87
6-73-80

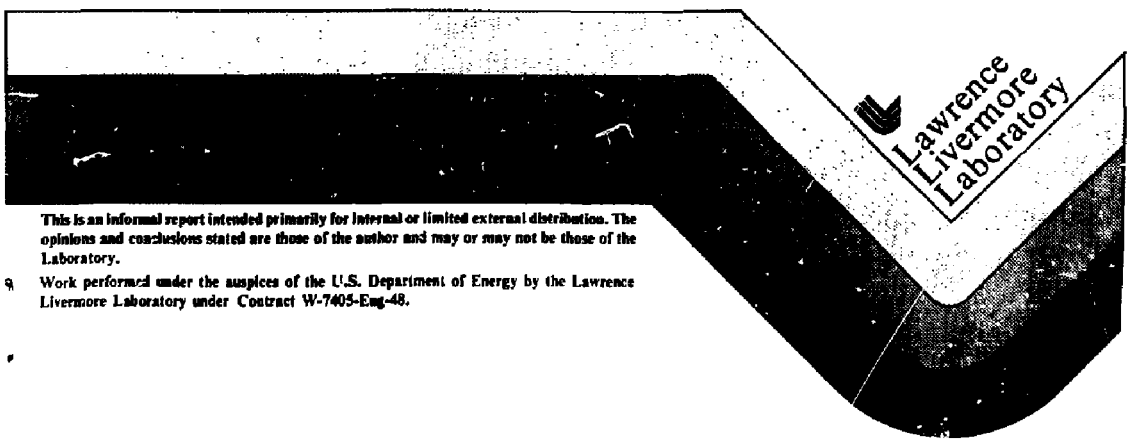
UCID-18701

MASTER

STEPPING MOTOR CONTROL PROCESSOR REFERENCE MANUAL

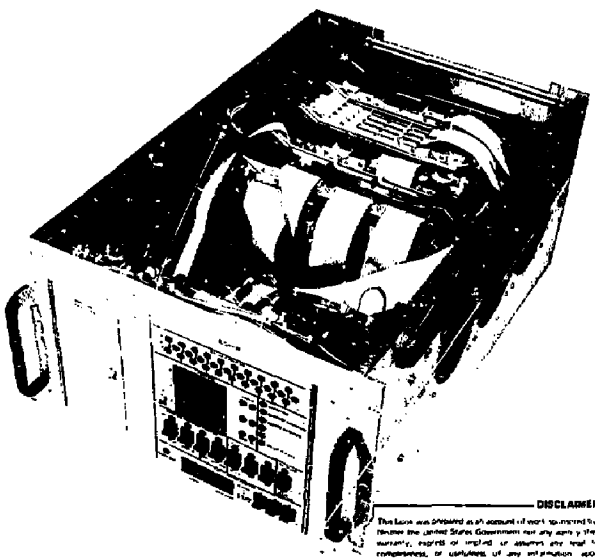
F. W. Holloway
P. J. VanArsdall
G. J. Suski
R. G. Gant
M. Rash

June 6, 1980



STEPPING MOTOR CONTROL PROCESSOR

Reference Manual Volume 1



SHIVA

Alignment Controls Team

F. W. Holloway
P. J. VanArsdall
G. J. Suski
R. G. Gant

EDITED BY: M. Rash NOVEMBER, 1977

LAWRENCE LIVERMORE LABORATORY
University of California Livermore, California 94550

TABLE OF CONTENTS - Volume 1

Worked performed under the auspices of the U.S. Department of Energy by Lawrence Livermore Laboratory under Contract No. W-7405-Eng-48.

TABLE OF CONTENTS (Con't)

TABLE OF CONTENTS (Con't)

LIST OF FIGURES

LIST OF TABLES

LIST OF PHOTOGRAPHS

ACKNOWLEDGEMENTS

1. This processor is housed within a Shiva standard LSI-11 chassis which was designed to meet a wide range of applications throughout the shiva control project by the Power Conditioning Controls Team led by P. Rupert. This chassis can be configured in a wide range of combinations with various numbers of power supplies, backplanes and custom panels.

2. Much of the software for this processor was developed on REBEL/BASIC based LSI-11's, and although it doesn't itself utilize REBEL/BASIC internally, it is optimized for connection to processors that do. REBEL/BASIC was designed and developed by J. Greenwood and M. Zarnstorff.

I. INTRODUCTION

Since its inception the control systems design of the SHIVA high energy laser facility included very large quantities of stepping motor driven devices for the various alignment tasks of the 20 laser beam system. Many applications also required corresponding electro-optical sensors with analog-to-digital conversion equipment to measure system parameters and performance for closed-loop control applications. To meet the needs of reliably controlling this array of motor driven manipulators and sampling the sensors, the Alignment Controls Team has designed a modular set of hardware and software for local and distributed control based on the Digital Equipment Corporation LSI-11 microcomputer.

A standard alignment package has been assembled from these components termed the "Stepping Motor Control (SMC) Processor." The package is flexible and contains a powerful ROM coded software set which provides most of the common functions necessary for typical locally-controlled alignment motions. More computing power is available to the user for closed-loop automatic control by remote operation of this processor from a higher level processor via the SHIVANET protocol.

This manual is intended to serve several purposes. The first goal is to describe the capabilities and operation of the "SMC processor' package from an operator or user point of view. Secondly, the manual will describe in some detail the basic hardware elements and how they can be used effectively to implement a step motor control system. Practical information on the use, installation and checkout of the hardware set is presented in the following sections along with programming suggestions. Available related system software is described in this manual for reference and as an aid in understanding the system architecture.

Section two presents an overview and operations manual of the SMC processor describing its composition and functional capabilities. The local control panel operators manual is included for reference and description of local mode of operation. Information on the setup and use of the local control panel is given. The description in this section also serves to illustrate the interconnection of the basic hardware set to create user-specific motor control systems not directly based on the SMC processor architecture.

Section three contains hardware descriptions in some detail for the LLL-designed hardware used in the SMC processor. Basic theory of operation and important features are explained. Programming notes are discussed in the text to aid in the understanding of system software and to provide the necessary information for user written software drivers

when required. Board adjustments and jumper strapping are also described to assist in installation.

Section four provides detailed descriptions of the capabilities and use of alignment system software developed for the SMC processor and its hardware elements. In particular, the SMC processor remote control routines and the SHIVANET communications network are described. Further software information and software listings are contained in Volume II.

II. STEPPING MOTOR CONTROL PROCESSOR

2.1 Purpose

Alignment control of the 20 beam SHIVA laser involves over 600 axis of motion (such as gimbals, translators and filter wheels) at widely dispersed points along the laser chains. In addition, several hundred more motions are required for the beam diagnostic sensors in the laser facility. Commercially available 4-phase stepping motors were selected to implement these motions using computer based control techniques. Stepping motor drive systems can meet the most demanding accuracy requirements and their exclusive use in positioners reduces considerably the range of interfacing effort needed in the control system.

The need for a practical and economic system to control these quantities of motors, along with some unusual constraints peculiar to the laser system, led to the development of the SMC processor. The system was implemented to meet the following criteria:

- complete digital control of up to 30 4-phase stepping motors in one flexible package. Thirty motors corresponds to the largest geographical concentration of motors in the laser system.
- motors grouped in threes (eg. X,Y,Z axis) to a given physical location as this arrangement is most commonly encountered. Other groupings can be accommodated by special cabling not described herein.
- local control panel with useful set of functions for open-loop stand alone or backup operations
- control from a higher level remote computer for closed-loop operation
- powerful set of remote commands for motor control and status verification
- reliable interprocessor communication network (SHIVANET) over optically isolated serial links at 9600 baud (code written for PDP-11 computers)

- complete power-fail hardware and software to preserve alignment system status during indeterminate power interruptions or DC supply failures

- ROM coded software to provide reliable program storage and rapid restart capabilities without down line loading or disk storage

- motor and limit switch circuits optically isolated from the computer for protection from the severe EMI environment of the laser system.

- motor power enabled only during motion to reduce heat dissipation near sensitive optical-mechanical hardware

- synchronous clock driven motor translators for constant speeds up to 300 steps/second to meet reasonable alignment speed requirements

- provisions for potentiometer feedback from each motor for position encoding (12 bit A/D)

- buffer capability provided in SMC processor for power-fail safe data storage for remote higher-level processor

- error detection logic for fault diagnosis warns of interface failure, communication link problems and disconnected motor cables.

The remainder of this section outlines the hardware, software and operation of the SMC processor.

2.2 Description

The standard SMC processor and peripheral motor equipment are shown on the system interconnection block diagram, Figure 1. This diagram concentrates several types of information on one drawing. Most importantly, the hardware structure and its components are illustrated to provide the user with a good picture of the system architecture. The interconnection cables are identified by type and destination. LLL drawing package numbers are also shown for easy reference to actual hardware or further documentation. Important system device addresses are given along with other parameters affecting system configuration and installation. This particular architecture is general but is shown as implemented for the spatial filter control system.

SMC PROCESSOR

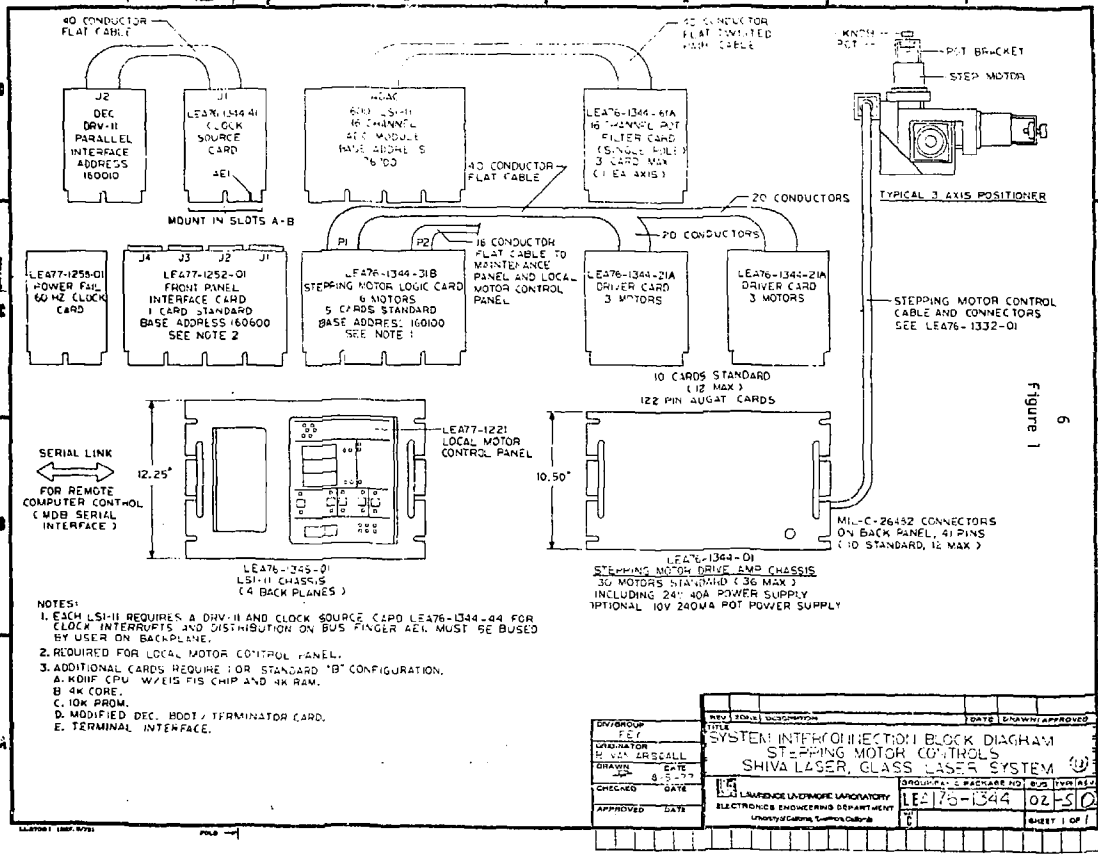


Figure 1

NOTES:

1. EACH LSI-11 REQUIRES A DRV-11 AND CLOCK SOURCE CARD LEA76-1344-44 FOR CLOCK INTERRUPTS AND DISTRIBUTION ON BUS FINGER AE1 MUST BE BUSED BY USER ON BACKPLANE.
2. REQUIRED FOR LOCAL MOTOR CONTROL PANEL.
3. ADDITIONAL CARDS REQUIRE J OR STANDARD "B" CONFIGURATION.
   A. KDIIF CPU W/EIS FIS CHIP AND 4K RAM.
   B. 4K CORE.
   C. 10K PROM.
   D. MODIFIED DEC. BOOT / TERMINATOR CARD.
   E. TERMINAL INTERFACE.

REV	ZONE	DESCRIPTION	DATE	DRAWN/APPROVED

DIV/GROUP: EE
ORIGINATOR: R. VAN ARSDALL
DRAWN: DATE 8-5-77
CHECKED: DATE
APPROVED: DATE

TITLE: SYSTEM INTERCONNECTION BLOCK DIAGRAM STEPPING MOTOR CONTROLS SHIVA LASER, GLASS LASER SYSTEM (U)

LAWRENCE LIVERMORE LABORATORY
ELECTRONICS ENGINEERING DEPARTMENT
University of California, Livermore, California

GROUP-FILE-PACKAGE NO.: LEA 76-1344 | 02-5-10 | BUS | TYP | REV
SIZE: D
SHEET 1 OF 1

The standard unit consists of:

A) A SHIVA standard LSI-11 chassis with 4 backplanes, two 25 amp 5 volt supplies, ± 12 volt supplies and the cards listed below

B) SMC processor local control panel mounted on the LSI-11 chassis

C) Stepping motor drive amplifier chassis and cards

D) Standard SHIVA stepping motor cables LEA 76-1332 as required.

TABLE 1

LSI-11 CARD CONFIGURATION

SLOT		
1	Front Panel Interface	
2	CPU/4K RAM (4-8K)	
3	Terminal DLV11	/// Network DLV-11
4	4K Core (0-4K)	
5	DRV-11 for Motor Clock	/// Powerfail Detect
6	Motor Clock	/// 4K Prom (8-12KO
7	----	/// 4K Prom (12-13K)
8	----	----
9	Stepping Motor Logic, Card 1	
10	Stepping Motor Logic, Card 2	
11	Stepping Motor Logic, Card 3	
12	Stepping Motor Logic, Card 4	
13	Stepping Motor Logic, Card 5	
14	ADAC A/D Converter (Optional)	
15	ADAC A/D Converter (Optional)	
16	----	/// Revll Term/Boot

The functions of the cards are briefly described below. More detailed information can be found in Section III for the LLL-designed hardware.

LSI-11 Hardware

The control panel interface card provides the interface to the local control panel from the computer. All switches, LEDs and displays are controlled with this card. The front panel and software driver provide extensive control capabilities for the unit as a stand-alone system. More information is available on control panel operation and capability in the next subsection.

The terminal and network DLV-11s are serial interfaces to an optional teletype (for maintenance purposes) and a remote control processor respectively. All upstream communication is routed through the network interface at 9600 baud by the SHIVANET communications software.

The 4K core unit is used for nonvolatile storage of data areas and control parameters. The control data base is maintained here and important parameters are stored during a power-down sequence for subsequent power up restoration. Buffer space is also available in core for the remote user by special software calls which are provided. This space can be used by a remote processor for power fail safe data storage. These calls are described briefly later in this manual.

A DEC DRV-11 parallel interface has been included to provide interrupt circuitry for the step motor system clock. A feature is currently being added in connection with this card to disable the motor clock whenever the power is off in the Stepmotor Drive amp chassis to prevent loss of position information.

The powerfail detect card monitors all three DC power supplies and the AC line. The circuitry generates the proper power fail and restart sequences for the LSI-11 CPU. The card also generates the line clock signal for the LSI-11 clock processor. Special circuitry has been included to allow software simulated power-fail sequences for restarting the processor from a remote location.

All stepping motors on the SMC processor are interrupt driven by the motor clock board. An oscillator generates timing pulses that are distributed to the stepmotor logic boards for synchronization on bus finger AE1. Note this pin must be user bussed to other types of cards. The clock signal also drives the DRV-11 interrupt logic for software synchronization.

Two PROM cards are used to store the 10K byte ROM software operating system for the SMC processor. All control and communication functions are handled by code permanently stored in the ROM chips. This approach insures rapid restart after a power fail even if not connected to another computer, and prevents the possibility of accidental code alteration. PROM chips of the UV erasable type are used, permitting occasional software changes.

Stepping motor logic cards provide the computer interface and step logic to control motors and limit switches in the system. From one to five cards may be installed depending on the number of motors to control. Each card controls six stepping motors in groups of 3 by optically isolated connections to cards in the stepmotor drive amp chassis.

From 0 to 3 converters (ADAC 600 LSI-11) may be installed to provide encoding of motor shaft positions via feedback potentiometers. Current software can read from 16 channels per board. Note: Special ADAC-available features such as programmable gain have not been implemented.

Stepping Motor Driver

The stepping motor drive system is packaged in a 10 1/2" rack chassis complete with card cage and power supply for 30 stepping motors. Any four phase motor drawing up to 2 amps per phase at 28V or less can be used simply by loading the appropriate series current-limiting resistors on the chassis resistor deck. The Augat card cage holds up to 10 3-axis stepmotor drive amp cards and has wired connectors for three pot filter cards. Connections to manipulators are made through the SHIVA standard stepmotor cable.

Manual Backup

Backup operation of any stepping motor driven system is possible for those users who adhere to the standard cable assignment. A manual Stepping Motor Test Box LEA77-1215 has been designed have been built) which can control any three axis motor set individually. The unit can be used whenever a complete computer system is unavailable or unnecessary as in testing of mechanical designs or simple open-loop alignment operation of up to three motions. About 12 units are in use.

Variable slew rates and step mode are available with complete limit switch circuitry. A resetable counter has been included to keep track of steps commanded for the selected axis. A meter on the box can monitor the performance of potentiometers installed on the stepmotors. Some units automatically check attached motors for case shorts. The test box is self contained and portable, making it useful for acceptance testing off site.

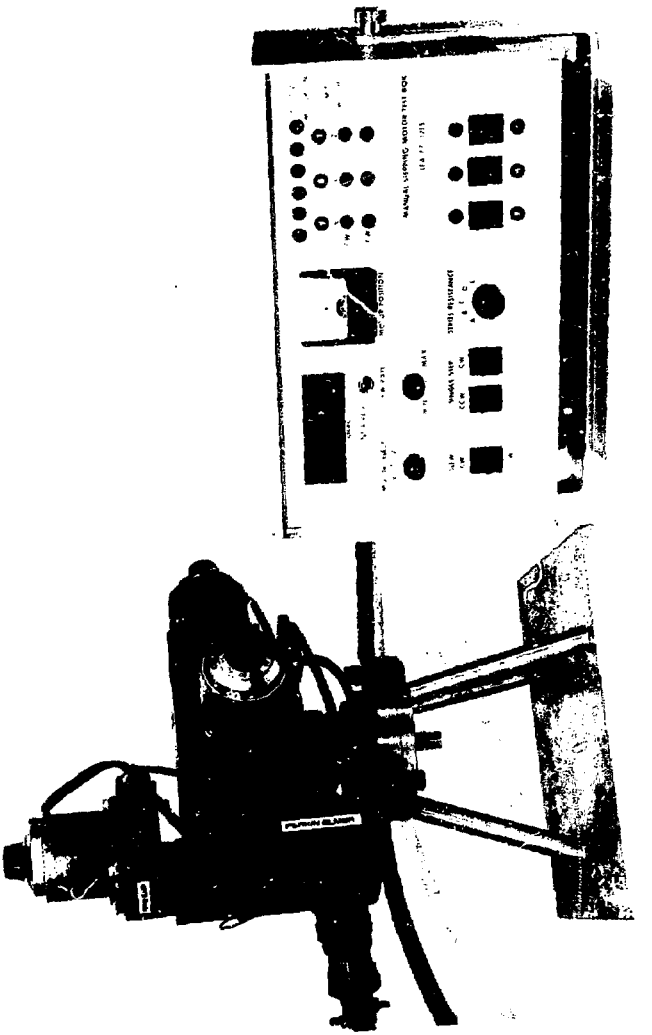
2.3 SMC Processor Local Control Operation

General

The local control panel (see Figure 2) provides the control and status functions normally required for open loop operation of stepmotor driven alignment systems. The SMC processor may be commanded by this panel in stand-alone or backup situations as well as by a remote computer over the SHIVANET communications network. Important features are summarized below:

- dialup selection of any of ten chain numbers (axis sets)

- slew control of each axis motor (X,Y,Z) in selected chain



MANUAL STEPPING MOTOR TEST BOX

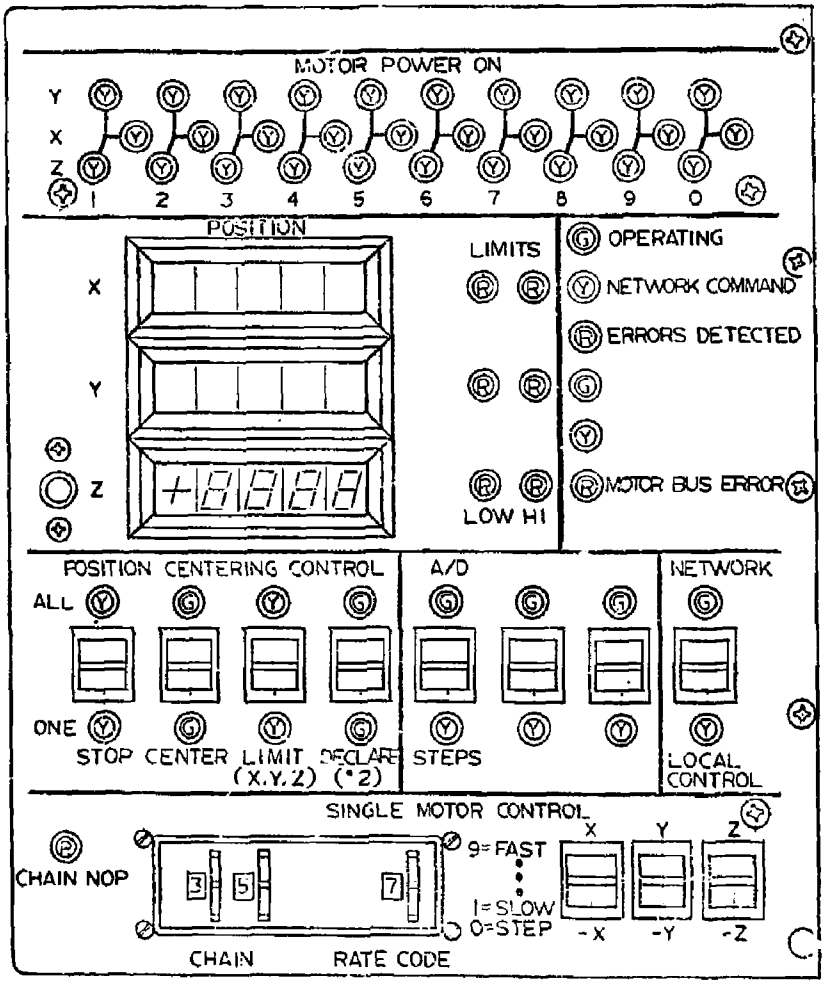


Figure 2.
SMC Processor Front Panel

- dialup slew speed control
- single step control
- effective general purpose position centering control group
- BCD position readout on 7 segment displays for X, Y, Z axis
- useful set of LED indicators for quick identification of system status
- hardware driven motor power lamps for each motor
- remote/local control select
- simultaneous operation possible from remote and local controls
- mappable chain and motor configurations
- control of 30 stepmotor driven devices
- provisions for potentiometer position encoding on each motor

The control panel software package is called MTBCTL and is implemented in ROM for nonvolatile program storage. Only one version of ROM code is necessary to implement various configurations of chain number-to-channel number mapping, number of operational channels, X-Y-Z axis to motor mapping and potentiometer scaling option.

Operation

Front panel controls are grouped into several classes according to their function and effect on each other as follows (see Figure 3):

SINGLE MOTOR CONTROL

1. CHAIN select is two decimal thumbwheel operated switches that indicate to the control program which set of three axis (X,Y,Z) is to be controlled by the motor slew controls and which motor positions and limit switch status is to be displayed. The actual correspondence between the dialup number and motor number (1-30) can be mapped using the configuration switches on the control panel interface board (see below).

2. CHAIN NOP is a red LED that is blinked whenever a non-valid chain number (as defined by the configuration switches) is dialed up. No action can be initiated in this condition.

3. RATE CODE is a decimal thumbwheel switch that selects one of nine slewing speeds or step mode only. These speeds are defined in the following table:

TABLE 2

SLEW RATE

Rate Code	Steps/Sec
0	Step Mode
1	1
2	2
3	4
4	10
5	30
6	60
7	100
8	150
9	300 (full speed)

4. + X, + Y, + Z are the manual motor slew controls. These switches will move the indicated motor at the speed set on the rate code while they are held down. One step per push is selected by using rate=0. Any combination of switches may be used simultaneously.

POSITION displays are signed decimal readouts with a range of + 9999. Positions are displayed in steps from some previously defined origin for the chain selected. Actual range is ± 32767 but cannot be displayed in four digits so an over range character, a "square 8", is displayed in the most significant digit when this condition occurs. During the over flow the step position is not lost or changed. The display is blanked out for nonvalid chain or motor numbers. These displays are also affected by configuration mapping.

Caution: Applications using gimbals with full scale ranges (limit switch to limit switch) of greater than 32767 steps cannot make use of the position displays in an absolute sense, or of the position centering control functions below.

LIMITS are LED status lamps which light low or hi to indicate that a positioner has actuated a limit switch. Under normal conditions, no further motor motion is possible in the limit switch direction. Both LEDS on indicate that the motor cable is that particular motor is disconnected (if the display is not blanked). All six LEDS flashing is a warning that the motor interface for the selected chain is inoperable. Limit switches are mappable.

MOTOR POWER ON is a set of 30 yellow LEDS arranged in axis tripletts which indicate that power is enabled for a given motor when a lamp is lit. Signals for these LEDS are produced by the stepmotor logic board and are a hardware indication of motor activity.

NETWORK/LOCAL CONTROL selects mode of operation of the "SMC processor." Local control permits operation of all control panel functions while disabling remote commands whereas network control disables the command functions of the panel and allows remote control over the SHIVANET. Both modes are selectable simultaneously, permitting control from local and remote locations. The control switch must be actuated twice to go from one mode to the other. One push will enable both functions.

STATUS LAMPS indicate conditions in the SMC processor.

1. OPERATING will blink on and off if the SMC processor is running its program

2. NETWORK COMMAND will flash as a message is received from a remote processor over SHIVANET.

3. ERRORS DETECTED will flash when the remote control processor detects an error in the message transmission. It should not remain on.

4. MOTOR BUS ERROR will light and remain lit if the processor detects a transient or static failure of the stepmotor interface boards. This lamp is only turned off by resetting the computer. A power fail will not reset the lamp. Location in core memory can be examined to determine the address of the interface failure, the number of failures and other significant information by using a S700 terminal in ODT. To determine the locations to examine in core memory, refer to the software load map, "MCORE" section, "MOTORS" program. This is the relative address of the following information within that section.

WORD	
0	
1	Address of last bad interface board
2	Number of motors being considered by interrupt routine
3	Number of motors with power on
4	Slow scan flag (if=1, on one in 30 interrupts are used)
5	
6	Number of bus errors since last SMOTOR call
7	Number of bus errors since LSI-11 restart
8	

(On version generated October 18, 1977, this table starts at 1204)

5. EXTRA LEDS are not currently used but are reserved for future functions.

Position Centering Control

In the following descriptions functions may be performed for one chain by pushing switch down and for all chains by pushing switch up. An exception is the limit operation which affects only one

axis, X or Y or Z, for one selected and all axis, X or Y or Z, for ALL selected.

1. STOP cancels any operation previously entered and stops any motor motion. Stop status lamps indicate whether the selected axis has any motors running or if any motor in any chain is running. Both LEDS on indicate no motor motion in process.

2. CENTER switch can be used to drive motor positions to zero from the current position. Center status lamps indicate position=0 when lit. Center=0 can be interpreted as the aligned state of a positioner.

   The corresponding LED will blink as the operation is being performed until either the operation is complete, aborted with the stop switch or a limit switch is reached and all motors are stopped. Selecting LIMIT will also terminate this mode.

3. DECLARE switch is used to define a current position to zero. Switch must be pushed twice since its operation destroys information about a previously defined center.

4. LIMIT is an operation which drives an axis in a selected direction until the limit switch is reached. The axis and direction are derived from the last used slew control switch. Stable status LEDS indicate at limit conditions. A blinking LED warns that the operation is in progress. The function terminates upon completion, stop switch actuation, or center switch actuation.

Analog to Digital Converter ADC

The position readout can be switched from steps counted to the output of an ADC connected to a pot encoder mounted on the user's positioner. Direct readout is in counts (12 bits) or can be converted to steps per revolution (there are 200 steps per revolution on most step motors) by selecting a configuration switch. Correct motor motion can be verified using the ADC logic if the transducers and ADC modules are installed. If the ADC module is not installed for given axis, the display is blanked in that location. The software can detect the converters on the LSI-11 bus automatically. See the MPOTAD listing for more details.

Configuration Switches

As mentioned above, the KOM software can be easily reconfigured for many motor mapping variations of panel to actual hardware by setting appropriate DIP switches on the control panel interface board in the "SMC processor". The switch settings apply to all sets of motors within the SMC unit. For example, the axis direction + and - can be reversed on any axis, or the X, Y, and Z axis can be exchanged. These switches change the actual motor(s) being controlled, by a particular front panel command. Other control functions can also be altered with the switches. The bit function table is given on the following page.

2.4 Remote Control of SMC Processor

The SMC processor can be controlled by a remote PDP-11 or LSI-11 computer over the SHIVANET communications network. The link consists of two DLV-11 interfaces, one at each end of the link, used to pass information in serial format at 9600 baud. The physical link can either be an optically isolated current loop or fiber optic transmission line, the choice determined by user requirements. Addressing and jumper arrangements for the interfaces are given in the appendix.

The heart of the communications network is the SHIVANET software package. SHIVANET controls transmission based on special error detection and recovery protocal. Reliable message transfer is ensured by cyclic redundancy checking of message packets and a handshaking scheme between processors. The net is interrupt driven, requiring no user-active control for message transfer. The link software checks for many error conditions and will not hang the processor if they occur, but will inform the user program of the error. Since the software and its protocal are complex, the reader should refer to the SHIVANET documentation package for further information.

A special set of software calls has been written to allow easy remote control of the SMC processor by imitating local control panel like operations over the network. Any function available on the local panel can be duplicated from a remote location with these calls. A requirement for this type of operation is that the switches and LEDS be wired to the remote control panel interface card in the same order as is the local panel. Unused functions are not wired with the space reserved to maintain the order of the arrays.

More flexible general purpose commands are also available for remote operation for most any application. Full control of the SMC processor is possible using these commands in closed or open loop control systems. See Section IV for a complete description of system software.

TABLE 3

DIPSWITCH FUNCTION

DIP SWITCH BIT POSITION

DS0
DS1
DS2 — 3 bit code for mapping displays and slew buttons between axis

Code	Motor No. XYZ Axis
0	123
1	132
2	213
3	231
4	312
5	321
6	121
7	212

DS3
DS4 — 2 bit code for selecting valid chain numbers for units location

CODE	CHAIN MAP
00	South Side, Laser Bay (1-5, 16-20)
01	North Side, Laser Bay (6-15)
10	Upper Level, Target Area (11-20)
11	Lower Level, Target Area (1-10)

DS5=1 Reverse +X with -X, =0 leave as marked
DS6=1 Reverse +Y with -Y, =0 " " "
DS7=1 Reverse +Z with -Z, =0 " " "

DS8
DS9
DS10
DS11 — 4 bit code to set maximum number of "chains/2" to control (for example, "0101" is proper code for 10 chains)

DS12 0=ADC displays counts, 1=ADC displays scaled value 0-199 to encode shaft revolution to steps.
DS13 Option to control 2 motors per chain instead of the normal 3
DS14
DS15 — UNUSED - Must be left off, or serious software errors may result.

Note: These switches are read only once after the LSI-11 "computer clear" button is pushed. Subsequent changes must be followed by computer clear.

III. HARDWARE DESCRIPTIONS

3.1 General

This section provides detailed information on the hardware components developed for the SHIVA laser system. Included are functional descriptions and brief principles of operation sufficient for construction and use of the SMC processor (or other stepping motor control system based on some or all of the components).

Board schematics and other technical information follow each description reference as an aid in understanding the hardware. All necessary instructions for system configuration and adjustment are given. Particular system configuration parameters are specified in the appendix in tabular format.

