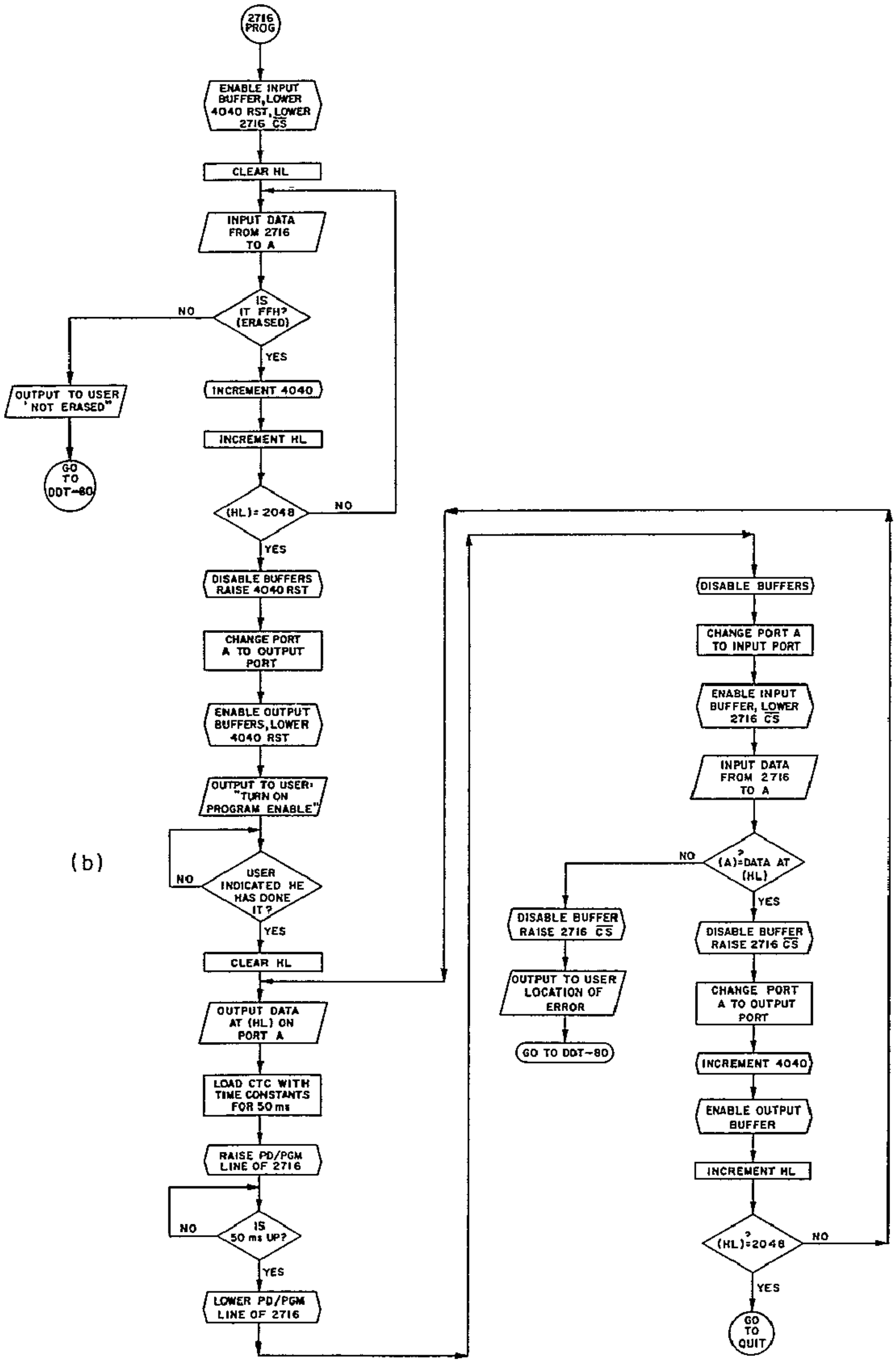

2716 PROG
ENABLE INPUT BUFFER, LOWER 4040 RST, LOWER 2716 $\overline{CS}$
CLEAR HL
INPUT DATA FROM 2716 TO A
IS IT FFH? (ERASED)
NO
YES
OUTPUT TO USER "NOT ERASED"
GO TO DDT-80
INCREMENT 4040
INCREMENT HL
(HL)= 2048
NO
YES
DISABLE BUFFERS RAISE 4040 RST
CHANGE PORT A TO OUTPUT PORT
ENABLE OUTPUT BUFFERS, LOWER 4040 RST
OUTPUT TO USER: "TURN ON PROGRAM ENABLE"
USER INDICATED HE HAS DONE IT?
NO
YES
CLEAR HL
OUTPUT DATA AT (HL) ON PORT A
LOAD CTC WITH TIME CONSTANTS FOR 50 ms
RAISE PD/PGM LINE OF 2716
IS 50 ms UP?
NO
YES
LOWER PD/PGM LINE OF 2716
DISABLE BUFFERS
CHANGE PORT A TO INPUT PORT
ENABLE INPUT BUFFER, LOWER 2716 $\overline{CS}$
INPUT DATA FROM 2716 TO A
(A) ?= DATA AT (HL)
NO
YES
DISABLE BUFFER RAISE 2716 $\overline{CS}$
OUTPUT TO USER LOCATION OF ERROR
GO TO DDT-80
DISABLE BUFFER RAISE 2716 $\overline{CS}$
CHANGE PORT A TO OUTPUT PORT
INCREMENT 4040
ENABLE OUTPUT BUFFER
INCREMENT HL
(HL) ?= 2048
NO
YES
GO TO QUIT

Figure 6.7 (continued) (b) 2716 program routine.

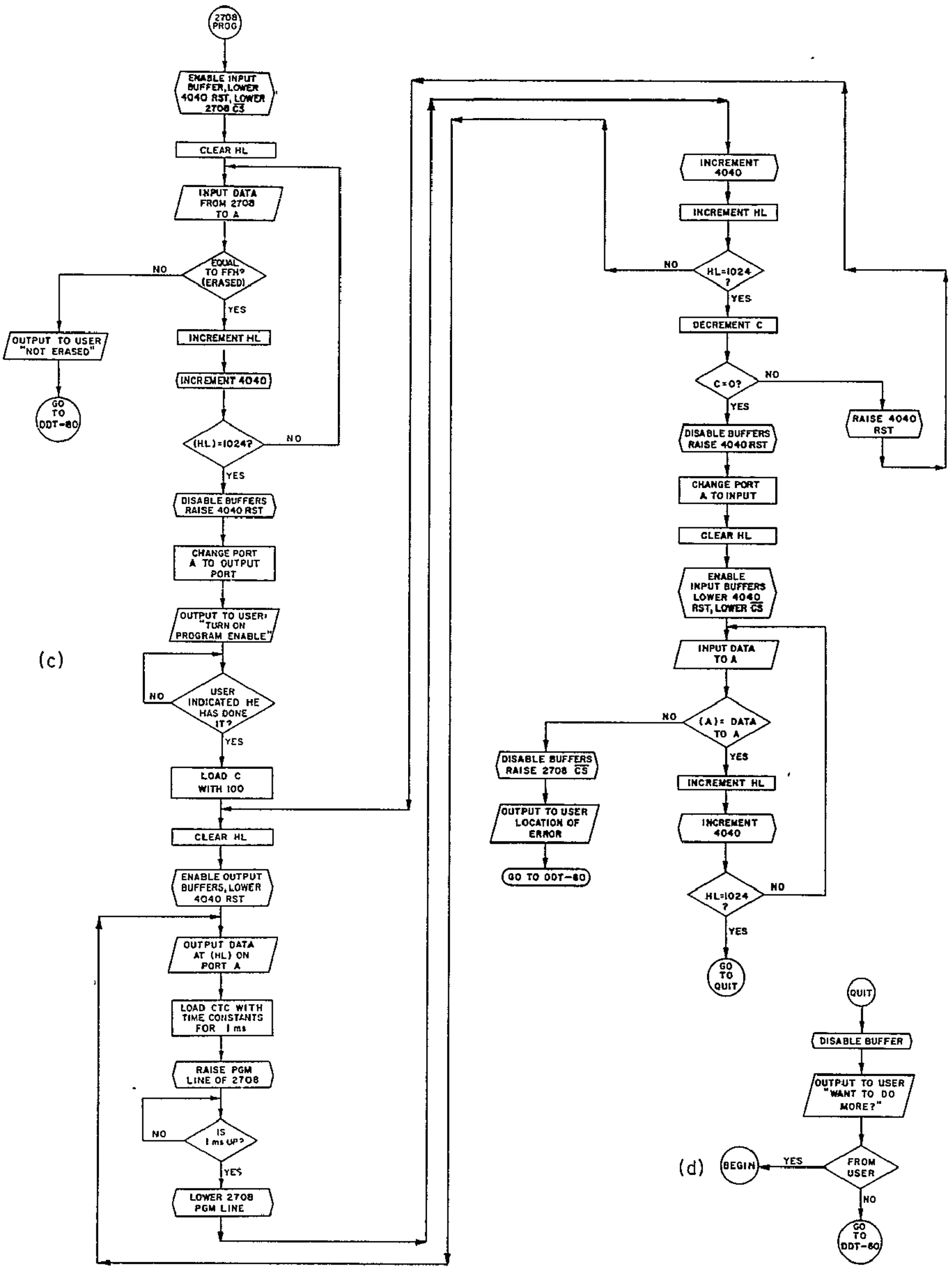

2708 PROG
ENABLE INPUT BUFFER, LOWER 4040 RST, LOWER 2708 $\overline{CS}$
CLEAR HL
INPUT DATA FROM 2708 TO A
EQUAL TO FFH? (ERASED)
NO
OUTPUT TO USER "NOT ERASED"
GO TO DDT-80
YES
INCREMENT HL
INCREMENT 4040
(HL)=1024?
NO
YES
DISABLE BUFFERS RAISE 4040 RST
CHANGE PORT A TO OUTPUT PORT
OUTPUT TO USER: "TURN ON PROGRAM ENABLE"
(c)
USER INDICATED HE HAS DONE IT?
NO
YES
LOAD C WITH 100
CLEAR HL
ENABLE OUTPUT BUFFERS, LOWER 4040 RST
OUTPUT DATA AT (HL) ON PORT A
LOAD CTC WITH TIME CONSTANTS FOR 1 ms
RAISE PGM LINE OF 2708
IS 1 ms UP?
NO
YES
LOWER 2708 PGM LINE
INCREMENT 4040
INCREMENT HL
HL=1024 ?
NO
YES
DECREMENT C
C=0?
NO
RAISE 4040 RST
YES
DISABLE BUFFERS RAISE 4040 RST
CHANGE PORT A TO INPUT
CLEAR HL
ENABLE INPUT BUFFERS LOWER 4040 RST, LOWER $\overline{CS}$
INPUT DATA TO A
(A)= DATA TO A
NO
DISABLE BUFFERS RAISE 2708 $\overline{CS}$
OUTPUT TO USER LOCATION OF ERROR
GO TO DDT-80
YES
INCREMENT HL
INCREMENT 4040
HL=1024 ?
NO
YES
GO TO QUIT
QUIT
DISABLE BUFFER
OUTPUT TO USER "WANT TO DO MORE?"
(d)
BEGIN
YES
FROM USER
NO
GO TO DDT-80

Figure 6.7 (continued) (c) 2708 program routine. (d) QUIT routine.

The program routines for the 2708 and 2716 differ chiefly in the way the programming pulses are applied and in the way the contents are checked after the EPROM is programmed. When either routine is entered, the contents are checked to make sure the EPROM has been erased (i.e., when an EPROM is erased, all locations contain FF H). After this is done, the 4040 and HL are both cleared, and port A is changed to an output port with the output buffers enabled. A message is then output to the user telling him to turn on the "program enable" switch on the programmer. He then hits any key on the teletype keyboard to start the programming operation.

To program one location of the 2716 data is output on port A and its PD/PGM line is raised for 50 ms as stated in the previous section. Since two channels of the CTC must be used to generate 50 ms, they must be programmed such that the sum of their time constants is 50 ms. One is programmed to start operating immediately while the other is programmed to start operating when its trigger line is brought low by the zero count output of the first channel. The second channel is checked continuously until it reaches zero. When this occurs, the PD/PGM line is brought low and the contents at that location are input and checked to make sure they agree with the data that was output to the 2716. To input the data port A is changed back to an input port, the input buffers are enabled, and the $\overline{CS}$ line of the 2716 is brought low. If the contents of the 2716 do not agree with the data that was output, the buffers are disabled, the $\overline{CS}$ line is brought high, and a message is output to the user notifying him of the error. The program routine is then exited and the SDB-80 jumps to the DDT-80 operating system. If the contents do agree, Port A is changed back to an output port with the output buffer enabled. HL and the 4040 are then incremenetd. If the contents of HL then equal 2048, the program routine then branches to "QUIT". Otherwise it continues as above.

To program the 2708 a 1 ms pulse is applied to the PGM line while the corresponding data is on port A for each location. This constitutes one loop and one-hundred loops must be gone through as stated in the previous section. Register C is used as the loop counter. Only one channel of the CTC has to be used since only a 1 ms pulse is required. After data is output to Port-A, the CTC is started and is checked continuously until it reaches zero. The PGM line is then brought low and the 4040 and HL are incremented. If the contents of HL equal 1024, then register C is incremented. Otherwise it continues as above. If the contents of C are less than one hundred, another loop is started.

