305
11/14/79

LA-7895

Ab. 304

A Microprocessor-Based Stepping Motor Driver

MASTER

University of California

LOS ALAMOS SCIENTIFIC LABORATORY
Post Office Box 1663 Los Alamos, New Mexico 87545

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

LA-7895
UC-32
Issued: September 1979

A Microprocessor-Based Stepping Motor Driver

J. K. Halbig
S. F. Klosterbuer

DISCLAIMER

This book was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

A MICROPROCESSOR-BASED STEPPING MOTOR DRIVER

by

J. K. Halbig and S. F. Klosterbuer

ABSTRACT

The Pion Generation for Medical Irradiations (PIGMI) program at the Los Alamos Scientific Laboratory requires a versatile stepping motor driver to do beam diagnostic measurements. We designed a driver controlled by a microprocessor that can move eight stepping motors simultaneously. It can monitor and respond to clockwise- and counterclockwise-limit switches, and it can monitor a 0- to 10-Vdc position signal. The software controls start and stop ramping and maximum stepping rates.

I. INTRODUCTION

The stepping motor driver board and associated software provide logic driver signals and limit and setpoint monitoring for up to eight stepping motors. The board supplies transistor-transistor logic (TTL) - level clockwise (cw) and counterclockwise (ccw) signals for each channel, and monitors cw- or ccw-limit lines. The software controls the rate at which the driver signals are sent; start and stop ramping are available. No further pulses are sent to a motor if the software senses that the motor has reached a specified limit. A 12-bit analog channel, which is associated with each motor, monitors a ±10-Vdc analog voltage. This voltage is proportional to the motor-driven position or setpoint.

This board could be part of a dedicated, smart controller, as shown in Fig. 1. Figure 1 shows the dedicated processor (UL1), a read only memory (ROM) that contains the controller operating-system software (UL2), two peripheral interface adapters (PIAs) (UL3 and UL5), and a PIA that interfaces to the outside world (UL4). The controller can be used in nondedicated applications, also. This report describes its use with a general-purpose microprocessor-based controller (MPC).[1,2]

II. HARDWARE

Two PIAs on an MPC-PIA board interface the stepping motor driver board to the MPC. Two 26-conductor ribbon cables connect the driver board to the PIAs. Figure 2 (in back pocket) shows the circuitry between the PIAs and the motors. The software that drives the system is described later.

The driver signals are sent through 50-Ω line driver, or SN74128, gates. The pulse is generated by the microprocessor, which changes the level of one of the PIA output lines. (One output line is dedicated to cw motion, another to ccw motion.) The signal is buffered by and sent through the 74128 gates, which must be "enabled" before the pulse is transmitted to the power driver electronics. A single output pulse is a high-to-low-to-high transition. The gates are enabled by storing an enable mask in the 8-bit, or SN74273, latch. For a motor to be enabled, a logic 0 must be stored in the corresponding bit of the latch, and conversely. The data are latched using two handshake lines, the CA2 and CB2 lines of the BERG 2 PIA. The CB2 line must be high to enable the latch signal, which is transmitted by the CA2 line in a high-to-low-to-high transition. After the data have been latched, the CB2 line must be lowered.

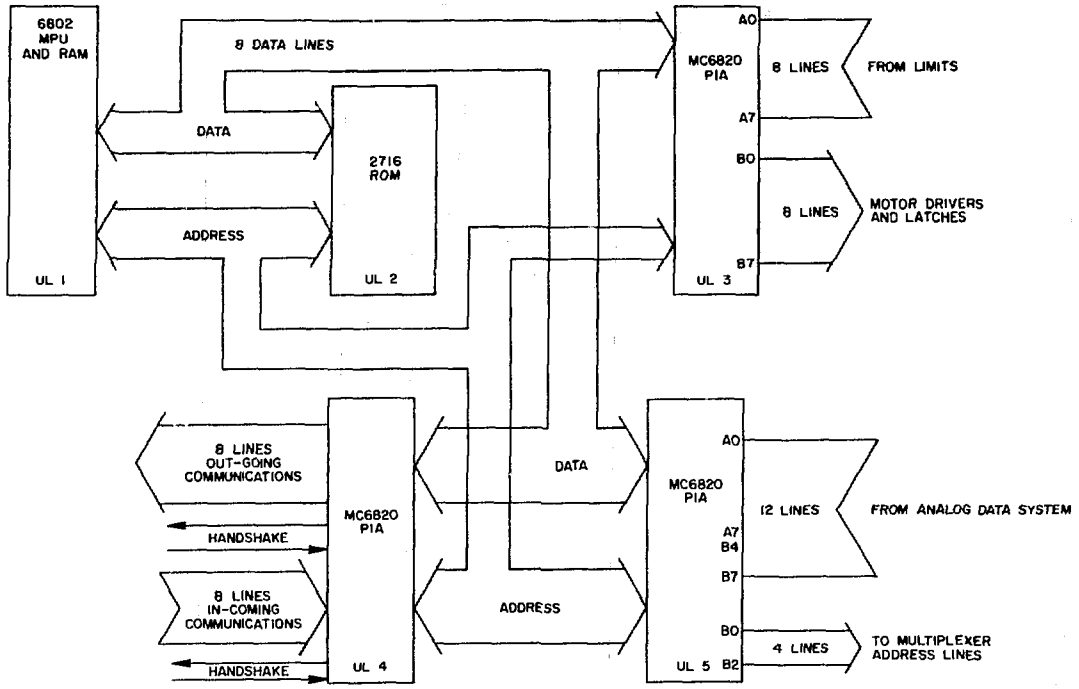


Fig. 1.
A dedicated, smart stepping motor driver.

Note that the latch and pulse circuitry share the same output lines of the BERG 2 PIA, but when the latch pulse is enabled (CB2 line high), the driver lines are disabled, and conversely.

The limit information comes over eight pairs of lines. Each line of an active pair must be attached to a mechanical switch or to an active device that sinks 1.6 mA when the limits are not satisfied. To activate a limit, the control switch must open, or the control gate must go high. These pairs of lines go into two-to-one line decoders. When the motor direction is selected, the processor must specify the decoder state that gates the limit information into the BERG 2 PIA. The decoder state is determined by the level of the PB0 line. The processor checks the limit status of the motor before sending a pulse to that motor's driver electronics.

The analog data are transmitted to the board via eight pairs of lines. Each pair is connected to a differential amplifier. The amplifier outputs connect to an eight-channel analog multiplexer (mux). The channel is selected by the mux, which decodes the PB0, or least significant bit (LSB), through PB2, or most significant bit (MSB), lines of the BERG 1 PIA. The selected signal is gated to the 12-bit analog-to-digital (A/D) converter. The conversion is started at least 5 μs after sending the mux address because of the settling time of the mux. The trigger for this start-conversion pulse is the CB2 line of the BERG 1 PIA. The processor is signaled that the conversion is complete when the A/D converter lowers the CA1 line.