3.2 SHIVA Standard LSI-11 Chassis LEA76-1345

The standard LSI-11 chassis was designed by the SHIVA engineering team as a modular, flexible framework for the implementation of large LSI-11 based control architectures. The chassis is a 12 1/4" rack mount unit containing the backplane and power supplies for the computer system. Either 4 or 5 backplanes may be installed, yielding 16 or 20 full width slots for Q bus compatible hardware.

The unit has mountings and wiring for one to three 5volt @ 25 amp power supplies complete a with current sharing network. A +12 volt supply is also standard. A -12 volt supply can be installed in space provided to supply the negative rail for PROM memories.

The chassis is equipped with a maintenance panel on the front that conceals the power switch, indicators and a reset button. Also included are connectors for teletype, floppy disc and paper tape. A meter monitors the +5 volt logic supply current.

The front panel of the chassis is also equipped with a removable door which is intended for user-designed control panels where required. A removable rear panel allows easy additions of connectors to the standard package. A large flat cableway has been included for ribbon cable commonly used in computer interconnections.

The backplane has been prebussed for all DEC standard bus signals. Special non-standard signals must be user-bussed at configuration time. In particular, the stepping motor system clock must be bussed on AE1 during installation. The power fail card described later also needs special bussing. Note that -12 volts is not prewired.

3.3 Control Panel Interface LEA77-1252-01

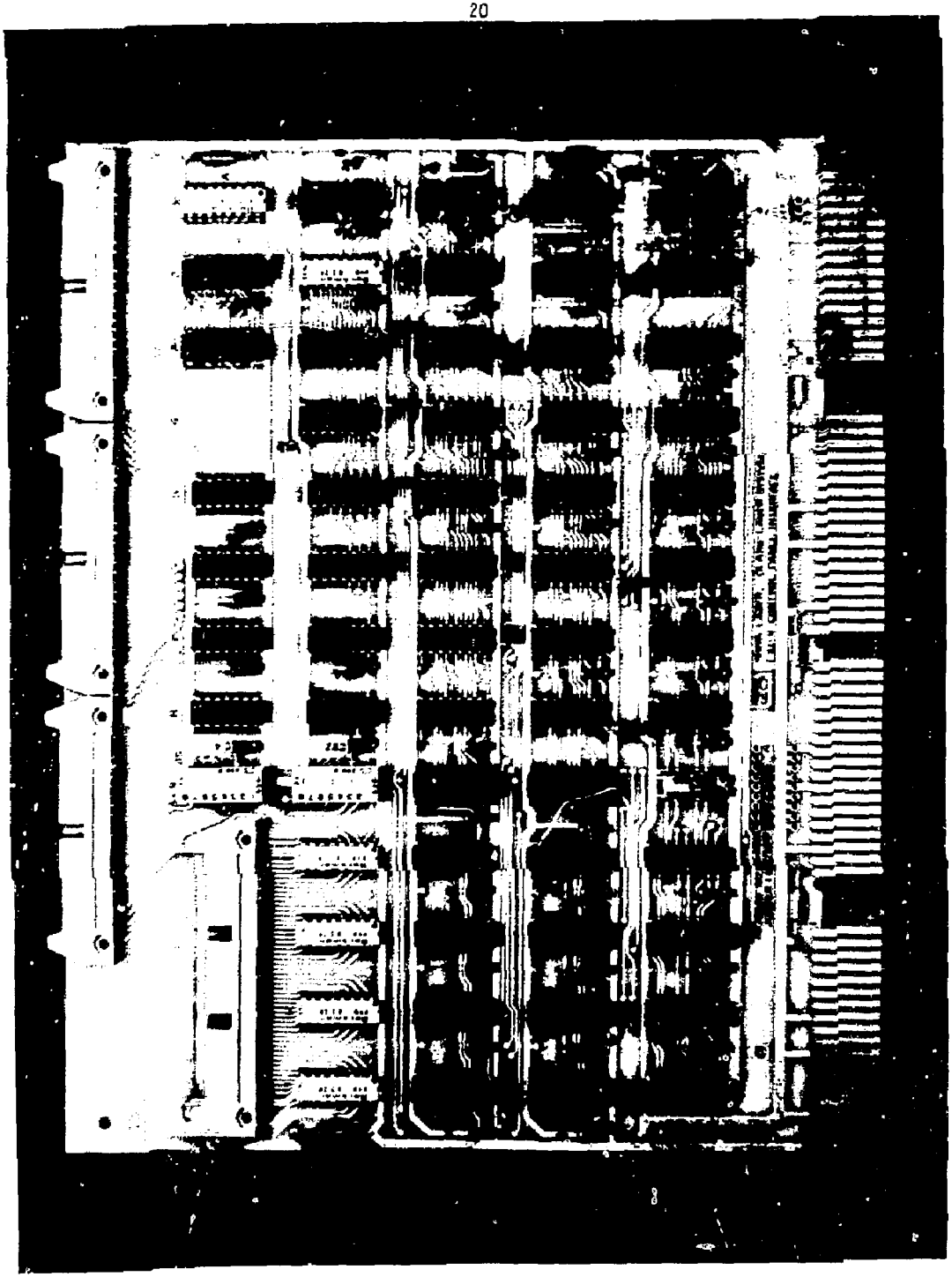
The control panel interface is a quad wide LSI-11 compatible card intended primarily for interfacing control panel functions with the computer. Design philosophy centered on maximum capability per card with a minimum of out-board components required. The interface is word addressable only and occupies eight words of address space located anywhere in the LSI-11 I/O page by selecting the desired address with jumpers on the card.

Register definitions are shown in Figure 3. At the most basic level of operation, the interface can be used as four parallel I/O ports, shown as IN1-4 and OUT1-4. Input words are identical to each other; each line is pulled up through a 2.2K resistor and is negative logic TTL. Any unused input bit positions are read as zero.

Programming Note: DPDT switches can be used on the control panel; one switch forms the function, the other is wired to other switches in a functional group. The group switch can be quickly read to determine if any switch in the group has been pushed. The application software can then determine which switch function in the group was actually selected. This technique can greatly improve the speed of panel service routines.

A second level multiplexing scheme is used to allow up to ten 16-bit digiswitch banks to be read by the interface. Implementation of this function requires one of the input words to be dedicated for this purpose. Current software packages reserve IN1 as the digiswitch input port. Multiplexing of digiswitches requires two processor operations. The first operation is to write the bank number (0-9) into the first register location (digi-select). The contents of this 4-bit register select one of the digiswitch words (if valid) to be present on the input word IN1. IN1 is then read to get the value of that particular Bank word. Figure 4 shows an easy way to implement the bank select scheme outlined above. Germanium diodes mounted on digiswitches made for the purpose allow only the selected bank to pull down the input lines. All unselected banks have their diodes reverse biased.

Circuitry has also been included to expand the interface capability to allow it to drive up to eight seven segment display words with sign. Positive true bus data is buffered to 10 TTL loads and is available on the I/O connectors Figure 5 shows how three display words are wired as in the SMC processor local control panel. Hewlett Packard 5082-7300 series display chips are used because they include on-chip latches, decoders and drivers for the display function. The display output will follow the data lines while the chip strobe line is held low. Writing to the displays takes three operations. As for the digiswitch bank select, the four bit digi-select register is loaded with the proper word number 0-7. Writing the 15th bit on the second output word will now be stored as the sign of the display word by a gated latch; each sign is



LSI-11 CONTROL PANEL INTERFACE

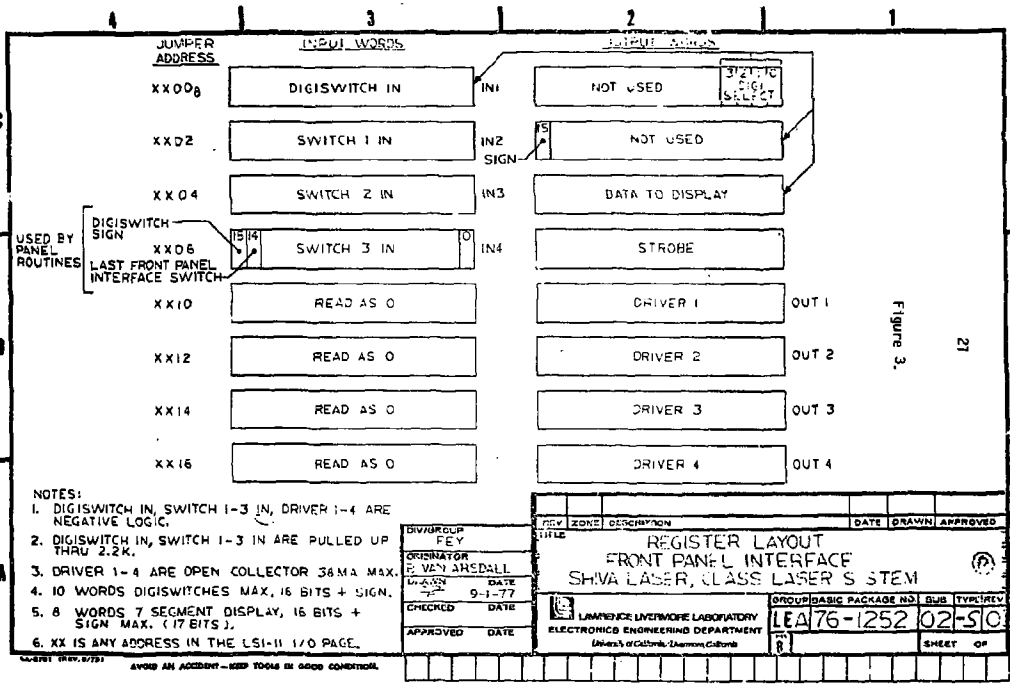
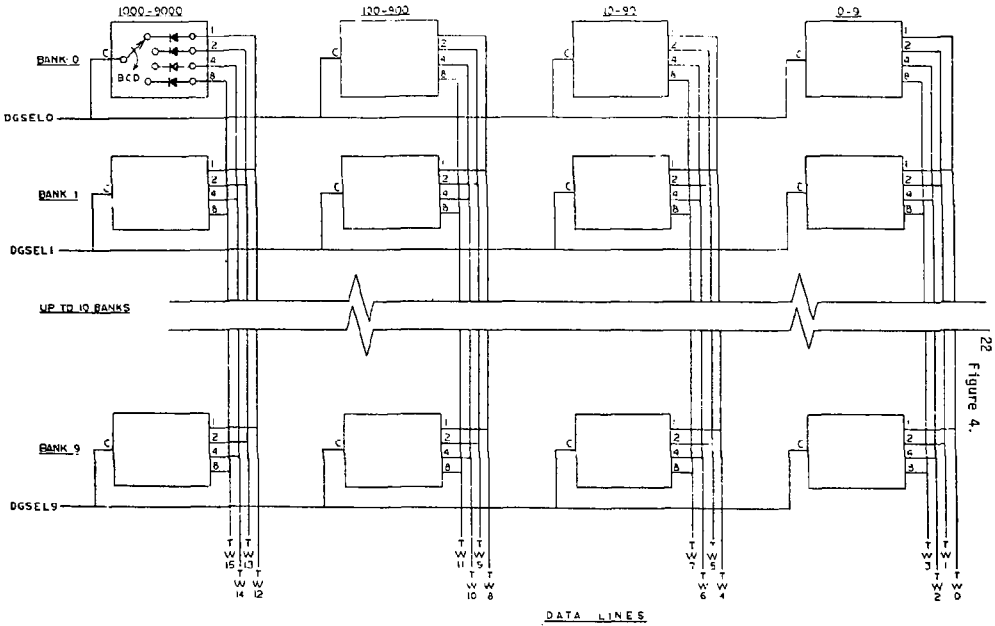

JUMPER ADDRESS
INPUT WORDS
OUTPUT WORDS
XX00₈
DIGISWITCH IN
IN1
NOT USED
SELECTS DIGISWITCH
XX02
SWITCH 1 IN
IN2
SIGN
NOT USED
XX04
SWITCH 2 IN
IN3
DATA TO DISPLAY
USED BY PANEL ROUTINES
DIGISWITCH SIGN
XX06
15
14
SWITCH 3 IN
0
IN4
STROBE
LAST FRONT PANEL INTERFACE SWITCH
XX10
READ AS 0
DRIVER 1
OUT 1
XX12
READ AS 0
DRIVER 2
OUT 2
XX14
READ AS 0
DRIVER 3
OUT 3
XX16
READ AS 0
DRIVER 4
OUT 4
NOTES:
1. DIGISWITCH IN, SWITCH 1-3 IN, DRIVER 1-4 ARE NEGATIVE LOGIC.
2. DIGISWITCH IN, SWITCH 1-3 IN ARE PULLED UP THRU 2.2K.
3. DRIVER 1-4 ARE OPEN COLLECTOR 36MA MAX.
4. 10 WORDS DIGISWITCHES MAX, 16 BITS + SIGN.
5. 8 WORDS 7 SEGMENT DISPLAY, 16 BITS + SIGN MAX. (17 BITS).
6. XX IS ANY ADDRESS IN THE LSI-11 I/O PAGE.
AVOID AN ACCIDENT – KEEP TOOLS IN GOOD CONDITION.
DIV/GROUP FEY
ORIGINATOR R. VAN ARSDALL
DRAWN 9-1-77
CHECKED
APPROVED
REV ZONE DESCRIPTION DATE DRAWN APPROVED
TITLE
REGISTER LAYOUT
FRONT PANEL INTERFACE
SHIVA LASER, GLASS LASER SYSTEM
LAWRENCE LIVERMORE LABORATORY
ELECTRONICS ENGINEERING DEPARTMENT
University of California, Livermore, California
GROUP BASIC PACKAGE NO. SUB TYP REV
LEA 76-1252 02 S 0
SIZE B
SHEET OF

Figure 3.



NOTES:
1. DIODES ARE IN270.
2. DIGISWITCHES ARE MODEL 721.

LSI-11 CONTROL PANEL INTERFACE
DIGISWITCH MULTIPLEXER

Figure 4.

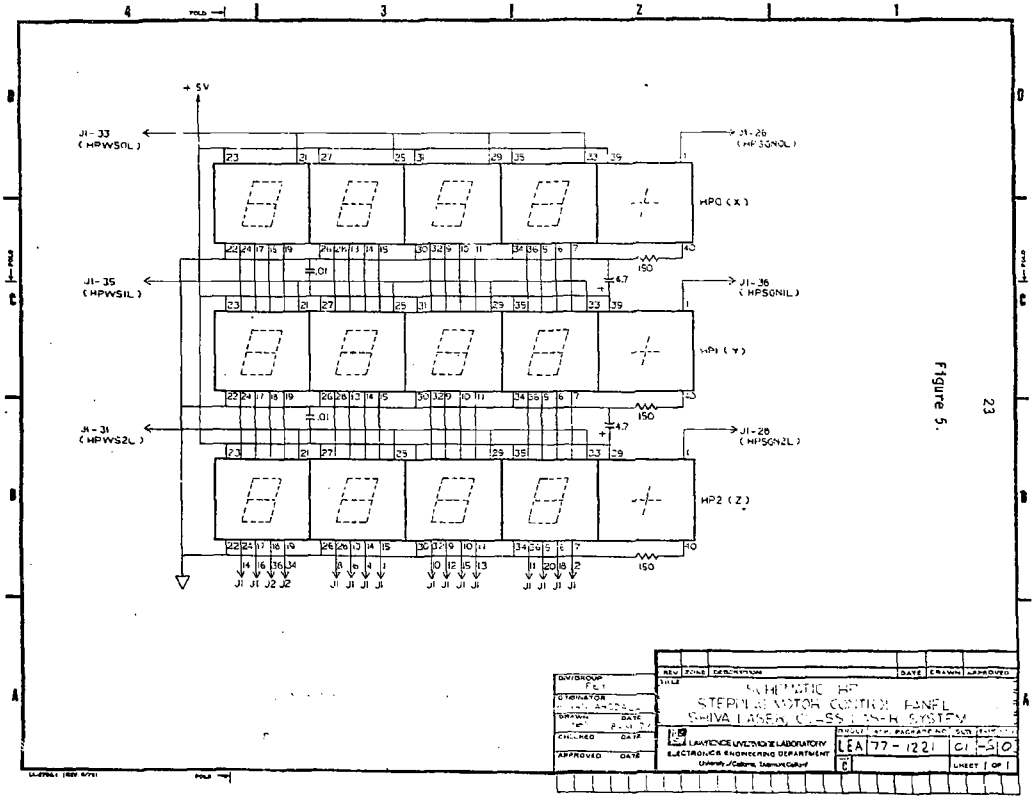

+5V
J1-33
(HPWSOL)
J1-28
(HPSONL)
HP0 (X)
J1-35
(HPWSIL)
J1-36
(HPSONIL)
HP1 (Y)
J1-31
(HPWS2L)
J1-26
(HPSON2L)
HP2 (Z)
.01
4.7
150
J1 J1 J2 J2
J1 J1 J1 J1
J1 J1 J1 J1
J1 J1 J1 J1
REV ZONE DESCRIPTION DATE DRAWN APPROVED
TITLE
SCHEMATIC HP
STEPPER MOTOR CONTROL PANEL
SHIVA LASER GLASS MAIN SYSTEM
LAWRENCE LIVERMORE LABORATORY
ELECTRONICS ENGINEERING DEPARTMENT
University of California, Livermore/Berkeley
DRAWING NO. PACKAGE NO. SIZE
LEA 77-1221 C1-510
SHEET 1 OF 1
DIV/GROUP
ORIGINATOR
DRAWN DATE
CHECKED DATE
APPROVED DATE

Figure 5.

individually addressable in this way. Similarly the data is loaded into the display chip register set by writing the appropriate data to the third output word. As a note, none of the output words may be read by the computer for verification so it is wise to perform the complete sequence each time.

Writing anything to the third output work generates a negative going (200 ns min) strobe capable of sinking up to 70 mA. This signal is brought to a jumper point at the bottom right hand corner of the PC card. The dashed lines indicate possible jumper connections to bus fingers AE1, AF1 and AH1 which are the first three user bus connections available on the backplane. Possible applications include software clock distribution and event strobing.

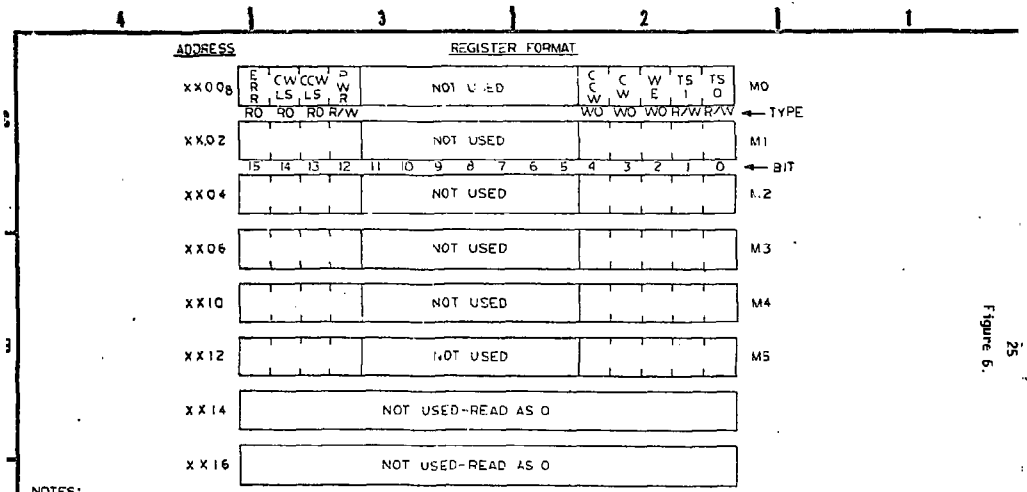
Addressing of the board is accomplished by installing appropriate jumpers A4-A12 in the address selection section at the bottom of the PC card. Note that: jumper installed = 0. Base address for system software is 160600 and increments in eight word (16 byte) slices per board. This jumper arrangement results in A4, A5, A6, A9, A10, A11, A12 jumpers installed. An adjustable interface response is provided using a simple delay circuit to widen the write strobe if necessary. This can be used to allow data to settle for the displays if long cables are used to connect to the panel. The delay adjustment is made by selecting an appropriate value for C2. R2 should be kept below 200 ohms.

3.4 Stepmotor Logic Card LEA76-1344-31B

The Stepmotor logic card is a quad-wide DEC LSI-11 compatible interface designed to provide stepping translator and power logic for six four-phase stepmotors. The interface synchronizes and logically executes step commands from a computer program. Also included are provisions for two limit switches per motor. The ability to access the translator state is an important feature that enables the preservation of the original motor position in the event of power failure.

Explanation of operation is best begun by referring to the register layout and definition in Fig. 6 and Table 4. Each of the six motor logic sections is identical and has its own unique status/control register and corresponding address in the LSI-11 I/O page. The two null words are necessary to simplify the logic design. Register bits are explained in the bit function table.

The stepmotor interface is driven by an external clock source whose frequency matches the fastest speed required of the motor in steps/sec. This clock signal actuates any step commands issued to the interface by the computer. A good internal picture of the logic is shown on the translator control block diagram, Fig. 7. The interface to the computer is not explicitly shown but uses standard techniques. The reader is referred to the DEC Micro Computer User's Manual for a discussion of the LSI-11 bus and interface timing. The diagram is organized with input bits above and output bits below.



NOTES:
1. THESE REGISTERS ARE BYTE ADDRESSABLE.
2. XX IS ANY ADDRESS IN THE LSI-11 I/O PAGE.
3. XX00 - XX12 ARE DUPLICATES.
4. FOR FUNCTION EXPLANATION SEE BIT FUNCTION TABLE.

Figure 6.

REV	ZONE	DESCRIPTION	DATE	DRAWN	APPROVED
B		REVISED	9-77	DF	

DIV/GROUP: FE
COORDINATOR: R VAN ARSDALL
DRAWN: DATE 9-1-77
CHECKED: DATE
APPROVED: DATE

TITLE: STEPMOTOR LOGIC BD REGISTER LAYOUT
STEPPING MOTOR CONTROL
SHIVA LASER, GLASS LASER SYSTEM

LAWRENCE LIVERMORE LABORATORY
ELECTRONICS ENGINEERING DEPARTMENT

SIZE	DRAWING NO.	REV
E	7E-1344	32-SIB

SHEET 1 OF 3

AVOID AN ACCIDENT—KEEP TOOLS IN GOOD CONDITION.

TABLE 4

BIT FUNCTION TABLE FOR STEP MOTOR LOGIC CARD

BIT	TYPE	FUNCTION
0	R/W	BIT 0 OF THE 4 STATE TRANSLATOR
1	R/W	BIT 1 OF THE 4 STATE TRANSLATOR
2	WO	THE 2 BITS OF THE 4 STATE TRANSLATOR CAN ONLY BE WRITTEN IF THIS BIT IS A ONE.
3	WO	WRITING A ONE TO THIS LOCATION CAUSES THE TRANSLATOR TO ADVANCE CW AT THE NEXT INTERRUPT CLOCK PULSE.
4	WO	WRITING A ONE TO THIS LOCATION CAUSES THE TRANSLATOR TO ADVANCE CCW AT THE NEXT INTERRUPT CLOCK PULSE.
12	R/W	WRITING A ONE TO THIS LOCATION WILL ENABLE THE POWER CIRCUITS IN THE DRIVE AMPLIFIER. CLEARING THIS BIT DISABLES THEM.
13	RO	CLOCKWISE LIMIT SWITCH. A ONE READ HERE INDICATES THAT THE MOTOR IS AT CW LIMIT.
14	RO	COUNTERCLOCKWISE LIMIT SWITCH. A ONE READ HERE INDICATES THAT THE MOTOR IS AT CCW LIMITS.
15	RO	ERROR. THIS BIT IS EQUAL TO THE LOGIC EXPRESSION: ERR= $\overline{PWR}$+CWLS+CCWLS. A ONE INDICATES THAT IT MAY NOT BE POSSIBLE TO MOVE THE STEP MOTOR.

A typical step sequence begins by turning on drive power (bit 12=1) to the motor. Status lines (bits 12-15) are then interrogated by the software driver to determine if motor motion is possible. A step command is issued to the interface by writing a one to either bit 3 for CW rotation or bit 4 for CCW rotation. Writing ones to both bits simultaneously is an error. The step command is latched asynchronously (wrt stepmotor clock) by the buffer. The next stepmotor clock pulse will cause the four state translator to advance in the correct direction. The trailing edge of the clock clears the buffers, making them ready to respond to new commands. The step logic actually delays part of a clock period before executing steps in order to synchronize the indeterminate timing of the step command from the driver program.

The translator is implemented with 2 bits of a four bit up/down counter. CW rotation results in the counter advancing up one count. The state decode logic decodes the two bit counter output into four states (see Table 2). Outputs to I/O connector PG1 are open collector drivers directly gated by the motor drive enable line. These outputs can sink 38 mA and are intended for driving opto-isolators on the stepmotor drive amp card. Power-off causes these outputs to go high and consequently removes drive from the switching transistors on the drive amp card.

System software (see MOTORS listing) is interrupt driven by the same external clock through a DEC DRV-11's interrupt controller. The step command timing diagram, Figure 8, illustrates the interaction between the clock, software and stepmotor interface. Each time a clock pulse is generated an interrupt to the LSI-11 is issued on the leading edge and causes the motor control software to run. The program then examines its command tables and decides on a per motor basis if any action need be done, performing any operation required. At most, only one step command per motor is given per interrupt.

On receiving a command to move some distance, MOTORS will turn on the power to the selected motor, wait several clock periods for the motor to stabilize to its detent position and begin to issue step commands to the step motor interface while decrementing its distance-to-go table. The software always checks for error conditions that may prevent proper motor motion. The software can generate many step speeds by dividing the stepmotor clock with a rate code corresponding to the correct speed. As the distance-to-go falls to zero, MOTORS will inhibit stepping, leave the motor power on long enough to freeze rotor inertia, then turn it off.

Proper timing depends on interrupt software finishing its task each clock cycle before the next clock pulse is generated. Maximum motor speed is therefore also dependent on software execution time. For this reason, a separate SMC processor is used to control any significant numbers of motors up to thirty. Small numbers of motors may be operated directly from a high level processor with sufficient time left over to run the user's control program.

TABLE 5

TRANSLATOR DECODE

Translator State		Motor Phase Outputs				
TS1	TS0	SW1	SW2	SW3	SW4	
0	0	0	1	0	1	0=phase on
0	1	0	1	1	0	
1	0	1	0	1	0	
1	1	1	0	0	1	

Each two bit translator may be read and conditionally initialized to any value by the computer. Since TTL circuitry is used in the interface, power failure will cause the translator contents to be lost. Coming up in a random state will usually cause the loss of several steps in one or the other direction and is unpredictable. During a power down sequence, the actual value is read from each translator and stored in a table in core memory by the power fail program. Because core memory is non-volatile, the power-up routine can access the old values and restore the translators to the states they held before powerdown. This ability is not known to be available on any commercially produced stepmotor drive system.

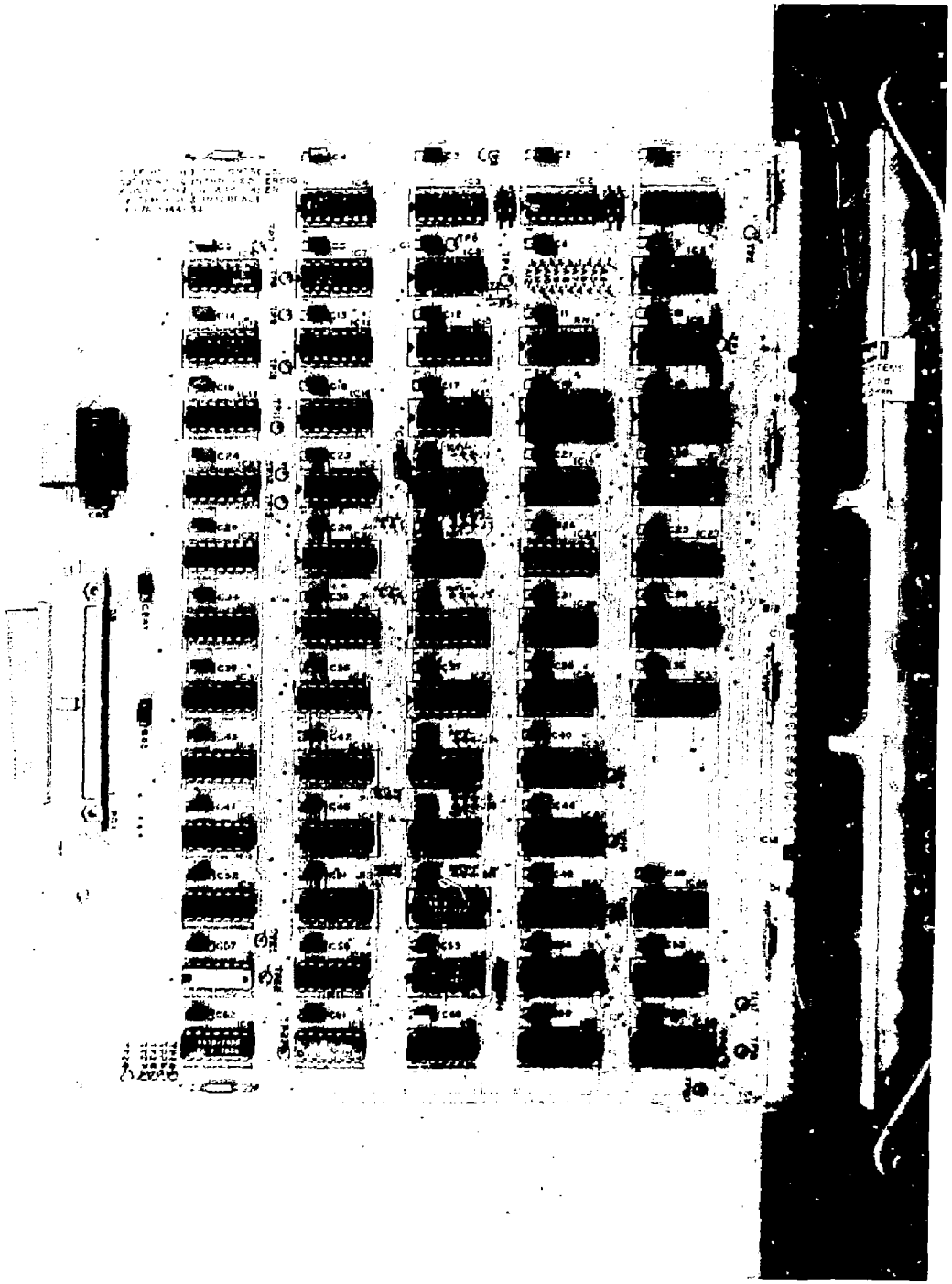
The on-board multiplexer acts to gate the selected motor status/control register onto the LSI-11 bus as data. Circuitry has been included to enable the multiplexer to function as a status panel driver for any number of motor boards installed in a given system. Since, most of the time, the computer is not accessing the interface, the multiplexer receives its selection address from a set of digiswitches located on an optional maintenance panel instead. One switch selects a register number 0-5 and another selects a particular board. Status information is gated through the multiplexer and out open collector drivers which can be wired together from other boards to drive LED lamps in a display. Power signals are not handled in this way, but are available continuously through dedicated open collector drivers. All signals described above are available through I/O connector PG2. This feature can be of considerable advantage in maintenance procedures and can also be used to drive part of the system control panel. The drive power LEDS on the SMC processor local control panel are driven from this source.

Addressing of the card is accomplished by installing appropriate jumpers A4-A12 on the address selection wire wrap posts near the bottom right hand corner. Note: Jumper installed = 1. Base address for system software is 160100 and increments in eight word slices per board. This address results in jumper A6 installed. Note that this interface is byte addressable for programming convenience.

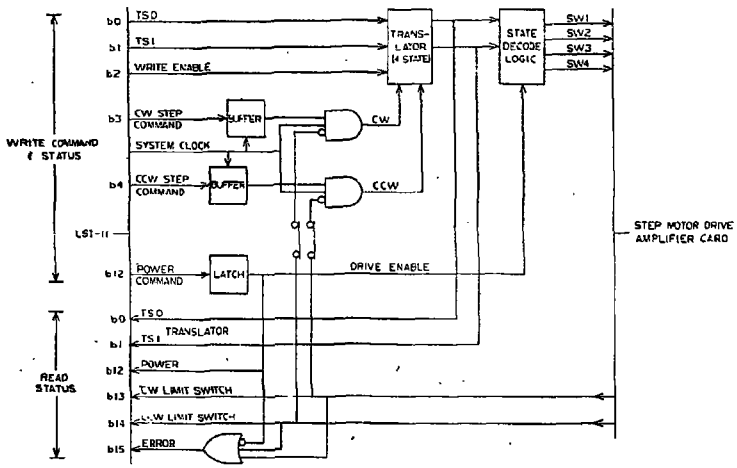
Maintenability is enhanced by the inclusion of test points, by the status panel driver and by a comprehensive software exerciser. Test points are called out on the schematics at the most important signal nodes. The software exerciser requires a special interface (to connect to PG1 in place of the two stepmotor drive amp cards) which reads the output signals for verification and also cycles the limit switches to check proper operation of their circuits. Many tests are performed and a comprehensive error log generated to aid the location of faults.

