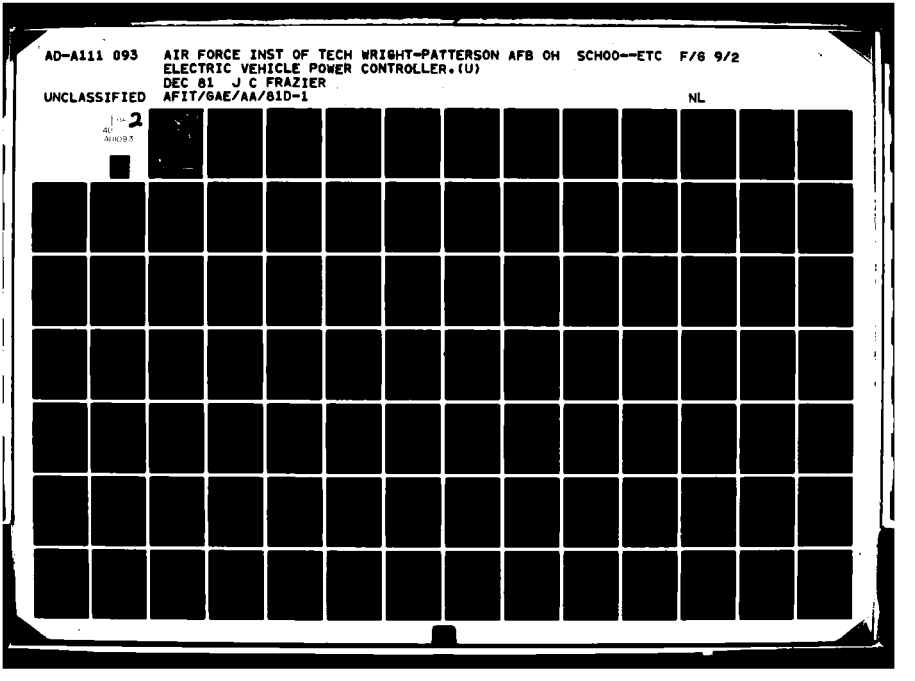
AD-A111 093 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 9/2
ELECTRIC VEHICLE POWER CONTROLLER.(U)
DEC 81 J C FRAZIER
UNCLASSIFIED AFIT/GAE/AA/81D-1 NL

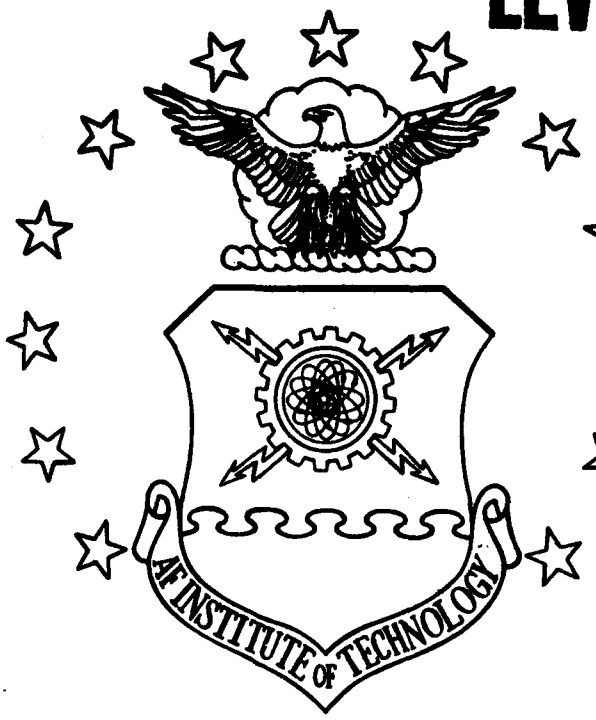


LEVEL II

1

(116p)

AD A111093



DTIC
ELECTE
FEB 18 1982
S D
E

DTIC FILE COPY

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY (ATC)

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

82 02 18 054

012225

AFIT/GE/AA/81D-1

ELECTRIC VEHICLE POWER CONTROLLER

THESIS

AFIT/GE/AA/81D-1 John C. Frazier
Captain USAF

AFIT/GE/AA/81D-1

ELECTRIC VEHICLE POWER CONTROLER

THESIS

Presented to the Faculty of the School of Engineering

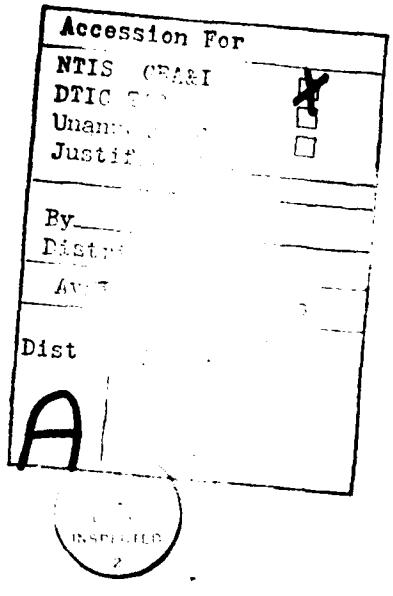
of the Air Force Institute of Technology

Air Training Command

in Partial Fulfillment of the

Requirements for the Degree of

Master of Science



by

John C. Frazier, B.S.

Captain USAF

Graduate Electrical Engineering

December 1981

Approved for public release; distribution unlimited

Preface

The purpose of this project was to design and construct a microcomputer-based control system for an electric powered vehicle. This effort was undertaken to determine the feasibility of a digital control system to produce the basic hardware for use by follow-on efforts.

I am most grateful to my advisors, Captain Aaron DeWispelare and Captain Clark Briggs, for their patience and guidance over the course of this project. I would like to extend special thanks to all of my friends who helped in the final edit of my thesis.

Finally, I want to express my grattitude to my wife, Patricia, for her patience and encouragement throughout this project.

Contents

List of Figures

List of Tables

Abstract

The design of a general purpose microcomputer-based control system for an electric vehicle drive unit is described. This is a research and development system designed with an active operator interface for evaluating performance, dynamic control, parameter modification, and test program interaction. A microcomputer controller processes speed commands and monitors battery energy consumption during the driving cycle. This design effort demonstrated that a unique one-of-a-kind microcomputer controller is easy to construct and interface with the vehicle's systems.

I. Introduction

Background

A great deal of interest has been expressed in electric vehicle transportation during recent years because of increased emphasis on oil conservation. NASA and the United States Air Force (USAF) have made important technological advances in reducing aerospace systems' fuel consumption and have begun complementary projects to develop economic alternatives for gas-powered vehicles. In the mid 1970s, replacement of various small conventional vehicles with electric-driven equivalents became an item of interest with Air Force Logistics Command (Ref. 1). In July 1980, the USAF became formally involved in an electric vehicle demonstration program through a Department of Energy Interagency Agreement (Ref. 2). This agreement committed the Air Force to procure and test fifteen electric vehicles which were to be distributed to three different locations for a period of four years. The electric vehicles in this test program do not use any type of active control to monitor or optimize energy consumption.

The Air Force Institute of Technology, Aeronautical Engineering Department, has constructed an electric vehicle which is used as a test bed for improved electric vehicle propulsion system concepts (Ref. 3). The vehicle uses the battery switching technique which is sufficient for rudimentary control of the propulsion system. This configuration is simple to design, reliable, and easy to maintain. However, electromechanical control systems may become inadequate as the complexity of the controller increases through the addition of control tasks. Electronic control using a microcomputer control system can provide sophisticated control, monitoring, and warning functions. A general purpose

microcomputer controller has the advantage that new control functions can be added through programming changes and minor interface adjustments.

Objective

The primary objective for this project was to design and validate a microprocessor-based controller (microcontroller) which could be used to regulate the operation of an electric test vehicle propulsion system and collect operating data.

Scope

The scope of this effort was to fabricate a microcontroller, battery switching interface, and appropriate energy monitoring interface hardware. This project also produced the initial operational software to demonstrate the functional capabilities of the controller in the test vehicle.

Approach

The initial phase of the project was to establish a baseline configuration for the test vehicle, develop performance specifications, and construct the control system. The specifications were used to design the control system hardware and application software. The microcontroller was based on the Zilog Z80 microprocessor family and on a monolithic analog data acquisition system (Ref. 4). The operating system was developed by modifying an existing Z80 "ROM Monitor" to use the Z80 serial input/output controller and operate compatibly with the added application software commands. The "ROM Monitor" was used to manage operation of the microcomputer. The monitor's input/output, memory, register, and program control instructions were used to test and debug the microcomputer controller. The control and monitoring interfaces were designed using discrete components. The application software was prepared using a

software development microcomputer. The application software contains drive system control, energy monitoring, and initialization routines.

The second phase of the project was to test each component individually and the system collectively in the laboratory. This testing served two purposes. First, it ensured that each component operated as designed and functioned properly in the system. Second, the tests were used to determine calibration constants for calculations.

The final phase of this project was the installation of the microcomputer controller and interface circuitry in the test vehicle. The vehicle was then given a functional road test to demonstrate the microcomputer control system.

II. Process Controllers

The electric vehicle digital controller is a member of a larger class of process controllers. In this section process controller characteristics and their relationship to digital controllers are described. The various types of digital controllers are reviewed to develop selection criteria for the electric vehicle controller.

General Description

A general process control system contains four basic components. They are measuring elements, a controller, actuators, and supervisory elements (Ref. 5). The measuring elements sense a process property and generate a corresponding output signal. The controller compares the measured signal with a predetermined setpoint and initiates a signal to counteract any deviations. The actuators receive the control signals and adjust the process through changes in valve positions, switch settings, or servo motor rotation. The supervisory elements are used to monitor and implement the operating strategy for the first three components. Supervisory control can vary over a broad range from managing one to many functions of a distributed control system.

Before the development of microprocessors, controllers were implemented using discrete analog circuits. The advent of the microprocessor brought about the development of the digital controller as a replacement for the analog controller. The external connections and adjustments for this controller remained basically unchanged because they are determined by the requirements of the process. A digital controller uses software to interpret the control loop functions at the process interfaces. The digital controller also brought with it increased flexibility in that a

control algorithm could be changed without disturbing the hardware or the interfaces.

Digital Controllers

Microprocessor-based digital controllers have evolved into three basic sizes: the modular, single board, and single chip controllers. These systems have varing degrees of complexity, versatility, and data handling capability. This section investigates the various properties of these digital controllers and some aspects of their design selection.

Complexity. The complexity of a microprocessor controller is measured by its operating and physical parameters. Operating parameters include the number and type of instructions the controller can process, the size of the control tasks it can manage and their number, and its processing speed. Physical parameters include the number of circuit packages needed to make an operational system, and their physical dimensions, power dissipation, and data types processed.

The single chip controller is the simplest and possibly the most elegant of process controllers. It has evolved from the all-in-one microprocessors used for high volume consumer products. These controllers are dedicated to a specific control process at the time of manufacture and are generally the product of a lengthy development effort. Single chip controllers contain the instructions and interface capability to perform, at most, only a few simple control tasks. They are being used as front end control processors, placed as close as possible to the process they manage. To enhance system reliability, these controllers often become a physical part of the device they control. Examples are the new generation fly-by-wire actuators and microwave oven controllers (Ref. 6).

The single board controller is the next step up from the single chip controller. It is a direct outgrowth of the single board microcomputer, and is generally used for simple to moderately complex control tasks. This type of controller has one major advantage over the single chip controllers in that it can be reprogrammed if the need arises. The single board controller is used in widely distributed control systems, as in chemical plants, to monitor and control different phases of a chemical process. Because the controller is based on a general purpose microprocessor, it usually has an elementary operating system which gives the operator greater diagnostic capability (Ref. 7, 8).

The modular controller is the largest and most sophisticated of the microprocessor-based controllers. Its functions are divided into separate modules, and it is capable of managing very complex or high demand control tasks. This controller can be completely adapted to the intended control task, and is generally capable of performing its own software development. Modular controllers are used in large facilities and in laboratory control systems (Ref. 8).

Versatility. Process controller versatility varies significantly with different sizes of controllers. This versatility is related to the type of input/output (I/O) data each controller can process and the number of I/O ports available. Versatility is also measured by the operating system level, resident program storage capability, and reprogrammability.

Single chip controllers are dedicated to perform a specific set of control tasks at the time they are manufactured. If the control functions of a chip or the application for which it is made changes, the chip is discarded and a new controller is built. Input and output data

formats are rigidly specified for each I/O port. Single chip controllers can process digital inputs, and can be equipped with level detectors or analog to digital converters, or both, for analog data input processing. These devices are limited by very small resident memory and typically have only one to two kilobytes of read only memory (ROM) for instructions and a maximum of 256 bytes of random access memory (RAM). Thus, only those instructions essential to a specific design function are programmed, and these devices seldom have the diagnostic capability seen in larger microcomputers.

Single board controllers are manufactured as general purpose systems to be programmed at a later date. These controllers may have a variety of input/output ports for serial, parallel and analog data. They usually have a small monitor operating system that supervises control program execution and offers diagnostic tools such as a debugger. Operating instructions are limited only by available board space for ROM and RAM, and memory chip capacity. The applications software for single board controllers is typically developed on larger, more sophisticated microcomputer systems, and transferred to the single board controller preprogrammed in read only memory.

Modular controllers are the most versatile types and are usually used in development projects to design single chip and board systems. These systems have microcomputers with large on-line memories. They also contain disk storage units and high level disk-based operating systems. Modular controllers can manage large and complicated tasks and process analog and digital control information. These controllers are also generally equipped to develop their own control programs.

Data Handling Requirements. The data handling requirements of a digital controller are determined by the intended application. The requirements specify the accuracy of control variables and allowable error relative to a specific setpoint for stable control. Microprocessors are inherently integer processors, and most are capable of doing 16 bit additions and subtractions at best. Computational throughput is chiefly affected by the complexity of control equations.

Digital control systems often require trade-offs between throughput, accuracy, and the frequency of calculation. Controllers may solve trade-off problems in different ways using both hardware and software. Hardware solutions include using higher speed components, use of special purpose processors for multiplication, fast Fourier transforms, Flash analog to digital converters and multi-processor configurations are also used. Software solutions involve optimizing numerical accuracy, using fixed precision programming, and employing control algorithms to predict the values of control variables.

The single chip system typically uses hardware solutions in conjunction with minimizing control tasks as speed and accuracy become more important. The single board and modular systems use both methods depending on the application. Special processers are not usually available in single board systems due to size limitations of the controller; however, single board system use faster components and more efficient software to meet control requirements. The modular system has the greatest capacity for data processing and the most significant feature being the use of multi-microprocessor architectures to increase the speed of large, multiple input/output systems.

The Intel 8022's high level functional integration provides a single-chip solution to sophisticated, high-volume controller applications that have required multi-chip designs in the past. An example is a controller for a stove having combined microwave and conventional ovens and range-top burners. Twenty keys are used to enter timing and cooking instructions. A four-digit display shows cooking time, temperature, and the time of day. Two temperature sensing thermistors are employed. One for standard oven use and the other for microwave use (Ref. 6).

A single board microprocessor can be used to perform the scanning operation of an acoustic microscope. The controller mechanically moves the mechanism that provides a line scan of an object. Motion through a focused acoustic beam is kept linear by the controller (Ref. 7).

A modular microprocessor controller has been used to operate a centrifugal spectrophotometer that analyzes in real time the chemical reactions of 30 blood samples in parallel and prints out the results. This system is based on the Intel 8080 microprocessor and has the capacity to perform 36 different tests (Ref. 15).

Selection Criteria

In the design of a microprocessor-based control system, a logical and modular approach improves the efficiency of the final design. Figure 1 illustrates a design methodology for designing a digital controller (Ref. 7).

The performance specification is in many ways the most important step of the whole design process. This specification is a clear statement of what processes the system must be able to perform and how the system must respond to external stimuli. The performance specification is used to develop the selection criteria for the design process. Some questions

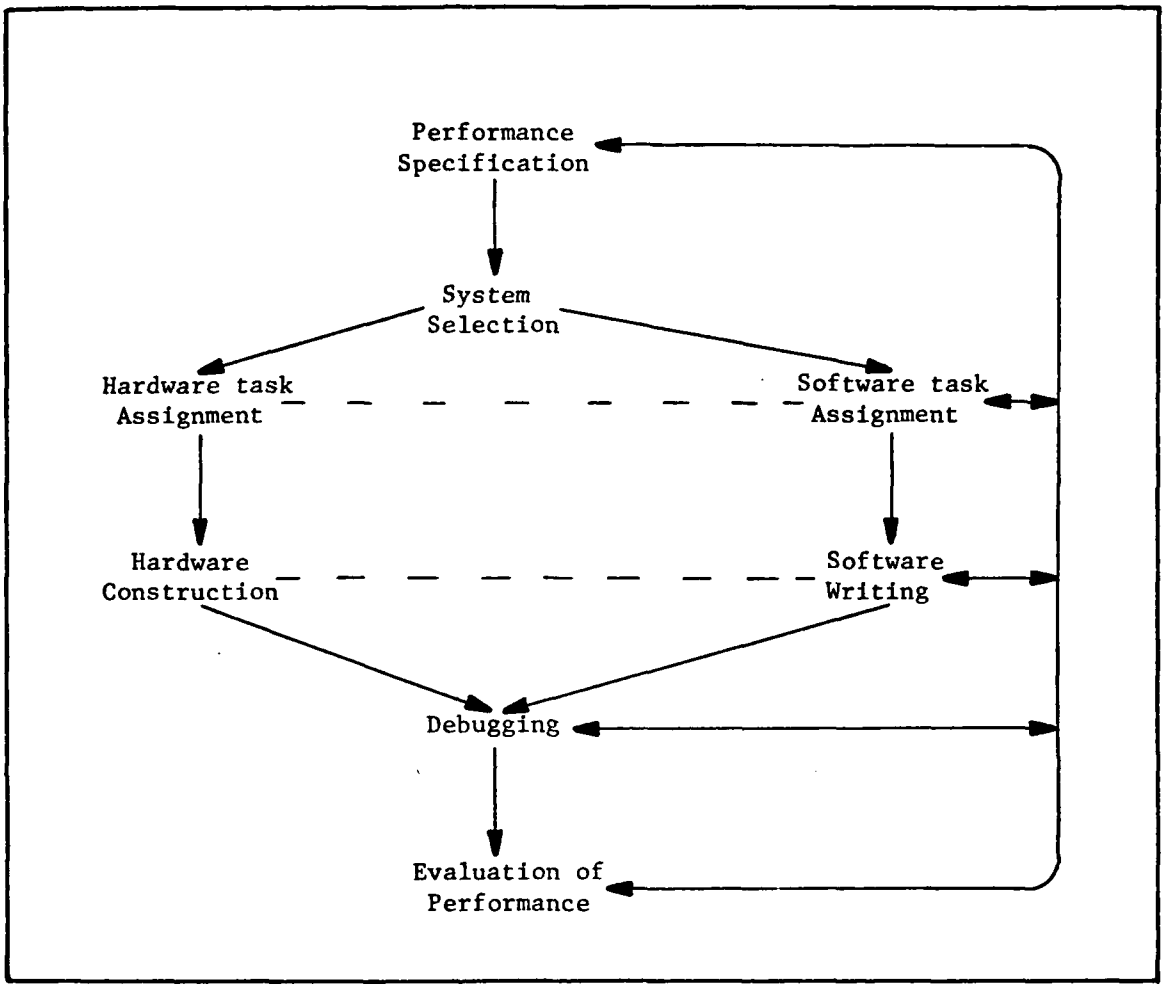


Fig. 1. Design Methodology (Ref. 7)

which are considered when designing a microprocessor-based controller for a specific application are:

1. What type of data exchanges will take place between the microprocessor and the external world? Will the data be in either serial or parallel digital form, or will they be analog signals?

2. What input and output data rates will be required?

3. What data processing and storage will be required? Will calculations be carried out on the data and if so what accuracy and speed will be required?

4. How will the operator interact with the control system? In what ways will the operator be able to influence the systems response to his requests?

5. How will the application software be developed for the controller? This consideration has a large impact on the design constraints associated with cost, power consumption, portability, weight and size.

Additional design constraints associated with the operating environment and noise immunity should also be taken into consideration. The next section addresses these issues in the context of an electric vehicle application.

III. Preliminary Analysis

The initial step in the design of a drive system controller for the electric test vehicle was to become familiar with the existing vehicle configuration. Once understood, this configuration was used to develop functional specifications for new hardware and software designs.

Test Vehicle Description

The test vehicle uses a hybrid propulsion system whereby the drive shafts of an electric motor and a small internal combustion engine are coupled in parallel to the drive train, as shown in Figure 2. In this configuration the speed of a series DC motor is governed by the applied voltage from a battery controller. The controller is a battery switching system which applies voltage to the DC motor in five steps. The internal

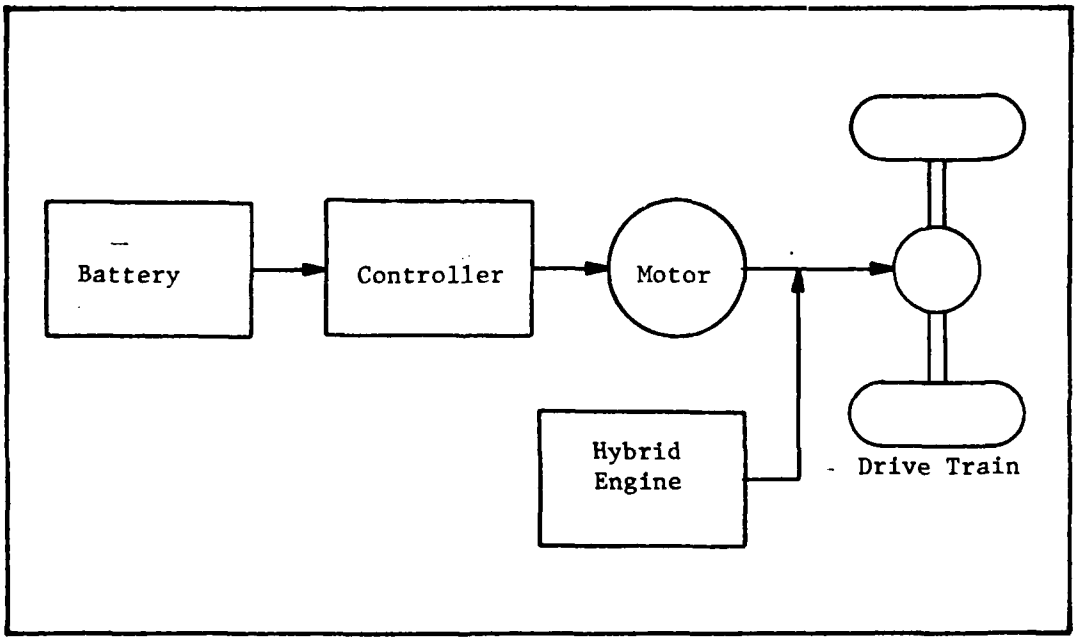


Fig. 2. Parallel Hybrid Vehicle Block Diagram (Ref. 2)

combustion engine is coupled to the drive shaft of the DC motor by a V-belt and electric clutch assembly. The engine is manually engaged during high speed cruising to reduce the current demand of the DC motor (Ref. 3).

Drive System Interfaces

The electric vehicle relies on either manual or electromechanical controls for speed regulation and energy management. The vehicle also has limited instrumentation for monitoring system performance.

Speed Regulation. The battery controller is an electromechanical interface which is linked to the accelerator pedal through a switch block. The switch block converts mechanical movement of the pedal into relay switching patterns for the battery controller. The controller then applies one of five voltages to the series DC motor giving five basic motor speeds.

Hybrid Energy Management. In the hybrid mode the operator is required to manage both the batteries for the electric motor and fuel for the gasoline engine. The engine is manually controlled with a switch to actuate an electric clutch and a throttle lever to control engine speed. The internal combustion engine is normally engaged only at highway speeds. At low speeds the engine cannot provide sufficient torque to propel the vehicle; however, at highway speeds torque requirements are greatly reduced, and adequate power output can be maintained by the engine. When the engine is engaged, it is used to cause the DC motor to overspeed. When this condition occurs, current demand by the motor is reduced and stored energy conserved. If additional power is required, as when climbing a hill, the DC motor compensates by supplying additional energy to the drive-train.

Performance Monitoring. The test vehicle uses direct meter readouts to monitor two DC motor test points and three tachometers. The motor test points measure terminal voltage and current. The current measurement uses a current shunt, rated at 2000 amps per volt, in series with the DC motor. The tachometers are used to measure speeds of the DC motor, engine, and vehicle.

Functional Specifications

Integration of a microcomputer controller into the electric vehicle propulsion system, as shown in Figure 3, requires consideration of present and future functional needs. The new control and monitoring system should perform the existing drive system tasks of speed regulation and energy management. This implies that the controller will manage the linkage between the accelerator pedal and the battery controller relays, and perform most of the energy management task. In order to perform these functions, the controller should be capable of measuring analog signals associated with the accelerator position, DC motor voltage and current, and vehicle speed.

The microcomputer controller should also maintain the expected operating characteristics of the vehicle. That is, it should perform only functional operations requested by the operator. Also, in the event of a computer or control system failure, it should be capable of automatic and manual shut down.

The control system needs to perform four tasks. It should be able to regulate the speed of the DC motor, control the internal combustion engine, keep a running total of battery energy consumed, and report energy consumption as required. The microcomputer should allow real-time operator interaction for entering control system commands. The operator

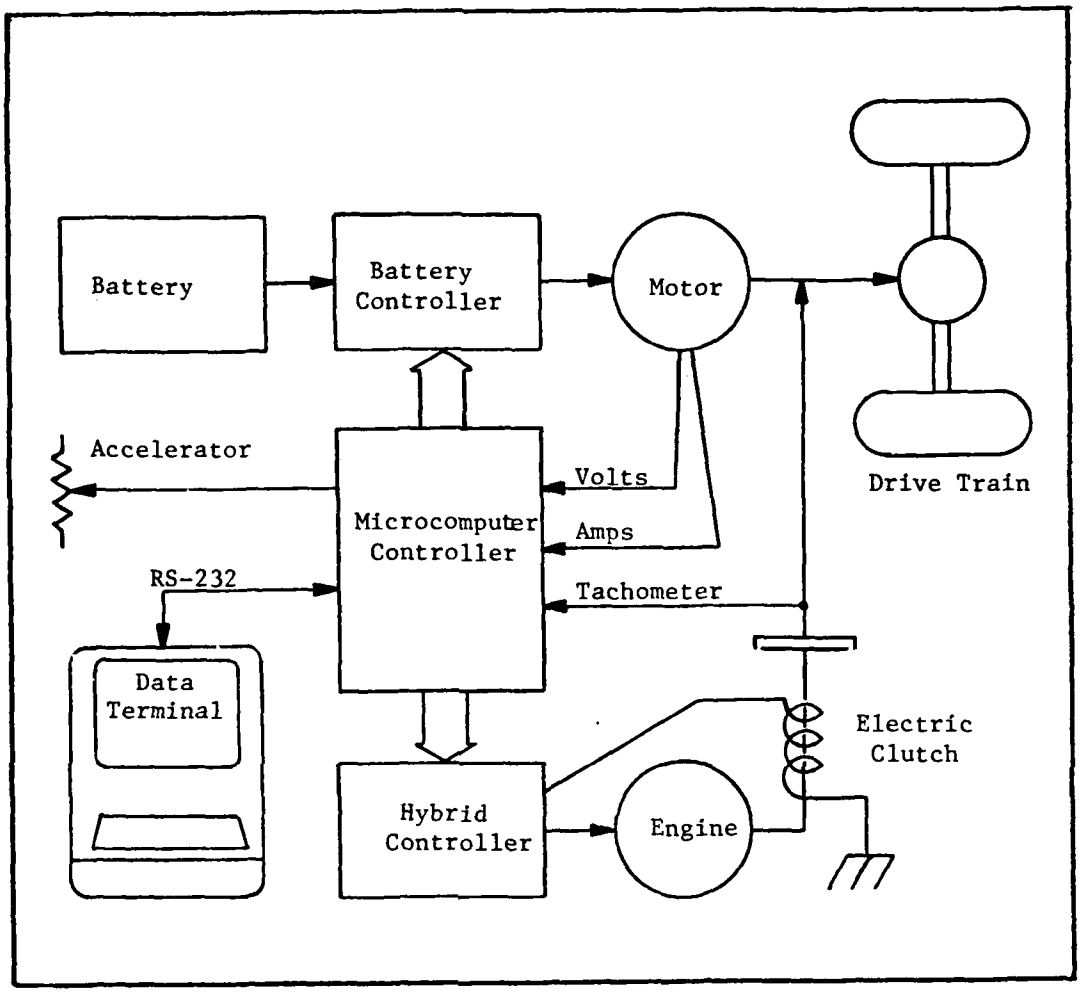


Fig. 3. Microcomputer Controller Parallel Hybrid Vehicle Block Diagram

must also be able to examine and trouble-shoot software and control system problems. Finally, the microcomputer needs to have both serial and parallel input/output ports, and approximately 12-16 kilobytes of on-board memory for operating instructions and temporary storage registers.

Throughput

Throughput is the measure of the maximum number of calculations that are accomplished per unit time. In the case of a control system, it is

defined as the number of control and monitoring tasks which can be accomplished during a sample period. In the case of the electric car controller throughput is influenced by three factors:

1. The ability to interface with a human operator.
2. The complexity of the control and monitoring calculations.
3. The total number of tasks which must be executed during the sample period.

The vehicle operator influences the control system sample period. There is a maximum allowable time delay which can be tolerated between a command at the accelerator and a response by the drive system. This time delay has been experimentally determined to be approximately 0.25 seconds for a natural frequency of 4 Hertz (Ref. 4). In order to achieve stable operation, the digital control system must have a sample rate at least twice its natural frequency. Therefore, a minimum sampling rate of 8 Hertz would be acceptable.

As their complexity increases, the control and monitoring calculations tend to increase the desirable sampling period. This occurs when the execution time of the control and monitoring operations equals or exceeds the time allotted by the sampling period. Computational complexity used in this context refers to the arithmetic operations and numerical precision needed to perform the task. In the case of many common microprocessors, simple routines using single-byte arithmetic typically execute in times less than 100 microseconds. However, typical control and monitoring tasks require multiple-byte arithmetic involving software multiplies. Reed and Mergler describe one microcomputer implementation of a digital position integral derivative (PID) control algorithm which employed extended precision calculations to control a

single plant. Their extended precision calculation required 7 milliseconds to perform (Ref. 10).

Finally, throughput is influenced by a combination of sample period and task complexity. These two factors, when combined, will allow only a finite number of tasks to be accomplished. In a system where the sample period is long and the tasks are reasonable, this problem has little impact. However, as control and monitoring requirements increase, numerical precision and sampling frequency of the task need to be carefully considered.

In this electric vehicle control system there are three tasks which would require periodic updating. They are speed regulation, hybrid energy management, and energy monitoring. Because of the small number of tasks and the relative computational speed of most microprocessors, these tasks can be accomplished sufficiently well at the human high frequency cutoff of 4 Hertz. Then, the initial control system should use a sampling frequency of 8 Hertz or greater.

Summary

The electric vehicle digital control system should be capable of managing multiple tasks during a sampling period. Presently, only four tasks are defined. The initial sample period should be 8 Hertz and the microcomputer should be capable of extended precision arithmetic. Finally, it should provide for operator interaction.

IV. Design and Fabrication

To design a digital control system for a hybrid powered automobile, serious consideration must be given to the automotive functional specifications. The control system requires that the hardware and software be closely related and balanced to optimize controllability. During this research project, the hardware and software were developed concurrently, on a modular basis, to ensure continuity. Each hardware module was designed and fabricated to produce a particular system function. The software was divided and written in the form of functional subroutines. Piecemeal assembly and testing of the hardware and software elements were performed to facilitate system debugging. This approach significantly shortened the construction and testing of the system in the laboratory. The information contained within this chapter addresses the preliminary considerations and full scale development of the control systems' hardware and software.

Preliminary Hardware Considerations/Specifications

Several hardware design factors were considered for the digital control system. Operator interaction with the controller is needed for system monitoring and on-line adjustability to ensure optimal control. To meet this requirement, the microcomputer within the operating system must have a comprehensive command set and a data terminal interface capability. The operating system commands should let the operator look at all the elements in the data path. The operator should have access to monitor the central processing units registers, memory registers, and inputs/output ports. In addition, execution of applicable programs stored in Random Access Memory (RAM) for testing and debugging purposes

ought to be available to the operator. The data terminal interface must present large blocks of information to the operator in a quick and user comprehensible format, and the terminal key board ought to be designed for simplified command and data entry.

A minimum of six input/output ports are required for interaction with the system. Two of these ports are required to be serial input/output ports: one to operate the system data terminal and the other to communicate witn the software development computer. At least four parallel input/output ports are also required: one used by the operating system as a storage register, to control the input/output configuration, two used by the battery controller and engine interfaces, and a discrete input interacting with the analog-digital converter. An additional alterable switch is required by the standard monitor operating system to define the operators initial input/output peripheral equipment.

The microcomputer needs to be self-initializing when it is powered up. This may be accomplished through the use of a hardwired boot-strap operation which supplies the microprocessor with the starting address of the operating system. The microcomputer must contain additional input/output and memory capacity for future control and monitoring functions. Finally, the controller should be configured on a single board as discussed in section II.

Hardware Selection and Design

The microprocessor is the heart of any microcomputer large or small, and it needs to be as versatile and simple as possible. The 8080A and Z80 microprocessors were evaluated as possible candidates for the digital control system. The Z80 microprocessor was selected for the following reasons:

1. Higher degree of compatibility with available equipment.

2. Larger machine language instruction set.

3. The architecture provides one level of interrupt capability within the central processing unit.

4. All timing and control logic controls are contained on the chip.

5. Support chips are synchronized by a single phase clock.

The Z80 is a bus orientated system requiring minimum additional control logic for addressing and reset, as indicated in Figure 4. A Z80 microcomputer also uses a single phase two megahertz system clock to synchronize its operation and a single five volt DC power supply. The serial input/output controller (SIO) has two independently programmable data communication channels. These channels are used to drive the RS-232 interfaces for the operators terminal and the software development computer. The Z80 counter timer circuit (CTC) contains four independently programmable channels, two of which are used as baud rate clocks for the serial ports of the SIO. The two remaining counter timer channels were cascaded to produce a 104 Hertz low frequency clock and a 8 Hertz clock for the control system time reference. Sample period is used to generate interrupt service requests 8 times a second and is used by the control software.

The microcomputer uses three parallel input/output controllers (PIO) giving a total of six 8-bit parallel data ports for use as input or output interfaces. Each PIO port is independently programmable and is TTL-compatible. When used as an output port, the PIO holds its most recent output state providing the hold mechanism for control interfaces.