If it equals one hundred, HL and the 4040 are cleared and Port A is changed to an input port along with the buffers. Data is then sequentially input and checked to see if it equals the original data. If it does not, the user is notified as in the case of a 2716 error and the SDB-80 jumps to the DDT-80 operating system. If all the locations agree, then the QUIT routine is entered.

In the QUIT routine the buffers are disabled and a message is output to the user asking if he wants to do anything else. If he answers yes, the program is started over at the beginning. Otherwise a jump statement to the DDT-80 operating system is executed.

6.2.5 *Operation of the programmer.* The programmer is simple to operate. The desired EPROM data is put into the SDB-80 RAM starting at location zero. Then the user causes the programmer firmware to start being executed by using the 'E' command of the DDT-80 operating system. The following sequence illustrates the operation of the programmer starting at the 'E' command.

```
'.E 4000'
ENTER A FOR 2708,B FOR 2716
'A'
ENTER P TO PROGRAM,R TO READ
'R'
WANT TO DO MORE? Y=YES,N=NO
'Y'
ENTER A FOR 2708,B FOR 2716
'A'
ENTER P TO PROGRAM,R TO READ
'P'
TURN ON PROGRAM ENABLE,THEN HIT ANY KEY
WANT TO DO MORE? Y=YES,N=NO
'N'
.
```

The above user responses are self-explanatory with the possible exception of the next to last one (the characters within the apostrophes are user responses). After the program switch is turned on, the user starts the programming operation by simply hitting a key on the teletype. Note that when the "N" is entered in the last statement, a jump to the DDT-80 operating system is executed.

The programmer firmware automatically checks to see if an EPROM is erased before it is programmed as noted in the last section. The following shows the response of the firmware when a user attempts to program an unerased EPROM.

```
'.E 4000'
ENTER A FOR 2708,B FOR 2716
'B'
ENTER P TO PROGRAM,R TO READ
'P'
NOT ERASED
```

If the data from a location in an EPROM does not agree with the data that should be there after it is programmed, the programmer firmware notifies the user as in the following sequence.

```
'.E 4000'
ENTER A FOR 2708,B FOR 2716
'B'
ENTER P TO PROGRAM,R TO READ
'P'
TURN ON PROGRAM ENABLE,THEN HIT ANY KEY
ERROR AT 0001
```

7. FLIGHT PERFORMANCE

The first flight test of the data-manipulation experiment took place on 19 June 1978: Nike Apache 14.543 was launched from Wallops Island, Virginia, at 2315 EST as part of the Joint American-Soviet Particle Intercalibration (JASPIC) Project.

The rocket attained an altitude of 185 km at 214 s after launch. It has a mid-flight spin rate of 6.85 Hz, well within the design range of the data-manipulation experiment. The spin period is 146 ms.

Figure 7.1 shows a section of the chart record about 5 s before the launch. The time interval between outputs of low order sectors is 160 ms and the output data sequence repeats at intervals of 2.54 s. Since the time interval between outputs of low order sectors is 160 ms instead of the expected value of 170 ms, it appears that the oscillator in the MSD has a frequency greater than the nominal value of 100 Hz (i.e., 17/160 ms $\approx$ 106 Hz). It can be seen that although there is no rocket spin the experiment is properly sequencing through the 16 sectors. (The signals from the detectors are principally noise).

A section of a chart record containing the signals from the data-manipulation experiment and the magnetometer is shown in Figure 7.2. This is taken at T + 95 s (T is the launch time), with the rocket at an altitude of 120 km.

The data output sequence repeats at intervals of 2.48 s. This is 17 revolutions (17 × 146 ms), as described in Section 5.2.3(c). The interval between the outputs from the low order sectors is 156 ms, in agreement with 1 + 1/16 revolutions. The intervals between the highest order sectors do not correspond exactly with the description contained in Section 5.2.3(b); in particular sector #15 finishes being output immediately before the output flag is set to output sector #0.

It appears that the flag to output sector #15 was not set when it should have been (i.e., when the MSD counter is reset to zero) and had to be set automatically when register B' had been incremented to 67; see Section 5.2.3(a). The problem could be due to an error in the program in the 2716 EPROM. Some of the object code involved in setting the flag for the last four sectors was entered manually instead of being assembled from the source listing given in Appendix II.1. Thus the error in setting the flag for sector #15 could have been the result of an error when typing in the object code. Possible improvements in this area are discussed in Chapter 8.

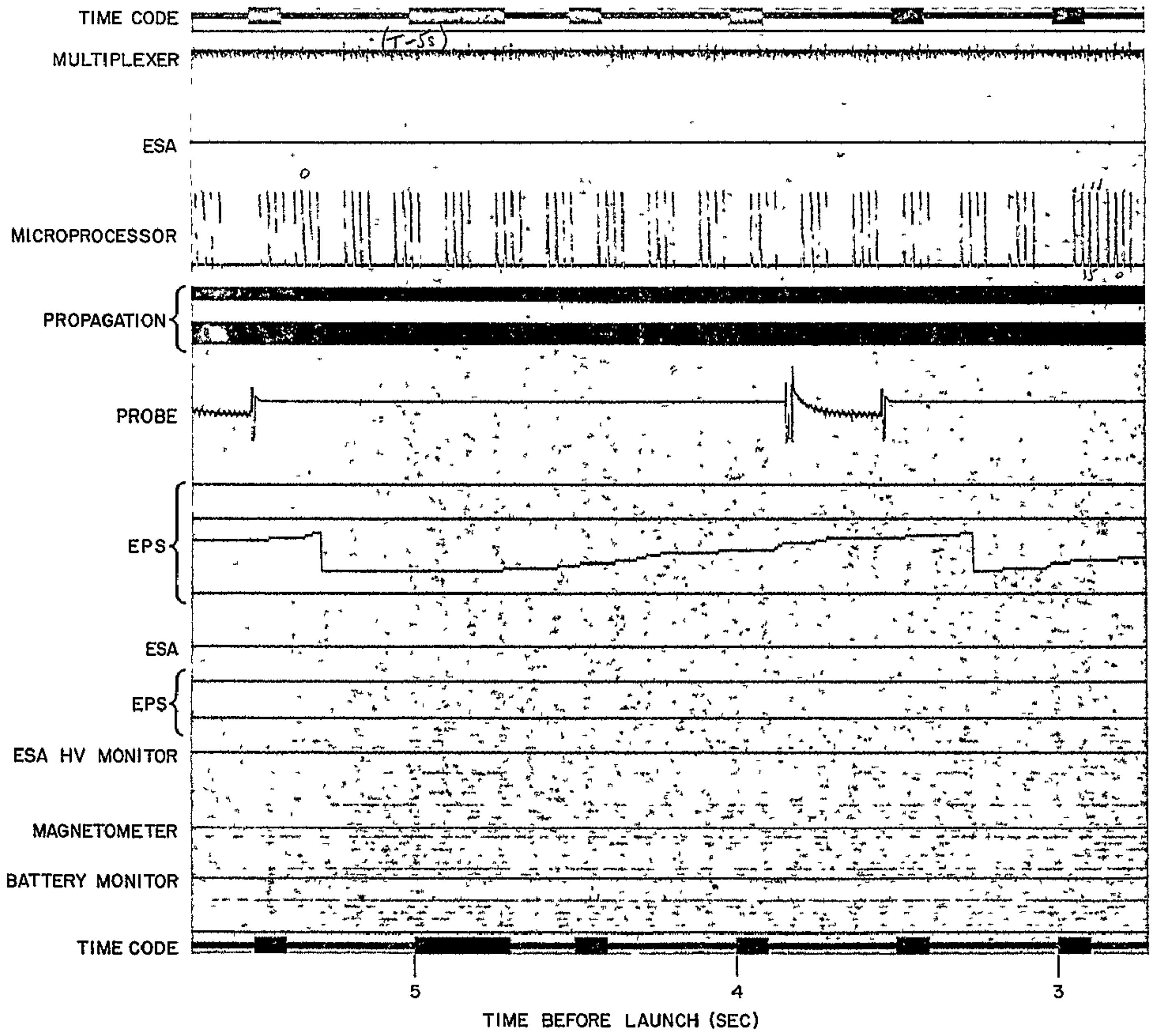


Figure 7.1 Section of chart record from Nike Apache 14.543: 5 seconds before launch.

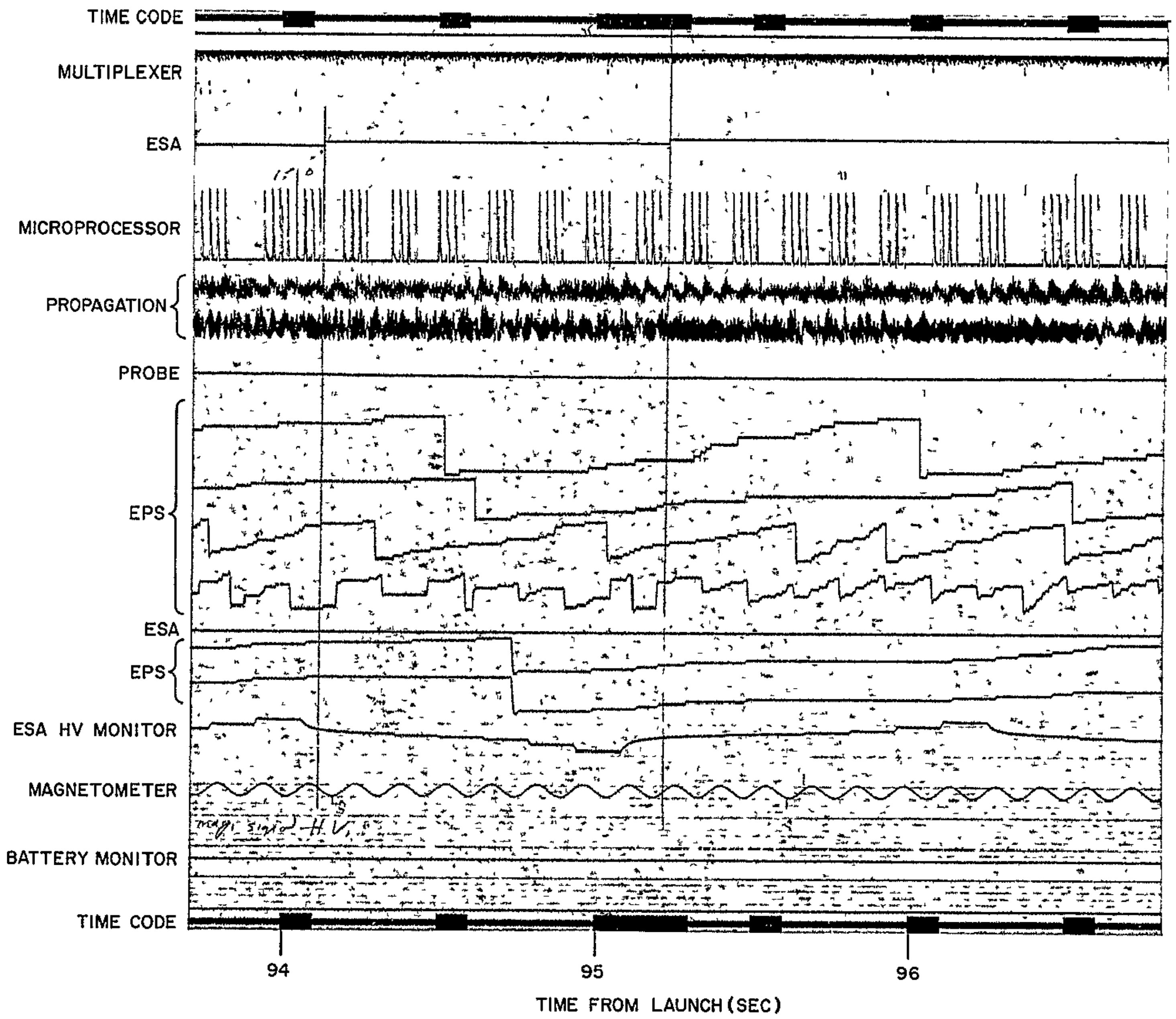


Figure 7.2 Section of chart record from Nike Apache 14.543: 95 seconds after launch.

Although sector #15 would not normally contain data for a 6.85 Hz spin rate, it did on this flight because of the 106 Hz frequency of the MSD oscillator. Since the spin period is 146 ms, sectors #0 to #14 have a duration (for input of data) of 9.4 ms for each revolution of the rocket. For these sectors each output represents a sampling time of 160 ms; see Section 5.2.3(c). Sector #15 has a duration of 5 ms for each revolution so that the sampling time for this sector is 85 ms (i.e., 17 × 5 ms).

The experiment performed well, with the anomalies noted above, until 182 s after launch (altitude 180 km). Subsequent to that time data appears in sector #0 with no data in the other sectors. Also the sequencing of sectors #12, 13 and 14 is anomalous. This is illustrated in Figure 7.3 with a section of chart record at T + 185 s. It appears that there was a failure involving the MSD or the PIO #2 (the magnetometer itself did not fail; the signal still appears on channel 6 of the telemetry system). Supporting this interpretation is the observation that the interval between the outputs of low order sectors was 203 ms; compare with Figure 5.7 which shows this interval for a low (or zero) spin rate. The data output sequence repeats at intervals of 3.02 s.

As noted all the data after T + 182 s is accumulated in one sector. The energy spectrums from the several detectors are obtained but without any information on variation with pitch angle. The signal from the microprocessor ends abruptly at T + 348 s (altitude 101 km, descending).