III. SOFTWARE

The software is executable using the MPC and is designed to demonstrate the capabilities of the stepping motor driver. Adaptation and extension of the code are straightforward.

The sample software that follows consists of two parts, general routines that can be called by any program and a specific test program that uses these routines. The general routines, shown in Listing 1, contain the necessary code to initialize the stepping motor PIAs; to run, retract, and insert motors; and to read the analog setpoints. The driver board is interfaced through two PIAs; one connects to the analog data circuitry and the other connects to the logic circuitry.

The RESET routine initializes the PIAs. To initialize the PIA connected to the analog data circuitry lines, PA7-PA0 and PB7-PB4 must be programmed as inputs, whereas PB3-PB0 must be outputs. The control register must be sensitive to the falling edge of the CA1 line; a $04 ($ denotes a hexadecimal No.) in the A-side control register (CRA) causes the IRQA1 flag to set when this condition occurs. A pulse on the CB2 line starts a conversion; a $2C in the B-side control register (CRB) causes the pulse to be sent when a store is made to the corresponding data register to address a particular analog channel. To initialize the PIA connected to the logic circuitry, which runs the motors, the A-side peripheral data register (PDRA) bits must be inputs, and the B-side peripheral data register (PDRB) bits must be outputs. The CA2 and CB2 lines must be high ($3C in CRA and CRB). The initialization software must also set default values used in the RUN routine.

The RUN subroutine moves the motors. Note that the motor enable mask must be in the A accumulator, and the total number of pulses, signed binary, must be in the X register when this routine is called. The motor enable mask contains a 1 in the bit position corresponding to the motor(s) to be enabled; for example, if motors 1 and 5 are to be enabled, a 1 is written in bits 0 and 4 of the A accumulator. To move cw, the number of pulses is positive; to move ccw, the number is negative. This routine ramps the motors by sensing the position of the upper and lower corners, which must be established with their delay increments before calling the routine.

This routine enables the motors by

(1) bringing the CB2 line high to disable the driver lines and enable the latches,
(2) complementing and storing the enable mask in the PDRB,
(3) strobing the CA2 line high-low-high to latch the motor enable data,
(4) clearing the PDRB of the PIA, and
(5) bringing the CB2 line low to enable the driver lines and disable the latch.

Once the selected motors are enabled by this procedure, one can run them by pulsing the PB7 or PB6 line of the PIA. The pulses are created by storing a 1 and then a 0 in the PB7 bit for ccw movement, or in the PB6 bit for cw movement. The pulses must be applied at intervals, set by predetermined delays, that produce the desired motor speed. The remainder of the routine checks whether a corner has been reached and takes appropriate action by increasing, decreasing, or not changing the delay time between pulses.

Before each pulse is applied, the code checks whether a limit has been reached. The limit status is read from the PDRA, which displays the status of the limits that are selected by the state of the PB0 line. The cw- or ccw-limit status is mux-ed to the PDRA by writing a 0 in PB0 when driving cw or a 1 when driving ccw. If any enabled motor has reached its limit, the routine executes a return—no more pulses are sent to any of the motors.

The routines to insert or retract motors, INSR and RETR, require the A accumulator to contain the motor enable mask before they are called. These routines drive the motors to the respective limits (insert, cw; retract, ccw) by sending groups of 2400 pulses to the RUN routine. The routines continue until all the selected motors reach the specified limit.

SETPT is the routine that reads the analog setpoint. It is called with the binary code of the channel (0-7) to be read in the A accumulator. The routine selects the proper mux channel by storing the binary code in the PDRB. This store initiates a start-convert pulse on the CB2 line. The program then waits for the CA1 interrupt flag to be set, which signals that conversion is complete. Then, the data are read back. The 12-bit complemented 2's complement representation of the analog voltage is read from PA7-PA0 and PB7-PB4. The MSB is PA7; the LSB is PB4. The selected channel number is read on PB3-PB0. The data are complemented and returned to the calling routine in the A and B accumulators.

The test program DEMO, shown in Listing 2, which uses these routines, also uses the push-button panel monitor software described in Ref. 2. The push buttons are programmed for the following.

Button 1 reads the decimal number of pulses on the hexadecimal (hex) switch and converts the number to binary; it reads the motor enable mask from the toggle switches and runs the enabled motors cw the number of counts using RUN; and it displays the setpoint of the first enabled motor in decimal on the hex display.

Button 2 does the same as button 1, except it runs the selected motors ccw.

Button 3 reads the motor enable mask from the toggle switches; it inserts the selected motors to the limit; and it displays the setpoint of the first enabled motor on the hex display at completion.

Button 4 does the same as button 3, except it retracts the motors.

Button 5 reads the motor enable mask from the toggle switches; and it displays the setpoint of the first enabled motor in decimal on the hex display.

Buttons 6 and 7 are initialized to be in the duration mode.[2] While button 6 or 7 is pushed, single pulses are continuously sent to the enabled motors, and then the setpoint of the first enabled motor is updated on the hex display. Button 6 runs the motors cw; button 7 runs them ccw.

Button 8 reads the decimal number of pulses from the hex switch; and it reads the motor enable mask from the toggle switches. The motors are moved cw this number of pulses using RUN, and then the setpoint of the first enabled motor is shown on the hex display. To complete the cycle, the motors are run ccw, and the setpoint is displayed. This cycle continues until button 8 is pushed again.

If used as a smart controller, as shown in Fig. 1, the controller accepts task commands over the communication lines and executes the tasks. The protocol provides for transmission of operational (op) codes, arguments, and busy indications. Table I lists typical tasks and their associated arguments.

The SET task initializes numbers used in the RUN routine. The delays and corner positions are examples of these numbers.

The READ SETPOINTS routine returns the digital setpoint data with the mux address. The most significant 8 bits are contained in the first word returned; the least significant 4 bytes and the address are contained in the second word returned. As many pairs of data are returned as are requested by the setpoint mask.

The CHECK LIMITS routine returns one word that contains the limit status of all motors, whether or not they are enabled.

The MOVE TO SETPOINT routine enables one to specify the setpoint to which a single motor is driven. The motor enable mask with only a single bit as a 1 is sent to the controller with the setpoint. If more than one motor is enabled by the mask word, or if the setpoint is not reached within an internal tolerance in a given number of attempts, an error message is returned to the commanding program. The slope (Δ setpoint/motor pulse) is defaulted to 1.

The SLOPE task specifies the slope, which is calculated after each move, and the updated value is saved for future use.

Other tasks can be defined. The smart controller can be extended easily to function with more than one interface board.

TABLE I

SMART STEPPING MOTOR INTERFACE PROTOCOL

Task Name	Op Code	Parameters
RESET	---	Hardwired
SET	1	(a) Initial stepping rate
		(b) Final stepping rate
		(c) Number of steps of slew
RUN	2	(a) Enable motor mask
		(b) Number of pulses
READ SETPOINTS	3	(a) Setpoint mask
		(b) Returns values 2 bytes/motor (12 bits of data, 4 bits for mux address)
CHECK LIMITS	4	Motor limit status returned
MOVE TO SETPOINT	5	(a) Motor enable mask
		(b) Setpoint
		(c) Returns setpoint or error
SLOPE	6	(a) Motor enable mask
		(b) Slope for each motor enable in order of occurrence.