The microcomputer also employs an analog data acquisition system (ADC 0816) as an integral part of its design to enhance system

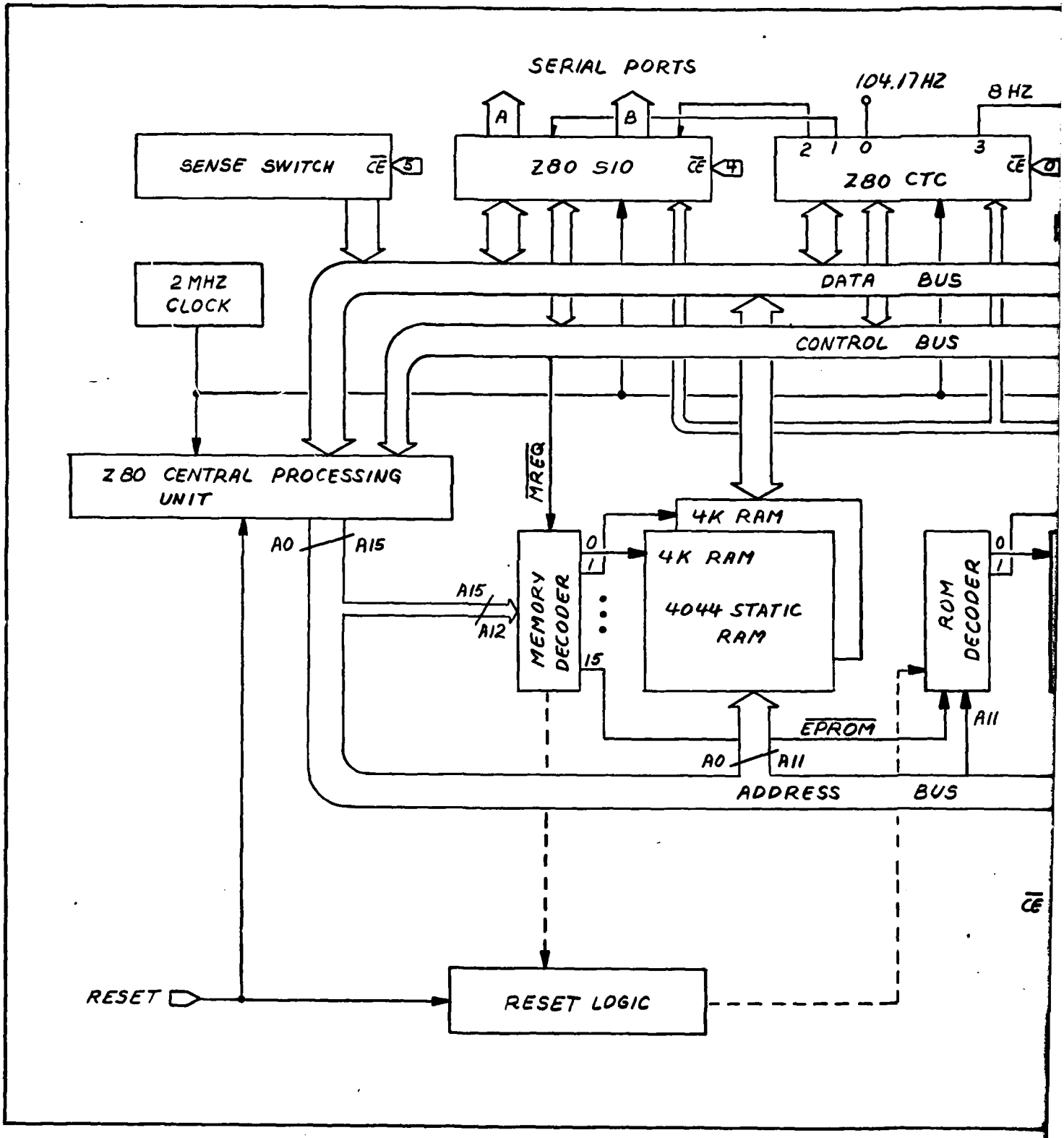
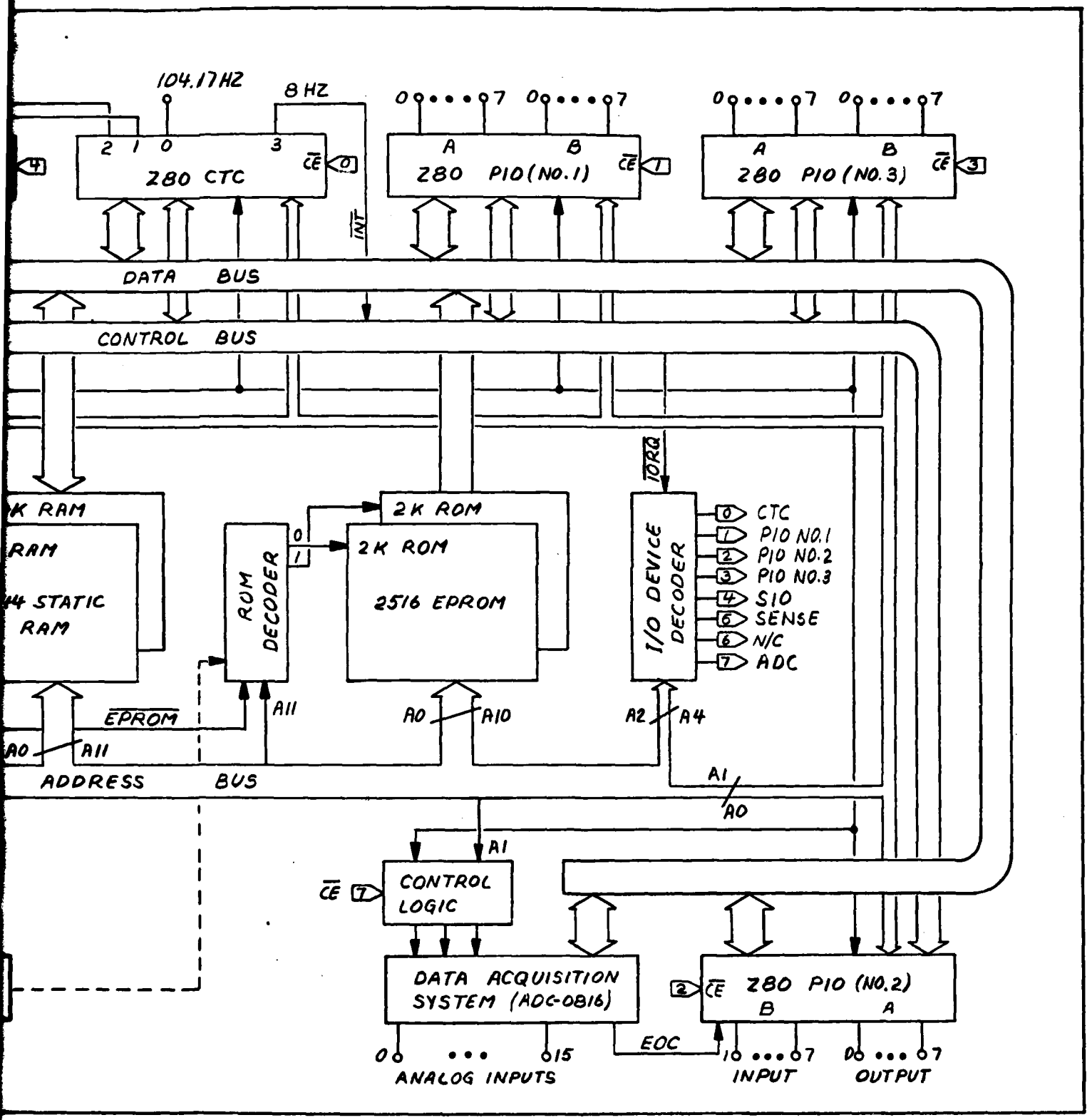


Fig. 4. Microcomputer Block Diagram


104.17HZ
8 HZ
Z80 CTC
Z80 PIO (NO.1)
Z80 PIO (NO.3)
INT
DATA BUS
CONTROL BUS
K RAM
RAM
4 STATIC RAM
ROM DECODER
2K ROM
2K ROM
2516 EPROM
IORQ
I/O DEVICE DECODER
CTC
PIO NO.1
PIO NO.2
PIO NO.3
SIO
SENSE
N/C
ADC
EPROM
A11
A0
A10
A2
A4
A0
A11
ADDRESS BUS
A1
A0
A1
CONTROL LOGIC
DATA ACQUISITION SYSTEM (ADC-0816)
Z80 PIO (NO.2)
EOC
0
15
ANALOG INPUTS
0
7
INPUT
0
7
OUTPUT

flexibility. The analog data acquisition system provides the common functions associated with the collection of analog measurements. It also uses a one Megahertz clock to operate its analog to digital converter and is designed to be interfaced with most microprocessors.

The microcomputer uses two types of memory, EPROM and RAM. The EPROM is used to store the system monitor, application programs, and provides a means of correcting software errors quickly. A single power supply (+5V) EPROM was selected to minimize power source requirements. RAM memory is used for scratch-pad storage and calculations. The RAM is also used during the software development process to hold new programs for testing and debugging.

The low power Schottky TTL components provide the address decoding for the computer memory and input/output devices. They also buffer the Z80 central processing unit and are used in the reset logic that initializes the microcomputer. Appendix A contains detailed circuit information and drawings.

Digital Interfaces

The control system design has two digital interfaces—one for the battery controller and the second for the engine controller. Both of these interfaces translate TTL logic level signals into twelve volt high current signals to drive the inductive loads of relays and motors. These interfaces also provide the power supply, signal ground, and chassis ground isolation for the analog data acquisition system by using optical isolators.

The battery controller uses seven parallel outputs from the microcomputer to derive one of five voltage settings for the DC motor and a power interrupt circuit. Figure 5 shows a simplified circuit diagram

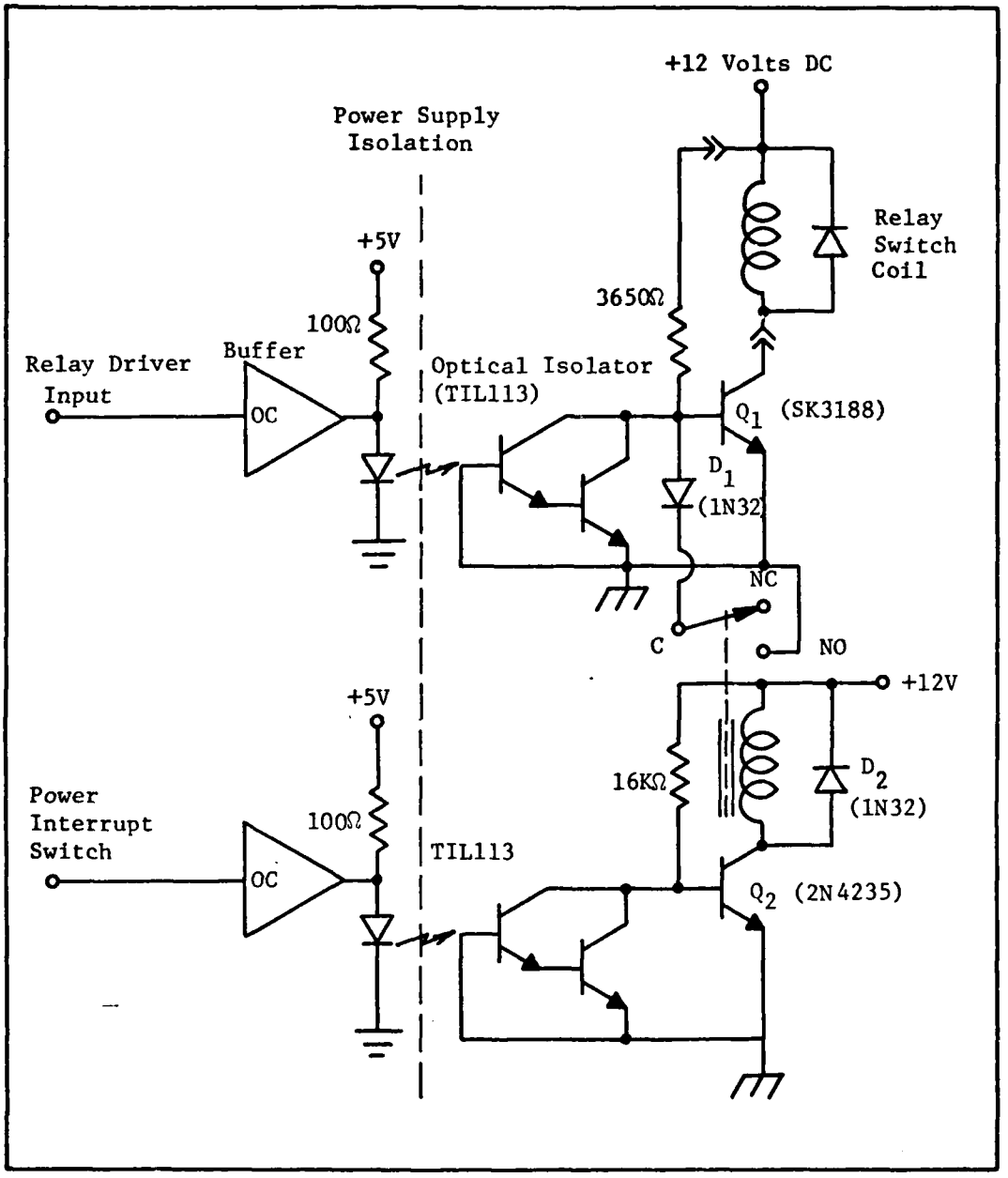


Fig. 5. Battery Controller Showing one Relay Driver and Power Interrupt Logic

of the relay driver interface. The relay driver is operated by applying a logical one to its input to close the relay switch and a logical zero to open it. The power interrupt circuit is activated in two ways. First,

application of a logical one to the power interrupt input connects the base of Q_1 in each of the relay drivers to ground turning them off and opening the battery controller relay switches. Second, a loss of the +5 volt power supply from the microcomputer will turn on Q_2 and interrupt power. The switching transistor (Q_1) must be able to conduct 0.8 to 1.4 amps to close the battery controller relay switch. The current rating of the optical isolator constrains the base drive current of Q_1 to less than 30 milliamps. Bread board testing showed the optimum base current to be 3 to 4 milliamps for 12 volt operation in the above switching current range.

A test circuit was bread boarded using design components, as shown in Figure 5. This test circuit was able to actuate the battery controller relay switch and then deactivate it under simulated power interrupt and command power interrupt conditions. Because of high continuous operating current, the power transistor in the switch driver circuit was heat-sinked to prevent thermal breakdown. A complete circuit diagram for the switching controller is given in Appendix A.

The engine controller was not built in this effort due to mechanical problems with the engine and technical problems with the stepping motor controller. The proposed controller for the engine would manage the throttle and the electric clutch, as indicated in Figure 6. This interface uses a stepping motor to drive the throttle of an internal combustion engine. However, because a stepping motor operates much slower than the microcomputer its interface must be programmable, and provided a low frequency clock. The typical step rate of 1 to 200 steps per second are quoted in most motor specifications. The microcomputer would load the throttle setting into the step counter, set the direction of throttle

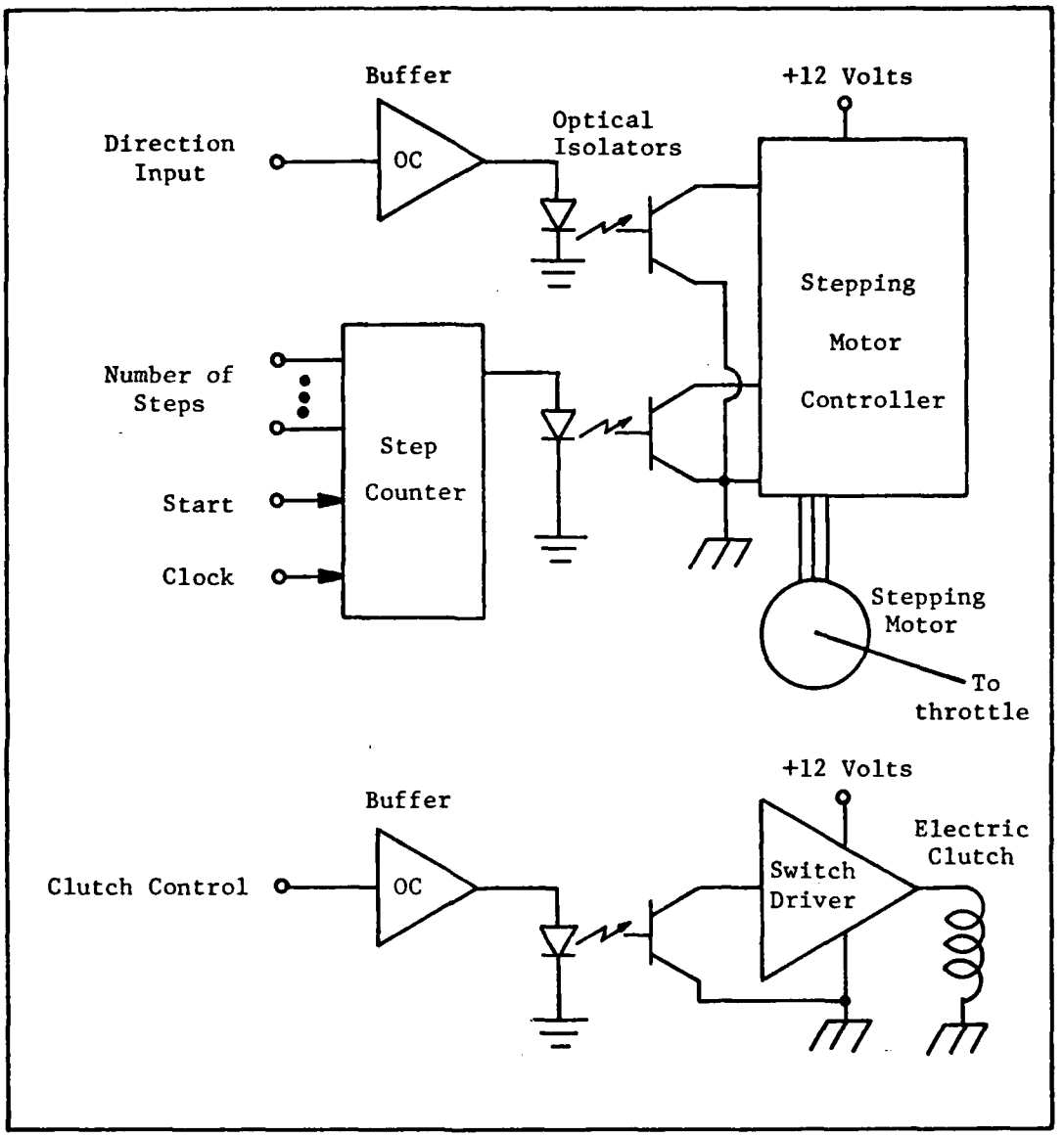


Fig. 6. Function Diagram of Hybrid Controller

movement, start the counter and activate the electric clutch. The 104 Hz clock from the CTC is used to control the stepping speed of the motor and sequence the control circuitry. The electric clutch driver circuit is similar to the relay driver in Figure 5 except for the switching transistor, which is sized for 16 amp steady state service.

Analog Inputs

The microcomputer analog data acquisition system uses an eight bit successive approximation analog to digital (A/D) converter with an input voltage range of 0 to +5 volts dc. The system also has a 16 channel single ended analog multiplexer. In order to make the best possible use of the A/D converter, analog interface circuits scaled the sampled signals to match the converters input voltage range. Three signals were measured as shown in Figure 7 for the demonstration control system—the DC motor terminal voltage, series current, and accelerator position. These signals were referenced to the same ground point due to the input characteristics of the data acquisition system. This single point ground reference required that all of the system's power supplies be referenced to this point to avoid the possibility of a high-current short in the ground plane of the microcomputer. The motor terminal voltage has a dynamic range of 0 to 72 volts DC, requiring that a voltage divider be employed to bring the measured value into the range of the A/D converter. The motor series current is measured by picking off the voltage drop across a calibrated current shunt. The current shunt is rated at 2000 amps per volt, and the peak expected current is 400 amps yielding a voltage range of 0.0 to 0.2 volts. This measurement requires that the sample voltage be amplified to take advantage of the full scale range of the A/D converter. The accelerator position measurement uses a variable resistor with a fixed +5 volt power supply connected as a variable voltage divider. A complete schematic and description of the scaling amplifiers is given in Appendix A.

Preliminary Software Considerations

To support the use of a single board microcomputer controller a

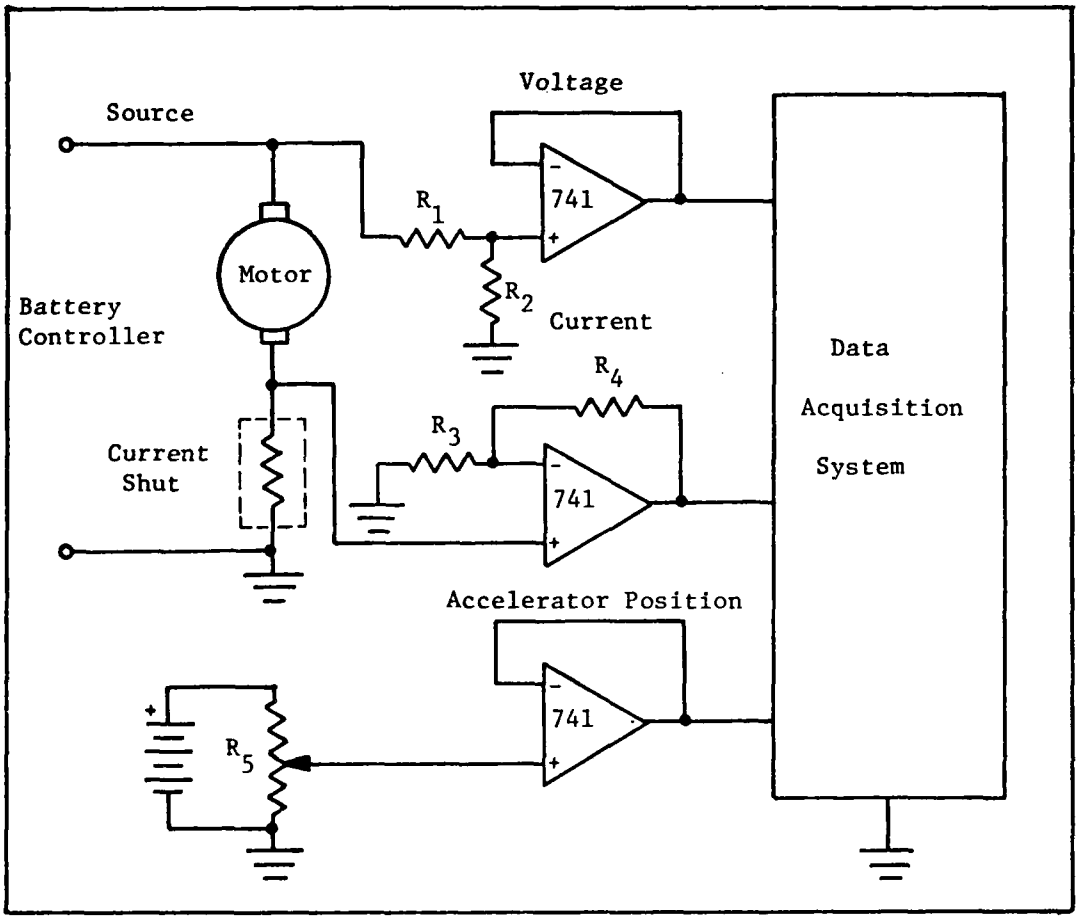


Fig. 7. Simplified Schematic Diagram of Analog Scaling Circuits

method of developing new programs quickly and efficiently was required. Consequently, a second, more sophisticated microcomputer was needed for the software development process. A microcomputer with a disk operating system, screen editor, and macro assembler was selected to satisfy this requirement.

The disk operating system allowed program software to be stored on floppy diskets during software verification. Upon satisfactory performance, the software was programmed into erasable programmable read only memories (EPROM's).

The screen editor was used to write and modify the software programs. The macro assembler was compatible with the controller's machine language to simplify the translation of the screen editor's source code to the object code of the controller.

Additional considerations included provisions for an operating system, a rudimentary debug capability, and a terminal interface. The operating system was required to have file transfer commands and memory loading routines compatible with standard file formats. The debug capability included breakpointing for testing of new programs, and the terminal interface allowed test diagnostics to be reviewed in one screen display.

Control System Software

The control system had two software programs stored in EPROM's. The first program was the standard "ROM Monitor" operating system for the microcomputer. The second program was application software which implemented the control and energy monitoring functions of the control system. The control functions regulated the vehicle and engine speed. The monitoring function kept a running value of battery energy consumed during a driving cycle. Because of the need for positive regulation of the control system by the operator a "Foreground and Background" program execution strategy was used. The monitor operating system cycled in the "foreground" waiting for either an operator input or a sample period interrupt. The application program was executed in the "background" on every interrupt cycle to update control and monitoring tasks. The monitor also contained application program commands used to initiate the driving cycle and report energy consumption to the operator.

Software Development

The software development task was shared between a software development computer and the electric vehicle microcontroller. The software development computer's disk operating system was used to write, assemble, and correct program segments. Thus, the operating system helped facilitate the software development process. The development system could also transfer program modules to the microcontroller where they were tested and debugged. This development effort consisted of three major tasks: modify a standard ROM monitor, develop the application program, and store the 2 programs in EPROM for the microcontroller.

Standard Monitor

The first software task was to modify a standard ROM monitor for Z80 microcomputers. The modification allowed the use of the serial I/O controller and included a command to activate the application program. The serial ports of the microcomputer were used to interface with a CRT terminal and with the software development computer. The CRT terminal was used to access the monitor and application program routines. The software development system's communication link was used to load program modules into the microcomputer's random access memory for testing and debugging.

The modifications of the standard monitor involved changing I/O device addresses to match the microcomputer's hard-wired addresses and providing the necessary initialization for the SIO and CTC. The initialization of the SIO and the CTC used the sequences given in their technical manuals (Ref. 4, 11). Both SIO channels were initialized for RS-232 asynchronous operation. CTC channels 1 and 2 were initialized as the baud rate clocks for the SIO ports. The CRT port was driven at

600 baud and the reader/punch port at 300 baud. The modified monitor was then stored in EPROM memory and installed in the microcomputer. This task needed to be accomplished before the microcomputer could be turned on and tested.

The monitor also needed to be modified to call routines located in the application program. Two user definable keys were used, the I key for the "Ignition" command and the O key for the "Output Energy" command. The Ignition command initialized the controller for driving the vehcile. The Output Energy command sampled the energy accumulator and displayed the energy consumed and the elapsed time of the test on the data terminal. See Appendix B for more detailed information.

Application Program

The application program contained all the routines which interpreted input data to produce the desired output at the control interfaces. These software routines also interfaced with the monitor operating system of the microcomputer. The control routines and part of the performance monitoring routines required continuous updating and were handled in an interrupt service loop. This sample rate was implemented by programming CTC channel 3 to provide a interrupt request 8 times per second.

The application program as shown in Figure 8, can be divided into three parts: initialization, time dependent functions, and energy calculations. The initialization segment was used to initialize the CTC channels 3 and 4, set up the relay driver's PIO port, create a work space in low RAM, and set the CPU to interrupt mode 1. The segment completes initialization by loading the interrupt JUMP VECTOR at location 38H in RAM and enabling interrupts. The monitor "I" command calls the initialization segment thus initiating the control and monitoring routines in

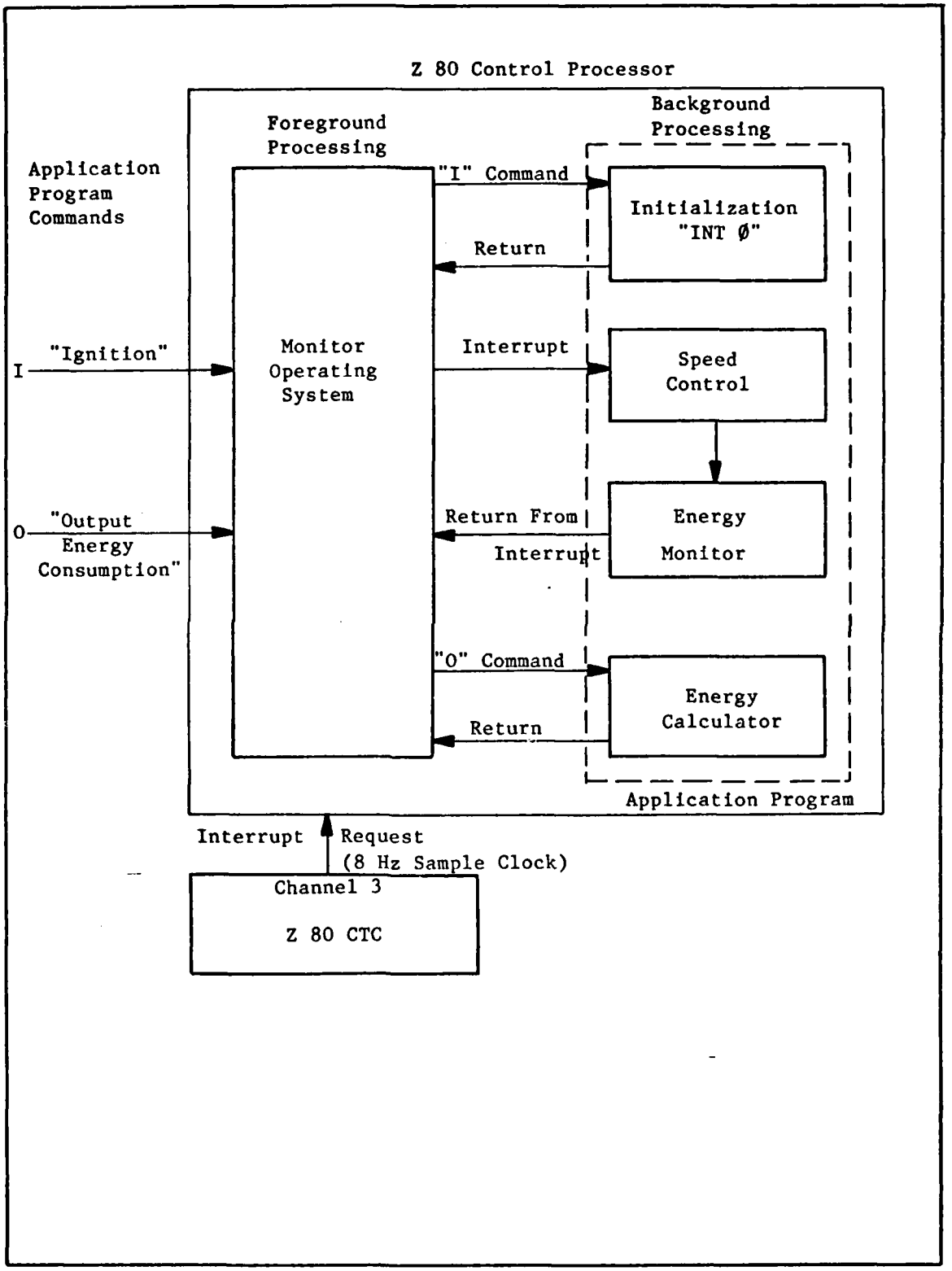


Fig. 8. Software Flow Diagram for the Controll System

the test vehicle. The time dependent functions of the demonstration program represented an interrupt service routine which was executed on every interrupt request. The speed control software, and energy monitor reside in this segment.

Speed control. Speed command was accomplished by measuring the accelerator position and selecting the proper battery switch setting from a table of discretized data. Figure 9 shows the flow chart for this control function. The application program uses a data acquisition system service routine to take these measurements, see Appendix C.

The speed control routine compares the present speed setting to a stored copy of a previous setting to determine whether a change is required. This yields a savings in execution time, prevents relay chatter, and prolongs switch contact life. It was necessary to drive the switches in a particular sequence when changing the setting to avoid power dropouts or surges that would cause bucking or lurching during speed changes. Because of the difference between the speed at which the microcomputer can execute instructions and the inertia of the switches, a wait loop was incorporated between sequence settings to allow the switches sufficient time to change positions before incrementing the sequence.

As an added design feature a standard accelerator set point table was placed in RAM when the system was initialized. This table in RAM was used by the speed control routine to set the battery controller switches. This was done so that the operator could modify the table to improve vehicle operation until the best values were determined.

The set point table contains accelerator position limits and corresponding battery controller switch settings. The table was stored in

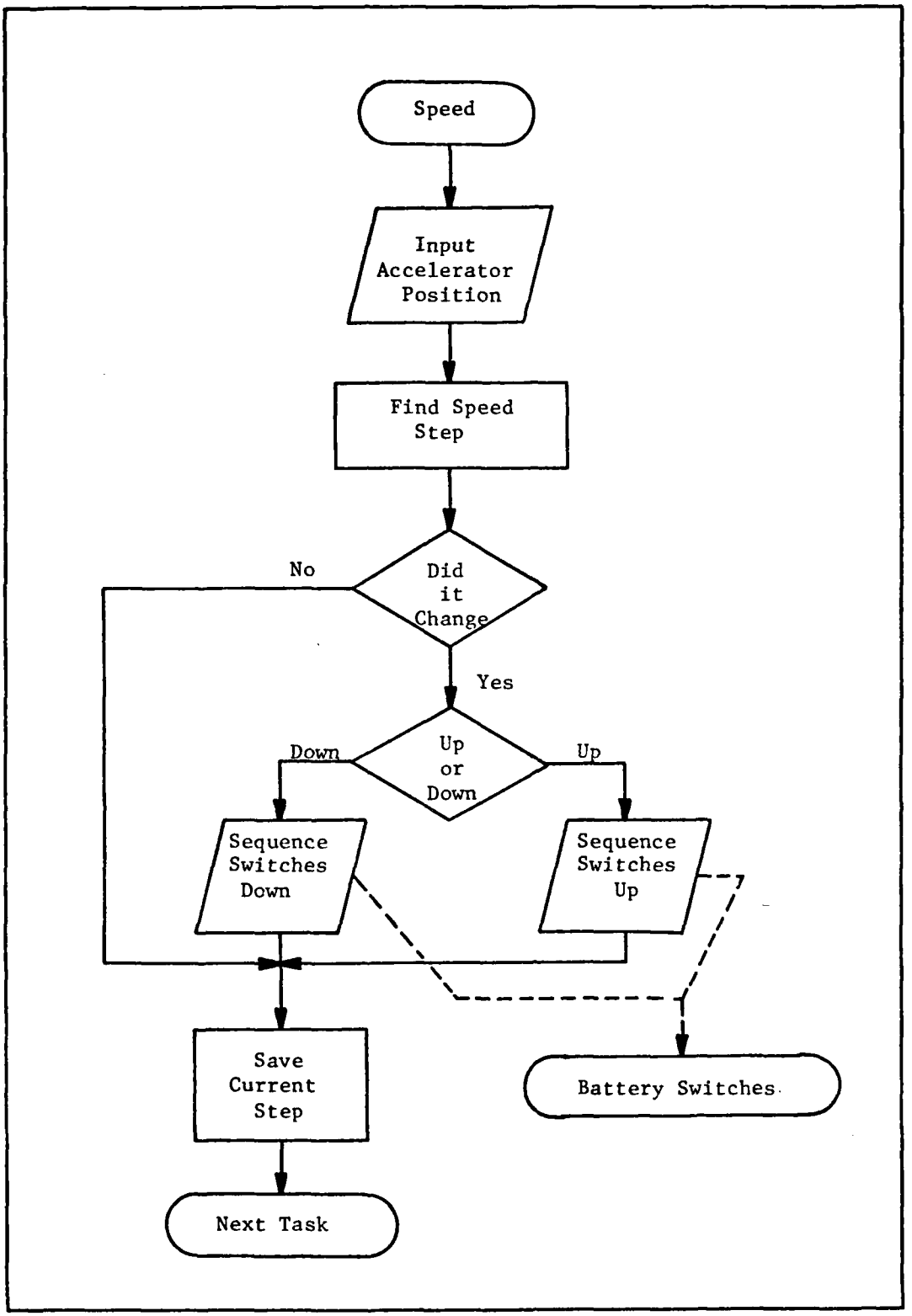


Fig. 9. Speed Control Flow Chart

matrix from beginning at location 01B2H in six rows of three elements. The first element is the accelerator position limit and the next two are switch settings, as shown in Figure 10. In the switch setting only bits 0 through 5 are used by the battery controller to set the switches. The switch setting in column two is the intermediate switch condition and the setting in column three is the final condition. The switches were always sequenced through an intermediate step to a final step to avoid the power dropouts. Table values can be adjusted by using the monitor substitute command. The source documentation is given in Appendix C.