The second rocket carrying the data-manipulation experiment, Nike Apache 14.542, was launched from Wallops Island, at 0030 EST on 27 September 1978. This rocket attained an apogee of 183 km at T + 212 s.

Preliminary examination shows that the microprocessor data-manipulation experiment performed perfectly for the duration of the flight. None of the anomalies noted earlier were observed during this flight.

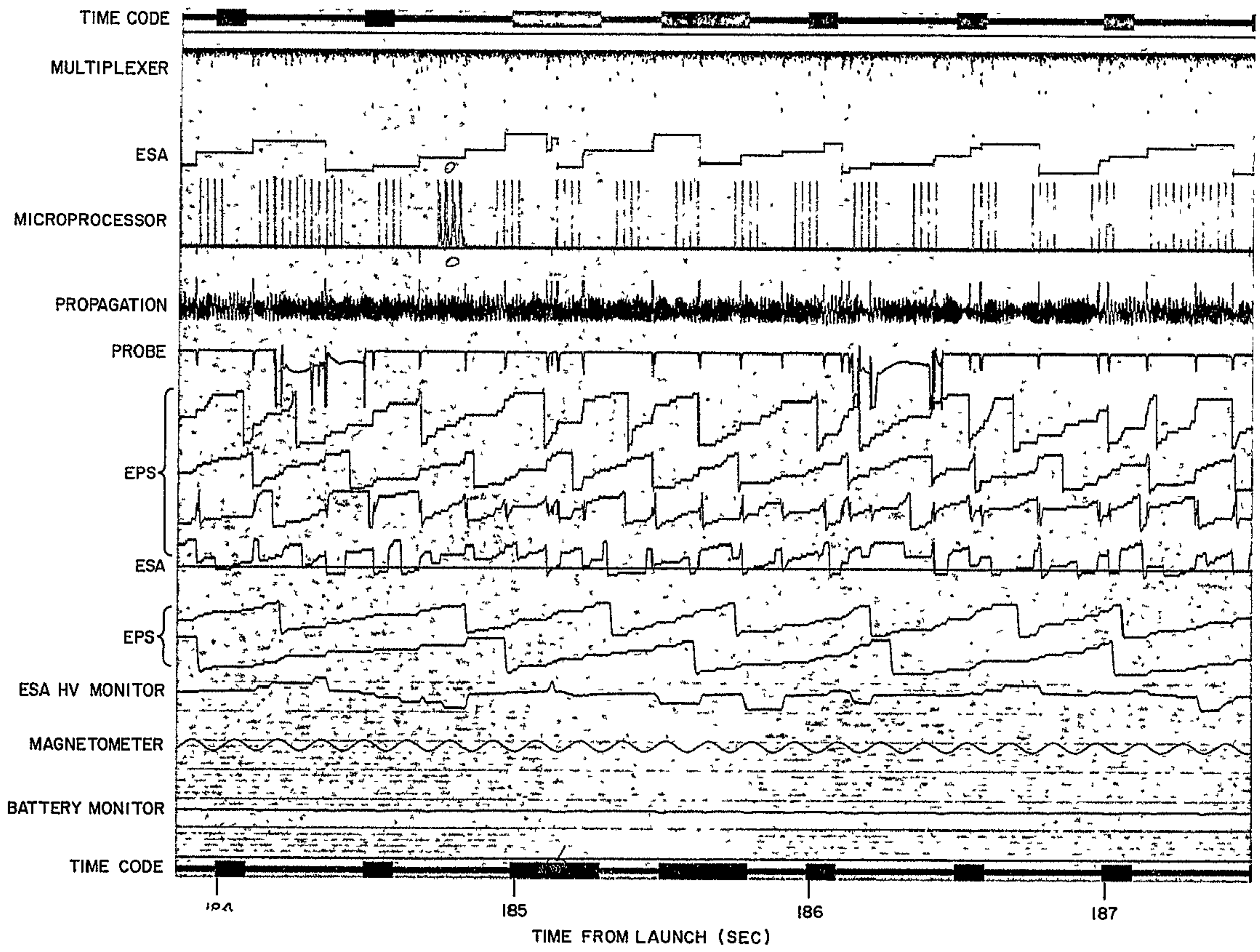


Figure 7.3 Section of chart record from Nike Apache 14.543: 185 seconds after launch.

8. CONCLUSION AND RECOMMENDATIONS FOR FUTURE WORK

8.1 *Conclusion*

In the preceding chapters the design and operation of the data-manipulation experiment has been discussed in detail. It has been shown that it is an effective system for reducing the data from the energetic particle experiments (described in Chapter 2) to a form correlating accumulations of particles at several discrete energy levels with their pitch-angle distributions. The output from this system will reduce the amount of time necessary for interpreting the data from the energetic particle experiment.

Without the use of a microprocessor this rocket-borne system could not have been constructed. Nike Apache 14.543 was the first to include a microprocessor on a University of Illinois payload. The success of this system on Nike Apache 14.543 will provide an impetus for updating other rocket-borne instrumentation by making greater use of digital processing and LSI integrated circuits.

8.2 *Suggestions for Future Work*

After the data-manipulation system was completed, it became evident that some things should be changed. These changes are outlined below and are followed by suggestions for other applications of microprocessors in rocket-borne instrumentation.

8.2.1 *The data-manipulation program.* A simple change in the data-manipulation program (listed in Appendix II.1) will enable the output flag to be set for the last four sectors in the same manner as it is for the other sectors; if the spin rate is too fast, it would continue to be set when the rocket had rotated to sector #0. To do this statement #102 ("JP NOSND") should be deleted and the two statements, "POP HL" and "PUSH HL", inserted in its place.

As an example of the result of this change suppose the spin rate is less than 6.67 Hz. The output flag to output sector #14 would be set when the rocket had rotated to sector #15, but if the spin rate were greater than 6.67 Hz (i.e., the MSD counter is reset before it reaches 15), the flag to output sector #14 would be set when the rocket had rotated to sector #0 as before.

8.2.2 *The MSD circuit.* The change, described above, in the data-manipulation program will not be necessary if the following changes are made in the MSD circuits.

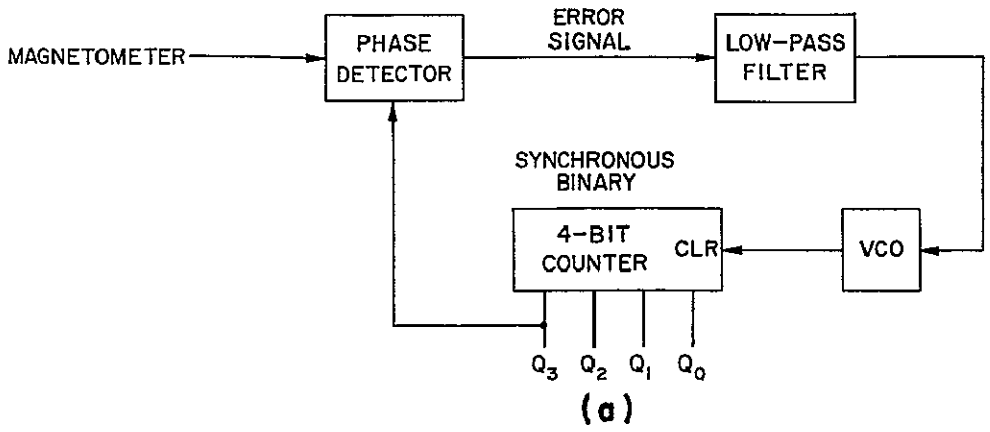
The method of converting the magnetometer signal to a digital form (Section 5.2) can be improved. It would be much better from a software standpoint and from a data interpretation standpoint to always have exactly sixteen sectors of data. If there were always sixteen sectors of data regardless of the spin rate, the output flag would be set in the same manner for all sectors and there would not be transmission of useless data; in the present system when the spin rate is above 6.67 Hz, the last few sectors may not contain any useful data.

The simplest approach would be to use an A/D converter to convert the magnetometer signal. The magnetometer signal would be made to vary between constant voltage limits, 0 and 5 V, for instance.

Another approach would be to use a counter as in the present system but with the counter clock changed such that its frequency was always sixteen times the frequency of the magnetometer signal. This can be done by phase-locking the clock oscillator to the magnetometer signal as shown in Figure 8.1(a). Since the frequency of the Q_3 output is equal to the frequency of the clock signal divided by 16, it is used to generate the error signal. The relation of the counter outputs in relation to the magnetometer signal is shown in Figure 8.1(b). The center frequency of the VCO should be about 108 Hz (6.75×16).

The CTC could also be used to generate a digital representation of the magnetometer signal. Since only one counter/timer channel (channel #0) is used in the present system, the other three are available for other applications. Referring to Figure 5.2(b) it is seen that the external clock/timer trigger input can be used to decrement the down counter. The zero count/timeout output goes high when the counter is decremented to zero. The magnetometer signal could be used to decrement the channel #1 counter. Since the CTC interrupts the CPU at intervals of 1.536 ms, the cycle time of the magnetometer signal can be determined by counting the number of intervals it takes to decrement the counter. For instance if it takes 93 intervals to decrement the counter by one, the time constant of the magnetometer signal is 142.8 ms (i.e., 93×1.536 ms = 1/7 Hz). Once the time constant is determined, channel #2, programmed to be a timer, can be given a time constant of 8.925 ms (i.e., 142.8 ms/16). The zero count/timeout output of channel #2 can then be used to decrement the counter of channel #3. The lower four bits of channel #3 then would be the digital representation of the magnetometer signal.

The advantage of this method over the phase-lock technique is that it takes little extra hardware and would cover a wider frequency range. The



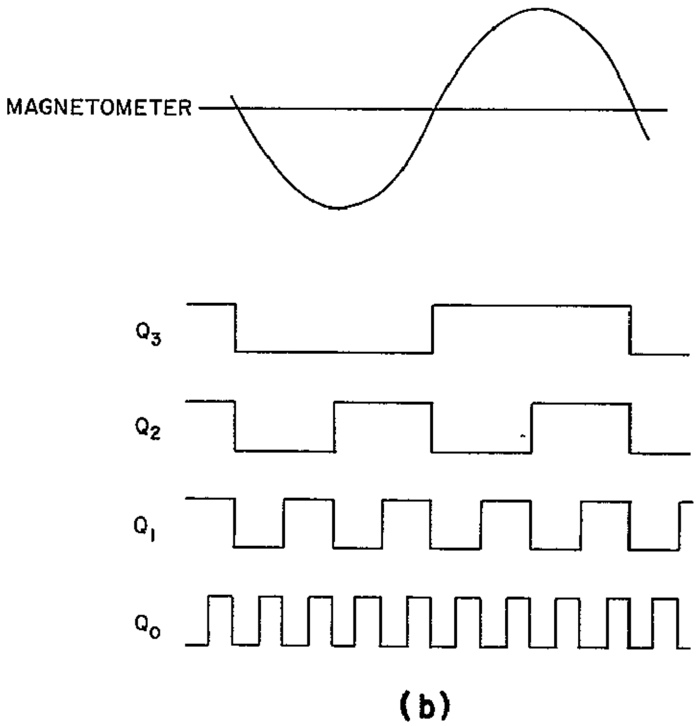


Figure 8.1 Suggestion for improvement in the MSD system. (a) Block diagram showing how the system can be phase locked to the magnetometer. (b) Timing diagram of the counter compared to the magnetometer signal.

disadvantage is that it has a much slower response time and it would require some CPU time. Figure 8.2 illustrates the extra connections to the CTC that would be required.

8.2.3 *Automatic reset circuit.* An automatic reset circuit would be useful during a flight if the CPU jumped to some unspecified location outside the defined program, perhaps due to noise. The circuit shown in Figure 8.3 would reset the CPU if it is not addressed after a period of time. For instance in the present system the CTC interrupts the CPU every 1.536 ms causing it to jump to the output routine. The reset circuit could be made to reset the system if this did not happen. One shot #1 would be triggered by the CPU during initialization. The pulse would be sustained if it were addressed periodically. If it were not retriggered, its 'Q' output would go low triggering one-shot #2 causing the system to receive a reset pulse. By requiring an I/O instruction to be used to retrigger one-shot #1 the probability of retriggering it unintentionally is reduced.

Other schemes such as requiring a particular data word at one address can be used to reduce that probability even further, thereby absolutely ensuring a reset if the system is not functioning properly.

8.2.4 *Circuit layout.* The printed circuit cards should be changed by separating standard microcomputer components from interface devices that will vary according to the requirements of a particular experiment. Cards containing such components as the CPU, memory and PIO's could be used for several applications without having to be changed. The method of connecting the cards together should be changed to edge connectors. This would allow a card to be easily replaced thereby simplifying debugging of the system. The cards could be strapped in by a bar on top of the cards during a flight to prevent any movement (the bar would also prevent movement when the system was foamed).

8.2.5 *The development system.* The development system should be improved by the addition of an in-circuit emulator and a good bulk-storage device. This will cut the development time by a significant amount.