ACKNOWLEDGMENTS

We thank M. P. Dugan and V. A. Martinez who did the technician and drafting work and the MP-11 staff for helping to check the circuitry.

REFERENCES

1. R. W. Goodwin, R. F. Kocanda, and M. F. Shea, "A Method for Implementing Microprocessor Controlled Systems," IEEE Trans. Nucl. Sci. NS-23, 1, 297-300 (1976).

2. J. K. Halbig, S. F. Klosterbuer, and D. A. Swenson, "A General-Purpose Microprocessor-Based Control Chassis," Los Alamos Scientific Laboratory report LA-7896 (1979).

LISTING 1

LASL Identification No. LP 1068

```
00001                           NAM     STEP
00002                    *
00003                    *      STEPPING MOTOR ROUTINES
00004                    *
00005        F024        DMICRO  EQU     $F024
00006        8900        ADPIA   EQU     $8900
00007        8902        RUNPIA  EQU     $8902
00008                    *
00009                    *
00010 FA00                       ORG     $FA00
00011 FA00 7E FA0F VPSET  JMP     RESET   INIT. PIAS
00012 FA03 7E FA42 VPUN   JMP     PUN     PUN MOTORS
00013 FA06 7E FB81 VPETP  JMP     PETP    RETRACT MOTORS
00014 FA09 7E FB96 VINSR  JMP     INSR    INSERT MOTORS
00015 FA0C 7E FBC4 VSTPT  JMP     SETPT   READ SETPOINT
00016                    ********************************************
00017                    *
00018                    *      RESET-INITIALIZE PIAS
00019                    *      ADPIA = ANALOG DATA PIA
00020                    *      RUNPIA = MOTOR RUN PIA
00021                    *
00022                    *
00023 FA0F 86 14   RESET  LDA A  #20      DEFAULT CORNER POSITION
00024 FA11 B7 A080        STA A  CORNP
00025 FA14 CE 00BB        LDX    #187     DEFAULT DELTA DELAYS
00026 FA17 FF A089        STX    RDDLA
00027 FA1A CE 1388        LDX    #5000    DEFAULT PULSE DELAYS
00028 FA1D FF A08B        STX    RDELA
00029 FA20 CE 0000        LDX    #$0000   SET UP CONTROL REG FOR DDR
00030 FA23 FF 8908        STX    ADPIA+8  *
00031 FA26 FF 890A        STX    RUNPIA+8 *
00032 FA29 CE 000F        LDX    #$000F   SET ADPIA I/O REG
00033 FA2C FF 8900        STX    ADPIA
00034 FA2F CE 00FF        LDX    #$00FF   SET RUNPIA I/O REG
00035 FA32 FF 8902        STX    RUNPIA
00036 FA35 CE 042C RESET1 LDX    #$042C   CA1:H.T.L.; CB2 PULSE MODE
00037 FA38 FF 8908        STX    ADPIA+8  SET CONTROL REG
00038 FA3B CE 3C3C        LDX    #$3C3C   CA2 AND CB2 HIGH
00039 FA3E FF 890A        STX    RUNPIA+8 SET CONTROL REG.
00040 FA41 39             RTS
00042                    ********************************************
00043                    *
00044                    *      SLEW MOTORS
00045                    *
00046                    *
00047                    *      TO ENTER:
00048                    *              A = MOTOR ENABLE MASK
00049                    *              X = TOTAL PULSES SIGNED BINARY
00050                    *
00051        FA42        PUN     EQU     *
00052 FA42 B7 A094               STA A   ENWRD   MOTORS TO ENABLE
00053 FA45 08                    INX
00054 FA46 FF A097               STX     PLSCTR  TOTAL # OF PULSES
00055 FA49 7F A093               CLR     DIRWD   CW OR CCW
00056 FA4C B6 A097               LDA A   PLSCTR
00057 FA4F 2A 0A                 BPL     RUN1    CW
00058 FA51 84 7F                 AND A   #$7F    REMOVE SIGN BIT
00059 FA53 B7 A097               STA A   PLSCTR
00060 FA56 86 80                 LDA A   #$80    SIGNALS CCW
00061 FA58 B7 A093               STA A   DIRWD
00062 FA5B FE A089 RUN1   LDX    RDDLA   SET DELAYS
00063 FA5E FF A081               STX     DDLA1
00064 FA61 FF A083               STX     DDLA2
00065 FA64 FE A08B               LDX     RDELA
00066 FA67 FF A085               STX     DELA1
00067 FA6A FF A087               STX     DELA2
```

```
00068 FA6D 7F A08D          CLR     LCFLG    CLEAR LOWER CORNER FLAG
00069 FA70 7F A08E          CLR     UCFLG    CLEAR UPPER CORNER FLAG
00070 FA73 7F A08F          CLR     UPCNR    SET UP UPPER CORNER
00071 FA76 F6 A080          LDA B   CORNR    *
00072 FA79 F7 A090          STA B   UPCNR+1  *
00073 FA7C FE A097          LDX     PLSCTR
00074 FA7F B6 A080          LDA A   CORNR    DETERMINE LOWER CORNER
00075 FA82 09               DEX
00076 FA83 09         RUN2  DEX
00077 FA84 4A               DEC A
00078 FA85 26 FC            BNE     RUN2
00079 FA87 FF A091          STX     LWCNR
00080 FA8A 86 3C            LDA A   #$3C     SET CB2 LINE HIGH
00081 FA8C B7 890B          STA A   RUNPIA+9 DISABLE DRIVE LINES AND ARM
00082 FA8F B6 A094          LDA A   ENWRD    GET MOTOR ENABLE WORD
00083 FA92 43               COM A
