(NASA-CR-163494) A PROPULSION SYSTEM FOR THE MARS ROVER VEHICLE (Rensselaer Polytechnic Inst., Troy, N. Y.) 63 p HC A04/MF A01 CSCL 13I

N80-30750

Unclas
G3/37 28512

RENSSELAER POLYTECHNIC INSTITUTE
KNOWLEDGE AND THOROUGHNESS
1824

Rensselaer Polytechnic Institute
Troy, New York 12181

RPI TECHNICAL REPORT MP-70

A PROPULSION SYSTEM
FOR THE MARS ROVER VEHICLE

by

David C. Bogdan

A Study Supported by the

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

under

Grant NSG-7369

and by the

JET PROPULSION LABORATORY

under

Contract 954880

School of Engineering
Rensselaer Polytechnic Institute
Troy, New York

August 1980

CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENT

I would like to express my sincere appreciation to all the people involved in the Mars Project for their generous help in getting me on board and up to speed. Most notably, Dr. Dean Frederick, Project Advisor, for assistance and encouragement in my microprocessor software efforts; Dr. David Gisser, Project Electronics Advisor, for support in the hardware development; Jim Odenthal for many enjoyable sessions of constructive criticism, and Bob Miller for writing the wirelist and building the steering and propulsion board. I would also like to thank my friends Roy Musselman and Kent Dramstad for drawing the finished logic diagrams and figures.

ABSTRACT

This report covers the vehicle control electronics for the Mars rover vehicle. First, a functional description of the electronics and its place in the entire system is given, then the hardware involved is described from a user's point of view. Finally, changes and additions to the software are included.

PART 1

INTRODUCTION

The Mars Project at RPI has been an ongoing effort supported by NASA. At its inception in the late 1960's, the main emphasis was on conceptual studies of space travel to Mars. In the early 1970's, the field of study was narrowed to the development of a surface vehicle. The main emphasis was to design a highly maneuverable, light-weight, collapsible vehicle. The project, therefore, was mainly a mechanical engineering effort. With the successful completion of this phase of the project, efforts turned to developing an autonomous vehicle capable of obstacle detection and avoidance. Obstacle detection was accomplished with a laser triangulation scheme. The laser and detector scanned the terrain approximately three meters ahead of the vehicle. If a laser shot was detected, the terrain was assumed safe; otherwise it was assumed an obstacle (either an object or a crevice) placed the laser spot out of the detector's viewing angle. The laser data along with data about the vehicle position and speed were transmitted to a Varian mini-computer.. The Varian used the data to select a safe path for the rover and transmitted steering and speed commands back to the rover. The rover had an electronically-controlled independent four-wheel-drive system. The entire system was tested in 1976 and performed impressively.

Since then, with the basic concept proved, the Mars Project group has opted to expand and improve upon the rover's terrain detection and avoidance abilities. To accomplish this, all the electronic systems have been redesigned, a new laser-detector system has been implemented, and the controlling computer has been changed. The new laser-detector system has a variable laser angle and multiple detectors to give more detailed terrain information. The vehicle electronics were redesigned to handle the increase in data collected and to give more precise control of the vehicle. Finally, the Varian computer was dropped in favor of the Prime computer since more complex decision-making algorithms were needed with the increased information. The Prime computer supports Fortran and other high level languages, whereas the Varian had to be programmed in assembler. These changes to the rover involved a large group effort over several years. This report covers the new steering and propulsion system on the rover.

PART 2

SUBSYSTEM OVERVIEW

The Mars rover vehicle electronics can be divided into four subsystems (Figure 1), which are commonly referred to as the mast controller, the telemetry system, the analog MUX and the steering and propulsion system. The laser, detectors and mast electronics are mounted on the mast. As the mast rotates, the laser and detectors scan the terrain ahead of the vehicle. The mast electronics controls laser firing times and collects the information from the detectors. The information about laser shots and returns is sent to the telemetry system. The telemetry system is the outgoing data communication link with the Prime computer. It takes data from the mast and other systems, properly formats the data, and converts them to serial data for transmission to the Prime. The Prime computer is the decision maker for the rover. As such, it requires both the laser data and vehicle data, such as velocity, heading and steering angle. These data are collected and converted to a digital format by the analog MUX. The data are also sent through the telemetry system to the Prime which interprets the information via a path selection algorithm. The Prime then sends commands to the steering and propulsion system.

This report details the function (Part 1), hardware (Part 2), and software (Part 4) involved in the steering and propulsion system.

The system has three basic functions. As was already mentioned, the Prime sends commands to the steering and propulsion system. This is actually done over an FSK radio link. The receiver converts

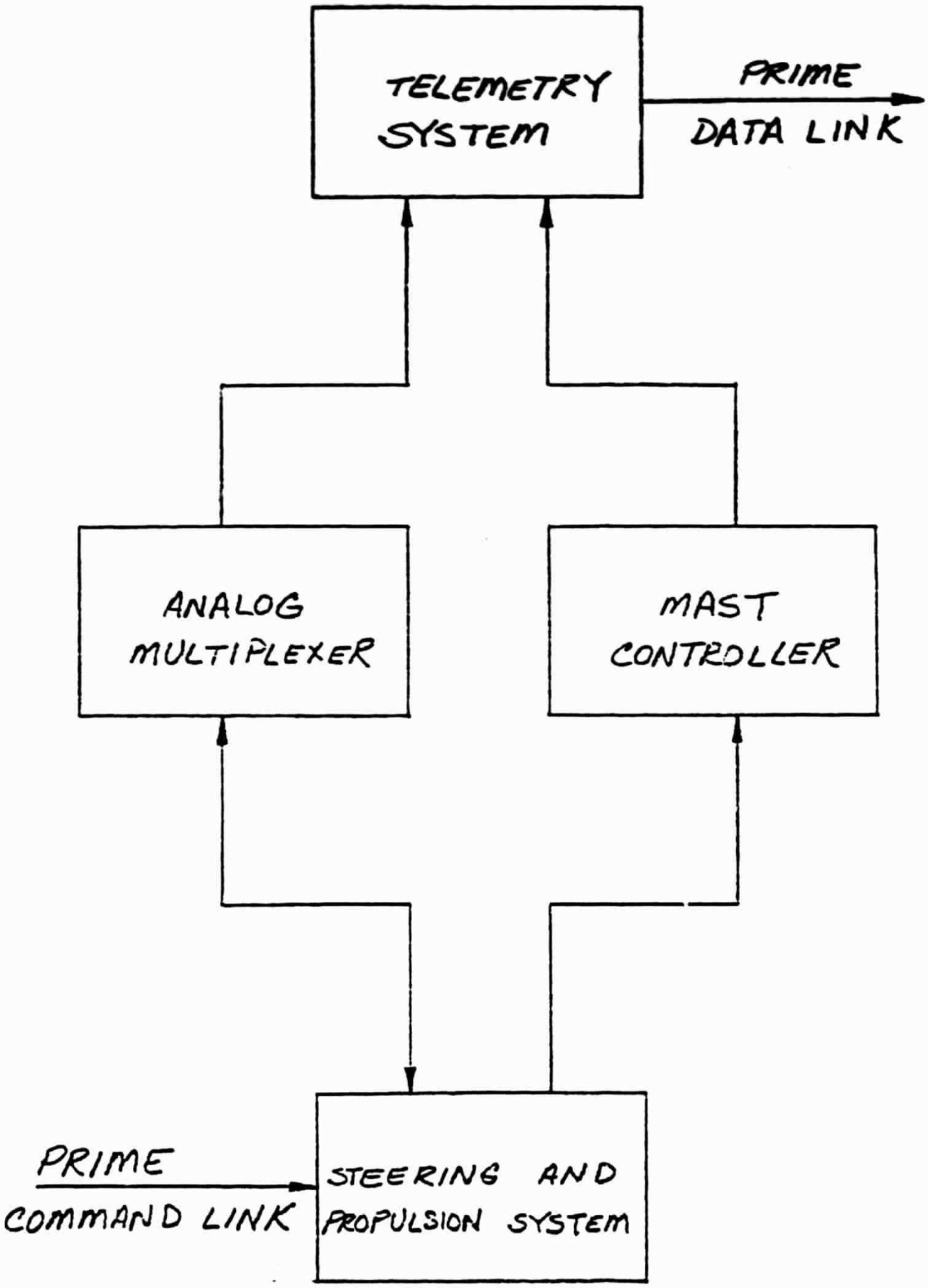


Figure 1. Rover Subsystems

the transmitted data into a TTL level serial stream for the steering and propulsion system. These commands control the vehicle. The vehicle, however, does not run "open loop" and therefore another function of the system is to obtain vehicle data. These data are actually obtained from the analog MUX and telemetry system. Knowing the present state of the vehicle and the desired state, the final function of the system is to effect the necessary vehicle controls.

The specific vehicle functions under control of the steering and propulsion system include the individual wheel speeds, and thus the steering angle, the stepping gyroscope, and finally, to a certain extent, the mast electronics. Basically, the system controls the mast in the sense that it merely passes mast commands from the Prime on to the mast electronics. The system does not interpret nor originate mast commands. These commands specify center of scan and other functions defined by the mast electronics. The stepping gyro, however, is truly controlled by the steering and propulsion system. The stepping gyro has a potentiometer output which is linear over 180° [1]. In addition, the gyro may be stepped around in two-degree increments providing a full 360 degree range. This is divided into ten 36° sectors. The steering and propulsion system must keep track of the gyro angle, and step the gyro to the next sector when a boundary is crossed. The current sector number must be saved and transmitted to the Prime computer. The Prime can only initialize the gyro, that is, it can issue a command to step the gyro until the pot angle is zero and then set the sector number to zero. This is normally done at the beginning of a vehicle run.

The main function of the steering and propulsion system is to control the wheel speeds and steering angle. The mechanical steering system is of the "wagon wheel" type and the steering angle is controlled by varying the individual wheel speeds. The mathematics involved in selecting the proper wheel speeds for a given angle are well documented in the Masters reports by both T. Geis [1] and J. Turner [2]. Each wheel is driven by a 24-volt D.C. motor. The power delivered to each wheel is varied by using a variable duty cycle pulsed voltage. Although the duty cycle is not linearly proportional to the wheel speed, feedback insures proper speeds. The duty cycle may be varied from 0 to 100% in 128 discrete steps. By comparison, the odd steering and propulsion system had 16 discrete steps.

The new steering and propulsion system uses a microprocessor based hardware-software package to perform the command communication, data gathering and vehicle control duties. The details of these functions are better explained in the hardware and software descriptions.

PART 3

STEERING AND PROPULSION SYSTEM HARDWARE

The steering and propulsion system is based on the M68SACI Stand Alone Computer Board [3]. In addition, a TTL board interfaces the microprocessor board to the rover and other electronic systems. In an attempt to keep the hardware description clear and concise, it will be described from a user's point of view. The user of a microprocessor system is generally interested in the memory organization, Input/Output definition and function, and interface signal definition. Since the M6800 microprocessor has what is commonly referred to as "memory mapped I/O," that is, memory and I/O are treated the same, the description will be organized by microprocessor address (labeled in hexadecimal format). Many of the addresses pertain to the microprocessor board itself and are briefly described here. More detailed information may be found in the User's Manual [3]. After the address description all interface signals will be defined except the standard microprocessor bus, which is also described in [3]. The logic diagrams are in Appendix A. The notation used in the diagrams along with the card build conventions used are described in Appendix B. This information will help in correlating the following I/O description to the actual hardware.

Address 0000 - 00FF, User's RAM.

This is the programmer's general workspace RAM. It is located on the microprocessor board (part number MC6810, locations U27 and U28).

Address 0100 - 7FFF, Undefined.

These addresses are unused and available for future expansion.

Address 8000 - 8003, Unavailable.

These addresses are not available to the user.

Address 8004 - 8007, PIA.

This is an unused Peripheral Interface Adapter. It contains two programmable 8 bit I/O ports. It is located on the microprocessor board (MC6820, U5).

Address 8008 - 8009, ACIA.

This Asynchronous Communication Adapter is a programmable serial I/O port used for communication with the teletype terminal and cassette recorder. The terminal can run up to 9600 baud, but 300 baud was chosen to be compatible with the cassette recorder. It is wired to the non-maskable interrupt line. It is also on the microprocessor board (MC6850, U4).

Address 8010 - 8011, ACIA.

This ACIA is the serial input port for the Prime computer-rover command link. The ACIA receives the 8 bit command word and detects parity errors, framing errors, overrun errors and loss of signal information. This comprehensive error checking provides a reliable command link. A baud rate of 300 is selected to be compatible with the FSK transmitter. Communication back to the Prime is only possible through the telemetry system, since the command link is one way. The ACIA issues a maskable interrupt when a command is received or a loss of signal is detected. The ACIA has a status register (8011) and a data register (8010). When interrupted, the microprocessor must read the status registers to check for errors. The command Format is software defined and will be covered in Part 4. The ACIA is located on the microprocessor board (M6850, U12). The baud rate and maskable interrupt must be jumpered in on the board.

Address 8012 - 801F, Unavailable.

These addresses are unavailable to the user.

Address 8020 - 8023, PIA.

Both halves of this PIA are used to transfer mast commands from the Prime to the mast electronics. The mast interface consists of a 12 bit command and two handshaking signals. The low four bits of port A are the most significant bits of the command. Port B contains the eight least significant bits and the handshaking controls. Control line CB2 is programmed to be a "Data Available" signal for the mast electronics. CB1 is programmed to be a "Data Accept" signal which is returned from the mast. Since these handshaking signals are activated by loading port B, port A must be loaded before port B to insure that the entire command word is available. This PIA does not interrupt the microprocessor, so the status register must be polled to determine if the mast accepted the data. The PIA is on the microprocessor board (MC6821, U13).

Address 8024 - 8FFF, Unavailable.

Address 9000 - 9FFF, Undefined.

These addresses are available for expansion.

Address A000 - A07F, ADS RAM.

This RAM is explained in Reference [3].

Address A080 - AFFF, Unavailable.

Address B000 - BFFF, Undefined.

These addresses are available for expansion.

Address C000 - C1FF, Display Memory.

This RAM is used by the Motorola software for a display station. In this application, it is not needed. The RAMs are located on a display interface board which is not included in the system. The addresses may be used for other applications.

Address C200 - C3FF, Unavailable.

Address C400 - C4FF, Shared RAM.

These RAMs are used for communication between the steering and propulsion system and the telemetry system. The microprocessor has complete control over the RAMs. It can permit the telemetry system to have read and/or write access. The microprocessor cannot access the RAM while the telemetry has either read or write access, but the microprocessor can regain control at any time. The RAM control register will be described later (Address C505). The telemetry system has a 16 bit data bus, while the microprocessor has an 8 bit architecture. To overcome this, two 8 x 128 bit RAMs are used such that the microprocessor addresses each alternately while the telemetry system addresses both in parallel. The microprocessors' even addresses correspond to the high bytes in the telemetry system, and odd addresses correspond to the low bytes. The shared RAM, and its associated control logic, are located on the steering and propulsion board.