Presently there is no equipment, other than an oscilloscope, available for debugging hardware problems. The in-circuit emulator would be useful in debugging both the system hardware and software. It interfaces external circuit boards to the SDB-80 by a 40-pin connector inserted in place of the Z-80 CPU. It allows the actual software to be run on the boards just as if the CPU were still there while giving the user such capabilities as a real-

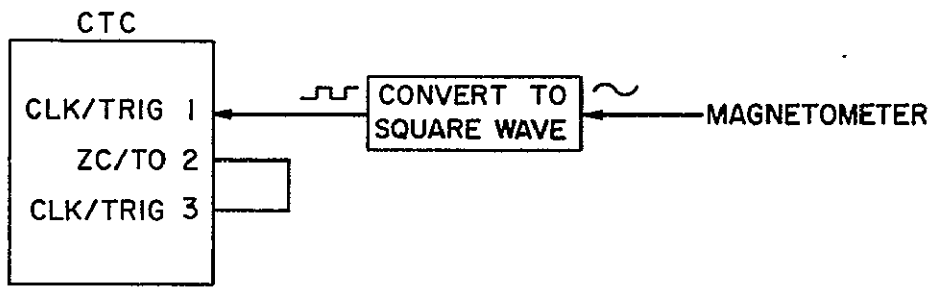
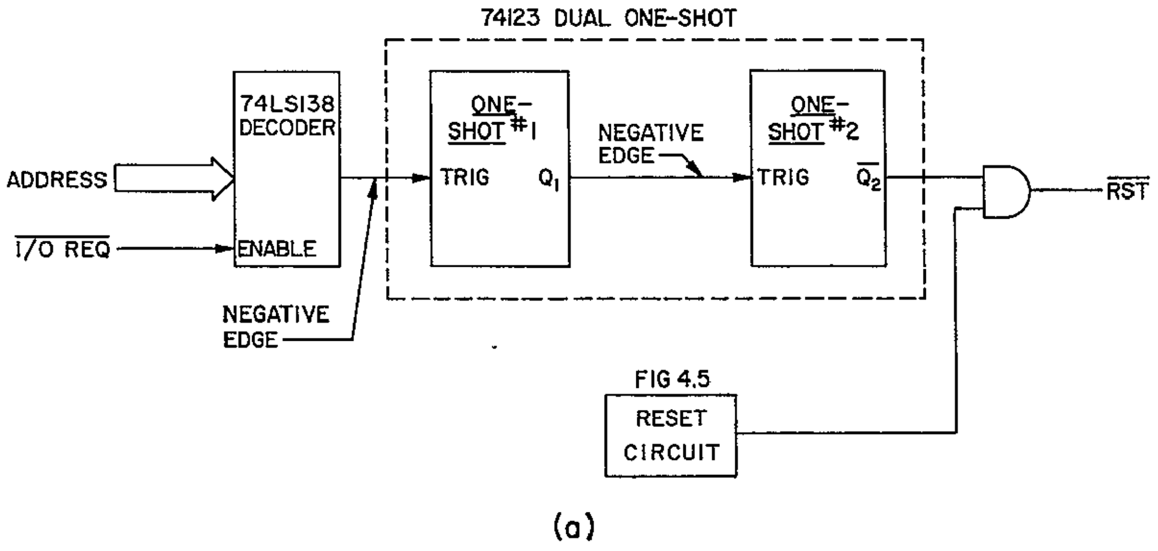


Figure 8.2 Using the CTC to generate the MSD signal.



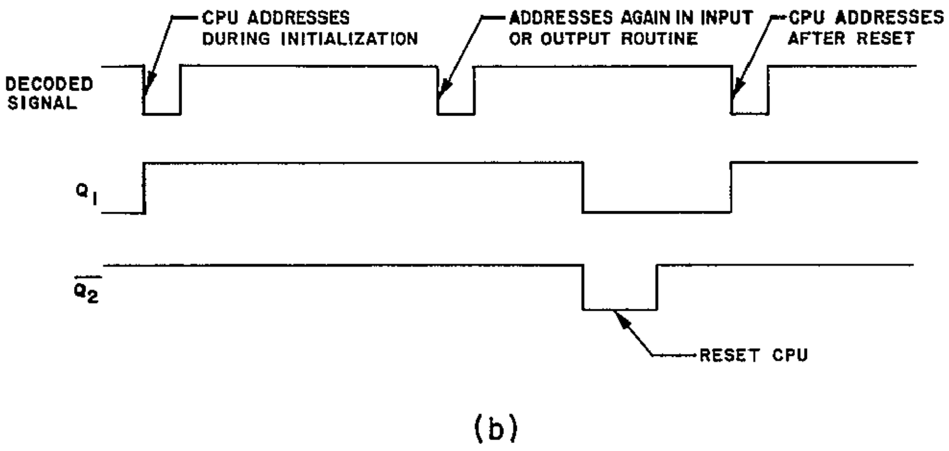

CPU ADDRESSES DURING INITIALIZATION
ADDRESSES AGAIN IN INPUT OR OUTPUT ROUTINE
CPU ADDRESSES AFTER RESET
DECODED SIGNAL
Q_1
$\overline{Q_2}$
RESET CPU
(b)

Figure 8.3 (a) Block diagram of the automatic reset circuit.
(b) Timing diagram showing when the CPU would be reset.

time trace of program execution and single-stepping through a program. A faster bulk storage device than the teletype paper type reader/punch would allow programs to be changed faster.

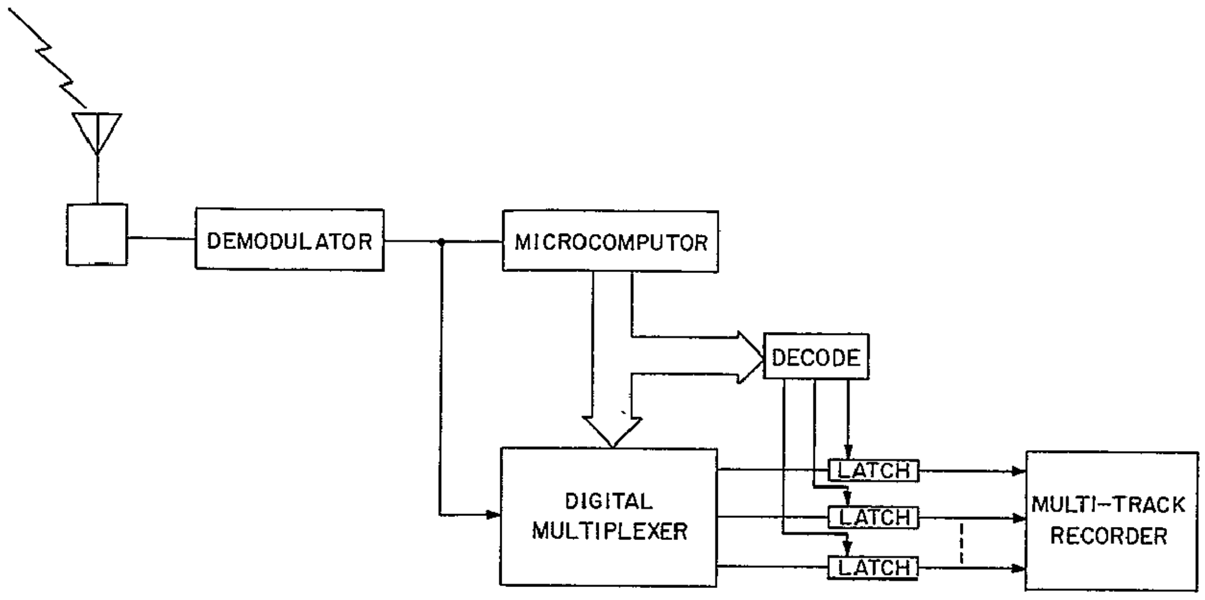
There are software drivers in the SDB-80 operating system (DDT-80) for a high-speed paper tape reader and a punch. There is also a compatible flexible disc unit available from MOSTEK. More sophisticated hardware and software could be accomplished with this additional equipment.

8.2.6 *Other applications of the rocket-borne data-manipulation experiment.* Another function a microprocessor could perform during the rocket flight is the calculation of electron temperature. This function could be added to the present system since one channel of PIO #2 was not used and since there is plenty of available space left on the EPROM. This calculation would not degrade the performance of the system since it would be done at relatively long time intervals.

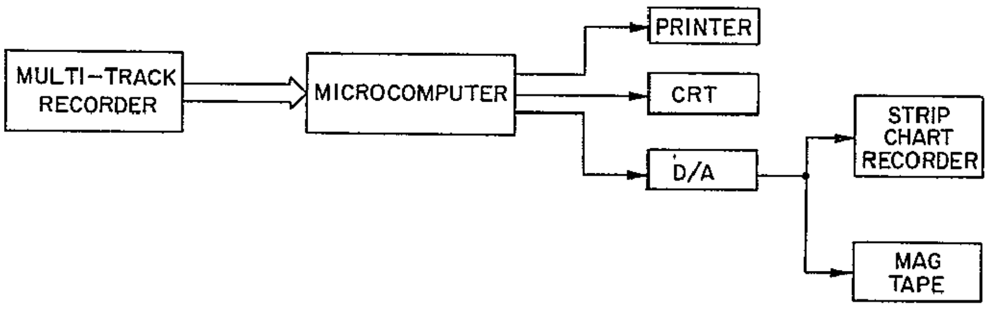
8.2.7 *Data-transmission systems.* An even more valuable application would be in the data-transmission system. Currently this system is an analog frequency-division multiplexed FM device. It should be upgraded to a time-division multiplexed digital system possibly using a microcomputer. This would allow higher resolution data to be sent back. By using a microprocessor-controlled digital telemetry system data flow could be optimized. Since some experiments will have faster transients, they could be allotted more time slots. Other experiments might require data to be sent only at non-periodic intervals depending on the occurrence or non-occurrence of external stimuli. Interrupts could be used to send back the non-periodic data with priorities assigned depending on the most important occurrence, and a time slot could be specifically allocated for all the non-periodic data.

Some of the code-decode (codec) devices that have recently become available could be useful in a rocket-borne digital telemetry system. These devices convert an analog signal in the voice-frequency range to a digital bit stream using either pulse code or delta modulation. Some even have facilities for microprocessor control included.

8.2.8 *Applications in ground-based data processing.* A ground-based microprocessor-controlled system could be used (in conjunction with the data transmission systems) for bit-synchronization and demultiplexing the data. After the systems were synchronized, the demultiplexed data would be placed on a multi-track tape. After the flight the same system could be used to process the data off-line. Figure 8.4 depicts block diagrams of the ground-based data-reduction systems.



(a)



(b)

Figure 8.4 Ground-based data-reduction system. (a) Real-time data-logging system. (b) Off-line data-reduction system.

REFERENCES

Geller, M. A., L. G. Smith and H. D. Voss (1975), Analysis of nighttime *E*-region winds and ionization production, *Radio Sci.*, *10*, 335-345.

Leung, W., L. G. Smith and H. D. Voss (1979), A rocket-borne pulse-height analyzer for energetic particle measurements, *Aeron. Rep. No. 83*, Aeron. Lab., Dep. Elec. Eng., Univ. Ill., Urbana.

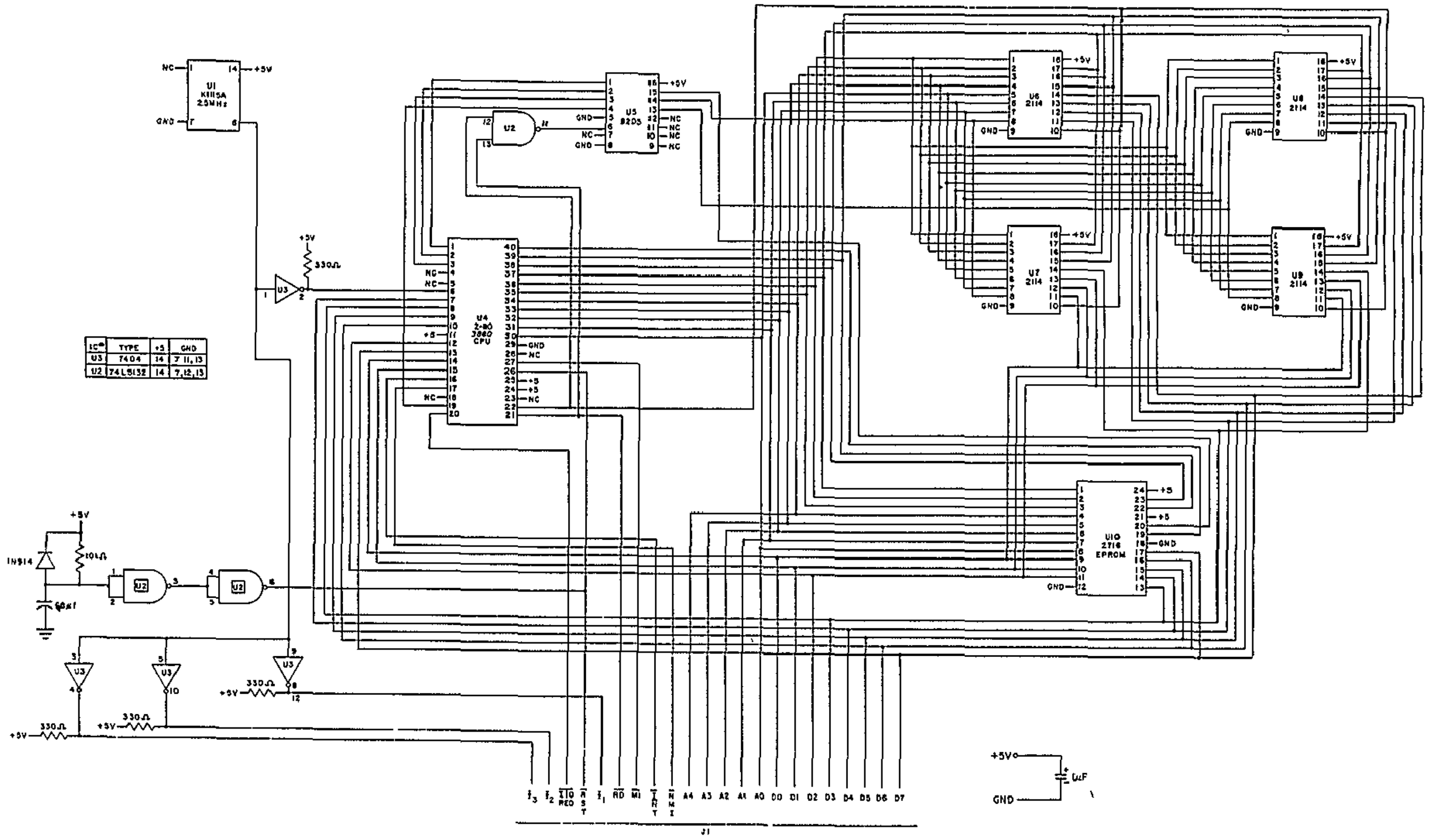
Pozzi, M. A., L. G. Smith and H. D. Voss (1979), A rocket-borne energetic analyzer for measurement of energetic particle flux, *Aeron. Rep. No. 82*, Aeron. Lab., Dep. Elec. Eng., Univ. Ill., Urbana.