00084 FA93 B7 8903          STA A   RUNPIA+1 PUT ENABLE WORD INTO LATCH
00085 FA96 86 3C            LDA A   #$3C     STROBE CA2 HIGH TO
00086 FA98 B7 890A          STA A   RUNPIA+8 LOW TO HI
00087 FA9B 86 34            LDA A   #$34     *
00088 FA9D B7 890A          STA A   RUNPIA+8 *
00089 FAA0 86 3C            LDA A   #$3C     *
00090 FAA2 B7 890A          STA A   RUNPIA+8
00091 FAA5 7F 8903          CLR     RUNPIA+1
00092 FAA8 86 34            LDA A   #$34     BRING CB2 LOW TO DISABLE
00093 FAAA B7 890B          STA A   RUNPIA+9 LATCH AND ENABLE DRIVE LINES
00094                 *
00095 FAAD 86 40            LDA A   #$40     SET DRWRD CW
00096 FAAF 7F A095          CLR     RNWRD    TO SET UP LIMIT MUX
00097 FAB2 7D A093          TST     DIRWD    CHECK DIRECTION
00098 FAB5 2A 04            BPL     RUN3     MUX CONTROLLED
00099 FAB7 7C A095          INC     RNWRD    BY PB0
00100 FABA 48               ASL A            MAKE DRWRD CCW
00101 FABB B7 A096    RUN3  STA A   DRWRD    SET DRWRD
00102 FABE B6 A095          LDA A   RNWRD    SET LIMIT MUX
00103 FAC1 B7 8903          STA A   RUNPIA+1 NO PULSE
00104 FAC4 FE A097    RUN4  LDX     PLSCTR   ALL PULSES SENT OUT?
00105 FAC7 09               DEX
00106 FAC8 FF A097          STX     PLSCTR
00107 FACB 26 03            BNE     RUN5     FINISHED
00108 FACD 7E FA35          JMP     RESET1
00109 FAD0 BD FB70    RUN5  JSR     CLIM     CHECK FOR LIMIT
00110 FAD3 24 01            BCC     RUN6
00111                 *     OUTPUT FAIL MESSAGE IF DESIRED
00112 FAD5 39               RTS
00113                 *
00114 FAD6 FE A085    RUN6  LDX     DELA1    DELAY BEFORE SETTING A 1
00115 FAD9 BD F024          JSR     DMICRO
00116 FADC B6 8903          LDA A   RUNPIA+1
00117 FADF B8 A096          EOR A   DRWRD    CREAT L.T.H. XITION
00118 FAE2 B7 8903          STA A   RUNPIA+1
00119 FAE5 FE A087          LDX     DELA2
00120 FAE8 BD F024          JSR     DMICRO   DELAY BEFORE SETTING A 0
00121 FAEB B6 8903          LDA A   RUNPIA+1
00122 FAEE B8 A096          EOR A   DRWRD    CREATE H.T.L. XITION
00123 FAF1 B7 8903          STA A   RUNPIA+1
00124                 *
00125 FAF4 FE A091          LDX     LWCNR    AT LOWER CORNER NOW?
00126 FAF7 BC A097          CPX     PLSCTR
00127 FAFA 26 03            BNE     RUN7     NOT THERE
00128 FAFC 7C A08D          INC     LCFLG    AT LOWER CORNER NOW?
00129 FAFF FE A08F    RUN7  LDX     UPCNR    AT UPPER CORNER NOW?
00130 FB02 BC A097          CPX     PLSCTR
00131 FB05 26 03            BNE     RUN8     NOT THERE
00132 FB07 7C A08E          INC     UCFLG    AT UPPER CORNER NOW
00133 FB0A B6 A08D    RUN8  LDA A   LCFLG    PAST LOWER CORNER?
00134 FB0D 26 2C            BNE     RUN9     NO
00135 FB0F B6 A08D          LDA A   LCFLG    DELAY TO EQUAL LOOP
00136 FB12 B6 A08D          LDA A   LCFLG    *
00137 FB15 F6 A086          LDA B   DELA1+1  ADJUST DELAY FOR RAMP
00138 FB18 B6 A085          LDA A   DELA1    *
00139 FB1B F0 A082          SUB B   DDLA1+1  *
00140 FB1E B2 A081          SBC A   DDLA1    *
```

```
00141 FB21 F7 A086          STA B  DELA1+1  *
00142 FB24 B7 A085          STA A  DELA1    *
00143 FB27 F6 A088          LDA B  DELA2+1  *
00144 FB2A B6 A087          LDA A  DELA2    *
00145 FB2D F0 A082          SUB B  DDLA1+1  *
00146 FB30 B2 A081          SBC A  DDLA1    *
00147 FB33 F7 A088          STA B  DELA2+1
00148 FB36 B7 A087          STA A  DELA2    *
00149 FB39 20 89            BRA    RUN4     RETURN TO TOP OF LOOP
00150 FB3B B6 A08E RUN9     LDA A  UCFLG    PAST UPPER CORNER?
00151 FB3E 27 27            BEQ    RUNA
00152 FB40 B6 A085          LDA A  DELA1    ADJUST FOR RAMP DOWN
00153 FB43 F6 A086          LDA B  DELA1+1  *
00154 FB46 FB A084          ADD B  DDLA2+1
00155 FB49 B9 A083          ADC A  DDLA2    *
00156 FB4C B7 A085          STA A  DELA1    *
00157 FB4F F7 A086          STA B  DELA1+1
00158 FB52 B6 A087          LDA A  DELA2    *
00159 FB55 F6 A088          LDA B  DELA2+1  *
00160 FB58 FB A084          ADD B  DDLA2+1
00161 FB5B B9 A083          ADC A  DDLA2    *
00162 FB5E B7 A087          STA A  DELA2    *
00163 FB61 F7 A088          STA B  DELA2+1  *
00164 FB64 7E FAC4          JMP    RUN4
00165 FB67 CE 0049 RUNA     LDX    #73
00166 FB6A BD F024          JSR    DMICRO   EQUALIZE LOOPS
00167 FB6D 7E FAC4          JMP    RUN4
00168                  ********************************************
00169                  *
00170                  *      CHECK FOR LIMIT
00171                  *
00172                  *      AT EXIT
00173                  *      CARRY CLEAR - NO ENABLED MOTORS AT LIMIT
00174                  *      CARRY SET - AT LEAST 1 ENABLED MOTOR AT LIMI