3.5 Powerfail Detect Card, LEA77-1258

The powerfail detect card is a dual wide LSI-11 compatable card whose primary function is to supply the two power status signals (BDCOK and BPOK)*with their associated timing needed to conduct a recoverable power failure as well as an intelligent start-up upon application of power. As well as the basic function, the powerfail detect card provides a number of features and options that significantly add to the usefulness of this card.



STEPPING MOTOR LOGIC CARD



NOTE: THIS LOGIC TYPICAL FOR 6 MOTOR CONTROLLERS PER LSI-11 QUAD MODULE.

Figure 7.

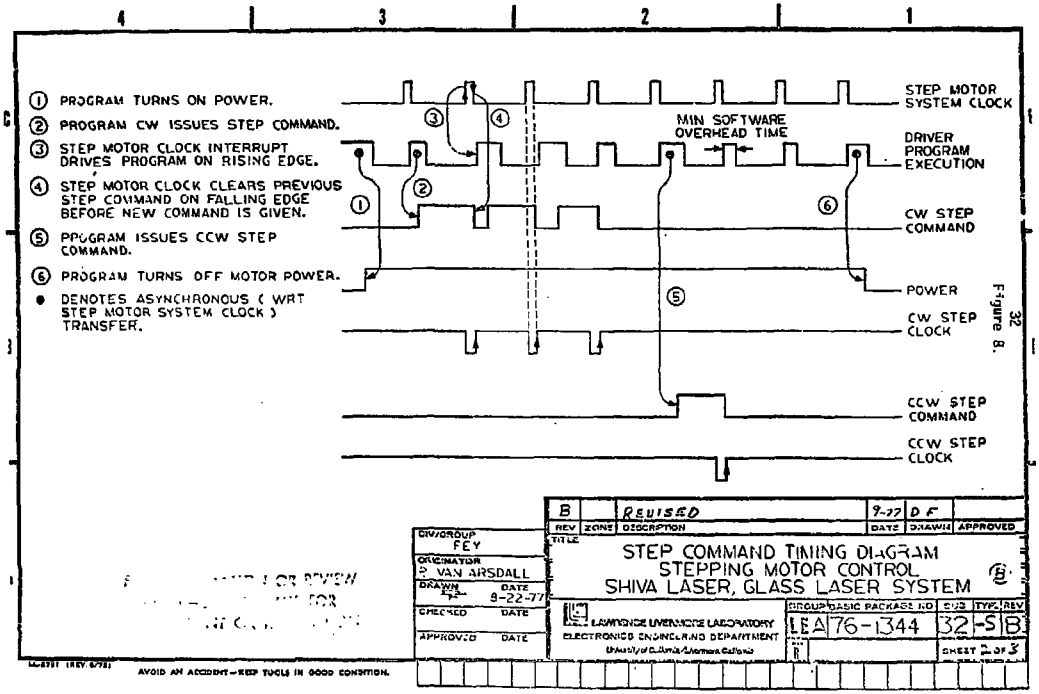


Figure 8. 32

REV	ZONE	DESCRIPTION	DATE	DRAWN	APPROVED
B		REVISED	9-77	D F	

DIV/GROUP FEY
ORIGINATOR P. VAN ARSDALL
DRAWN DATE 9-22-77
CHECKED DATE
APPROVED DATE

TITLE
STEP COMMAND TIMING DIAGRAM
STEPPING MOTOR CONTROL
SHIVA LASER, GLASS LASER SYSTEM

LAWRENCE LIVERMORE LABORATORY
ELECTRONICS ENGINEERING DEPARTMENT
University of California/Livermore, California

GROUP	BASIC PACKAGE NO	SUB	TYP	REV
LEA	76-1344	32	-S	B

SHEET 2 OF 3

LL-8781 (REV. 6/78) AVOID AN ACCIDENT—KEEP TOOLS IN GOOD CONDITION.

On board of the powerfail detect card are the necessary buffers and dividers to provide one of three jumper selectable line-based clocks that is distributed on the bus as BEVENTL.* The frequencies available are 10HZ, 60HZ and 120HZ. The line time clock is controllable from two sources; one hardware and one software. The hardware control is via a toggle switch near the top of the powerfail card. With the switch in the "OFF" position the clock is turned off and software control of clock disabled; with switch in "ON" position the clock is turned on and software control is enabled. With the switch in the ON position it is possible to control the ON/OFF status of the clock via a flat ribbon jumper cable to the output connector (J1) of a DRV-11. In this configuration when power is applied the line time clock is off due to the output word of the DRV-11 being cleared. With the loading of data Bit 00 of the output word with a logic "1" the clock can be turned on.

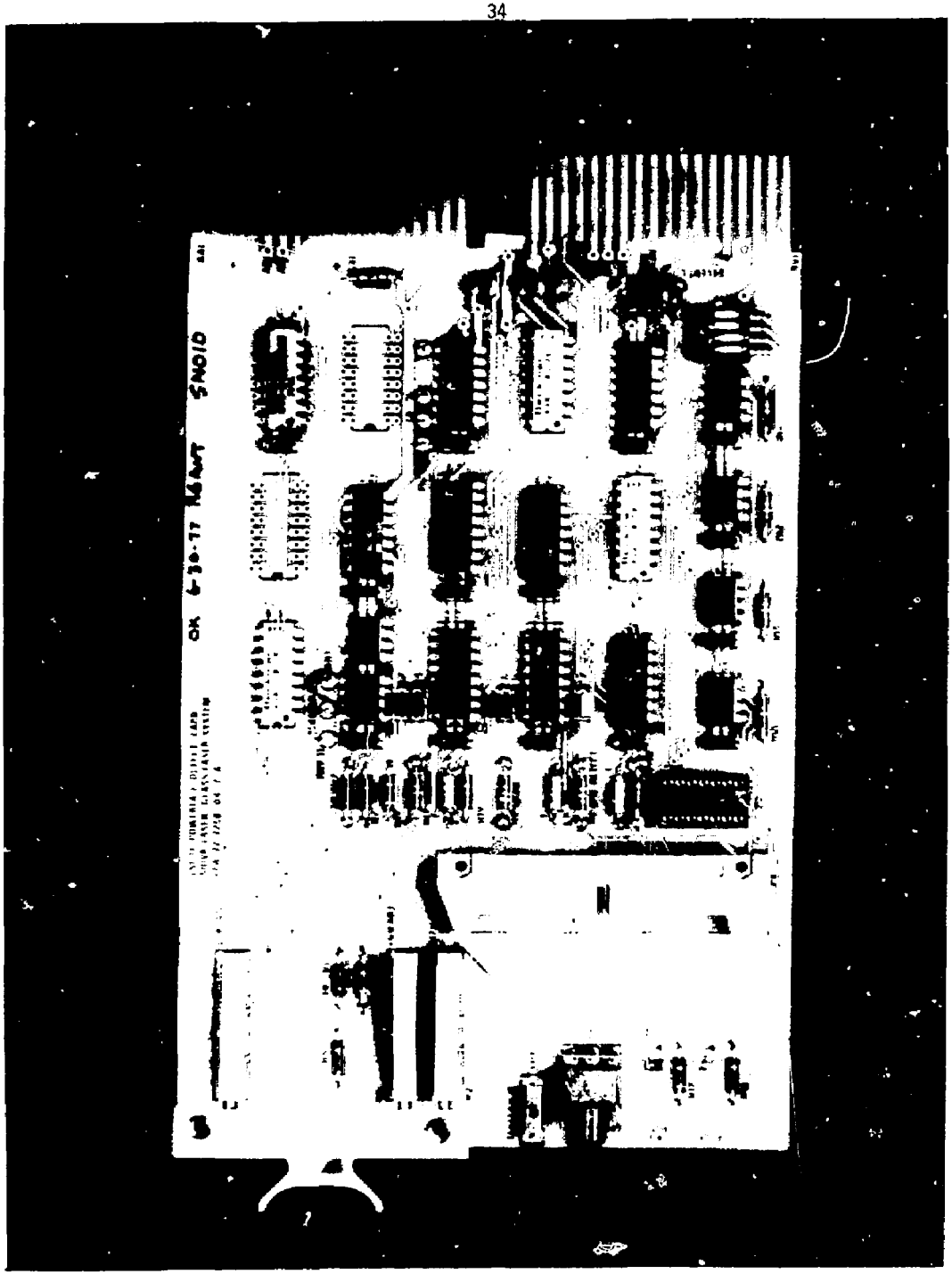
The powerfail card has two indicators that greatly simplify both the initial adjustments of the card itself and detection of power supply failure. The "CPU RUN" LED (green) has the necessary logic to use the SRUNL from the backplane to drive both the light on board as well as an open collector output for driving a front panel LED cartridge to indicate that the CPU is active or "running." The "PSOK" LED (yellow) indicates the sum of all enabled powersupply monitor comparators by remaining on if any or all of the selected power supplies are below their adjusted monitor levels.

There are three adjustments that require at least a rough calibration for proper operation of the powerfail card, they are: 1) clock level, 2) +12v monitor level, 3) +5v monitor level. The adjustment of the clock level determines at what line voltage the powerfail card will use to initiate a power fail sequence due to insufficient line voltage to properly operate the processor. The adjustment of the +12v and +5v monitor levels similarly determines at what level of +12v or at what level of +5v the card will initiate a powerfail sequence due to DC voltages insufficient to maintain the proper operation of the processor.

The power supply monitor select switch is a four position dip switch that is used to select the composite signals that generate the PSOK signal. Although at least one switch must be one, the user has the option of how many and which power supplies need to be monitored to determine sufficient operating voltages. For instance the switch labeled PS1 enables the +12v comparator and the switches labeled PS1 enables the +12v comparator and the switches labeled PS2, PS3, PS4 controls which +5v supply comparators are enabled.

A feature that is a little more unique to this particular design of powerfail card is the "TESTFAIL" signal. The "TESTFAIL" signal is used for generating a logic powerfail with power still applied (generally used for test purposes). There are three sources for this

*LSI-11 signal names are explained in depth in the DEC 1976-77 Microcomputer Handbook.



LSI-11 POWERFAIL DETECT CARD

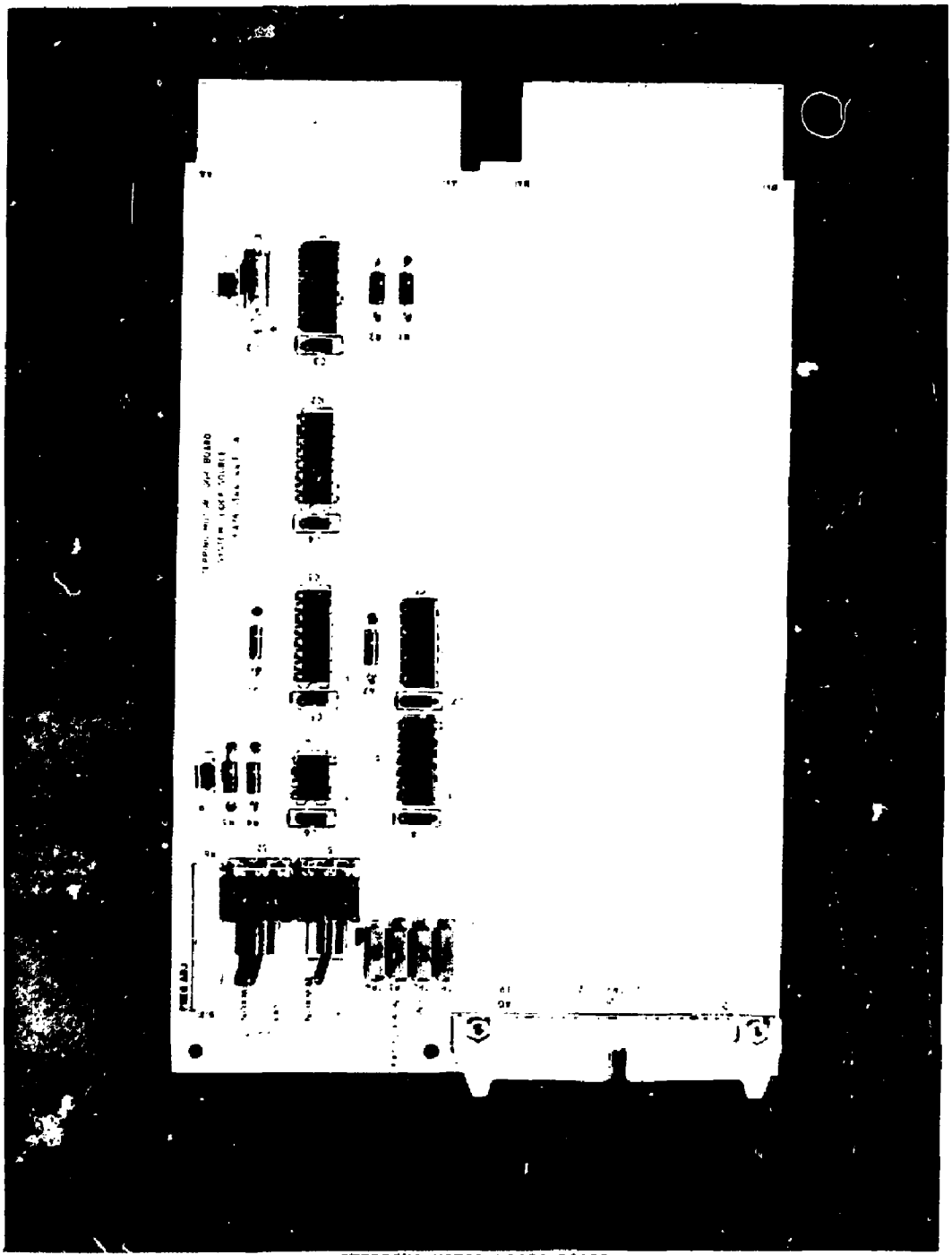
signal. The one that is the most accessable for a manual test is via the tip style test point jack (grey) at the top of the card. With the application of +5v or a logic "1" the card will generate a powerfail sequence and stay down until the signal either becomes a logic "0" or removed entirely which will then allow a power-up sequence. One of the two other sources may be selected by a jumper on the powerfail card. The options are to have either a DRV-11 generate the signal or have the signal taken from a spare bus finger "BC1" off the backplane. When jumpered to use the DRV-11 the signal is generated by changing Bit 01, of the output word of the same DRV-11 that is used for software control of the line time clock, to a logic "1." In this configuration a power-up sequence will be generated seemingly immediately due to the fact that the powerfail sequence ends with an initialization of the computer which resets the DRV-11 output word which in turn allows the TESTFAIL H signal to go to a logic "0." The third source of the "TESTFAIL" signal is probably the most useful as far as normal operation is concerned. With the jumper selecting the backplane bus finger as signal source the powerfail sequence can be generated by the user asserting a logic "0" on pin BC1. This can be done either by user's logic or by using the framing error signal of a DLV-11. This last configuration is the most useful because of the added ability to trigger the computer for downline loading ("A" processor level) from a master computer or for remote restarts (SMC processor level) from a higher level processor either of which is extremely useful for multi-level networks and remote recoverability.

For added versatility in usage, the timing resistors that control the timing relationships of the DCOK and POK signals, are mounted on forked terminals for calibration or special purpose. Also available, for user modification or addition are two 16pin IC positions and 14 remaining 220Ω/330Ω bus terminations which are not used by standard powerfail detect card configuration.

The operation of the powerfail card's power status signals (BPOK and BDCOK) is based around the monitoring of the AC line voltage, the +12 volt power supply and up to three +5 volt power supplies. The +5 volt supplies are arranged by way of diode isolation and calibrated output lead lengths, to share the load current in normal operation and continue operation during the powerfail sequence if one supply fails. The +12 volt supply similarly has diode isolation from two storage capacitors that provide additional operating time for the powerfail sequence. Although the software requirement for powerfail is <2MS, the unmodified powerfail card has between 5ms and 20ms with the option of extending it to greater than 40ms without loss of operating voltages. (See DWG #LEA77-1258 02-T 0 timing diagram.)

3.6 Stepmotor Logic System Clock Source LEA76-1344-44A

This half-quad card functions to provide a common clock signal to all stepmotor logic cards on the LSI-11 bus and also provides an interrupt signal to synchronize the software control routine.



STEPPING MOTOR LOGIC BOARD
SYSTEM CLOCK SOURCE

The system clock is open collector (38mA max sink) negative logic driving bus finger AE1, the first available user option bus finger position. This bus position must be bussed on the backplane by the user. It is recommended that only the right side of the backplane be bussed and the clock board be mounted in this half to minimize the clock bus length.

Interrupt requests are strobed by the clock and latched in a flipflop. The request is carried over 40 conductor flat cable from J1 on the clock board to J2, the input port, on a DEC DRV-11 which is used for the interrupt logic. (REQ B). The interrupt request flip flop is reset under program control by reading the input word on the DRV-11 or by a bus init signal. Latching the request in this manner prevents the KD11F processor from hanging up during requests where the request line drops before the interrupt vector is placed on the bus.

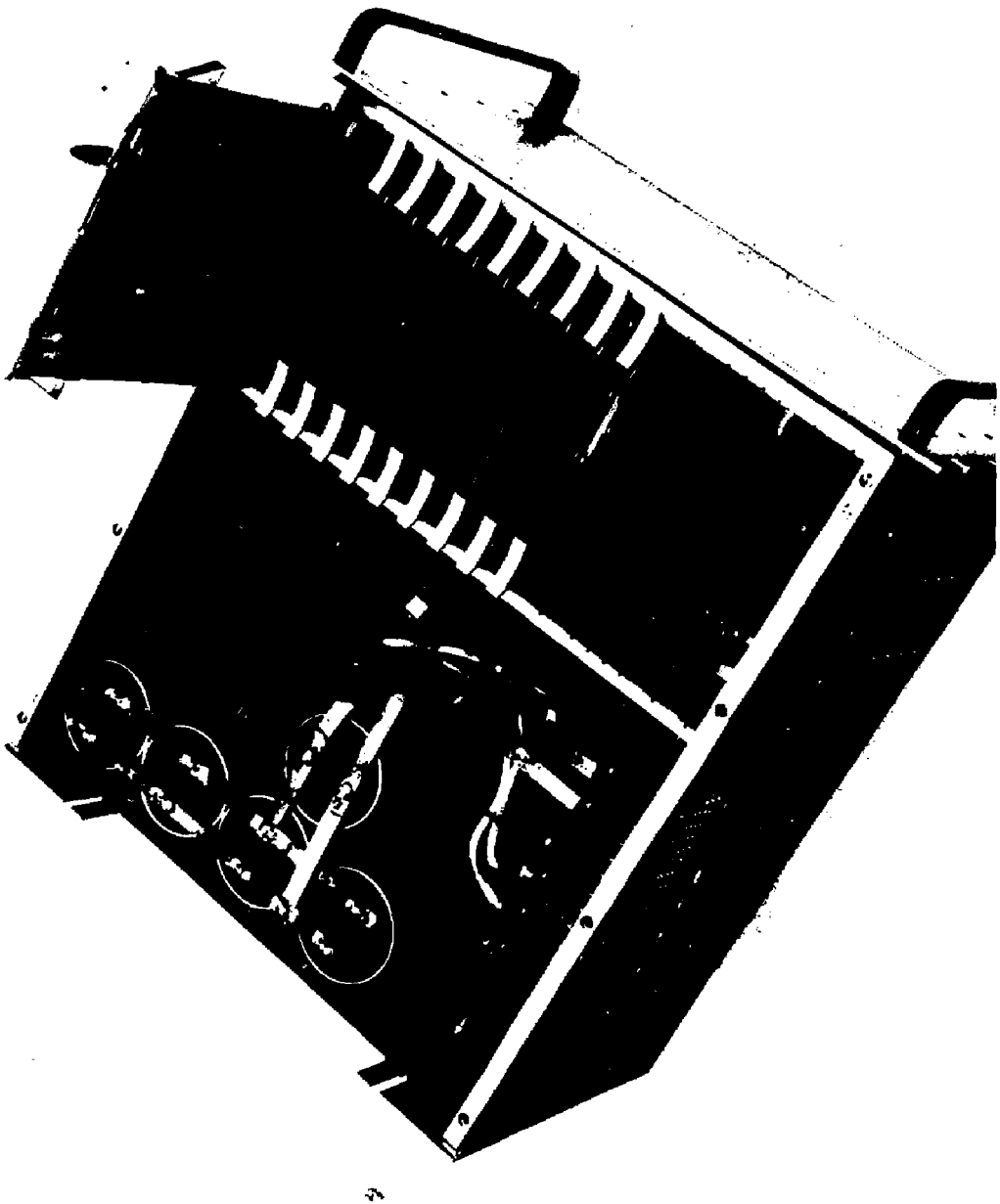
Two switchable options are included to add more flexibility to the clock. The clock switch selects either the on-board oscillator or an external source through TP2-TP3. The external oscillator is divided by 256 before driving the clock bus. A second switch deselects the board-clock source and divider and instead selects DRV-11 status R/W bit CSR0 as input to drive the bus. The position is marked TEST on the left hand switch. (Clock switch should be in EXT for proper gating of the test signal). This feature can be used to drive the stepmotor clock directly from software by toggling the CSR0 bit. This technique has been used with special interface hardware and software to exercise and diagnose stepmotor logic cards. Some simple motion schemes could also be implemented for small systems in this way (versus interrupt driven software).

Standard clock frequency for most applications is 300HZ. This can be measured at TP1 as a 3.3 ms negative going pulse. A useful maintenance point is TP4 which is the output of the clock oscillator running at 76.8KHZ (13μs). Frequency trimming is made by adjusting R6 with both option switches in the normal position.

3.7 Stepmotor Drive Amp Chassis LEA76-1344-01

This is a 10.5" high rack mount unit designed to accommodate up to 30 stepmotors in its standard configuration. The unit has a complete card guide assembly with ten fully wired Augat sockets for stepmotor drive amp cards LEA76-1344-21A. Space is available in the cage for two more Augat sockets for user defined cards. Holes have been prepunched in the rear panel to allow easy expansion of this space for six extra motor circuits.

Rear panel 41 pin female connectors (MILC26482) are provided for each of the 10 Augat card motor locations to supply and pickup gimbal control signals over the standard step motor control cable (see LEA76 1332-01 for a description of this cable and the signals it carries). Six spare wires are available in each cable for user defined functions.



STEPPING MOTOR DRIVE AMP CHASSIS
(Photo of Prototype)

The chassis includes a 24V (tap adjustable to 28vor 20v) 40 amp dc power supply for the circuit cards and stepmotors. A plate under the supply has wiring and mounting space for 60 20-watt current limiting resistors (2 each motor). The particular value must be selected for the actual motors to be driven. For .44 amp/phase motors the resistor is 25Ω, for 1 amp/phase motors the value is 20Ω.

Three Augat sockets are also included and wired for pot filter cards LEA76-1344-61A which pickup pot signals in groups by axis from the rear panel connectors. For potentiometer power, a 10v regulated supply may be mounted near the pot filter connectors on the side wall. An Acopian model 10EB24 fits without drilling any holes and will supply 240 mA, enough to drive 30 2K or larger pots. Bipolar operation can be implemented by adding a second 10EB24 and by making appropriate connections

For proper cooling of the electronics a solid top and perforated bottom cover are recommended to help guide air through the card cage and over the resistor deck.

3.8 Stepmotor Drive Amp Card LEA76 134421A

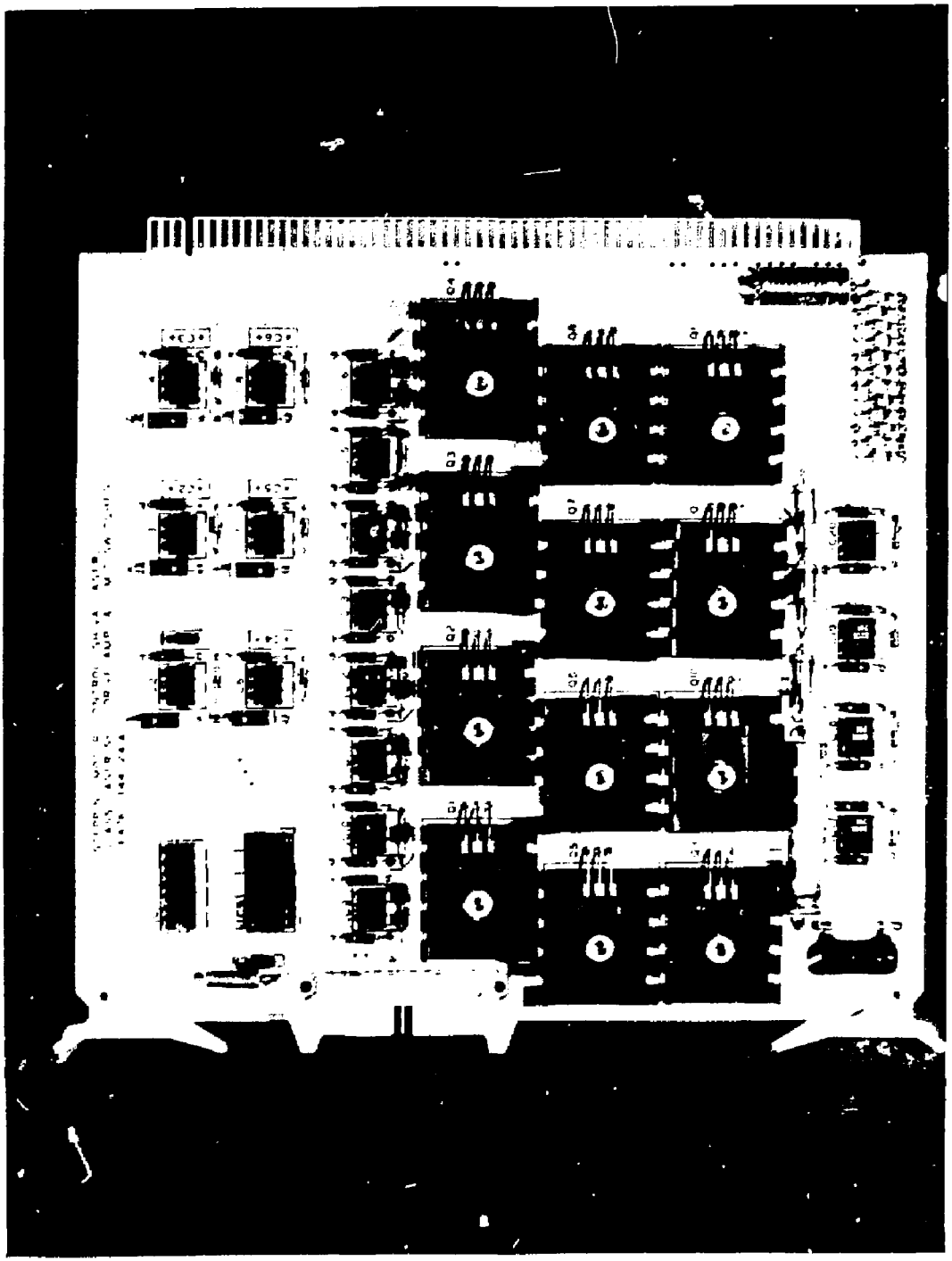
The drive amplifier card contains R/L motor drives and limit switch circuitry for three 4-phase stepping motors. Transistor bias is on-board regulated to allow operation of any number of cards from the unregulated stepping motor power supply (24V+ 4V). Both drive circuits and limit switches are optically isolated from the computer for protection and noise immunity.

Limit switches are operated from the 24 V supply and buffered by schmidt triggers. A plug in chip IC5 allows operation from either normally closed (IC5=8097) or normally open (IC5=8098) limit switch configurations. Limit switches may be debounced by loading a .15μF cap in C1-C6 if this is necessary in a given application.

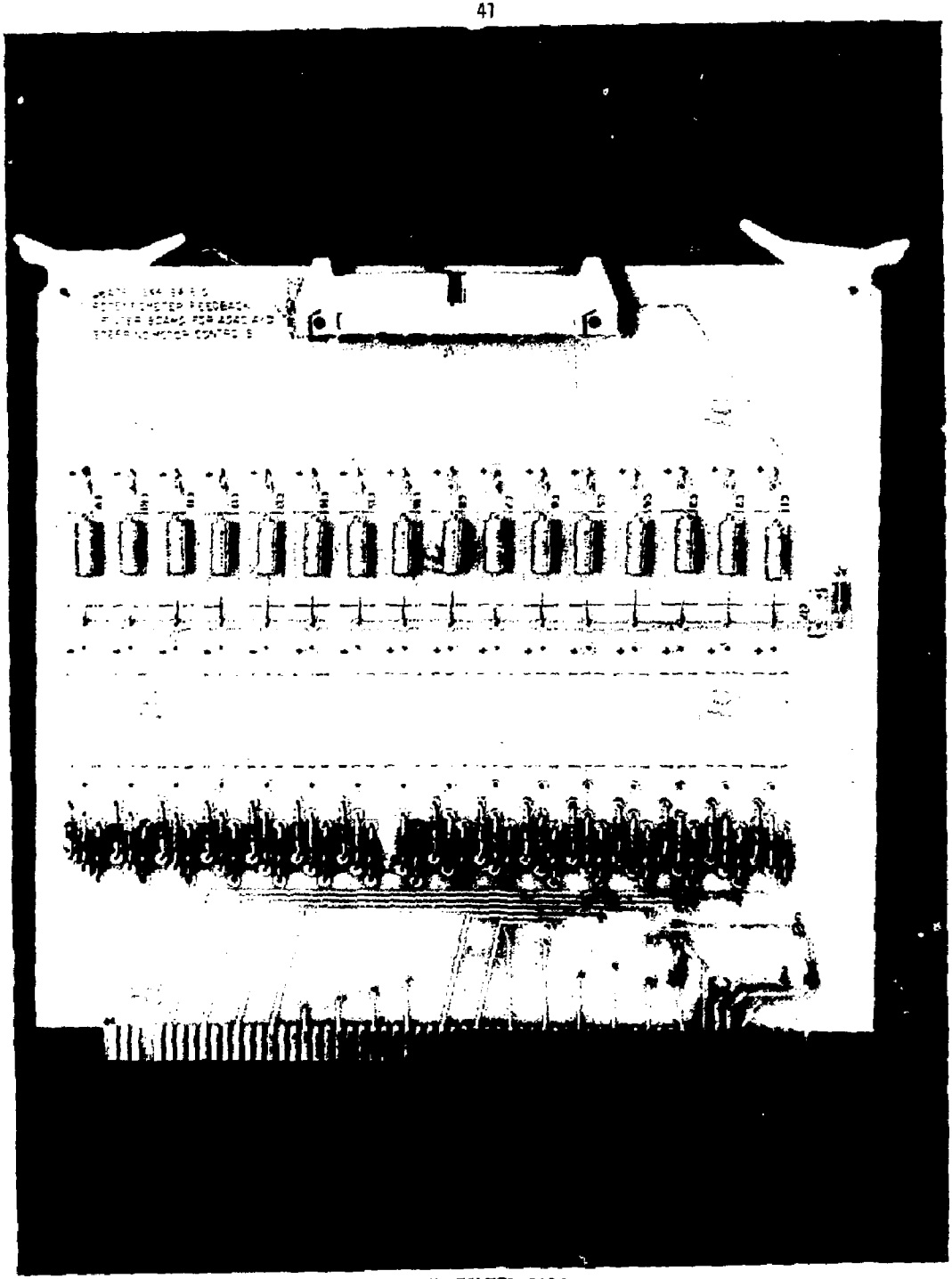
The drive circuits have been designed to sink up to four amp/phase motors with appropriate external series dropping resistors. At these high currents, objectionably large resistors (100W) are required however, and adequate forced cooling is needed. The board can be used comfortably with motors have winding current ratings up to 2 amp/phase.

3.9 Pot Filter Card LEA76-1344-61A

This card is designed for implementing single pole RC filters on 16 analog data lines from a 122 pin Augat format to an ADAC 600 LSI-11 analog to digital converter in the LSI-11. The card installs in the standard step motor drive amp chassis or other Augat card cage. Analog signals are routed on twisted pair 40 conductor flat ribbon cable from this card to the ADAC board. (Spectra Strip PN #455-248-40). Analog signals are arranged single ended coming in the board and differential out for driving the cable to the ADAC module. All 16 lines are diode clamped to the high and low pot supply rails for transient protection.