Smith, L. G., M. A. Geller and H. D. Voss (1974), Energetic electrons in the midlatitude nighttime *E* region, *J. Atmos. Terr. Phys.*, *36*, 1601-1612.

Smith, L. G. and H. D. Voss (1976), Ionization by energetic electrons in the midlatitude nighttime *E* region, *Space Res.*, *17*, 427-432.

Voss, H. D. and L. G. Smith (1974), Design and calibration of a rocket-borne electron spectrometer for investigation of particle ionization in the nighttime midlatitude *E* region, *Aeron. Rep. No. 62*, Aeron. Lab., Dep. Elec. Eng., Univ. Ill., Urbana.

Voss, H. D. and L. G. Smith (1977), Energetic particles and ionization in the nighttime middle and low latitude ionosphere, *Aeron. Rep. No. 78*, Aeron. Lab., Dep. Elec. Eng., Univ. Ill., Urbana.



IC	TYPE	+5	GND
U3	7404	14	7 11, 13
U2	74LS132	14	7,12,13

Figure I.1 CPU-memory board.

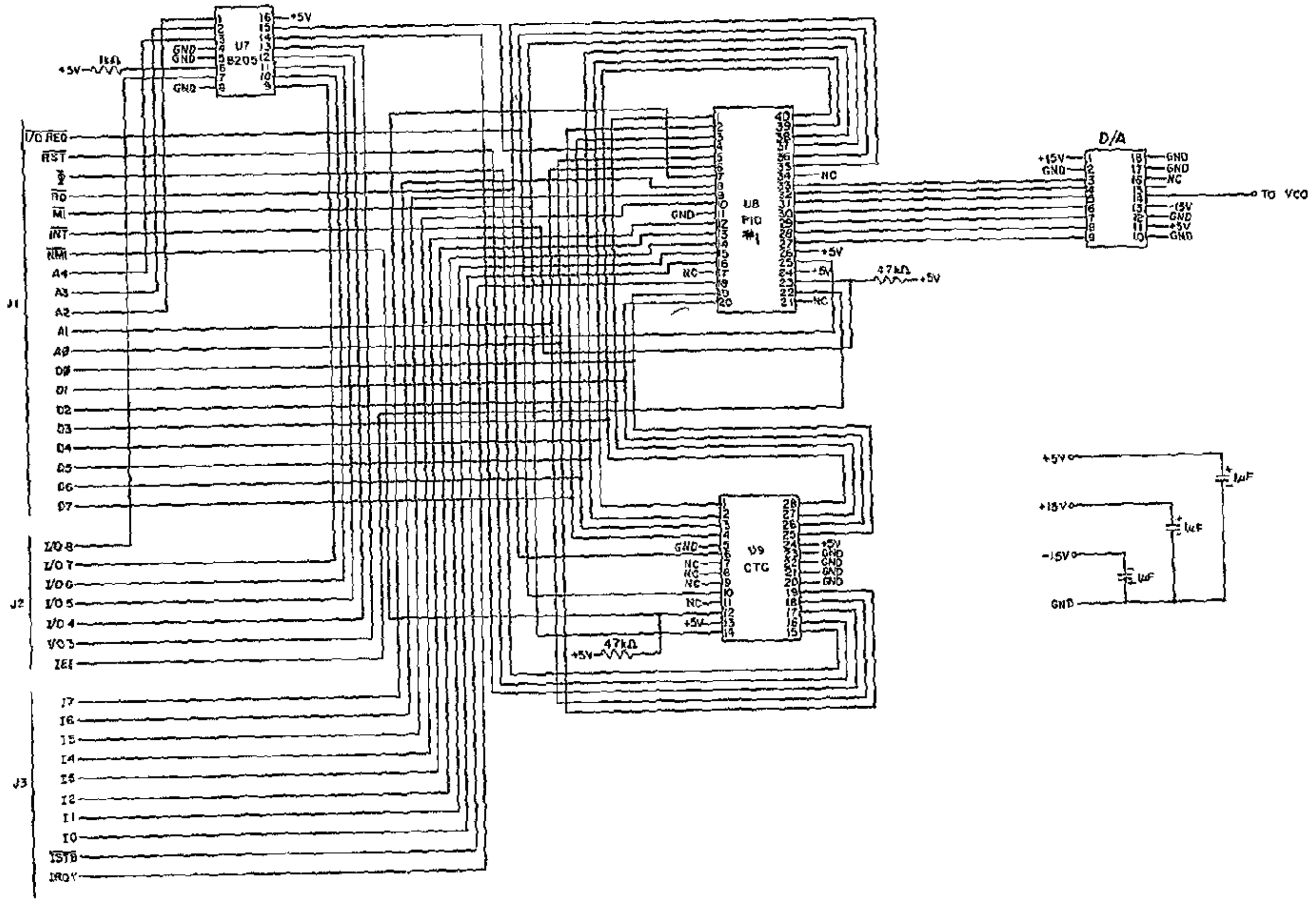


Figure I.2 PIO#1 and CTC board.

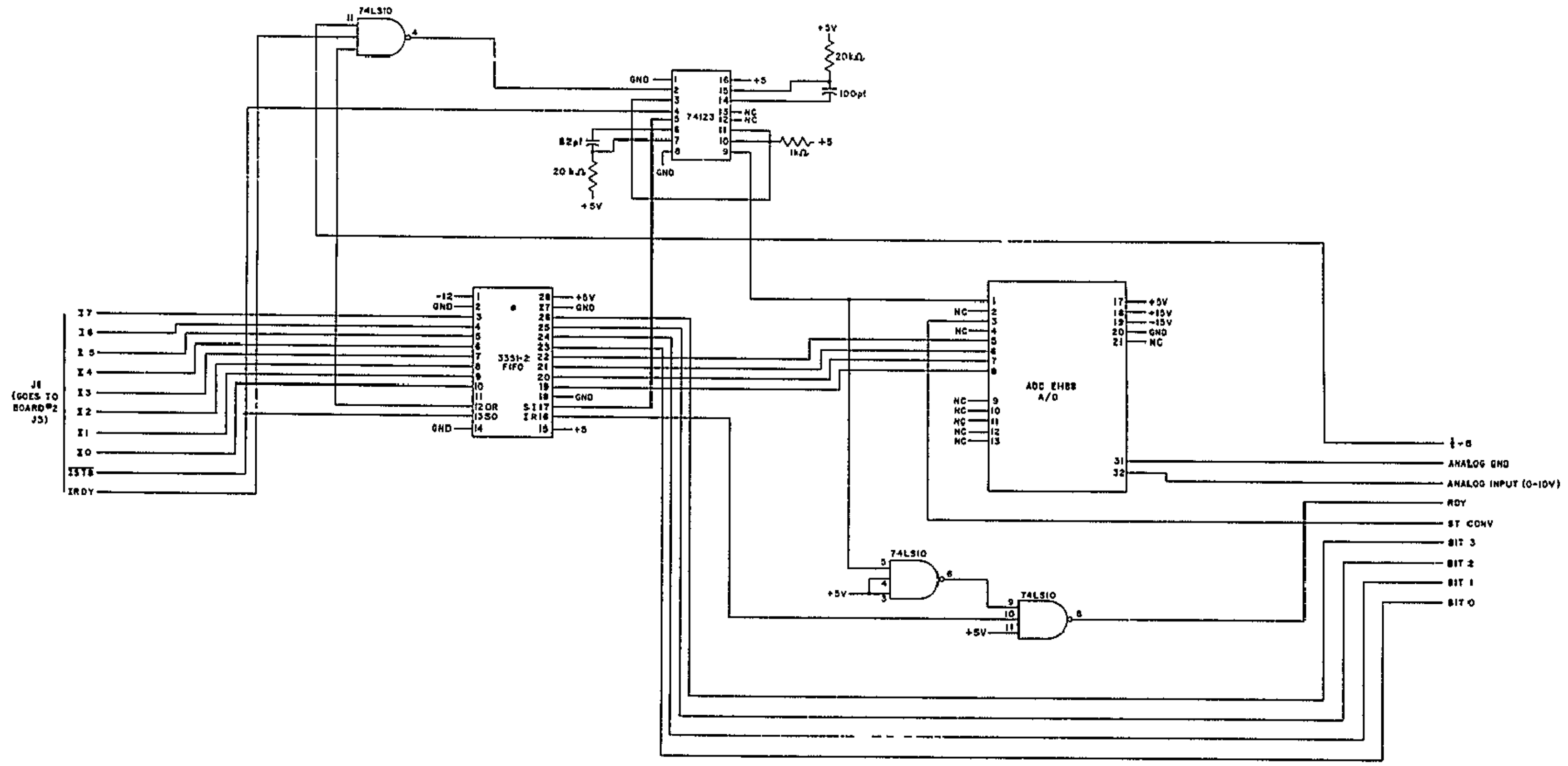


Figure I.3 A/D and FIFO board.

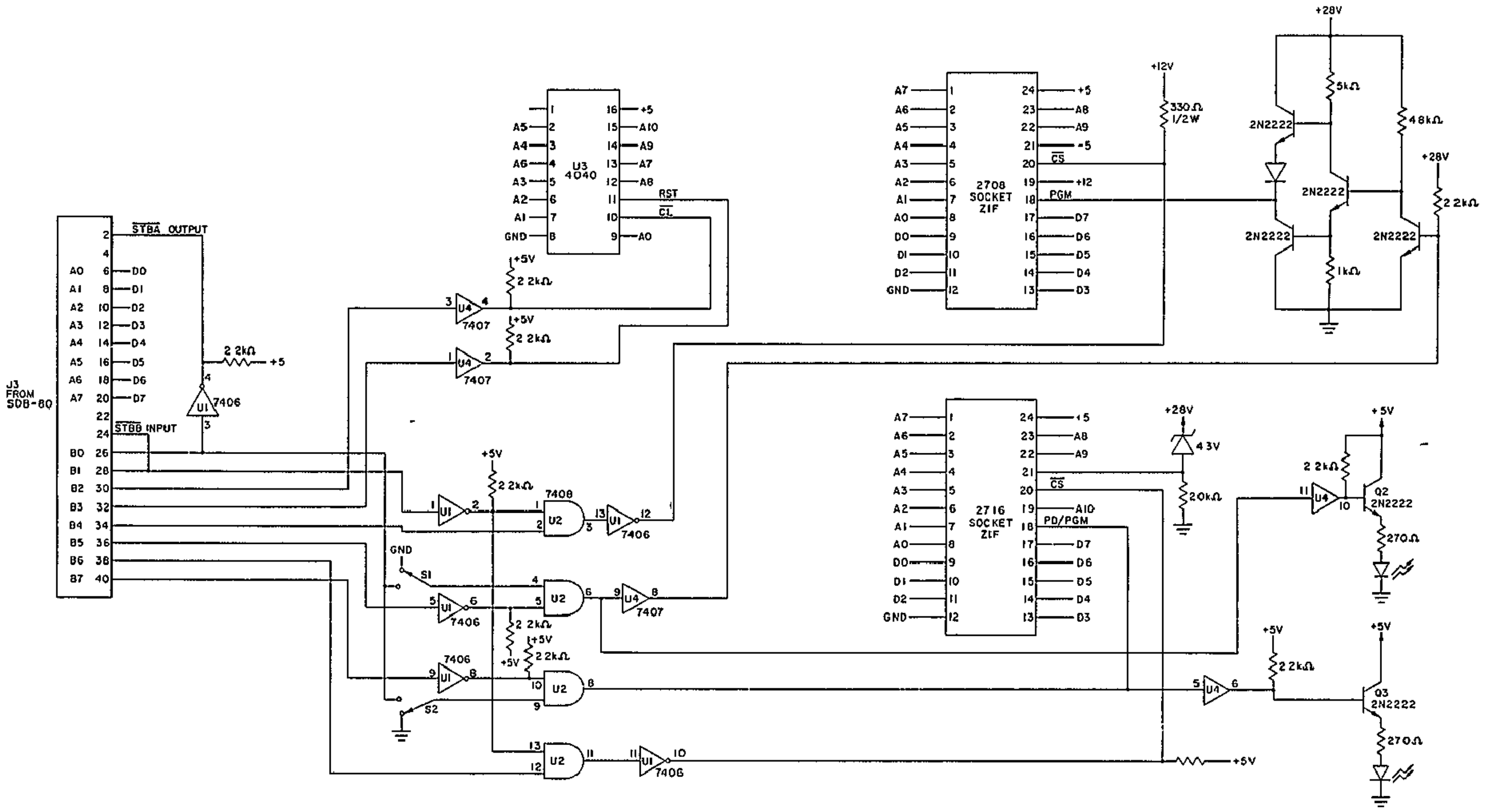


Figure I.4 EPROM programmer.