00175                  *      LIMFL - INDICATES WHICH ENABLED MOTORS HAVE
00176                  *              REACHED THEIR LIMITS
00177                  *
00178                  *
00179 FB70 7F A099 CLIM     CLR    LIMFL    CLEAR LIMIT FLAG
00180 FB73 B6 8902          LDA A  RUNPIA   LOAD LIMIT STATUS
00181 FB76 B4 A094          AND A  ENWRD    MASK UNENABLED MOTORS
00182 FB79 0C               CLC             CLEAR CARRY FOR NO LIMIT
00183 FB7A 27 04            BEQ    CLIM1
00184 FB7C B7 A099          STA A  LIMFL    SET LIMIT STATUS
00185 FB7F 0D               SEC             SET CARRY TO SIGNAL LIMIT
00186 FB80 39          CLIM1  RTS             RETURN
00188                  ********************************************
00189                  *
00190                  *      RETRACT ALL SPECIFIED MOTORS, I.E. DRIVE TO
00191                  *      CCW LIMIT.
00192                  *
00193                  *      TO ENTER :
00194                  *      ACC A CONTAINS MOTORS TO BE RETRACTED.
00195                  *
00196 FB81 B7 A094 RETR     STA A  ENWRD
00197 FB84 BD FBB0 RETR1    JSR    CCWLM    DRIVE TO CCW LIMIT
00198 FB87 B6 A094          LDA A  ENWRD    DISABLE MOTOR AT LIMIT
00199 FB8A 73 A099          COM    LIMFL
00200 FB8D B4 A099          AND A  LIMFL    *
00201 FB90 B7 A094          STA A  ENWRD    *
00202 FB93 26 EF            BNE    RETR1    DRIVE REMAINING MOTORS
00203 FB95 39               RTS
00204                  ********************************************
00205                  *
00206                  *      INSERT ALL SPECIFIED MOTORS, I.E. DRIVE
00207                  *      TO CW LIMIT.
00208                  *
00209 FB96 B7 A094 INSR     STA A  ENWRD
00210 FB99 BD FBAB INSR1    JSR    CWLM     DRIVE TO CLOCKWISE LIMIT
00211 FB9C B6 A094          LDA A  ENWRD
00212 FB9F 73 A099          COM    LIMFL    *
00213 FBA2 B4 A099          AND A  LIMFL    *
00214 FBA5 B7 A094          STA A  ENWRD    *
```

```
00215 FBA8 26 EF          BNE    INSP1     DRIVE TO NEXT LIMIT
00216 FBAA 39             RTS
00217                *
00218                *     DRIVE TO CLOCKWISE OR COUNTERCLOCKWISE LIMIT
00219 FBAB 7F A093 CWLM   CLR    DIRWD     SET DIRECTION OF MOTION
00220 FBAE 20 05          BRA    DLIM      CLOCKWISE
00221 FBB0 86 80   CCWLM  LDA A  #$80      CCW
00222 FBB2 B7 A093        STA A  DIRWD
00223 FBB5 B6 A094 DLIM   LDA A  ENWRD     GET MOTORS TO ENABLE
00224 FBB8 CE 0960        LDX    #2400     # OF PULSES
00225 FBBB FF A097        STX    PLSCTR
00226 FBBE BD FA5B        JSR    RUN1
00227 FBC1 24 F2          BCC    DLIM      NO MOTORS AT LIMITS
00228 FBC3 39             RTS
00230                ********************************************
00231                *
00232                *     READ SETPOINT
00233                *
00234                *     TO ENTER :
00235                *     ACC A - CONTAINS CHANNEL (0-7) TO BE READ
00236                *
00237                *     AT EXIT :
00238                *     VALUE READ IS 2'S COMP.
00239                *     ACC A - CONTAINS MS BYTE OF VALUE READ
00240                *     ACC B - CONTAINS LS BYTE OF VALUE READ
00241                *
00242 FBC4 B7 8901 SETPT  STA A  ADPIA+1   SELECT CHANNEL
00243 FBC7 B6 8908 SETPT1 LDA A  ADPIA+8   WAIT FOR EOC
00244 FBCA 2A FB          BPL    SETPT1
00245 FBCC B6 8900        LDA A  ADPIA     READ DATA
00246 FBCF F6 8901        LDA B  ADPIA+1
00247 FBD2 37             PSH B
00248 FBD3 C4 07          AND B  #$07      SAVE CHANNEL NUMBER
00249 FBD5 F7 A09A        STA B  CHNSAV
00250 FBD8 33             PUL B
00251 FBD9 43             COM A            COMPLEMENT VALUES READ
00252 FBDA 53             COM B
00253 FBDB C4 F8          AND B  #$F8
00254 FBDD FA A09A        ORA B  CHNSAV    ADD CHANNEL NUMBER
00255 FBE0 39             RTS
00256                *
00258 A080                ORG    $A080
00259 A080 0001    COPNP  RMB    1
00260 A081 0002    IDLA1  RMB    2
00261 A083 0002    IDLA2  RMB    2
00262 A085 0002    IELA1  RMB    2
00263 A087 0002    IELA2  RMB    2
00264 A089 0002    RDILA  RMB    2
00265 A08B 0002    RIELA  RMB    2
00266 A08D 0001    LCFLG  RMB    1
00267 A08E 0001    UCFLG  RMB    1
00268 A08F 0002    UPCNR  RMB    2
00269 A091 0002    LWCNR  RMB    2
00270 A093 0001    DIRWD  RMB    1
00271 A094 0001    ENWRD  RMB    1
00272 A095 0001    RNWRD  RMB    1
00273 A096 0001    DRWRD  RMB    1
00274 A097 0002    PLSCTR RMB    2
00275 A099 0001    LIMFL  RMB    1
00276 A09A 0001    CHNSAV RMB    1
00277                *
00278                      END

TOTAL ERRORS 00000

!
```