STEPPING MOTOR DRIVE AMP CARD



POTENTIOMETER FEEDBACK FILTER CARD

Either unipolar or bipolar supplies may be used. 1. For unipolar supplies (note: a 10V supply is standard in the Drive amp chassis) jumper C to COM on the board and install tantalum capacitors at C1 to C16 in the solid outlines. The jumper sets the proper clamp down voltage for the input diodes. 2. For bipolar supplies jumber C to V- and install back to back tantalum caps in the solid and dotted areas on the board. This will prevent reverse leakage on the caps. (See schematic)

Additional jumpers are provided to allow monitoring V- and V+ supplies on analog channels 14 and 15 respectively. R17 and C17 provide a high frequency path from analog (differentially floating) gnd to supply gnd. These values are selected to minimize the effect of high frequency noise on converter stability. Typical values are 1 meg and .01 μF. Filter resistors should be 5K ohms or less for proper operation with the ADAC board.

IV. SOFTWARE DESCRIPTIONS

4.1 Software Summary

The software for SMC is burned into ROM memory. It is configured for either a maximum of ten channels of three motors per channel, or twenty channels of two motors per channel, selectable by an internal switch. This description applies to ROM chips labled "RM5".

Package Contents

Internal control software in the SMC includes the following routines:

1. (BSLAVE) Remote control B-processor SLAVE execution
2. (MTBXEC) Local control MoTor B processor eXECutive
3. (MTBCTL) MoTor B processor ConTroL
4. (PANEL) Control PANEL handler
5. (MOTORS) Stepping MOTORS handler
6. (MPOTAD) Motor POtentiometer A-to-D handler
7. and a number of small support routines

Internal communication software consists of a complete SHIVANET package in a slave configuration. This package is described in a separate chapter.

4.2 Brief Overview of the Internal Control Programs

Figure 9 diagrams the SMC processor software structure, including each of the control software programs, and a single block symbolizing all of the SHIVANET software. All of the entry points and calling points between the control software modules, and between the control software and the SHIVANET software are shown.

At the bottom of Figure 9 are the standard BASIC-language-callable handlers developed for stepping motor operations ("MOTORS"), and for control panel interactions ("PANEL"). Also shown is the special analog-to-digital handler developed to read the optional potentiometer voltages for motor shaft encoding provided by the SMC-processor package. Many of the subroutines were written in a basic language callable form so as to be easily checked out. The routine "BCALLR" is used to simulate a basic call, doing the same functions as the BASIC interpreter regarding the calling arguments.

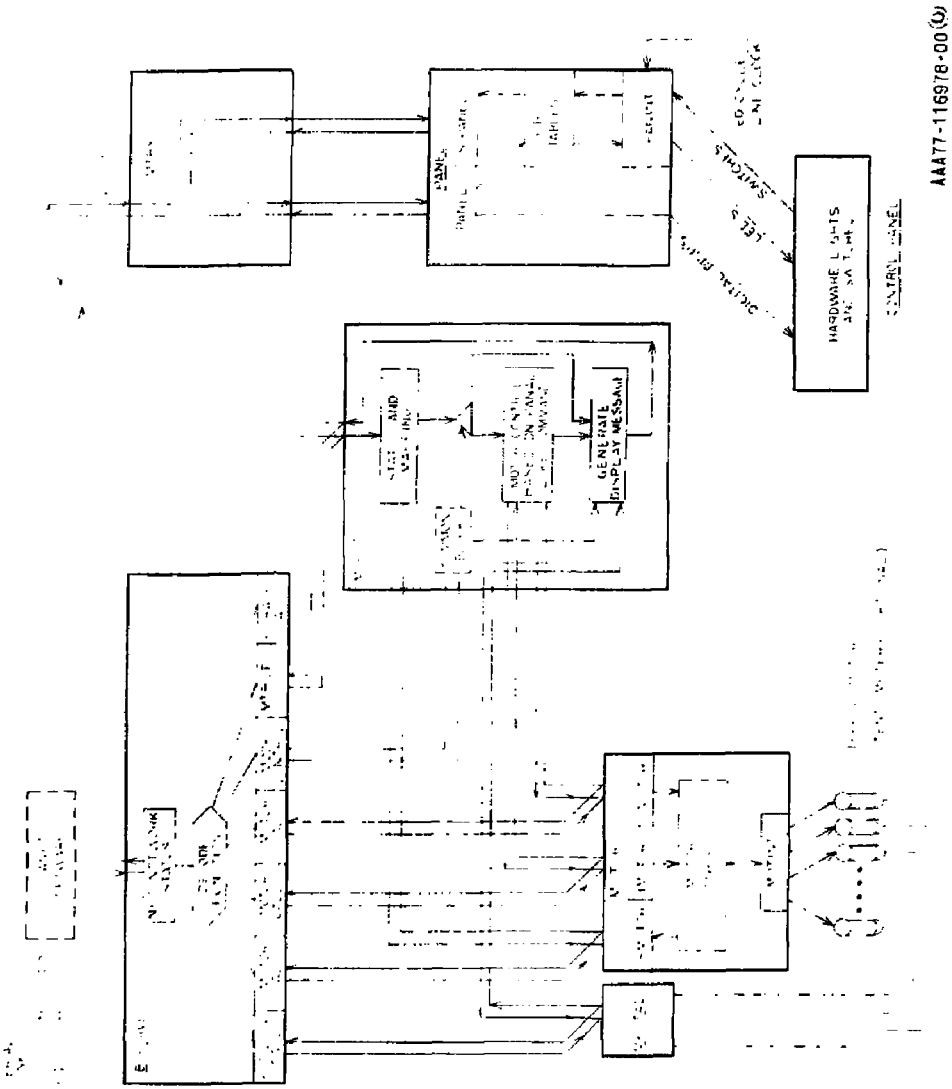


Figure 9.
SMC Processor Software Structure Diagram

"MOTORS" and "PANEL" each have internal buffer tables and function on an interrupt basis. They provide the functions requested by the calling programs at the response speed of the hardware, while allowing the other internal programs (network link for example) to continue to function. Calls to the handlers can originate from two completely parallel, concurrently operable sources:

1. The internal motor SMC processor control loop, "MTBCTL" which communicates with the local control panel.
2. Direct network calls from programs in higher level processors making use of the control panel and motors for their own reasons (via BSLAVE(. Numerous extensions beyond the capabilities of the local panel are provided, including: multiple position declare operations and execution of downline loaded routines.

MTXMTR

The BASIC callable MTXMTR subroutine resides in an upper level processor and performs a set of convenient parameter array - to - network array conversions for simplifying the remote control of a SMC unit. Many arguments used in the calls provided in MTXMTR have identical names and functions as arguments described under the MOTORS subroutine, and the SHIVANET software. Those descriptions should be consulted.

Method:

BASIC calls from the master to the stepping motor control units (SMC) are available in a "TRIPLET" form. This method has allowed the separate development of the network software from the motor MTX unit software. TRIPLET form calls consist of three secuential BASIC calls:

1st: Packs normal BASIC-like arguments into a byte array for network transmission
2nd: Network transmissions - (send and receive(
3rd: Unpack network received byte array into BASIC-like arguments

The BASIC user must not call the third form until the network transmission is complete and error free.

Triplet forms for the motor MTX unit are listed below (Table 9). For reference, the send and receive buffer formats are also listed. The BASIC user should refrain from direct use of the buffer contents as their formats may change in later versions.

Errors in the first form calls result in BASIC return to "ENTER" mode with error #104. Errors in the third form calls return an error code as an argument.

TABLE 6

TRIPLET FORM TABLE

Note: Arguments for the following calls are described only once. If the argument is not described under a call, it is the same as described under the previous call.

Set a declarable position for a list of motors

```
CALL   MXIMP   (DCPN, NS, AXIS, POSITN, NUMS, SARRAY)
CALL   SLINK   (ID, NUMS, NUMR, SARRAY, RARRAY, LKERR)
CALL   MXIMU   (NUMR, RARRAY, ERCODE)
```

Where:

DCPN = Integer Value,
Declarable position number (1 to 9) (cannot set #0)
(Position #0 is always the absolute reference)

NS = integer array,
(0) No. of axis (i.e. mapped MOTORS) being referred to

AXIS = Byte Array, NS(0) Bytes long
Containing the axis numbers being referred to (for example, on a 1-to-1 mapped SMC processor with 3 motors per chain, the following codes would point to the chains and motors given:

1	Chain 1	X
3	Chain 1	Z
4	Chain 2	X
5	Chain 2	Y

POSITN = Integer array, NS(0) words long
containing the declarable position data

CMDBUFF: (Route/ FC / NS(0) / AXIS / POSITN / DCPN)
RSPBUFF: (ROUTE /FC/ERR)

Arguments for all network calls:

ID = Network link unit number

NUMS = Integer variable set by packing routines to indicate amount of SARRAY used.

NUMR = Integer variable set by SLINK call which must be passed intact to unpacking routine

SARRAY = Byte array used by packing routines to put other packing routine arguments into for transmission.
Note: MTXMTR packing routines automatically fill in 0,0,0,0 for routing bytes

```
     RARRAY  = Byte array used to receive SLINK response to be
               passed to unpacking routines
     LKERR   = SLINK error code
```

To command motor motions:

```
     CALL   MXCMP   (NS, AXIS, RATEDV, DIST, NUMS, SARRAY)
     CALL   SLINK   (ID, NUMS, NUMR, SARRAY, RARRAY, LKERR)
     CALL   MXCMU   (NUMR, RARRAY, ERCODE)

     NS     = Integer array
              (0) = Number of axis to move
              (1) = Option flag for all motors referred to in this call
                       0 = No options
                       1 = Override upper limit switch
                       2 = Override lower limit switch
                     Note:  for protection, the option flag is cleared
                            after each call so the user must reset it

     RATEDV  = Integer array, NS(0) words long
               containing the motor rate divider values,
               speed = 300 / (RATEDV-1)      Steps per second
     DIST    = Integer array, NS(0) words long
               containing the  distance to move each motors (in steps)
CMDBUFF:       (Route/FC/NS(1)/NS(0)/ AXIS / RATEDV / DIST )
RSPBUFF:       (Route/FC?ERR)
```

Get motor status and positions relative to a declarable position

```
     CALL   MXSMP   (DCPN, NS, AXIS, NUMS, SARRAY)
     CALL   SLINK   (ID, NUMS, NUMR, SARRAY, RARRAY, LKERR)
     CALL   MXSMU   (NS, CDIST, CPOSIT, CTRANS, MSTAT, NUMR, RARRAY,
                    ERCODE)

     NS        = Integer array,
                 (0) No. of axis to get status of
                 following filled in by MXSMU:
                 (5) No. of motors in entire SMC unit with power on
                 (6) No. of axis in entire SMC unit which are at center
                 (7) No. of SMC processor bus errors since last call
     CDIST     = Integer array, NS(0) words long
                 Set to distance to go (if not 0, its in motion or at limit)
     CPOSIT    = Integer arry, NS(0) words long
                 Set to current position relative to the declared position no.
     CTRANS    = Byte array, NS(0) Bytes Long
                 Set to current stepping motor translator state
     MSTAT     = Byte array, NS(0) Bytes Long
                 Set to current motor status:
                 -3 = Interface not operational
                 -2 = Cable not connected
                 -1 = Stopped at lower limit
                  0 = Stopped at upper limit
```

```
CMDBUFF:        (Route/FC/NS(0)/...AXIS.../DCPN)
RSPBUFF:        (Route/FC/ERR/NS(5)/NS(6)/NS(7)/...CDIST.../...CPOSIT...
                    /...CTRANS.../...MSTAT...)
```

Read Motor Potentiometers ---Similiar to "MXSMP"
See 'MPOTAD' for argument descriptions

```
     CALL   MXAMP   (0, NS, PCHANS, NUMS, SARRAY)
     CALL   SLINK   (ID, NUMS, NUMR, SARRAY, BARRAY, LKERR)
     CALL   MXAMU   (NS, POTDATA, NUMR, RARRAY, ERCODE)
```

The '0' in MXAMP does nothing but allow sharing of code with MXSMP

```
CMDBUFF:        (ROUTE/FC/NS(0)/ AXIS / DCPN)
RSPBUFF:        (ROUTE/FC/ERR/ POTDATA)
```

Control mode and display control of mix unit

```
     CALL   MXMOP   (MODE, NUMS, SARRAY)
     CALL   SLINK   (ID, NUMS, NUMR, SARRAY, BARRAY, LKERR)
     CALL   MXMOU   (PMODE, NUMR, RARRAY, ERCODE)

     MODE   = Integer Value,
            Mode to set SMC unit into
            0 = Local control only
            1 = Local or network control
            2 = Network control only
     PMODE  = Same as mode - for response verification

CMDBUFF:    (ROUTE/FC/MODE)
RSPBUFF:    (ROUTE/FC/ERR/PMODE)
```

Remote control "MTXLOP" processor call ---Similiar to "SPANEL"
"MTXLOP"
"LPANEL"

```
     CALL   MXCTP   (HEADER, PBARRY, IGARRY, NUMS, SARRAY)
     CALL   SLINK   (ID, NUMS, NUMR, SARRAY, RARRAY, LKERR)
     CALL   MXCTU (HEADER, BITLIT, DIGLIT, NUMR, RARRAY, ERCODE)
```

This call is provided for applications where an exact (or simulated) duplicate control panel exist on the master and on the SLAVE, and it is desired to operate the slave unit "directly" from the master control panel.

HEADER = Byte array, 4 bytes long
Used to keep user specific flags related to special operations by the SMC unit. This array must be present, initially preset to 0 and passed on all 'MXCTP' and "MXCTU' calls

PBARRY = Byte array, 22 bytes long
containing push button like data as defined by the control panel on the SMC unit:

(0)	= +X slew
(1)	= -X
(2)	= +Y
(3)	= -Y
(4)	= +Z
(5)	= -Z
(6)	= Stop All Motors
(7)	= Stop 1 Chain of Motors
(8)	= Center All Motors
(9)	= Center 1 Chain of Motors
(10)	= Move to Limit, All Chains
(11)	= Move to Limit, 1 Chain
(12)	= Declare Centered, All Chains
(13)	= Declare Centered, 1 Chain
(14)	= Display Pot Data
(15)	= Display Motor Positions (not pot data9
(16)	= Display Motor Positions Relative to Declared Pos 1
(17)	= Display Absolute Motor Positions
(18)	= Unused
(19)	= Unused
(20)	= Select Network Control
(21)	= Select Local Control

DGARRY = Integer array, 3 words long
Containing digiswitch like data

(0)	= Chain No.
(1)	= Rate Code
(2)	= Declarable Position No.

BITLIT = Byte array, 29 bytes long
to receive led like response information

(0)	= X high limit
(1)	= X Low
(2)	= Y High
(3)	= Y low
(4)	= Z High
(5)	= Z Low
(6)	= Stop All Completed
(7)	= Stop 1 Completed
(8)	= Center All Completed
(9)	= Center 1 Completed

```
            (10)     = Slew to Limit, All Completed
            (11)     = Slew to Limit, 1 Completed
            (12)     = Declare All Completed
            (13)     = Declare 1 Completed
            (14)     = Displaying Pot Data
            (15)     = Displaying Motor Positions (Not Pot Data)
            (16)     = Displaying Relative Motor Positions
            (17)     = Displaying Absolute Motor Positions
            (18)     = Unused
            (19)     = Unused
            (20)     = Network Control Enabled
            (21)     = Local Control Enabled
            (22)     = Control Loop Operating Light (Blinks)
            (23)     = Network Command Received Light
            (24)     = Network Errors Detected Light
            (25)     = Unused
            (26)     = Unused
            (27)     = Unused
            (28)     = Chain Not Operable from Here
DIGLIT      = Integer Array, 3 Words Long
            To Contain the Digital Display Data Response
            (0)      = X Axis Data
            (1)      = Y Axis Data
            (2)      = Z Axis Data

CMDBUFF:    (ROUTE /FC/...HEADER.../...PBARRY (BITS).../...DGARRY...)
RSPBUFF:    (ROUTE /FC/ERR/...HEADER.../...BITLIT (BITS).../...DIGLIT...)
```

Data Storage and Retrieval in B-Processor Core Buffer

```
    CALL   MSTRP   (FUNCT, SEGMNT, N, ADDR, DATA, NUMS, SARRAY)
    CALL   SLINK   (ID, NUMS, NUMR, SARRAY, BARRAY, LKERR)
    CALL   MSTRU   (NN, DATA, NUMR, RARRAY, ERCODE)
```

Where:

```
    FUNCT  = Integer value specifying operation
           = 0,   Return total length of buffer available (bytes)
           = 1,   Read buffer starting at relative address 'ADDR'
           = 2,   Write buffer starting at relative address 'ADDR'
           = 3,   Call a previously loaded subroutine
                        (The 'N' parameter is ignored)

    SEGMNT = Integer value specifying segment no. of B-processor memory
           = 0,   Fake segment, default when reports errors
           = 1,   Users data buffer
           = 2,   Users downline loaded subroutine area
           = 3,   Declarable positions buffer (MTBCTL)
           = 4,   Configuration word          (MTBCTLO
           = 5,   Motor no's as mapped        (MTBCTL)
```

```
     N       = No. of bytes to read or write (240 maximum)
     NN      = No. of bytes of data returned

     ADDR    = Integer value of byte address in buffer (0 to max size-1)

     DATA    = Callers basic I/O array (any type)

CMDBUFF:        (ROUTE/FC/SEGNO/UNUSED/N/ADDR/...DATA....)
RSPBUFF:        (ROUTE/FC/ERR/...DATA....)
```

Error Codes (ERCODE):

```
     0    = CAll accepted and processed
     1    = Protection violation - this call not acceptable
              Because MTX unit is in local control only mode
     2    = Illegal function code received by slave
     3    = Function code returned not same as sent
     4    = Attempted access beyond legal buffer in BSLAVE
            ('MSTRP' call)
     5    = Combination code no good on call to downline load SUBR

     -101, -102          Error codes defined in motors and returned
                         for SLAVE unit
```

Basic Error Returns to Enter Mode

```
     104   = Arguments wouldn't all fit into SARRAY
S              or 'N' in "MSTRP' call not 0 to 240.
     105   = "NUMR" parameter indicates that RARRAY was not
             large enough on unpack call (i.e. program error...
                     ...you or Greg!...)
     106   = Tried to set declarable position no. 0
```

Changes:

```
     1/3/78 Fixed bug in MSTRU call found by Gordon Snyder
     1/9/78 Added segment parameter to storage and retrieval functions
     1/23/78 Removed swit and BCNT parameters from MXMOP call
     2/14/78 Added DCPN parameter to select declarable positions
                       (no code changes)
```

SMC PROCESSOR REFERENCE FRAMES, REVERSALS, AND DECLARABLE POSITIONS

Reference Frame

The absolute motor position reference frame for all time and all operations (local and remote) is the position information within the motor subroutine tables.

In the case where motors are inadvertently wired up backward, or somebody decides to reverse coordinates, switches and provided on the local control panel interface board for each axis to allow axis mapping and data reversal of all data to and from the motor subroutine. This applies to the limit switch status information also.

This reversal occurs immediately prior and after all calls to the motor subroutine from all sources, so it affects all motor operations.

Access to the A/D converters for the Pot data are mapped according to the Axis/motor no's mapping in use, and the data is reversed in the case of reversed motor motions on corresponding motors the current position data in the motor tables (absolute reference) cannot be changed by any local or 'A' processor by any means except by moving the motors.

Declarable Positions

There are now 10 different declarable positions which can be used for accuratley moving to one of 10 predefined points.

A parameter specifying the declarable position number of reference has been added to the remote IMOTOR (MXIMP) call and the remote SMOTOR (MXSMP) call.

Declarable position #0 is always the absolute reference frame zero point (motor table position zero), and cannot be changed by man or beast.

The SMC processor local control loop normally displays and operates with respect to declarable position #1, since it has no selector digiswitch.

The digital display data generated by the control loop for local or remote (MXCTP) operations is normally calculated as:

Display Data = Declared Position(n) - Motor Position

where:

N	= Declarable position No.	(1 in local) (0 to 9 remote)
Motor Position	= Motor table value * Reverse switch	

There is a momentary action switch on the control panel which overrides the declarable position selector switch and causes the display to be referenced to the absolute position reference (uses declarable position #0)

The declarable position data is preserved thru power failures and restarts.

The declarable position data can be changed by two means:

1. Remote use of the IMOTOR call (MXIMP) which has a parameter defining the declarable position number of reference (cannot be #0 from BASIC)

2. Use of the "declare" push button in either the local or remote mode (MXCTP). The local operator can only change declarable position #1, while the remote operator can change #1 thru #9.

To set up a declarable position so that a "preset" value will be displayed when at the present location, without ever moving the motors to the actual declarable position:

```
DCP(#1) = Absolute Position - Preset Value
        = SMOTOR (#0)       - Preset Value
```

The displayed value is:

```
Display = [ABS - DCP]
        = [ABS - (ABS - PRESET)]
        = PRESET
```

MTBXEC

This internal control loop routine runs in the processor idle loop at approximately 20 times per second, and executes a single sequence of calls to:

1. "SPANEL" to get switch information
2. "MTBCTL" process switch information, control motors, get results
3. "LPANEL" to display the results on the local panel

It is responsible for all local control panel operations.

MTBCTL

The MTBCTL subroutine is the heart of the control operations provided by the SMC processor. Calls to this subroutine come from the local executive MTBXEC or from the remote control request processor BSLAVE (described below).

The caller passes an array of command information (control panel switches(to MTBCTL to be processed. MTBCTL in turn calls for motors to be moved, or potentiometers to be sampled, as required, then passes back an array of results to be displayed on a control panel or returned back to a remote caller.

This BASIC language callable subroutine functions in a single *pass manner to control stepping motors from a special purpose command* array containing control-panel-like switch information.

The "MOTORS" subroutine is called as required by the command array and a response array containing control-panel-like light information is generated and returned to the caller. MTBCTL contains all arrays necessary for its communications with the MOTORS subroutine and the MPOTAD A/D routine.

The command information is an array of the following format:

- 4 Bytes of Header Information
  Byte 0 = 0 for local control, ≠ 0 for remote control
  Byte 1, 2, 3 set up by MTBCTL. Must be returned intact in subsequent calls.
- 22 Bytes of control panel switch information;

Byte 0 +X slew
1 -X
2 +Y slew
3 -Y
4 +Z slew
5 -Z
6 stop all motors
7 stop 1 chain of motors
8 center all motors
9 center 1 chain of motor
10 move to limit, all chains, select X, Y, Z
11 move to limit, this chain, select X, Y, Z
12 declare centered, all chains
13 declare centered, this chain

14 display pot data
15 display current motor position
16
17 display absolute motor position
18
19
20 network control select
21 local control select

. 2 words of BDC information:
word 0 = chain number
word 1 = rate code (0 to 9)
word 2 = declarable position number (1 for local calls)

. 3 words of configuration data -(local calls only)
word 0 - not used
1 - not used
2 - used for mapping function described below

The response information is returned in the following format:

. 4 Bytes of Header - as in command

. 30 Bytes of Response Information:

0	X high limit light
1	low
2	Y high limit light
3	low
4	Z high limit light
5	low
6	stop all motors completed
7	1
8	center all motors completed
9	1
10	limit slew to, all motors completed
11	1
12	declare all motors centered
13	1
14	displaying pot data
15	displaying motor positions
16	Relative motor position displayed
17	Absolute motor positions displayed
18	Option 3 up
19	down
20	Network control enabled
21	Local control enabled
22	Control loop operating light
23	Network command received
24	Errors detected
25	Extra light No. 1
26	2
27	3
28	Chain not operable from here
29	Not used

The header is used as storage for conditions which are unique to a particular caller and must be remembered from call to call. These conditions are: single step mode flag, potentiometer data display flag, X-Y-Z slew on flag, declare all flag, declare one flag. Users must return header on subsequent calls.

As currently assembled, the following parameters define the capabilities:

```
MAXCHN = 20.          ; maximum number of chains (SHIVA)
MNPERC =  3           ; maximum motors per channel
MCHPBOX = 20.         ; maximum channels per box
MNMOTOR = 40.         ; maximum number of motors per box
MAXMOV = 32766.       ; maximum number of steps - slew mode
DTMAX = 100.          ; maximum number of cycles to wait for declare
                          2nd push
MAXDCP = 10.          ; maximum number of declarable positions
```

MTBCTL performs the following detailed sequence of operations:

ENTRY and Initialize

- BASIC LANGUAGE ENTRY
- Transfers command array into internal buffer
- If it's a network call, and mapping hasn't been done yet, ignore call
- If mapping hasn't been done, do it by calling an internal subroutine "MAP"
- Reverse S, Y, Z commands if required by mapping options
- Convert desired chain number to channel number
- Process the "network/local control" switch and set up mode
- Process the "Display Pot Data" switch
- Get current status of all motors

Control Allowed?

- If control not allowed by this caller, as determined by network/local switch, go to Display Set up section

Motor Controls

- Convert "Declare" PB's to single shot operation
- Clear any "Z" direction commands if in 2 motor/channel mode
- Perform the "centering control' functions (i.e. "center," "limit," "declare," "stop") as a call to "CSWSUB" subroutine for each chain as required.
- perform the X-Y-Z slew operations

Set Up Display Results

- Initially set all lights off
- Set up position displays, reverse axis as configuration

- . If pot data to be displayed, do A/D conversion (call MPOTAD) and scale it if configuration bit set
- . Turn on "This Chain Stopped" LED if it is stopped.
- . Turn on, or blink, "Slew to Limit" LED as required
- . Turn on, or blink, "Declare One" LED as required
- . Turn on "at ± X,Y,Z limit LED's of at limits (solid if at limit, or cable disconnected, blinking if interface is not operational)
- . Turn off "Declare All," "Limit-all", "Stop-All", "Center-all" LED's as required.
- . Set up other control panel LED's as required.

Exit

- . Transfer internal response button to caller
- . Return to caller

The "CSWSUB" centering control internal subroutine performs the stop, center, slew-to-limit. and declare-center functions for a particular channel. It keeps track of what is happening on each channel by a value in that channel's element of a "DOING" array. The sequence of operation is:

- . If stop PB is pushed, stop motors, clear "Doing" and exit.
- . If center PB is pushed and not already doing a center, stop motors, get new motor status, calculate distance to move to center (=-current position) and move the motors.
- . If center PB is not pushed, and was doing center operation which is now complete, reset "DOING" value.
- . If slew-to-limit PB is pushed, and motors are not now doing a slew to limit, command motors to go 32K steps in proper direction at rate selected. (assumes 32K will get to limit)
- . If slew-to-limit PB is not pushed, and have just completed a slew-to-limit operation, reset DOING value
- . If "Declare" PB is pushed first time, set timeout for second push.
- . If "Declare" PB is pushed second time, set current positions of this channels motors to zero.

The "MAP" internal subroutine is used to map motor numbers onto the X,Y,Z axis of each channel by filling in numbers in a motor number array which is indexed into by the channel number. The filling in of these motor numbers is done based on a repetitive cycle using the information in the following table. The "map code" is the three least significant bits of the control panel interface board configuration switches.

Codes to Map X, Y, Z Axis Into Motor Numbers

Map Code =	0	1	2	3	4	5	6	7
Xmotor:	1	1	2	2	3	3	1	2
Ymotor:	2	3	1	3	1	2	2	1
Zmotor:	3	2	3	1	2	1	1	2

MAP also sets up the following internal parameters based on the control panel configuration switches for use by all other sections of MTBCTL:

CFIGWD = configuration operating on (axis rotations, reversals, etc.)

NPERC = number of motors per channel (2 or 3)

CHPBOX = number of chains operational (0 to 20)

NMOTOR = number of motors operational (0 to 40)

BSLAVE

This subroutine is called whenever a command message is received by the Stepping Motor Control (SMC) processor from a higher level processor over the SHIVANET network. It decodes the command message, calls the appropriate subroutine, and returns the response message. The following separate functional capabilities are provided:

1. Direct calls to the various handlers so that an upper level processor can implement its own control structure.

2. Nonvolatile (saved during power on/off cycles) storage of any data an upper level processor desires, in the unused portions of the internal core memory.

3. Calls to "MTBCTL" with control panel like data, which then lets "MTBCTL" do all the control logic. Results and present status are then passed back in a control panel like data array for immediate display on the upper level processor control panel.

Functions provided are those described below under the MTXMTR program which is BASIC callable and runs in the higher level processor. All functions are performed completely in parallel with the local functions provided by MTBXEC. BSLAVE contains its own arrays for MOTOR calls.

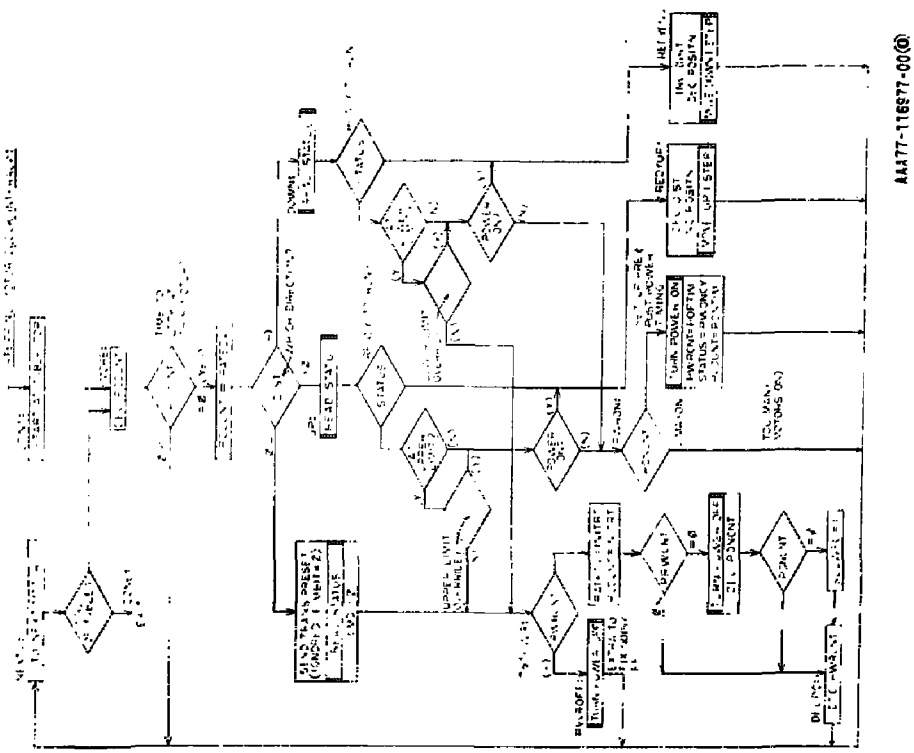


Figure 10
Stepping Motor 300Hz Clock Service Routine

BSLAVE also contains a buffer for storage of general data and downline loaded programs for an upper level processor. This area is what remained of the 4K core memory within the SMC unit after the needs of all other routines were met.

A small common area "MTBCOM" is shared between BSLAVE, MTBXEC, and MTBCTL for network related parameters.

MOTORS

This BASIC language callable subroutine, in conjunction with a set of special LLL designed stepping motor control interface boards, will control a large number of stepping motors directly from an LSI-11 on a one-step-per-motor-per-interrupt basis.

Interrupts are generated by a separate clock interface. The interrupt causes this program to scan once through an internal table and output an up or down command to each motor if motions have been requested and the limit switches permit.

The same clock interrupt is used by the motor control interface boards to control buffering of the commands to the motors.

The interrupt handling section is written to minimize execution time. It is estimated that:

- With interrupt rate = 300 HZ
- 30 motors being considered

The LSI-11 processor time consumed is:

2%	If 0	motors in motion	
25%	If 1	" " "	at highest rate (300 HZ)
48%	10		
95%	30		(Can't supply power)

This establishes the maximum capability - with a lower clock rate or fewer than 30 motors "considered," the % times would be less. The interrupt routine keeps track of the number of motors with power on and limits it to 10. Additional requests are saved until motions on another motor are completed and its power is turned off.

The BASIC language calls are:

1.) To initialize motor positions and internal tables:

```
CALL     IMOTOR    (NS, MNUMB, POSITN, TRANS)
```

where:

NS = array of control numbers - see below
MNUMB = byte array of motors numbers (1 to max. no. of motors)

POSITN = array of initial preset positions
TRANS = byte array of initial trans preset states (0,1,2,3)
(-1 = leave current trans state alone)

NS format is:

NS(0) = Number of words to use from other arrays
NS(1) = - not used -
NS(2) = - not used -
NS(4) = - not used -
NS(5) = - not used -
NS(6) = - not used -
NS(7) = - not used -

Operation:

The passed arguments are inserted into the tables, while unselected motors are not disturbed.