APPENDIX II.1 . Data-Manipulation Program

```
    DATA MANIPULATION PROGRAM
 ADDR  OBJECT     ST #

 >ØØØØ            ØØØ2 P1A      EQU  ØØ              ;PIO#1 PORT A
 >132Ø            ØØØ3 END      EQU  132ØH           ;END OF SECTOR ADDR.
 >ØØØ1            ØØØ4 P1AC     EQU  Ø1              ;PIO#1 PORT A CONTROL
 >ØØØ2            ØØØ5 P1B      EQU  Ø2              ;PIO#1 PORT B
 >ØØØ3            ØØØ6 P1BC     EQU  Ø3              ;PIO#1 PORT B CONTROL
 >ØØØ8            ØØØ7 P2A      EQU  Ø8              ;PIO#2 PORT A
 >ØØØ9            ØØØ8 P2AC     EQU  Ø9              ;PIO!2 PORT A CONTROL
 >13FØ            ØØØ9 STAK     EQU  13FØH           ;STACK ADDRESS
 >ØØØ4            ØØ1Ø CT1      EQU  Ø4              ;CTC CHANNEL #1 ADDRES
                  ØØ11          ORG  ØØ
'ØØØØ  ED56       ØØ12          IM   1               ;INTERRUPT MODE 1
'ØØØ2  31FØ13     ØØ13          LD   SP,STAK         ;STACK POINTER
'ØØØ5  3E4F       ØØ14          LD   A,Ø1ØØ1111B     ;MODE 1 CONTROL WORD
'ØØØ7  D3Ø1       ØØ15          OUT  (P1AC),A        ;PIO#1 PORT A
'ØØØ9  3ECF       ØØ16          LD   A,11ØØ1111B     ;MODE 3 CONTROL WORD
'ØØØB  D3Ø3       ØØ17          OUT  (P1BC),A        ;PIO#1 PORT B
'ØØØD  D3Ø9       ØØ18          OUT  (P2AC),A        ;PIO#2 PORT A
'ØØØF  3EØØ       ØØ19          LD   A,Ø
'ØØ11  D3Ø3       ØØ2Ø          OUT  (P1BC),A        ;OUTPUT PORT
'ØØ13  3EFF       ØØ21          LD   A,ØFFH          ;PROGRAM TO BE
'ØØ15  D3Ø9       ØØ22          OUT  (P2AC),A        ;INPUT PORT
'ØØ17  3EA5       ØØ23          LD   A,1Ø1ØØ1Ø1B     ;CTC CONTROL WORD
'ØØ19  D3Ø4       ØØ24          OUT  (CT1),A         ;CHANNEL #1
                  ØØ25 ;CLEAR LOWER 1K OF RAM USING BLOCK MOVE.
'ØØ1B  21ØØØ8     ØØ26          LD   HL,8ØØH         ;BEGINNING ADDRESS
'ØØ1E  36ØØ       ØØ27          LD   (HL),ØØ         ;CLEAR FIRST LOCATION
'ØØ2Ø  11Ø1Ø8     ØØ28          LD   DE,8Ø1H         ;ADDRESS TO MOVE TO
'ØØ23  Ø1FFØ3     ØØ29          LD   BC,1Ø23         ;NUMBER OF MOVES - 1
'ØØ26  EDBØ       ØØ3Ø          LDIR                 ;BLOCK MOVE
'ØØ28  D9         ØØ31          EXX
'ØØ29  Ø1ØØØØ     ØØ32          LD   BC,ØØ           ;SET UP BC' AND HL'
'ØØ2C  21ØØØ8     ØØ33          LD   HL,8ØØH         ;FOR OUTPUT ROUTINE
'ØØ2F  D9         ØØ34          EXX
'ØØ3Ø  C35ØØ1'    ØØ35          JP   SPLT            ;GO AROUND OTHER STUFF
                  ØØ36          ORG  15ØH
'Ø15Ø  DBØ8       ØØ37 SPLT:    IN   A,(P2A)         ;INPUT MSD DATA
'Ø152  E6ØF       ØØ38          AND  ØØØØ1111B       ;GET RID OF EXTRA BITS
'Ø154  57         ØØ39          LD   D,A
'Ø155  3EØØ       ØØ4Ø          LD   A,ØØ
'Ø157  CB3A       ØØ41          SRL  D               ;SHIFT TO PROPER BIT
'Ø159  1F         ØØ42          RRA                  ;POSITIONS.
'Ø15A  CB3A       ØØ43          SRL  D
'Ø15C  1F         ØØ44          RRA                  ;LOWER TWO BITS IN A
'Ø15D  6F         ØØ45          LD   L,A             ;PUT SHIFTED DATA
'Ø15E  3EØ8       ØØ46          LD   A,Ø8            ;IN HL AND DE AFTER
'Ø16Ø  82         ØØ47          ADD  A,D             ;ADDING ON UPPER BIT.
'Ø161  57         ØØ48          LD   D,A
'Ø162  62         ØØ49          LD   H,D
'Ø163  3E83       ØØ5Ø          LD   A,1ØØØØØ11B     ;ENABLE PIO#1 INT.
'Ø165  D3Ø1       ØØ51          OUT  (P1AC),A
'Ø167  3EØF       ØØ52          LD   A,15            ;CTC TIME CONSTANT
'Ø169  D3Ø4       ØØ53          OUT  (CT1),A         ;START CTC
'Ø16B  FB         ØØ54          EI                   ;ENABLE INTERRUPT FF
'Ø16C  DBØØ       ØØ55          IN   A,(P1A)         ;RAISE ARDY LINE
'Ø16E  76         ØØ56 HLT:     HALT                 ;WAIT FOR INTERRUPTS
'Ø16F  C36EØ1'    ØØ57          JP   HLT
                  ØØ58          ORG  38H             ;INPUT ROUTINE ADDRESS
'ØØ38  62         ØØ59          LD   H,D
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
     DATA MANIPULATION PROGRAM
 ADDR  OBJECT     ST #

'0039  DB00       0060          IN    A,(P1A)        ;INPUT DATA FROM FIFO
'003B  83         0061          ADD   A,E            ;CONCATENATE DATA WITH
'003C  6F         0062          LD    L,A            ;MSD DATA IN DE.
'003D  7E         0063          LD    A,(HL)         ;INPUT DATA TO A
'003E  C604       0064          ADD   A,04           ;INCR. THIRD BIT
'0040  77         0065          LD    (HL),A
'0041  FB         0066          EI                   ;REENABLE MASKABLE INT
'0042  ED4D       0067          RETI
                  0068          ORG   66H            ;OUTPUT ROUTINE ADD.
'0066  08         0069          EX    AF,AF'         ;EXCHANGE AF REGISTERS
'0067  DB08       0070          IN    A,(P2A)        ;UPDATE MSD DATA IN DE
'0069  E60F       0071          AND   00001111B
'006B  57         0072          LD    D,A
'006C  3E00       0073          LD    A,00
'006E  CB3A       0074          SRL   D
'0070  1F         0075          RRA
'0071  CB3A       0076          SRL   D
'0073  1F         0077          RRA
'0074  5F         0078          LD    E,A
'0075  3E08       0079          LD    A,08H
'0077  82         0080          ADD   A,D
'0078  57         0081          LD    D,A
'0079  D5         0082          PUSH  DE             ;STORE DE
'007A  D9         0083          EXX                  ;EXCHANGE REGISTERS
'007B  D1         0084          POP   DE             ;PUT DE DATA IN DE'
'007C  3E01       0085          LD    A,01
'007E  B9         0086          CP    C              ;SEE IF OUTPUT FLG SET
'007F  CAC800'    0087          JP    Z,SEND1        ;GO TO SEND IF IT IS
'0082  04         0088          INC   B
'0083  78         0089          LD    A,B
'0084  FE43       0090          CP    67             ;SEE IF B' HAS BEEN
'0086  CAB600'    0091          JP    Z,SNDF1        ;INCR. TO 67,IF SO SET
'0089  E5         0092          PUSH  HL
'008A  7C         0093          LD    A,H
'008B  FE0B       0094          CP    0BH            ;SEE IF LAST 4 SECTORS
'008D  C29D00'    0095          JP    NZ,NOTBE       ;IF NOT GO TO NOTBE
'0090  37         0096          SCF                  ;CLEAR CARRY
'0091  3F         0097          CCF
'0092  ED52       0098          SBC   HL,DE          ;SUBTRACT DE' FROM HL'
'0094  7C         0099          LD    A,H
'0095  FE03       0100          CP    03             ;IS UPPER BYTE 03.
'0097  CAB500'    0101          JP    Z,SNDFL        ;IF IT IS ,SET FLAG.
'009A  C3AC00'    0102          JP    NOSND          ;IF NOT,EXIT ROUTINE
'009D  37         0103 NOTBE:   SCF                  ;CLEAR CARRY
'009E  3F         0104          CCF
'009F  ED52       0105          SBC   HL,DE
'00A1  7C         0106          LD    A,H
'00A2  FEFF       0107          CP    0FFH           ;SEE IF DIFFERENCE IS
'00A4  C2AC00'    0108          JP    NZ,NOSND       ;FFC0H (2'S COMPLEMENT
'00A7  FEC0       0109          CP    0C0H           ;OF 0040H).
'00A9  CAB500'    0110          JP    Z,SNDFL        ;IF IT IS,SET FLAG
'00AC  E1         0111 NOSND:   POP   HL
'00AD  3E00       0112          LD    A,00
'00AF  D302       0113          OUT   (P1B),A        ;SET D/A OUTPUT TO 0
'00B1  D9         0114          EXX                  ;EXCHANGE REG. SETS
'00B2  08         0115          EX    AF,AF'
'00B3  ED45       0116          RETN
'00B5  E1         0117 SNDFL:   POP   HL             ;SET FLAG ROUTINE
```

```
     DATA MANIPULATION PROGRAM
 ADDR  OBJECT     ST #

'ØØB6  ØEØ1       Ø118 SNDF1: LD   C,Ø1          ;SET OUTPUT FLAG
'ØØB8  Ø6ØØ       Ø119        LD   B,ØØ          ;CLEAR B'
'ØØBA  D5         Ø12Ø        PUSH DE
'ØØBB  E5         Ø121        PUSH HL
'ØØBC  114ØØØ     Ø122        LD   DE,64
'ØØBF  19         Ø123        ADD  HL,DE         ;LOAD (END) WITH
'ØØCØ  22Ø213     Ø124        LD   (END),HL      ;END ADDRESS OF SECTOR
'ØØC3  E1         Ø125        POP  HL
'ØØC4  D1         Ø126        POP  DE
'ØØC5  C3FAØØ'    Ø127        JP   MARK
'ØØC8  Ø6ØØ       Ø128 SEND1: LD   B,ØØ          ;CLEAR B
'ØØCA  7D         Ø129        LD   A,L
'ØØCB  E6ØF       Ø13Ø        AND  ØFH
'ØØCD  FEØØ       Ø131        CP   ØØ            ;SEE IF Ø ENERGY POS.
'ØØCF  C2D7ØØ'    Ø132        JP   NZ,SEND2
'ØØD2  3E7F       Ø133 SEND6: LD   A,7FH         ;OUTPUT MAX LEVEL
'ØØD4  C3DAØØ'    Ø134        JP   SEND3
'ØØD7  7E         Ø135 SEND2: LD   A,(HL)        ;PUT ACCUMULATION IN A
'ØØD8  36ØØ       Ø136        LD   (HL),ØØ       ;CLEAR TO ZERO
'ØØDA  D3Ø2       Ø137 SEND3: OUT  (PIB),A       ;OUTPUT TO D/A
'ØØDC  23         Ø138        INC  HL
'ØØDD  3A2Ø13     Ø139        LD   A,(END)       ;SEE IF END OF SECTOR
'ØØEØ  BD         Ø14Ø        CP   L
'ØØE1  C2F6ØØ'    Ø141        JP   NZ,SEND5
'ØØE4  3A2113     Ø142        LD   A,(END+1)
'ØØE7  BC         Ø143        CP   H
'ØØE8  C2F6ØØ'    Ø144        JP   NZ,SEND5
'ØØEB  3EØC       Ø145        LD   A,ØCH         ;SEE IF ALL 16 SECTORS
'ØØED  BC         Ø146        CP   H             ;HAVE BEEN SENT.
'ØØEE  C2F4ØØ'    Ø147        JP   NZ,SEND4
'ØØF1  21ØØØ8     Ø148        LD   HL,8ØØH       ;RESET TO BEGINNING
'ØØF4  ØEØØ       Ø149 SEND4: LD   C,ØØ          ;RESET OUTPUT FLAG
'ØØF6  D9         Ø15Ø SEND5: EXX                ;EXCHANGE REGISTERS
'ØØF7  Ø8         Ø151        EX   AF,AF'
'ØØF8  ED45       Ø152        RETN
'ØØFA  7C         Ø153 MARK:  LD   A,H           ;MARKER SIGNAL ROUTINE
'ØØFB  E6Ø3       Ø154        AND  ØØØØØØ11B     ;EXTRACT SECTOR NUMBER
'ØØFD  85         Ø155        ADD  A,L           ;FROM HL'
'ØØFE  ØF         Ø156        RRCA               ;SHIFT TO RIGHT POS.
'ØØFF  ØF         Ø157        RRCA
'Ø1ØØ  CB3F       Ø158        SRL  A
'Ø1Ø2  D3Ø2       Ø159        OUT  (PIB),A       ;OUTPUT TO D/A
'Ø1Ø4  D9         Ø16Ø        EXX                ;EXCHANGE REGISTERS
'Ø1Ø5  Ø8         Ø161        EX   AF,AF'