Starting Location	Position Limits	Switch Settings	
01B2H	1AH	00H	00H
01B5H	33H	20H	20H
01B8H	66H	2AH	2AH
01BBH	9AH	28H	2DH
01BEH	CDH	29H	3BH
01C2H	FFH	2FH	2FH

Fig. 10. Speed control Set Point Table

Energy Monitor. The energy monitoring routine measures the amount of energy used by the DC motor during a driving sequence. The flow diagram is shown in Figure 11 for this routine. The monitoring routine numerically integrates with respect to time the produce of the DC motor applied voltage and current, as indicated in Equation 1.

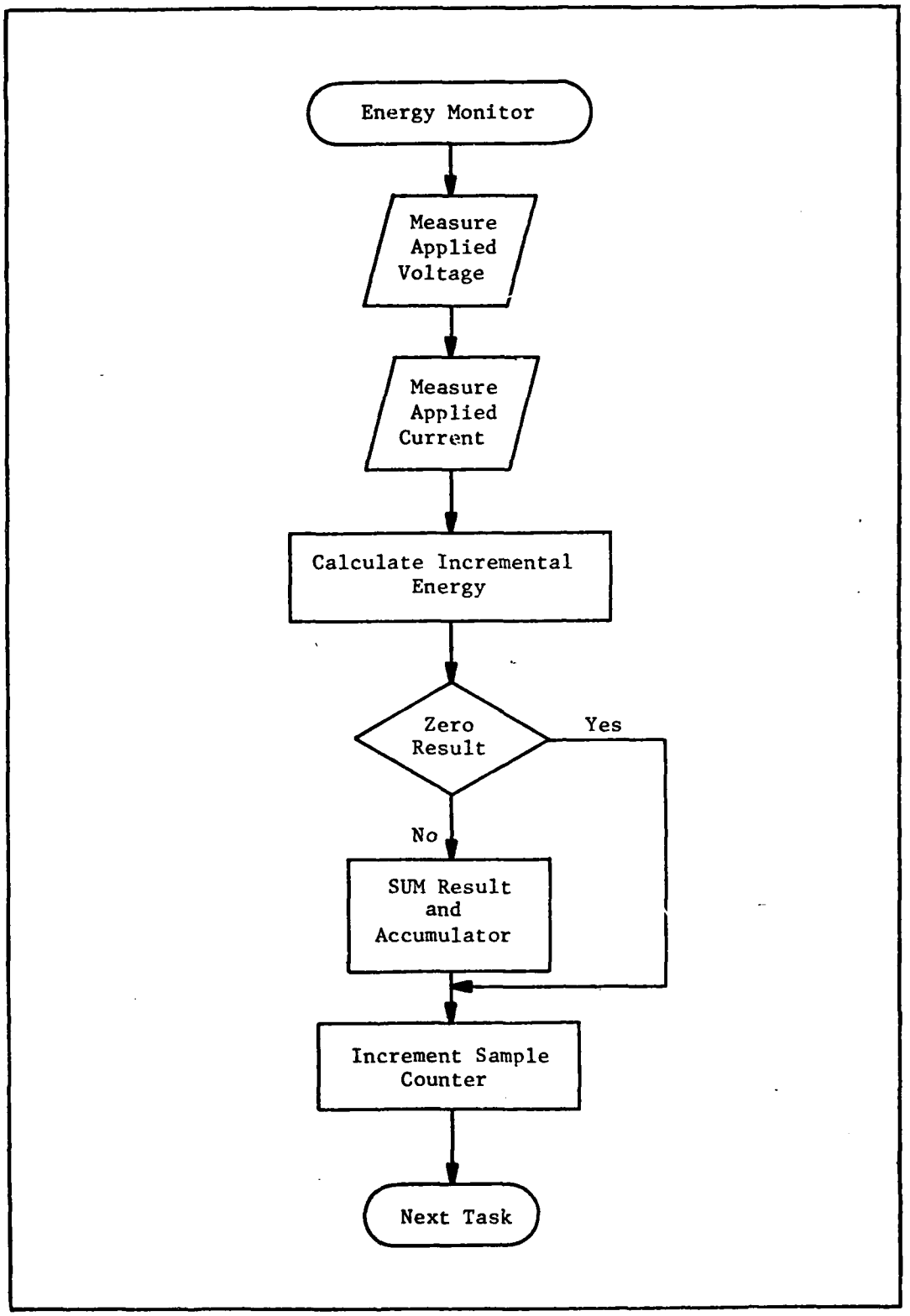


Fig. 11. Energy Monitor Flow Chart

$$\text{Energy} = \sum_{j=1}^{N} k_1 E_j \times k_2 I_j \times k_3 T = k^* \sum_{j=1}^{N} E_j I_j \text{ Joules} \qquad (1)$$

where

k_1 = Volts per count (See Appendix D.2)

E_j = A/D converter counts in the jth voltage measurement

k_2 = Amps per count (See Appendix D.3)

I_j = A/D converter counts in the jth current measurement

k_3 = Number of sample periods per measurement

T = Sample period in seconds (See Appendix D.1)

The energy monitoring routine implements the summation part of equation 1. It accomplishes this by sampling the applied voltage and current during each pass through the interrupt-serviced control routines and accumulates the SUM of incremental energy. The value in the energy accumulator is an equivalent value in terms of counts-squared and is converted to engineering units by multiplying it by the composite constant k* Joules per count-squared. The k* constant is used by the energy calculator.

To implement the energy monitor function in the application program multiple byte software multiply and summation subroutines were programmed. The routines were designed to retain numerical accuracy during the data gathering process and to let the energy calculator round off the result. Thus, these routines were capable of performing multiple word arithmetic.

A question of how big the product and summations could get during a normal driving period of approximately 2 hours needed to be addressed (Ref. 3). The voltage and current measurements were eight bit outputs from the A/D converter. The resultant product was 16 bits or two words. The summation process accumulated this product 8 times per second for

two hours, or 57,600 samples. If a worst case product was generated (all ones) during every sample, a carry bit was generated on every two-word sum. These carries needed to be accumulated to increment the accumulator on every sample period or 57,600 times. This required 16 additional bits in the summation accumulator or a total of four additional words to minimize storage requirements.

Considering the above information, the multiply and summation routines were designed to use multiple word memory registers that could accommodate calculation results of any word length. An additional accumulator was incremented once per sample period. The accumulated value was used by the energy calculator to compute total elapsed time.

Energy Calculator. The energy calculator routine was used to translate the value accumulated by the energy monitor into a quantity more readily understood by the vehicle operator. As described earlier, the output from the energy monitor was placed into the energy accumulator. The energy calculator was used to multiply a requested sample of this accumulator's contents by the lumped constant k* in equation 1. The value of k* was determined experimentally using known voltage references (see Appendix D.4). Figure 12 shows the flow chart for this function of the energy calculator routine, which includes the computation of elapsed time mentioned in the energy monitor section. For more detailed information, see Appendix C.

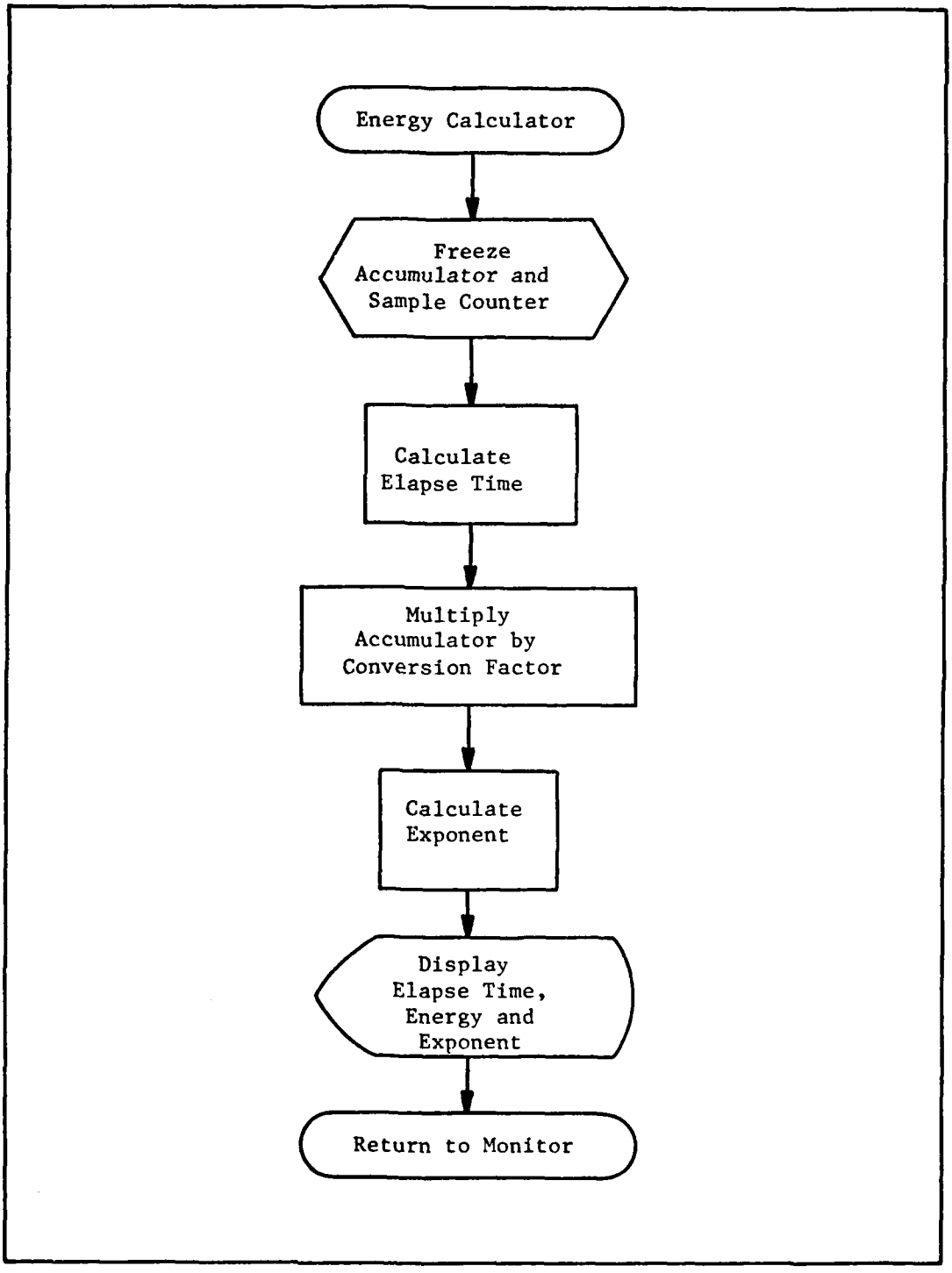


Fig. 12. Energy Calculation Flow Chart

V. Installation and Performance

The installation and performance of the vehicle controller consisted of two major phases. First, the control system was assembled and tested in the laboratory. Second, the control system was installed in the electric vehcile for testing.

Laboratory Testing

The laboratory testing involved the verification and trouble shooting of the control system hardware and software. The hardware and software modules were checked individually and operationally verified through functional testing. Then, the entire control system was assembled and functionally tested under simulated operating conditions. The energy conversion factor was also determined during testing of the control system. The system met the design operating criteria.

Vehicle Testing

After laboratory testing was completed, the control system was installed in the test vehicle. Testing was conducted in two steps. First, the digital control system was operated with the electric vehicle turned off to ensure that the system was installed properly. Second, power was applied to the vehicle to operationally test the controller operation in the vehicle.

Quiescent testing of the control system in the electric vehicle was used to verify the laboratory simulation results. No problems were encountered with the control routines. Energy measurements could not be made since the vehicle was turned off. In later tests, the vehicle was operated to check the controller's performance under driving

conditions. The control system failed to operate as the first voltage step was applied to the DC motor. Investigation revealed that the analog channels had saturated, and the A/D converter outputs were all ones. The control system signal ground also displayed a sharp voltage shift when the DC motor was turned on and off. This ground plane shift was of sufficient magnitude to momentarily change the logic levels of instructions such that they were improperly interpreted by the microprocessor halting operation.

The microcomputer sensitivity to DC motor engagement was also checked without initiating the control software. Tests showed that the monitor operated properly. This was probably due to the difference in the relative lengths of the program loops. The monitor waits for a terminal input with a loop that is 16 instructions long. The control software loop is several hundred instructions in length. Apparently then, the monitor loop is so short that it is unaffected by spurious noise, whereas the longer control loop is so grossly affected that the software stops functioning. The failure is probably caused by noise in both the analog inputs and on the signal ground.

VI. Conclusions and Recommendations

Conclusions

The general conclusion evident during the development and testing of the electric vehicle controller was that a microcomputer based controller could be constructed and programmed to operate and monitor test vehicle functions. With very little effort, it was clearly demonstrated that control and monitoring parameters could be calibrated under simulated conditions to provide desired performance and monitoring accuracy. This ability, coupled with a general purpose development microcomputer controller, should greatly enhance the study and development of electric vehicle drive systems.

Additional conclusions derived from this effort were as follows:

1. The use of standard prototyping techniques and the selection of a mature microprocessor family allowed the hardware development and checkout to be completed in minimal time.

2. The availability of a software development microcomputer and a versatile "ROM Monitor" for the controller produced a powerful software development team. This technique shortened turn around time during the verification, testing, and debugging of the application software.

3. The development of a control system of this type brings together many technical disciplines. Complementary studies need to be undertaken in digital instrumentation systems, mechanics, control theory, and software development to solve the technical problems presented by this project.

Recommendations

The availability of a general purpose microcomputer controller, as constructed by this thesis project, provides many opportunities for

further study. Several areas requiring further investigation are apparent:

1. Instrumentation. This is an important area of study in that external sensors provide the control stimuli to the microcomputer. Instrumentation needs to be able to function in the high electromagnetic noise environment of test vehicle and still report measurements without errors. To this end the analog scaling circuits need to be modified to improve their immunity to electronic pulse and noise interference.

2. Hybrid Control. The hybrid energy management control interface described in Section III and control software modules need to be constructed. The design described is directed toward an energy management feedback controller. This controller should also provide speed regulation as battery voltage decays. The present controller uses a scheduled gain with the operator managing the control loop to set the speed, as illustrated in Figure 13.

The hybrid controller would maintain vehicle cruising speed while unloading the DC motor. Figure 14 shows a possible control system block diagram. The ultimate goal of this control configuration would be to optimize the test vehicle's driving range through efficient application of the hybrid engine.

3. Chopper Controller. The electric vehicle should be adapted to use a chopper power supply in place of the present battery switching voltage regulator. In a system using a two-quadrant chopper voltage regulator, nearly continuous linear speed control is achievable. In the present system there are essentially five discrete speed settings. This system would require that the series DC motor be reconfigured as a "shunt," separately excited, DC motor. The armature current of the motor

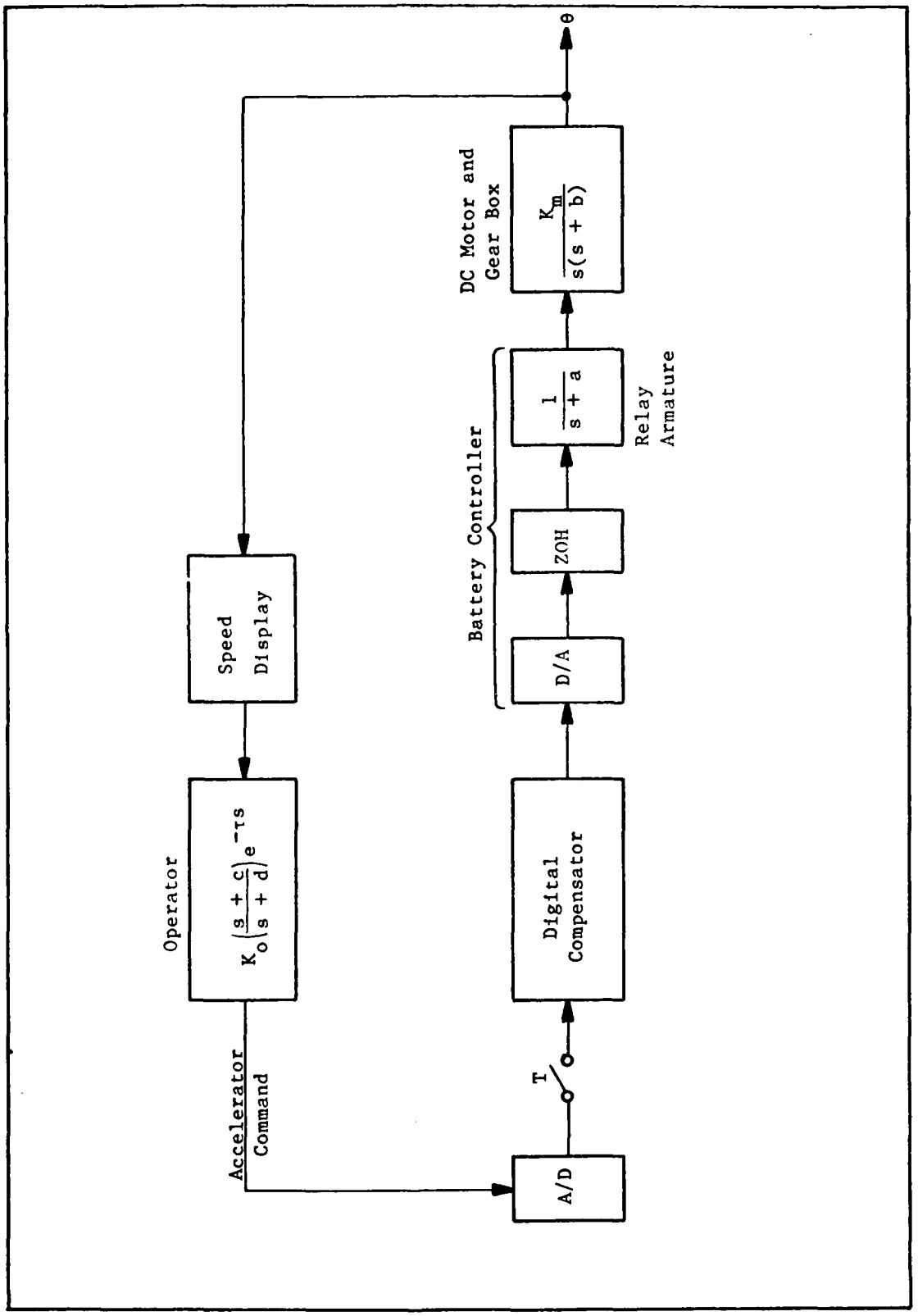


Fig. 13. Current Vehicle Control System Block Diagram

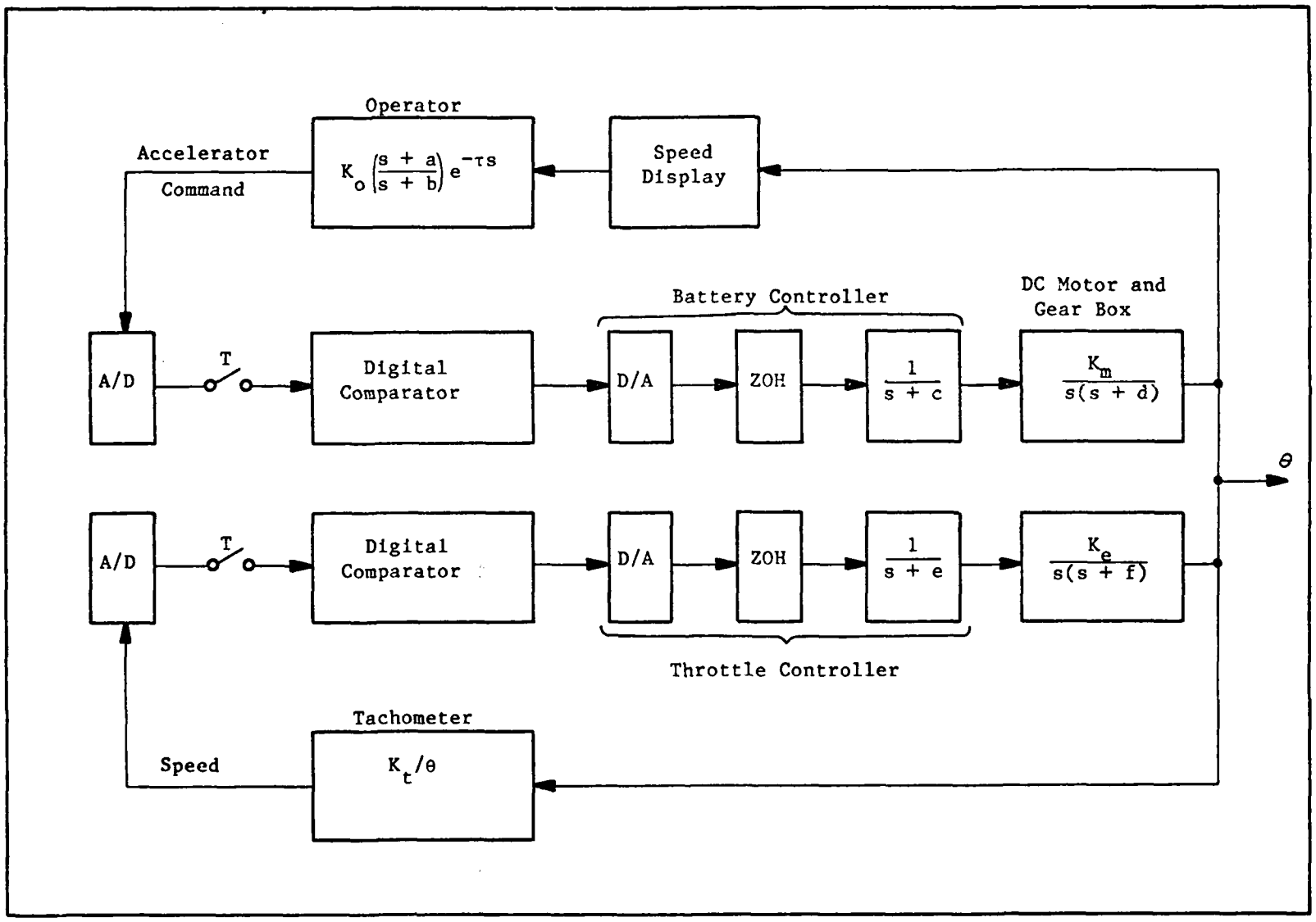


Fig. 14. Hybrid Control System Block Diagram

is continuously controlled in either direction to check motoring and regeneration torques of the vehicle. Between zero and base speed of the motor, the armature current is controlled by a two-quadrant chopper in the armature circuit. At high speed the field current is regulated to control armature current. The system would also allow experiments in regeneration of battery energy through dynamic braking (Ref. 12).

4. Data Storage. A low power data storage unit should be designed and constructed for use with the electric vehicle controller. This unit would be used to collect historical test data which could subsequently be used to verify the Stafford EVSIM program, and vehicle component models. This can be accomplished by either using the microcomputer RAM or removable media storage. The fundamental difference is the amount of available data storage space, and the removable media has the distinct advantage in this regard. Datel/Intersil manufactures miniature low power digital cassette recorders which can interface to either a serial or parallel port (Ref. 13).

5. Control Panel. A control panel which can be used by the vehicle operator needs to be designed to replace the standard data terminal currently in use. The panel should provide continuous status of battery discharge and operating mode, and be able to accept commands. One solution may be to develop a low power, light weight data terminal.

Bibliography

1. Neisler, Troy H. letter, subject "Recommendation Resulting from MEEP Project H75-21C, Vehicles, Small Type," to HQ AFLC/LOWCS and HQ USAF/LGTN, 28 June 1977.

2. Department of Energy, DOE Interagency Agreement with the USAF, DE-AIO-80CS5028, Washington: Goverment Printing Office, July 1980.

3. Stafford, Kennith A. Electric Vehicle Simulation for Design Optimization, AFIT/GEA/AA/80D-19, Air Force Institute of Technology, Air University, Dec. 1980.

4. Zilog 1981 Data Book, Microcomputer Component Product Information Catalog, Zilog Inc., 1981.

5. Liptak, Bela G. Instrumentation Engineering Handbook, Volume 1, Process Measurement, Chilton Bok Company, 1969.

6. Check, W. E., Cheng, G. Hill, M. Hollen, J. Miller, "Microcontroller Includes A-D Converters for Lowest-cost Analog Interfacing," Electronics, 122-127, (25 May 1978).

7. Sinclair, David "System Design with Single Board Microprocessor Systems," Science Abstracts on Computers and Control, NO. 81-9712, 1981.

8. Denmark, Anthony M. "Distributed Control Boosts Process Reliability," Electronics, 163-164 (15 April 1976).

9. Van Cott, Harold P., Robert G. Kinkade Human Engineering Guide to Equipment Design, Mc Graw-Hill Company, 1972

10. M. Reed and H. W. Mergler, "A Microprocessor-Based Control System," IEEE Transaction on Electronic Control and Instrumentation, VIECI-24, 253-257, Aug 1977.

11. Z-80 SIO Technical Manual, 03-3033-03, Zilog Inc.,August 1978.

12. Bose, Bimal K. and Sutherland A. Hunt, "A Microcomputer Based Real Time Feedback Controller for an Electrtic Vehicle Drive System," IEEE Industry Applications Society, Conference Record, ISA79:25F, 743-748,1979.

13. "Instrumentation and Systems Handbook," Goldbook V3, 218S-226S, Hayden Publishing Company, Inc., 1980/81.

14. Z80-CPU Z80A-CPU Technical Manual, 03-0029-01, Zilog Inc, 1977.

15. Miranda H., M. Hatziemmanuel, "Blood Analyzer Tests 30 Samples Simultaneously," Electronics, 150-154, 15 April 1976.

Appendix A

Hardware Data

This section contains the design drawings and operating theory for selected circuits of the microcomputer, analog input circuits, battery controller, computer power supply and vehicle installation.

Microcomputer

The microcomputer is a single board design using the Zilog Z80 microprocessor component family. A microcomputer built with the Zilog components allows the use of a bus orientated structure and can be assembled with a minimum number of components. Figure 15 shows the general component layout for this computer architecture and Figure 16 is the schematic diagram. Table I summarizes the microcomputer characteristics. The basic microcomputer structure is fairly straight forward and is well documented in Zilog technical literature (Ref. 14). The microcomputer has some system-peculiar circuits which require further discussion.

Sense Switch. This switch, as indicated in Figure 16, is an input device required by the ROM monitor. It is used to set the initial peripheral hardware configuration and, as a minimum, bits Ø and 1 must be set to conform to the terminal type used to access the computer. See Appendix B, Table VI.

Reset Memory Masking. The reset memory masking control logic is shown in Figure 17. This circuit was used to temporarily overlay the first three words of the ROM monitor as the first three words of RAM memory. This puts a special requirement on the ROM monitor to contain an unconditional jump instruction in the first three words. The jump

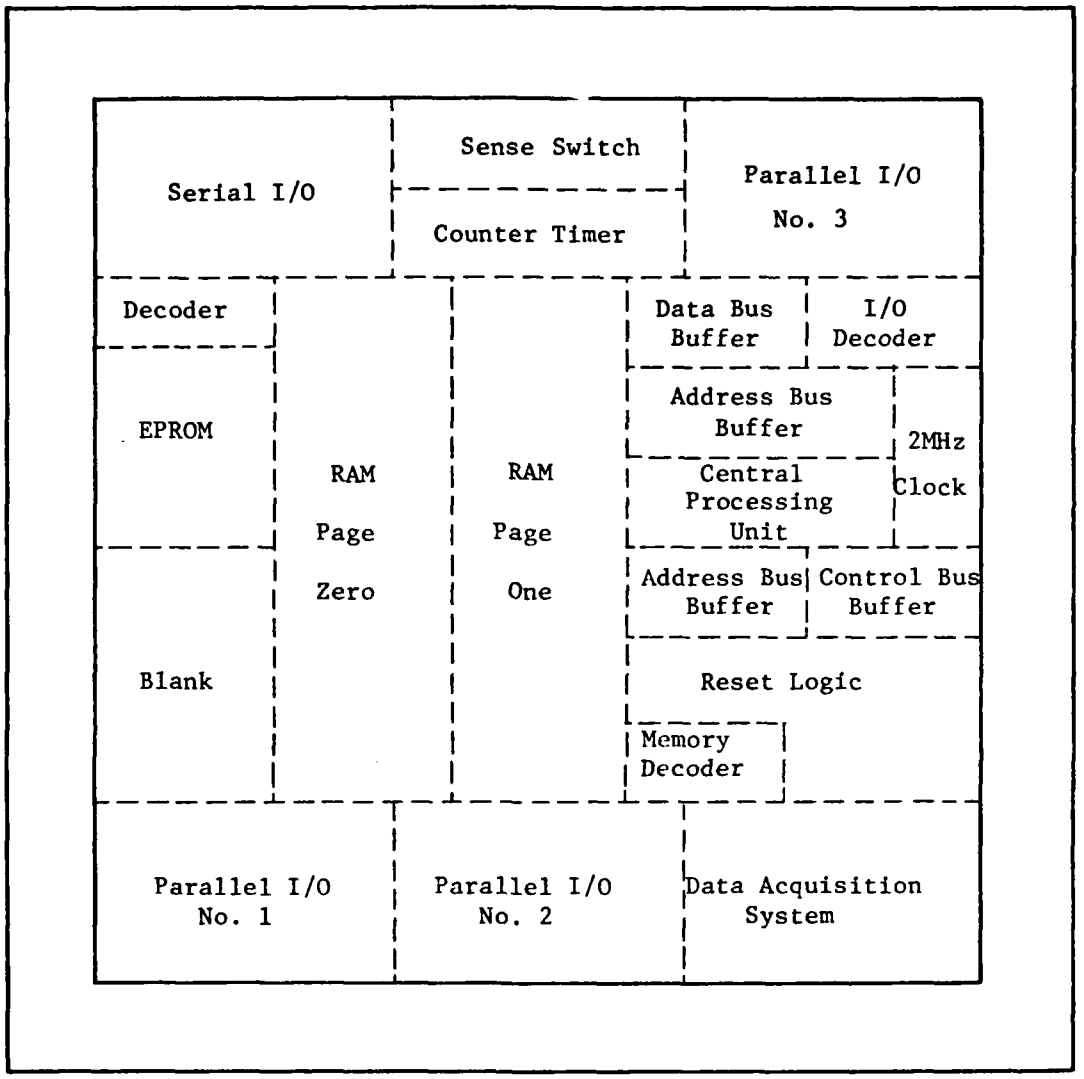


Fig. 15. Functional Layout of Microcomputer

instruction in turn points to the starting address in the ROM-based operating system. The circuit uses the fact that the Z80 CPU does an op-code fetch at memory location zero after a system reset. The logic circuitry steers the active low zero page select to the EPROM decoder enable. The steering flip-flop is reset by the system reset so that the Q output is low and the $\overline{Q}$ output is high. The central processing

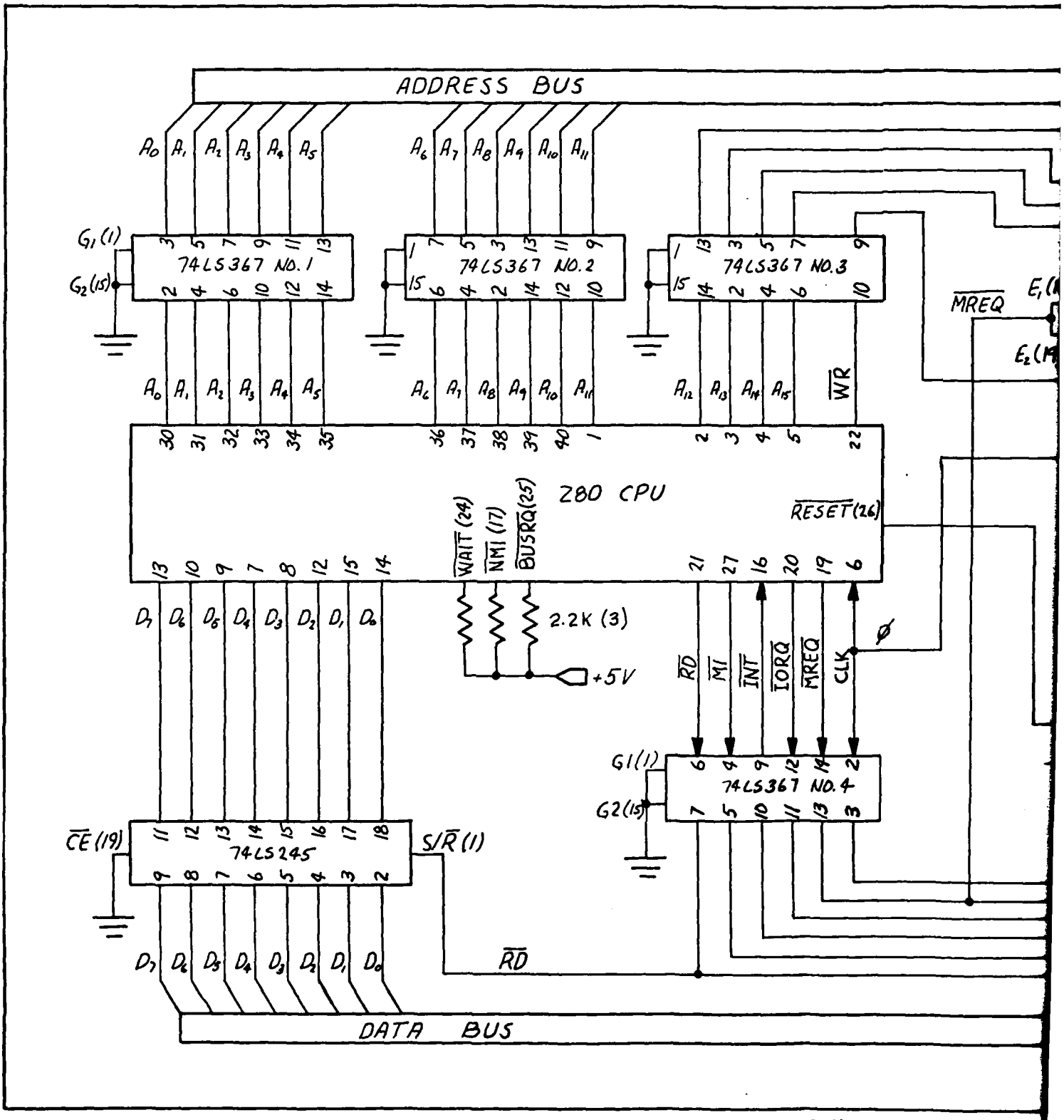


Fig. 16. Microcomputer Schematic Diagram, Central Processing Unit (Sheet 1 of 5)

it (Sheet 1 of 5)