2.) To command motion of motors:

CALL CMOTOR (NS, MNUMB, RATEDV, DIST)

Where:

NS = array of control numbers - see below
MNUMB = byte array of motor numbers for this call
RATEDV = array of rate divisors (rate=clock/RATEDV-1)
DIST = array of (+/-+ distances to go

and NS format is:

NS(0) = number of words to use from other arrays
NS(1) = option flags for all motors referred to in this particular call
= 1, override upper limit switch
= 2, override lower limit switch

Note: For protection, this word is cleared after each call to CMOTOR so the user must intentionally set it again.

Operation:

This call can be made concurrently with motor motions. Great care is exercised within the motors subroutine in inserting the new distance and rate divisors so as to cause a smooth transition from the old rate of motion to the new rate.

3.) To determine current status of motors:

```
CALL    SMOTOR    (NS, MNUMB, CDIST, CPOSIT, CTRAN, MSTAT)
```

Where:

```
NS      = array of control numbers
MNUMB   = byte array of motor numbers for this call
CDIST   = array to put current value of distance to go into
CPOSIT  = array to put current value of position
CTRAN   = byte array to put current translator status (0,1,2,3)
          (5,7,7,8 if trans not yet sent to motors)
          byte array to put current motor status in
MSTAT   = (-3 = interface not operational)
          (-2 = motor cable not connected)
          (-1 = stopped at lower limit)
          ( 0 = not at limit)
          (+1 = stopped at upper limit)
```

and NS format is

```
NS(0)   = number of words to use from other arrays
NS(1)   = - not used -
NS(2)   = - not used -
NS(3)   = - not used -
NS(4)   = - not used -
NS(5)   = number of motors with power on (returned)
NS(6)   = number of motors at center (i.e. CPOSIT = 0)
NS(7)   = number of bus errors since last SMOTOR call
```

Motor Tables

The MOTORT routine contains the actual motor tables which use eight words per motor. The number of motors to be controlled is a parameter at system generation time which sets up the table size. The format of the motor tables is:

	PARAMETER	DESCRIPTION
Word 0	PCOUNT	Counter to gen rates
Word 1	RATEDV	Rate divider
Word 2	MBUSAD	Bus addr of this motor
Word 3	DIST	Distance to go
Word 4	POSITN	Current position
Word 5	STATUS	Motor status
Word 6	CMD	Translater MBIT command
Word 7 LSB	MFLAGS	Mode flags
Word 7 MSB	PWRCND	Power turn off counter

This format repeats for every motor, and is initially set up by the 'MOTPUP' entry.

The motor bus address sequence is:

160100
160102
160104
160106
160110
160112
.....
160120
160122

MOTORS can generate two BASIC error returns:

#101 - illegal motor number (not 1 to max motor no.)
#102 - arrays not large enough on SMOTOR call

The interrupt routine scans through motor tables and outputs up or down pulses for any motor which is due to be moved and which is not at a limit switch which restricts the motion.

The general functions of this interrupt service routine are:

. Current position tracking
. Power on/off control for low heat dissipation
. Limits number of concurrent power on's to protect supply
. Catches bus failures (on motor I/O)
. Programmable distances
. Selectable rates
. Direct translator read/writes - can override limit switches
. Interface status feedback
. Software limit switch override control

Power is applied to the stepping motor for 200 ms prior to any motion, and for one second after motion has completed. This is necessary for some types of electro-mechanical activators.

The interrupt routine runs at 0 (low) priority so as to not interfere with serial link timing. It is interlocked so as to ignore processing of an interrupt if it is already processing an interrupt.

Also, when all power is off and hence no motors are in motion, the interrupt routine will only use one out of every 30 interrupts, thereby reducing the average CPU time usage to -early zero.

The following figure shows a flow diagram for the interrupt routine.

PANEL

This subroutine, in conjunction with LLL designed control panel interface boards for the LSI-11, will receive a large amount of push buttons and switch information of BIT and BCD format from a control panel, and send a large amount of 1-BIT LED and BCD coded digital display values to a control panel.

Formats and calling sequences are written for BASIC language user convenience.

Five interface cards can be handled by this routine. The maximum hardware interfacing capability per card is:

Input:	3 ea	16 bit	Push button words
	1 ea	16 bit	Combination push button and internal "Dip" switch word for configuration control
	10 ea	+/-9999	BCD coded digiswitch banks
Output:	4 ea	16-bit	LED light driver words

BASIC calling sequences are:

1.) To test if any of the push buttons have changed since the last "TPANEL" or "SPANEL" call:

```
CALL    TPANEL    (ICHNFG)
```

Where:

ICHNFG = Integer changed flag
(0 = no change)
(1 = PB's changed since TPANEL call)

2.) To get latest control panel switch data:

```
CALL   SPANEL  (ICHNFG, CONFIG, PBARRY, NP,
                DGARRY, ND)
```

Where:

ICHNFG = (See above)
CONFIG = (Integer array to put PB switch data into unformated
PBARRY = Byte array to put push button switch data into, formatted 1 byte/PB
NP = Number of PB's to read into PBARBY
DGARRY = Integer array to put digiswitch banks into. Format is converted from +/-9999 BCD to binary integer
ND = Number of digiswitch banks to read

The amount of data transferred is determined by the lesser of:
a) Length of arrays
b) Number of interface cards * words per card
c) "NP" and "ND" parameters

Note the PB's are read by a clock interrupt routine and "pushes" are saved. In this way any push of a PB will not be missed even if the button is not held down long enough for a BASIC call to service.

Digiswitch values are only read and converted once per "SPANEL" call.

3.) To write information onto the control panel lights:

CALL LPANEL (BITLIT, NB, SB, DIGLIT, NG, SG, MFLAG)

Where:

BITLIT = Byte array of LED light data
(-1 = Don't chance current state)
(0 = Turn light off)
(1 = Turn light on)
(2 = Blink light)

NB = Number of LED lights to control
SB = Starting hardware bit no. (0 to N0
DIGLIT = Integer array of digital data to display
NG = Number of integers to send to lights
SG = Starting hardware BCD display number (0 to N0
MFLAG = Mode flag
= 0, Single precision mode, blank leading zeros, and light overflow code on overflow
<>0, Double precision mode, blank leading zeros, including all)'s of most significant word (MSW) (no overflow is possible)

Diglit integer to BCD conversions normally operate in range of +/-9999. (Single precision mode) Double precision mode is useful to obtain the fifth digit for full +/-32767. Displays. If data = -32768. (Special blank code) display will be blanked, in single or double precision mode.

Double precision mode displays must be wired in sequential order, most significant word (1 Digit) first, then least significant word (4 digits). The sign bit is valid on both LSW and MSW word addresses.

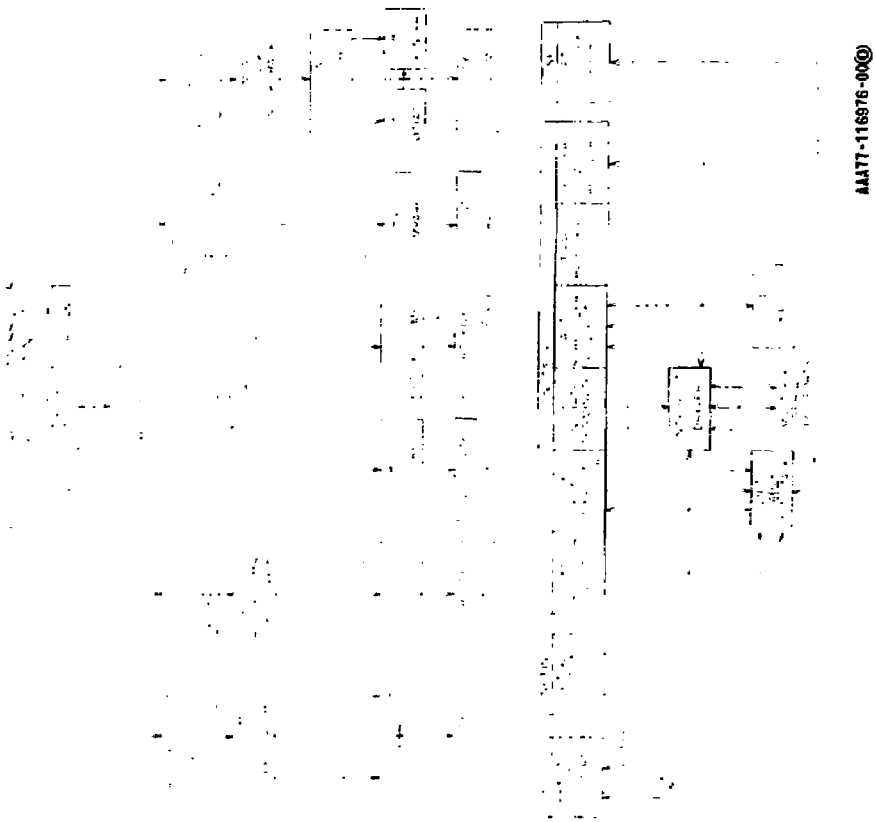


Figure 11
SHIVANET Control Flow Diagram

4.3 Brief Overview of the SHIVANET Package

The SHIVANET Control Protocol and support system is used in the stepping motor control processor (SMC) to facilitate error free communications over serial data links between the SMC and other processors (such as in remote control or data transfer functions).

The SHIVANET package is implemented in three versions: RSX-11M mapped, REBEL BASIC, and stand-alone. The stand-alone version is used in the stepping motor controller. This version of SHIVANET and the corresponding REBEL/BASIC interface is summarized in this section. Further information may be found in the SHIVANET Control Network document in Volume 2.

REBEL/BASIC Calls for Shivanet Communications

ILINK - Initialization

CALL ILINK (NUMBER, FIRST)

PURPOSE: To initialize the SCP network driver routines and tables and to start the network clock processor.

PARAMETERS:

NUMBER — The maximum number of network ports to be processed in this program. Note, there by definition is one port per serial interface (currently, all are asynchronous - DLV-11's, DL-11's).

FIRST — The unit number of the first network port. Unit numbers are used in other calls (SLINK, ISLAV) to designate the port over which communication or processing is desired. In REBEL/BASIC SCP, the unit numbers may be assigned independently of the Rebel unit numbers, but for compatibility with a possible RSX / REBEL-BASIC version of the network, it is recommended that non-conflicting unit numbers be used. Two sequential unit numbers per port are allocated. The first is arbitrarily assigned to output operations, the second (one greater) is assigned to input operations. This is done for compatibility with some operating systems which require simultaneous I/O to be performed on separate unit numbers. Either number may be used in calls to the network subroutines.
Note: 2 .LE. FIRST .LE. 2**15-1.
Note: FIRST should be an even number.

ISLAV - Initialize port as slave

CALL ISLAV (UNIT)

PURPOSE: To reset a port to slave status from the master state (ILINK initializes all ports as masters).

PARAMETERS:

UNIT Unit number corresponding to the port whose status is to be switched to slave. Any number of units may be operated in slave mode. See the following section for a description of master-slave operations in SCP.

SLINK - *Send / Receive Network Message Sequence*

CALL SLINK (UNIT, NUMS, NUMR, SARRAY, RARRAY, LKERR)

PURPOSE: SLINK is used to send a network message to another processor and to receive a message response. Note that both the sent and received network messages are individually processed and acknowledged by the network handlers, thus the response message is more appropriately viewed as a task acknowledgement or status return, rather than a message acknowledgement denoting correct receipt of date. SLINK is used both in master and slave modes. SLINK initiates the transmission of the message and returns immediately allowing further processing in the BASIC program. LKERR is set non-zero when the operation, including receipt of the response message, is complete. Multiple calls to SLINK over different units can be active at the same time, however, it is an error to initiate a call to SLINK on a unit which is busy (LKERR on previous call is still 0).

PARAMETERS:

UNIT The unit number of the port through which communications are to take place. Remember that each port has two units associated with it, one for output and one for input. The unit number for output should always be even and the input port should always be an odd number one greater than the corresponding output port no. SLINK may be called with either the input or output unit number specified as they are interchangeable in this usage.

NUMS Size of the message to be transmitted in array elements. (<257. bytes)

NUMR Size of the received message in array elements (need not be initialized before calling SLINK)

SARRAY The array holding the information to be transmitted. See NUMS.

RARRAY The array which will receive the task response.

LKERR Status word. This word is set to zero by SLINK, and is set non-zero when the reply message has been received, or the link is down (determined when a specified message retry/ timeout sequence fails for a specified number of times). The following values are returned:

0 operation pending
1 operation complete and successful
-1 ambiguous situation. The slave has accepted and acknowledged the transmitted message, but the expected task response message has not been received.
-3 unit number range is in error
-5 transmit message too large or receive buffer is zero length
-7 unit is busy on previous request

OTHER Operation unsuccessful. The following bits have meaning in this case:

Bit - = 1 if done
Bit 1 = 1 if input operation failed
Bit 2 = 1 if output operation failed
Bit 3 = 1 if input timeout was detected
Bit 4 = 1 if output timeout was detected (usually missing or defective interface)
Bit 5 = 1 if a device error was detected (can be a device interface missing or particularly noisy communication line)
Bit 6 = 1 I/O error. If not RSX system, this is a hardware error of some sort. If RSX, upper byte of this word contains the RSX error code on a call or I/O status error code.

Notes:

If the UNIT is a master, the user program calls SLINK to send a message to a remote processor. The message usually contains imbedded codes to direct it to the correct task if more than one task in the target processor exists. LKERR will be set to a 1 when the task's response message is successfully received. This response message is usually either a status array (such as in an A/D query) or merely an acknowledgement that the target task has processed the message (or initiated processing) and the processor is now ready for another network command.

In a slave processor, SLINK is called initially with a 0 length transmit message. This posts a read request to the network driver to accept any message which may arrive from the master processor connected to this unit. The slave processor, upon detecting message receipt via LKERR going to a 1 will process the message and then call SLINK with the task response message (which may be status or other data). Note that this call to SLINK automatically posts the next read for data from this unit.

Stand-alone SHIVANET

The purpose of stand-alone SHIVANET is to provide a ROM resident network package and executive to allow simple scheduling of network directed and background tasks. Asynchronous tasks, clocked events, and a down-line loading capability are included in the stand-alone version. (Down-line loading is handled by a modified LSI-11 REV-11 terminator/board card.) The SMC software package includes the network package, a clock driven control panel driver, a background processor to handle stepping motor driver initiation, and a network task routine which interprets network messages, initiates appropriate operations, and sends reply messages conveying status and task response information.

The SHIVANET Control Protocol

PROTOCOL OVERVIEW.

Two protocol levels are used in SHIVANET to facilitate interprocessor communication. The lowest level protocol provides for message transfer and error checking, with every data message requiring acknowledgement, rejection, or qualified acknowledgement responses from the target processor. Error recovery and timeouts, as described later, are used at both levels of the protocol to assure correct message transfer.

The next higher level of protocol is termined the task level protocol. This level was defined specifically to meet the needs of SMC communications. At this level, every valid message received by a target processor must be not only acknowledged by the lower level message protocol, but the target task, when the message has been received and processed, must also respond with a message. Since the content of the task response message is undefined, it can be and often is an array of status information, usually in direct response to the original message which is, in these cases, a request for status data.

This higher level protocol uses the Master-Slave principal of communication. In this implementation, either processor can take on either state. Stepping motor control processors always remain in the slave state, requiring another processor operating in master mode to initiate sequence by transmitting a message to the target slave processor. The master then waits for the slave task response. (This waiting of course does not imply task suspension - all operations are asynchronous and it is the setting of the status reply word which signifies completion.) The slave in turn is required to keep a read posted (via SRS/ISR) and to periodically check the status flag indicating whether or not a message has been received. If so, the message is processed and a response sent. Note that each message is verified by the network message level protocol before the next step in a message transfer operation can proceed.

The master processor has the responsibility of detecting links which are down and unresponsive slave processors. It also has the responsibility for recovering from timeout situations where expected responses (either message acknowledgements or tasks acknowledgements) have not been received after a given length of time. Both processors handle message errors such as bad checksums (CRC), redundant messages, and invalid formats via message acknowledgements and rejects (see below).

An error-free message interchange appears as follows:

MASTER	SLAVE
Transmit Request ---->	
	----> Receive Request Message
	<---- Send ACK of This Message
Receive ACK <----	
	<---- Send Task Response Message
Receive Task <---- Response Message	
Transmit ACK ----> of Response Msg.	
	----> Receive ACK of Response Msg.

Error Detection and Correction

Several methods of error detection and recovery are used in the SCP support package. The following summary, for the sake of completeness, applies to a port acting as a master. Slave ports operate identically with a few exceptions which are noted in the text.

ERROR CASE 1:

The simplest case of error detection and recovery occurs when one or more bits has been incorrectly transmitted, causing the CRC imbedded in the message to disagree with the CRC calculated at the receiving node. In such a case, the message receiver responds with a reject message containing the message number in which the error was found. Note that a check is made first to verify that the last byte received was an EOM. If not, we assume loss of message synchronization. At this point, if this is a slave, we wait for the master to resynchronize due to a timeout on our response: if this is a master, we send a synchronization header followed by a retransmission of the message. A port acting as a master records the number of retries in such a sequence. If a specified number of retries fails, the line is declared to be down and an appropriate error code is returned to the calling program. The program may undertake any one of several options, the simplest of which is to attempt to obtain status information from the remote link via another message sequence.

ERROR CASE 2:

Another case of error detection involves the detection of redundant messages. If a receiver detects a message with an identical message number to that just previously acknowledged and processed, it sends back a redundant acknowledgement message to inform the transmitting node that it has lost our previous ackrowledgement. Since we have already processed this message, the originator has presumably lost the target task response message and should do a status check to determine the result of the previous operation. This must be done at the user level since it involves interpreting task specific information.

ERROR CASE 3:

When the originating task message is sent and acknowledged, but the response message from the target processor task has not been received, or has been received incorrectly with recovery attempts unsuccessful, a special error code is returned to the caller. The proper task response is to try and complete a status inquiry of the processor. Successive failures in this situation indicate that some hardware or task related error condition exists.

ERROR CASE 4:

A final set of error conditions exist which are detected only in network ports which are masters (these error conditions in slave ports cause detectable error conditions in the corresponding master port resulting in the proper error correction procedure). These errors are timeout conditions in which a response (message level ACK or task level message response) was expected but did not occur in a specified amount of time (on the order of 4 to 30 seconds). In such a case, either one or more bytes of a message were missed, thus causing the byte count never to be exhausted, or the target processor or link is

down and thus not responding. In such cases, retransmission and recovery are attempted a preset number of times. If unsuccessful, a Timeout or Device Error error code is returned to the caller.

In the cases where message synchronization loss is suspected (timeouts, successive failures of message transmit/ACK sequences, invalid EOM, etc.), the retransmitted message is preceded by a large number of SYN synchronization characters. The number sent is selected to be one greater than the largest message possible. Thus if the target processor is hung waiting for an invalid or obsolete byte count to be exhausted, this series of synchronization characters will in fact exhaust it, and the remaining synchronization characters will be ignored while waiting for the SOM character which signifies the start of new message.

Message Formats

This protocol supports two types of messages, network control and user data. The format of all messages is illustrated below:

SYN HDR.	SOM	CTRL. BYTE 0	MSG. NO.	MSG. COUNT	0-3 CTRL. BYTES	Message Data	CRC1	CRC2	EOM

Where:

Synchronization Header — An indefinite number of SYN characters may precede any message. These are ignored. (Typically one or two is used.)

SOM — The SOM, Start-of-message, character is used to indicate the end of a synchronization header and the beginning of the message prefix and data.

Control Byte — Used to denote the number of additional control bytes in the message as well as the type of control message. In addition, specifies whether or not data is present in the message.

Message No. — The message number is used to detect redundant messages. Numbers are advanced by one whenever a message sequence (at the task level) is initiated.

Message length — This byte holds the length-1 of the data portion of the message in bytes.

Additional	Additional control bytes as are specified by the first control byte.
Message data	The actual data portion of the message. This has no content restrictions other than length. This is the data found in SARRAY in the call to SLINK.
CRC1, CRC2	These two bytes form a 16 bit CRC identical to that used in DECNET. It applies to all elements of the message number, length, control bytes, and data.
EOM	This end of message character is used to denote the end of this packet. It's use is primarily in increasing the speed of error recovery in certain situations.

For stepping motor control processors, network messages fall into two groups:

Control Messages - message acknowledgements, retransmit, and restart requests

Data Messages - arbitrary data up to 256. bytes.

Internal Specifications

The following elements are of interest in attempting to gain an understanding of the internal workings of the SHIVANET Control Protocol support package. They are not intended to be an exhaustive description and specification of the SHIVANET package structure and elements. This description should, however, provide assistance in understanding the assembly language listings which, together with the SHIVANET reference manual, comprise the source document for the support package.

Asynchronous Tasking

All network operations are asynchronous to the issuing task. Once the initial request to send a message is posted, I/O interrupts and timeouts are used to propagate the remaining steps of the message transfer, acknowledgement, task message receipt, and its acknowledgement. To accomplish this, each port has its own status table which includes task status, device driver status, buffer addresses, timeout counters, pointers, and a stack area which is used to keep track of the current task in process on a given port. This stack is managed independently of the normal program stack which is managed through R6 in the PDP-11's.

The unit stack is manipulated in a manner similar to that used in normal stack operations involving the R6 register. Two macros, using the unit stack, have been defined which provide the ability to initiate an AST (Asynchronous Task) type process, return to the system (or background program or other unit AST task), and return to the current AST code when the AST operation has been completed. These are the CALLA and RTA macros. An illustration in section 8.1 of the SHIVANET document (Volume 2) provides an indication of how these and other AST macros are used in the network package. Note that the AST's we are referring to here are internally processed in the network handler package. They are not the AST's of RSX-11/M. However, the concept is similar in both cases.

Asynchronous Task (AST) Scheduling

There are two requirements in the scheduling of the asynchronous tasks which propagate network messages. The first is that, although initiated in some cases directly by an interrupt level driver, CPU interrupts must be enabled in order to assure proper servicing of other interrupt driven devices, including other serial I/O network ports in process. The second requirement is the necessity to inhibit AST's from interrupting themselves. In general, this does not cause problems unless AST's for the same unit are simultaneously initiated. This condition is exemplified by the case where a message is received, an AST process is initiated to handle it, but before the input timeout request for that unit can be cancelled, the timeout occurs, thus causing an attempt to execute the input timeout processor AST.

To handle such cases and to assure execution of AST's in the order in which they occur, a software lockout flag is used to prevent initiating an AST while another is in process. AST's which are attempted to be queued during this time are saved in an AST silo (first-in, first-out circular buffer), and processed immediately upon exit from the currently executing AST. During this processing, CPU interrupts are, of course, enabled.

Timeouts

Timeout errors on the serial line are not detected in slaves; they are processed by the master node. See section 8.3 of the SHIVANET document for a description of master processing of timeouts.

SHIVANET Modules

Several routines comprise the SHIVANET package used in the stepping motor control processors. The following section summarizes the functions and interactions of each module. Further information can be found in the SHIVANET reference manual and in the documentation imbedded in the source listings.

MODULE	ENTRY	FUNCTION
SLVLP	(MAIN)	This is the main program loop in an SMC. A jump instruction at location 40000 (first word of ROM), always points to the beginning of SLVLP. It will initialize the clock, networks and motor routines on startup, then enter the main loop where, when no network message has been received, MTBXEC (execute motor executive) is repeatedly called. If a network command is received (determined by checking flag MSRRDY), a call to BSLAVE is executed to process the message. Upon return from BSLAVE, SRS is called to send the response message (which has been set up by BSLAVE).
	SRS	Send and Receive in Slave - Called from the main loop to send a message response and to post the next read on the network I/O port.
NETINI		Contains various initialization routines:
	IUT	Initialize Unit Table
	IUD	Initialize Unit Devices
	INS	Initialize Network Slave
	PLA	Preset Low core memory Addresses
	IST	Initialize System tables and activate clock processing
RNM	SNM	Send network message (called by SRS)- queues request to SLM
	RNM	Receive network message - (called by SRS) receives message and, since this is a slave, processes device errors internally without passing them on to the caller. (Calls RLM)
SLM	SLM	Send Link Message - (Called by SNM) initiates transmission of a data message and checks the response. Processes message errors and retransmits if necessary, returning an error code if the link is down.

SCM	SCM	Send CRC Message. - (Called by SLM, SMA) This routine initiates transmission of a message, computes the CRC check while the message is being sent and reactivates the driver (if necessary) to assure transmission of the CRC and EOM in the message trailer.
	CMC	Compute Message CRC. This routine is called to compute the CRC for a series of arbitrary bytes.
RLM	RLM	Receive Link Message. (Called by RNM) This is the counterpart to SLM. RLM issues read requests, checks for message errors, and acknowledges correct messages to the master. RLM returns to RNM with a successful status upon receipt of a valid message.
GLM	*GLM*	*Get Link Message - Called by RLM and RMA. Receives a* link message and computes the CRC for it.
RMA	RMA	Receive Message Acknowledgement - Called by SLM. This routine accepts a message acknowledgement, checks for errors, and if valid, returns the type (ack, nack, redundant ack) to the caller.
	SMA	Send message Acknowledgement - Called by RLM. Constructs and sends a network ACK for the last received message.
	SMN	Send Message Negative Acknowledgement - Called by RLM.
	SMR	*Send Message Redundant Acknowledgement - Called by RLM*
PPF	PPU	Process Power Up - Executed via a trap at power up time.
	PPF	Process Power Fail - Executed after power failure or after incorrect network initiated boot sequence by the ROM bootstrap code.
CRC16	$CRC16	Compute CRC (Cyclic Redundancy Check) for a single byte. Called by CMC.
PUSHPL	PUSH	Save registers and return with R0 as the parameter pointer. Used by the stepping motor subroutines via the PUSH macro.
RETSUB	RETSUB	Restore registers as saved by PUSH.
GNE	GNE	Get Network Errors - Called by BSLAVE. Retrieves serial network related error counters.
SLVTRP	PTI	*Process trap Instruction - Called by PUSH and RETSUB* macros. Reads the parameter on the trap instruction and branches to the PUSH or RETSUB routines as appropriate.

NETSY — A collection of routines which handle scheduling and I/O related functions. Only the principal entries are listed below:

- ASTO — Process input and output AST's (Asynchronous tasks) which are initiated when a message has been completely received or sent.

- TIMI, TIMO — Process input and output timeouts. These occur when there is a device error or loss of synchronization somewhere in the net. Processing initiates at these entries which cancel the I/O requests and return to the suspended procedure which initiated the I/O.

- RETAST — Called to exit from the AST state to normal processing.

- WIO — Wait I/O Complete - Called after initiating an I/O operation if the initiator has no further function to perform until I/O is done. The unit stack pointer and task address are saved and a return to the background system is effected.

- WLB — Write Logical Block - Called by SCM. This routine activates the I/O driver (NXOT) on the appropriate port.

- WSH — Write Synchronization Header - Called by SCM. This activates the output driver to transmit a syncronization header in cases where loss of synchronization is suspected. The synchronization header consists of 256 SYN characters.

- RLB — Read logical Block - Called each time as AST is queued or a previous AST process exists. This routine extracts the next AST process from the AST queue and executes it. However, PQA exits immediately if an AST procedure is already active.

- QAT — Queue an AST Procedure - called when I/O operation completes or a timeout occurs. Places an AST procedure and its unit table pointer in the AST silo.

- PTO — Process timeouts - Called once each second. This routine scans the unit tables for active timers on active I/O ports. Times are decremented and if one goes to zero, the corresponding input or output timeout processor is invoked.

	PCI	Process Clock Interrupt - Called 60 times each second. PCI calls PTO once for every 60 clock ticks (one second). It also invokes any other clock processor which existed in the system before network initialization. In this case, PANINT is executed to process panel switches.
NTDRV	NXIN	Serial link input driver. This entry will accept an entire logical message (including header, CRC, etc.) under interrupt control. Message content is placed in the caller's specified buffer. The message envelope (header-CRC) is placed in the unit table.
	NXOT	Serial link output driver. This entry transmits an entire logical message including synchronizations, header, and CRC under interrupt control.

SHIVANET TABLES

TABLE 7

Unit Table:

There is one unit table for each I/O port. In the SMC's there is only one such port and its unit table is as follows:

ORDINAL	NAME	CONTENTS
0	UNSTA	Unit status (0=busy,≠ free)
2	UNNUM	Unit number
4	UNRSA	Receive status array word 1
6	UNRS2	Receive status array word 2 (byte count received)
10	UNTSA	Transmit status array word 1
12	UNTS2	Transmit status array word 2
14	UNRFA	Receive buffer address
16	UNRSZ	Receive buffer size
20	UNTFA	Transmit buffer address
22	UNTSZ	Transmit buffer size
24	UNWTC	Transmit working counter for control bytes
25	UNTCB	Transmit control byte
26	UNTMN	Transmit message number
27	UNTMC	Byte count-1 of data
30	UNTC1-TC3	Transmit control bytes in header (optional)
33	UNDST	Device status byte
34	UNTHA	Working header address during transmission
36	UNLRN	Last received message number
40	UNTIM	Number of receive timeouts
42	UNIOE	Number of I/O errors
44	UNMEC	Number of CRC errors
46	UNTEC	Total error count
50	UNUSP	Unit stack pointer save area
52	UNWIO	Address of I/O wait processor
54	UNRST	Receive status
55	UNTST	Transmit status
57	UNRMN	Last acked message number
60	UNPRT	Receive timeout value
61	UNPTT	Transmit timeout value
62	UNNRM	Number of received messages
64	UNRSB	Receive driver status
65	UNTSB	Transmit driver status
66	UNSYC	Sync character
67	UNNSC	Number of sync characters in normal message
70	UNITT	Input timeout wait in seconds
71	UNOTT	Output timeout wait in seconds
72	UNITP	Input timeout processor
74	UNOTP	Output timeout processor
76	UNTXA	Transmit driver working address

Unit Table (Continued):

ORDINAL	NAME	CONTENTS
100	UNTXC	Transmit driver working byte count
102	UNRCA	Receive driver working address
104	UNRCC	Receive driver working byte count
106	UNTSC	Working sync count
110	UNSRA	Read status register address, this device
112	UNTAS	Current transmit AST address
114	UNRAS	Current receive AST address
116	UNWRC	Working receive header byte count
117	UNRCB	Received control byte
120	UNIMN	Received message number
121	UNRBC	Received data byte count
122	UNRC1-RC3	Optional received control bytes
126	UNRHA	Working receive header address
130	UNTCR	Transmit CRC
132	UNRCR	Received CRC
134	UNREM	Received EOM
		Unit stack area

AST TABLE

The Asynchronous Task Table is used to queue asynchronous tasks for sequential execution. It is a silo implemented as a circular buffer at "ASTQ." Each entry includes two words:

1. AST task address
2. Unit table pointer for this AST

SMC Startup

A special ROM coded version of the standard DEC REV-11 boot code is used in SMC's to initiate their operation. Using this code, the following steps are performed:

1. The CPU is strapped (on the LSI-11) to execute the instruction at location 173000. This then is the first word of the instruction sequence to handle initialization operations.

2. For a period of approximately 200 milliseconds, the incoming data on the serial link card at CSR address 176520 is examined for a valid restart request message from the network. If a valid request is found, an initial message received from this network link is loaded into core and executed. This is termed a down line load which, in the case of an SMC processor, merely restarts the ROM resident program.

3. If no valid message to restart is detected, location 26 is examined. This is the power up routine CPU status. If set to 357, this is assumed to be a valid power up after power down situation and the power-up processor is executed normally, after resetting location 26 to 340.

4. If location 26 is not 357, then this is assumed to be an initial startup or a case of the front panel initialize button being depressed. In this case, the first word of ROM, found dynamically, is executed. In a SMC, this is at location 40000 and starts the SMC program. If no ROM is found before address 160000, a bootstrap operation from floppy disc unit 0 is initiated (not used in SMC's).

The remote restart sequence is initiated by a break command on the serial link which forces a power failure condition through a connection to the power fail detect/clock card used in the SHIVA LSI-11's. Power failure eventually drops the LSI-11 bus signal DCOK L which clears the UART and thus immediately removes the break condition causing a power-up sequence to initiate. This starts up the code at 173000 which will check for the validity of the restart request. Since the normal power fail sequence is executed, it is possible to restart without reinitialization in those cases where superfluous break codes on the serial link incorrectly initiated the restart sequence. However, any RAM which clears as a result of negating DCOK L must be properly handled in this instance.