Address C500 - C503, Motor Speed Registers.

Writing to address C500 - C503 loads the motor speed registers.

The microprocessor must keep track of the wheel speed data since the registers cannot be read. The registers are:

C500 - Left Front Wheel

C501 - Right Front Wheel

C502 - Right Rear Wheel

C503 - Left Rear Wheel

Each register may be considered a throttle for its corresponding wheel. The registers are loaded with the desired power in two's complement form. Using two's complement notation provides continuous speed variations around zero (Figure 2). Care must be taken around wheel speeds 7F, 80, 81, since these correspond to full speed forward, stop, and full speed reverse, respectively. Any numbering system would have one discontinuity, so the hardware is designed for the two's complement form in order to conveniently match the microprocessor numbering system. The motor speed controller produces two 450 Hz variable duty cycle signals for each wheel, one for forward and the other for reverse. The pulse width is proportional to the number loaded in the motor speed register. The pulse width is not proportional to the wheel speed. Figure 2 shows roughly how the wheel speeds vary with pulse width for the unloaded motor. Note that the wheels need a considerable amount of power just to start turning. The low 7 bits of the motor speed registers provide 128 discrete pulse widths. The sign bit directs the signal to the proper wheel. In addition, the signal is inverted for the reverse speeds in order to accomplish the two's complement function. Figure 3 shows the signal relationships for speeds 20, and E0, which

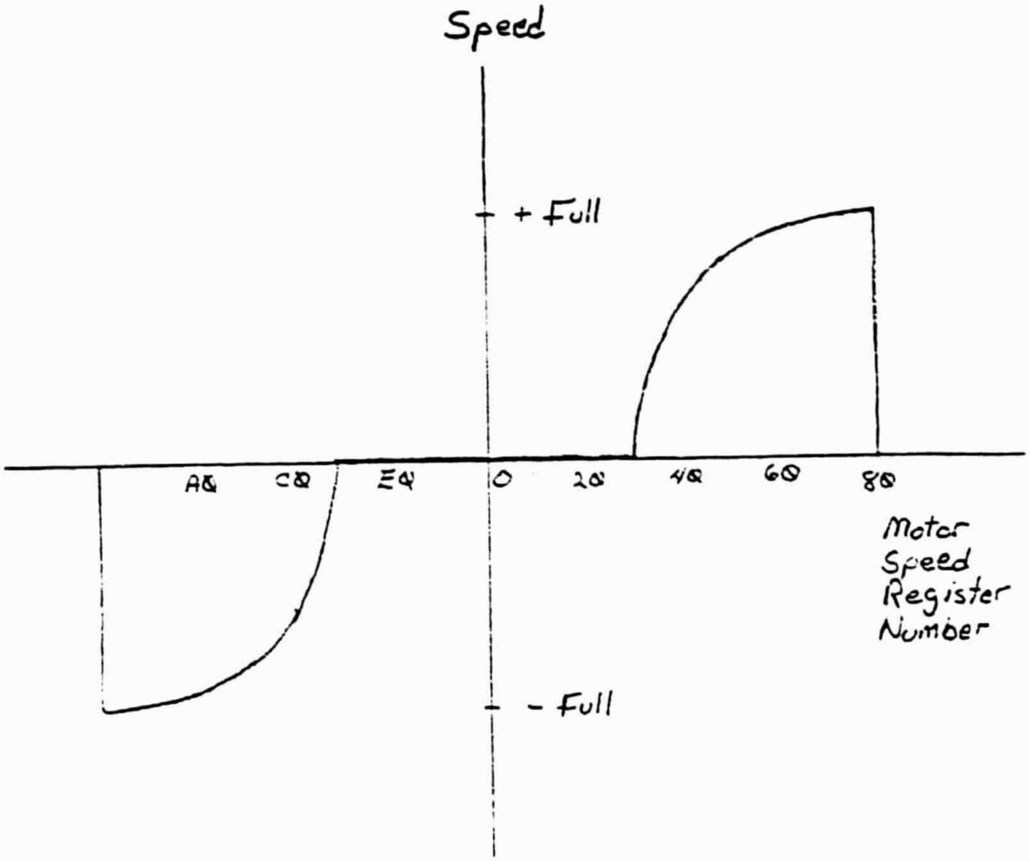


Figure 2. Wheel Speed Variations.

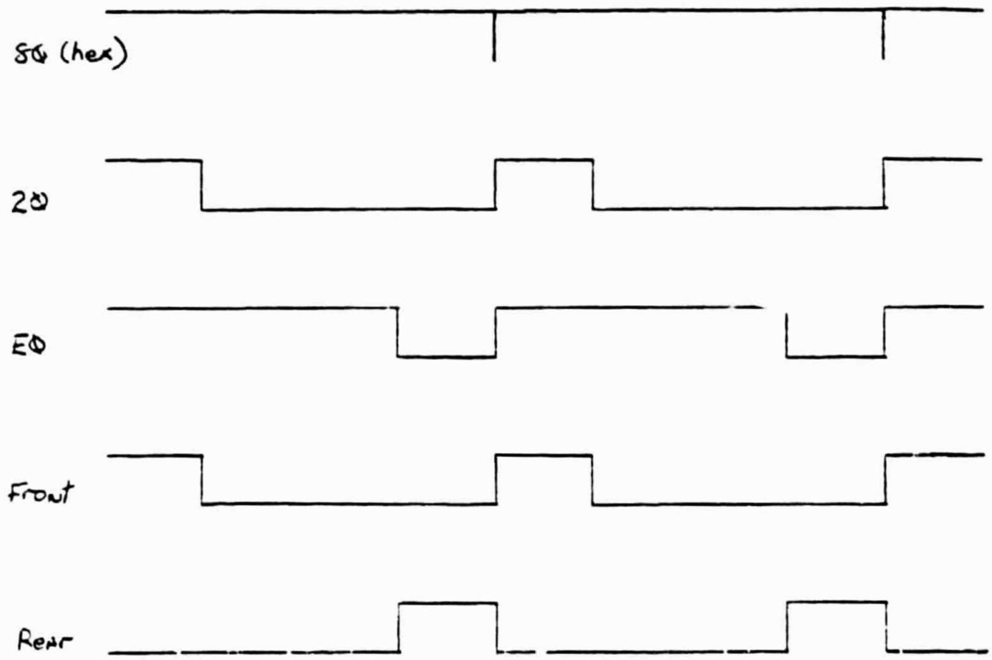


Figure 3. Pulse Width Modulator Signals

are 25% pulse widths forward and reverse, respectively. With an E0 in the motor speed register, the pulse width modulator (which only uses the low 7 bits) sees a 60 and produces a 75% pulse width. The reverse signals are inverted, however, giving a 25% pulse width signal as desired. These signals drive TTL high voltage open collector buffers which in turn directly drive the analog motor drivers (Figure 4). The motor speed registers are cleared by a master reset or by the micro when it writes to address C504. The motor speed registers and the pulse width modulator are located on the steering and propulsion board.

Address C504, Clear Motor Speed Registers.

Writing to this address clears all motor speed registers, thus stopping the vehicle.

Address C500 - C504, Read DIP Switches.

To conserve on the board space required for address decoding, these addresses have dual functions. Writing to C500 - C504 controls the motor drivers as was already mentioned above. Five 8 bit DIP switches are assigned C500 - C504 for reading. These switches are used to select control parameters defined in the software. The switches are assigned:

C500 -	DIP Switch 1
C501 -	DIP Switch 2
C502 -	DIP Switch 3
C503 -	DIP Switch 4
C504 -	DIP Switch 5

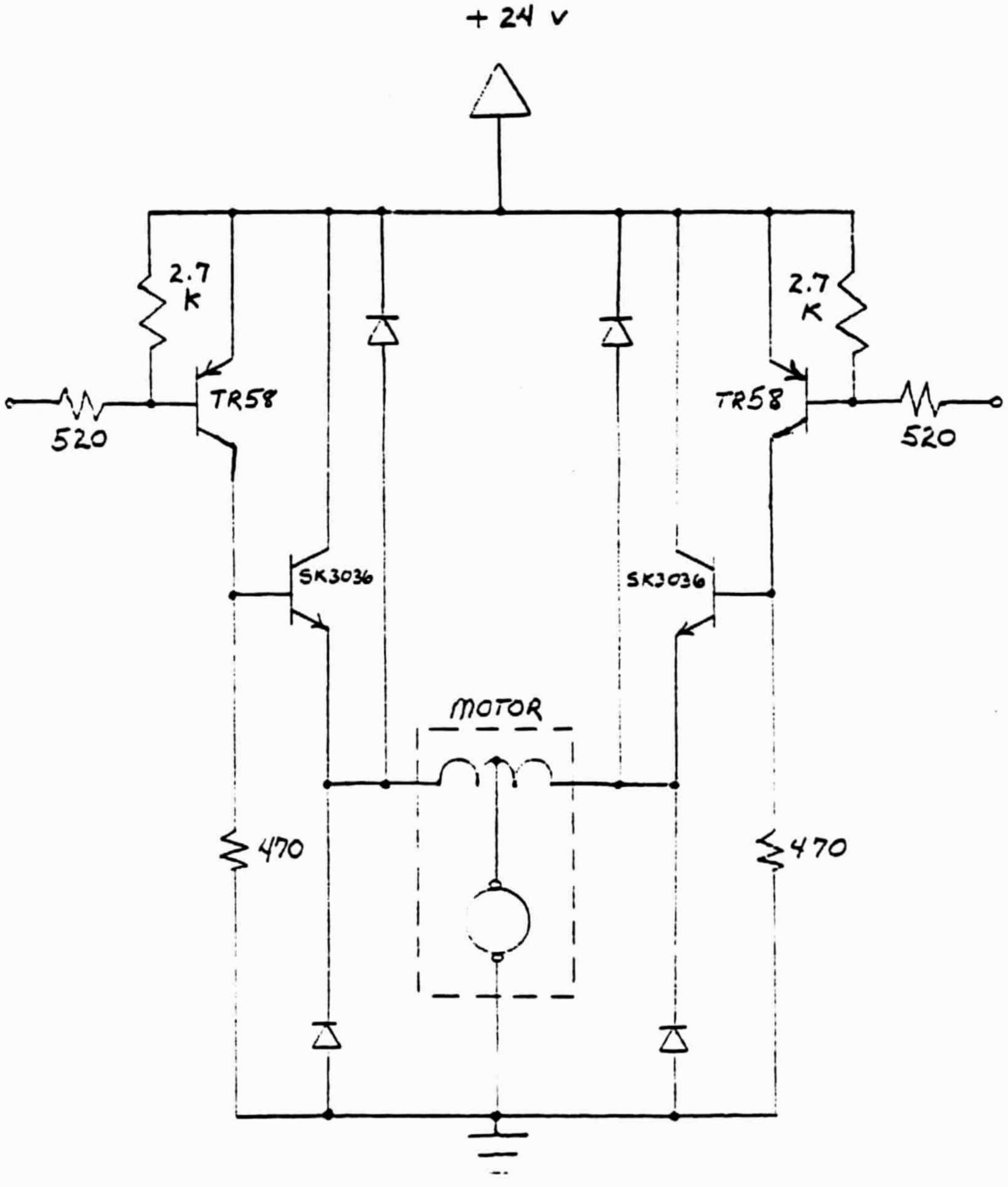


Figure 4. Analog Motor Drivers.

They are located on the steering and propulsion board.

Address C505, Shared RAM Control.

Each data bit in the address has a different function. They are:

Bit 7 - Enable telemetry read
6 - Enable telemetry write
5 - Read - valid telemetry address present
Write - Reset card
4 - Undefined, available for read
3 - Undefined
2 - "
1 - "
0 - "

The main function of this address is to control the shared RAM. Bits 7 and 6 independently control telemetry read and write access. A logic "1" enables telemetry access while a logic "0" disables access. These bits are latched and the microprocessor may read them back to see which system has RAM access. Both telemetry read and write must be disabled before the microprocessor can access the RAM. Bit 5 has a dual purpose depending upon whether a read or write is performed. If a write is performed with bit 5 equal to a logic "1" the card and microprocessor are reset. When reading, bit 5 monitors a telemetry signal, TVA, which indicates the telemetry system has valid RAM addresses on its address bus. It is not recommended that this read information be used since the telemetry system and the microprocessor are linked asychronously. The TVA signal may not be stable during a microprocessor read cycle. Bits 4 through 0 are undefined. If it is desirable to read any other signals, they may be easily wired into these bits since it is an 8 bit input port. All of the logic involved is located on the steering and propulsion board.

Address C506, Worm Gear Data.

The D.C. motors drive the rover wheels through worm gears. Due to mechanical play, these gears shift along their axes depending on whether the motors are driving or dragging. Microswitches are mounted on the vehicle to detect this motion and thus detect which wheels are driving or dragging. The microprocessor can read these microswitches through address C506 and use the information to adjust the power delivered to each wheel. This feature is more fully described in the Master's report by J. Turner [2].

Address C507, Gyro Control and Status.

This write-only address controls the gyro stepping motor and latches the gyro status. As was mentioned in the subsystem overview, the stepping gyro must be stepped around to keep its potentiometer output in the linear range. Address C507 data bits 7 and 6 are latched and used to control the stepping motor. Bit 7 determines the stepping direction, "1" implies clockwise and "0" implies counterclockwise. Bit 6 actually enables the stepping motor. The microprocessor both starts and stops the stepping pulse. The software must keep track of how long the pulse has been on and which sector the gyro is in. The sector number is saved in the low four bits of C507 for the telemetry system to read. An "F" should be stored in this status register while the gyro is stepping. The hardware for this function is on the steering and propulsion board.

Address C508 - D3FF, Undefined.

These addresses are available for future expansion.

Address D400 - D7FF, EPROM 1.

This is one of the user's 1k x 8 EPROM locations. This EPROM is located on the microprocessor board (MCM68708, U21).

Address D800 - DBFF, EPROM 2.

This is the second user EPROM (U22).

Address DC00 - DFFF, EPROM 3.

This is the third user EPROM (U23).

Address E000 - E3FF, MINIBUG II ROM.

This ROM contains Motorola's MINIBUG II monitor. It is located on the microprocessor board (SCM44506, U24).

Address E400 - EFFF, Undefined.

These addresses are available for future expansion.

Address F000 - FFFF, Unavailable.

This covers the entire memory and I/O function. The interface signals associated with the steering and propulsion system may be grouped according to connector location. Three connectors are involved, the backplane connector, a ribbon cable connector on the microprocessor board, and the ribbon cable connectors on the steering and propulsion board. Not all signals have been assigned connector pin locations due to unknown system requirements.