LISTING 2

LASL Identification No. LP 1068

```
00001                          NAM    DEMO
00002                    *
00003                    *     STEPPING MOTOR DEMO PROGRAM
00004                    *
00005                    *     BUTTON 1 - RUN MOTOR CLOCKWISE
00006                    *     BUTTON 2 - RUN MOTOR COUNTERCLOCKWISE
00007                    *     BUTTON 3 - INSERT MOTOR
00008                    *     BUTTON 4 - RETRACT MOTOR
00009                    *     BUTTON 5 - DISPLAY SETPOINT
00010                    *     BUTTON 6 - RUN MOTOR CW WHILE PB HELD
00011                    *     BUTTON 7 - RUN MOTOR CCW WHILE PB HELD
00012                    *     BUTTON 8 - CYCLE MOTOR
00013  A10E    PUSHB   EQU    $A10E
00014  F021    DMILLI  EQU    $F021
00015  FA00    STMOTOR EQU    $FA00     STEPPING MOTOR ROUTINE
00016  A1F0    VSUB1   EQU    $A1F0
00017  F01E    PSX     EQU    $F01E
00018  A1E0    VCTR1   EQU    $A1E0
00019  A100    HEXLD   EQU    $A100
00020  A102    HEXSW   EQU    $A102
00021  A104    TOGGL   EQU    $A104
00022  FA00    RESET   EQU    STMOTOR
00023  FA03    RUN     EQU    STMOTOR+3
00024  FA09    INSERT  EQU    STMOTOR+9
00025  FA0C    SETPT   EQU    STMOTOR+12
00026  FA06    RETRCT  EQU    STMOTOR+6
00027  F01B    IPANEL  EQU    $F01B
00028                    *
00029  F800              ORG    $F800
00030                    *     INITIALIZE PROGRAM LOOP
00031  F800 BD F01B INIT  JSR    IPANEL    INITIALIZE FRONT PANEL
00032  F803 CE F850      LDX    #RUNCW
00033  F806 FF A1E0      STX    VCTR1     1=RUN CLOCKWISE
00034  F809 CE F851      LDX    #RUNCCW
00035  F80C FF A1E2      STX    VCTR1+2   2=RUN COUNTERCLOCKWISE
00036  F80F CE F87E      LDX    #INSERT
00037  F812 FF A1E4      STX    VCTR1+4   3=INSERT MOTOR
00038  F815 CE F888      LDX    #RETRAC
00039  F818 FF A1E6      STX    VCTR1+6   4=RETRACT MOTOR
00040  F81B CE F892      LDX    #RDISP
00041  F81E FF A1E8      STX    VCTR1+8   5=READ SETPOINT
00042  F821 CE F8C9      LDX    #DURCW
00043  F824 FF A1EA      STX    VCTR1+10  6=RUN CW IF PUSHED
00044  F827 CE F8C2      LDX    #DURCCW
00045  F82A FF A1EC      STX    VCTR1+12  7=RUN CCW IF PUSHED
00046  F82D CE F8E1      LDX    #CYCLE
00047  F830 FF A1EE      STX    VCTR1+14  8=CYCLE MOTOR
00048  F833 CE 3C3C      LDX    #$3C3C    RECONFIGURE PUSHBUTTONS
00049  F836 FF 810E      STX    PUSHB+8   BRING CA2+CB2 HIGH
00050  F839 86 9F        LDA A  #$9F      DURATION OF I-P
00051  F83B B7 8107      STA A  PUSHB+1
00052  F83E CE 3C34      LDX    #$3C34    LATCH IN MODES
00053  F841 FF 810E      STX    PUSHB+8
00054  F844 7F 8107      CLR    PUSHB+1   TURN OFF LIGHTS
00055  F847 BD FA00      JSR    RESET
00056  F84A CE F01E      LDX    #PSX      LOOP IS FLASHING LIGHTS
00057  F84D FF A1F0      STX    VSUB1
00058  F850 39           PTS
00060                    ********************************************
00061                    *
00062                    *     RUN CLOCKWISE OR COUNTERCLOCKWISE
00063                    *     BUTTON 1 OR 2
00064                    *
00065                    *     READ NUMBER OF COUNTS FROM HEXSWITCH.
00066                    *     SELECT MOTORS FROM TOGGLE SWITCHES
00067                    *
00068  F851 86 01  RUNCCW LDA A  #1
```

```
00069 F853 B7 A053          STA A  MINUS     SIGNAL CCW
00070 F856 20 03            BRA    PUNCW1
00071 F858 7F A053 PUNCW    CLR    MINUS     SIGNAL CW
00072 F85B B6 8102 PUNCW1   LDA A  HEXSW     READ # OF COUNTS
00073 F85E F6 8103          LDA B  HEXSW+1
00074 F861 BD F90C          JSR    CV4DB     CONVERT TO BINARY
00075 F864 7D A053          TST    MINUS     NEGATIVE COUNTS?
00076 F867 27 02            BEQ    PUNCW2
00077 F869 8A 80            ORA A  #$80      SET MS BIT IF NEG.
00078 F86B B7 A051 PUNCW2   STA A  COUNT     SAVE BINARY COUNTS
00079 F86E F7 A052          STA B  COUNT+1
00080 F871 B6 8104 PUNM     LDA A  TOGGL     GET MOTORS TO ENABLE
00081 F874 FE A051          LDX    COUNT     GET # OF COUNTS
00082 F877 BD FA03          JSR    PUN       RUN MOTORS
00083 F87A BD F892          JSR    PDSPT     OUTPUT STPT OF 1ST MOTOR ENA
00084 F87D 39               RTS
00085                  ********************************************
00086                  *
00087                  *       INSERT OR RETRACT MOTORS
00088                  *       BUTTON 3 OR 4
00089                  *
00090                  *       SELECT MOTORS FROM TOGGLE SWITCHES.
00091                  *
00092 F87E B6 8104 INSERT LDA A  TOGGL     GET MOTORS
00093 F881 BD FA09        JSR    INSEP     INSERT THEM
00094 F884 BD F892        JSR    PDSPT     READ SETPOINT
00095 F887 39             RTS
00096                  *
00097 F888 B6 8104 RETRAC LDA A  TOGGL     GET MOTORS
00098 F88B BD FA06        JSR    RETRCT    RETRACT THEM
00099 F88E BD F892        JSR    PDSPT     READ SETPOINT
00100 F891 39             RTS
00101                  ********************************************
00102                  *
00103                  *       READ MOTOR SETPOINT
00104                  *       BUTTON 5
00105                  *
00106                  *       DISPLAY SETPOINT OF FIRST MOTOR
00107                  *       ENABLED ON HEX LEDS.
00108                  *
00109 F892 F6 8104 PDSPT  LDA B  TOGGL     GET DATA FROM TOGGLES
00110 F895 4F             CLR A            ENCODE CHANNEL NO.
00111 F896 54          PDSP1 LSR B         *
00112 F897 25 05          BCS    PDSP5     *
00113 F899 4C             INC A            *
00114 F89A 5D             TST B            *
00115 F89B 26 F9          BNE    PDSP1     *
00116 F89D 39             RTS              NO CHANNEL SPECIFIED
00117                  *
00118 F89E 36          PDSP5  PSH A
00119 F89F CE 0028        LDX    #40
00120 F8A2 BD F021        JSR    DMILLI    DELAY BEFORE READING
00121 F8A5 32             PUL A
00122 F8A6 BD FA0C        JSR    SETPT     READ SETPOINT
00123 F8A9 37             PSH B
00124 F8AA C4 07          AND B  #$07      ABSTRACT CHANNEL NO.
00125 F8AC F7 A050        STA B  CHAN      SAVE IT
00126 F8AF 33             PUL B
00127 F8B0 44             LSR A            SHIFT TO RIGHT
00128 F8B1 56             ROR B            JUSTIFY 12 BIT
00129 F8B2 44             LSR A            2'S COMPLEMENT
00130 F8B3 56             ROR B            NUMBER
00131 F8B4 44             LSR A            *
00132 F8B5 56             ROR B            *
00133 F8B6 44             LSR A            *
00134 F8B7 56             ROR B            *
00135 F8B8 BD F950        JSR    CVB4D     CONVERT TO BCD
00136 F8BB B7 8100        STA A  HEXLD     DISPLAY IN HEX LEDS
00137 F8BE F7 8101        STA B  HEXLD+1   *
00138 F8C1 39             RTS
00139                  ********************************
00140                  *
00141                  *       DURATION OF PUSH
```