'Ø1Ø6  ED45       Ø162        RETN
                  Ø163        END

ERRORS=ØØØØ
ERRORS=ØØØØ
.
```

```
     EPROM PROGRAMMER
 ADDR   OBJECT     ST #

                   ØØØ2              ORG   4ØØØH
'4ØØØ   315ØFF     ØØØ3              LD    SP,ØFF5ØH         ;STACK POINTER ADDR
'4ØØ3   3ECF       ØØØ4              LD    A,ØCFH            ;MODE 3 CONTROL WORD
'4ØØ5   D3D7       ØØØ5              OUT   (ØD7H),A          ;PIO#2 CONTROL
'4ØØ7   3EØØ       ØØØ6              LD    A,ØØ
'4ØØ9   D3D7       ØØØ7              OUT   (ØD7H),A          ;DEFINE  TO BE OUTPUT
'4ØØB   3ECF       ØØØ8 BGN:         LD    A,ØCFH            ;MODE 3 CONTROL WORD
'4ØØD   D3D5       ØØØ9              OUT   (ØD5H),A          ;PIO#2 PORT B CONTROL
'4ØØF   3EFF       ØØ1Ø              LD    A,ØFFH
'4Ø11   D3D5       ØØ11              OUT   (ØD5H),A          ;DEFINE TO BE INPUT
'4Ø13   3EAE       ØØ12              LD    A,1Ø1Ø111ØB       ;RAISE 4Ø4Ø RST,
'4Ø15   D3D6       ØØ13              OUT   (ØD6H),A          ;DISABLE BUFFERS
'4Ø17   1EØØ       ØØ14 PT:          LD    E,Ø
'4Ø19   217442'    ØØ15              LD    HL,MESGØ          ;OUTPUT ON TTY TO
'4Ø1C   CDC7E3     ØØ16              CALL  PTXT              ;USER. ASK TO DEFINE
'4Ø1F   CD9CE5     ØØ17              CALL  CRLF              ;27Ø8 OR 2716. DDT-8Ø
'4Ø22   CD97E5     ØØ18              CALL  ECHO              ;ROUTINES FOR TTY I/O
'4Ø25   42         ØØ19              LD    B,D
'4Ø26   CD9CE5     ØØ2Ø              CALL  CRLF
'4Ø29   78         ØØ21              LD    A,B               ;USER SPECIFIED
'4Ø2A   FE41       ØØ22              CP    'A'               ;27Ø8 OR 2716?
'4Ø2C   CA4841'    ØØ23              JP    Z,X27Ø8
'4Ø2F   FE42       ØØ24              CP    'B'
'4Ø31   CA374Ø'    ØØ25              JP    Z,X2716
'4Ø34   C3174Ø'    ØØ26              JP    PT                ;REPEAT IF WRONG CHAR
'4Ø37   1EØØ       ØØ27 X2716:       LD    E,Ø
'4Ø39   219Ø42'    ØØ28              LD    HL,MESG1          ;ASK WHETHER TO READ
'4Ø3C   CDC7E3     ØØ29              CALL  PTXT              ;OR PROGRAM.
'4Ø3F   CD9CE5     ØØ3Ø              CALL  CRLF
'4Ø42   CD97E5     ØØ31              CALL  ECHO
'4Ø45   42         ØØ32              LD    B,D
'4Ø46   CD9CE5     ØØ33              CALL  CRLF
'4Ø49   78         ØØ34              LD    A,B
'4Ø4A   FE5Ø       ØØ35              CP    'P'               ;USER SPECIFIED
'4Ø4C   CA794Ø'    ØØ36              JP    Z,PROG            ;READ OR PROGRAM?
'4Ø4F   FE52       ØØ37              CP    'R'
'4Ø51   CA574Ø'    ØØ38              JP    Z,READ
'4Ø54   C3374Ø'    ØØ39              JP    X2716             ;REPEAT IF WRONG CHAR
'4Ø57   11ØØØ8     ØØ4Ø READ:        LD    DE,8ØØH           ;NUMBER OF BYTES
'4Ø5A   21ØØØØ     ØØ41              LD    HL,ØØØØ           ;STARTING ADDRESS
'4Ø5D   3EE4       ØØ42              LD    A,111ØØ1ØØB       ;LOWER 4Ø4Ø RST AND
'4Ø5F   D3D6       ØØ43              OUT   (ØD6H),A          ;2716 CS.
'4Ø61   DBD4       ØØ44 RE1:         IN    A,(ØD4H)          ;INPUT DATA FROM 2716
'4Ø63   77         ØØ45              LD    (HL),A            ;STORE AT ADDR (HL)
'4Ø64   23         ØØ46              INC   HL
'4Ø65   3EEØ       ØØ47              LD    A,111ØØØØØB       ;INCREMENT 4Ø4Ø
'4Ø67   D3D6       ØØ48              OUT   (ØD6H),A
'4Ø69   3EE4       ØØ49              LD    A,111ØØ1ØØB
'4Ø6B   D3D6       ØØ5Ø              OUT   (ØD6H),A
'4Ø6D   E5         ØØ51              PUSH  HL
'4Ø6E   37         ØØ52              SCF                     ;CLEAR CARRY
'4Ø6F   3F         ØØ53              CCF
'4Ø7Ø   ED52       ØØ54              SBC   HL,DE             ;SEE IF (HL)=8ØØH
'4Ø72   CA5542'    ØØ55              JP    Z,QUIT            ;QUIT IF IT DOES
'4Ø75   E1         ØØ56              POP   HL
'4Ø76   C3614Ø'    ØØ57              JP    RE1               ;CONTINUE OTHERWISE
'4Ø79   11ØØØ8     ØØ58 PROG:        LD    DE,8ØØH
'4Ø7C   21ØØØØ     ØØ59              LD    HL,ØØØØ
```

```
     EPROM PROGRAMMER
 ADDR   OBJECT     ST #

'407F   3EE4       Ø06Ø            LD    A,111ØØ1ØØB    ;ENABLE INPUT BUFFERS
'4Ø81   D3D6       ØØ61            OUT   (ØD6H),A       ;LOWER 2716 CS
'4Ø83   DBD4       ØØ62 PR2:       IN    A,(ØD4H)       ;INPUT TO SEE IF
'4Ø85   FEFF       ØØ63            CP    ØFFH           ;IT IS ERASED
'4Ø87   C2BØ41'    ØØ64            JP    NZ,PG6         ;IF NOT, TELL USER.
'4Ø8A   23         ØØ65            INC   HL
'4Ø8B   3EBØ       ØØ66            LD    A,1Ø11ØØØØB    ;INCREMENT 4Ø4Ø
'4Ø8D   D3D6       ØØ67            OUT   (ØD6H),A
'4Ø8F   3EB4       ØØ68            LD    A,1Ø11Ø1ØØB
'4Ø91   D3D6       ØØ69            OUT   (ØD6H),A
'4Ø93   E5         ØØ7Ø            PUSH  HL
'4Ø94   37         ØØ71            SCF                  ;CLEAR CARRY
'4Ø95   3F         ØØ72            CCF
'4Ø96   ED52       ØØ73            SBC   HL,DE          ;SEE IF (HL)=82ØH
'4Ø98   CA9F4Ø'    ØØ74            JP    Z,PR3          ;STOP IF IT DOES
'4Ø9B   E1         ØØ75            POP   HL
'4Ø9C   C3834Ø'    ØØ76            JP    PR2            ;OTHERWISE CONTINUE
'4Ø9F   E1         ØØ77 PR3:       POP   HL
'4ØAØ   3EAE       ØØ78            LD    A,1Ø1Ø111ØB    ;DISABLE BUFFERS, †
'4ØA2   D3D6       ØØ79            OUT   (ØD6H),A       ;2716 CS AND 4Ø4Ø RST
'4ØA4   3ECF       ØØ8Ø            LD    A,ØCFH
'4ØA6   D3D5       ØØ81            OUT   (ØD5H),A
'4ØA8   3EØØ       ØØ82            LD    A,ØØ           ;CHANGE PORT A TO
'4ØAA   D3D5       ØØ83            OUT   (ØD5H),A       ;OUTPUT PORT.
'4ØAC   3EA7       ØØ84            LD    A,1Ø1ØØ111B    ;ENABLE OUTPUT BUFFER
'4ØAE   D3D6       ØØ85            OUT   (ØD6H),A       ;LOWER 4Ø4Ø RST.
'4ØBØ   1EØØ       ØØ86            LD    E,Ø
'4ØB2   21AD42'    ØØ87            LD    HL,MESG2       ;TELL USER TO TURN ON
'4ØB5   CDC7E3     ØØ88            CALL  PTXT           ;PROGRAM ENABLE
'4ØB8   CD9CE5     ØØ89            CALL  CRLF           ;SWITCHES.
'4ØBB   CD22E5     ØØ9Ø            CALL  RDCHR          ;WAIT FOR AN INPUT
'4ØBE   CD9CE5     ØØ91            CALL  CRLF           ;FROM USER.
'4ØC1   11ØØØ8     ØØ92            LD    DE,8ØØH
'4ØC4   21ØØØØ     ØØ93            LD    HL,ØØØØ
'4ØC7   7E         ØØ94 PR1:       LD    A,(HL)         ;OUTPUT DATA ON PORTA
'4ØC8   D3D4       ØØ95            OUT   (ØD4H),A
'4ØCA   3E37       ØØ96            LD    A,ØØ11Ø111B    ;SET UP CTC FOR 5Ø MS
'4ØCC   D3DA       ØØ97            OUT   (ØDAH),A       ;INTERVAL. HAVE TO
'4ØCE   3EFØ       ØØ98            LD    A,24Ø          ;USE TWO CHANNELS.
'4ØDØ   D3DA       ØØ99            OUT   (ØDAH),A
'4ØD2   3E3F       Ø1ØØ            LD    A,ØØ111111B
'4ØD4   D3DB       Ø1Ø1            OUT   (ØDBH),A
'4ØD6   3EE8       Ø1Ø2            LD    A,232
'4ØD8   D3DB       Ø1Ø3            OUT   (ØDBH),A
'4ØDA   3E27       Ø1Ø4            LD    A,ØØ1ØØ111B    ;RAISE PD/PGM LINE OF
'4ØDC   D3D6       Ø1Ø5            OUT   (ØD6H),A       ;2716.
'4ØDE   DBDB       Ø1Ø6 PULSL:     IN    A,(ØDBH)       ;WAIT FOR CTC TO
'4ØEØ   FEØ1       Ø1Ø7            CP    Ø1             ;COUNT DOWN.
'4ØE2   C2DE4Ø'    Ø1Ø8            JP    NZ,PULSE
'4ØE5   3EA7       Ø1Ø9            LD    A,1Ø1ØØ111B    ;LOWER 2716 PD/PGM
'4ØE7   D3D6       Ø11Ø            OUT   (ØD6H),A
'4ØE9   3EA6       Ø111            LD    A,1Ø1ØØ11ØB    ;DISABLE BUFFERS
'4ØEB   D3D6       Ø112            OUT   (ØD6H),A
'4ØED   3ECF       Ø113            LD    A,ØCFH         ;CHANGE PORT A TO
'4ØEF   D3D5       Ø114            OUT   (ØD5H),A       ;AN INPUT PORT.
'4ØF1   3EFF       Ø115            LD    A,ØFFH
'4ØF3   D3D5       Ø116            OUT   (ØD5H),A
'4ØF5   3EA4       Ø117            LD    A,1Ø1ØØ1ØØB    ;ENABLE INPUT BUFFERS
```

```
     EPROM PROGRAMMER
 ADDR  OBJECT

'40F7  D3D6                  OUT  (ØD6H),A
'40F9  3EE4                  LD   A,11100100B   ;LOWER 2716 CS
'40FB  D3D6                  OUT  (ØD6H),A
'40FD  DBD4                  IN   A,(ØD4H)      ;INPUT DATA & VERIFY
'40FF  BE                    CP   (HL)          ;IT IS CORRECT.
'4100  C22841'               JP   NZ,ERROR
'4103  3EA4                  LD   A,10100100B   ;RAISE 2716 CS
'4105  D3D6                  OUT  (ØD6H),A
'4107  3EA6                  LD   A,10100110B   ;DISABLE BUFFERS
'4109  D3D6                  OUT  (ØD6H),A
'410B  3ECF                  LD   A,ØCFH        ;CHANGE PORT A TO
'410D  D3D5                  OUT  (ØD5H),A      ;OUTPUT PORT.
'410F  3EØØ                  LD   A,ØØ
'4111  D3D5                  OUT  (ØD5H),A
'4113  3EA3                  LD   A,10100011B   ;INCREMENT 4040
'4115  D3D6                  OUT  (ØD6H),A
'4117  3EA7                  LD   A,10100111B
'4119  D3D6                  OUT  (ØD6H),A
'411B  23                    INC  HL
'411C  E5                    PUSH HL
'411D  37                    SCF                ;CLEAR CARRY
'411E  3F                    CCF
'411F  ED52                  SBC  HL,DE         ;SEE IF (HL)=8ØØH
'4121  CA5542'               JP   Z,QUIT        ;QUIT IF IT DOES
'4124  E1                    POP  HL
'4125  C3C74Ø'               JP   PR1           ;OTHERWISE CONTINUE
'4128  3EA4           ERROR: LD   A,10100100B   ;DISABLE BUFFERS
'412A  D3D6                  OUT  (ØD6H),A
'412C  E5                    PUSH HL
'412D  1EØØ                  LD   E,Ø
'412F  21F142'               LD   HL,MESG4      ;TELL USER WHERE
'4132  CDC7E3                CALL PTXT          ;ERROR WAS.
'4135  E1                    POP  HL
'4136  7C                    LD   A,H
'4137  CD8BE5                CALL PACC
'413A  7D                    LD   A,L
'413B  CD8BE5                CALL PACC
'413E  CD9CE5                CALL CRLF
'4141  3EA6                  LD   A,10100110B   ;DISABLE BUFFERS
'4143  D3D6                  OUT  (ØD6H),A