The backplane connector includes the standard microprocessor bus that is defined in Reference [3]. Since the T.V. interface, which is a part of the microprocessor bus is not required for this application, the signal wires TV0-4 are cut on the microprocessor board. This allows the pins to be used for other signals. The following backplane signals are for communication between the telemetry system and the steering and propulsion board:

Pin	Name	Description
U	TD0N	Telementary Data Bus Bit 0 (LSB)
V	TD1N	" 1
W	TD2N	" 2
X	TD3N	" 3
Y	TD4N	" 4
Z	TD5N	" 5
A	TD6N	" 6
B	TD7N	" 7
C	TD8N	" 8
D	TD9N	" 9
E	TD10N	" 10
F	TD11N	" 11
25	TD12N	" 12
26	TD13N	" 13
27	TD14N	" 14
28	TD15N	" 15 (MSB)
14	TR/$\overline{W}$	Telemetary read/write
15	VJA	Telemetry address valid
17	TDSBN	Telemetry data stroke
18	RAMAV	RAM available to telemetry
19	TA4	Telemetry address bit 4 (MSB
20	TA3	" 3
21	TA2	" 2
22	TA1	" 1
23	TA0	" 0

The handshaking signals, TR/$\overline{W}$, VTA, TDSBN and RAMAN require further explanation. Since the telemetry system has a 16 bit address bus and the shared RAM needs only five address bits, the telemetry system sends the VTA signal when the five address bits are sequencing through valid RAM addresses. The TR/$\overline{W}$ signal indicates a read when high, and a write when low. The TDSBN indicates the telemetry address and data lines are stable. The VTA and TDSBN signals are used as chip selects when the telemetry system has access to the RAM. The RAMAV signal is an output indicating to the telemetry system that it has read access to the RAM.

The ribbon cable connector, P3 on the microprocessor board, contains the mast command interface signals:

Pin	Description
25	Mast address bit 3 (MSB)
26	" 2
27	" 1
28	" 0
33	Mast data bit 7
34	" 6
35	" 5
36	" 4
37	" 3
38	" 2
39	" 1
40	" 0 (LSB)
41	Data accept
42	Data available

This connector is wired directly to the mast PIA. The mast address bits correspond to the low four bits of port A while the data bits correspond to port B. The data available signal goes active low when port B is loaded. It is cleared when an active low data accept signal is returned from the mast.

The ribbon cable connectors on the steering and propulsion board do not have pins assigned. They will be defined when the total system requirements are determined. The rover vehicle interface signals can be organized into several groups.

Motor Drive Signals

LFRVN	-	Left Front Reverse
LFFWN	-	Left Front Forward
RFRVN	-	Right Front Reverse
RFFWN	-	Right Front Forward
RRRVN	-	Right Rear Reverse
RRFWN	-	Right Rear Forward
LRRVN	-	Left Rear Reverse
LRFWN	-	Left Rear Forward

These are active low signals and are driven by high voltage open collector drivers. They may be connected directly to the analog motor drivers (Figure 4) which are mounted near the rover motors.

Gyro Controls

STEPCCW	-	Step Gyro Counterclockwise
STEPCW	-	Step Gyro Clockwise

These are positive TTL level signals.

Worm Gear Microswitch Data

LRRN	-	Left Rear Reverse
LRFN	-	Left Rear Forward
RRRN	-	Right Rear Reverse
RRFN	-	Right Rear Forward
RFRN	-	Right Front Reverse
RFFN	-	Right Front Forward
LFRN	-	Left Front Reverse
LFFN	-	Left Front Forward

These are the worm gear drive/drag microswitch indicators. They are active low signals and must be pulled up.

Gyro Sector Indicators

SEC0	Gyro sector number bit	0 (LSB)
SEC1	"	1
SEC2	"	2
SEC3	"	3

These signals indicate which sector the stepping gyro is in. They are for the telemetry system to read and transmit to the Prime computer.

Motor Driver Clock

MDCK - Motor driver clock

MDCKR - Motor driver clock return

This signal is from the baud rate generator on the microprocessor board. The 115.2 kHz frequency will give approximately a 450 Hz fundamental frequency for the pulse width modulator (since it is divided by 256). The clock should be on a twisted pair cable with the return grounded on both boards.

PART 4

SYSTEM SOFTWARE

The microprocessor software is organized into several functional subroutines. Most of these subroutines were developed by J. Turner and are well documented in Reference [2]. Emphasis here will be placed on changes to these subroutines and on the new subroutines. The alterations and additions were required to support an expanded command Format and the redesigned hardware interface. Before covering the changes made, a brief description of each subroutine is given.

1. COMMON - This is not actually a subroutine. The common section contains the symbol definitions which are common to all subroutines. It contains no executable code.
2. MAINLP - This is the main loop which calls the subroutines.
3. CMDDEC - This is the command decode subroutine. It interprets commands sent by the Prime computer, and performs the necessary actions.
4. GETDAT - This routine gathers the vehicle data needed for vehicle control. It accesses the shared RAM to read the data collected by the Analog MUX. This includes wheel speeds, steering angle and gyro angles. It also echos previous commands and command status back to the Prime through the shared RAM. The worm

gear microswitch and DIP switches are read in this routine.

5. STRCOR - This routine calculates the wheel speeds necessary for the present steering angle. It then compares the present steering angle to the desired steering angle and makes corrections.
6. TERCOR - This routine evaluates the worm gear microswitch data and corrects the wheel speeds accordingly. This routine may be skipped by issuing an override terrain compensation command.
7. FLTR - This routine is a discrete low pass filter. It is used to filter the wheel speed controls.
8. CONTRL - This is a proportional controller for the wheel speeds. It also performs the actual motor drive I/O.
9. DISPLY - This routine is mainly for debug. It is used in conjunction with a terminal to display information about the vehicle and microprocessor.
10. GYRO - This routine controls the stepping gyro. It keeps track of the gyro sector.
11. MULT8 - This routine performs an 8 bit multiply and saves the most significant byte of the product.
12. CMD - This is the interrupt service routine. It is entered upon a maskable interrupt which is only issued by the command ACIA.

13. INIT - This is the initialization routine. It initializes the PIA's,ACIAs, the stepping gyro, and zeros the RAM. It also copies the main loop into RAM and uses the copy for execution. In this way the programmer may change the main loop for debugging purposes.

Of these subroutines, all but two are fully described in Reference [2]. Both the CONTROL and GETDAT routines have been modified due to hardware changes. These are the only routines that actively perform I/O. The CONTRL routine had the motor speed register addresses changed. The speeds are also directly output in two's complement form. Before, the speed had to be converted to a signed binary format. The GETDAT routine required address changes also. In addition, the method of RAM access was altered to match the new hardware. Otherwise, these routines perform the same functions as before.

Two routines dealing with the command link have been replaced. The interrupt handler, CMD, replaced the old interrupt routine, NEWCMD, and the command decoder, CMDDEC, replaced GETCMD. Both of these changes were necessary to provide multiple byte command capability. In this way, two byte steering commands give 8 bit steering resolution (approximately 1.4°) as opposed to the previous command with 4 bit resolution (approximately 129°). This gives the path selection algorithm on the Prime computer greater leeway in choosing a safe path. The multiple byte command capability also permits the use of the 12 bit mast command.

The new command format is partially compatible with the old command format. This was done so that the old command control box would still work. The UART in the box must be wired to send 8 bit commands, with the eighth bit equal to zero. With this modification, the box will be able to issue the main drive command, single byte steering command, and the front wheel drive command. The command format is presently defined as:

0X	-	one wheel drive
1X	-	disable terrain compensation
2X	-	vehicle reset/gyro initialize/display control
3X	-	undefined
4X	-	single byte steering*
5X	-	main drive*
6X	-	undefined
7X	-	front wheel drive*
8X	-	two byte steering
9X	-	two byte mast
AX	-	undefined
BX	-	"
CX	-	"
DX	-	"
EX	-	"
FX	-	"

*compatible with previous format

The new command decode routine decodes by a table look-up method. Since the general function of a command is indicated by the first four bits, these bits are used as a relative address for a look-up table (see Figure 5). Each entry in the table is the starting address of an action routine. The relative address is added to the base address of the table and used to look up the required action routine address. This method provides a very flexible decoder. The

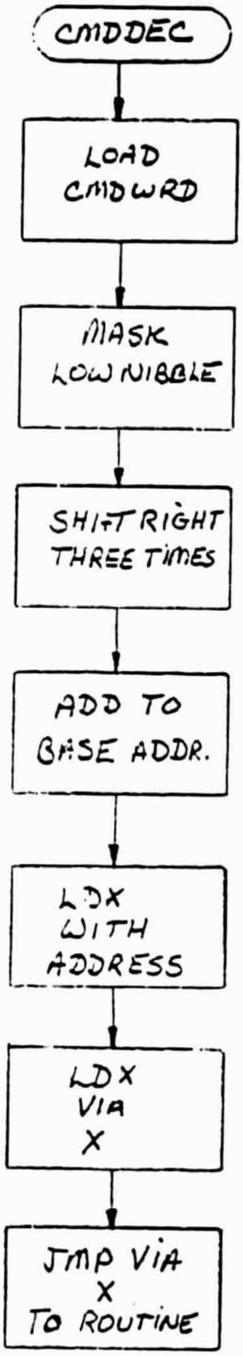


Figure 5. CMDDEC Chart.

command format may be easily changed by merely rearranging the table. New commands may be added by writing a new action routine and placing its starting address in a previously undefined command location in the table. When writing new multiple byte command action routines, two flags must be checked. Bit 6 of the TSKBTE word is a flag set by the interrupt handler to tell it the next word is data, rather than a new command. The action routine must clear this bit when all the data are received. The action routine has a flag (MVTIBT) that tells it when data is available. This is needed because the first time the routine is entered, no data will be available. At this time the MUTIB flag must be set. Then the next time the routine is entered, data will be available. This flag should also be cleared when all the data is received.

The interrupt handler is entered when a maskable interrupt is issued by the command ACIA (see Figure 6). The ACIA status and data are then read and saved to be echoed back to the Prime computer through the telemetry system. The status is checked for a loss of signal flag. If the signal is lost, all interrupts are masked to insure the vehicle has time to respond. Otherwise, if the signal was erratic, the ACIA may interrupt the microprocessor often enough to prevent the vehicle from stopping. After the mask is set, the interrupt handler fakes a stop command to prevent the rover from moving uncontrolled. Once the stop command is processed, the interrupt mask is cleared. Normal commands are received at such a slow rate that they will always have time to be processed. If any errors have

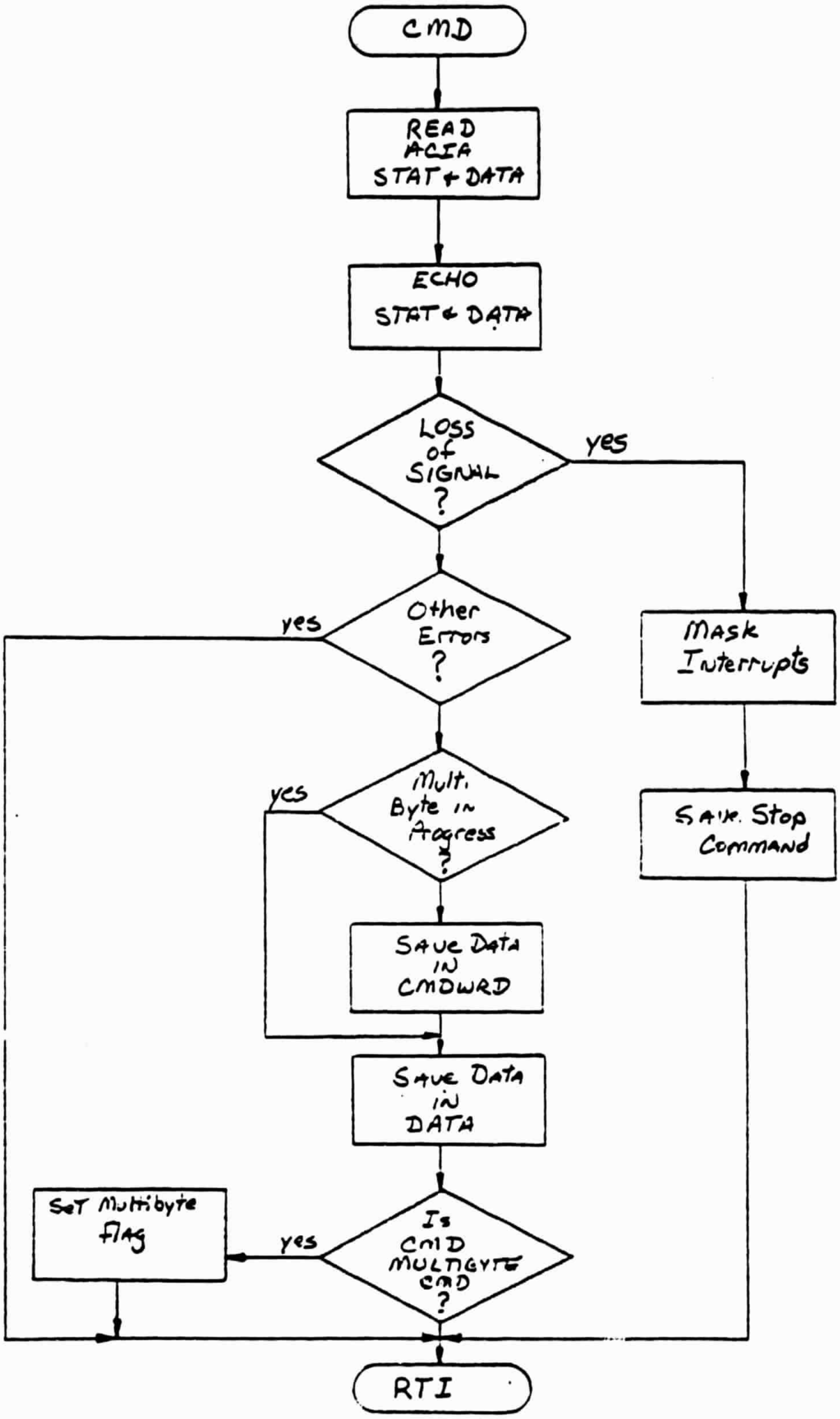


Figure 6. CMD Routine Flow Chart.

occurred, such as a parity error, framing error or overrun error, the command is ignored and is echoed back to the Prime. If no errors occurred, the MUTIBT flag is checked to determine whether the data is a command word or data. If it is a new command it is saved in CMDWRD; otherwise it is saved in DATA. Also, if it is a new command, bit 7 is tested to determine if it is a multiple byte command. If it is, then the MVTIBT flag is set so that the next interrupt will be interpreted as data. As was mentioned, the MUTIBT flag must be cleared by the action routine when all the data are received. In this way the number of data bytes may be defined in the action routine rather than having a fixed amount defined by the interrupt handler.

PART 5

DISCUSSION AND CONCLUSIONS

The steering and propulsion system board was built and tested during the spring of 1980. Although it was not tested in a vehicle environment, all I/O functions were exercised using the microprocessor monitor, MINIBUG II, and small test programs. The shared RAM telemetry interface has not been thoroughly tested since the telemetry board was not complete. All other functions performed correctly.