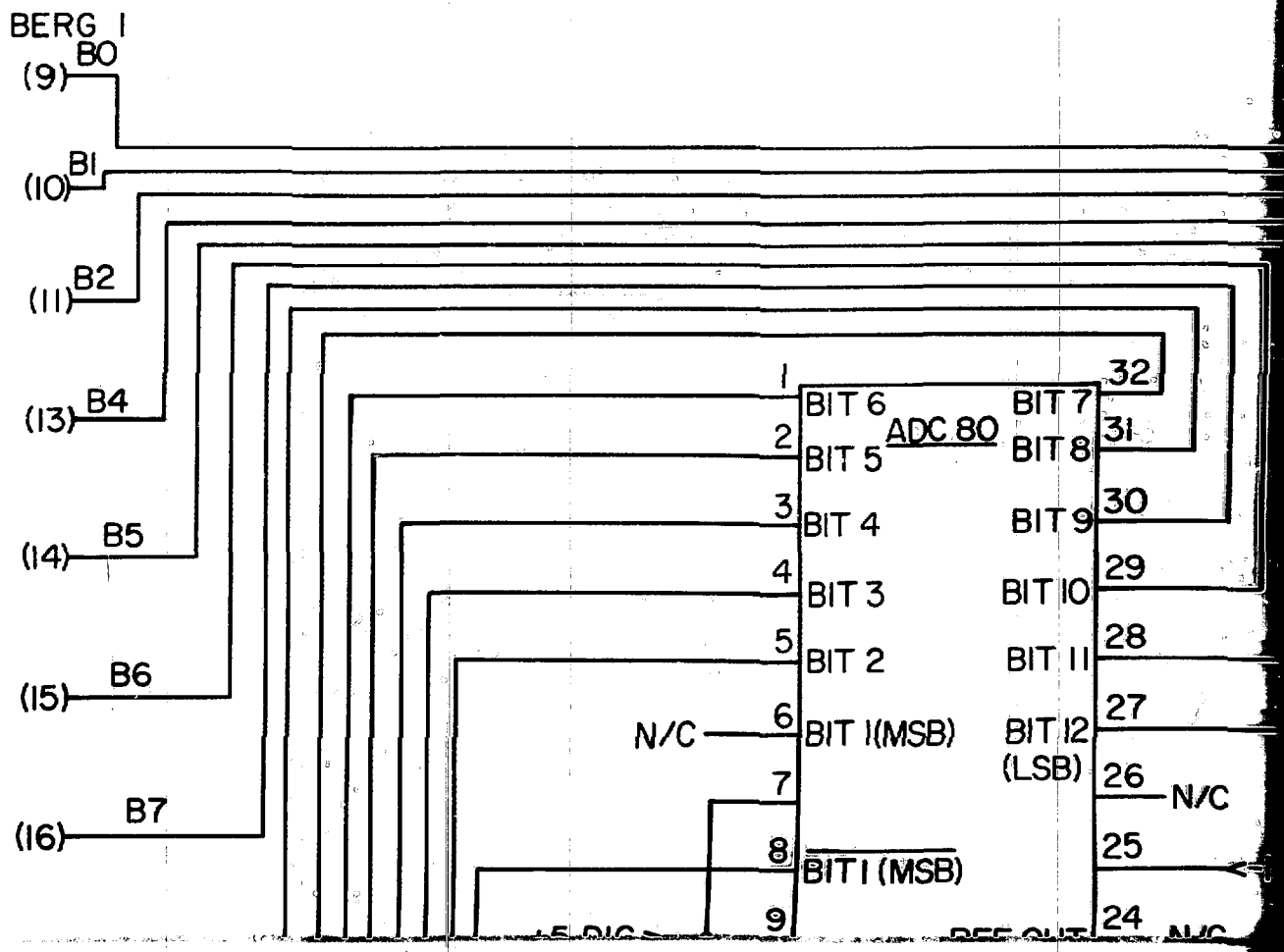
```
                    *       BUTTON 6 OR 7
                    *
                    *       RUN MOTOR CW OR CCW WHILE BUTTON
                    *       IS PUSHED.
                    *
F8C2 86 01    DUPCCW LDA A  #1
F8C4 B7 A053          STA A  MINUS    SIGNAL CCW
F8C7 20 03            BRA    DUPCW1
F8C9 7F A053  DUPCW   CLR    MINUS    SIGNAL CW
F8CC CE 0001  DUPCW1  LDX    #1       PULSE TO SEND
F8CF FF A051          STX    COUNT
F8D2 7D A053          TST    MINUS    CW OR CCW
F8D5 27 06            BEQ    DUPCW2
F8D7 4F               CLR A
F8D8 8A 80            ORA A  #$80     SET MS BIT IF NEG
F8DA B7 A051          STA A  COUNT
F8DD BD F871  DUPCW2  JSR    RUNM     RUN MOTOR AND DISPLAY SETPOI
F8E0 39               RTS
                    ********************************
                    *
                    *       CYCLE
                    *       BUTTON 8
                    *
                    *       DRIVE MOTOR CW AND CCW NUMBER
                    *       OF COUNTS SPECIFIED IN HEXSWITCH
                    *       UNTIL BUTTON IS PUSHED AGAIN.
                    *
F8E1 B6 8102  CYCLE   LDA A  HEXSW    READ # OF COUNTS
F8E4 F6 8103          LDA B  HEXSW+1
F8E7 BD F90C          JSR    CV4DB    CONVERT TO BINARY
F8EA F7 A052          STA B  COUNT+1
F8ED B7 A051  CYCLE1  STA A  COUNT
F8F0 BD F871          JSR    RUNM     RUN ONE WAY
F8F3 F6 8106          LDA B  PUSHB
F8F6 C5 80            BIT B  #$80     LAST BUTTON PUSHED AGAIN?
F8F8 26 07            BNE    CYCLE2
F8FA B6 A051          LDA A  COUNT
F8FD 88 80            EOR A  #$80     SIGNAL OPPOSITE DIRECTION
F8FF 20 EC            BRA    CYCLE1
F901 86 34    CYCLE2  LDA A  #$34     RESET PUSHBUTTON
F903 B7 810E          STA A  PUSHB+8
F906 86 3C            LDA A  #$3C
F908 B7 810E          STA A  PUSHB+8
F90B 39               RTS
                    ************************************************
                    *
                    *       CONVERT 4 DIGIT DECIMAL TO BINARY
                    *
                    *       12 BIT BINARY--11 BITS+SIGN
                    *
                    *       TO ENTER:
                    *       A = 2 MS DECIMAL DIGITS
                    *       B = 2 LS DECIMAL DIGITS
                    *
                    *       AT EXIT:
                    *       A = MS BYTE OF CONVERTED NUMBER
                    *       B = LS BYTE OF CONVERTED NUMBER
                    *
F90C 7F A055  CV4DB   CLR    BINUPP
F90F B7 A054          STA A  SAVEA    SAVE MS DIGITS
F912 17               TBA
F913 C4 0F            AND B  #$0F     SAVE ONLY ONES VALUE
F915 44               LSR A           MOVE TENS VALUE
F916 44               LSR A           TO LOWER BITS OF A
F917 44               LSR A
F918 44               LSR A
F919 27 05    TEN     BEQ    DOHUND   DO HUND WHEN TENS=0
F91B CB 0A            ADD B  #10      ADD 10 TO TOTAL
F91D 4A               DEC A           DECREMENT 10S DIGIT
F91E 20 F9            BRA    TEN
F920 0C       DOHUND  CLC             RESET CARRY
F921 B6 A054          LDA A  SAVEA    GET HUND AND THOU DIGIT
F924 84 0F            AND A  #$0F     SAVE 100S DIGITS
```