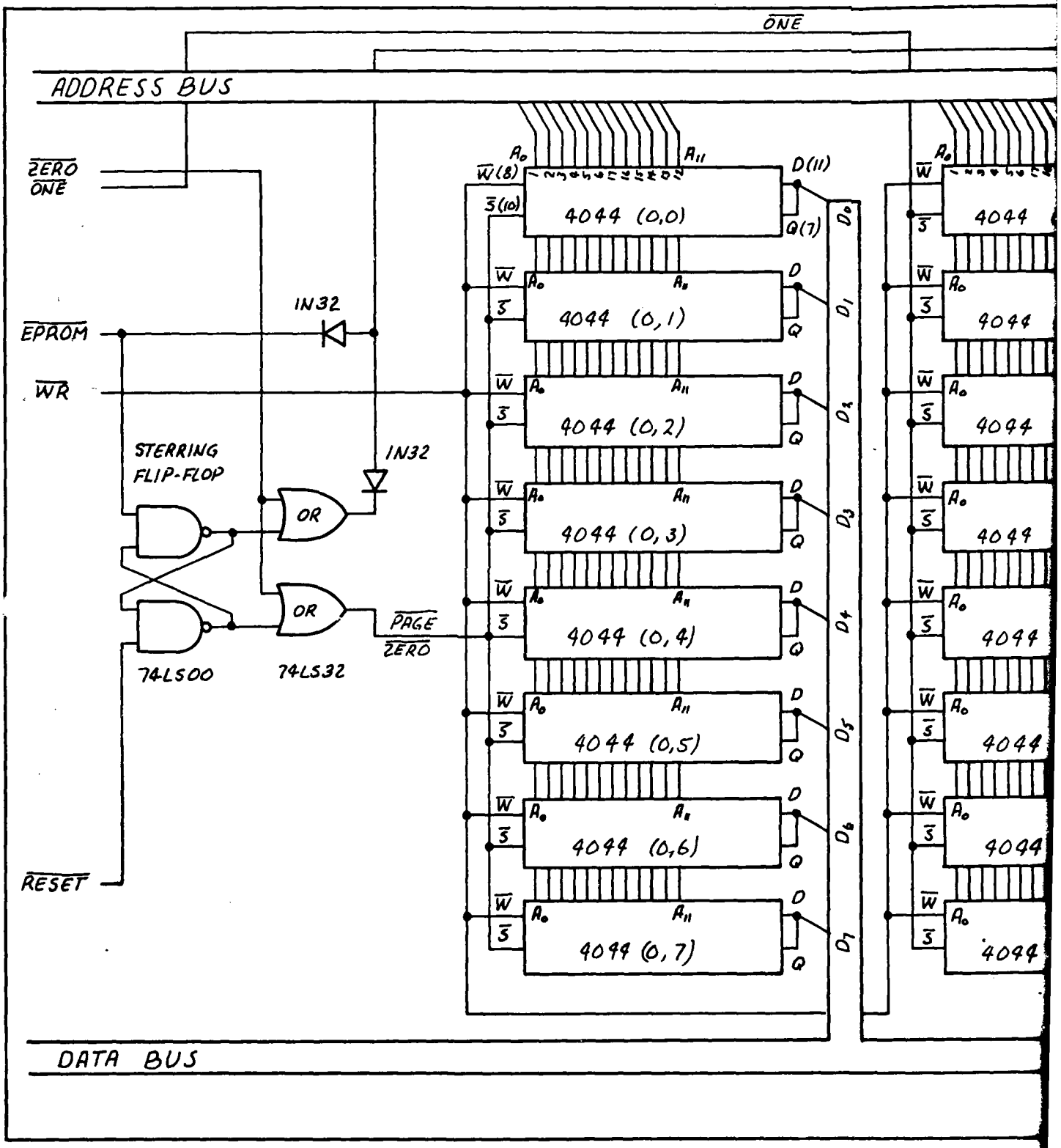
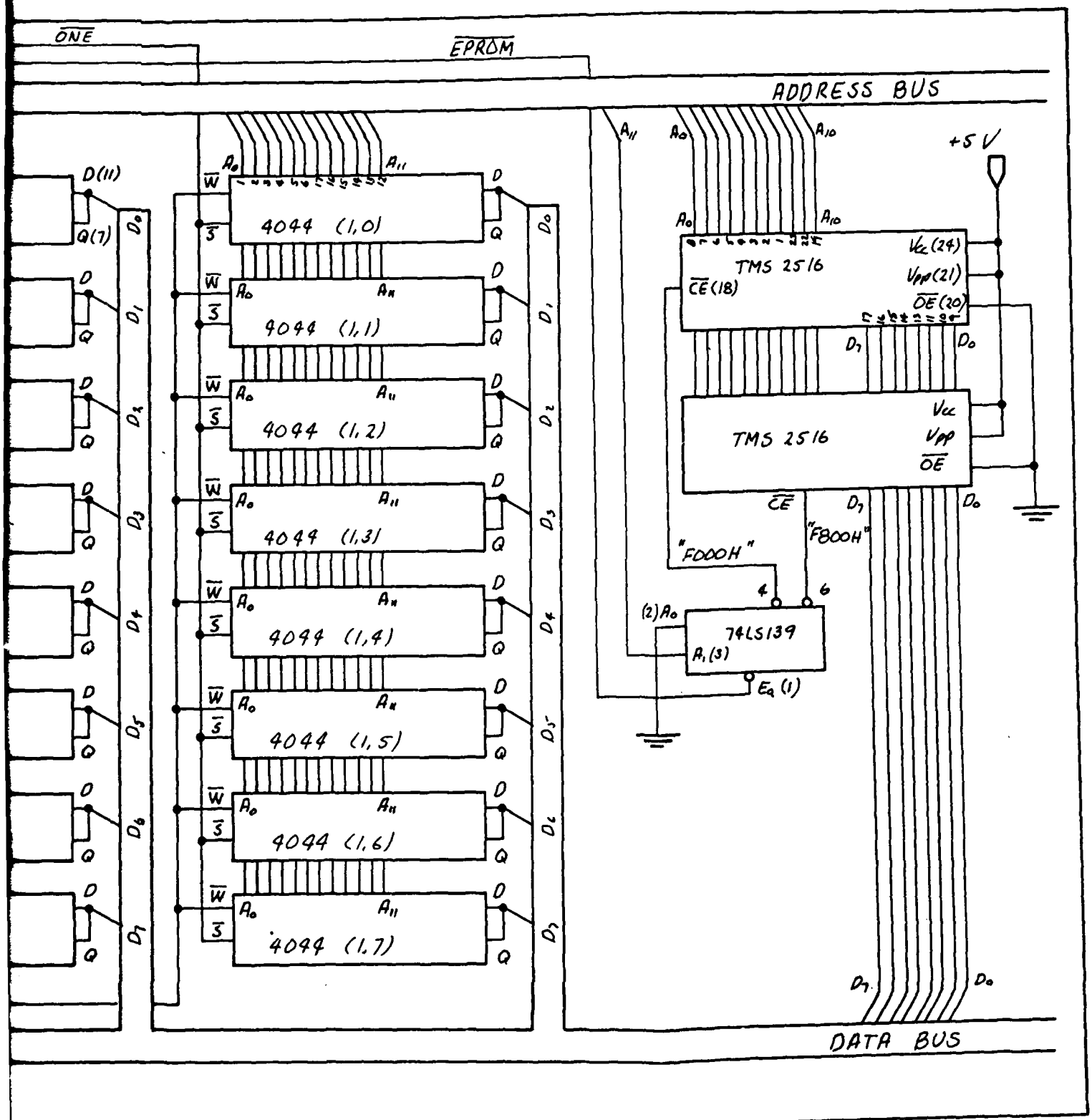


Fig. 16. Microcomputer Schematic Diagram, Memory (Sheet 2 of 5)


ONE
EPROM
ADDRESS BUS
+5 V
4044 (1,0)
4044 (1,1)
4044 (1,2)
4044 (1,3)
4044 (1,4)
4044 (1,5)
4044 (1,6)
4044 (1,7)
TMS 2516
TMS 2516
74LS139
"F000H"
"F800H"
DATA BUS

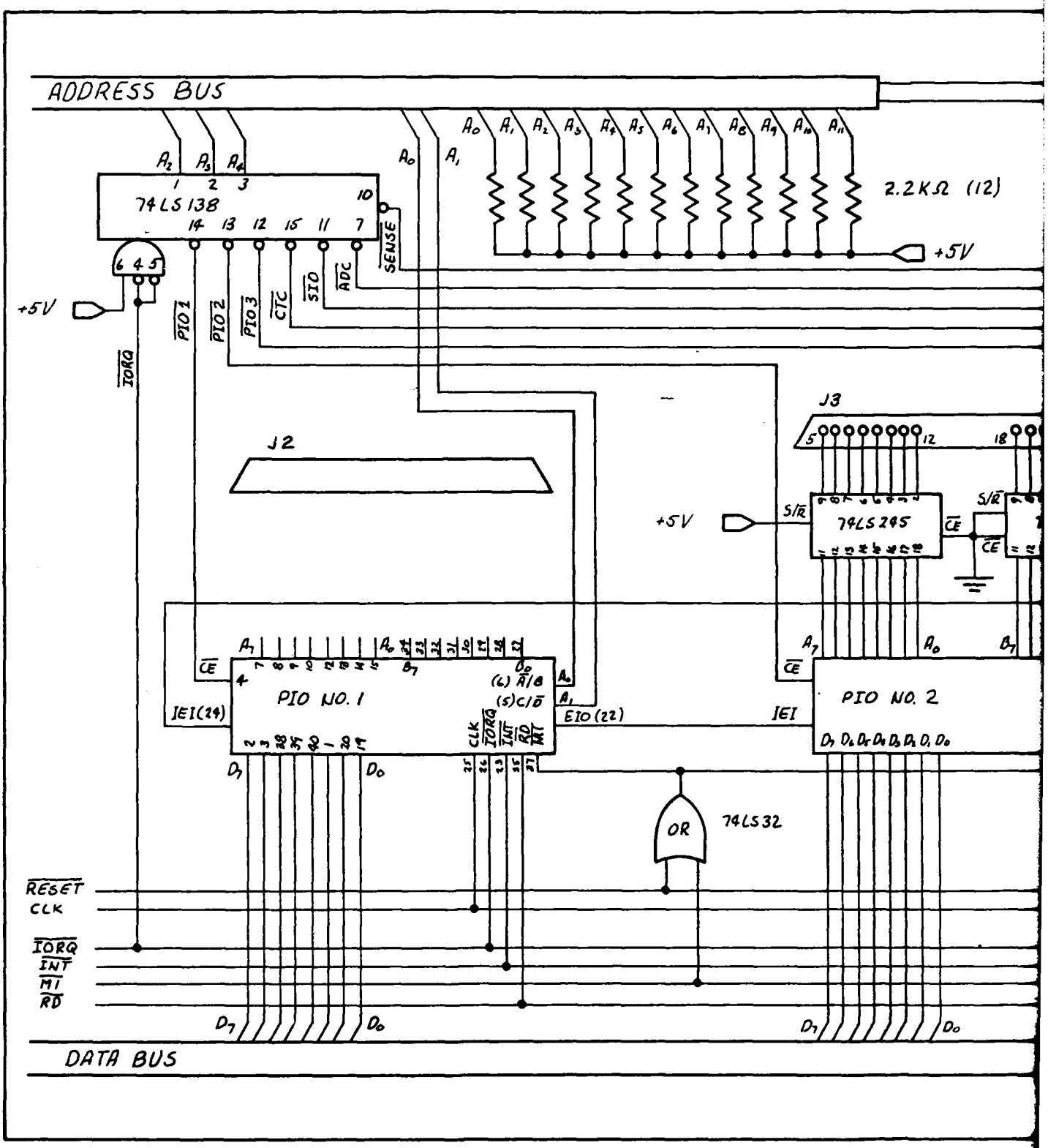
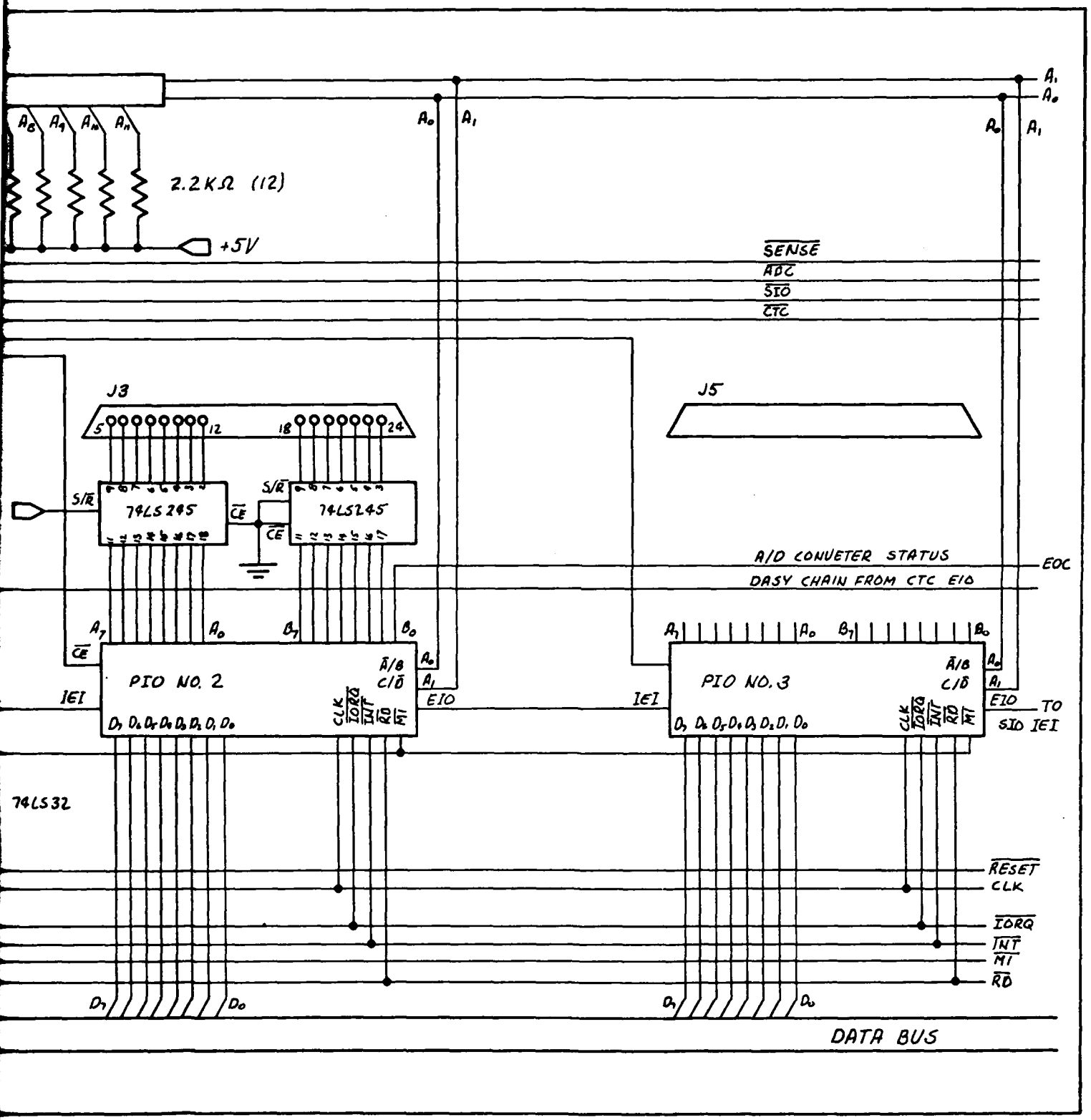


Fig. 16. Microcomputer Schematic Diagram, Parallel I/O (Sheet 3 of 5)


2.2KΩ (12)
+5V
SENSE
ADC
SIO
CTC
J3
J5
74LS245
74LS245
S/R
CE
A/D CONVETER STATUS
DASY CHAIN FROM CTC EIO
EOC
PIO NO. 2
PIO NO. 3
IEI
EIO
TO SIO IEI
74LS32
RESET
CLK
IORQ
INT
M1
RD
DATA BUS

of 5)

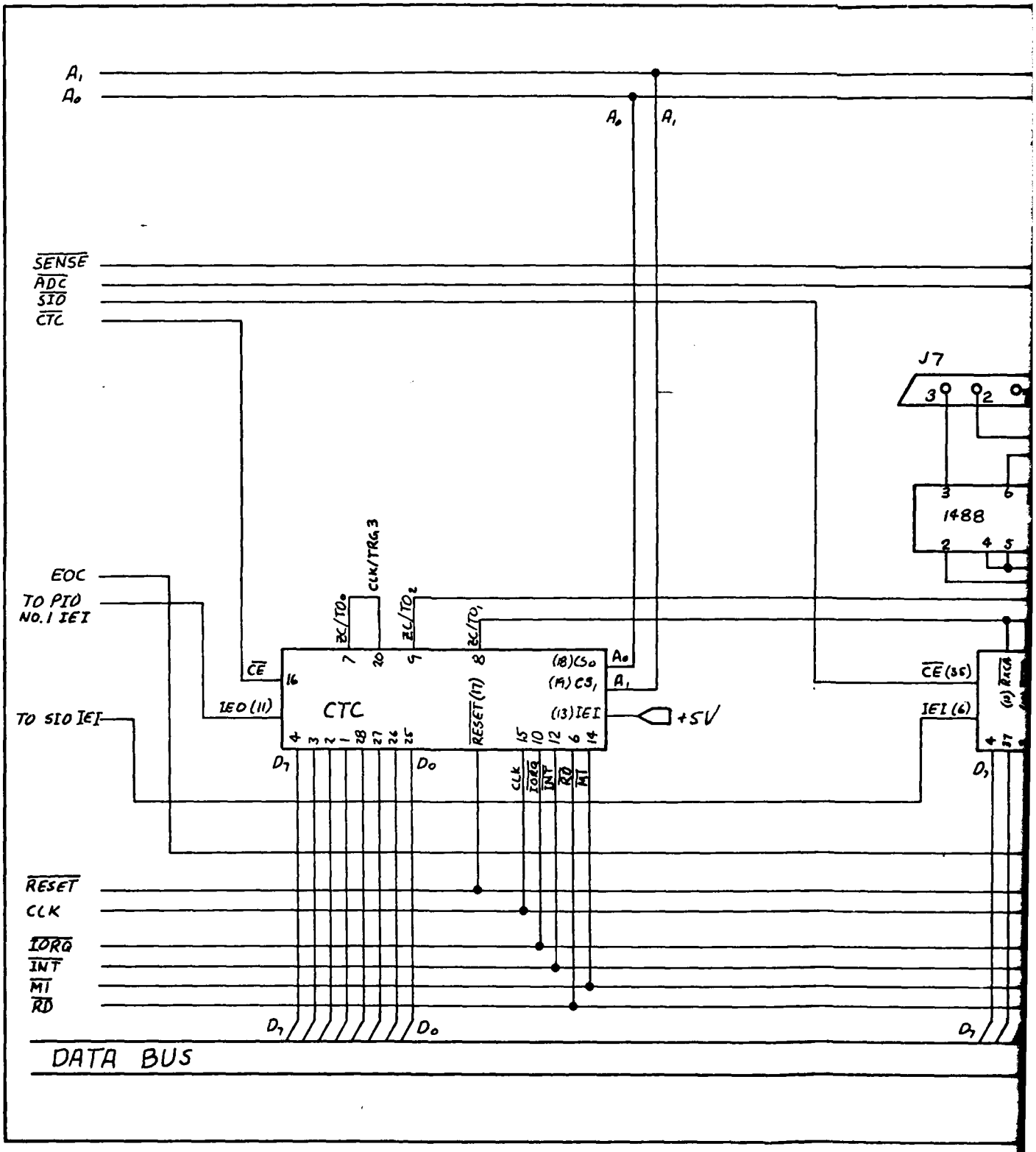


Fig. 16. Microcomputer Schematic Diagram, Serial I/O (Sheet 4 of 5)

A1
A0
A1
SENSE
AOC
J7
J6
3
2
7
7
3
2
3
6
14
Vcc
+12V
1488
1
4
1489
2
4
5
1
VEE
-12V
6
3
A0
A1
CE (35)
IEI (6)
(1) RxCA
(14) TxCA
(15) TxDA
(12) RxDA
(21) RxCB
(19) TxDB
(18) RxDB
(34) A/B
(33) C/D
SIO
4
37
8
38
2
39
1
40
21
20
36
6
32
8
D1
D0
RESET
CLK
IORQ
INT
RD
M1
EOC
RESET
CLK
IORQ
INT
M1
RD
D1
D0
DATA BUS

of 5)

2

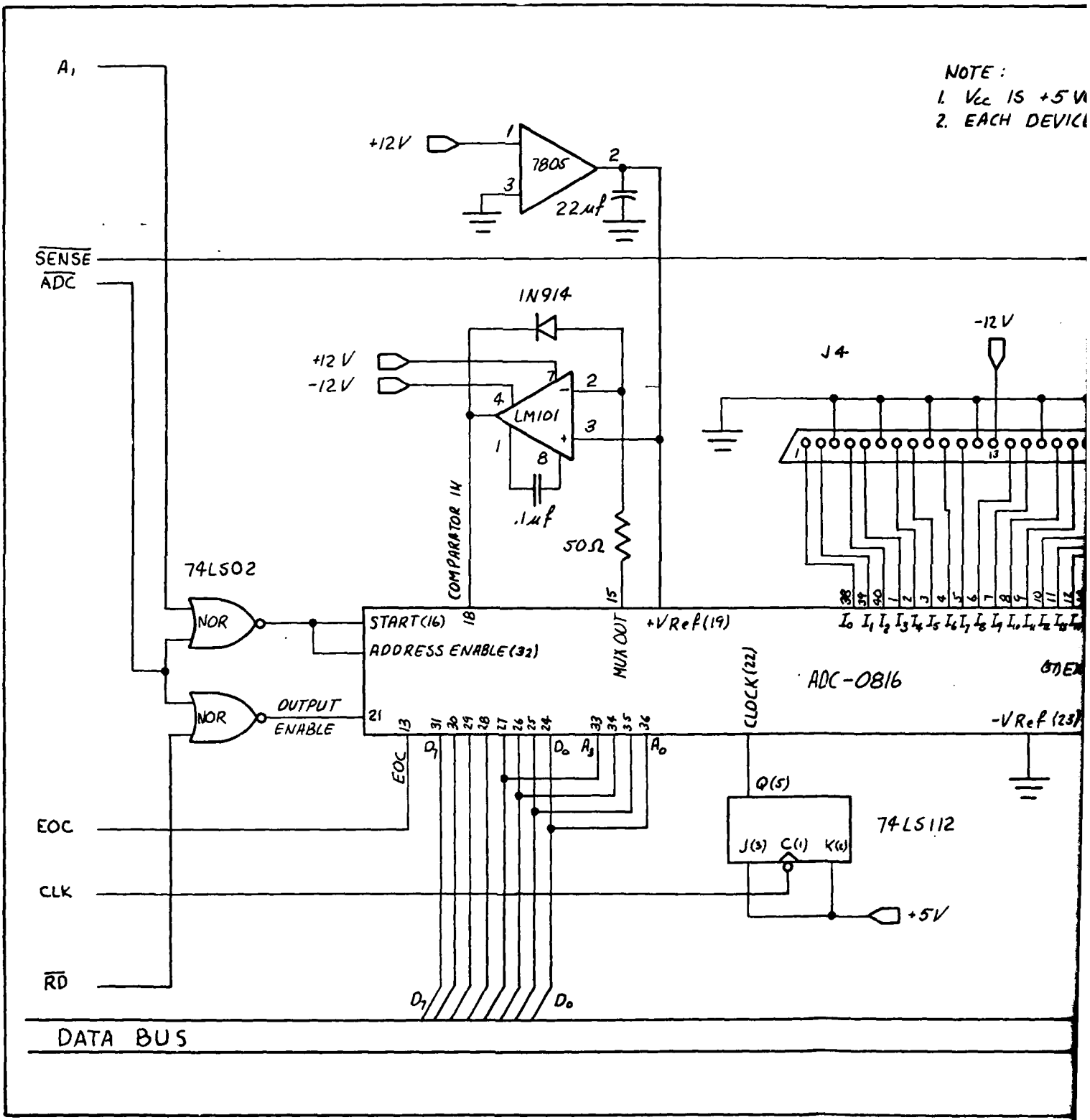


Fig. 16. Microcomputer Schematic Diagram, A/D Converter and Sense (Sheet 5 of 5)

NOTE :

1. V_{cc} IS +5 VOLTS IF NOT OTHERWISE INDICATED
2. EACH DEVICE IS BYPASSED BY A 0.1 μf CAPACTOR

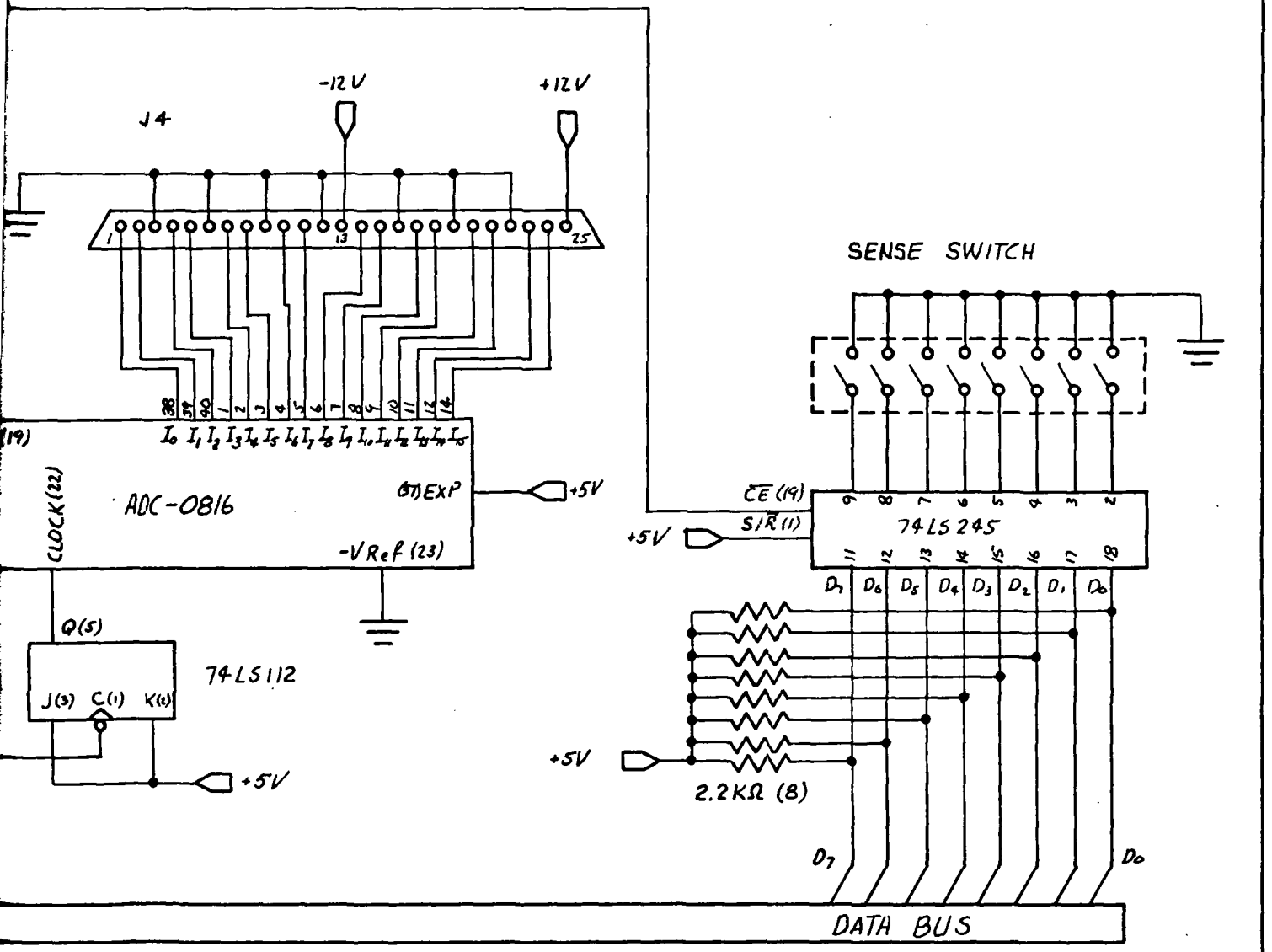


Table I

Microcomputer Characteristics

Central Processing Unit	2 MHz, Z80 microprocessor
Input/Output	Two programmable, RS-232 compatible, Serial channels, interfaced through Jack 6 and 7.
	Six programmable parallel data channels, interfaced through Jacks 2, 3 and 5.
	Sixteen 0 to +5 volts analog inputs interfaced through Jack 4.
System Clock	2 MHz, crystal controlled
Special Timing	Four counter timer channels, channels 1 and 2 are baud rate generators for the serial channels, channels 0 and 3 are cascade for long internal timing.
Memory	4K, Static-Random Access Memory (RAM) memory location 0 to 1FFF.
	4K, Erasable Programmable Read only Memory (EPROM), organized in 2K segments, memory location F000 to FFFF.
Power	Three regulated supplies: +5 volts and ±12 volts, supplied to Jack 1.
Dimensions	12" x 12" x 4 1/2"
Cooling	Active cooling, 6v dc fan

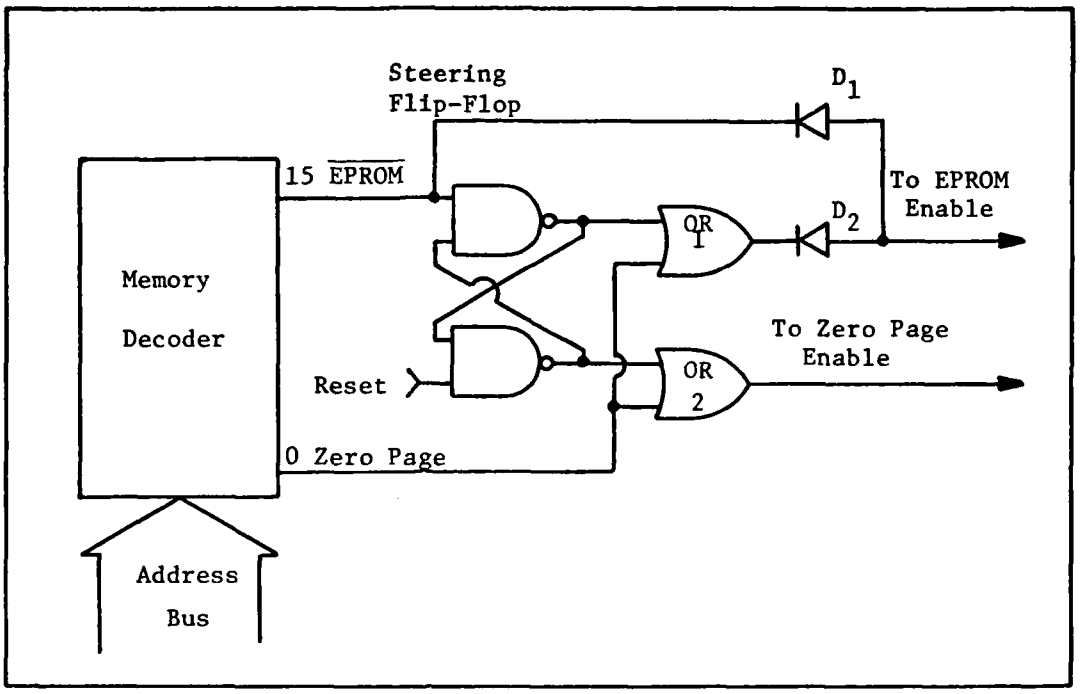


Fig. 17. Memory Overlay Logic Diagram

unit reads the jump instruction and, on the next op-code fetch address, the instruction indicated by the jump. The address is decoded as an EPROM memory location setting the steering flip-flop output and restoring the normal memory select connections until the next reset. The diodes D_1 and D_2 provide signal isolation needed for this circuit to function and prevent shorting of the logic power supply to ground.

Data Acquisition System. This microcomputer design incorporates a monolithic data acquisition system for collection of analog data, and is interfaced with the microcomputer as shown in Figure 18. The addressing logic maintains addressing continuity with the Z80 input/output components and software. The A/D converter starts a conversion automatically when it is addressed. The interface uses a PIO input channel bit for the end of conversion (EOC) detection test. The EOC signal is only valid after

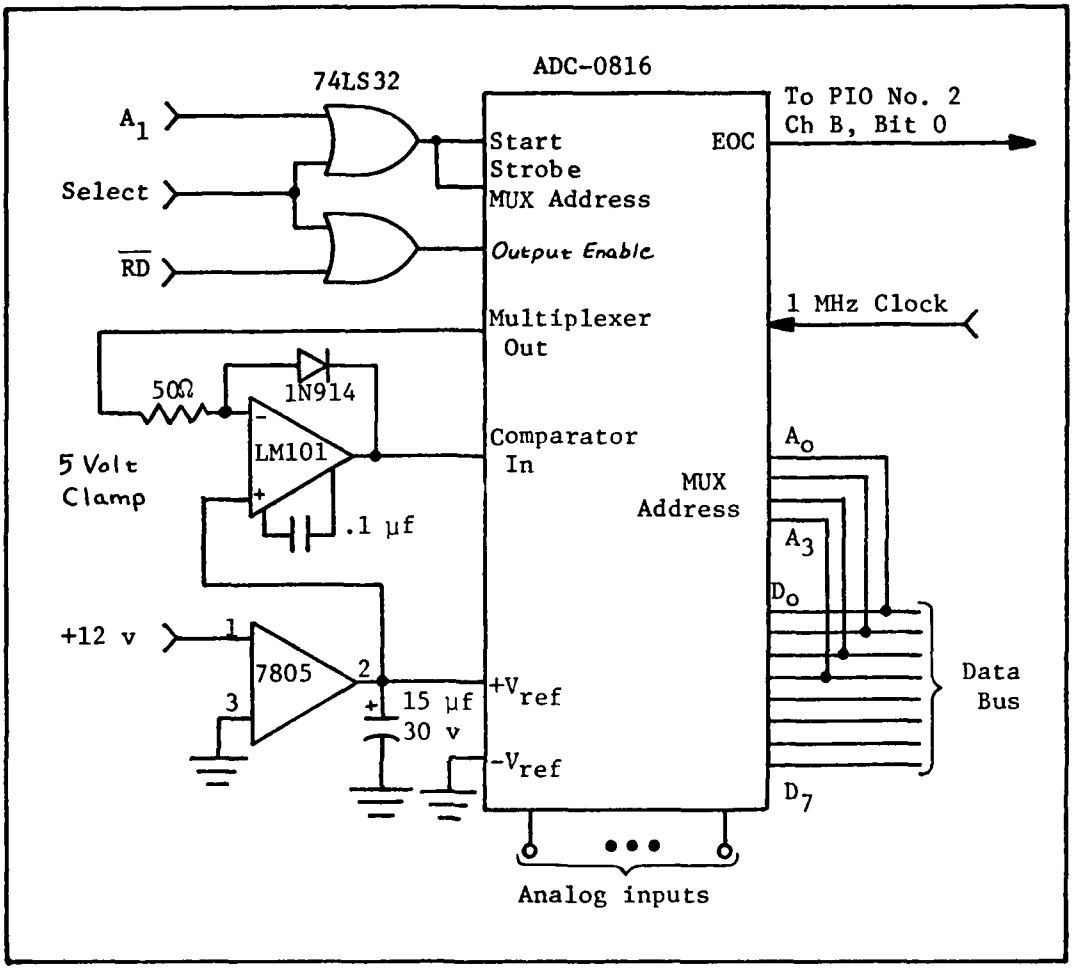


Fig. 18. Data Acquisition System Interface Logic

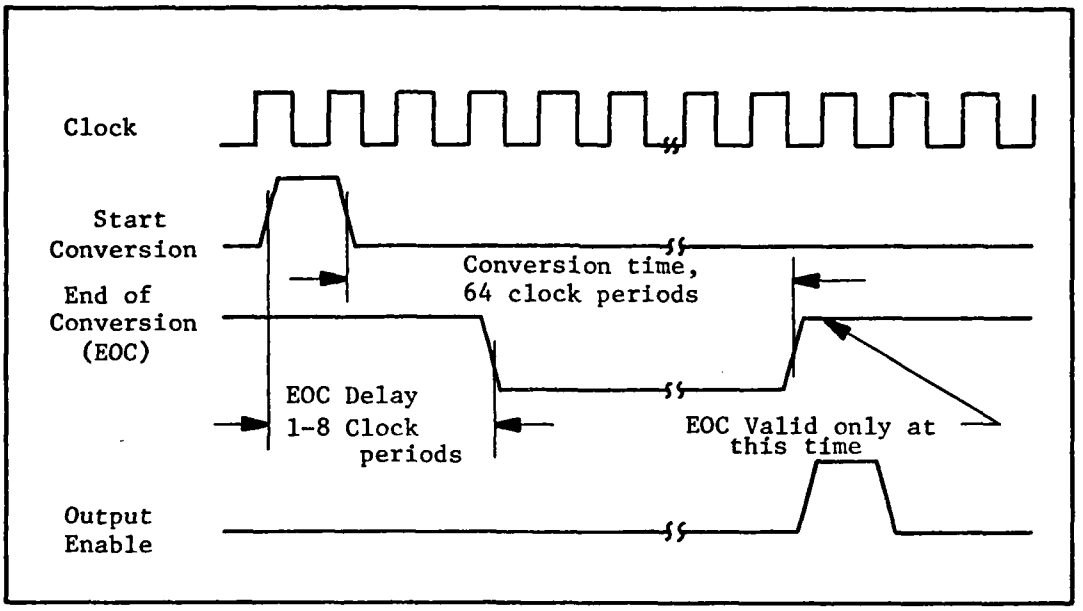


Fig. 19. End of Conversion Timing Diagram

it has been low and then goes high as shown in Figure 19. A software routine is used to test for the EOC transitions before reading the data register. Refer to the ADIO subroutine in Appendix C.