The motor speed pulse width modulator was tested both with a frequency generator and with the actual baud rate generator as a clock input. The modulator was capable of driving an unloaded motor at 24 volts with a final output frequency of anywhere between 4Hz to 40 kHz without noticeable effects. Below 4 Hz the motors tend to jerk. Although other reports (two) have noted motor overheating at high frequencies, this was not observed. The frequency of 450 Hz was chosen to match as closely as possible the old system's frequency of 400 Dz. If further experimentation is desired, any of the baud rate outputs may be used as an input clock. This clock is divided by 256 to give the final output frequency.

The motor driver clock and other interface signals have not been assigned connector pins. Once system requirements are defined, these signals may be assigned pins on the top card ribbon cable connectors. The microprocessor bus and the telemetry bus have been assigned pins on the backplane connector.

The microprocessor software has been tested on the Motorola Emulator which is installed on the IBM 3033. The emulator is available for use in signon AU70. The assembler used was developed by the University of Michigan and is available in signon UNSP. The software has been assembled - error free - and debugged on the emulator. Actual hardware tests are scheduled for the summer of 1980.

PART 6

LITERATURE CITED

1. Geis, T., "Control Electronics for the Mars Roving Vehicle," Master's Project Report, Rensselaer Polytechnic Institute, Troy, New York.

2. Turner, J.M., "A Propulsion and Steering Control System for the Mars Rover," Master's Project Report, Rensselaer Polytechnic Institute, Troy, New York, August 1979.

3. "M68ADS2A Development System User's Guide," Motorola Semiconductor Products, Inc.

APPENDIX A

LOGIC DIAGRAM NOTATION

The logic diagram is spread over several pages. Each page has functionally related logic. A description of the function is given at the bottom of each page. The pages are numbered MS010, MS020, ... Each page has input signals on the right, and output signals on the left. Bidirectional signals may be on either side. Each signal has a name which is related to its function (such as DOP stands for Data bit 0 powered). In addition any name that ends with an "N" is an active low signal.

All logic blocks are drawn in rectangular form (Figure 7). The part number is at the bottom of the block. The "74" on 74LS series parts was left off. The board location is entered above the block. Pin numbers are listed outside the block. Only signals going from one page to another are assigned names. The input signals have the signal name, and the page number of the signal source. Outputs have a signal name, and a list of all the pages the signal goes to. Also note that pages MS130 - MS150 have all the signals coming to or leaving the board originate or terminate on connectors.

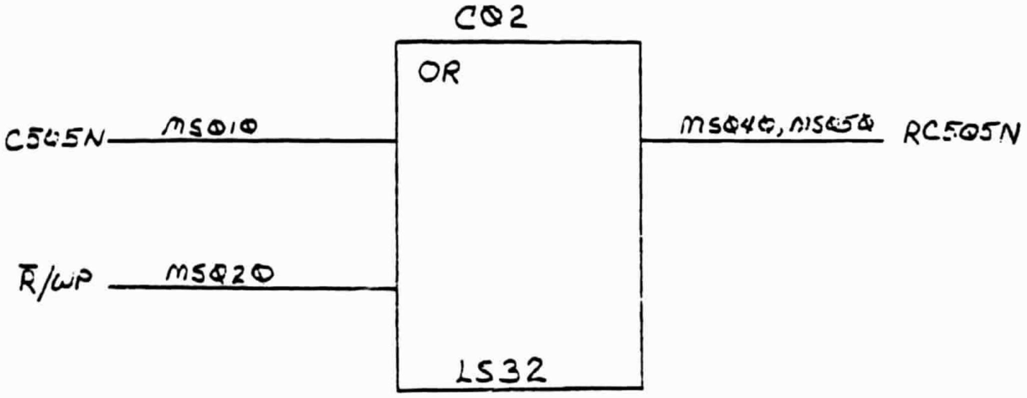


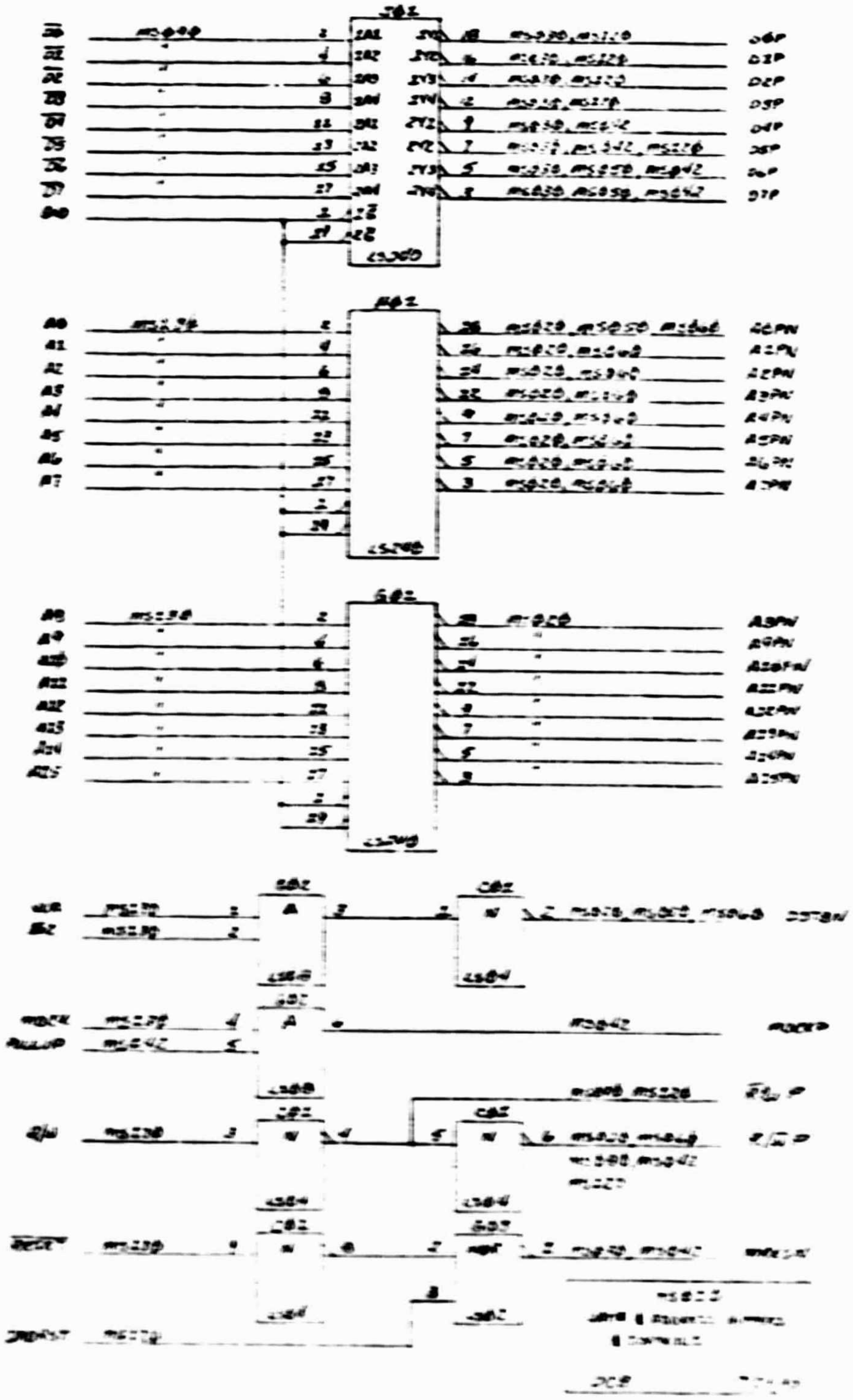
Figure 7. Example Logic Block.

APPENDIX B

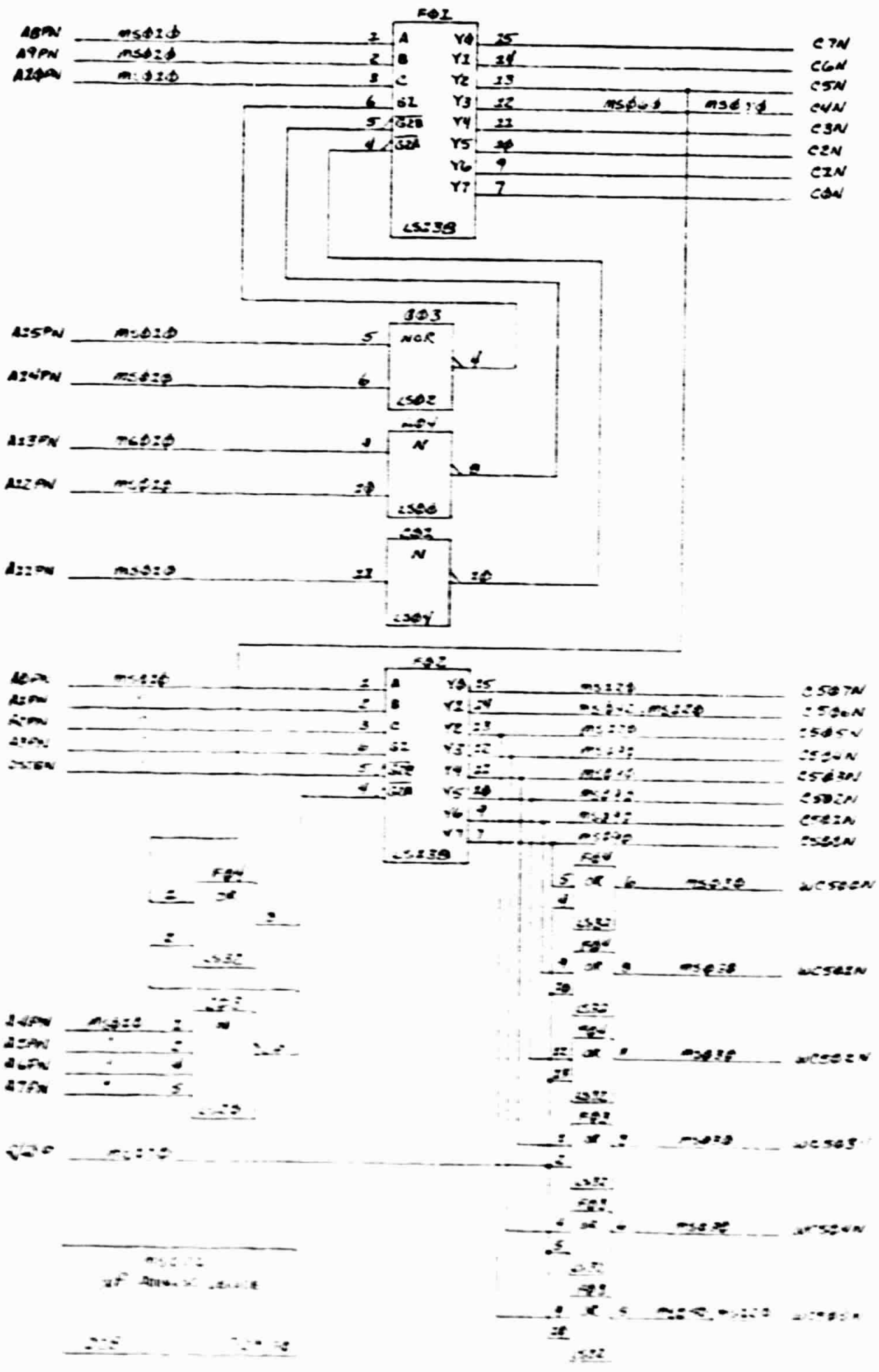
LOGIC DIAGRAMS

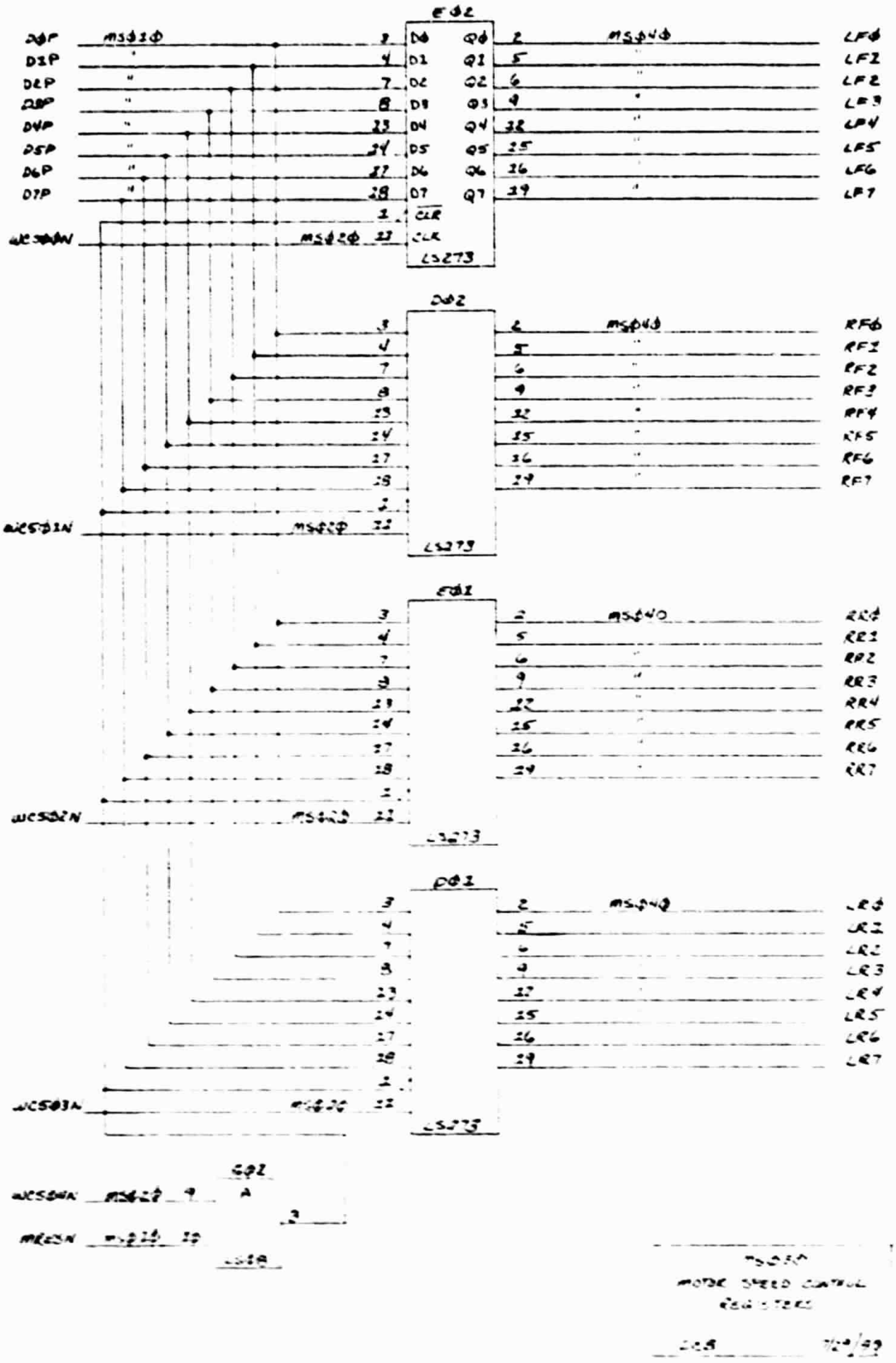
The logic diagrams for the steering and propulsion board follow. The pages included are:

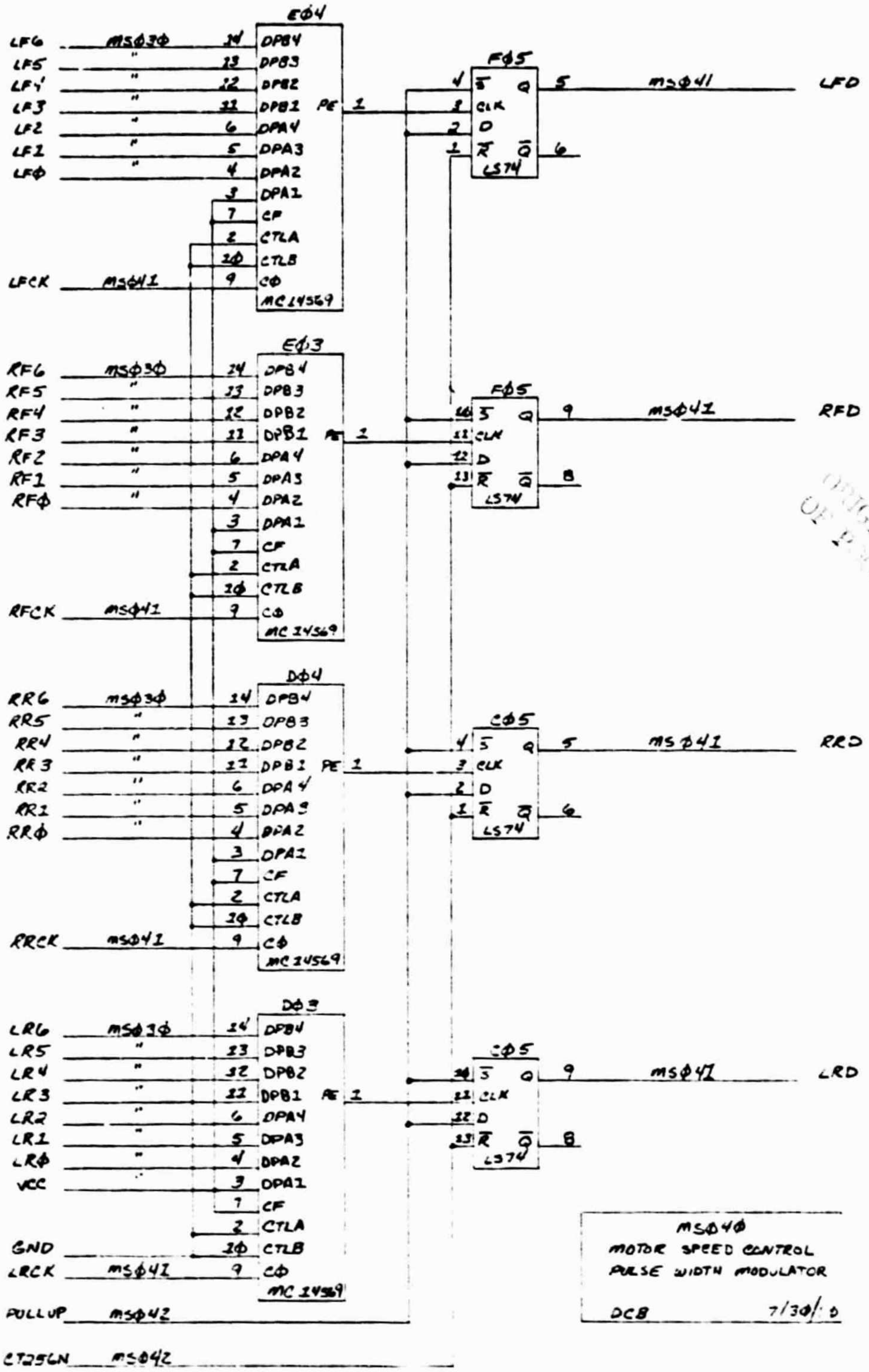
MS010	Microprocessor bus buffers
MS020	Microprocessor address decode
MS030	Motor speed registers
MS040	Pulse width modulator
MS041	Pulse width modulator
MS042	Pulse width modulator and gyro step control
MS050	RAM controls
MS060	RAM address and chip select MUX
MS070	RAMs
MS080	RAM data bus MUX
MS090	DIP switch gates
MS100	DIP switch pullups
MS110	DIP switch pullups
MS120	Microswitch gates and odds and ends
MS130	Backplane connector
MS140	Telemetry connector
MS150	Motor speed connector



D0
D1
D2
D3
D4
D5
D6
D7
LS240
A0
A1
A2
A3
A4
A5
A6
A7
LS240
A8
A9
A10
A11
A12
A13
A14
A15
LS240
LS08
LS04

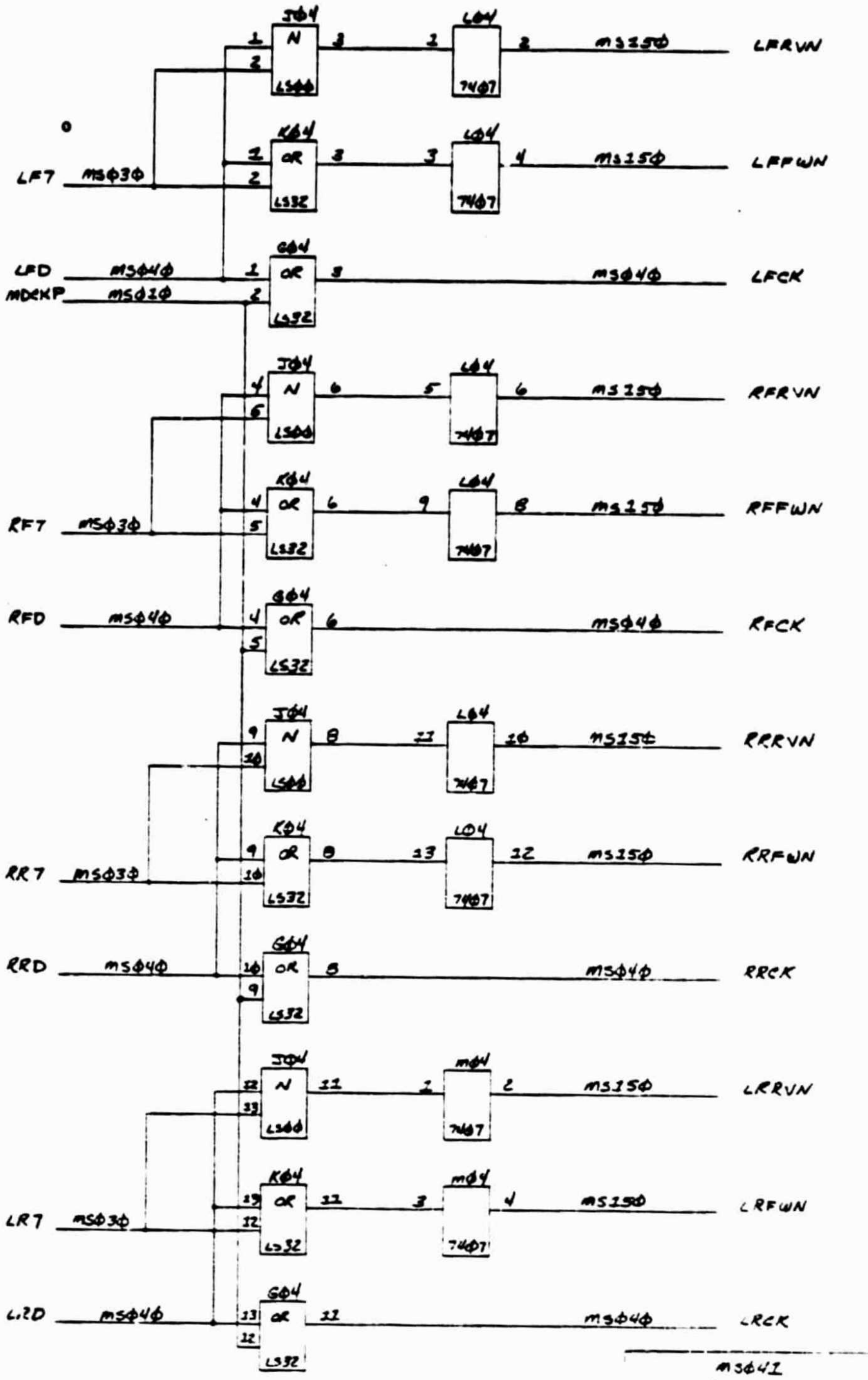

F01
A8PN
A9PN
A10PN
A
B
C
G1
G2B
G2A
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
LS138
C7N
C6N
C5N
C4N
C3N
C2N
C1N
C0N
NOR
LS02
N
LS06
LS04
A15PN
A14PN
A13PN
A12PN
A11PN
F02
LS138
OR
LS32


E02
D0P
D1P
D2P
D3P
D4P
D5P
D6P
D7P
MS010
D0
D1
D2
D3
D4
D5
D6
D7
CLR
CLK
Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
MS040
LF0
LF1
LF2
LF3
LF4
LF5
LF6
LF7
WC500N
MS020
LS273
D02
RF0
RF1
RF2
RF3
RF4
RF5
RF6
RF7
WC501N
LS273
E01
RR0
RR1
RR2
RR3
RR4
RR5
RR6
RR7
WC502N
LS273
D01
LR0
LR1
LR2
LR3
LR4
LR5
LR6
LR7
WC503N
LS273
G02
WC504N
MS020
A
MRESN
LS08
MS030
MOTOR SPEED CONTROL
REGISTERS
LAB
7/24/89



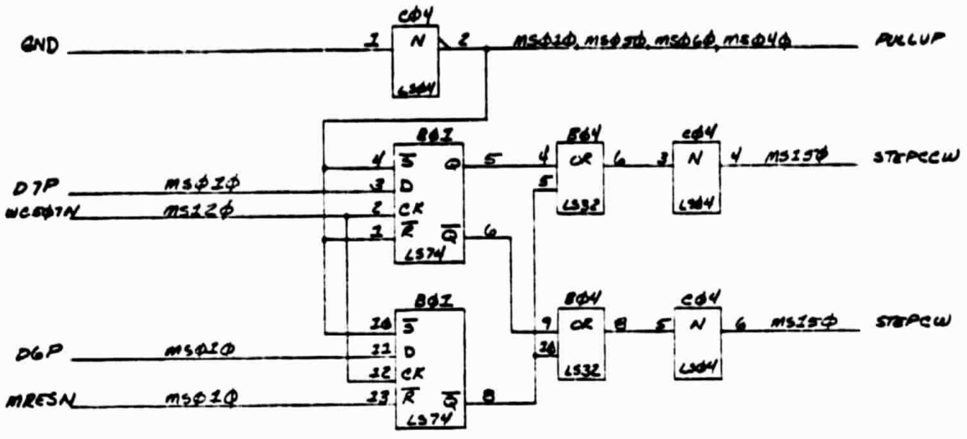
MSΦ4Φ
MOTOR SPEED CONTROL
PULSE WIDTH MODULATOR