'4145  C31DE1                JP   MNTR          ;RETURN TO DDT-8Ø
'4148  1EØØ          X27Ø8:  LD   E,Ø
'414A  219Ø42'               LD   HL,MESG1      ;ASK USER WHETHER TO
'414D  CDC7E3                CALL PTXT          ;READ OR TO PROGRAM.
'415Ø  CD9CE5                CALL CRLF
'4153  CD97E5                CALL ECHO
'4156  42                    LD   B,D
'4157  CD9CE5                CALL CRLF
'415A  78                    LD   A,B
'415B  FE5Ø                  CP   'P'           ;USER SPECIFIED
'415D  CA8A41'               JP   Z,PG          ;READ OR PROGRAM?
'416Ø  FE52                  CP   'R'
'4162  CA6841'               JP   Z,RD
'4165  C34841'               JP   X27Ø8         ;REPEAT IF WRONG CHAR
'4168  11ØØØ4         RD:    LD   DE,4ØØH       ;# OF BYTES IN 27Ø8
'416B  21ØØØØ                LD   HL,ØØØØ
'416E  3EB4                  LD   A,10110100B   ;LOWER 27Ø8 CS,
'417Ø  D3D6                  OUT  (ØD6H),A      ;ENABLE INPUT BUFFER
```

```
     EPROM PROGRAMMER
 ADDR  OBJECT     ST #

'4172  DBD4       0176 RD1:    IN   A,(0D4H)       ;INPUT DATA FROM 2708
'4174  77         0177         LD   (HL),A         ;STORE AT ADDR (HL)
'4175  23         0178         INC  HL
'4176  3EB0       0179         LD   A,10110000B    ;INCREMENT 4040
'4178  D3D6       0180         OUT  (0D6H),A
'417A  3EB4       0181         LD   A,10110100B
'417C  D3D6       0182         OUT  (0D6H),A
'417E  E5         0183         PUSH HL
'417F  37         0184         SCF                 ;CLEAR CARRY
'4180  3F         0185         CCF
'4181  ED52       0186         SBC  HL,DE          ;SEE IF (HL)=400H
'4183  CA5542'    0187         JP   Z,QUIT         ;QUIT IF IT DOES
'4186  E1         0188         POP  HL
'4187  C37241'    0189         JP   RD1
'418A  110004     0190 PG:     LD   DE,400H
'418D  210000     0191         LD   HL,00
'4190  3EB4       0192         LD   A,10110100B    ;LOWER 2708 CS,
'4192  D3D6       0193         OUT  (0D6H),A       ;ENABLE INPUT BUFFER
'4194  DBD4       0194 PG5:    IN   A,(0D4H)       ;INPUT TO SEE IF
'4196  FEFF       0195         CP   0FFH           ;IT IS ERASED
'4198  C2B041'    0196         JP   NZ,PG6         ;IF NOT,TELL USER
'419B  23         0197         INC  HL
'419C  3EB0       0198         LD   A,10110000B    ;INCREMENT 4040
'419E  D3D6       0199         OUT  (0D6H),A
'41A0  3EB4       0200         LD   A,10110100B
'41A2  D3D6       0201         OUT  (0D6H),A
'41A4  E5         0202         PUSH HL
'41A5  37         0203         SCF                 ;CLEAR CARRY
'41A6  3F         0204         CCF
'41A7  ED52       0205         SBC  HL,DE          ;SEE IF (HL)=400H
'41A9  CAC241'    0206         JP   Z,PG7          ;IF SO,GO TO PG7
'41AC  E1         0207         POP  HL
'41AD  C39441'    0208         JP   PG5            ;OTHERWISE CONTINUE
'41B0  1E00       0209 PG6:    LD   E,0
'41B2  21FB42'    0210         LD   HL,MESG5
'41B5  CDC7E3     0211         CALL PTXT
'41B8  CD9CE5     0212         CALL CRLF
'41BB  3EA6       0213         LD   A,10100110B    ;DISABLE BUFFERS,
'41BD  D3D6       0214         OUT  (0D6H),A       ;RAISE CS
'41BF  C31DE1     0215         JP   INTR
'41C2  E1         0216 PG7:    POP  HL
'41C3  3EAE       0217         LD   A,10101110B    ;DISABLE BUFFERS,
'41C5  D3D6       0218         OUT  (0D6H),A       ;RAISE 4040 RST.
'41C7  3ECF       0219         LD   A,0CFH         ;CHANGE PORT A TO
'41C9  D3D5       0220         OUT  (0D5H),A       ;OUTPUT PORT.
'41CB  3E00       0221         LD   A,00
'41CD  D3D5       0222         OUT  (0D5H),A
'41CF  1E00       0223         LD   E,0
'41D1  21AD42'    0224         LD   HL,MESG2       ;TELL USER TO TURN
'41D4  CDC7E3     0225         CALL PTXT           ;ON PROGRAM ENABLE
'41D7  CD9CE5     0226         CALL CRLF           ;SWITCHES.
'41DA  CD22E5     0227         CALL RDCHR          ;WAIT FOR INPUT
'41DD  CD9CE5     0228         CALL CRLF           ;FROM USER.
'41E0  110004     0229         LD   DE,400H
'41E3  0E64       0230         LD   C,100          ;C IS LOOP COUNTER
'41E5  210000     0231 PG1:    LD   HL,00
'41E8  3EA7       0232         LD   A,10100111B    ;LOWER 4040 RST,
'41EA  D3D6       0233         OUT  (0D6H),A       ;ENABLE OUTPUT BUFFER
```

```
     EPROM PROGRAMMER
 ADDR   OBJECT      ST #

'41EC   7E          0234 PG2:    LD    A,(HL)          ;OUTPUT DATA ON
'41ED   D3D4        0235         OUT   (0D4H),A        ;PORT A.
'41EF   3E17        0236         LD    A,00010111B     ;SET UP CTC FOR 1 MS
'41F1   D3DA        0237         OUT   (0DAH),A        ;INTERVAL.
'41F3   3E97        0238         LD    A,151
'41F5   D3DA        0239         OUT   (0DAH),A
'41F7   3E87        0240         LD    A,10000111B     ;RAISE 2708 PGM LINE
'41F9   D3D6        0241         OUT   (0D6H),A
'41FB   DBDA        0242 PG3:    IN    A,(0DAH)        ;WAIT FOR CTC TO
'41FD   FE01        0243         CP    01              ;COUNT DOWN.
'41FF   C2FB41'     0244         JP    NZ,PG3
'4202   3EA7        0245         LD    A,10100111B     ;LOWER 2708 PGM LINE
'4204   D3D6        0246         OUT   (0D6H),A
'4206   3EA3        0247         LD    A,10100011B     ;INCREMENT 4040
'4208   D3D6        0248         OUT   (0D6H),A
'420A   3EA7        0249         LD    A,10100111B
'420C   D3D6        0250         OUT   (0D6H),A
'420E   23          0251         INC   HL
'420F   E5          0252         PUSH  HL
'4210   37          0253         SCF                   ;CLEAR CARRY
'4211   3F          0254         CCF
'4212   ED52        0255         SBC   HL,DE           ;SEE IF (HL)=400H
'4214   CA1B42'     0256         JP    Z,PG4           ;GO TO PG4 IF SO.
'4217   E1          0257         POP   HL
'4218   C3EC41'     0258         JP    PG2             ;OTHERWISE CONTINUE
'421B   E1          0259 PG4:    POP   HL
'421C   0D          0260         DEC   C
'421D   CA2742'     0261         JP    Z,CHK           ;IF 100 LOOPS,STOP
'4220   3EAF        0262         LD    A,10101111B     ;RAISE 4040 RST
'4222   D3D6        0263         OUT   (0D6H),A
'4224   C3E541'     0264         JP    PG1             ;IF <100,CONTINUE
'4227   3EAE        0265 CHK:    LD    A,10101110B     ;DISABLE BUFFERS
'4229   D3D6        0266         OUT   (0D6H),A        ;RAISE 4040_RST.
'422B   3ECF        0267         LD    A,0CFH          ;CHANGE PORT A TO
'422D   D3D5        0268         OUT   (0D5H),A        ;INPUT PORT.
'422F   3EFF        0269         LD    A,0FFH
'4231   D3D5        0270         OUT   (0D5H),A
'4233   210000      0271         LD    HL,00
'4236   3EB4        0272         LD    A,10110100B     ;ENABLE INPUT BUFFER,
'4238   D3D6        0273         OUT   (0D6H),A        ;LOWER 4040 RST & CS.
'423A   DBD4        0274 CHK1:   IN    A,(0D4H)        ;SEE IF CONTENTS OF
'423C   BE          0275         CP    (HL)            ;2708 ARE CORRECT.
'423D   C22841'     0276         JP    NZ,ERROR        ;IF NOT,GO TO ERROR
'4240   23          0277         INC   HL
'4241   E5          0278         PUSH  HL
'4242   37          0279         SCF                   ;CLEAR CARRY
'4243   3F          0280         CCF
'4244   ED52        0281         SBC   HL,DE           ;SEE IF (HL)=400H
'4246   CA5542'     0282         JP    Z,QUIT          ;IF SO QUIT
'4249   E1          0283         POP   HL
'424A   3EB0        0284         LD    A,10110000B     ;INCREMENT 404C
'424C   D3D6        0285         OUT   (0D6H),A
'424E   3EB4        0286         LD    A,10110100B
'4250   D3D6        0287         OUT   (0D6H),A
'4252   C33A42'     0288         JP    CHK1            ;OTHERWISE CONTINUE.
'4255   3EA6        0289 QUIT:   LD    A,10100110B     ;DISABLE BUFFERS
'4257   D3D6        0290         OUT   (0D6H),A
'4259   1E00        0291         LD    E,0
```

```
      EPROM PROGRAMMER
 ADDR   OBJECT      ST #

'425B  21D542'     0292              LD    HL,MESG3          ;ASK USER IF HE WANTS
'425E  CDC7E3      0293              CALL  PTXT              ;TO DO MORE.
'4261  CD9CE5      0294              CALL  CRLF
'4264  CD97E5      0295              CALL  ECHO
'4267  42          0296              LD    B,D
'4268  CD9CE5      0297              CALL  CRLF
'426B  78          0298              LD    A,B
'426C  FE59        0299              CP    'Y'               ;IF YES GO TO BGN
'426E  CA0B40'     0300              JP    Z,BGN
'4271  C31DE1      0301              JP    MNTR              ;OTHERWISE TO DDT-80
'4274  454E5445    0302 MESG0:       DEFM 'ENTER A FOR 2708,B FOR 2716'
'428F  03          0303              DEFB 03                 ;SIGNIFIES END OF MSG
'4290  454E5445    0304 MESG1:       DEFM 'ENTER P TO PROGRAM,R TO READ'
'42AC  03          0305              DEFB 03
'42AD  5455524E    0306 MESG2:       DEFM 'TURN ON PROGRAM ENABLE,THEN HIT A'
'42D4  03          0307              DEFB 03
'42D5  57414E54    0308 MESG3:       DEFM 'WANT TO DO MORE? Y=YES,N=NO'
'42F0  03          0309              DEFB 03
'42F1  4552524F    0310 MESG4:       DEFM 'ERROR AT '
'42FA  03          0311              DEFB 03
'42FB  4E4F5420    0312 MESG5:       DEFM 'NOT ERASED'
'4305  03          0313              DEFB 03
 >E11D             0314 MNTR         EQU   0E11DH            ;DDT-80 ADDRESS
                   0315 ;THE FOLLOWING ARE DDT-80 ROUTINES FOR TTY I/O
 >E3C7             0316 PTXT         EQU   0E3C7H            ;PRINT TEXT
 >E59C             0317 CRLF         EQU   0E59CH            ;RETURN,LINE FEED
 >E597             0318 ECHO         EQU   0E597H            ;INPUT CHAR.,ECHO IT
 >E522             0319 RDCHR        EQU   0E522H            ;READ A CHAR.
 >E58B             0320 PACC         EQU   0E58BH            ;OUTPUT ACCUMULATOR
                   0321              END

ERRORS=0000
ERRORS=0000
.
```

ORIGINAL PAGE IS
OF POOR QUALITY