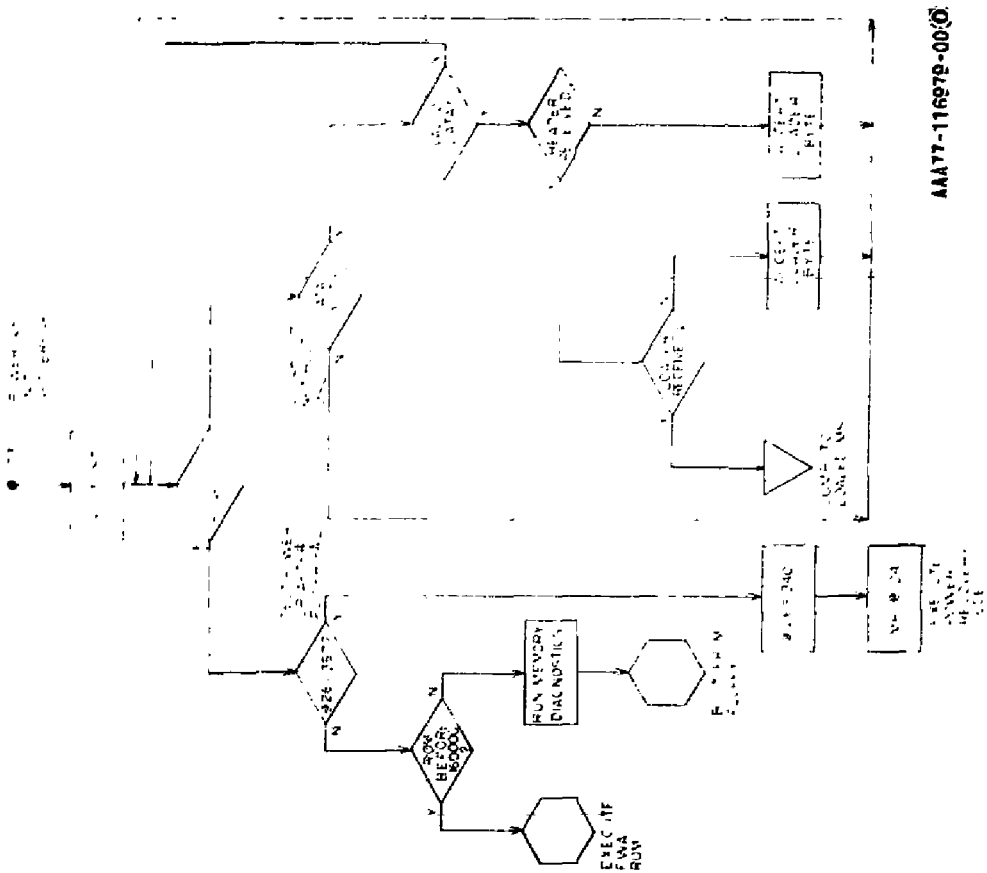


Figure 12

Flow Diagram, Modified REV-11 Power-up Sequence

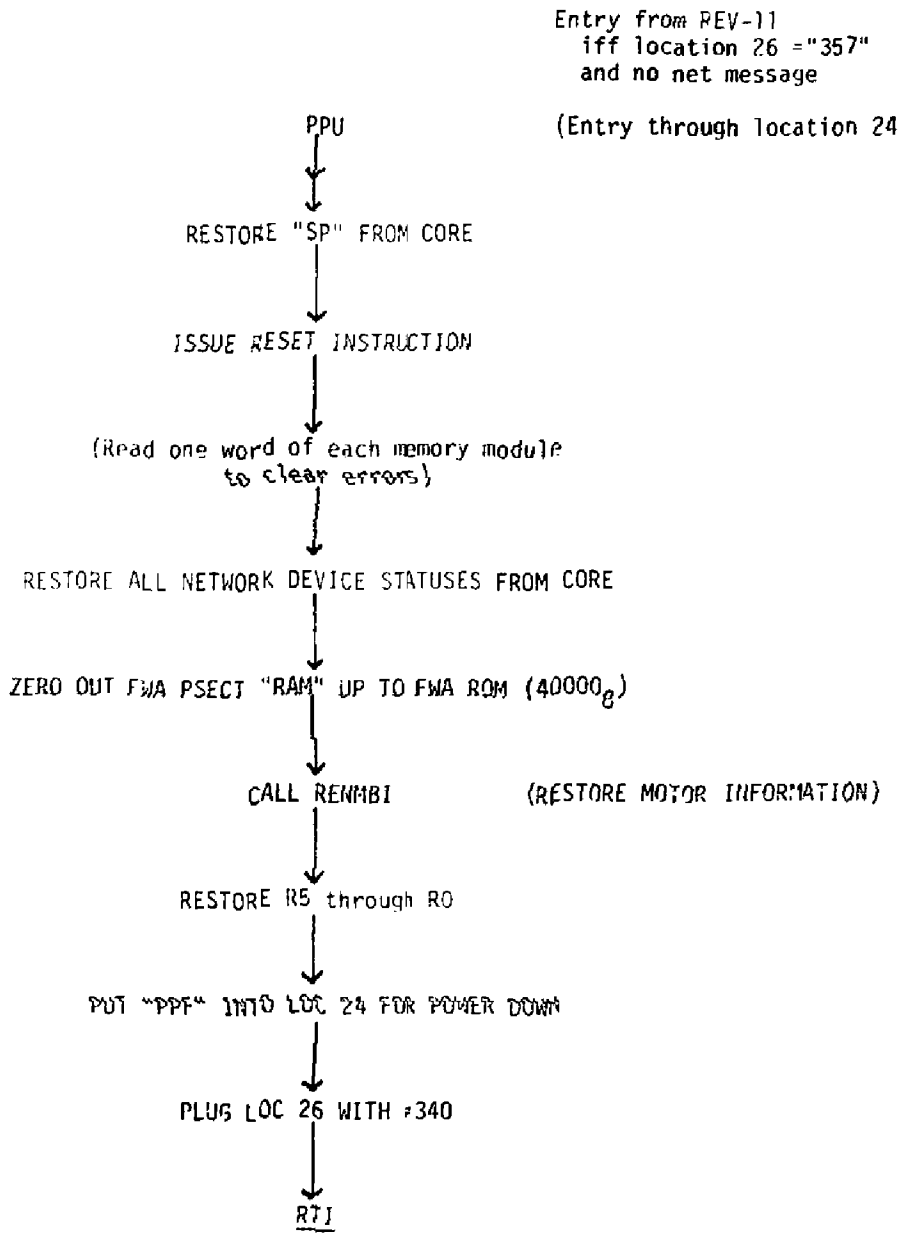


Figure 13. Power Up Processing - SMC Processor

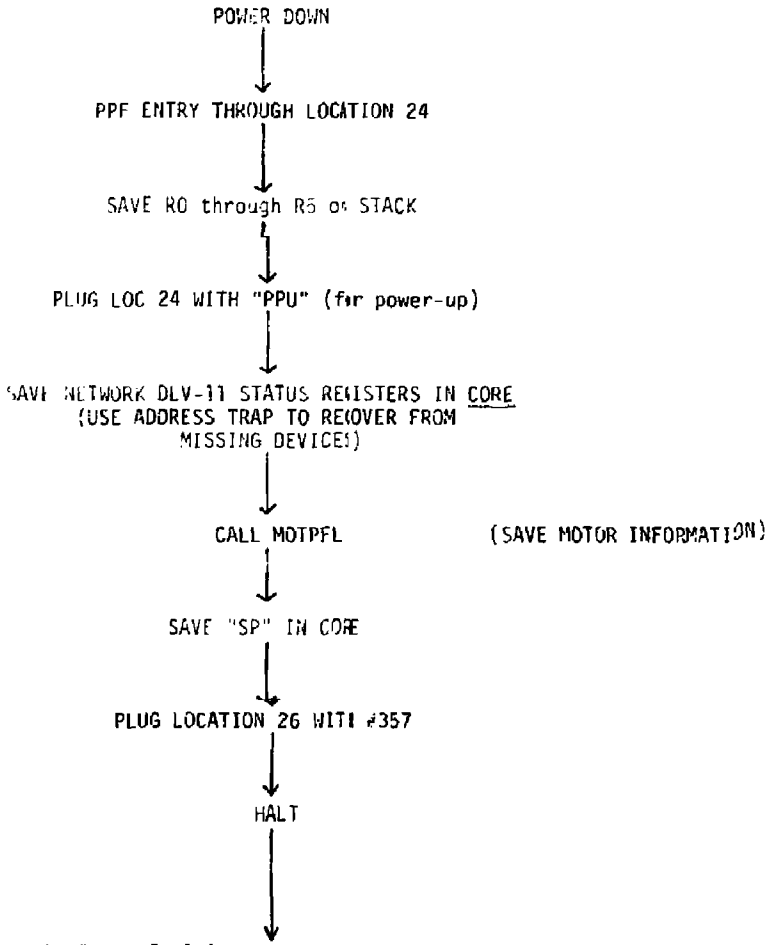
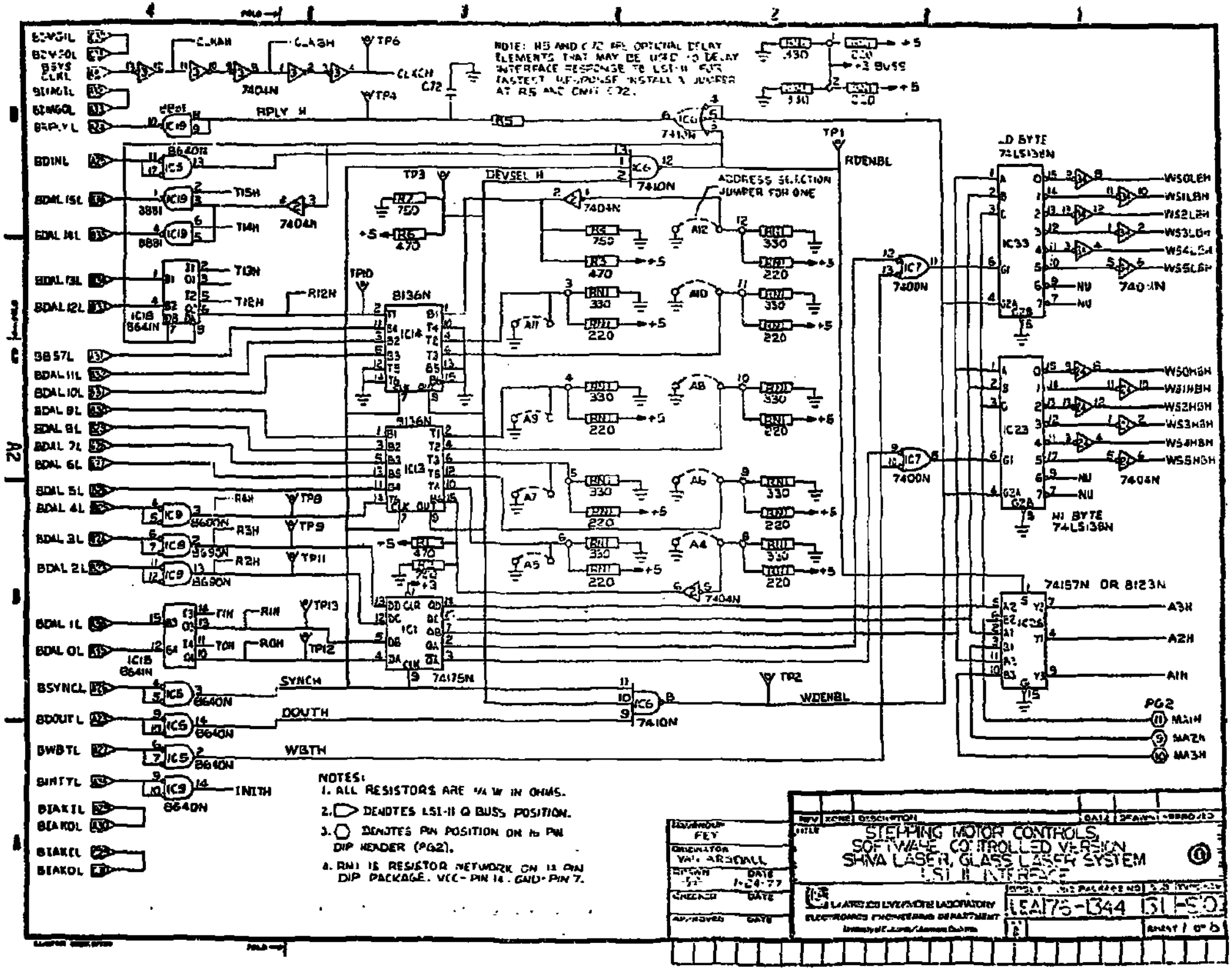


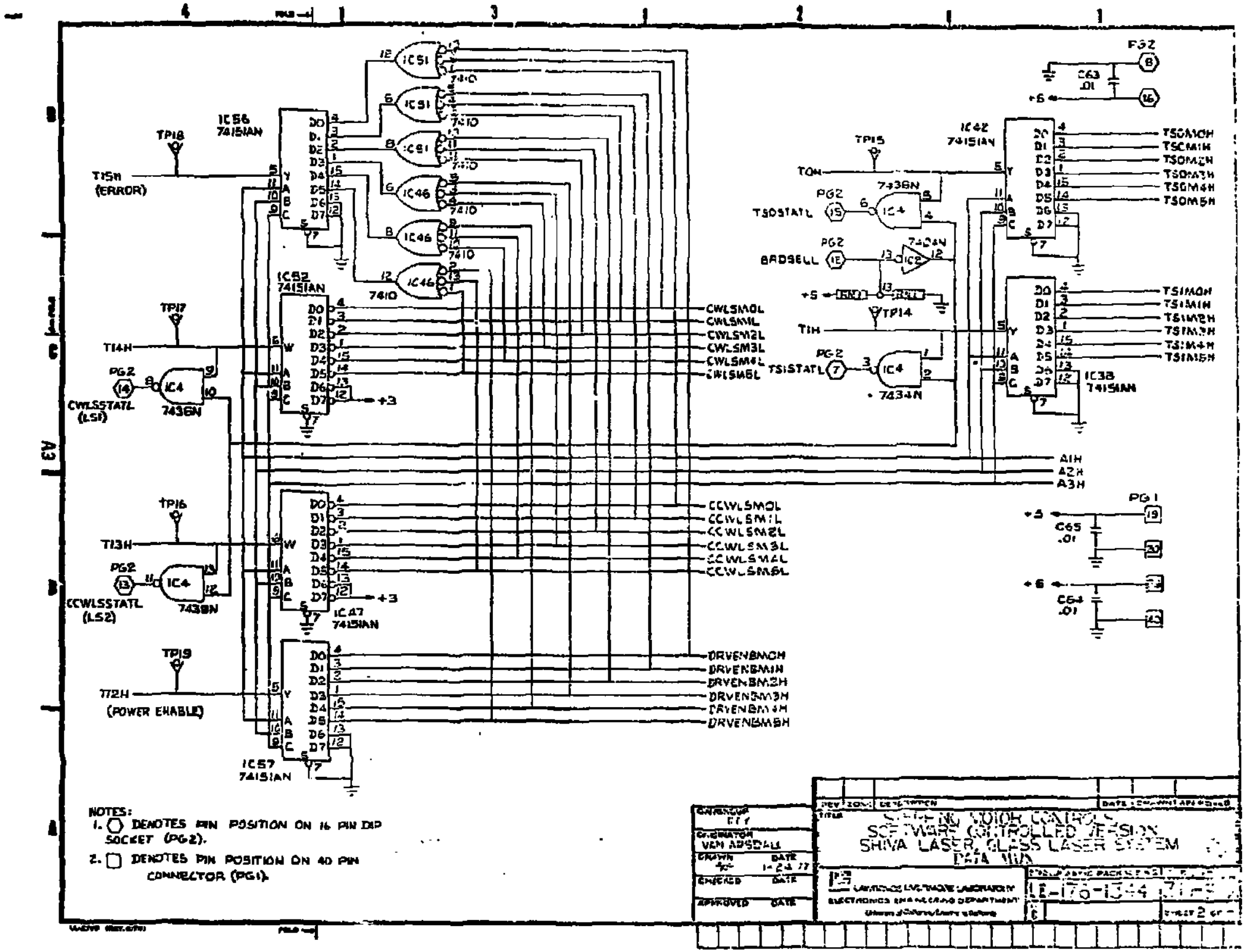
Figure 14. Power Fail Processing - SMC Processor

APPENDIX A

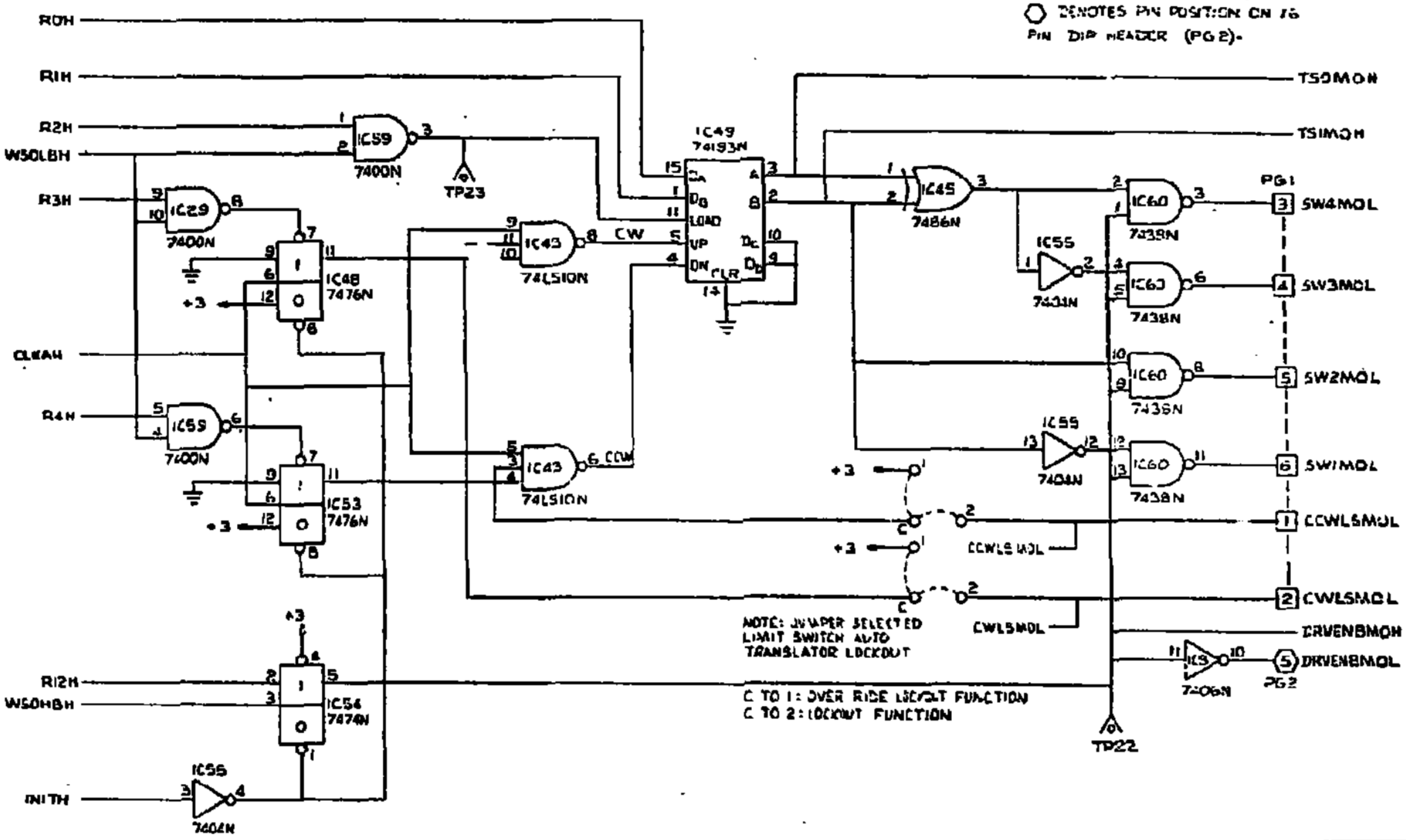
HARDWARE SCHEMATICS

A1


NOTES:
1. ALL RESISTORS ARE 1/4 W IN OHMS.
2. DENOTES LSI-11 Q BUS POSITION.
3. DENOTES PIN POSITION ON 16 PIN DIP HEADER (PG2).
4. RN1 IS RESISTOR NETWORK ON 13 PIN DIP PACKAGE. VCC - PIN 14, GND - PIN 7.
STEPPING MOTOR CONTROLS, SOFTWARE CONTROLLED VERSION
SHIVA LASER, GLASS LASER SYSTEM
LSI 11 INTERFACE
LEA75-1344
LSI-SO
ADDRESS SELECTION JUMPER FOR ONE
LO BYTE 74LS138N
HI BYTE 74LS138N
74157N OR 8123N
PG2


IC56 74151AN
IC52 74151AN
IC47 74151AN
IC57 74151AN
IC42 74151AN
IC38 74151AN
IC51 7410
IC46 7410
IC4 7436N
IC4 7434N
IC2 7404N
TP18
T15H (ERROR)
TP17
T14H
CWLSSTATL (LS1)
TP16
T13H
CCWLSSTATL (LS2)
TP19
T12H (POWER ENABLE)
TP15
T0H
TSOSTATL
BADSELL
TP14
T1H
TS1STATL
CWLSM0L
CWLSM1L
CWLSM2L
CWLSM3L
CWLSM4L
CWLSM5L
CCWLSM0L
CCWLSM1L
CCWLSM2L
CCWLSM3L
CCWLSM4L
CCWLSM5L
DRVENBM0H
DRVENBM1H
DRVENBM2H
DRVENBM3H
DRVENBM4H
DRVENBM5H
TSOM0H
TSOM1H
TSOM2H
TSOM3H
TSOM4H
TSOM5H
TS1M0H
TS1M1H
TS1M2H
TS1M3H
TS1M4H
TS1M5H
A1H
A2H
A3H
PG2
PG1
C63 .01
C65 .01
C64 .01
+3
+5
+6
NOTES:
1. ○ DENOTES PIN POSITION ON 16 PIN DIP SOCKET (PG2).
2. □ DENOTES PIN POSITION ON 40 PIN CONNECTOR (PG1).
GENERATOR VEN APSDALL
DRAWN DATE 1-24-77
CHECKED DATE
APPROVED DATE
SCANNING MOTOR CONTROL
SOFTWARE CONTROLLED VERSION
SHIVA LASER, GLASS LASER SYSTEM
DATA MVA
LAWRENCE LIVERMORE LABORATORY
ELECTRONICS ENGINEERING DEPARTMENT
LE-176-1344
SHEET 2 OF

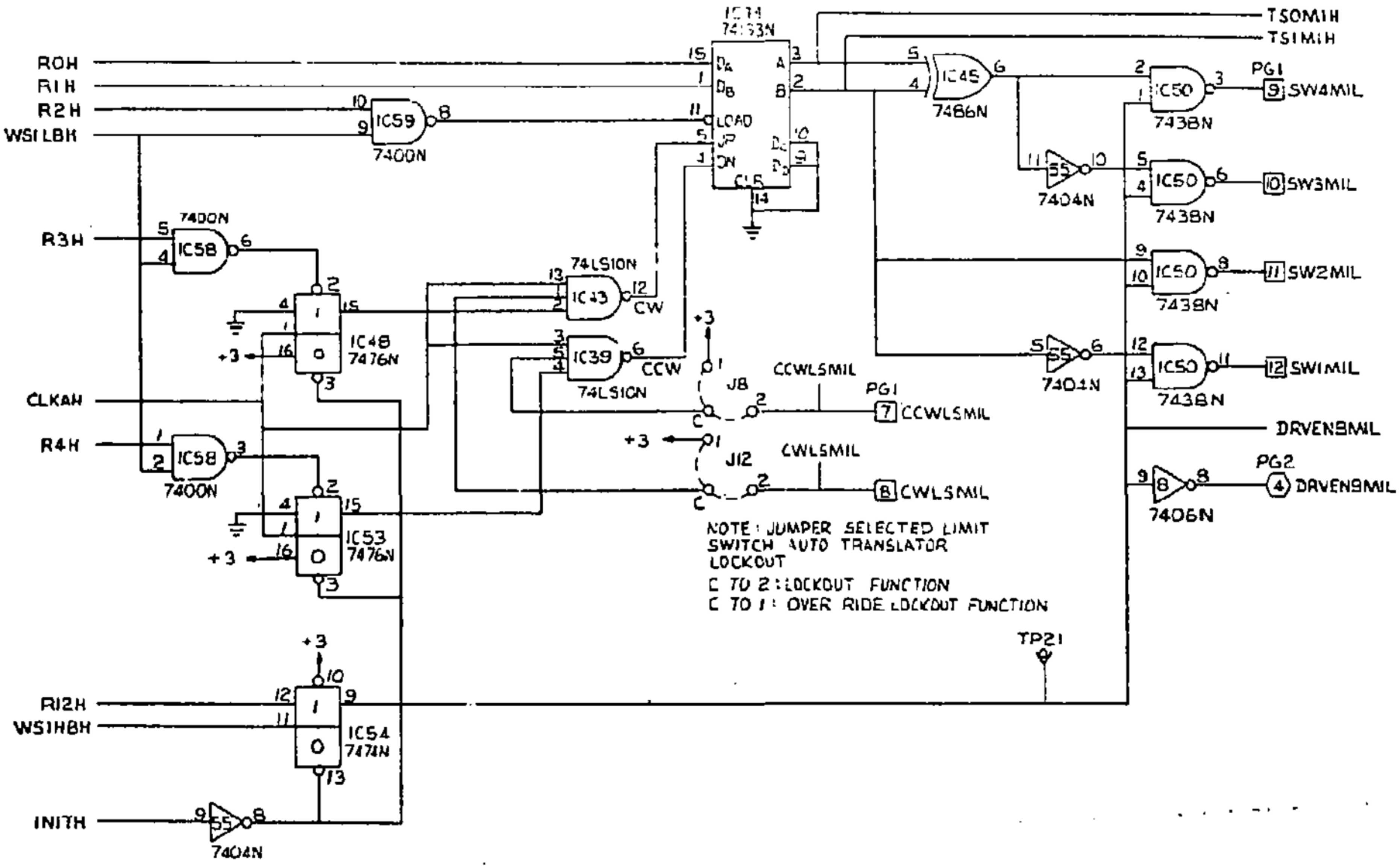
NOTE: ☐ DENOTES PIN POSITION ON 40 PIN CONNECTOR (PG1).
⬡ DENOTES PIN POSITION ON 16 PIN DIP HEADER (PG2).



NOTE: JUMPER SELECTED LIMIT SWITCH AUTO TRANSLATOR LOCKOUT

C TO 1: OVER RIDE LOCKOUT FUNCTION
C TO 2: LOCKOUT FUNCTION

ENGR/GROUP FEY	STEPPING MOTOR CONTROLS, SOFTWARE CONTROLLED VERSION, SHIVA LASER, GLASS LASER SYSTEM MOTOR ϕ TRANSLATOR [illegible]
ORIGINATOR VAN ARSDALL	
DRAWN DATE 1-24-77	LAWRENCE LIVERMORE LABORATORY ELECTRONICS ENGINEERING DEPARTMENT University of California/Livermore, California
CHECKED DATE	LEA 76-1344 31-SO
APPROVED DATE	SHEET 2 OF 3



NOTE: JUMPER SELECTED LIMIT SWITCH AUTO TRANSLATOR LOCKOUT
C TO 2: LOCKOUT FUNCTION
C TO 1: OVER RIDE LOCKOUT FUNCTION

NOTES:

1. ☐ DENOTES PIN POSITION ON 40 PIN CONNECTOR (PG1).
2. ⬡ DENOTES PIN POSITION ON 16 PIN DIP HEADER (PG2).

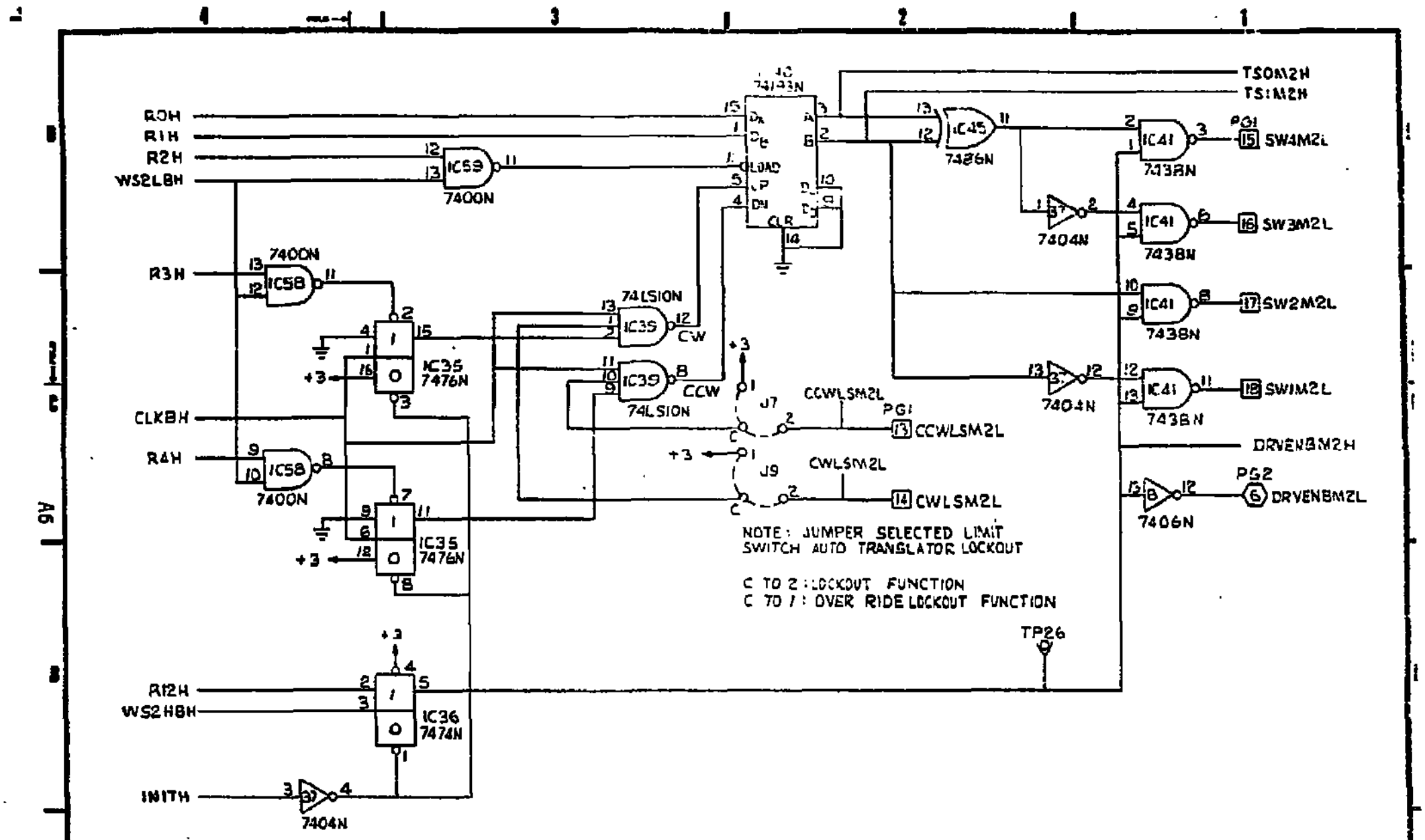
REV	ZONE	DESCRIPTION	DATE	DRAWN	APPROVED

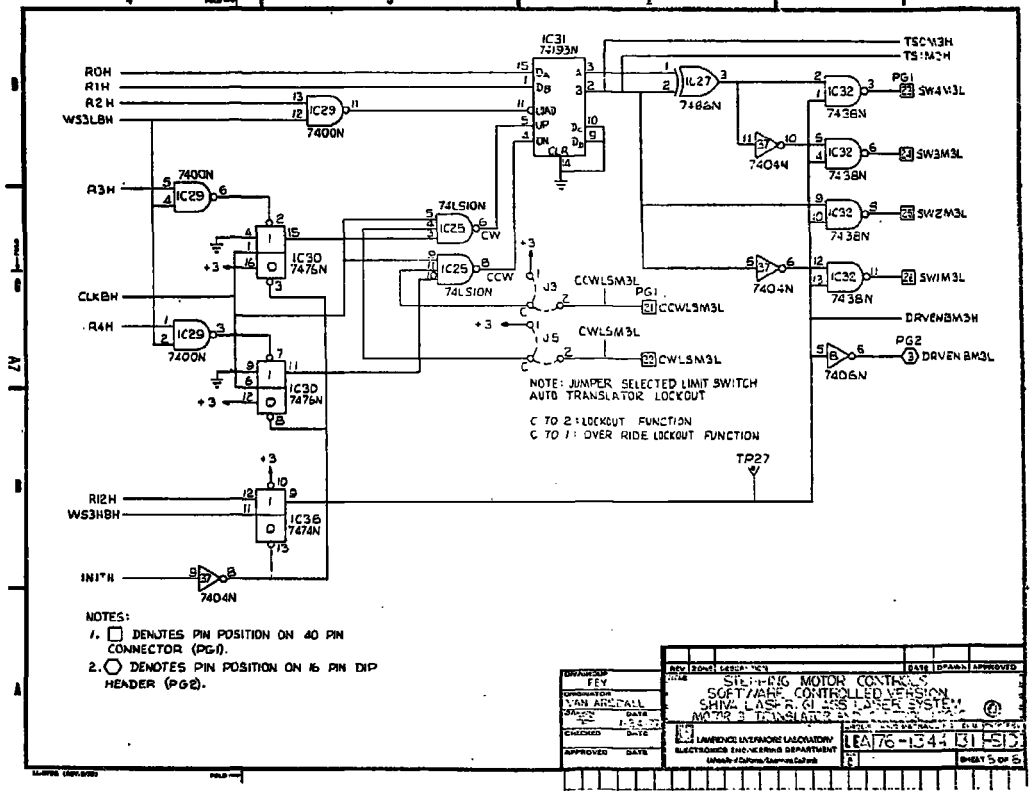
DIV/GROUP FFY
ORIGINATOR VAN ARSDALL
DRAWN DATE 1-24-77
CHECKED DATE
APPROVED DATE