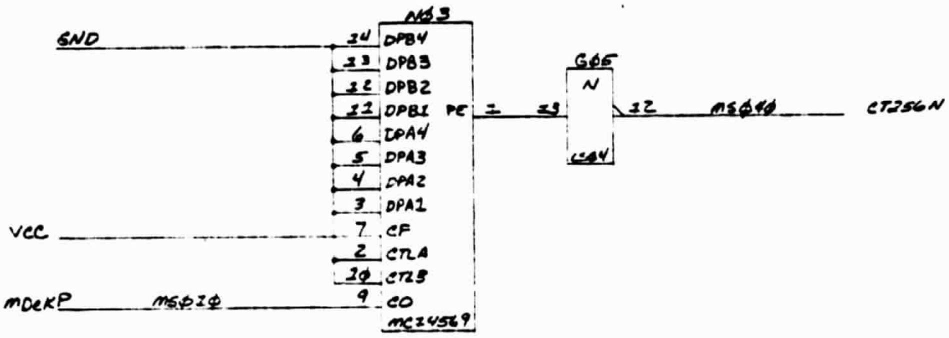
DCB 7/3Φ/8Φ



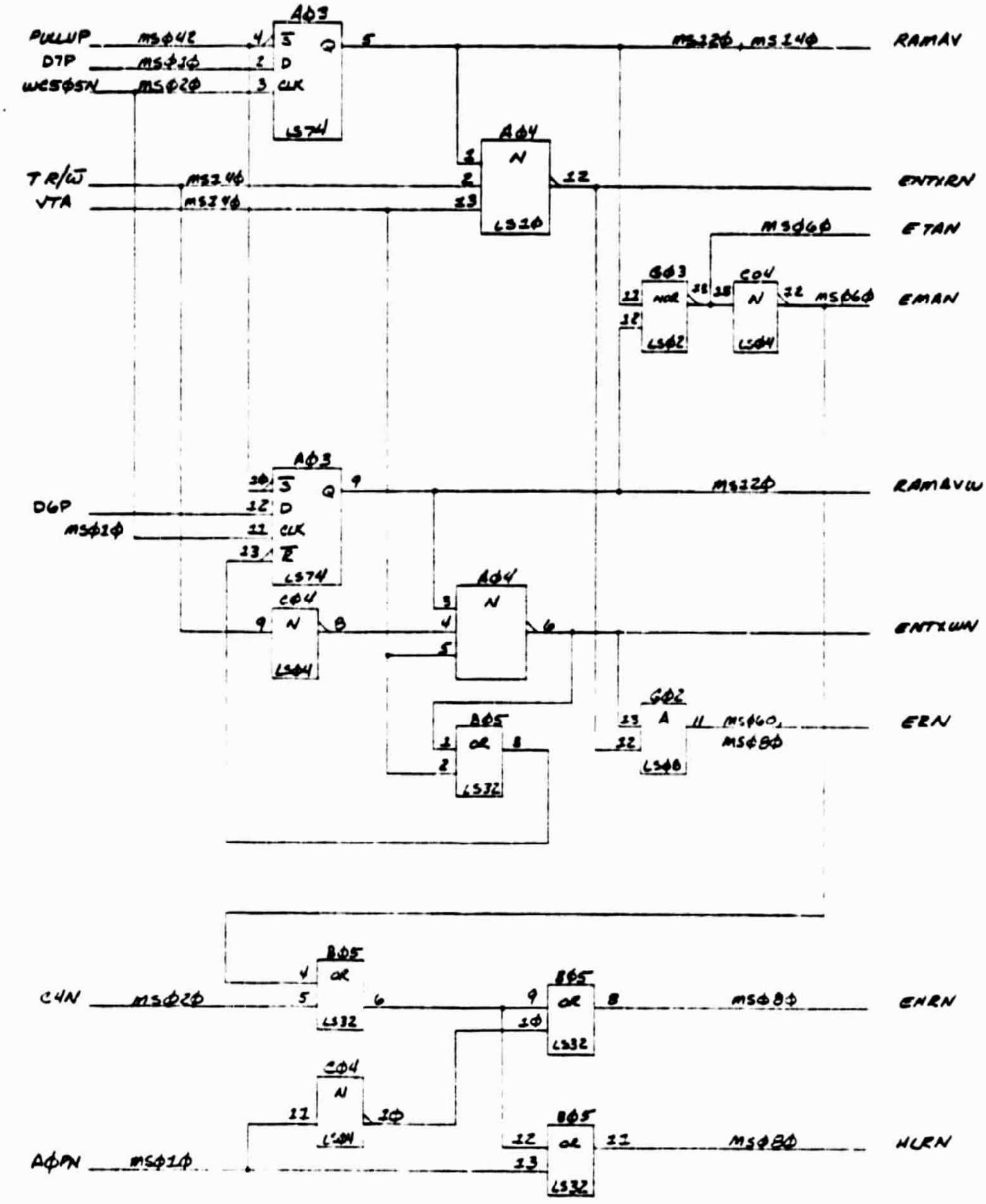
MS041
TWO'S COMPLEMENT
MOTOR DRIVER

DCB 7/30/86



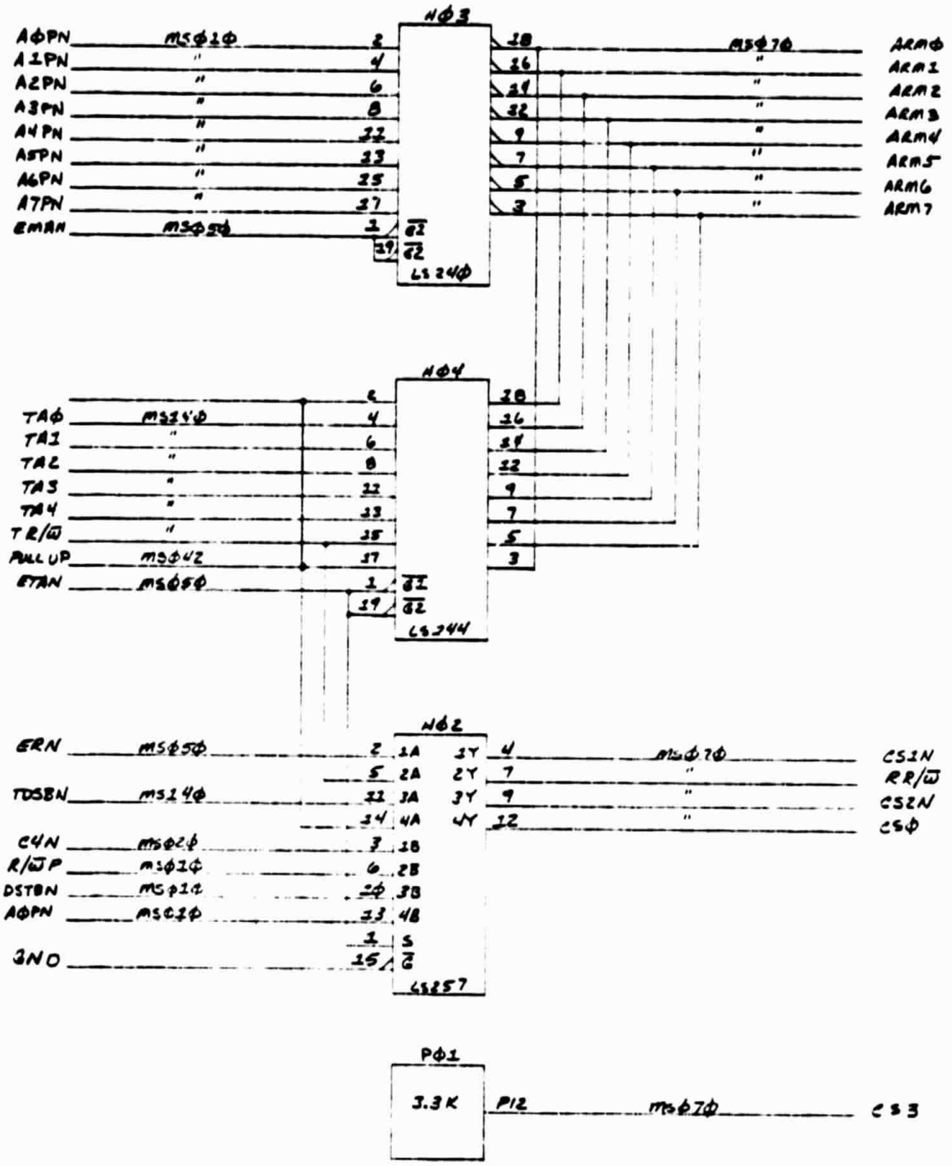


MSØ42
GYRO I/O
MOTOR SPEED COUNTER
JCB 7/30/80



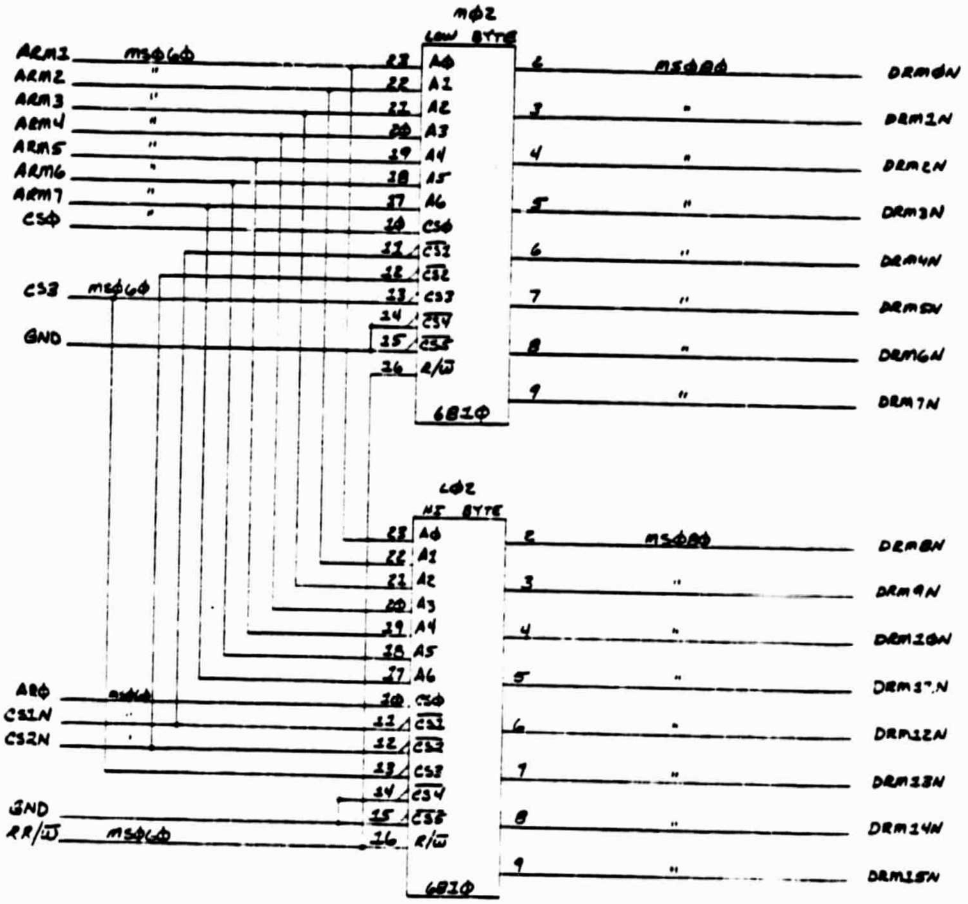
MSØ5Ø
RAM CONTROLS

DCB 7/30/80



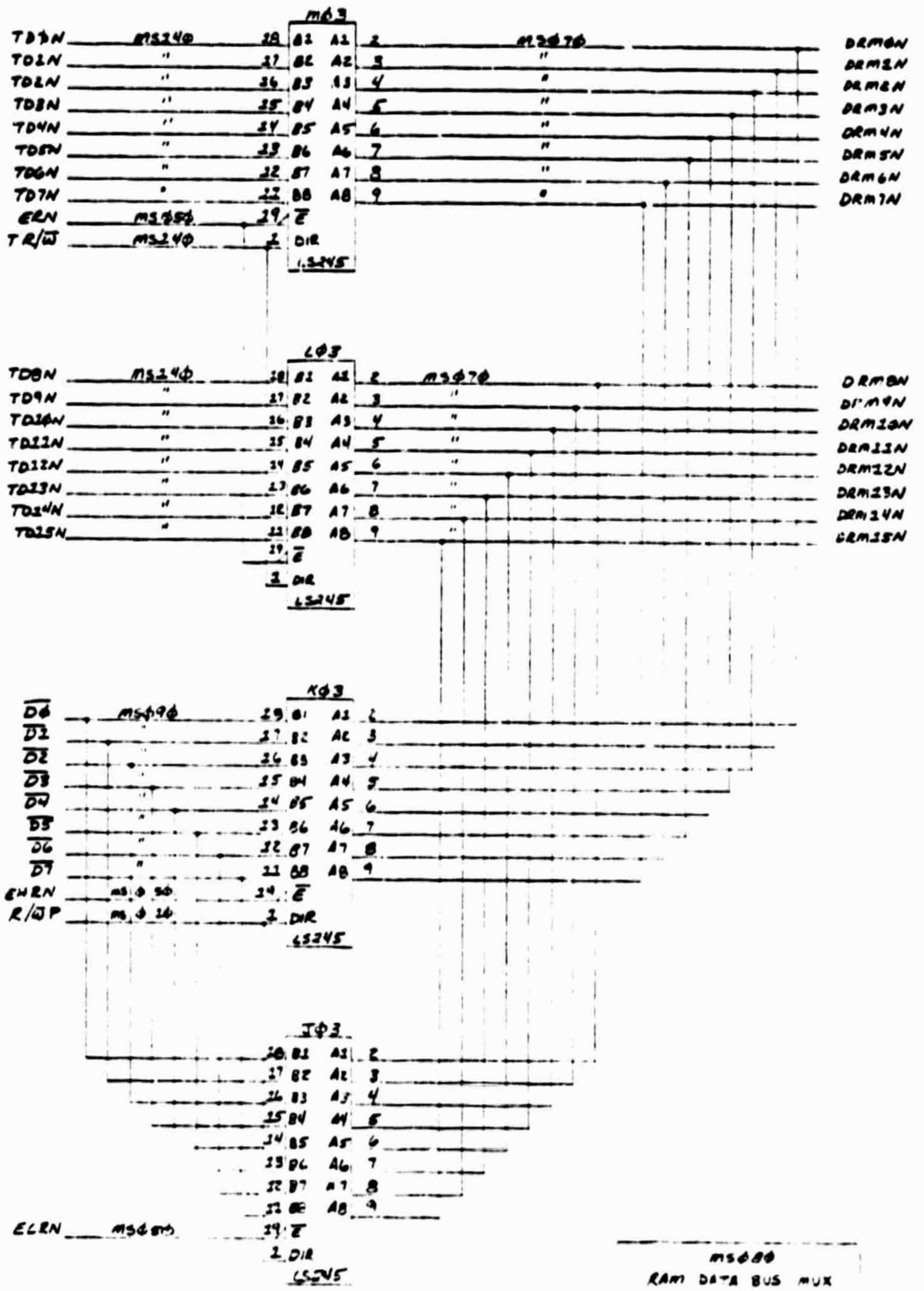
MSΦ6Φ
RAM ADDRESS
& CS MUX

DCB 7/30/80



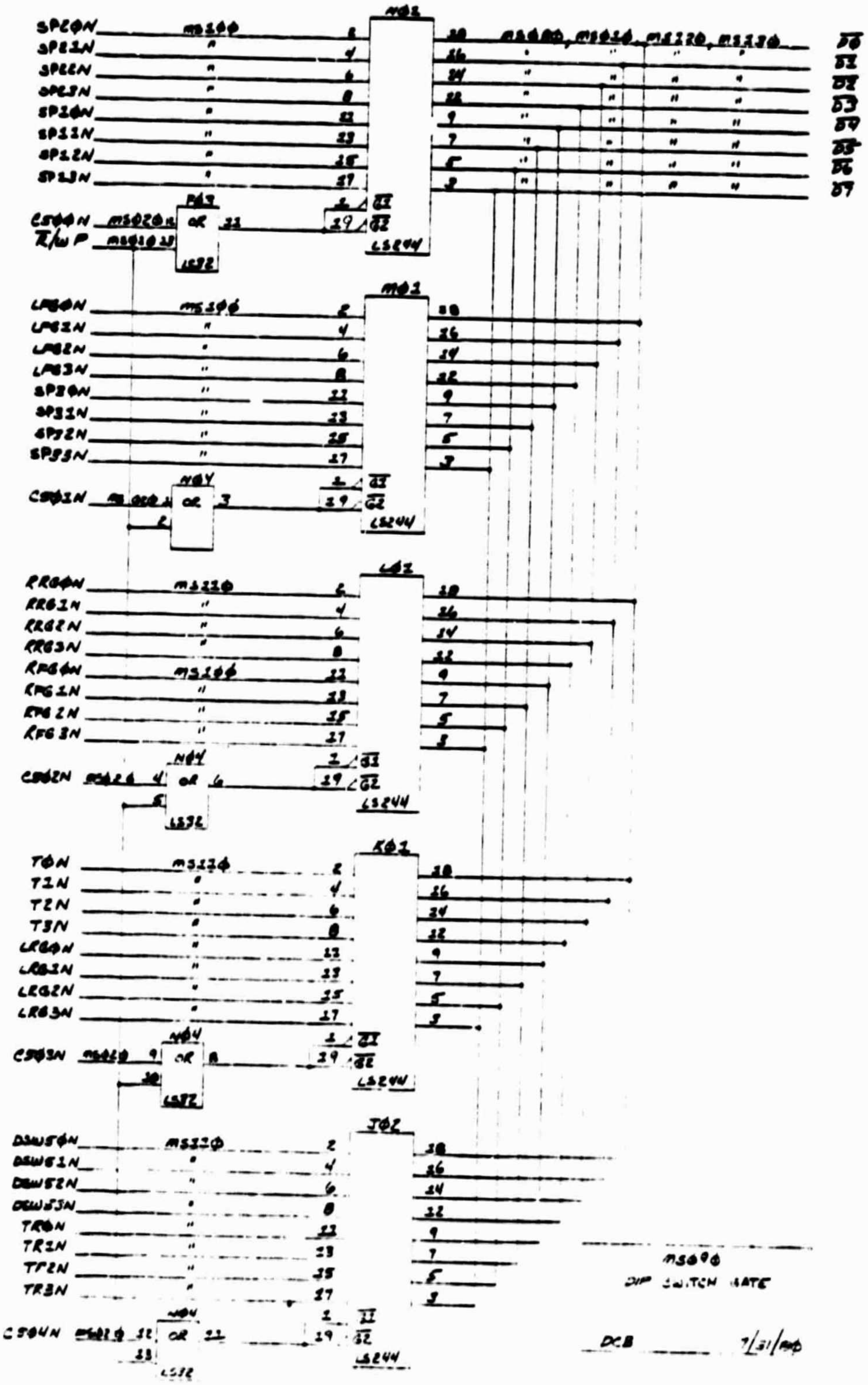
MSØ 7Ø
RAM

DCB 7/30/80



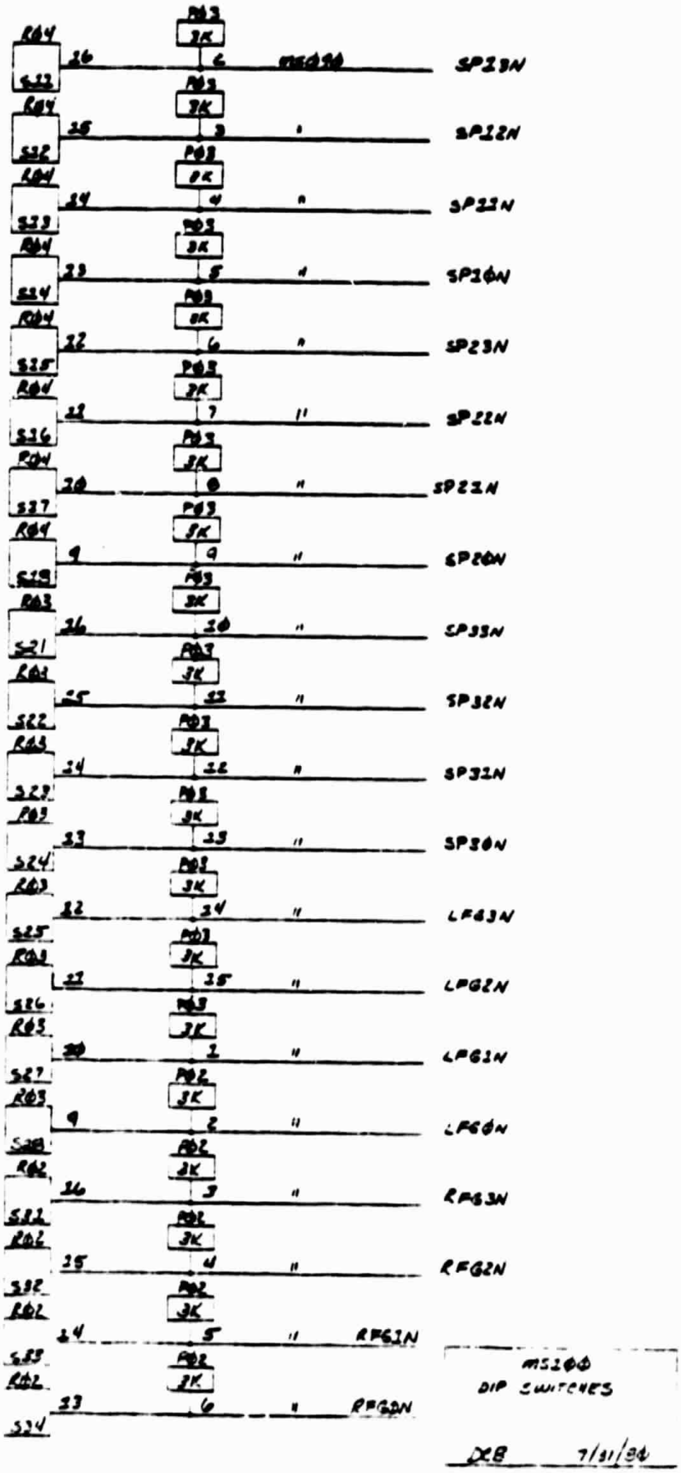
MSØ8Ø
RAM DATA BUS MUX

DCB 7/My82

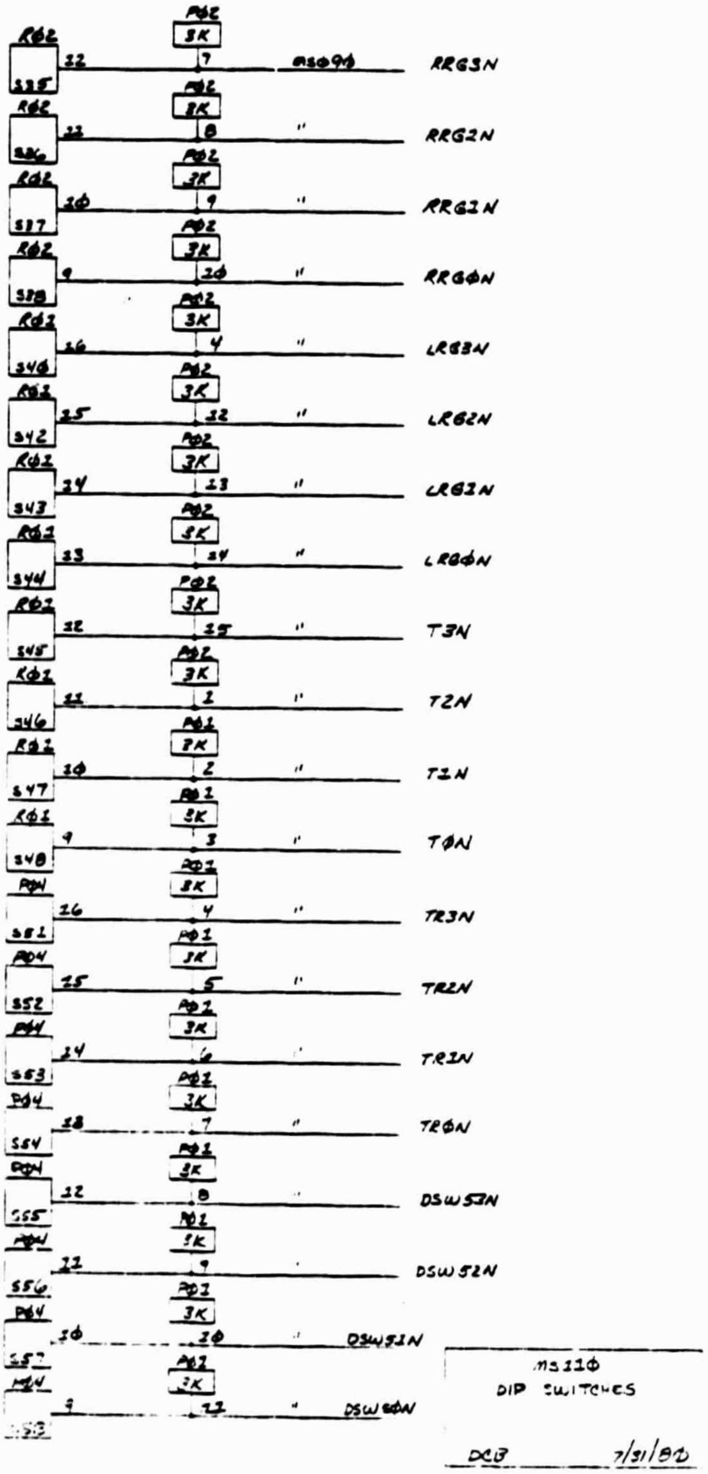


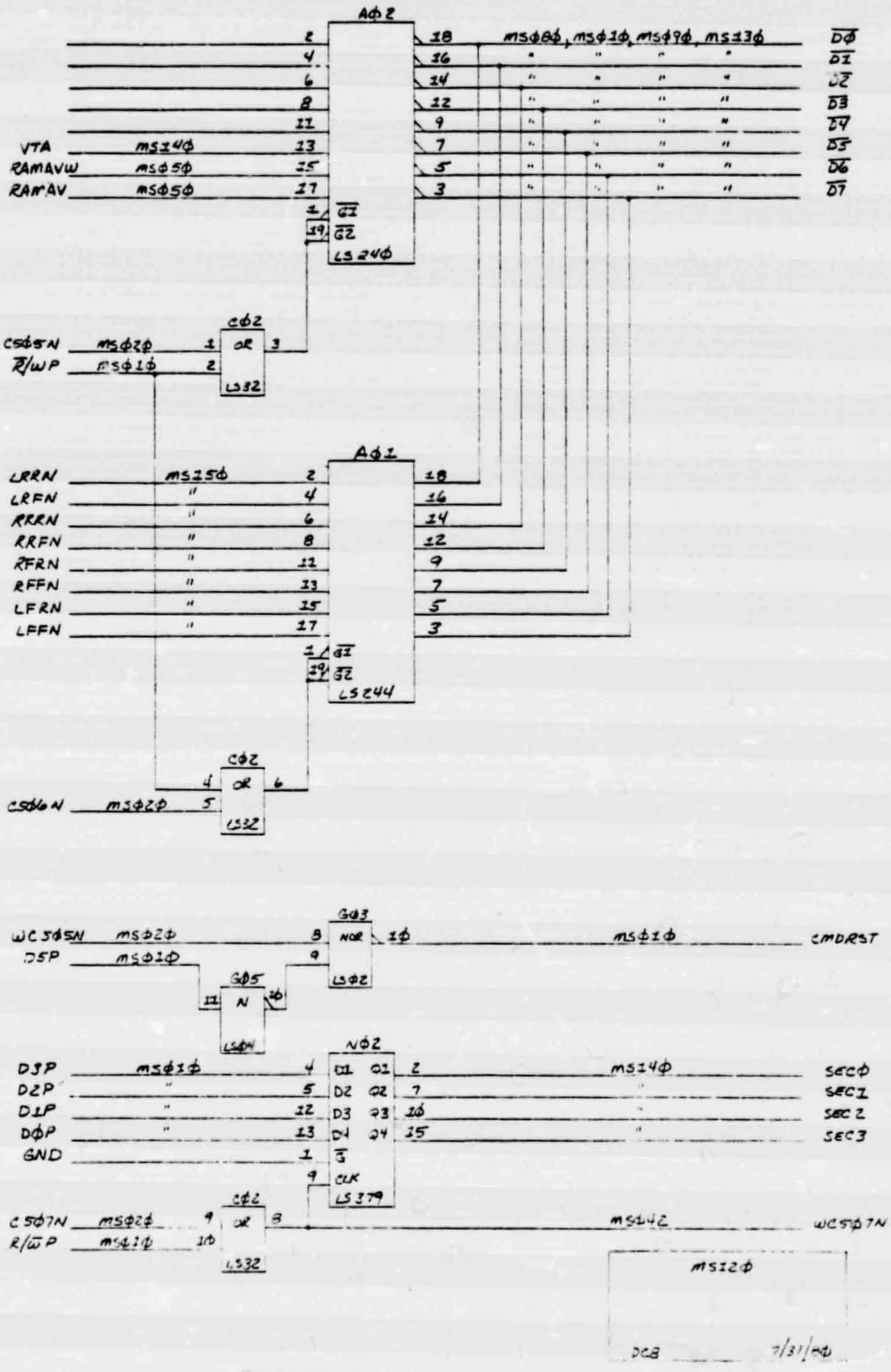
MS090
DIP SWITCH GATE

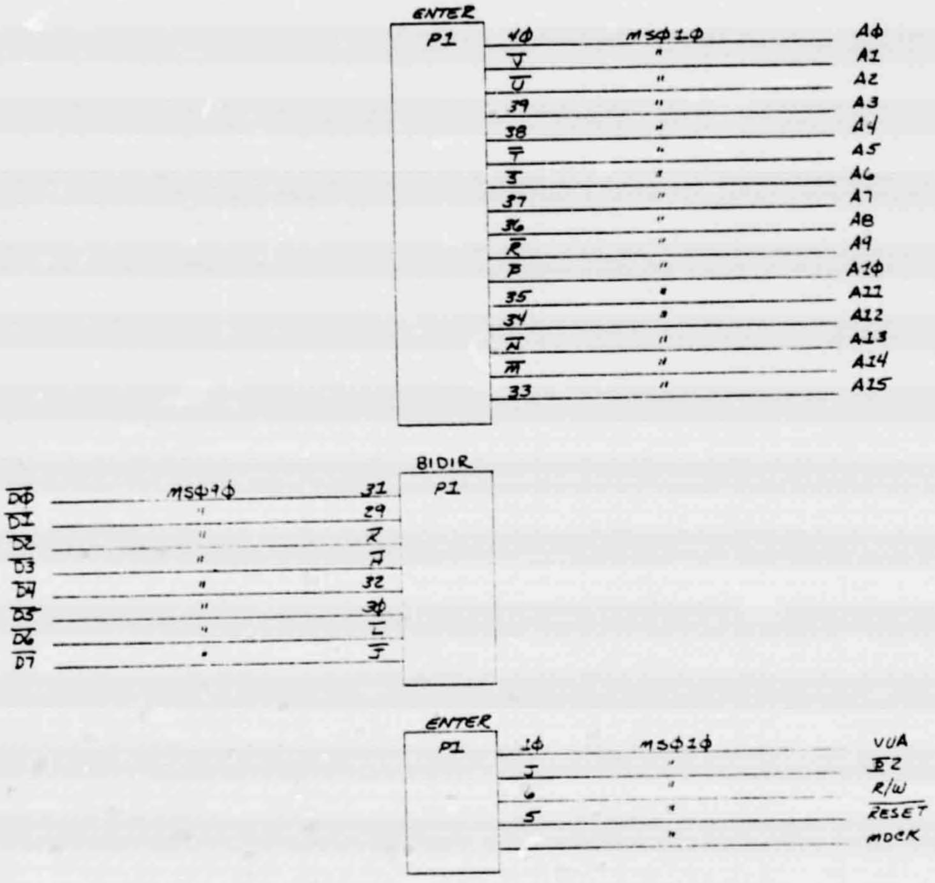
DCB 7/31/80



ORIGINAL PAGE IS
OF POOR QUALITY

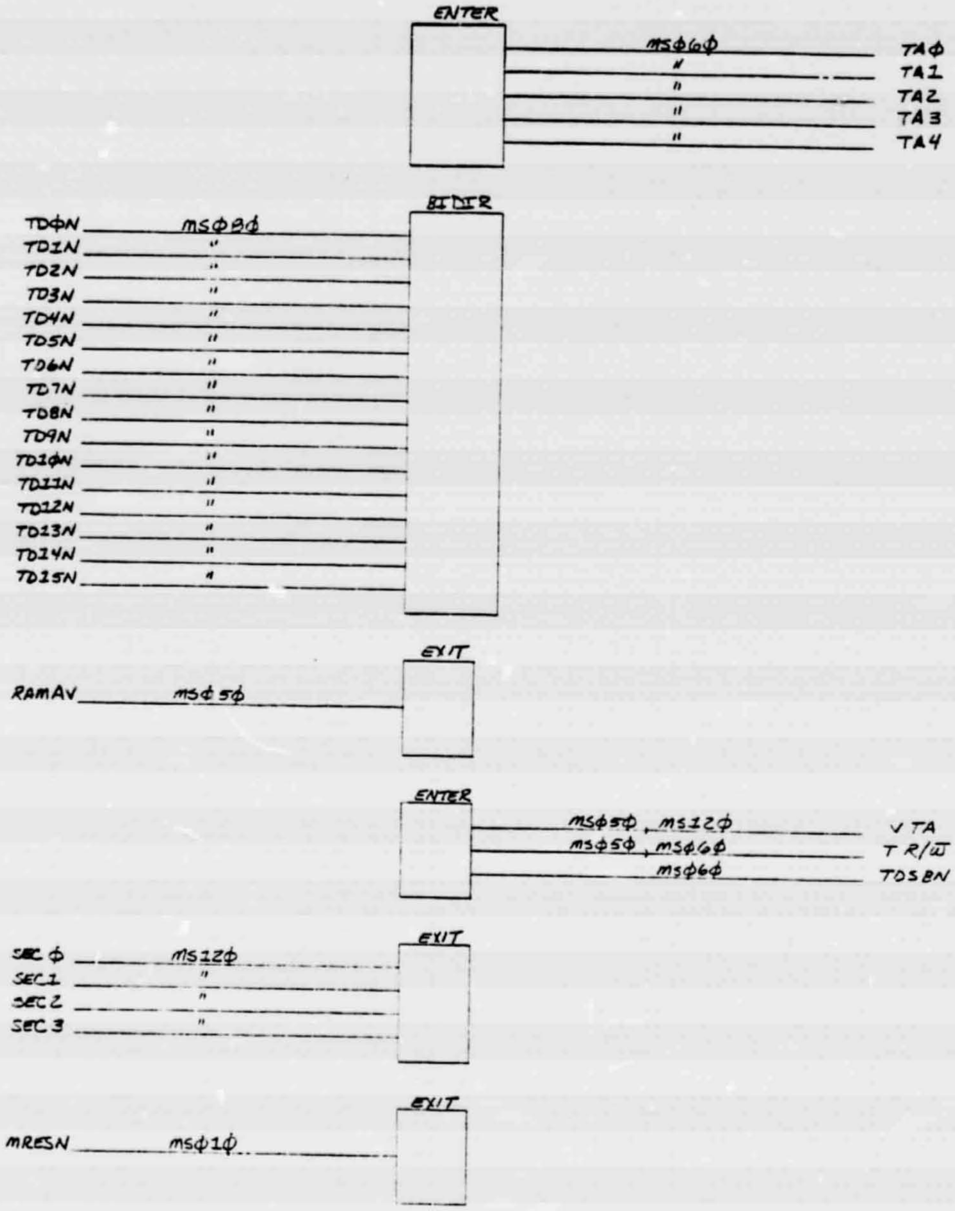

RØ2
12
S35
PØ2
3K
7
MSØ9Ø
RRG3N
RØ2
11
S36
PØ2
3K
8
"
RRG2N
RØ2
1Ø
S37
PØ2
3K
9
"
RRG1N
RØ2
9
S38
PØ2
3K
1Ø
"
RRGØN
RØ1
16
S4Ø
PØ2
3K
4
"
LRG3N
RØ1
15
S42
PØ2
3K
12
"
LRG2N
RØ1
14
S43
PØ2
3K
13
"
LRG1N
RØ1
13
S44
PØ2
3K
14
"
LRGØN
RØ1
12
S45
PØ2
3K
15
"
T3N
RØ1
11
S46
PØ2
3K
1
"
T2N
RØ1
1Ø
S47
PØ1
3K
2
"
T1N
RØ1
9
S48