EPROM Memory Expansion. The microcomputer can be easily modified to add additional ROM program storage space when source programming exceeds the available space. The microcomputer currently uses two 2K EPROMs (TMS2516). Board space is available to install two additional ROM memory sockets with memory beginning at location EØØØH, as shown in Figure 20. The existing addressing logic is compatible with this improvement.

The microcomputer input/output channels are accessed through software calls and jacks on the enclosure frame as shown on Figure 21. Table II lists the hardwired address on the I/O devices and Figure 22 shows the wiring pin-outs of the connectors. The enclosure was built as indicated in Figure 23.

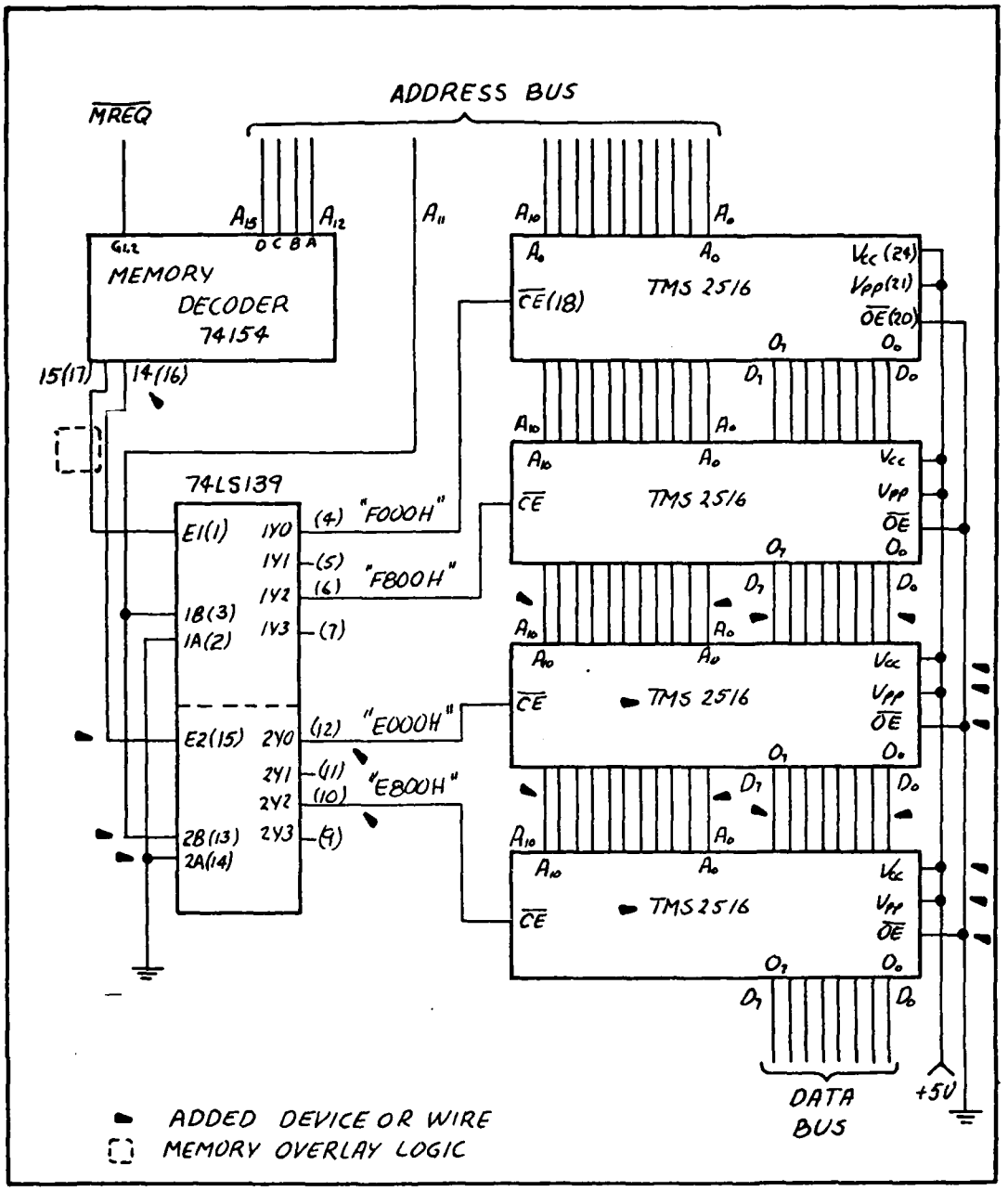


Fig. 20. EPROM Enlargement Schematic Diagram

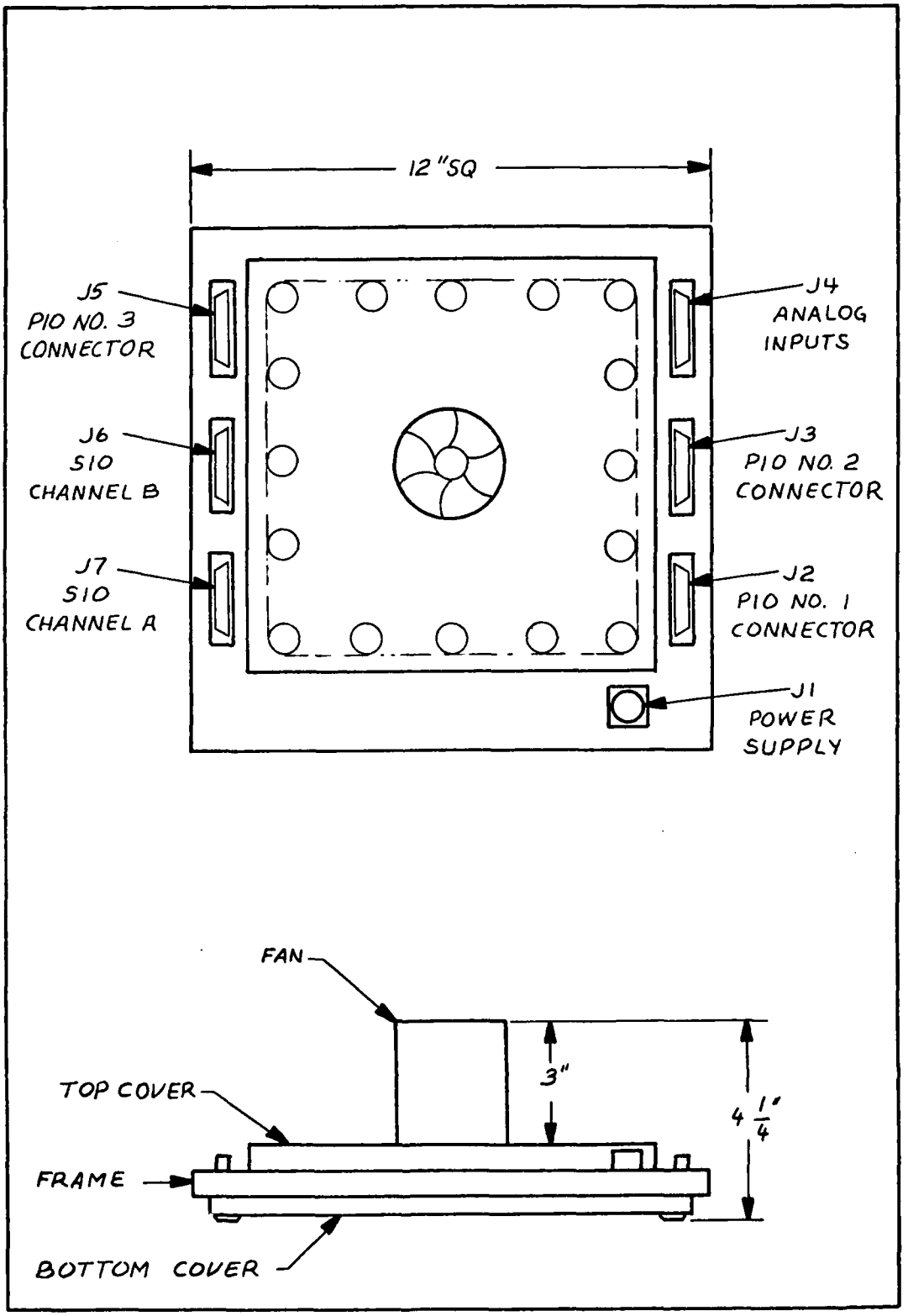


Fig. 21. Outline Drawing of Microcomputer and Connector Designation

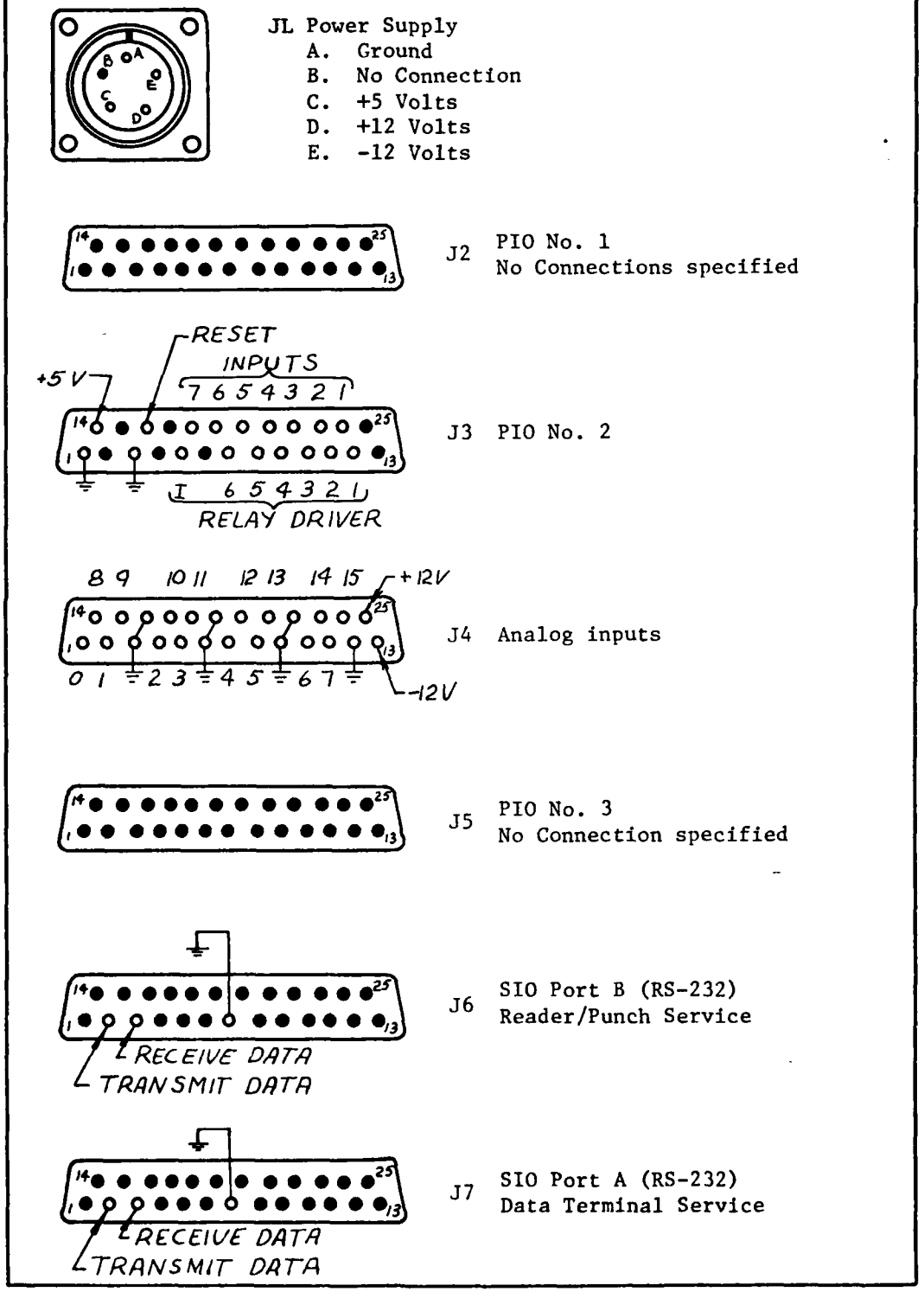


Fig. 22. Microcomputer Connector Pin-out Diagram

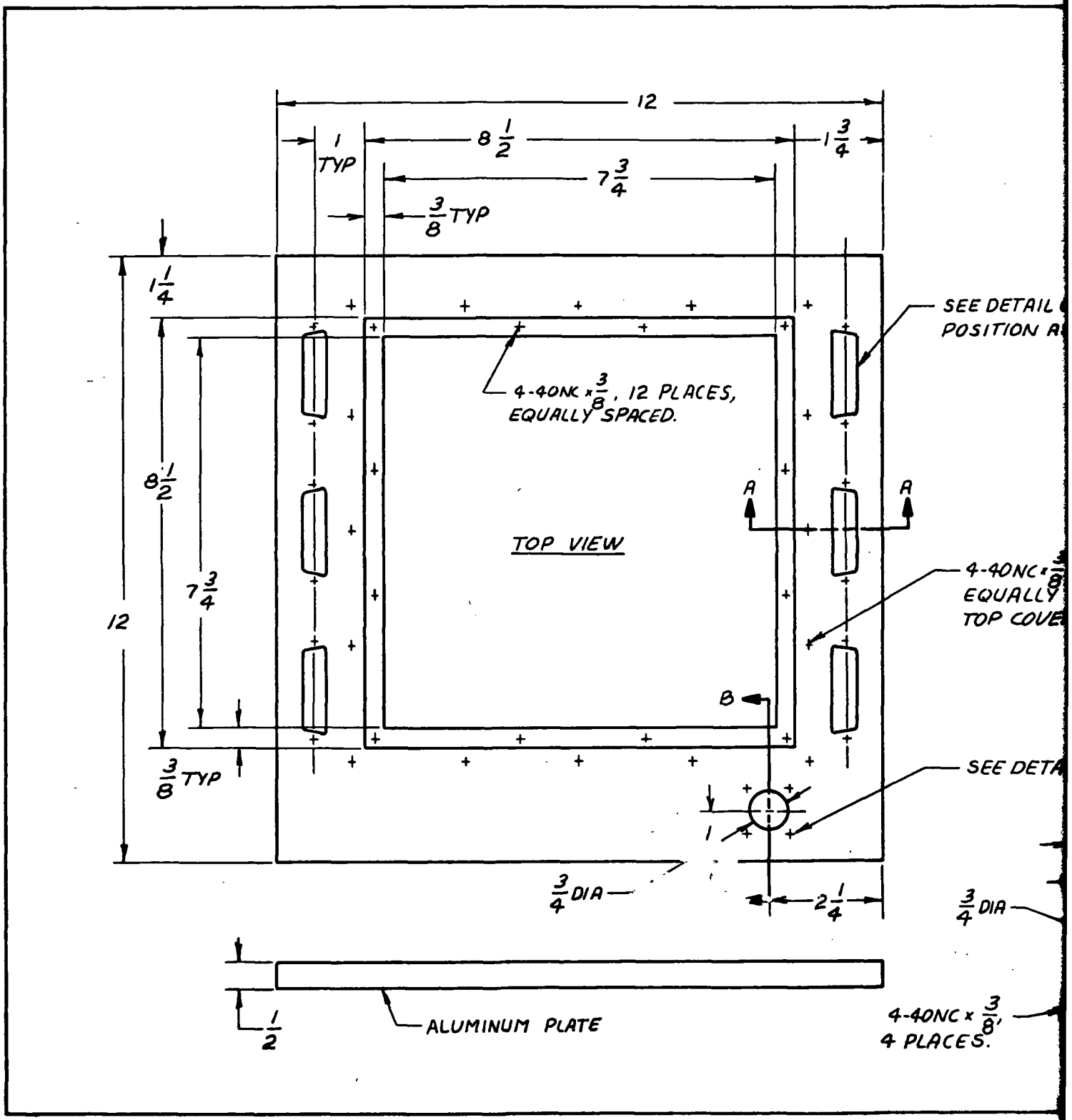
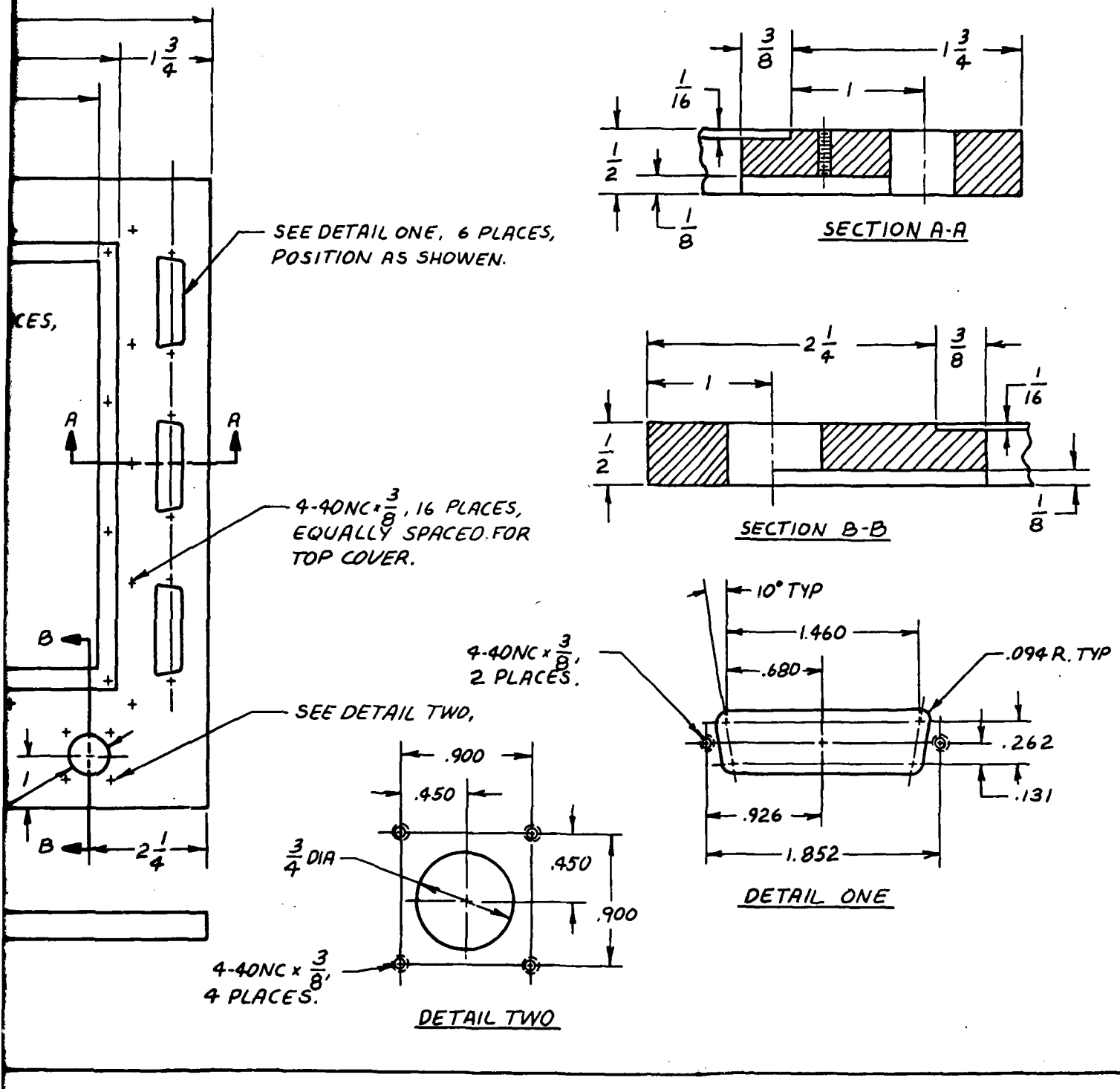


Fig. 23. Microcomputer Enclosure, Frame (Sheet 1 of 3)


1 3/4
SEE DETAIL ONE, 6 PLACES, POSITION AS SHOWEN.
CES,
A
A
4-40NC x 3/8, 16 PLACES, EQUALLY SPACED FOR TOP COVER.
B
SEE DETAIL TWO,
1
B
2 1/4
3/8
1 3/4
1/16
1
1/2
1/8
SECTION A-A
2 1/4
3/8
1
1/16
1/2
1/8
SECTION B-B
10° TYP
1.460
4-40NC x 3/8, 2 PLACES.
.680
.094 R. TYP
.262
.131
.926
1.852
DETAIL ONE
.900
.450
3/4 DIA
.450
.900
4-40NC x 3/8, 4 PLACES.
DETAIL TWO

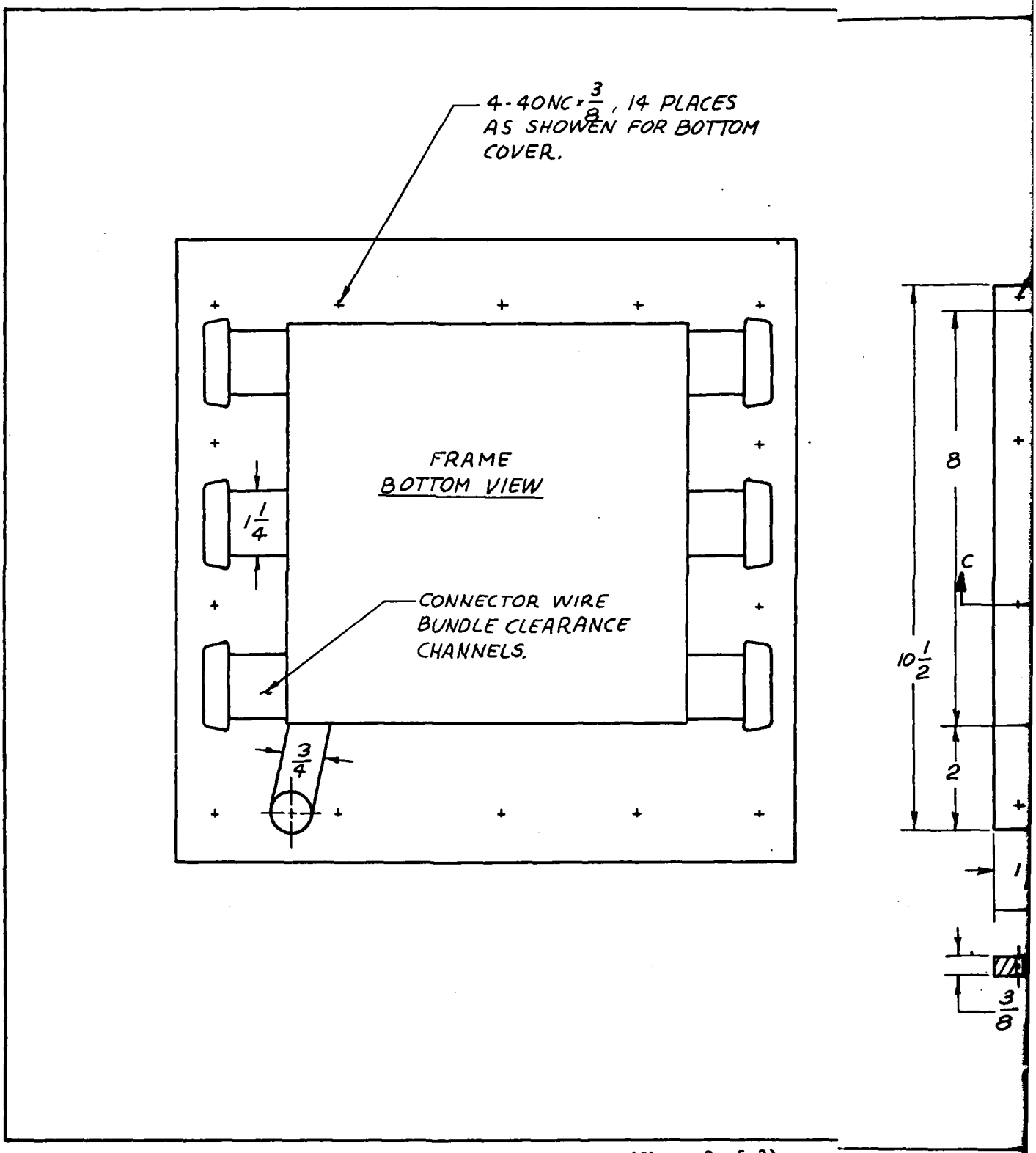
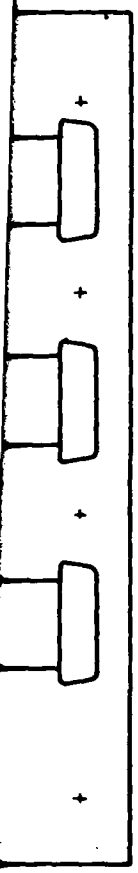
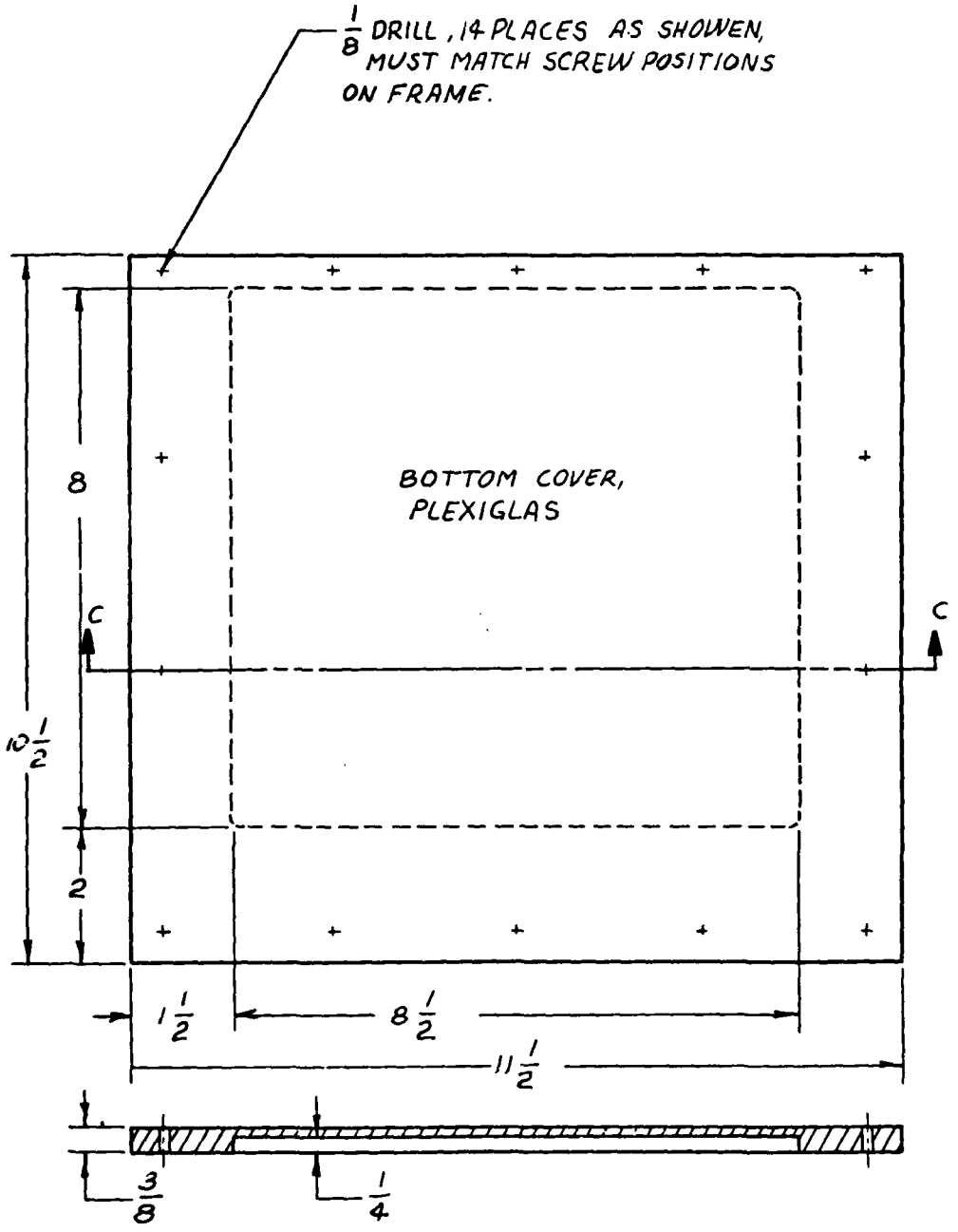


Fig. 23. Microcomputer Enclosure, Frame and Bottom Cover (Sheet 2 of 3)




1/8 DRILL, 14 PLACES AS SHOWEN, MUST MATCH SCREW POSITIONS ON FRAME.
BOTTOM COVER, PLEXIGLAS
C
C
8
10 1/2
2
1 1/2
8 1/2
11 1/2
3/8
1/4

SECTION C-C

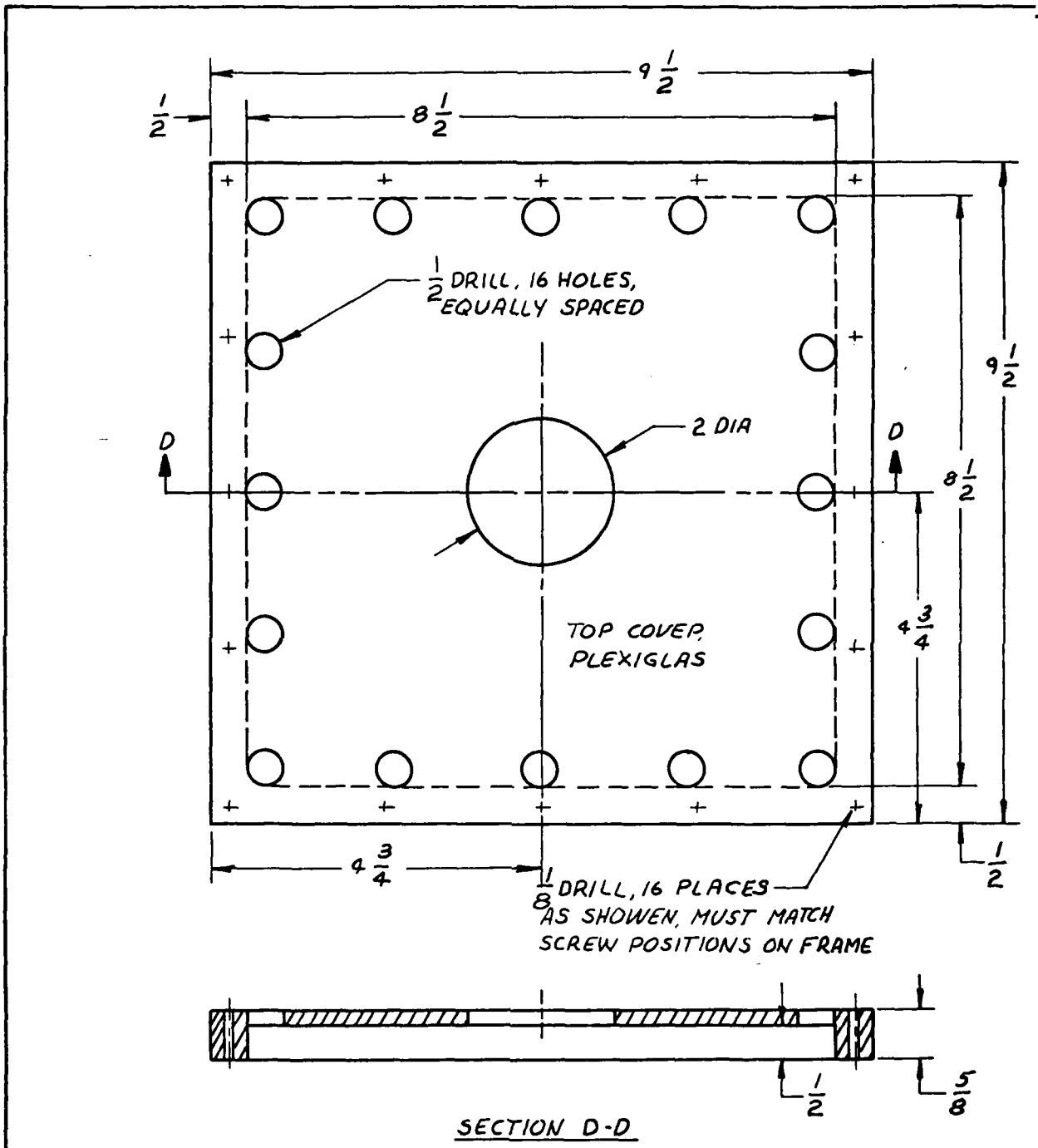
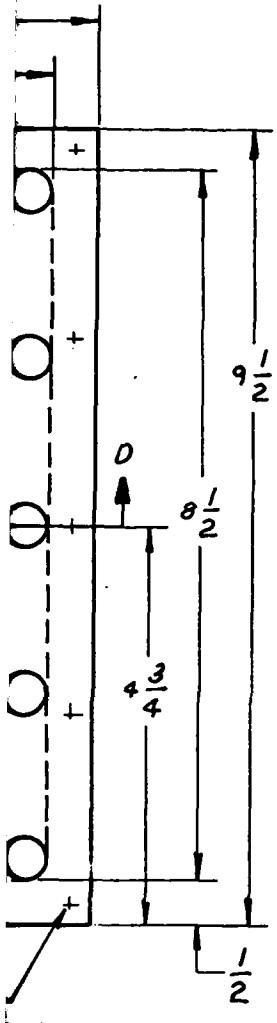
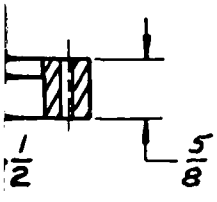


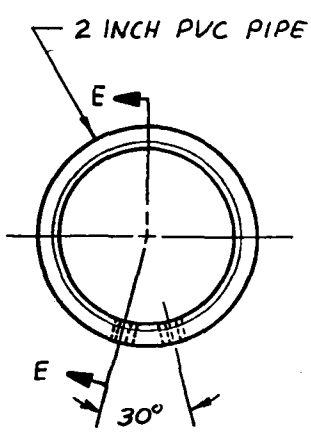
Fig. 23. Microcomputer Enclosure, Top Cover (Sheet 3 of 3)

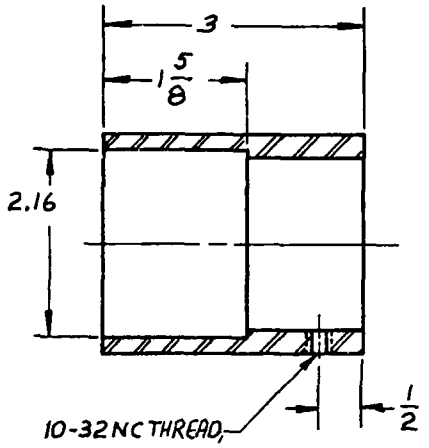


CH
FRAME



FAN HOUSING





SECTION E-E

NOTE:
FAN WAS SALVAGED FROM OLD HAIR DRYER, PARTS MDADE TO SUITE. FAN HOUSING IS GLUED TO TOP COVEP OVER 2" DIA HOLE.