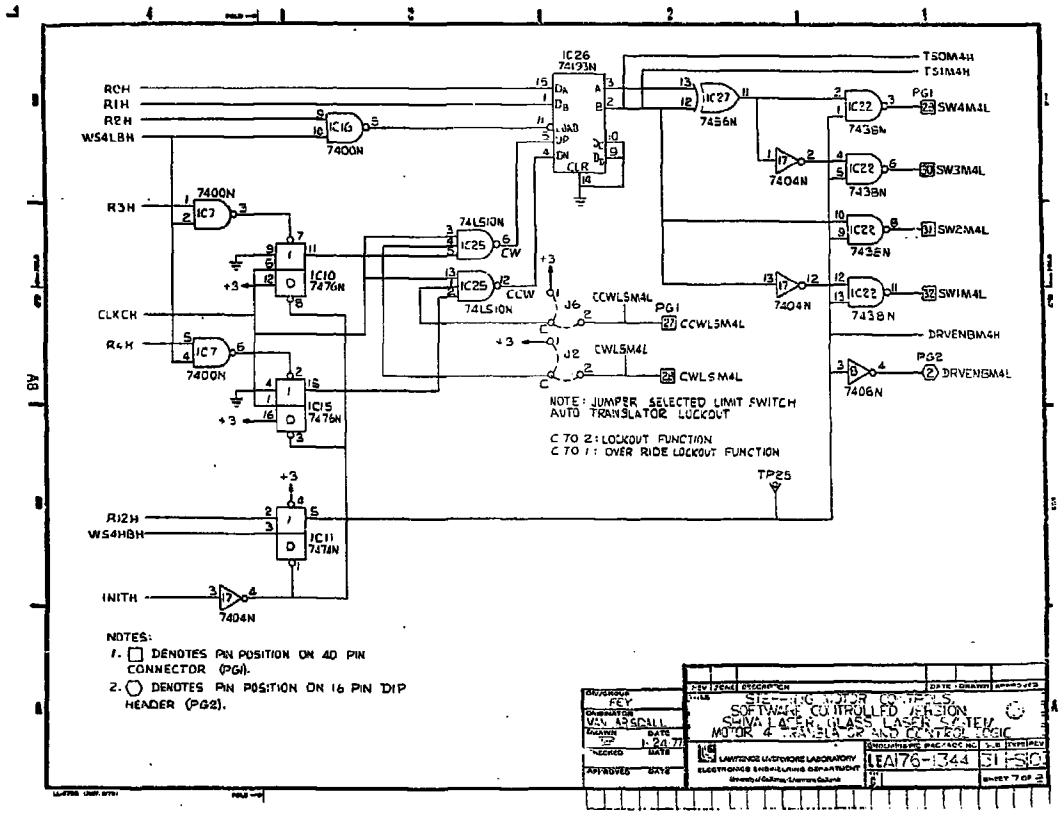
TITLE
STEPPING MOTOR CONTROLS,
SOFTWARE CONTROLLED VERSION
SHIVA LASER, GLASS LASER SYSTEM
MOTOR 1 TRANSLATOR AND CONTROL LOGIC

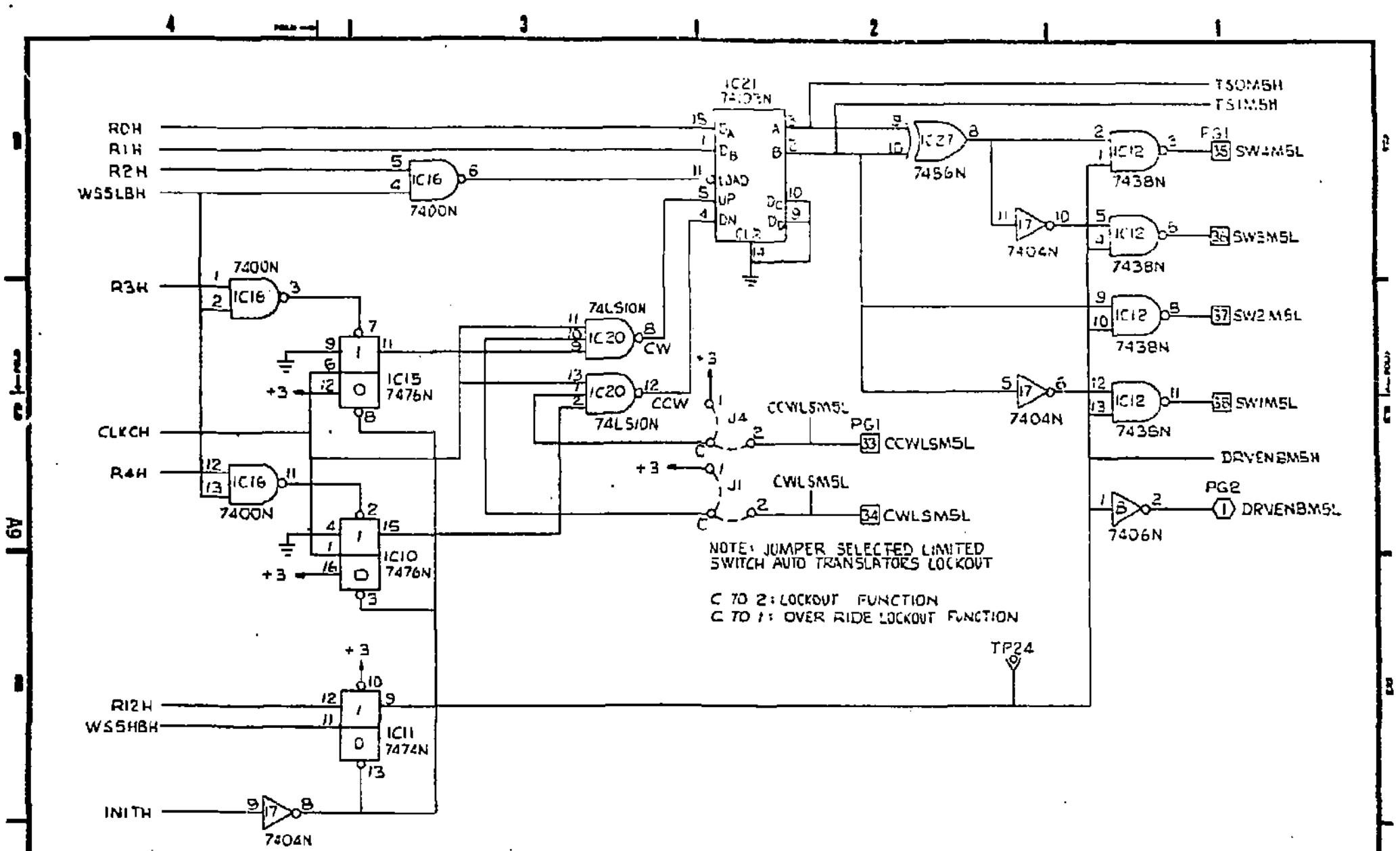
LAWRENCE LIVERMORE LABORATORY
ELECTRONICS ENGINEERING DEPARTMENT
University of California, Livermore, California

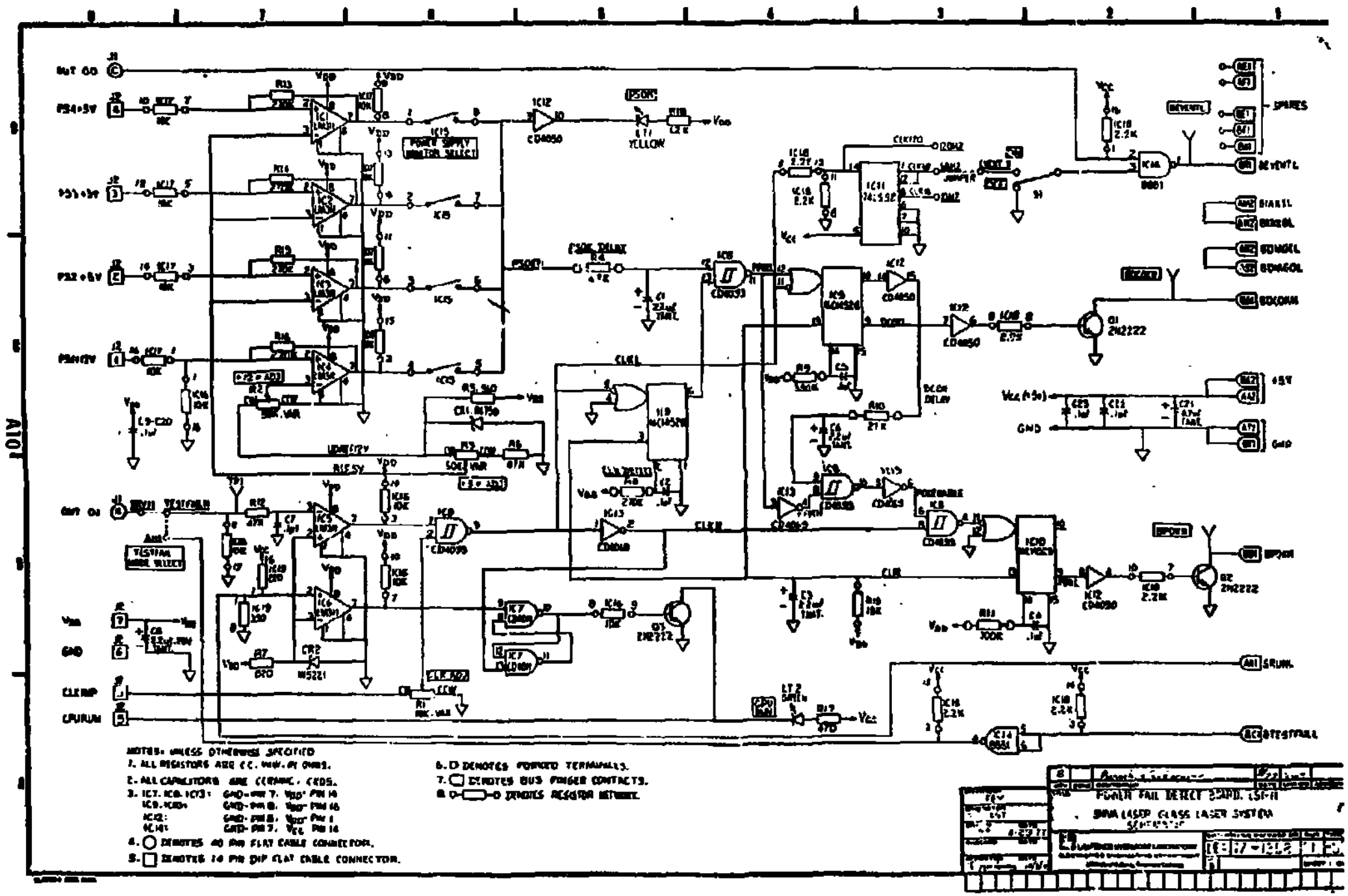
LEA 76-1344 31-S10
SHEET 4 OF 8

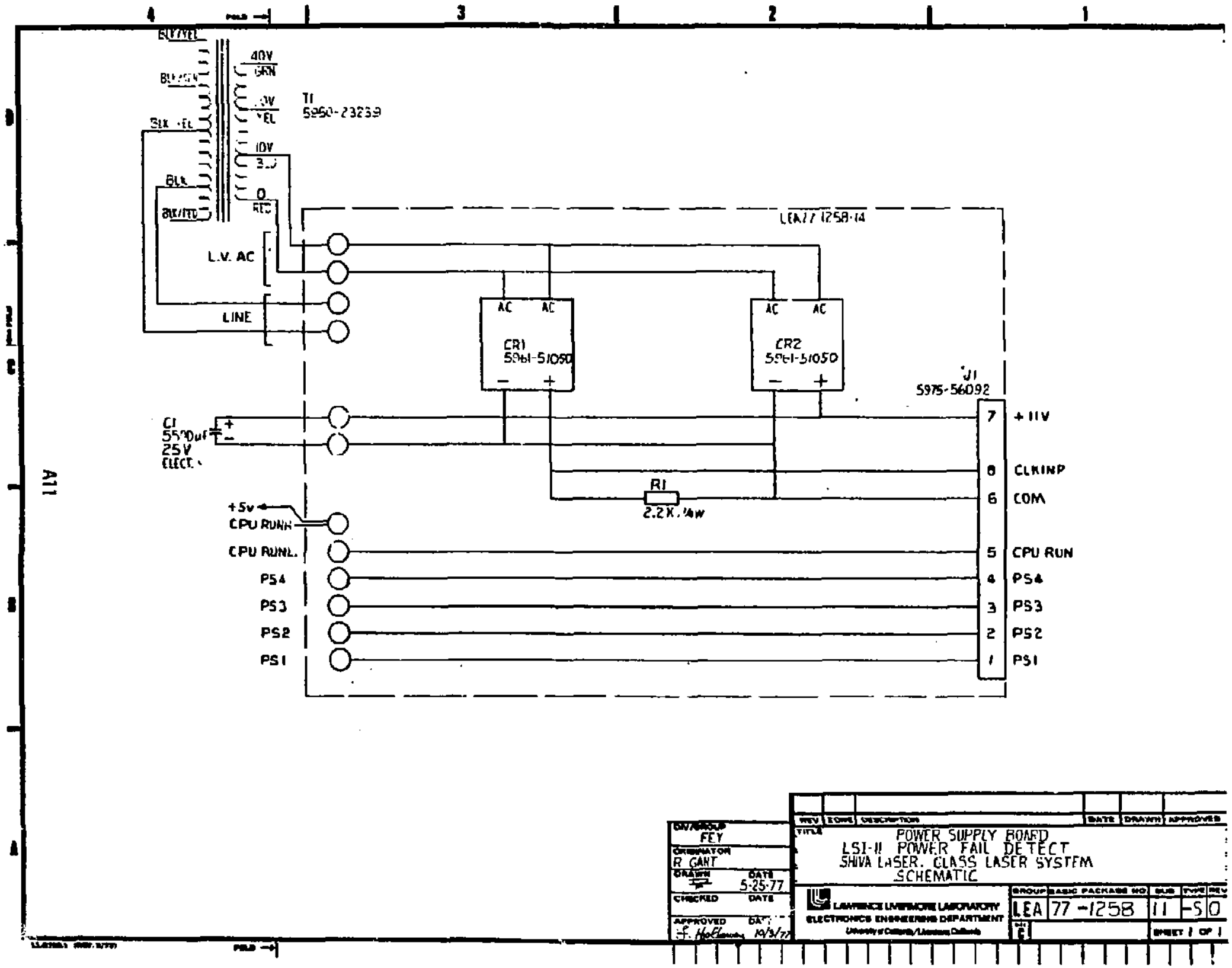

R0H
R1H
R2H
WS2LBH
R3H
CLKBH
R4H
R12H
WS2HBH
INITH
IC59
7400N
IC58
7400N
IC35
7476N
IC36
7474N
7404N
74LS10N
IC39
74LS10N
CW
CCW
IC45
7486N
IC41
7438N
7406N
TS0M2H
TS1M2H
SW4M2L
SW3M2L
SW2M2L
SW1M2L
DRVENBM2H
DRVENBM2L
CCWLSM2L
CWLSM2L
J7
J9
PG1
PG2
TP26
NOTE: JUMPER SELECTED LIMIT SWITCH AUTO TRANSLATOR LOCKOUT
C TO 2: LOCKOUT FUNCTION
C TO 1: OVER RIDE LOCKOUT FUNCTION
NOTES:
1. ☐ DENOTES PIN POSITION ON 40 PIN CONNECTOR (PG1).
2. ⬡ DENOTES PIN POSITION ON 16 PIN DIP HEADER (PG2).
FEY
ARSDALL
1-24-77
STEPPING MOTOR CONTROLS.
SOFTWARE CONTROLLED VERSION
SHIVA LASER GLASS LASER SYSTEM
MOTOR 2 TRANSLATOR
LAWRENCE LIVERMORE LABORATORY
ELECTRONICS ENGINEERING DEPARTMENT
LEA76-1344
SHEET 5 OF 6

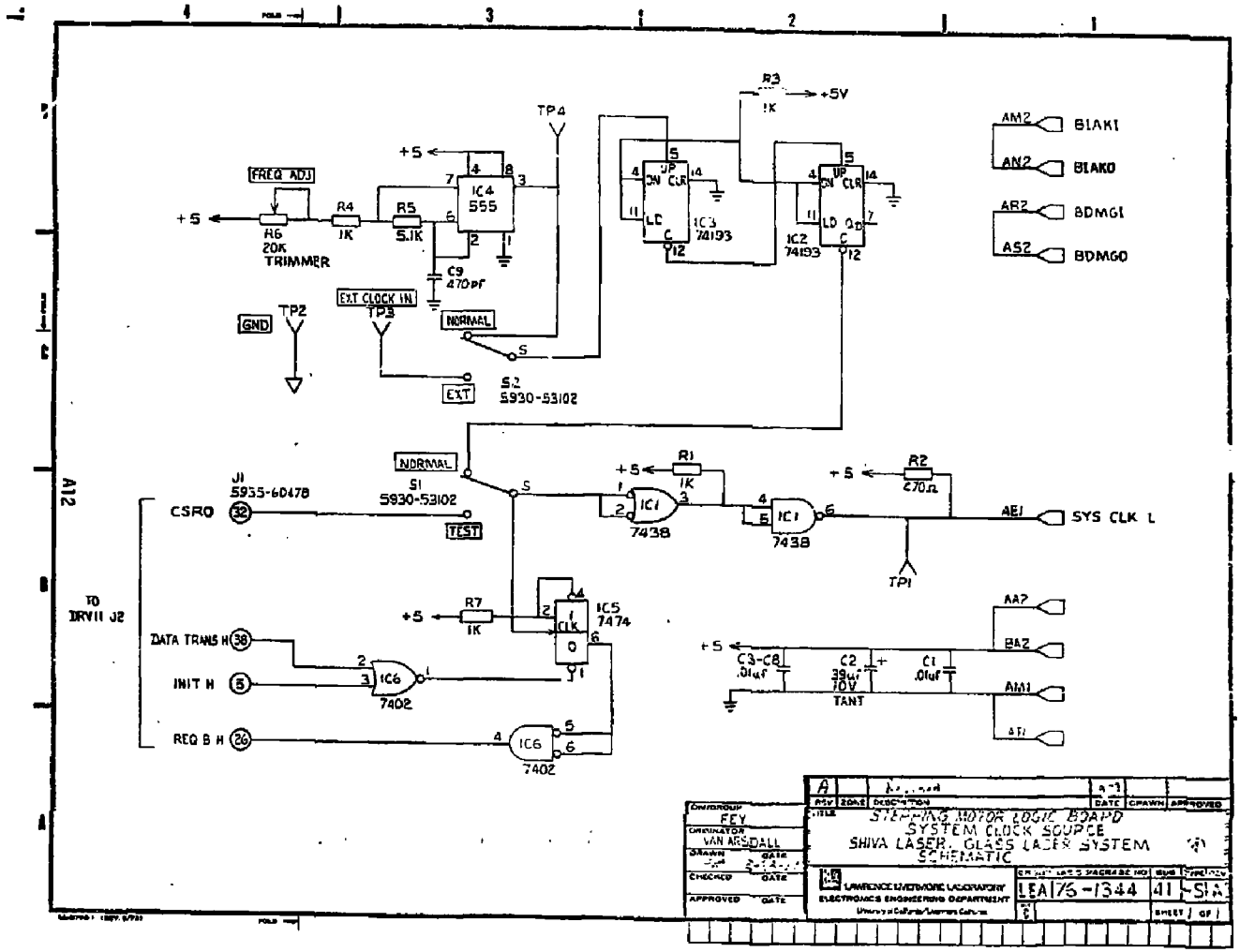

R0H
R1H
R2H
WS3LBH
R3H
CLKBH
R4H
RI2H
WS3HBH
INITH
IC31 74193N
IC29 7400N
IC30 7476N
IC36 7474N
7404N
IC27 7486N
IC32 7438N
IC25 74LS10N
CW
CCW
+3
U3
U5
CCWLS M3L
CWLS M3L
PG1
TSCW3H
TSCC3H
SW4M3L
SW3M3L
SW2M3L
SW1M3L
DRVENBM3H
PG2
DRVEN BM3L
TP27
NOTE: JUMPER SELECTED LIMIT SWITCH AUTO TRANSLATOR LOCKOUT
C TO 2: LOCKOUT FUNCTION
C TO 1: OVER RIDE LOCKOUT FUNCTION
NOTES:
1. ☐ DENOTES PIN POSITION ON 40 PIN CONNECTOR (PG1).
2. ○ DENOTES PIN POSITION ON 16 PIN DIP HEADER (PG2).
DRAWN FEY
DESIGNED IVAN ARSDALL
CHECKED
APPROVED
STEPPING MOTOR CONTROL SOFTWARE CONTROLLED VERSION SHIVA LASER GLASS LASER SYSTEM MOTOR TRANSLATOR
LAWRENCE LIVERMORE LABORATORY
ELECTRONICS ENGINEERING DEPARTMENT
LEA 76-1344 B1-S13
SHEET 5 OF 6

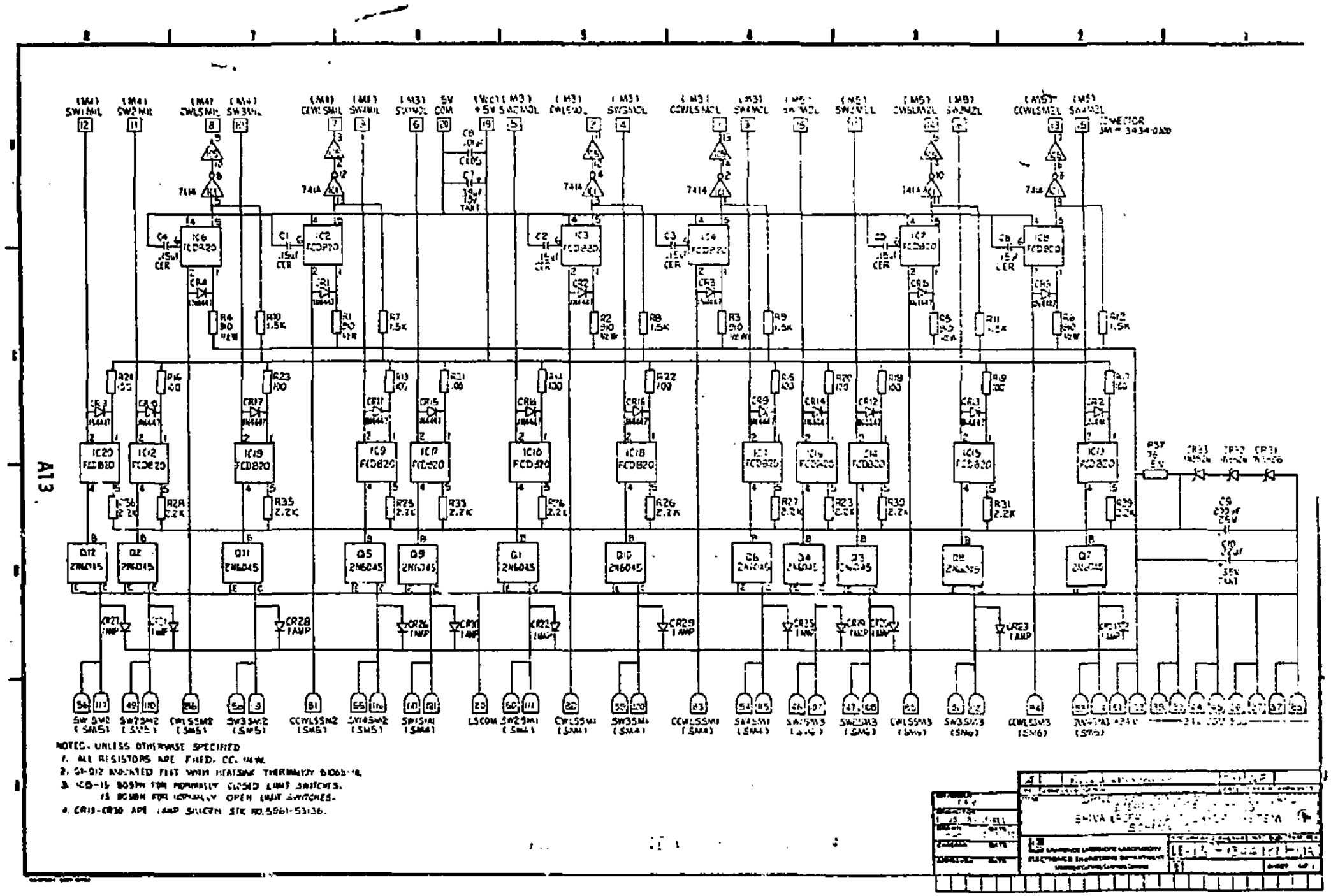

RCH
RIH
R2H
WS4LBH
IC16
7400N
IC26
75193N
T50M4H
T51M4H
IC27
7456N
IC22
7438N
PG1
25 SW4M4L
30 SW3M4L
31 SW2M4L
32 SWIM4L
7404N
R3H
IC7
7400N
74LS10N
IC25
CW
CCW
IC10
7476N
CLKCH
R4H
IC15
7476N
J6
CCWLSM4L
27 CCWLSM4L
J2
CWLSM4L
28 CWLSM4L
DRVENBM4H
PG2
2 DRVENBM4L
7406N
NOTE: JUMPER SELECTED LIMIT SWITCH
AUTO TRANSLATOR LOCKOUT
C TO 2: LOCKOUT FUNCTION
C TO 1: OVER RIDE LOCKOUT FUNCTION
TP25
R12H
WS4HBH
IC11
7474N
INITH
NOTES:
1. ☐ DENOTES PIN POSITION ON 40 PIN CONNECTOR (PG1).
2. ○ DENOTES PIN POSITION ON 16 PIN DIP HEADER (PG2).
DESIGNED FEY
DRAWN
CHECKED
APPROVED
DATE 1-24-77
SOFTWARE CONTROLLED VERSION
SCIVA LASER GLASS LASER SYSTEM
MOTOR 4 INTERFACE AND CONTROL LOGIC
LAWRENCE LIVERMORE LABORATORY
ELECTRONICS ENGINEERING DEPARTMENT
LEA76-1344
SHEET 7 OF 9

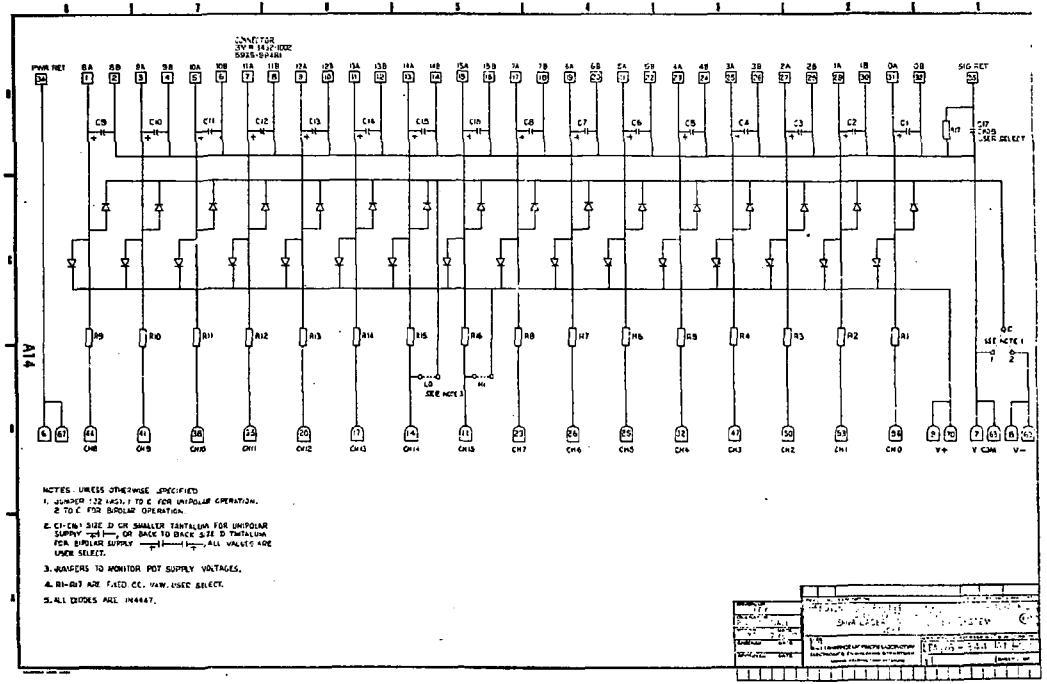

R0H
R1H
R2H
WSSLBH
R3H
CLKCH
R4H
R12H
WSSHBH
INITH
IC16
7400N
IC15
7476N
IC10
7476N
IC11
7474N
7404N
IC21
74LS10N
IC20
74LS10N
CW
CCW
J4
J1
CCWLSMSL
CWLSMSL
PG1
PG2
IC27
7456N
IC12
7438N
TSOMSH
TS1MSH
SW4MSL
SW3MSL
SW2MSL
SW1MSL
DRVENBMSH
DRVENBMSL
7406N
TP24
NOTES: JUMPER SELECTED LIMITED SWITCH AUTO TRANSLATORS LOCKOUT
C TO 2: LOCKOUT FUNCTION
C TO 1: OVER RIDE LOCKOUT FUNCTION
NOTES:
1. ☐ DENOTES PIN POSITION ON 40 PIN CONNECTOR (PG1).
2. ◯ DENOTES PIN POSITION ON 16 PIN DIP HEADER (PG2).
LAWRENCE LIVERMORE LABORATORY
ELECTRONICS ENGINEERING DEPARTMENT
LEA76-1344


A101
NOTES: UNLESS OTHERWISE SPECIFIED
1. ALL RESISTORS ARE CC. MW. PI OHMS.
2. ALL CAPACITORS ARE CERAMIC. CKDS.
3. IC1, IC2, IC3: GND-PIN 7, VDD PIN 14
IC4, IC5: GND-PIN 8, VDD PIN 16
IC12: GND-PIN 8, VDD PIN 1
IC14: GND-PIN 7, VCC PIN 14
4. ○ DENOTES 40 PIN FLAT CABLE CONNECTOR.
5. □ DENOTES 14 PIN DIP FLAT CABLE CONNECTOR.
6. ⊡ DENOTES POINTED TERMINALS.
7. ▭ DENOTES BUS FINGER CONTACTS.
8. ▭ DENOTES RESISTOR NETWORK.
POWER FAIL DETECT BOARD, 151-11
DMA LASER GLASS LASER SYSTEM


T1
5950-23239
40V
GRN
10V
YEL
10V
0
RED
BLK
BLK/RED
L.V. AC
LINE
LEA77 1258-14
AC
AC
CR1
5961-51050
CR2
5961-51050
J1
5975-56092
C1
25V
ELECT.
R1
2.2K 1/4W
+5V
CPU RUN
CPU RUN.
PS4
PS3
PS2
PS1
7 +11V
8 CLKINP
6 COM
5 CPU RUN
4 PS4
3 PS3
2 PS2
1 PS1
A11
FEY
R GANT
5-25-77
POWER SUPPLY BOARD
LSI-11 POWER FAIL DETECT
SHIVA LASER. CLASS LASER SYSTEM
SCHEMATIC
LAWRENCE LIVERMORE LABORATORY
ELECTRONICS ENGINEERING DEPARTMENT
LEA 77-1258 11 -S 0
SHEET 1 OF 1


FREQ ADJ
R6 20K TRIMMER
R4 1K
R5 5.1K
IC4 555
C9 470pf
EXT CLOCK IN
TP3
GND
TP2
NORMAL
EXT
S2 5930-53102
TP4
R3 1K
+5V
IC3 74193
IC2 74193
AM2
AN2
AR2
AS2
BLAKI
BLAKO
BDMGI
BDMGO
J1 5935-60478
CSRO
S1 5930-53102
TEST
R1 1K
IC1 7438
R2 270Ω
AEI
SYS CLK L
TP1
TO DRVII J2
R7 1K
IC5 7474
DATA TRANS H
INIT H
REQ B H
IC6 7402
C3-C8 .01uf
C2 39uf 10V TANT
C1 .01uf
AA2
BA2
AM1
AT1
A12
FEY
VAN ARSDALL
STEPPING MOTOR LOGIC BOARD
SYSTEM CLOCK SOURCE
SHIVA LASER, GLASS LASER SYSTEM
SCHEMATIC
LAWRENCE LIVERMORE LABORATORY
ELECTRONICS ENGINEERING DEPARTMENT
LEA 75-1344
41-S1A


A13
NOTES: UNLESS OTHERWISE SPECIFIED
1. ALL RESISTORS ARE FIXED, CC, 1/4W.
2. Q1-Q12 MOUNTED FLAT WITH HEATSINK THERMALLOY 6060-18.
3. K5-15 BOSTH FOR NORMALLY CLOSED LIMIT SWITCHES.
15 BOSTH FOR NORMALLY OPEN LIMIT SWITCHES.
4. CR19-CR30 ARE 1AMP SILICON STK NO.5961-53136.