PØ1
3K
3
"
TØN
PØ4
16
S51
PØ1
3K
4
"
TR3N
PØ4
15
S52
PØ1
3K
5
"
TR2N
PØ4
14
S53
PØ1
3K
6
"
TR1N
PØ4
13
S54
PØ1
3K
7
"
TRØN
PØ4
12
S55
PØ1
3K
8
"
DSW53N
PØ4
11
S56
PØ1
3K
9
"
DSW52N
PØ4
1Ø
S57
PØ1
3K
1Ø
"
DSW51N
PØ4
9
S58
PØ1
3K
11
"
DSW5ØN
MS11Ø
DIP SWITCHES
DCB
7/31/80


AΦ2
ms Φ8Φ, ms Φ1Φ, ms Φ9Φ, ms 13Φ
DΦ
D1
D2
D3
D4
D5
D6
D7
VTA
ms14Φ
RAMAVW
msΦ5Φ
RAMAV
msΦ5Φ
G1
G2
LS24Φ
CΦ2
OR
CSΦ5N
msΦ2Φ
R/WP
msΦ1Φ
LS32
AΦ1
LRRN
ms15Φ
LRFN
RRRN
RRFN
RFRN
RFFN
LFRN
LFFN
LS244
CΦ2
OR
CSΦ6N
msΦ2Φ
LS32
GΦ3
NOR
WCSΦ5N
msΦ2Φ
DSP
msΦ1Φ
msΦ1Φ
CMDRST
LSΦ2
GΦ5
N
LSΦ4
NΦ2
D3P
msΦ1Φ
D2P
D1P
DΦP
GND
D1 Q1
D2 Q2
D3 Q3
D4 Q4
CLK
LS379
ms14Φ
SECΦ
SEC1
SEC2
SEC3
CΦ2
OR
CSΦ7N
msΦ2Φ
R/W̄P
ms41Φ
LS32
ms14Z
WCSΦ7N
ms12Φ
DCB
7/31/80



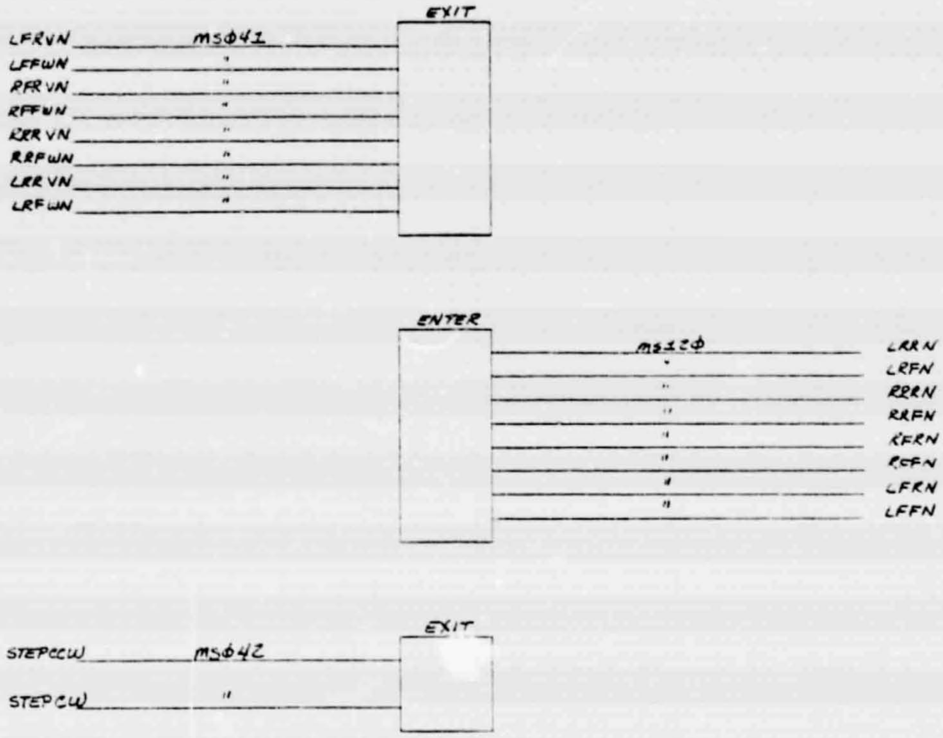
MS13Φ
BACK PLANE CONNECTORS

DCB 7/31/8Φ



MS14Φ
TELEMETRY CONNECTOR

JCB 7/31/8Φ



MS15Φ
MOTOR SPEED CONVERTER
& MICRΦ SWITCH

DCB 7/31/80

APPENDIX C

BOARD BUILD CONVENTIONS

T.e board was built following a few conventions that insure a clean board that is easily debugged and altered. The socket locations are shown in Figure 8. All sockets