Table II

Hard-wired I/O Port Addresses

	Control Address	Data Address
Counter Timer Circuit (CTC)		
Channel 0	ØØH	-
Channel 1	Ø1H	-
Channel 2	Ø2H	-
Channel 3	Ø3H	-
Parallel Input/output Controllers (PIO)		
PIO No. 1		
Channel A_1	Ø6H	Ø4H
Channel B_1	Ø7H	Ø5H
PIO No. 2		
Channel A_2	ØAH	Ø8H
Channel B_2	ØBH	Ø9H
PIO No. 3		
Channel A_3	ØEH	ØCH
Channel B_3	ØFH	ØDH
Serial Input/output Controller (SIO)		
Channel A	12H	10H
Channel B	13H	11H
Analog Data Acquisition System		
Channel Select and Start	1CH	-
Data Request	-	1EH
Sense Switch	-	14H

Analog Input Circuits

The analog circuits built for this project provide for seven inputs. The demonstration project uses three channels—a voltage follower for the accelerator position potentiometer, and two scaling amplifiers for the motor voltage and current measurements. The additional four channels were constructed for three tachometer inputs and a throttle position

potentiometer on the hybrid engine. Table III gives the channel assignments and design data.

Table III

Analog Input Amplifiers

Microcomponent channel (J4)	Input data	Amplifier input voltage range	Design converse factor
0	RPMs	0 to 1.2	4.6875×10^{-3} V/count
1	RPMs	0 to 1.2	4.6875×10^{-3} V/count
2	RPMs	0 to 1.2	4.6875×10^{-3} V/count
3	Accelerator position	0 to 5	1.9531×10^{-2} V/count
4	Volts	0 to 72	2.8125×10^{-1} V/count
5	Amps	0.0 to 0.200	1.5625 A/count
6	Throttle position	0 to 5	1.9531×10^{-2} V/count

All the amplifiers have a common signal ground and care should be taken to reference all measurements to the same ground point. The output voltage range of the amplifiers was designed to match the 0 to 5 volt input range of the data acquisition system. These amplifiers do not have input load resistors; thus, all unused inputs must be tied to signal ground to prevent the operational amplifier from saturating. Figure 24 shows the component layout and Figure 25 is the schematic diagram for the analog input amplifier.

Battery Controller

The battery controller is made up of the existing relay switches and diode network as shown in Figure 26, and the digital interface electronics. The digital interface circuitry contains a high current relay

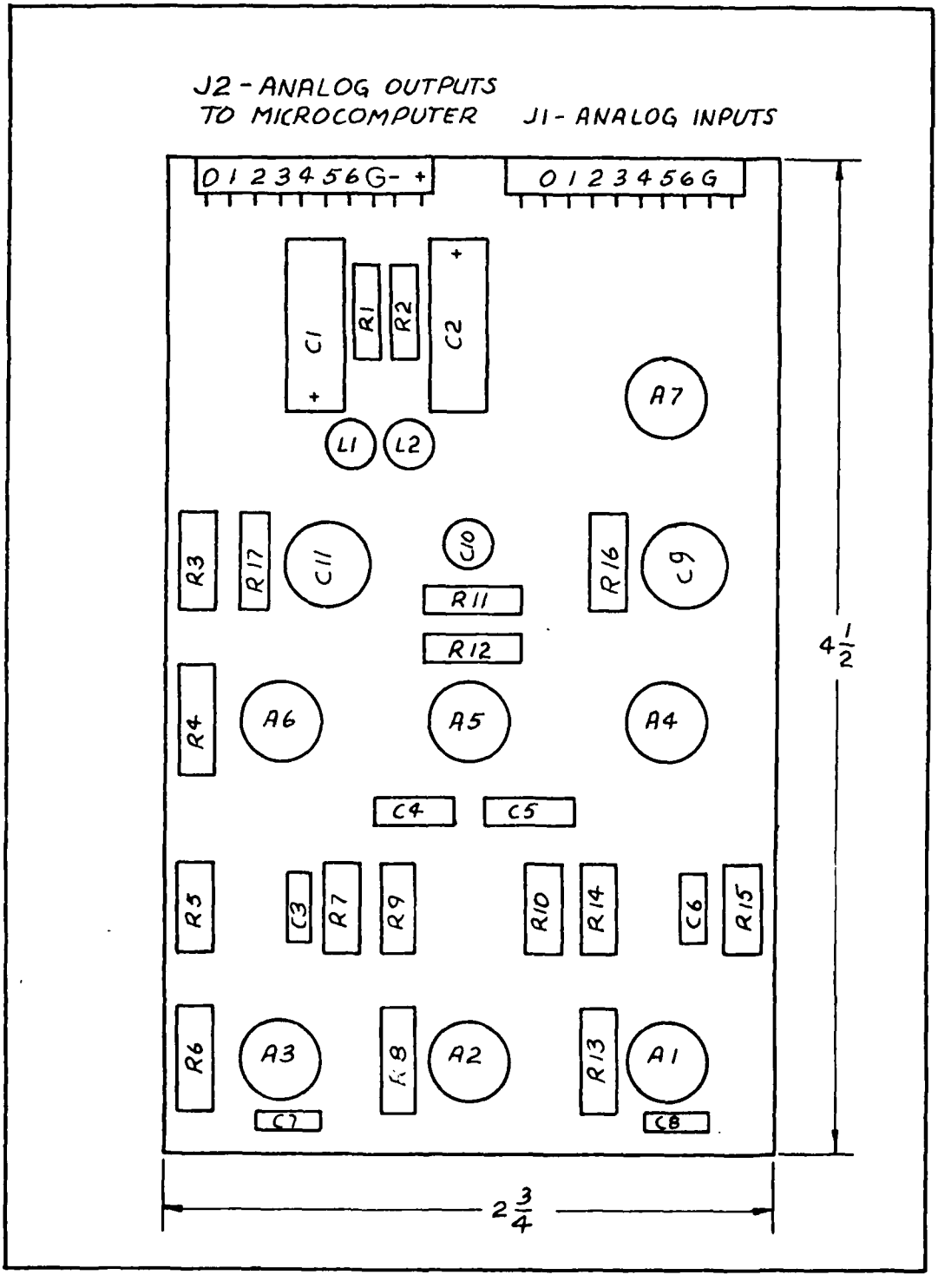


Fig. 24. Component Layout for Analog Input Amplifiers.

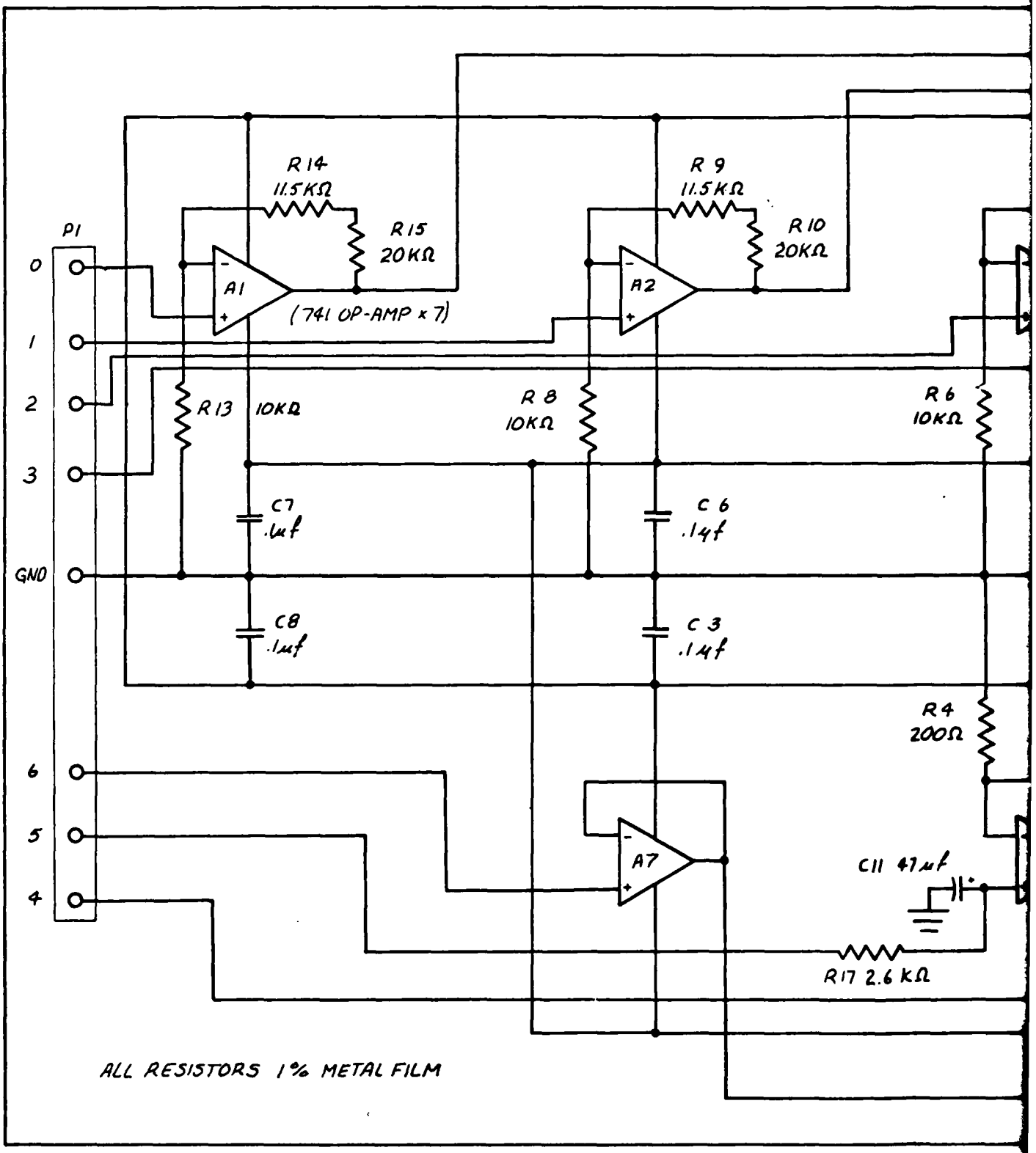
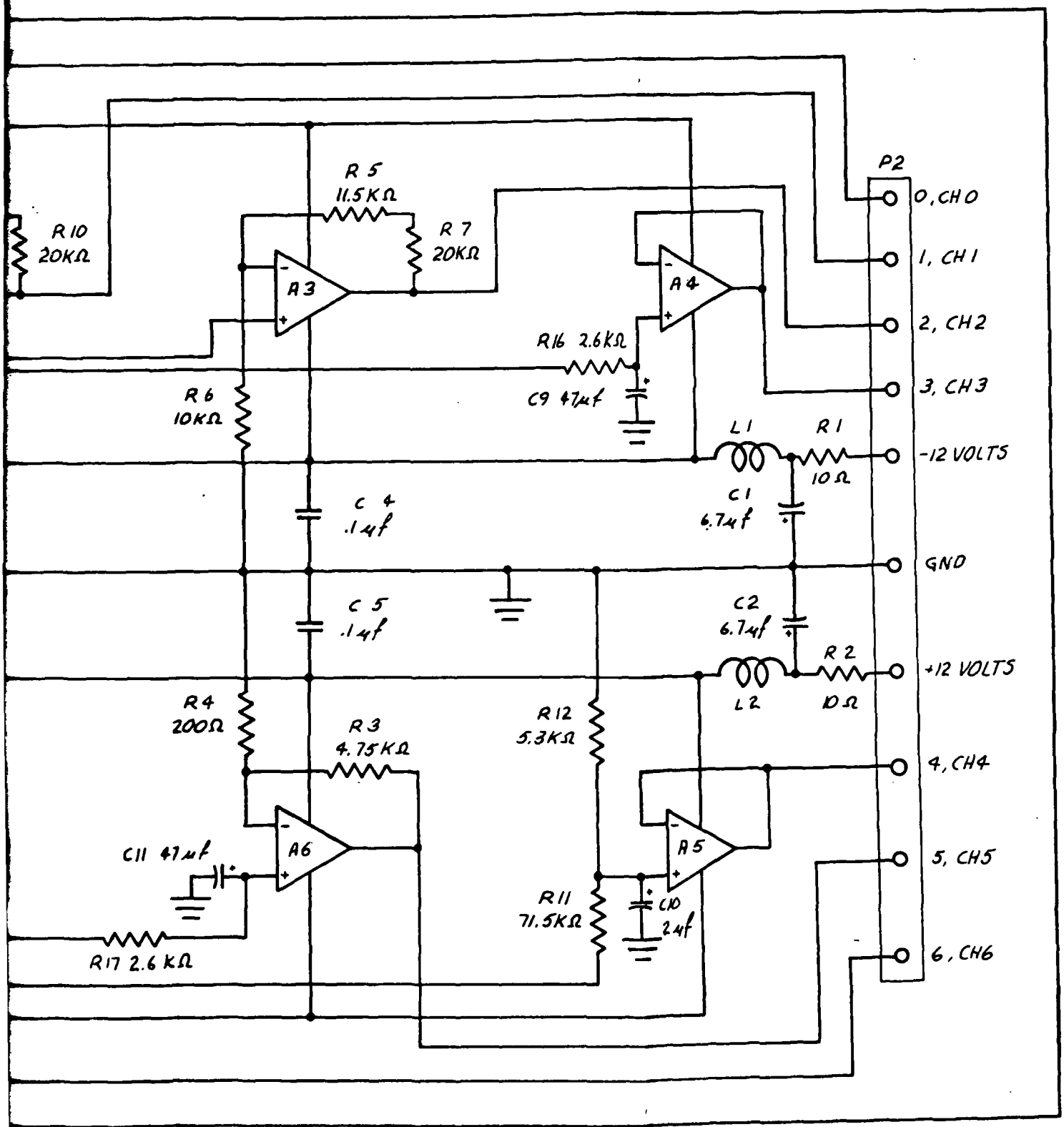


Fig. 25. Analog Input Amplifiers Schematic Diagram


P2
0, CH0
1, CH1
2, CH2
3, CH3
-12 VOLTS
GND
+12 VOLTS
4, CH4
5, CH5
6, CH6
R10 20KΩ
R5 11.5KΩ
R7 20KΩ
A3
A4
R16 2.6KΩ
C9 47μf
R6 10KΩ
L1
R1
10Ω
C1 6.7μf
C4 .1μf
C5 .1μf
C2 6.7μf
L2
R2 10Ω
R4 200Ω
R3 4.75KΩ
R12 5.3KΩ
A6
A5
C11 47μf
R11 71.5KΩ
C10 2μf
R17 2.6KΩ

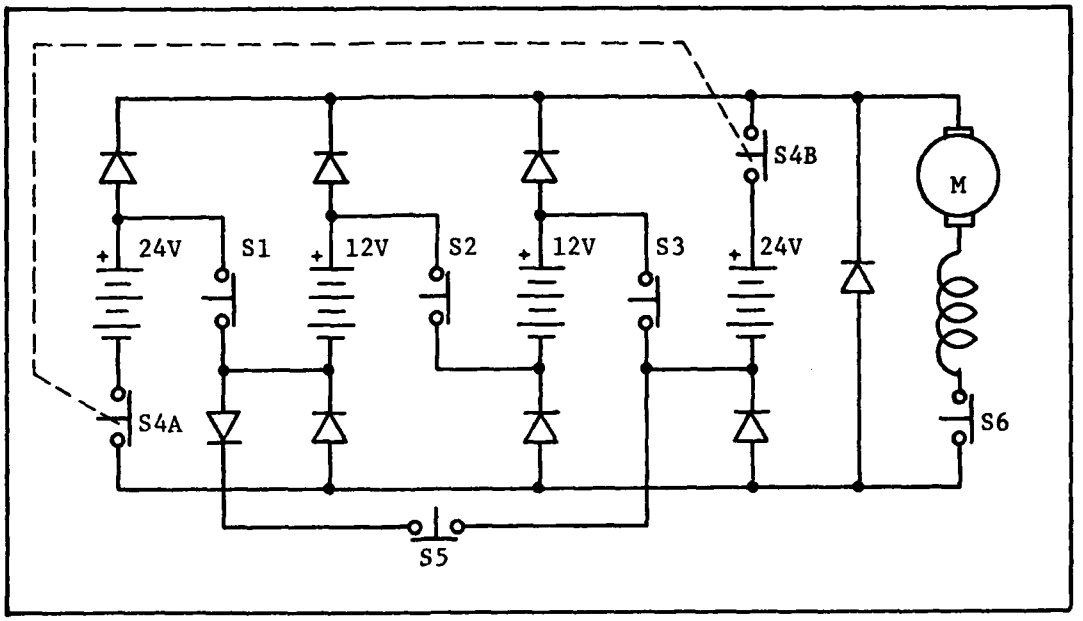


Fig. 26. Existing 5 Step Battery Switching System

driver for each switch in the battery switching network and a power interrupt circuit as shown in Figure 27. The relay drivers and power interrupt circuit have active high inputs but, because of the action of the power interrupt logic, the interface has a normally open relay switch state at turn-on. Figure 28 shows a component layout for the digital interface assembly.

Computer Power Supply

This power supply is the bench power supply used to power the computer when it is out of the test vehicle. The power supply was built with a Lambda LYT-W-5152 switching power supply as a base unit. Source voltages of -5 volts and ±12 volts were derived from the Lambda unit's ±15 volt supplies. The power supply schematic diagram is shown in Figure 29. The power supply is capable of providing voltages and currents shown in Table IV without loss of regulation or overheating.

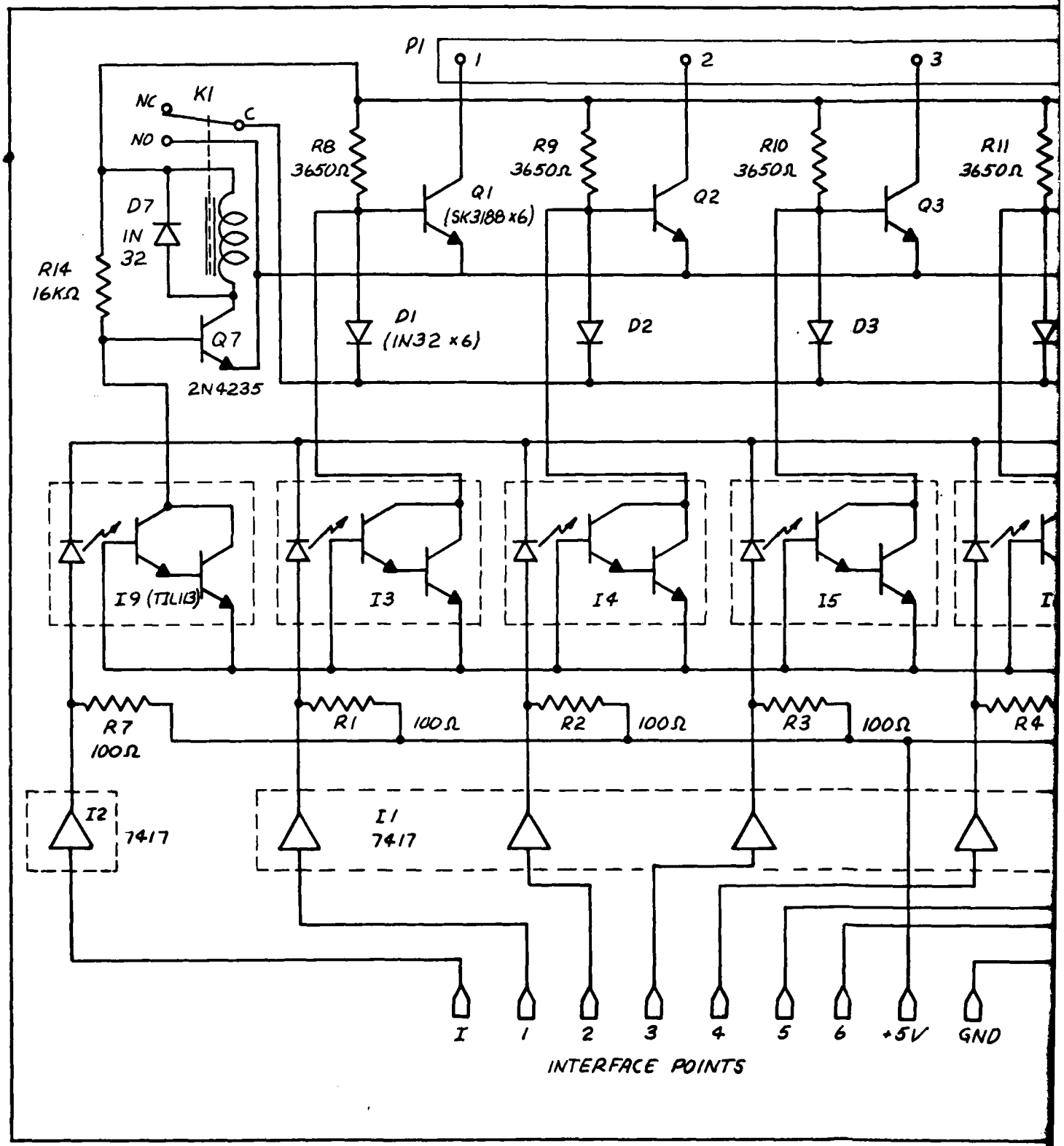
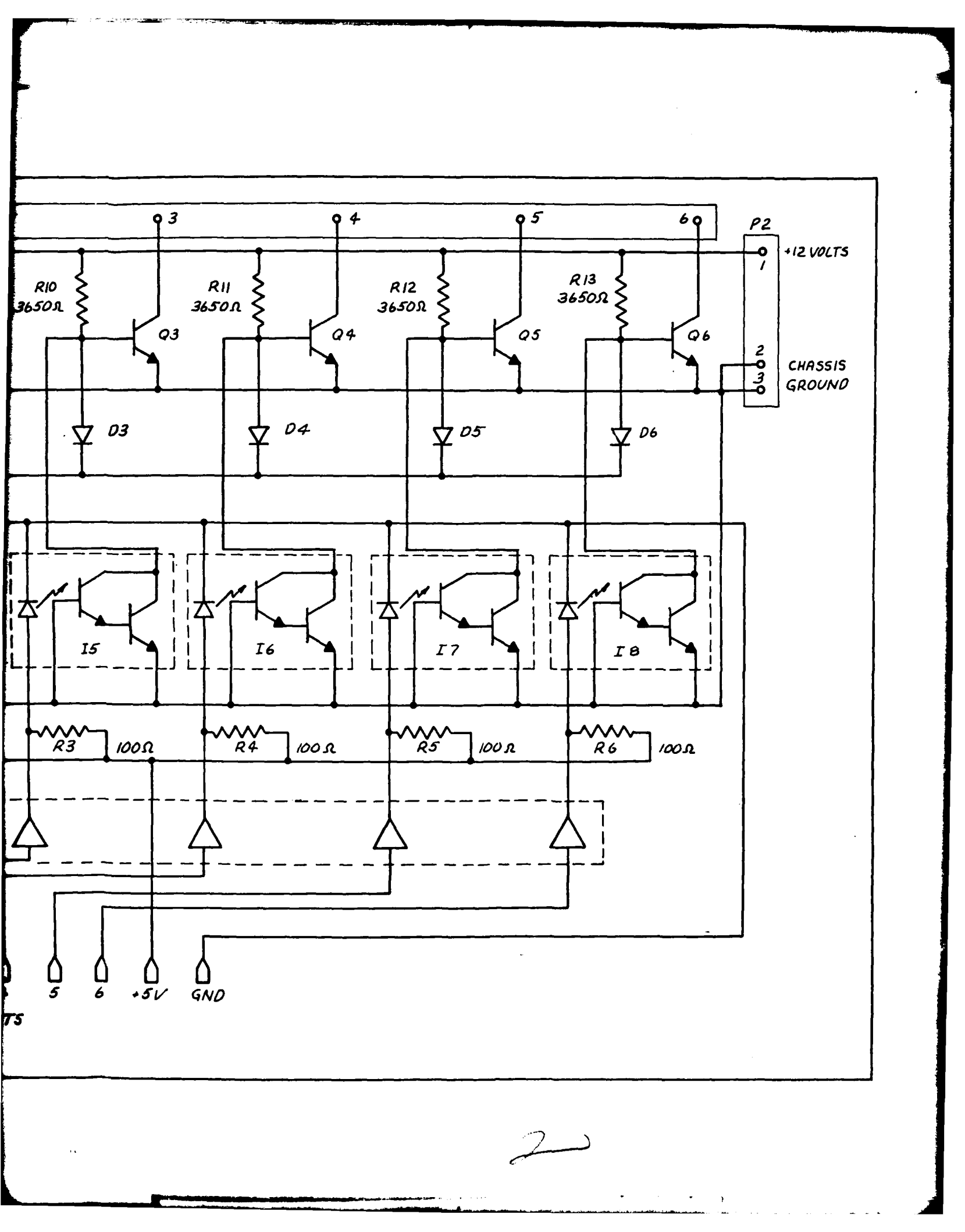


Fig. 27. Schematic Diagram of Digital Interface


3
4
5
6
P2
+12 VOLTS
1
R10
3650Ω
R11
3650Ω
R12
3650Ω
R13
3650Ω
Q3
Q4
Q5
Q6
2
CHASSIS
3
GROUND
D3
D4
D5
D6
I5
I6
I7
I8
R3
100Ω
R4
100Ω
R5
100Ω
R6
100Ω
5
6
+5V
GND
TS

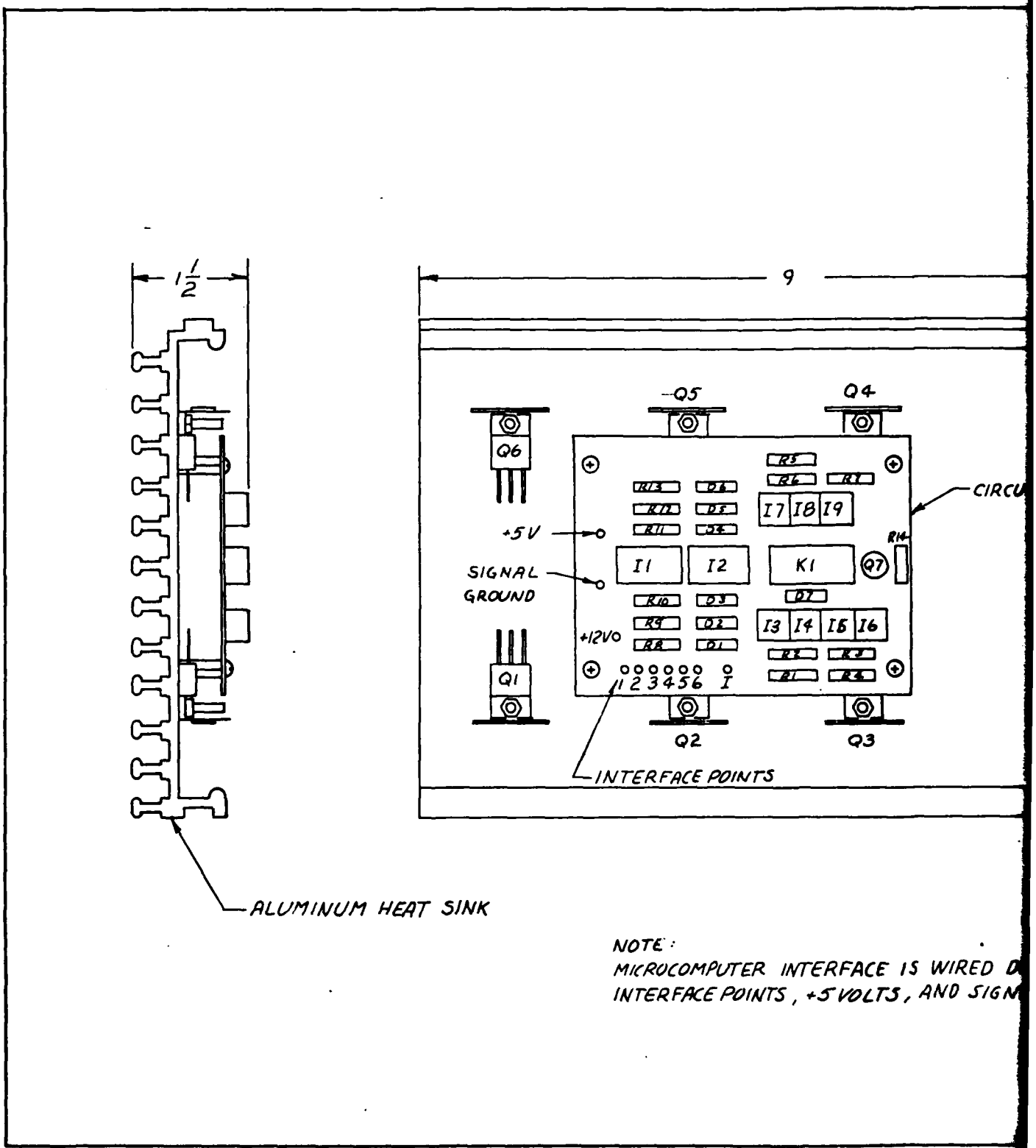
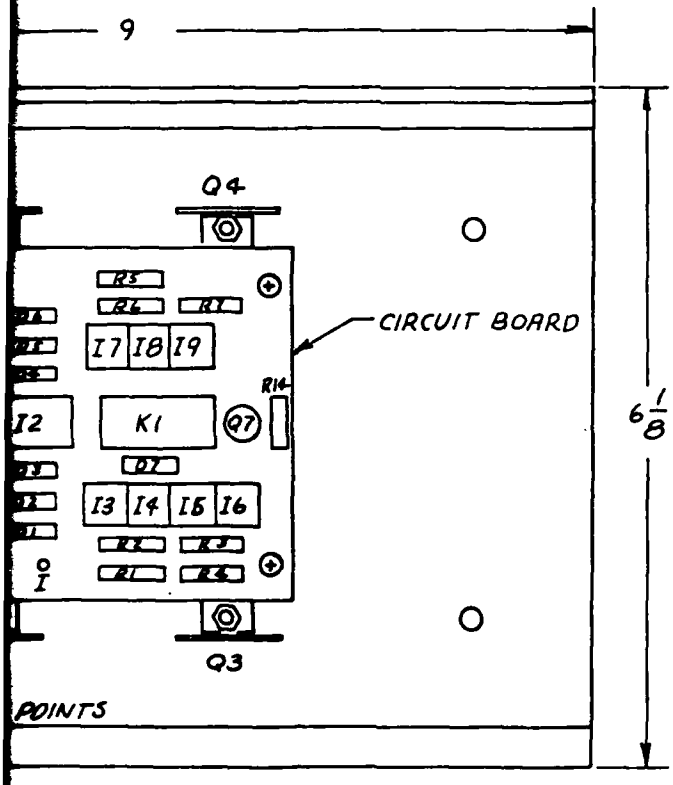
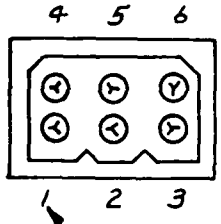


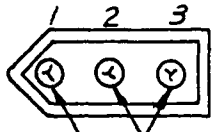
Fig. 28. Component Layout of Digital Interface Assembly



P1 - RELAY SWITCH PLUG



P2 - POWER SUPPLY PLUG



PUTER INTERFACE IS WIRED DIRECTLY TO
E POINTS, +5 VOLTS, AND SIGNAL GROUND.