NOTES: UNLESS OTHERWISE SPECIFIED

1. JUMPER 122 (A5), 1 TO E FOR UNIPOLAR OPERATION.
   2 TO C FOR BIPOLAR OPERATION.
2. C1-C16 SIZE D OR SMALLER TANTALUM FOR UNIPOLAR SUPPLY ⏚|—, OR BACK TO BACK SIZE D TANTALUM FOR BIPOLAR SUPPLY —||—, ALL VALUES ARE USER SELECT.
3. JUMPERS TO MONITOR POT SUPPLY VOLTAGES.
4. R1-R17 ARE FIXED CC. VAL. USER SELECT.
5. ALL DIODES ARE 1N4447.

APPENDIX B

SMC PROCESSOR HARDWARE CONFIGURATION

CHASSIS MODIFICATION LIST

The standard LSI-11 package as designed for SHIVA project must have a few minor changes in order to be used for the needs of SHIVA alignment. Below is a list of the changes in the recommended order of modification.

1) Bus the following special spare bus pins throughout the backplane (see Figure A-1). AE1* (stepping motor clock), BC1*, DC1* (testfail), CH1, AH1 (SRUNL signal for CPU run light).

2)* Install 12v 3amp power supply for prom memory cards. The positive terminal of the power supply should be wired to the ground plate in the bottom of the LSI-11 chassis (use 16GA black). The negative of the supply should be wired to the -12v terminal on the backplane terminal strips (use 16 GA white).

3)* Turn backplane around so that slot 1 is towards front of chassis.

4) Remove the red wire, labeled "PS 2" from PS2 + sense terminal and put it on the +5v output of that supply. (Note: Leave the wire that was on the spade lug with "PS2" on the + sense terminal).

5) Repeat step 4 for all +5v 25amp power supplies in chassis.

6) Add a 22GA wire (orange) from the +12v output terminal on PS1 to the powerfail detect power supply.

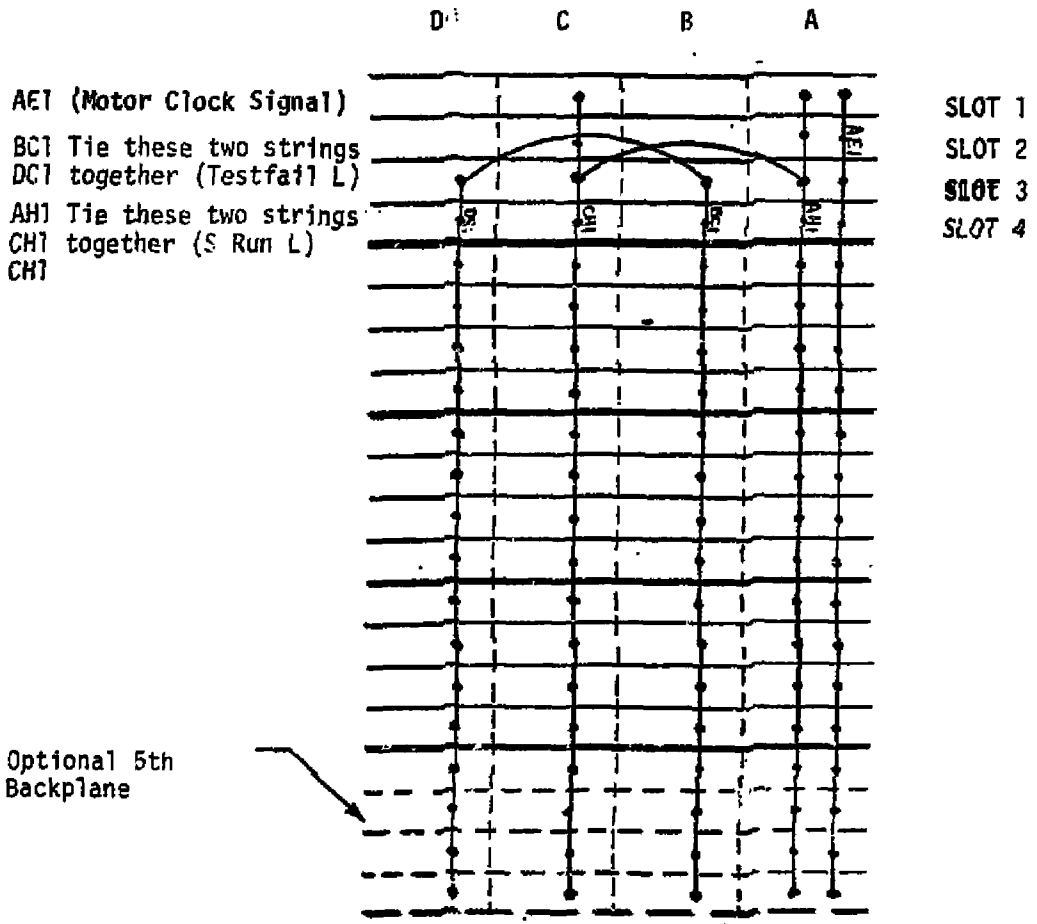
7) Replace the green discrete LED with a green 5v cartridge type LED. Place red wire on anode (long terminal) and black wire on cathode (short terminal).

8) Install and wire the powerfail detect power supply (see Figure A-2).

*These modifications may not be needed in applications other than SHIVA Alignment Controls Group.

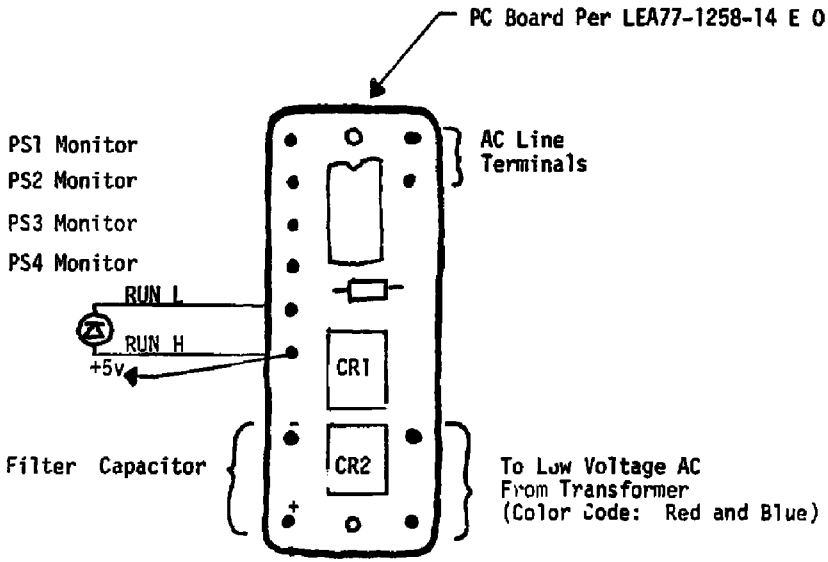
Figure A - 1

5 FINGERS MUST BE BUSSED TOGETHER AS FOLLOWS



Looking From Bottom Side of Backplanes

Figure A - 2



Powerfail Power Supply Board Pinouts

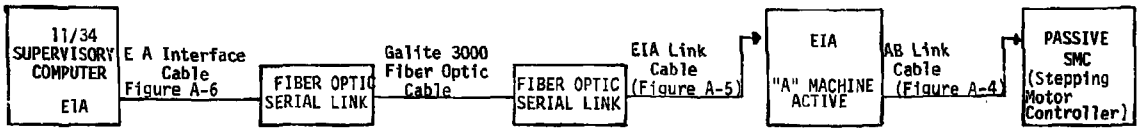
Figure A - 3

NETWORK INTERCONNECTION

The interconnection of an "A" level FEP and an SMC processor requires that the link DLV-11 of one be active and the other passive(current loop).

As a means of uniformity the DLV-11 in the "A" level processor has been made active (as jumpered from factory) and the DLV-11 in the SMC processor has been rejumpered to be passive.

The link from the "A" level processor to higher level must be wired EIA in order to interface to a fiber optic serial link box.



NETWORK LINK CABLES

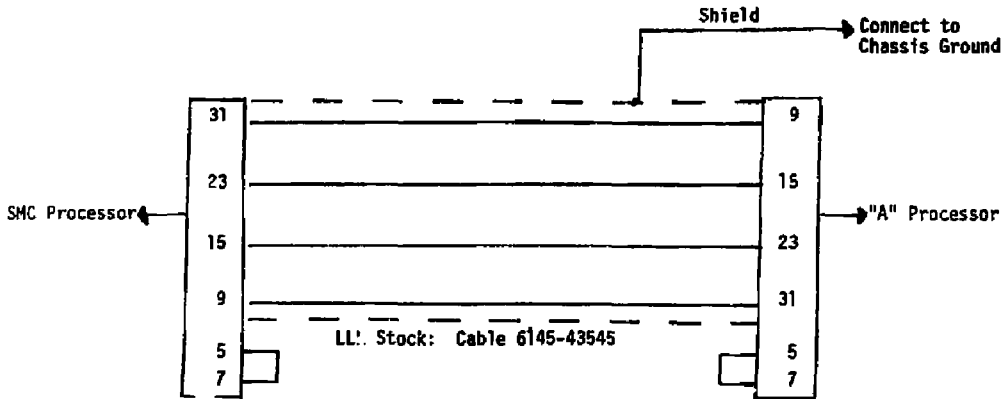


Figure A - 4
AB Link Cable

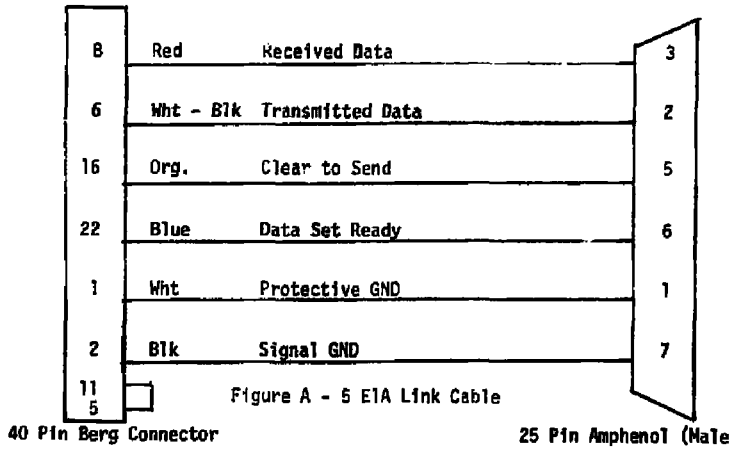


Figure A - 5 EIA Link Cable

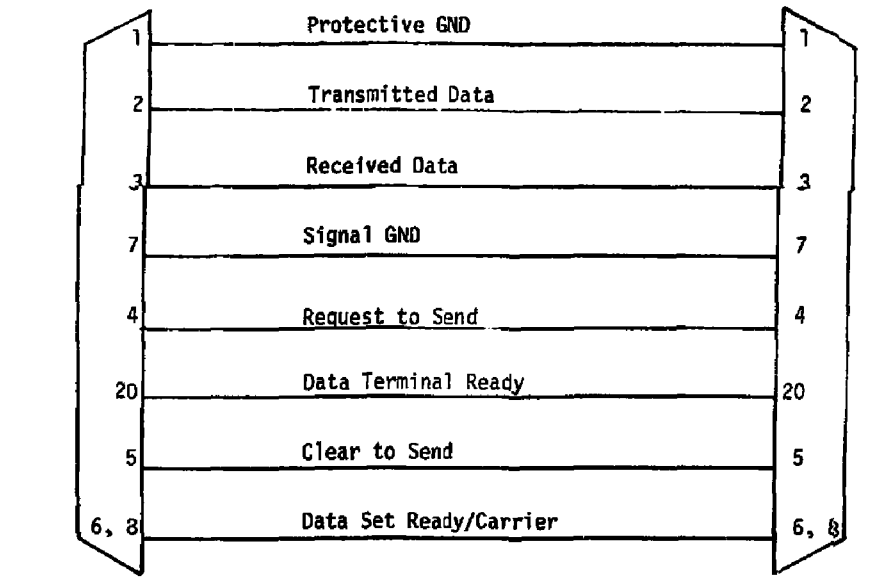


Figure A - 6 EIA Interface Cable

STRAPABLE OPTIONS

The following strap options are to be used on the "B" level or SMC processors.

CPU - Jumpers W1, W11 and W6 are installed; all others are left out.

Terminal DLV-11 - Address: 177560
Vector : 60
Jumpers A3, A7, V3, V6, V7 installed

Remove "FEW" and "E1A" jumpers
Install FR1 if 300 Baud terminal is used

Network DLV-11 - See table in Figure -7

4K Prom - Board 1 Hex switch position #2 (040000 starting address)
Jumpers installed: 3N-2N, 3M-2M, 3L-2L, 2K-1K, 3J-2J, 3H-2H, 3F-2F
Board 2 Hex switch position #3 (060000 starting address)
Jumpers installed: 3N-2N, 3M-2M, 2K-1k, 3H-2H

Motor Clock DRV-11 = Address: 160010 Jumpers installed A4 through A12
Vector : 110 Jumpers installed V4, V5, V7

Motor Clock = Both switches set to normal position
Adjust clock for 300HZ to AE1

SLC card 1 -	address	160100	Jumper installed	A6
SLC card 2 -	"	160120	"	A6, A4
SLC card 3 -	"	160140	"	A6, A5
SLC card 4 -	"	160160	"	A4, A5, A6
SLC card 5 -	"	160200	"	A7

Front Panel Interface - address 160600 All jumpers installed except A7, A8.

Terminator/Boot Card - Remove jumper W2 and replace all four prom IC's with SHIVANET Rev proms

4K Core Memory - All switches in the "on" positions (0-4K)

MDLV-11 JUMPER CONFIGURATIONS

"A" Processor Network Link to 11/34

Address	Vector	Jumpers Installed
176520	310	12 N-M, 12 J-H, 11 N-M, 11 H-K 10 H-J, 10 L-M, 9 J-K, 9 L-M 8 J-K, 8 L-M, 7 J-K, 7 L-M

NOTE: Used as EIA Interface

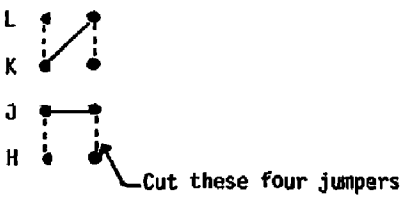
Jumper #5 and #2 (Vector)

Remove Jumper #1 and Jumper FEH Signal to Bus Pin BC1

SMC Processor Network Link to "A" Machine

All jumpers are the same as above in addition to changing the mode of operation to passive current loop.

Cut all four stock jumpers and replace with the two jumpers *illustrated below.*



Passive Current
Loop Operation

Dip switch positions for all of the above are the same. Switch #6 is "on" and all other switches are open.

MDLV-11 JUMPER CONFIGURATIONS (CON'T)

"A" Processor to SMC Processor

Install the basic address jumpers in addition to the jumpers indicated below for each address listed.

Basic address jumpers:

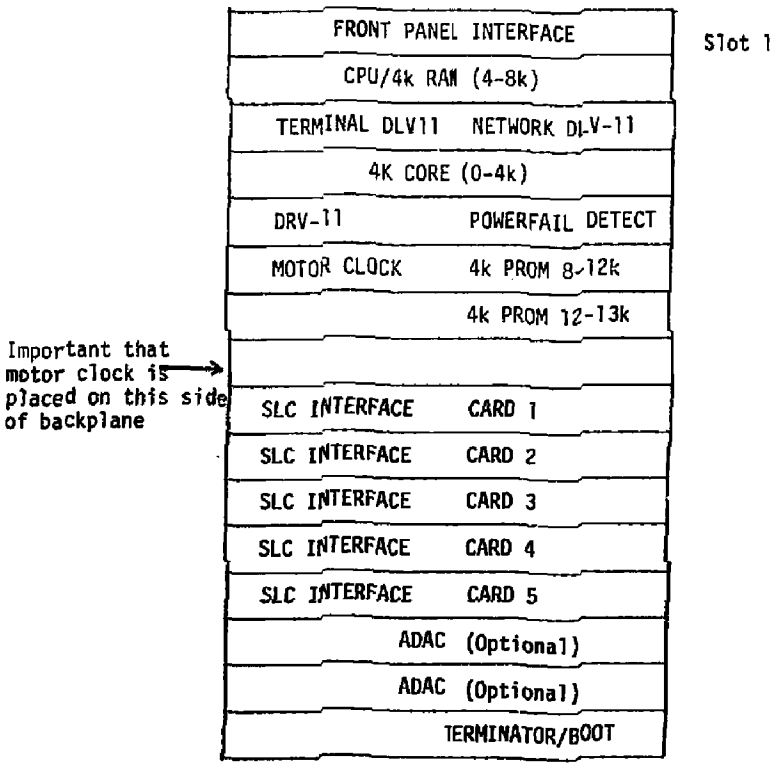
11 N-M, 12 N-M, 12 H-J, 9 K-J, 10 H-J

	ADDITIONAL ADDRESS JUMPERS	VECTOR JUMPERS INSTALLED
SMC 0	10 L-M, 9 L-M, 8 L-M, 8 K-J, 7 H-J	#3, #4, #5
SMC 1	10 L-M, 9 L-M, 8 N-M, 8 H-J, 7 K-J	#3, #4, #5, #6
SMC 2	10 L-M, 9 L-M, 8 N-M, 8 H-J, 7 H-J	#3, #4, #2
SMC 3	10 L-M, 9 L-M, 8 N-M, 8 K-J, 7 K-J	#3, #4, #2, #6
SMC 4	10 L-M, 9 L-M, 8 N-M, 8 K-J, 7 H-J	#3, #4, #2, #5
SMC 5	10 N-M, 9 N-M, 8 L-M, 8 H-J, 7 K-J	#3, #4, #2, #5, #6
SMC 6	10 N-M, 9 N-M, 8 L-M, 8 H-J, 7 H-J	#3, #5
SMC 7	10 N-M, 9 N-M, 8 L-M, 8 K-J, 7 K-J	#3, #5, #6
SMC 8	10 N-M, 9 N-M, 8 L-M, 8 K-J, 7 H-J	#3, #2
SMC 9	10 N-M, 9 N-M, 8 N-M, 8 H-J, 7 K-J	#3, #2, #6

Dip switch positions for all of the above are the same. Switch #6 is "on" and all other switches are open.

All of the above interfaces are to be used in the active current loop mode. (Factory etched jumpers)

LSI-11 CONFIGURATION
B LEVEL PROCESSOR



LSI II POWERFAIL DETECT CARD CONFIGURATION

JUMPERS:

Testfail H Source: BCI
Event H Frequency: 60Hz

ADJUSTMENTS: *

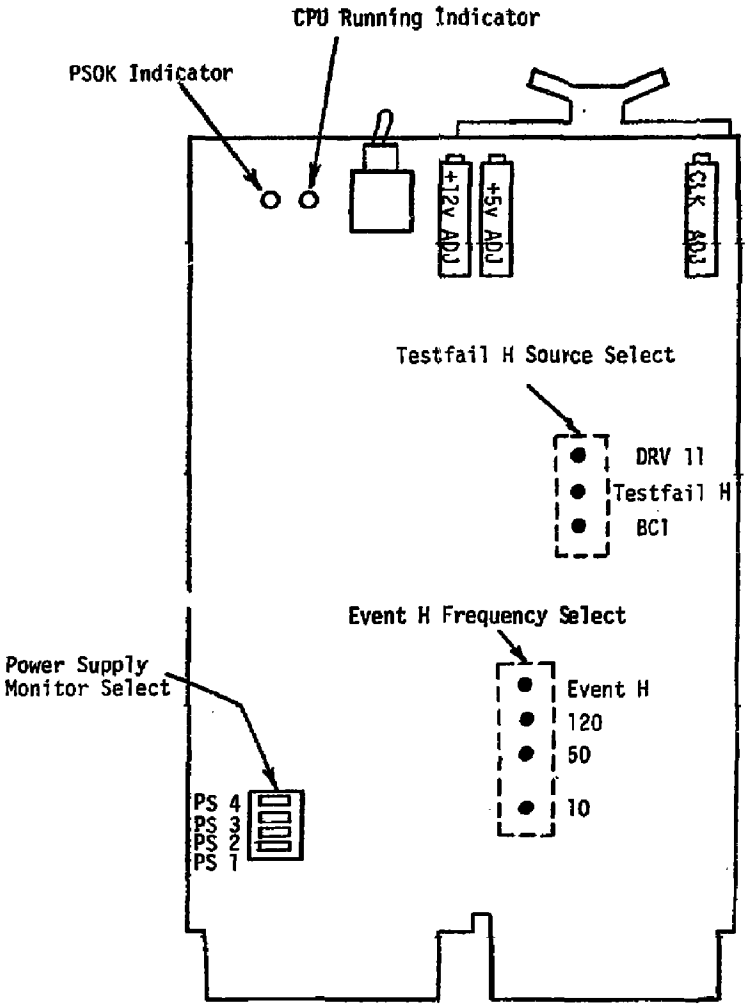
1. Select +12v powersupply monitor. Turn +12v ADJ. clockwise until the "PSOK" indicator lights, then turn counter-clockwise until indicator goes off. Turn adjustment an additional one and one-half turns counter-clockwise.

2. Deselect +12v powersupply monitor and select All +5v supplies that are up to voltage. Similarly turn +5v ADJ. clockwise until the "PSOK" indicator lights, then turn counter-clockwise until indicator goes off. Turn adjustment an additional one and one half-turn counter-clockwise.

3. Select powersupply monitors for all the supplies that are up to voltage.

4. Adjust clock ADJ. counter-clockwise until signal on B Event L testpoint disappears, then clockwise until it reappears.(Clock Switch) Turn adjustment an additional 2 turns clockwise.

*NOTE: *These are only initial adjustments and should not be used for* precise powerfail requirements.



POWERFAIL DETECT CARD LAYOUT

APPENDIX C

SMC PROCESSOR ADDRESS UTILIZATION

SMC PROCESSOR ADDRESS UTILIZATION

Memory Address Usage:

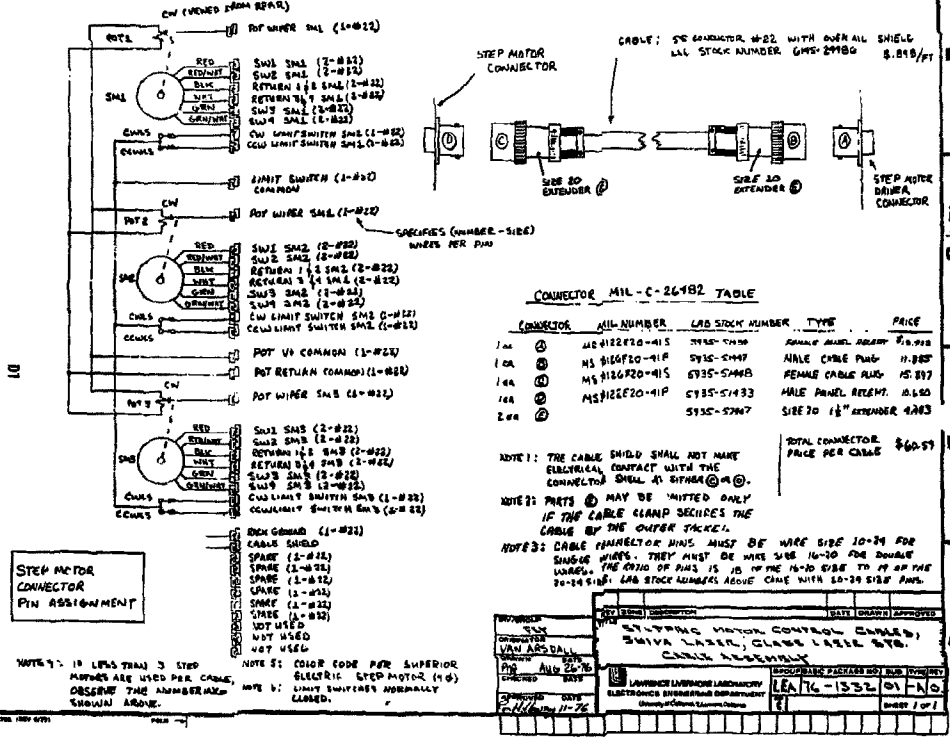
```
000000  Reserved
000004  Bus Timeout and Illegal Instruction Trap
000010  Illegal and Reserved Instruction Trap
000014  BPT Instruction and T Bit
000024  Power Fail and Recovery Interrupt
000034  Trap Instruction
000060  Console Input Interrupt
000064  Console Output Interrupt
000100   60 HZ   Clock Interrupt

000114  Stepping Motor Clock Interrupt ((DLV11)
000120  Network Serial Link Interrupt   (In Master, Slave 7)
000130                                  (In Master, Slave 8)
000140                                  (In Master, Slave 9)
000150                                  (In Master, Slave 10)
000310  Network Serial Link Interrupt   (Slave)
000320  Network Serial Link Interrupt   (In Master, Slave 1)
000330                                  (In Master, Slave 2)
000340                                  (In Master, Slave 3)
000350                                  (In Master, Slave 4)
000360                                  (In Master, Slave 5)
000370                                  (In Master, Slave 6)

    .
    .
    .     Stack
    .
    .
001176
001200  R/W Core Buffers
    .
    .
037776

040000
    .
    .     Rom Code
    .
060523
160000  ..... I/O Page Starts Here .....
```

```
160100  Stepping Motor #01        Card #1
     160102              2
     160104              3
     160106              4
     160110              5
     160112              6

     160120              7
     160122              8
     160124              9
     160126             10
     160130             11
     160132             12

        Etc.
        .
        .
     160202             30     (End of 30 Step Motors)
     160600
        .
        .
        .
     160616
     165776
     173000
        .
        .       Revl1 Rom Programs
        .
     165776
     173000
        .
        .       Revl1 Rom Programs
        .
     173766
     176520  Network Serial Link I/O (Slave)
     176530  Network Serial Link I/O (In Master, To Slave 1)
     176540                          (In Master, To Slave 2)
     176550                          (In Master, To Slave 3)
     176560                          (In Master, To Slave 4)
     176570                          (In Master, To Slave 5)
     176600                          (In Master, To Slave 6)
     176610                          (In Master, To Slave 7)
     176620                          (In Master, To Slave 8)
     176630                          (In Master, To Slaved9)
     176640                          (In Master, To Slave 10)
```

```
176700  ADAC Pot Reading Converter #1
    2
176704                                  2
    6

177560  Console Terminal,       Input Control, Status
177562     "         "            Input Buffer
177564     "         "            Output Control, Status
177566     "         "            Output Buffer
```

APPENDIX D

STANDARD STEPPING MOTOR CONTROL CABLE


CW (VIEWED FROM REAR)
POT1
POT WIPER SIG1 (1-#21)
SW1 SIG1 (2-#22)
SW2 SIG1 (2-#22)
RETURN 1 & 2 SIG1 (2-#22)
RETURN 3 & 4 SIG1 (2-#22)
SW3 SIG1 (2-#22)
SW4 SIG1 (2-#22)
SIG1
RED
RED/WHT
BLK
WHT
GRN
GRN/WHT
CW LIMIT SWITCH SIG1 (1-#22)
CCW LIMIT SWITCH SIG1 (1-#22)
CWLS
CCWLS
LIMIT SWITCH (1-#22) COMMON
POT2
POT WIPER SIG2 (1-#22)
SPECIFIES (NUMBER - SIZE) WIRES PER PIN
SW1 SIG2 (2-#22)
SW2 SIG2 (2-#22)
RETURN 1 & 2 SIG2 (2-#22)
RETURN 3 & 4 SIG2 (2-#22)
SW3 SIG2 (2-#22)
SW4 SIG2 (2-#22)
SIG2
CW LIMIT SWITCH SIG2 (1-#22)
CCW LIMIT SWITCH SIG2 (1-#22)
POT V+ COMMON (1-#22)
POT RETURN COMMON (1-#22)
POT3
POT WIPER SIG3 (1-#22)
SW1 SIG3 (2-#22)
SW2 SIG3 (2-#22)
RETURN 1 & 2 SIG3 (2-#22)
RETURN 3 & 4 SIG3 (2-#22)
SW3 SIG3 (2-#22)
SW4 SIG3 (2-#22)
SIG3
CW LIMIT SWITCH SIG3 (1-#22)
CCW LIMIT SWITCH SIG3 (1-#22)
RICK GERARD (1-#22)
CABLE SHIELD
SPARE (1-#22)
SPARE (1-#22)
SPARE (1-#22)
SPARE (1-#22)
SPARE (1-#22)
SPARE (1-#22)
NOT USED
NOT USED
NOT USED
STEP MOTOR CONNECTOR PIN ASSIGNMENT
NOTE 4: IF LESS THAN 3 STEP MOTORS ARE USED PER CABLE, OBSERVE THE NUMBERING SHOWN ABOVE.
NOTE 5: COLOR CODE PER SUPERIOR ELECTRIC STEP MOTOR (4-#)
NOTE 6: LIMIT SWITCHES NORMALLY CLOSED.
STEP MOTOR CONNECTOR
CABLE; 5# CONDUCTOR #22 WITH OVER ALL SHIELD
LAL STOCK NUMBER GMS-29180 $.89/FT
SIZE 20 EXTENDER (D)
SIZE 10 EXTENDER (C)
STEP MOTOR DRIVER CONNECTOR
CONNECTOR MIL-C-26482 TABLE
CONNECTOR
MIL NUMBER
LAB STOCK NUMBER
TYPE
PRICE
1 ea A MS3122E20-41S 5935-57490 FEMALE PANEL RECEPT. $10.92
1 ea B MS3106F20-41P 5935-57447 MALE CABLE PLUG 11.85
1 ea C MS3126F20-41S 5935-57448 FEMALE CABLE PLUG 15.87
1 ea D MS3122E20-41P 5935-57433 MALE PANEL RECEPT. 10.60
5935-57447 SIZE 20 1½" EXTENDER 4.83
2 ea E
TOTAL CONNECTOR PRICE PER CABLE $60.57
NOTE 1: THE CABLE SHIELD SHALL NOT MAKE ELECTRICAL CONTACT WITH THE CONNECTOR SHELL AT EITHER A OR B.
NOTE 2: PARTS E MAY BE OMITTED ONLY IF THE CABLE CLAMP SECURES THE CABLE BY THE OUTER JACKET.
NOTE 3: CABLE CONNECTOR PINS MUST BE WIRE SIZE 20-24 FOR SINGLE WIRES. THEY MUST BE WIRE SIZE 16-20 FOR DOUBLE WIRES. THE RATIO OF PINS IS 1B IN THE 16-20 SIZE TO 19 IN THE 20-24 SIZE. LAB STOCK NUMBERS ABOVE CAME WITH 20-24 SIZE PINS.
KEY
ZONE
DESCRIPTION
DATE
DRAWN
APPROVED
DRAWN BY TLK
CONTRACTOR VAN ARSDALL
APP. Aug 26-76
CHECKED DATE
APPROVED DATE
STEPPING MOTOR CONTROL CABLES; SWIVA LASER; CLASS LASER SYS.
CABLE ASSEMBLY
LAWRENCE LIVERMORE LABORATORY
ELECTRONICS ENGINEERING DEPARTMENT
GROUP/BASIC PACKAGE NO.
SUB
SYSTEM
LEA
76-1332
01
NO
SHEET 1 OF 1