are soldered to the board at the power and ground pins. In addition, the bypass capacitors are also soldered directly across power and ground. Before wire wrapping the board, a From-To wire list was developed from the logic diagrams. After double checking the wire list, the board was wired from it. The key advantage to using a wire list is that the wires may be assigned wiring levels. Only two levels of wire were used (see Figure 9). In this way, wire nets may be easily broken and altered as opposed to the daisey chain method which is normally employed when wiring from a logic diagram.

The logic was designed using Low Power Shottkey TTL. The only exceptions are the Motorola RAMs and counters and the high voltage motor drive buffers. The "LS" logic was used to keep power consumption down since the rover runs on batteries.

Two 14 pin sockets are available for additions at locations B02 and B03.

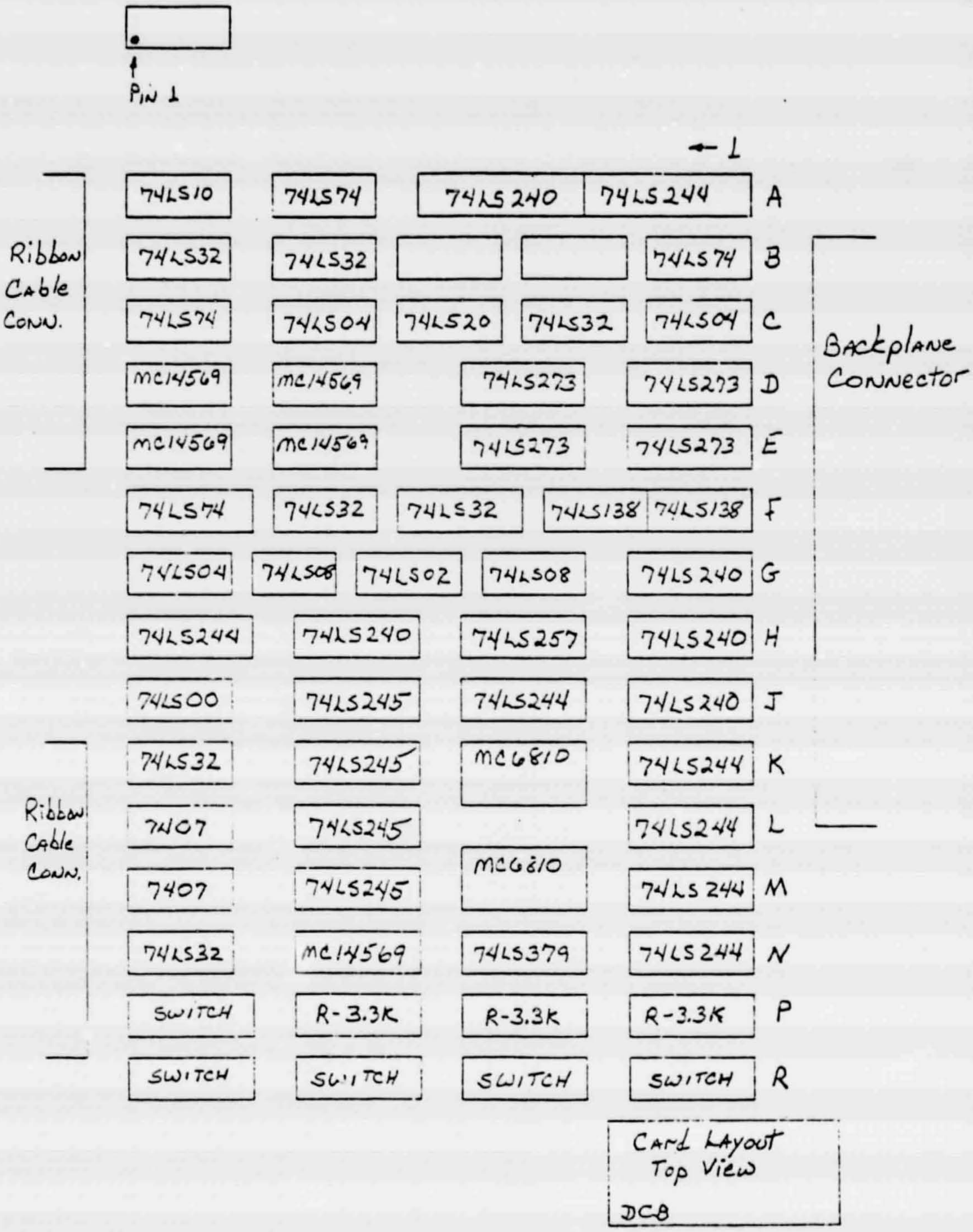


Figure 8. Board Layout.

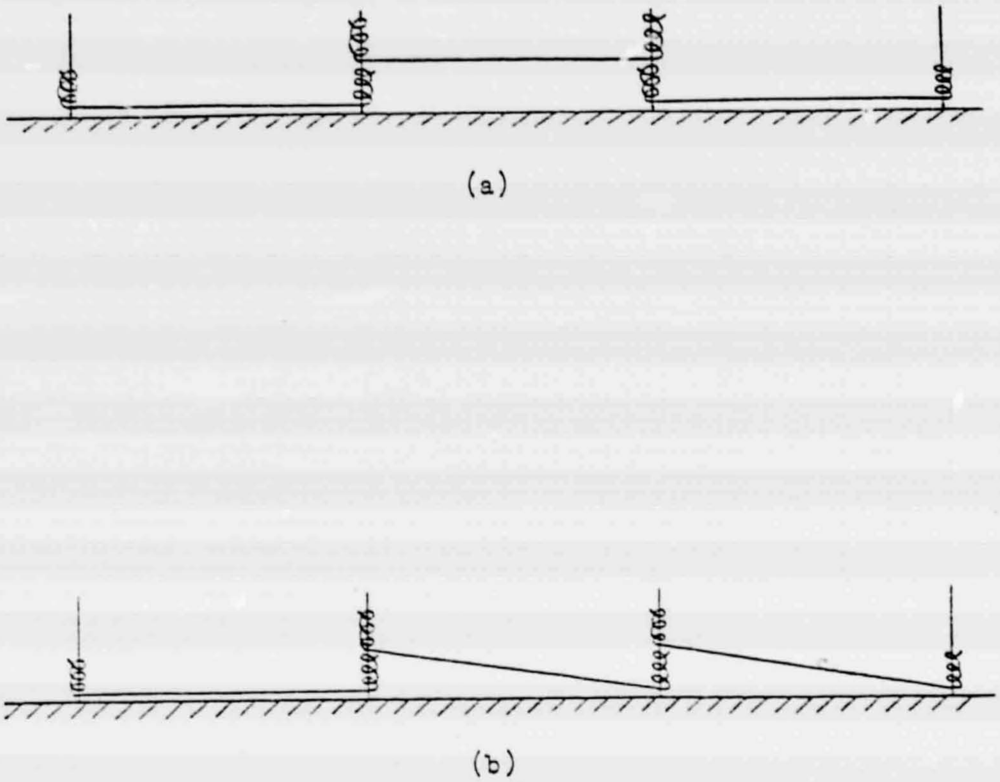


Figure 9. Wiring Methods.

(a) Two Level Wiring Method
(b) Daisey Chain Wiring Method