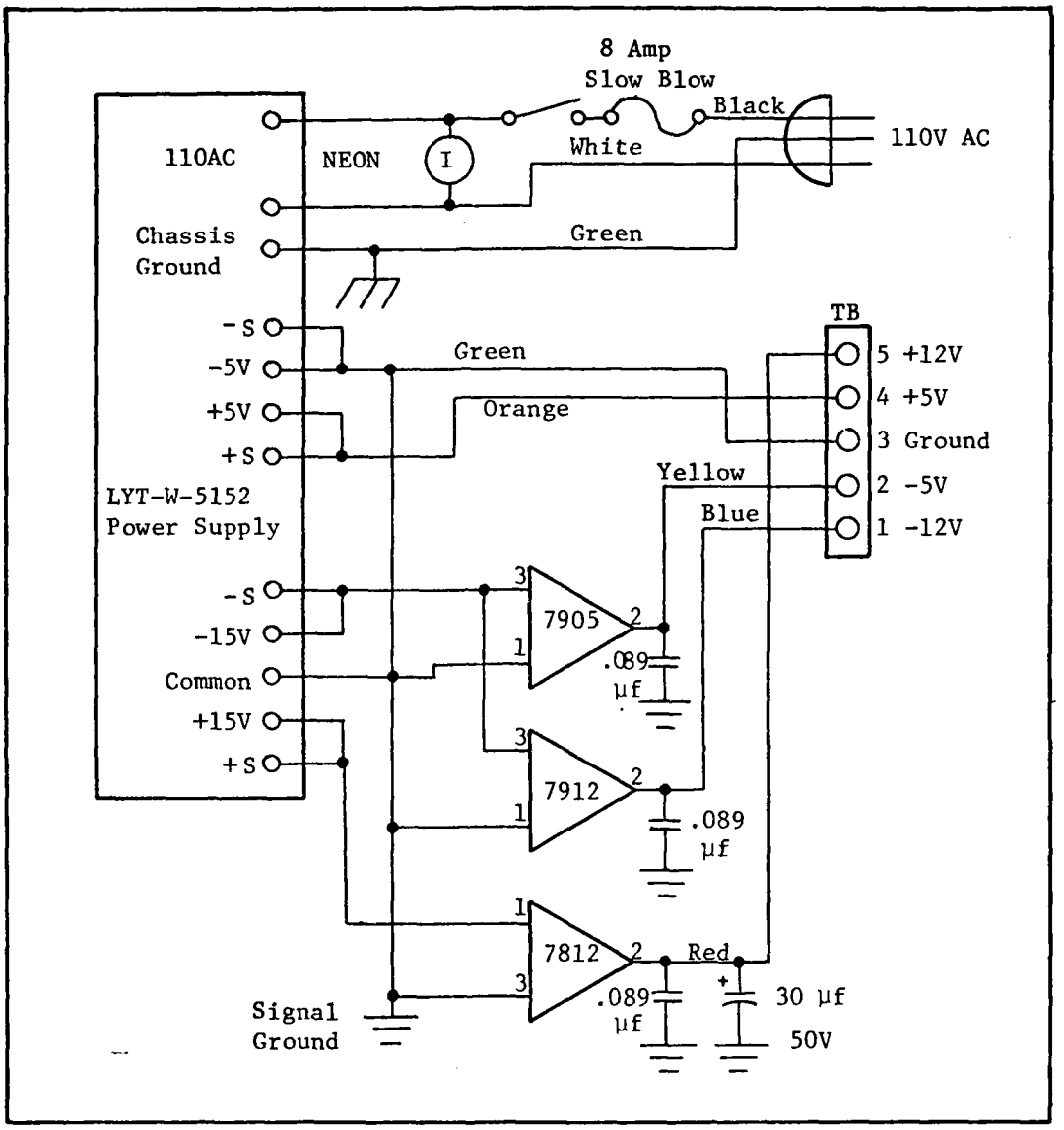


Fig. 29. Electrical Diagram of Power Supply

Test Vehicle Installation

The test vehicle installation drawing is shown in Figure 30. The static inverter AC power supply is being used to power the bench computer power supply and a CRT terminal. This is a temporary arrangement used to shorten development time so the microcomputer controller could be

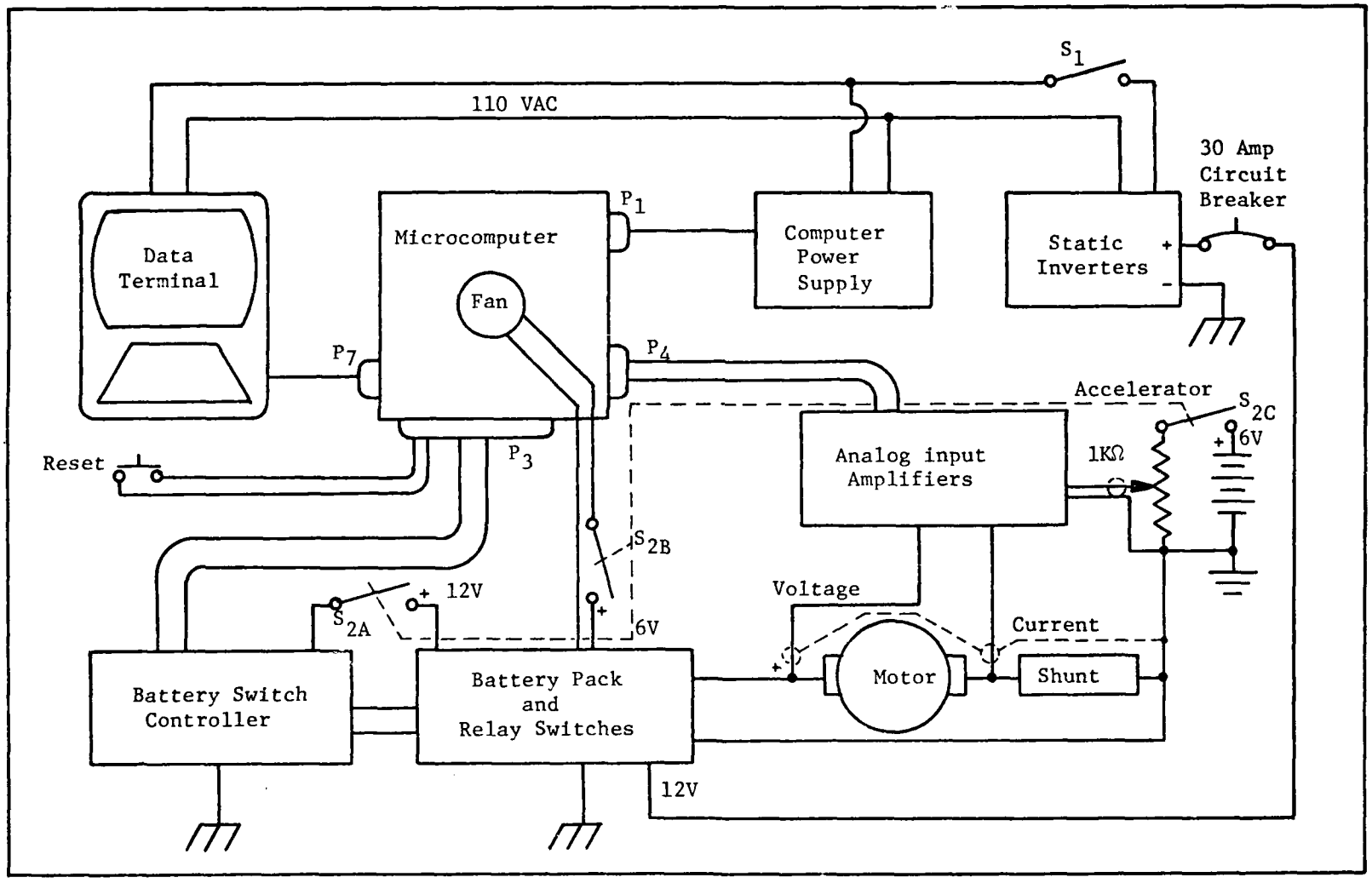


Fig. 30. Electric Vehicle Control System, Test Vehicle Installation

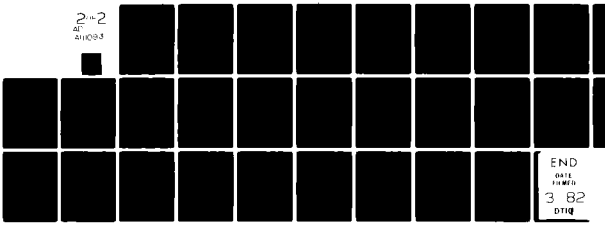
Table IV

Power Supply Voltage

Output Voltage	Maximum Current at 40°C
5 ± 5% Volts	14 Amps
±12 ± 5% Volts	0.6 Amps
-5 ± 5% Volts	0.6 Amps

demonstrated in the test vehicle. In the installation of the system it was imperative that the bench power supply and the CRT terminal be isolated from the test vehicle chassis ground. This ground plane isolation is required by the data acquisition system and to avert a high current short in the ground plane.

AD-A111 093 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/6 9/2
ELECTRIC VEHICLE POWER CONTROLLER.(U)
DEC 81 J C FRAZIER
UNCLASSIFIED AFIT/GAE/AA/81D-1 NL



Appendix B

Standard ROM Monitor

The standard ROM monitor is adapted from an AFIT Z80 based ROM monitor. This monitor is an operating system with comprehensive DEBUG and I/O handling capabilities. It contains all the tools necessary to fully debug both hardware and software as well as support the I/O used by the Electric vehicle controller.

Configuration

The monitor is presently configured to support a CRT data terminal from an asynchronous serial port operating full duplex, with 7 bit character at 600 baud from jack 7 and a reader/punch device using the same format operating at 300 baud from jack 6 of the microcomputer. The reader/punch port was connected to a software development computer using a crossover cable. The software development computer then transferrd program files to the microcomputer for debugging and testing.

The monitor software uses both a hardware switch (sense switch) and a software controlled register to control the microcomputers I/O configuration. This is covered in more detail in a later section.

Commands

Table V is a list of commands for the standard ROM monitor. Certain input and output commands have been adapted for use by the monitor to access the Electric Vehicle demonstration program.

The Ignition (I) command starts the control routines in the demonstration program. This command must be initiated before the vehicle will respond to accelerator inputs.

Table V

Monitor Commands

Command	Description
A	Assign reader, punch, console or list device options from the console
B	BYE - system shut down, recover control with a control H.
C	Compare the contents of memory with the reader input and display any differences
D	Display the contents of any defined memory are in hexidecimal notation, D <Starting address>, <Ending address> (Cr)
E	End of file statement generator
F	Fill any defined area of memory with a constant F <Starting address>, <Ending address>, <Hex value> (Cr)
G	Goto an address and execute with breakpointing G <Starting address>, <Breakpoint> (Cr) or G, <Next Breakpoint> (Cr)
H	Hex math - gives the sum and difference of two hexidecimal numbers, H <First value>, <Second value> (Cr)
I	Ignition - begins electric vehicle demonstration program
J	Justify memory - a non-destructive test for hard memory failures
K	*User defined (not used)
L	Load a binary file
M	Move a defined memory area to another starting address, M <Starting address>, <Ending address>, <New starting address> (Cr)
N	Nulls to the punch device
O	Output Energy - returns the elapse and energy consumed in Hex (see description below)

Monitor Commands - continued

Command	Description
P	Put ASC II characters into memory from the keyboard P <Starting address> (Cr), end entry control U
Q	Query I/O ports - may output or input any value to or from any I/O port output to port QO <Port address>, <Hex value> (Cr), input from port QI <Port address> (Cr)
R	Read a hex file. Performs check run, relocating, offsetting, etc. R (Cr, Start Reader)
S	Substitute and/or examine any value at any address (in hex) S <Starting address> (Sp or Cr)
T	Type the contents of a defined memory block in their ASC II equivalent T <Starting address>, <Ending address> (Cr)
U	Unload a binary tape to the punch device
V	Verify the contents of a defined memory block against that of another block and display the differences
W	Write a checksummed hex file to the punch device
X and X'	Examine and or modify any or all of the Z80 CPU registers
Y	"Y is there" - search memory for defined byte strings and display all their address locations Y <First hex character>, <Second>, etc. <Last> (Cr)
Z	"Z end" - locate and display the highest address in memory

The Output Energy (O) command samples the current values of the sample counter and energy accumulator, and displays the elapsed time, energy, and exponent in hexadecimal. The elapsed time has units of seconds. The energy and exponent represent the amount of energy used

during the time period given by the elapsed time. Energy in this calculation has units of joules and is expressed in scientific notation as shown in Figure 31.

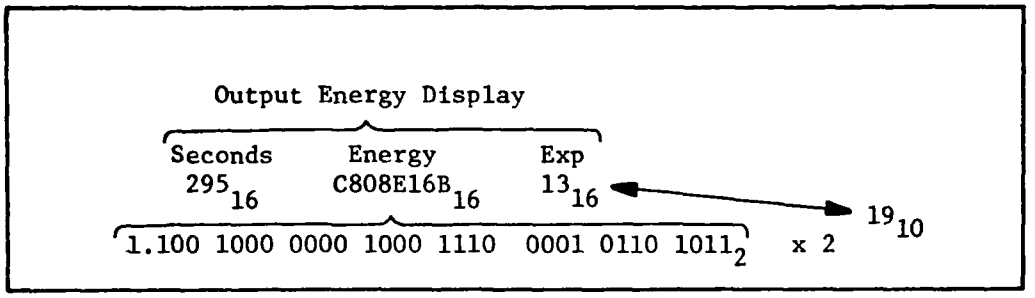


Fig. 31. Consumed Energy Expressed in Scientific Notation

Once the energy is expressed in the form shown in Figure 31, it is a simple matter to compute the decimal equivalent with a scientific calculator by summing all the powers of two for which there is a corresponding one in the binary expression as shown in the following example:

$$2^{19} + 2^{18} + 2^{15} + 2^7 + 2^3 + 2^2 + 2 = 819342 \text{ Joules.}$$

Note that the negative powers of two have been truncated for convenience and this expression can be truncated at the decimal point for order of magnitude checks.

Monitor SIO Modification

This version of the standard ROM monitor takes advantage of technical improvements in serial circuit devices by using the Z80 Serial input/output controller. This is a two channel device versus the commonly used UART which is a single channel device. Figure 32 shows the SIO initialization source code. The CTC channel initialization is also shown. The CTC channels are used to generate the baud rate clock for

```
SIOTBL: DB      018H ; CHANNEL RESET
        DB      014H ; POINTER 4, RESET EXT/STATUS INTERUPTS
        DB      04FH ; ASYNC, EV PARITY, 2 SB, X16 SAMPLE
        DB      03   ; POINTER 3
        DB      041H ; RX ENABLE, AUTO ENABLE, 8 BIT CHAR
        DB      05   ; POINTER 5
        DB      0AAH ; REQUEST TO SEND, TX ENABLE, 8 BIT CHAR, DTR
        DB      011H ; POINTER 1, RESET EXT/STATUS INTERUPTS
        DB      00   ; EXT INT ENABLE, TX & RX INT DISABLED
;
;       LET US BEGIN
;
BEGIN:  MVI     A,015H ; INITILIZE THE HARDWARE
        OUT     01     ; CTC CHANNEL ONE
        OUT     02     ; CTC CHANNEL TWO
        MVI     A,0DH  ; CTC TIME CONSTANT  600 BAUD
        OUT     01
        MVI     A,1AH  ;CTC TIME CONSTANT  300 BAUD
        OUT     02
        MVI     A,0FH
        OUT     0FH    ;SET PIO #3 CH B TO OUTPUT FOR IOBYT
        MVI     B,09H  ; NUMBER OF BYTS TO XFER TO SIO
        LXI     H,SIOTBL ; TABLE POINTER
        MVI     C,CRTS ; SIO CHANNEL A CONTROL PORT
;       OTIR
        DB      0EDH,0B3H ; LOAD THE TABLE
        LXI     B,0913H ;TABLE LENGTH/SIO CONTROL CH B
        LXI     H,SIOTBL
;       OTIR
        DB      0EDH,0B3H ; LOAD THE TABLE
```

Fig. 32. SIO and CTC initialization instructions

the SIO channels. Refer to the Zilog technical manuals for detailed initialization instructions (Ref. 4, 11).

Sense Switch

The sense switch initializes the hardware I/O configuration of the microcomputer. This switch is read every time the system is reset and its value is stored in the IOBYT (PIO port ØDH). The sense switch bit definitions and device settings are given in Table VI. Because of the special purpose function of the microcomputer, the monitor has been

Table VI

Sense Switch

Bit Definition		Device Settings	
Bits	Description	Device	Setting
0-1	Console Device	Teleprinter	00
2-3	Reader Device	CRT	01
4-5	Punch Device	Cassett/Batch	10
6-7	List Device	User Defined	11

modified not to accept user defined or cassette devices. Batch inputs and output can only be used when the list device is defined as a teleprinter or CRT.

Appendix C

Demonstration Software

The demonstration software is made up of control and utility routines. The utility and control routines perform the handshaking operation with the A/D converter, multiple word length arithmetic, initialization of the control system, and vehicle speed control.

The Analog to Digital Input Subroutine (ADIO) processes analog measurement calls from the user program. It is designed specifically for use with the ADC-0816, 16 channel, 8 bit Monolithic Data Acquisition System and performs the software/hardware handshaking operation. Figure 33 is a flow diagram for this subroutine. To call the subroutine, the channel number that will be measured must be prestored in the Z80 CPU 'E' register. The result of the measurement is returned to the 'E' register when the measurement is complete.

The multiple word length arithmetic set uses memory vectors to perform calculations. A memory vector is a set of adjacent words where the operands are stored. These vectors are passed to the arithmetic routines by means of pointers. The pointers are the memory address of the least significant byte (LSB) in the operand. Operand word significance is assigned by ascending memory address. The operand word length minus one word must also be stored in the pointer as shown in Figure 34.

The arithmetic routines are a software multiply, extended precision summer, and the energy calculator. The software multiply (MLTPY) uses a recusive method where as the multiplier is shifted right and the multiplicand is added to an accumulator if a carry bit is generated. The multiplicand is then shifted left and continues until the multiplier is zero, as indicated in Figure 35. This routine must be supplied with

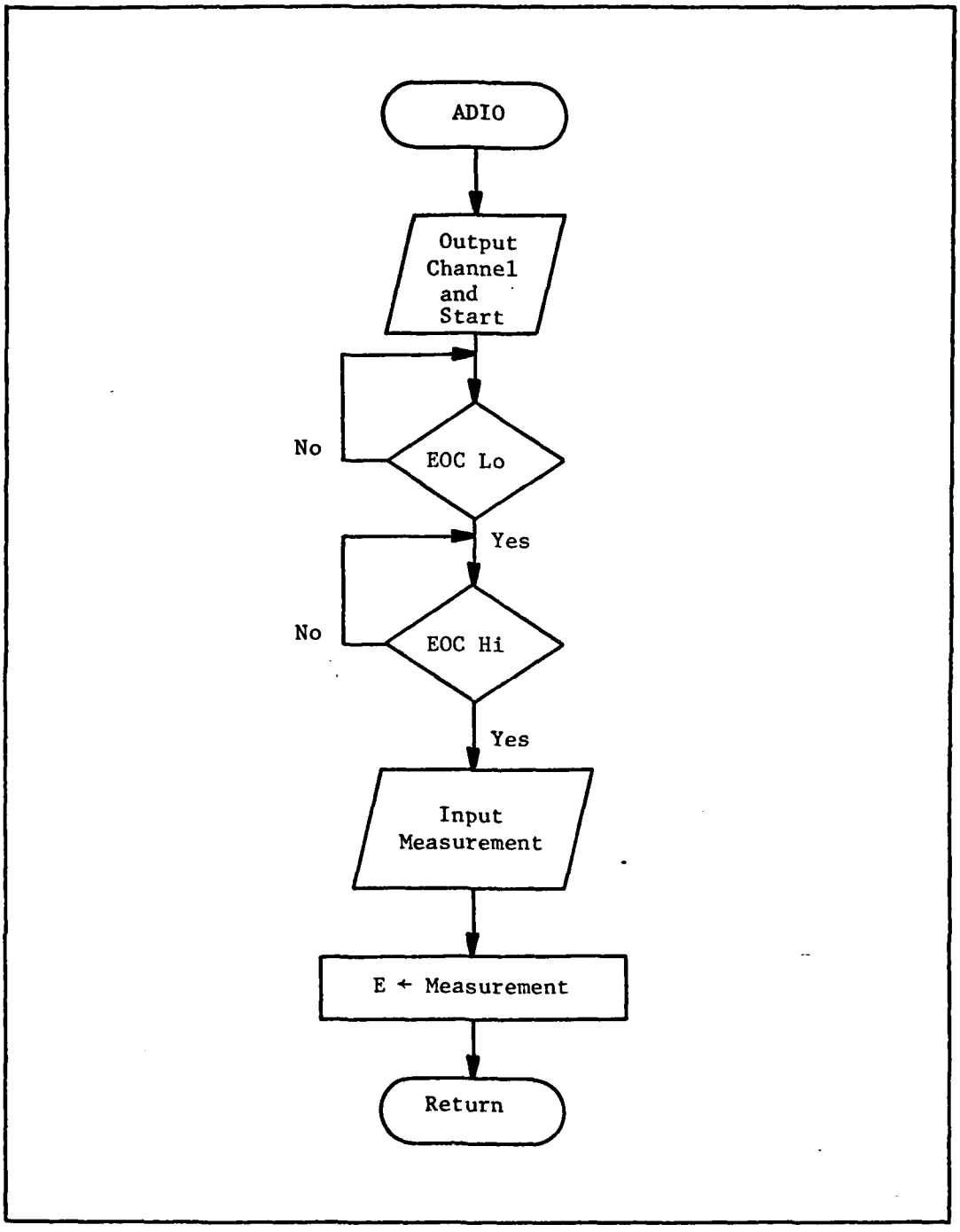


Fig. 33. Analog to Digital Input Subroutine Flow Chart

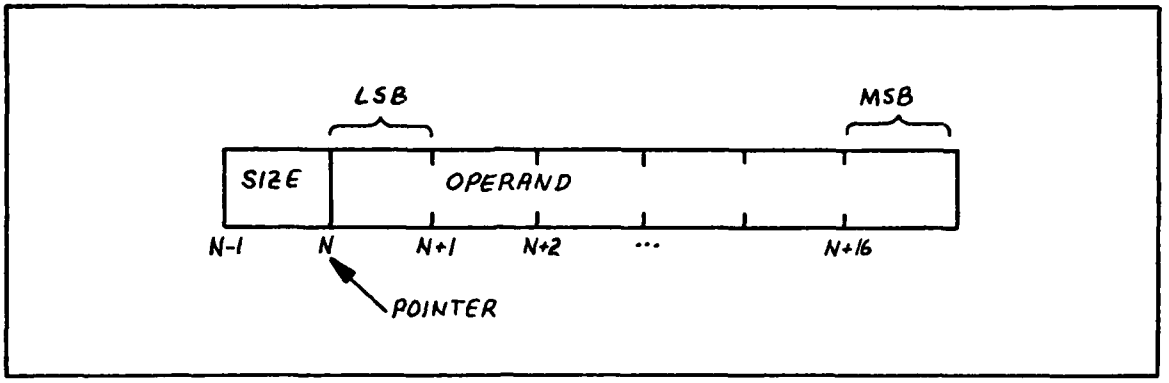


Fig. 34. Memory Vector

three pointers before it can be called, the multiplier point in the 'IX' register, the multiplicand pointer in the 'IY' register, and the result pointer stored in memory location Result. The routine returns the product in the result vector. This routine also sets the Z80 Zero flag according to the contents of the Result vector.

The software multiply uses three subroutine calls to perform its computation—a right shift, a left shift, and a sum. The right shift subroutine (RShift) shifts all the bits in the operand vector one place to the right through carry, as in the Z80 instruction shift operand right logical (SRLm). Before this routine is called the vector pointer must be in the 'IX' register. The routine returns the vector shifted right and the flag register as indicated by the SRLm instruction. The subroutine flow diagram is shown in Figure 36.

The left shift subroutine (LShift) shifts all the bits in the operand vector one place to the left through carry, as in the Z80 instruction shift operand left arithmetic (SLAm). Before this routine is called the vector pointer must be in the 'IY' register. The routine returns the vector shifted left and the flag register set as indicated by the SLAm

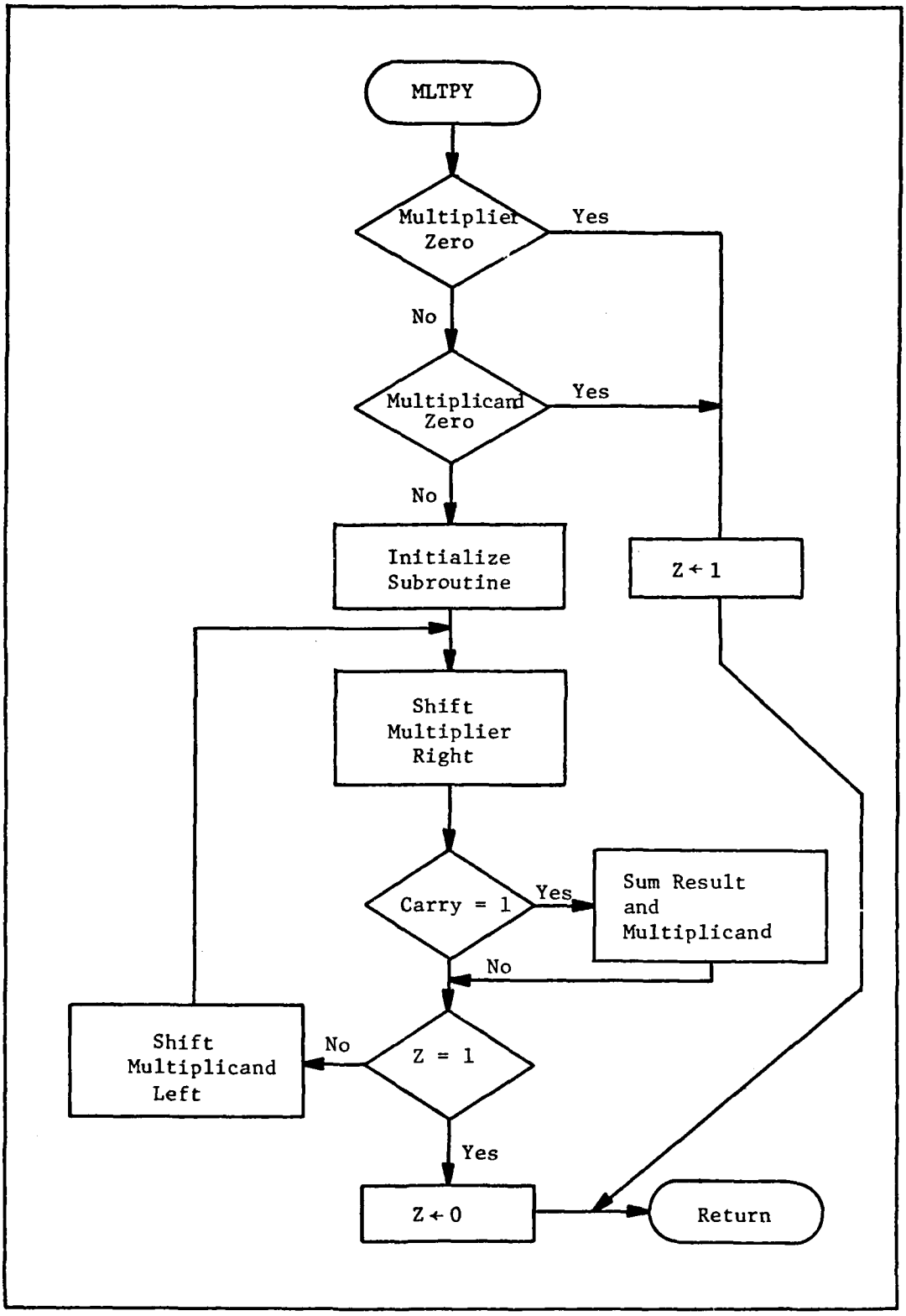


Fig. 35. Multiplication Subroutine (MLTPY) Flow Chart

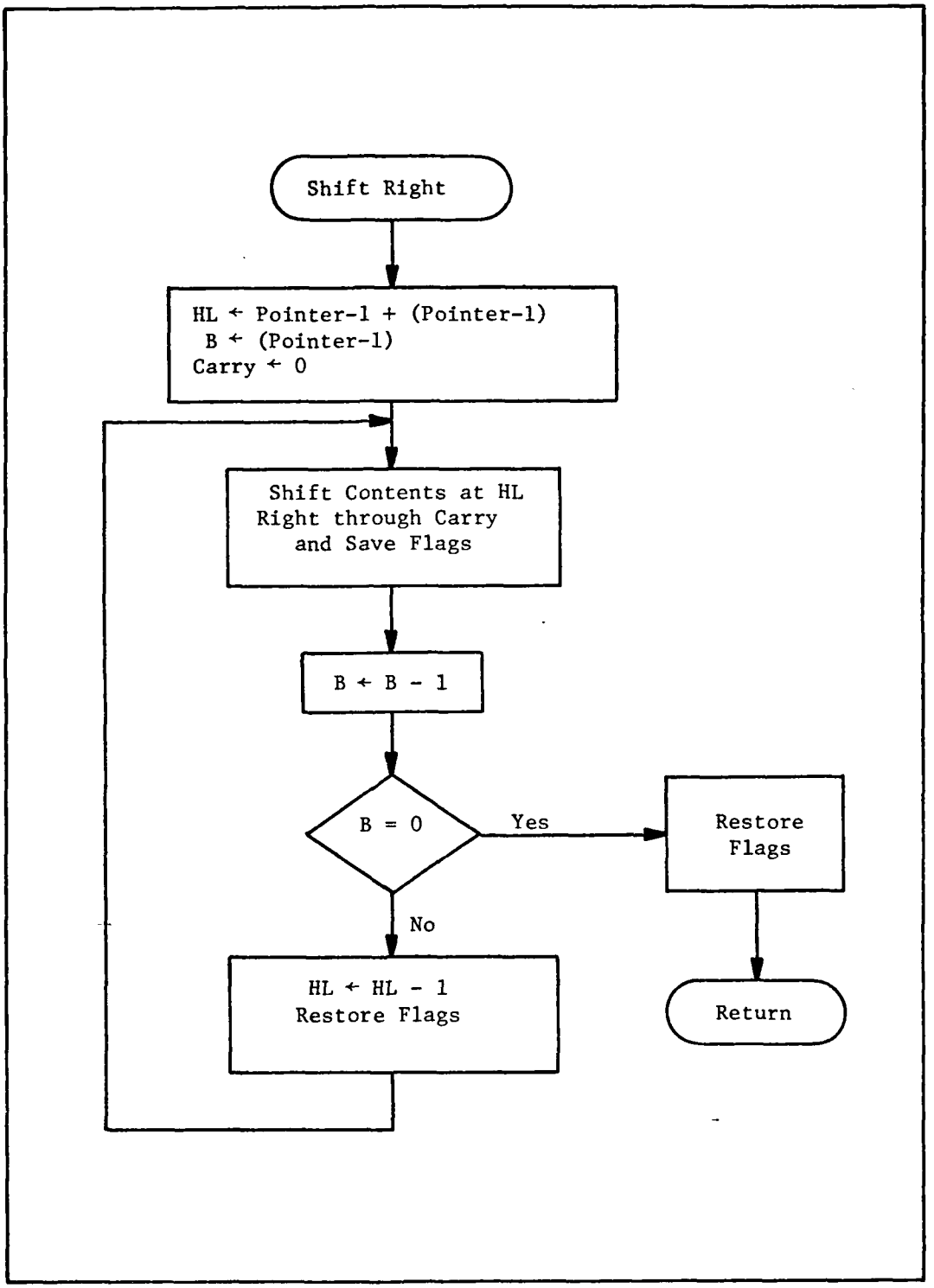


Fig. 36. Vector Shift Right (RSHIFT) Flow Chart

instruction. The subroutine flow diagram is shown in Figure 37.

The summation subroutine (SUM) adds two vectors. To call this routine the accumulator pointer must be stored in the 'IX' register and the new value pointer in the 'IY' register. Also, the accumulator vector must be larger than the new value vector. This routine returns the sum of the two vectors in the accumulator and, if the sum outgrows the accumulator, this routine concatenates another word of memory to the accumulator vector and adjusts its size. Figure 38 shows the flow diagram for this subroutine.

The energy calculator is a two part operation which uses the summation routine. The first part is a numerical integration routine taking place inside the control loop. The second part converts the result of the numerical integration into engineering units and displays them. These operations make full use of the utility routines described above and the monitor. Their function is discussed in chapter 4.

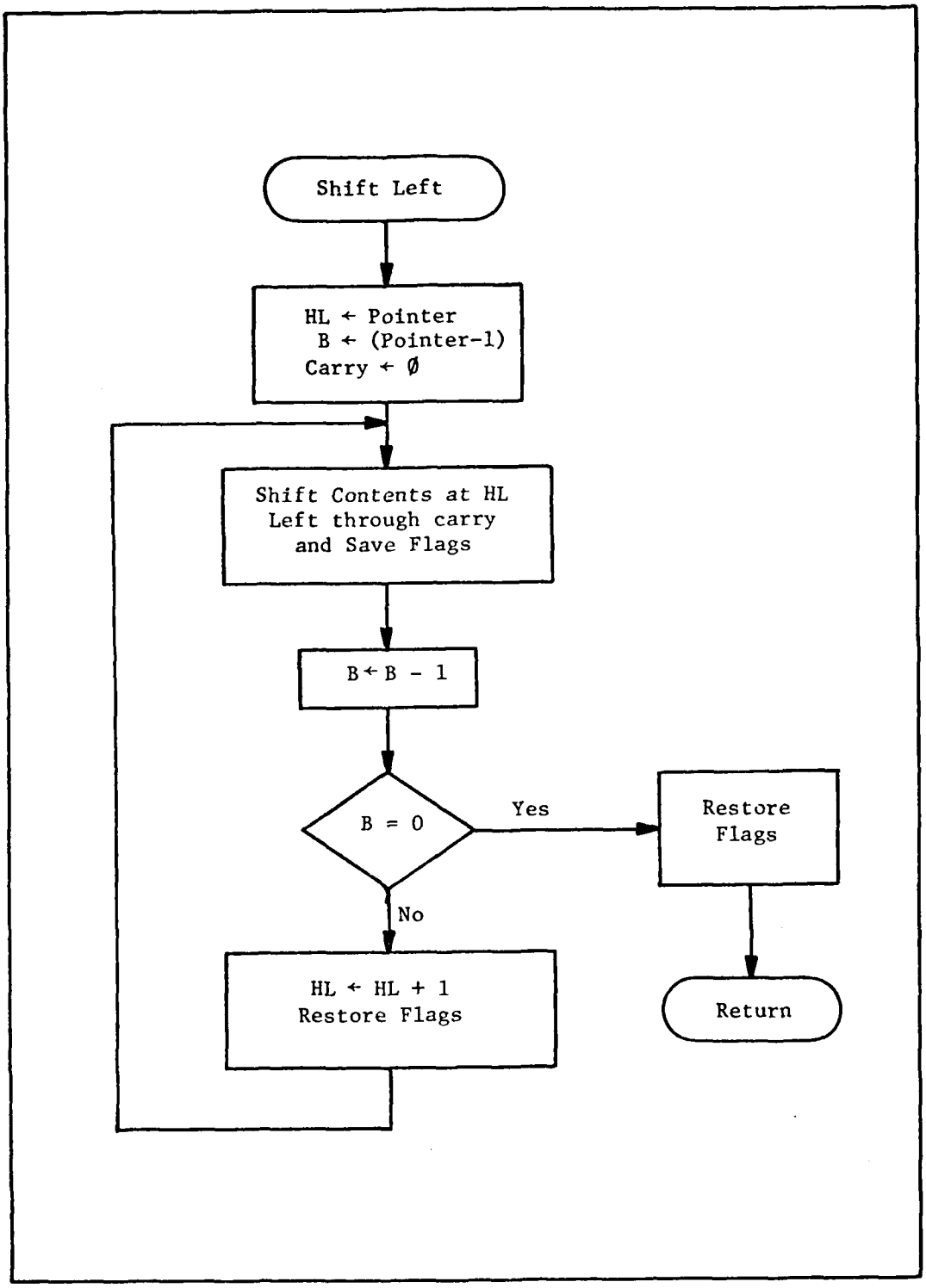


Fig. 37. Vector Shift Left (LSHIFT) Flow Chart

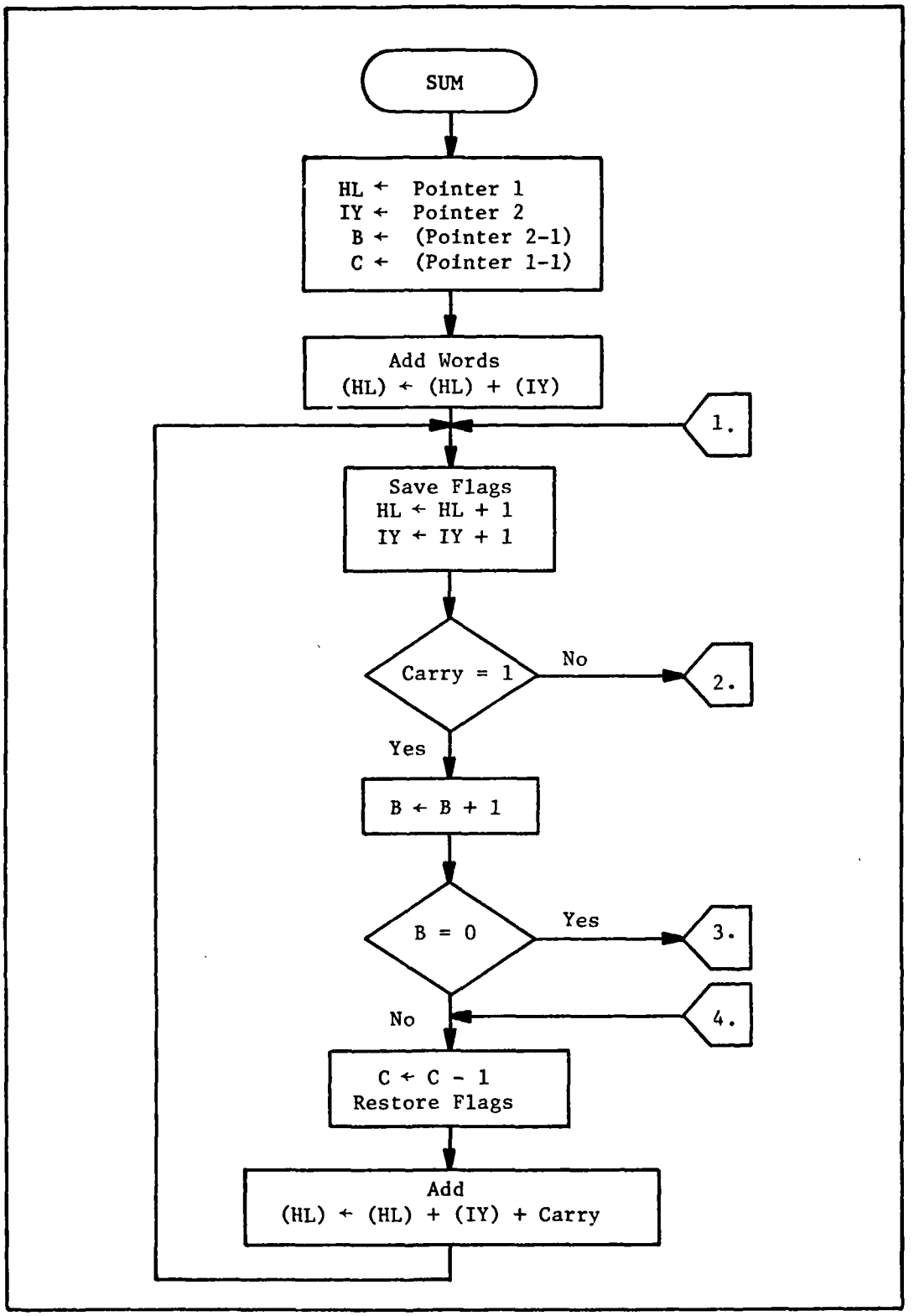


Fig. 38. Vector Summation (SUM) Flow Chart (Sheet 1 of 2)

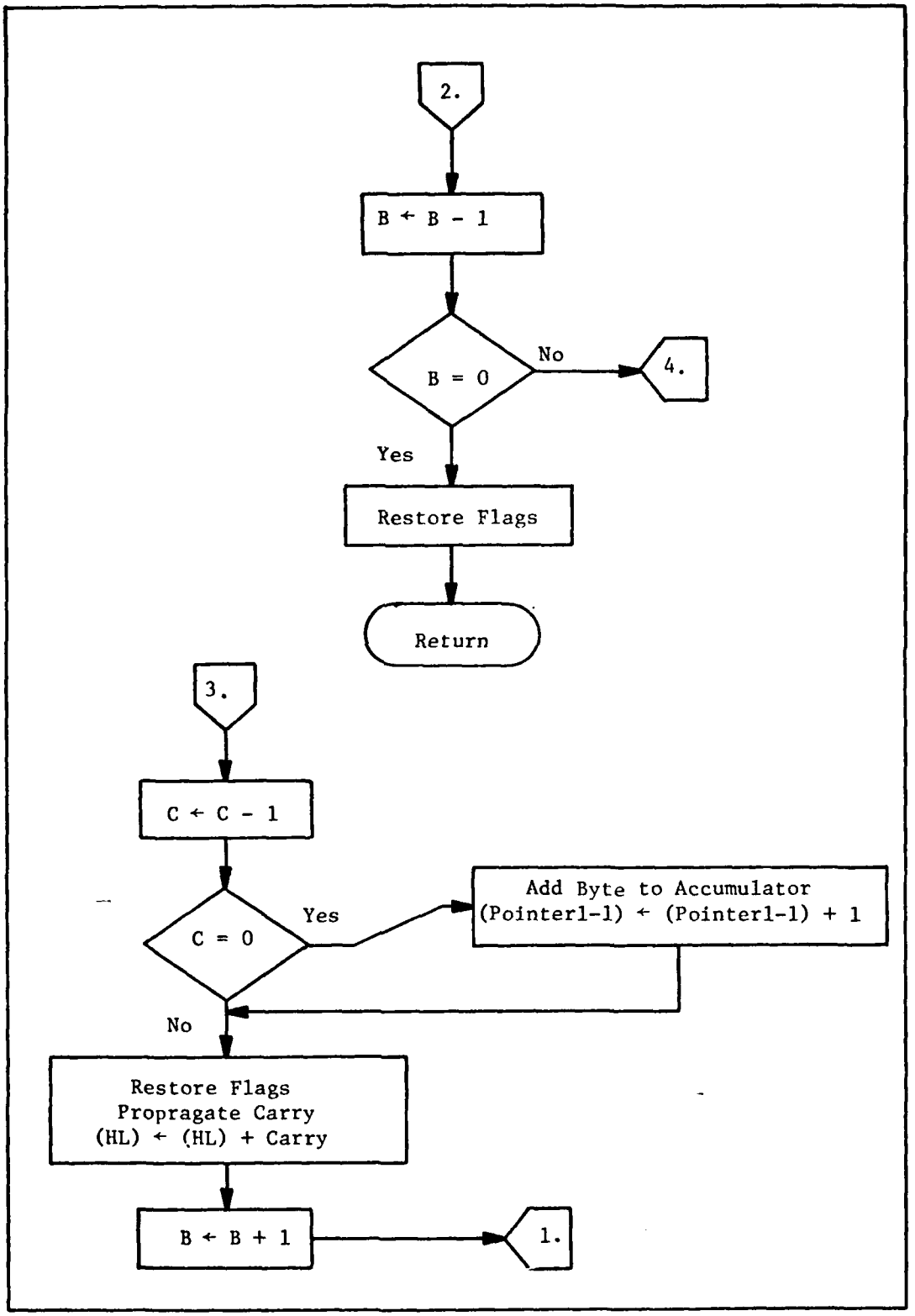


Fig. 38. Vector Summation (SUM) Flow Chart (Sheet 2 of 2)

Demonstration Program Source Listing

THE FOLLOWING ARE COMAND JUMPS USED BY THE MONITOR KEY COMMANDS.

```
EXT     CPWR,INITO
JP      CPWR
JP      INITO
```

This subroutine performs the right shift operation for a software multiply routine. It is designed to shift binary numbers which occupy a vector in memory. The following information about the vector must be passed to this routine;

a. A pointer positioned at the least significant word must be stored in the IX register,
b. The absolute number of words in the vector must be stored in the word (IX-1),
c. The number to be shifted must be stored in the vector.

This routine returns the vector with the number shifted one place the right. The flag register contains the carry information and when the Z flag is set to one (Z=1) the vector is empty.

```
RSHIFT: PUSH    IX      ;PUT POINTER IN HL
        POP     HL
        DEC     HL      ;MOVE TO SIZE
        LD      C,(HL)  ;GET SIZE
        LD      B,0
        ADD     HL,BC   ;MOVE POINTER TO MOST SIGNIFICANT WORD
        RR      B       ;SET CARRY TO ZERO
        RR      (HL)    ;SHIFT RIGHT
        PUSH    AF      ;SAVE FLAGS
SR1:    DEC     C
        JR      Z,SR2   ;ARE WE DONE YET?
        DEC     HL      ;MOVE TO NEXT WORD
        POP     AF      ;RESTORE FLAGS
        RR      (HL)    ;SHIFT RIGHT
        PUSH    AF      ;SAVE FLAGS
        JR      SR1     ;NEXT WORD
SR2:    POP     AF      ;RESTORE FLAGS
        RET
```

This subroutine performs the left shift operation for a software multiply routine. It is designed to shift binary numbers which occupy a vector in memory. The follwoing information about the vector must be passed to this routine;

a. A pointer positioned at the least significant word must be stored in the IY register,
b. The absolute number of words in the vector must be stored in the IY-1 word,
c. The number to be shifted must be stored in the vector.

This routine returns the vector with the number shifted one place to the left. The flag register contains flag conditions for the most significant word of the vector.

```
LSHIFT: PUSH    IY        ;PUT POINTER IN HL
        POP     HL
        LD      A,0
        RRA               ;SET CARRY TO ZERO
        LD      B,(IY-1)  ;PUT SIZE INTO COUNTER
SL2:    RL      (HL)      ;SHIFT LEFT
        PUSH    AF        ;SAVE FLAGS
        DEC     B
        JR      Z,SL1     ;ARE WE DONE?
        INC     HL        ;NEXT WORD
        POP     AF        ;RESTORE FLAGS
        JR      SL2
SL1:    POP     AF        ;CLEAR STACK
        RET
```

This subroutine adds two binary numbers of extended precision which are stored in vectors in memory. The following information about the vector must be passed to this routine;

a. The pointer for the accumulating vector must be in the IX register, and its absloute size in the IX-1 word,
b. The pointer for the new value vector must be in the IY register, and its absolute size in the IY-1 word,
c. The numbers must be stored in the vectors.

This routine returns the sum of the two numbers in the accumulating vector. Also, if the number grows too large, extra words are added as required to allow some extra room above the most significant word for unexpected growth. Care should be taken to estimate vector size requirements and initialization.

```
SUM:    PUSH    IX          ;PUT ACC. PINTER IN HL
        POP     HL
        LD      A,(HL)      ;FIRST WORD
        LD      B,(IY-1)    ;SIZE OF NEW VALUE
        LD      C,(IX-1)    ;SIZE OF ACCUMULATOR
        LD      E,0         ;USED WHEN PROPAGATING CARRY
        ADD     A,(IY+0)    ;SUM FIRST WORD
PS3:    LD      (HL),A      ;PUT PARTIAL SUM IN ACC.
        INC     HL
        INC     IY
        PUSH    AF          ;SAVE FLAGS
        JR      NC,PS1      ;DID WE GET A CARRY?
        DEC     B
        JR      Z,PS2       ;ANY MORE NUMBER TO ADD
PS6:    DEC     C
        POP     AF          ;RESTORE FLAGS
        LD      A,(HL)      ;GET NEXT WORD
        ADC     A,(IY+0)
        JR      PS3         ;NEXT WORD
PS2:    DEC     C
        JR      Z,PS4       ;OVERFLOW
PS7:    POP     AF          ;RESTORE FLAGS
        LD      A,(HL)
        ADC     A,E         ;PROPAGATE CARRY
        INC     B
        JR      PS3         ;NEXT WORD
PS1:    DEC     B
        JR      Z,PS5       ;ANY MORE NUMBER TO ADD
        JR      PS6
PS4:    LD      (HL),0      ;ADD WORD TO ACCUMULATOR
        INC     (IX-1)
        JR      PS7
PS5:    POP     AF          ;CLEAR STACK
        RET
```

This subroutine multiplys two binary numbers of extended precision which are stored in memory as vectors. This routine requires the following information;

a. The pointer for the multiplyer vector must be in the IX register, and its absolute size in the IX-1 word.
b. The pointer for the multiplication vector must be in the IY register, and its absolute size in the IY-1 word.
c. The pointer for the result accumulator must be stored memory location Result. This is a sixteen bit address.
d. The numbers must be stored in the vectors.

This routine returns the product of the two vectors in the vector pointed to by Result. The Z flag is set according to the contents of Result. Also, the contents of all vectors passed to this subroutine are lost execept for the work space assignd to Result.

```
MLTPY:  LD      (MPY1),IX       ;SAVE MULTIPLIER POINTER
        LD      (MPY2),IY       ;SAVE MULTIPLICAND POINTER
        LD      B,0
        LD      A,0
        LD      HL,(MPY1)       ;TEST MULTIPLIER TO SEE
        LD      C,(IX-1)        ;IF IT IS ZERO
MP2:    CPI
        JR      NZ,MP1
        RET     PO              ;RETURN ZERO RESULT
        JR      MP2
MP1:    LD      HL,(MPY2)       ;TEST MULTIPLICAND TO SEE
        LD      C,(IY-1)        ;IF IT IS ZERO
MP4:    CPI
        JR      NZ,MP3
        RET     PO              ;RETURN ZERO RESULT
        JR      MP4
MP3:    LD      C,(IY-1)        ;INITILIZE MULTIPLIER
        LD      A,(IX-1)
        ADD     A,C             ;COMPUTE SIZE OF RESULT
        LD      (IY-1),A        ;ADD SPACE TO MULTIPLICAND VECTOR
        LD      DE,(RESULT)     ;SET SIZE OF RESULT
        DEC     DE
        LD      (DE),A
        ADD     IY,BC           ;ZERO ADDED WORDS TO MULTIPLICAND
        LD      C,(IX-1)
MP5:    LD      (IY+0),B
        INC     IY
        DEC     C
        JR      NZ,MP5          ;END OF ZERO
```

```
        LD      HL,(RESULT)     ;ZERO RESULT VECTOR
MP6:    LD      (HL),B
        INC     HL
        DEC     A
        JR      NZ,MP6          ;END ZERO RESULT
MP9:    CALL    RSHIFT          ;SHIFT MULTIPLIER
        PUSH    AF              ;SAVE FLAGS
        JR      NC,MP7          ;DID WE GET A CARRY?
        LD      IX,(RESULT)     ;PUT POINTER IN IX
        LD      IY,(MPY2)       ;PUT POINTER IN IY
        CALL    SUM             ;ADD MULTIPLICAND TO RESULT
MP7:    POP     AF              ;RESTOR FLAGS
        JR      Z,MP8           ;ARE WE DONE YET?
        LD      IY,(MPY2)       ;PUT MULTIPLICAND POINTER IN IY
        CALL    LSHIFT          ;SHIFT TO LEFT
        LD      IX,(MPY1)       ;GET READY FOR NEXT CYCLE
        JR      MP9
MP8:    LD      A,0
        INC     A               ;SET Z=0
        RET
RESULT  EQU     0100H
MPY1    EQU     0108H
MPY2    EQU     0110H
WS1     EQU     0141H
        END
```

This is the control software for the electric vehicle battery switches and the real time clock interrupt. This is the frame work for all user modules whether they be for drive control or performance monitoring.

Additional Initilization to monitor including control software requirements

```
INITO:  LD      A,35H   ;CTC channel 0 - 100Hz Clock
        OUT     (00),A
        LD      A,4BH
        OUT     (00),A
;
        LD      A,0D5H  ;CTC channel 3 - counter for
        OUT     (03),A  ;real time clock
        LD      A,0DH
        OUT     (03),A
;
        LD      A,0C3H  ;Interrupt jump load
        LD      (38H),A
        LD      IY,CNTRL
        LD      (39H),IY
;
        LD      A,0FH   ;Set PIO 2, ch A for output mode
        OUT     (0AH),A
        LD      A,80H   ;Disable battery switches
        OUT     (08),A
;
        LD      HL,TBL-1        ;LOAD SWITCH TABLE INTO RAM
        LD      DE,SWTBL-1
        CALL    COPY
        LD      A,2
        LD      (CLOCK-1),A     ;SET UP CLOCK VECTOR
        LD      A,0
        LD      (SWTBL-1),A
        LD      (CLOCK),A
        LD      (CLOCK+1),A
        LD      HL,CLV5-1       ;SET UP POWER VECTOR
        LD      DE,POWER-1
        CALL    COPY
;
        IM      1       ;Interrupt Mode 1
        EI              ;Enable Interrupts
        JP      BEGIN   ;Jump to begin in monitor
```

```
Control program loop and interrupt service routine
Save registers

CNTRL:  EX      AF,AF'
        PUSH    BC
        PUSH    DE
        PUSH    HL
        LD      BC,(MPY1)
        LD      DE,(MPY2)
        LD      HL,(RESULT)
        EXX
        PUSH    IX
        PUSH    IY
;
;       CONTROL PROGRAM
;
        LD      E,03H   ;Accelerator channel addr.
        CALL    ADIO
;
;       FIND STEP
;
STEP:   LD      BC,03   ;Set point increment
        LD      HL,SWTBL
        LD      A,E     ;Put measurement in accumulator
        LD      D,0     ;Step counter
LOOP:   CP      (HL)    ;Compare to set point
        JR      NC,NEXT
        JR      SET
NEXT:   ADD     HL,BC   ;Get next setpoint
        INC     D       ;Increment counter
        JR      LOOP
;
;       OUTPUT SWITCH SETTING
;
SET:    LD      A,D     ;Move step into accumulator
        LD      IX,SWTBL
        CP      (IX-1)  ;Compare to last step
        JR      NZ,UPDN
        JR      BOUT    ;Setting equal
;
;       WICH WAY DID WE MOVE
;
UPDN:   PUSH    AF      ;Save flags
        CALL    BITS    ;Output measurement on monitor
        CALL    CRLF
        POP     AF
        JR      NC,DOWN
UP:     INC     HL      ;Move to settings
        LD      A,(HL)  ;Get first setting
        OUT     (08),A
        CALL    WAIT    ;Wait for switch inertia
        INC     HL
        LD      A,(HL)  ;Get second setting
        OUT     (08),A
        JR      BOUT
```

```
DOWN:   LD      BC,4    ;Increment to first setting
        ADD     HL,BC
        LD      A,(HL)  ;Get first setting
        OUT     (08),A
        CALL    WAIT    ;Wait for switch inertia
        DEC     HL
        DEC     HL
        LD      A,(HL)  ;Get second setting
        OUT     (08),A
BOUT:   LD      (IX-1),D ;Put step away
        JP      BTASK
;
;       WAIT LOOP FOR SWITCH INERTIA
;
WAIT:   LD      A,7CH   ;1 millisecond delay
W1:     DEC     A
        JR      NZ,W1
        RET
;
;       Time delay  T=116y + 1211/sys clock
;       Pick a delay and solve for y.  The value
;       is then converted to a hexadecmal number
;
;       A/D converter I/O subroutine
;
AD10:   LD      C,1CH   ;A/D control port addr.
        OUT     (C),E   ;Select channel & start
LOOP1:  IN      A,(09)  ;Test EOC Hi/Lo transit
        RRA
        JR      C,LOOP1
LOOP2:  IN      A,(09)  ;Test EOC Lo/Hi transit
        RRA
        JR      NC,LOOP2
        LD      C,1EH   ;Data port
        IN      E,(C)   ;Get measurement
        RET
;
;       This task takes voltage and current measurements
;       during each interrupt period and calculates the
;       incremental power.  The incremental power is summed
;       into a memory vector named Power.  The total number
;       of sample periods is also accumulated in a memory
;       vector named Clock.  Clock has a scale factor based
;       on the real time clock interupt.
;
;
BTASK:  LD      E,04H           ;MOTOR TERMINAL VOLTAGE A/D CH
        CALL    AD10            ;GET MEASUREMENT
        LD      A,E
        LD      (VOLTS),A       ;SAVE MEASUREMENT
        LD      A,1
        LD      (VOLTS-1),A
        LD      E,05H           ;MOTOR CURRENT A/D CH
```

```
                CALL    ADIO            ;GET MEASUREMENT
                LD      A,E
                LD      (AMPS),A        ;SAVE MEASUREMENT
                LD      A,1
                LD      (AMPS-1),A
                LD      IX,VOLTS        ;CALCULATE POWER INCREMENT
                LD      IY,AMPS
                LD      BC,WS1
                LD      (RESULT),BC     ;PUT POINTER FOR WORK SPACE IN
                                        ;RESULT
                CALL    MLTPY
                JR      Z,BT1           ;DONT ADD IF RESULT IS ZERO
                LD      IX,POWER        ;ADD POWER INCREMENT TO TOTAL
                LD      IY,WS1          ;POWER ACCUMULATOR
                CALL    SUM
BT1:            LD      HL,(CLOCK)      ;INCREMENT CLOCK COUNTER
                INC     HL
                LD      (CLOCK),HL
                JP      BKOUT
;
;       RETURN FROM INTERRUPT
;
BKOUT:          POP     IY
                POP     IX
                EXX
                LD      (RESULT),HL
                LD      (MPY2),DE
                LD      (MPY1),BC
                POP     HL
                POP     DE
                POP     BC
                EX      AF,AF'
                EI
                RETI
;
;       THIS ROUTINE COPIES A VECTOR INTO THE DESIGNATED LOCATION
;
                COPY:   LD      B,0
                        LD      C,(HL)
                        INC     C
                        LDIR
                        RET
;
;       This is a monitor commanded task which computes current
;       energy consumed and the elapsed time.
;       Time is given in seconds in hexadecmal format.  Energy
;       is given in hexadecmal plus an exponent.  The exponent
;       is the power of two (2**exp) of the MSB of the energy.
;       Energy here has units of joules or watt-seconds in this
;       calculation.
;
;
```

```
CPWR:   LD      A,(EP)          ;SET UP FACTOR AND EXP1
        LD      (EXP1),A
        LD      HL,CVFCT-1
        LD      DE,FACTOR-1
        CALL    COPY
        LD      HL,POWER-1      ;GET CURRENT VALUE OF POWER ACC.
        LD      DE,PCON-1       ;AND CLOCK ACC.
        CALL    COPY
        LD      HL,CLOCK-1
        LD      DE,ELTM-1
        CALL    COPY
        LD      IY,PCON         ;CALCULATE CURRENT ENERGY
        LD      IX,FACTOR
        LD      BC,CRPWR
        LD      (RESULT),BC
        CALL    MLTPY
TIME:   LD      A,3             ;CONVERT TIME IN TO SECONDS
T1:     LD      IX,ELTM
        CALL    RSHIFT
        DEC     A
        JR      NZ,T1
;
;       ROUND OFF ENERGY RESULT
;
        LD      A,(CRPWR-1)     ;COMPUTE NUMBER OF BITS IN CRPWR
        LD      B,3
RO1:    SLA     A
        DJNZ    RO1
        LD      E,A             ;SAVE NUMBER
        LD      C,0             ;COUNT LEADING ZEROS IN CRPWR
RO3:    LD      IY,CRPWR
        CALL    LSHIFT
        JR      C,RO2           ;DID WE GET A CARRY?
        INC     C               ;NO
        JR      RO3
RO2:    INC     C               ;CORRECT FOR SCIENTIFIC NOTATION
        LD      A,E
        SUB     C               ;EXPONENT OF NUMBER IN CRPWR
        LD      B,A
        LD      A,(EXP1)        ;GET EXPONENT OF FACTOR
        ADD     A,B             ;EXPONENT OF ENERGY
        LD      (EXP),A         ;SAVE EXPONENT
        LD      IX,CRPWR        ;BACK UP ONE BIT
        CALL    RSHIFT
        LD      HL,CRPWR        ;PUT BACK CARRY
        DEC     HL
        LD      B,0
        LD      C,(HL)
        ADD     HL,BC
        LD      A,80H
        ADD     A,(HL)
        LD      (HL),A          ;END PUT BACK
```

```
;
;       OUTPUT NUMBERS
;
        LD      B,HDSZ          ;HEADER SIZE
        LD      HL,HEDER
        CALL    TOM             ;TELL THE WORLD
        LD      B,2             ;OUTPUT SECONDS
        LD      HL,ELTM
        INC     HL
R04:    LD      A,(HL)
        CALL    LBYTE
        DEC     HL
        DJNZ    R04
        LD      B,5
R05:    CALL    BLK             ;SPACE OVER
        DJNZ    R05
        LD      B,0             ;MOVE TO MOST SIGNIFICANT
        LD      HL,CRPWR        ;END OF CRPWR
        DEC     HL
        LD      C,(HL)
        ADD     HL,BC
        LD      B,4             ;OUTPUT 4 MOST SIGNIFICANT WORDS
R06:    LD      A,(HL)
        CALL    LBYTE
        DEC     HL
        DJNZ    R06
        CALL    BLK
        LD      A,(EXP)         ;OUTPUT EXPONENT
        CALL    LBYTE
        CALL    CRLF
        RET
;
;       SWITCH AND SETPOINT TABLE
;
        DB      12H
TBL:    DB      1AH,0,0         ;.5 VOLTS
        DB      33H,20H,20H     ;1.0 VOLTS
        DB      66H,2AH,2AH     ;2.0 VOLTS
        DB      9AH,28H,2DH     ;3.0 VOLTS
        DB      0CDH,29H,3BH    ;4.0 VOLTS
        DB      0FFH,2FH,2FH    ;5.0 VOLTS
;
;       MONITOR CALLS
;
BITS    EQU     0F74CH
CRLF    EQU     0F517H
BEGIN   EQU     0F038H
BLK     EQU     0F494H
TOM     EQU     0F45EH
LBYTE   EQU     0F594H
```

```
;
;
;       WORK SPACE POINTERS -- WORK SPACE BEGINS AT 0100H
;       AND ENDS AT 01FFH
;
RESULT  EQU     0100H
MPY1    EQU     0108H
MPY2    EQU     0110H
CLOCK   EQU     0119H
AMPS    EQU     0121H
VOLTS   EQU     0129H
ELTM    EQU     0131H
EXP1    EQU     0138H
EXP     EQU     013BH
WS1     EQU     0141H
POWER   EQU     0151H
FACTOR  EQU     0161H
CRPWR   EQU     0171H
PCON    EQU     0191H
SWTBL   EQU     01B2H
CR      EQU     0DH
LF      EQU     0AH
;
;       HEADER
;
HEDER:  DB      CR,LF
        DB      'SECONDS  ENERGY   EXP'
        DB      CR,LF,LF
HDSZ    EQU     $-HEDER
;
;       CONVERSION FACTOR FOR ENERGY OUT PUT AND EXPONENT
;       USED TO INITIALIZE FACTOR  AND EXP1
;
EP:     DB      0ECH
        DB      02
CVFCT:  DB      005H,0EFH,00,00,00,00,00,00
        DB      05H
CLV5:   DB      00,00,00,00
        DB      00
        END
```

Appendix D

Test Data

This appendix contains the test data and calculations used to determine the conversion constants and time reference. The following data are contained in this section:

D.1 Timing Reference Measurement

D.2 Voltage Conversion Factor

D.3 Current Conversion Factor

D.4 Simulated Energy Measurement

D.1 Timing Reference Measurement

Sample No.	Sample Start	Counter Stop	Counts	Stop Watch[1] Seconds	Period Second/Counts
1	078FH[2]	30F5H	10598	1323.01	0.124834825
2	3A14H	891EH	20234	2525.82	0.124830483
3	1A02H	4156H	10068	1256.89	0.124840087
4	04B5H	21FFH	7498	935.92	0.124822619
5	009DH	21FFH	4879	609.09	0.124839106
6	1C78H	39BFH	7495	935.89	0.124868579

Average sample period ($\hat{T}$) = 0.124839854 seconds

Notes

1. Stop watch was a Hewlett Packard 55 calculator operating in its timer mode.

2. The 'H' is the Hexadecimal Number Base Identifier

D.2 Voltage Conversion Factor

A/D Converter Voltage Reference Measurement

$$\underline{5.109}\ \text{Volts};\ \frac{\text{A/D Ref.}}{256} = \underline{0.019957031}\ \frac{\text{Volts}}{\text{Count}}$$

No.	Battery Voltage[1] e_1	Counts A/D output[3]		e_0 Based Counts x $\frac{\text{volts}}{\text{count}}$	Ladder Ratio e_1/e_0
1	70.86	FGH[2]	246	4.909429688	14.43394838
2	67.96	EAH	234	4.669945313	14.55263294
3	66.14	E4H	228	4.550203125	14.53561482
4	64.54	EOH	224	4.47037500	14.43726757
5	61.38	D4H	212	4.290890625	14.50758373
6	59.80	CFH	207	4.131105469	14.47554425
7	53.49	BAH	186	3.712007813	14.40999122
8	47.15	A3H	163	3.252996094	14.49433035
9	44.00	98H	152	3.033468750	14.50484697
10	42.43	93H	147	2.933683594	14.46304574
11	37.69	82H	130	2.594414063	14.52736498
12	36.15	7DH	125	2.494628906	14.49113330
13	29.99	G7H	103	2.055574219	14.58989726
14	23.83	52H	82	1.636476563	14.56177286
15	20.66	47H	71	1.416949219	14.58062133
16	19.08	42H	66	1.317164063	14.48566700
17	17.49	3DH	61	1.217378906	14.36693320
18	14.31	32H	50	0.997851563	14.34081033
19	12.72	26H	44	0.878109375	14.48566700
20	6.35	16H	22	0.439054688	14.46289080
21	0	0	0	0	-

Average ladder ratio equals $\hat{X} = \underline{14.48544815}$

$$\text{Voltage Conversion Factor} = \frac{\hat{X} \times \text{Ref. Voltage}}{256} = \underline{0.289084346}\ \frac{\text{Volts}}{\text{Count}}$$

Notes

1. Battery voltage measured with HP3466A Digital Voltmeter.
2. The 'H' is the Hexdecimal Numbers Base Identifier.
3. A/D converter measurements were taken using the following terminal command sequencies.

Q01C, Ø4 Carriage Return (C/R)

QI1E C/R [counts]

D.3 Current Conversion Factor

A/D Converter Voltage Reference Measurement

$$\underline{5.109}\ \text{Volts};\ \frac{\text{A/D Ref.}}{256} = \underline{0.019957031}\ \frac{\text{Volts}}{\text{Count}}$$

No.	Shunt Voltage[1] e_i	Counts A/D output[3]		e_0 Based on counts x $\frac{\text{A/D Ref.}}{\text{Count}}$	Gain e_0/e_i
1	.1998	F7H[2]	247	4.929386719	24.67160520
2	.1763	DAH	218	4.350632813	24.67144080
3	.1504	BAH	186	3.712007813	24.68690301
4	.1249	9BH	155	3.093339844	24.76653198
5	.0983	7AH	122	2.43475813	24.76864551
6	.0752	5DH	93	1.856003906	24.68090301
7	.04913	3DH	61	1.217378906	24.77872799
8	.02521	1FH	31	0.618667969	24.54057790
9	.00893	ØBH	11	0.219527341	24.58312920
10	.00022	ØØ	Ø	0	-

Average Gain equals $\hat{X} = \underline{24.68316269}$

Shunt constant (K_s) = 2000 Amps/Volt

$$\text{Current Conversion Factor} = \frac{K_s \times \text{A/D Ref.}}{\hat{X} \times 256} = \underline{1.61705625}\ \text{Amps/count}$$

Notes:

1. Shunt voltage measured with HP3466A Digital Voltmeter.
2. 'H' is the Hexadecimal Numbers Base Identifier.
3. A/D converter measurements were taken using the following command sequence.

QOIC, Ø5 C/R

QI1E C/R [counts]

D.4 Simulated Energy Measurement and Calibration Test

Energy Conversion Factor

Energy Conversion Factor (E_{SF}) = V_{SF} x I_{SF} x Sample period

$$= \underline{0.05835478959} \ \frac{\text{Joules}}{\text{Count}^2}$$

Converted to Hex: E_{SF} = EFØ5H x 2^{ECH}

Test 1 Data

Simulated inputs

Input volts 23.83

Current .02664 Volt x 2000 $\frac{\text{Amp}}{\text{Volt}}$ = 53.28 Amps

A/D Ref. = 5.109 Volts

Start test ØØØØ counts, Stop 14ADH counts (D119, 11F)

Energy calculation

Seconds 294H = 661

Energy C808E16BH = $1.100\ 1000\ 0000\ 1000\ 1110\ \ 0001\ 0110\ 1001_B$

Exp 13H = 2^{19}

Energy = $2^{19} + 2^{18} + 2^{15} + 2^7 + 2^3 + 2^2 + 2$ = 819342 Joules

Side calculation

23.83 Volts x 53.28 Amps x 661 second = 839246.85 Joules

% Difference -2.3%

Test 2 Data

Simulated inputs

Input volts 23.81

Current .02650 Volt x 2000 $\frac{\text{Amp}}{\text{Volt}}$ = 53.00 Amps

A/D Ref. 5.109 Volts

Start test ØØØØ counts, Stop 2D1ØH counts (D119, 11F)

Energy Calculator

Seconds 5A2H = 1442

Energy D907D7D7H = $1.101\ 1001\ 0000\ 0111\ 1101\ 0\ 111\ 1101\ 0111_B$

Exp 14H = 2^{20}

Energy = $2^{20} + 2^{19} + 2^{17} + 2^{16} + 2^{13} + 2^6 + 2^6 + 2^5 + 2^4 + 2^3 + 2 =$ 1777914 Joules

Side Calculation

23.81 Volts x 53 Amps x 1442 Second = 1819703.06 Joules

% Difference -2.30%

Note

To improve the correlation it is necessary to increase the Energy Conversion Factor by at least 2.3%.

Vita

John Charles Frazier was born on 5 August 1950 in Tucson, Arizona. After graduation from high school in Long Beach, California in 1969, he entered Long Beach City College, majored in Industrial Technology and worked part time as a tool designer. He enlisted in the United States Air Force in 1971, and was trained as a ground radio technician. The following year he was selected for the Airmans Education and Commissioning Program and entered the University of Florida. He received his Bachelor of Science degree in Electrical Engineering in 1976 and, shortly thereafter, entered Officer Training School to receive his commission as a second lieutenant the same year. He was then assigned to the Space and Missile System Organization, now Space Division, at Los Angeles, California, where he worked as the Titan IIIC Guidance System Manager and later became the Titan 34D core vehicle integrator, a capacity he filled until entering the School of Engineering, Air Force Institute of Technology, in June 1980.

Permanent Address: 12249 E. 211 Street
Lakewood, CA 90715

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: AFIT/GE/AA/81D-1	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE *(and Subtitle)*: Electric Vehicle Power Controller	5. TYPE OF REPORT & PERIOD COVERED: MS Thesis
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): John C. Frazier, Captain USAF	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: AIR FORCE INSTITUTE OF TECHNOLOGY (AFIT-EN) WRIGHT-PATTERSON AFB, OHIO 45433	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS	12. REPORT DATE: DECEMBER 1981
	13. NUMBER OF PAGES: 119
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*	15. SECURITY CLASS. *(of this report)*: Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited

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

[illegible] LYNCH, Major, USAF
[illegible] Public Affairs

18. SUPPLEMENTARY NOTES

Approved for public release; IAW AFR 190-17

28 JAN 1982

Air Force Institute of Technology (ATC)
Wright-Patterson AFB, OH 45433

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

Microprocessor
Microcomputer
Electric Vehicle
Digital Controller
Interactive Control

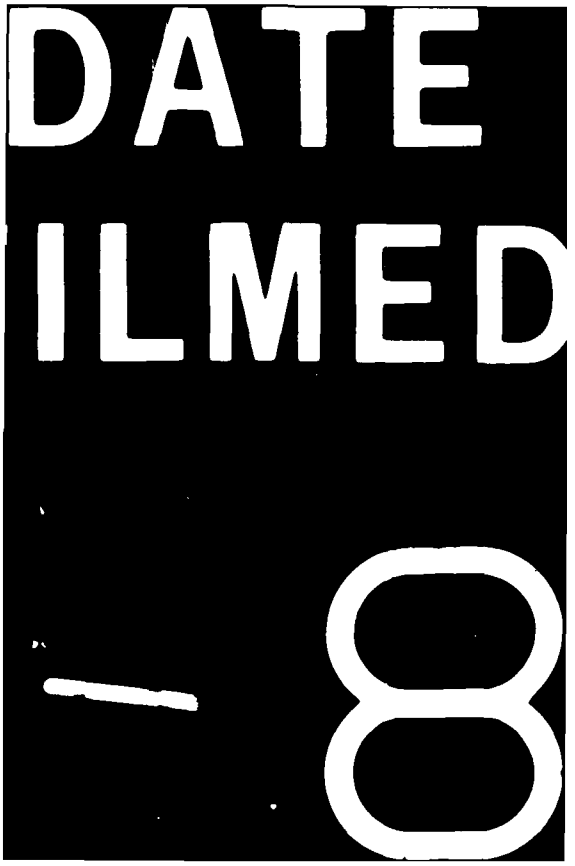
20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

The design of a general purpose microcomputer-based control system for an electric vehicle drive unit is described. This is a research and development system designed with an active operator interface for evaluating performance, dynamic control, parameter modification, and test program interaction. A microcomputer controller processes speed commands and monitors battery Consumption during the driving cycle. This design effort demonstrated that a unique one-of-a-kind microcomputer controller is easy to construct and interface with the vehicle's systems.

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

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*


DATE
ILMED
8