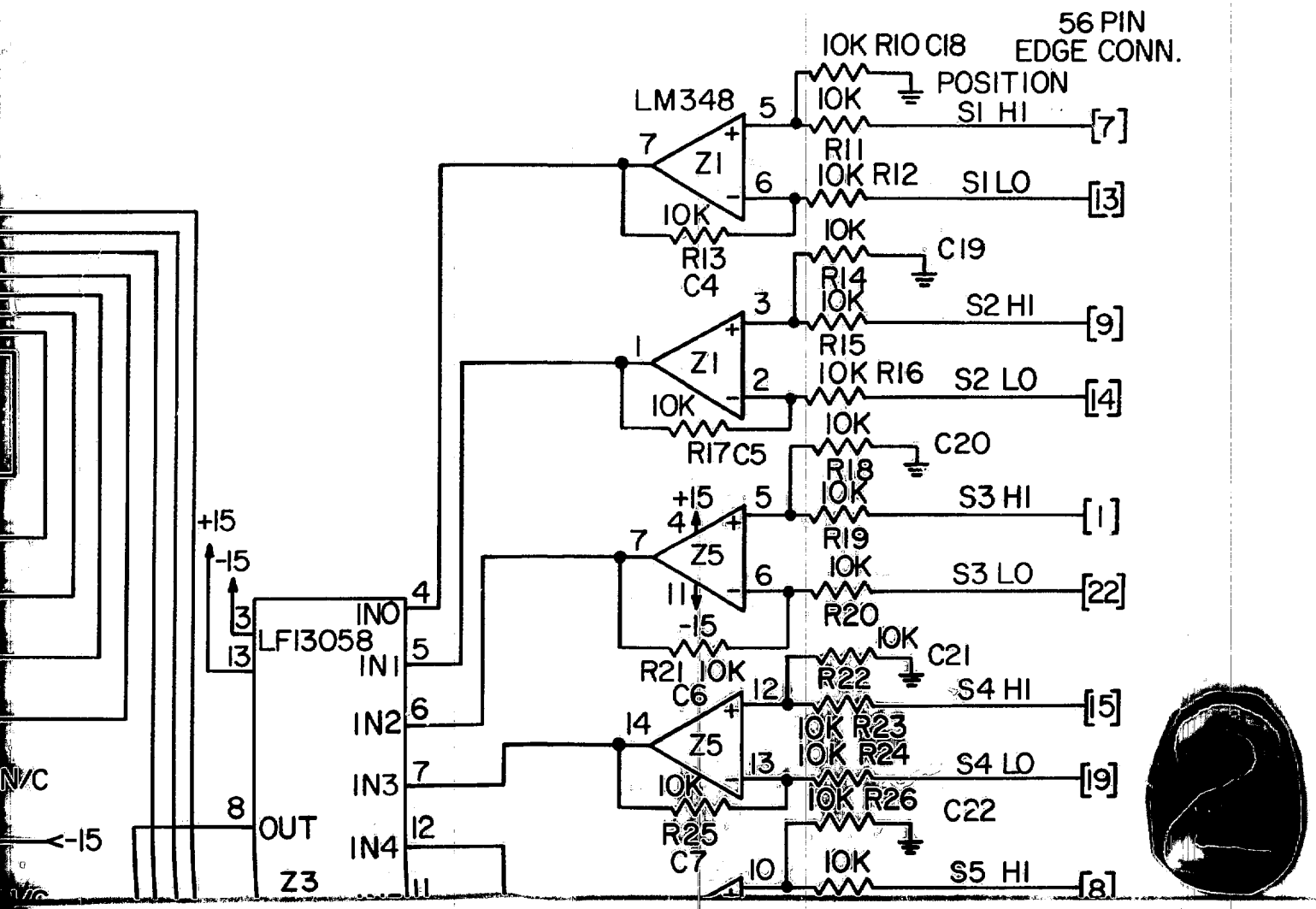
```
00216 F92C 27 0A      HUN1    BEQ     DOTHOU    DO 1000'S WHEN 1000'S=0
00217 F92E CB E4              ADD B   #100     ADD 100 TO TOTAL
00218 F930 24 02              BCC     HUN2
00219 F932 7C A055            INC     BINUPR    ADD 256 TO TOTAL
00220 F935 4A         HUN2    DEC A             DECREMENT 100'S DIGIT
00221 F936 20 F4              BPA     HUN1
00222 F938 B6 A054    DOTHOU  LDA A   SAVER     GET 1000'S AND 100'S DIGIT
00223 F93B 44                 LSR A             MOVE 100'S TO LOWER
00224 F93C 44                 LSR A             4 BITS
00225 F93D 44                 LSR A
00226 F93E 44                 LSR A
00227 F93F B7 A054            STA A   SAVER
00228 F942 27 0E              BEQ     THOU1     1000'S=0?
00229 F944 B6 A055            LDA A   BINUPR
00230 F947 0C         THOU    CLC               RESET CARRY
00231 F948 CB E8              ADD B   #$E8      ADD 1000 TO TOTAL
00232 F94A 89 03              ADC A   #$03
00233 F94C 7A A054            DEC     SAVER     DECREMENT 1000'S DIGIT
00234 F94F 26 EF              BNE     THOU
00235 F951 39                 RTS
00236 F952 B6 A055    THOU1   LDA A   BINUPR
00237 F955 39                 RTS
00238                 ********************************************
00240                 *
00241                 *       CONVERT BINARY TO 4 DIGIT DECIMAL
00242                 *
00243                 *       12 BIT BINARY--11 BITS+SIGN
00244                 *
00245                 *       TO ENTER:
00246                 *         A = MS BYTE OF NUMBER TO CONVERT
00247                 *         B = LS BYTE OF NUMBER TO CONVERT
00248                 *       AT EXIT:
00249                 *         A = 2 MS DECIMAL DIGITS
00250                 *         B = 2 LS DECIMAL DIGITS
00251                 *
00252 F956 CE F9AC    CVB4D   LDX     #CONST    POINT TO 1000
00253 F959 B7 A056            STA A   NUM       SAVE BINARY DIGITS
00254 F95C F7 A057            STA B   NUM+1
00255 F95F 7F A059            CLR     DIGIT+1
00256 F962 7F A05A            CLR     DFLAG     ADD TO UPPER DIGIT
00257 F965 8D 1E              BSR     SUB       DETERMINE MS (4TH) DIGIT
00258 F967 09                 INX               POINT TO 100
00259 F968 09                 INX
00260 F969 7C A05A            INC     DFLAG     ADD TO LOWER DIGIT
00261 F96C 8D 17              BSR     SUB       DETERMINE 3RD DIGIT
00262 F96E B6 A059            LDA A   DIGIT+1
00263 F971 B7 A058            STA A   DIGIT
00264 F974 08                 INX               POINT TO 10
00265 F975 08                 INX
00266 F976 7F A059            CLR     DIGIT+1
00267 F979 7F A05A            CLR     DFLAG     ADD TO UPPER DIGIT
00268 F97C 8D 07              BSR     SUB       DETERMINE 2ND DIGIT
00269 F97E FB A059            ADD B   DIGIT+1   DETERMINE 1ST DIGIT
00270 F981 B6 A058            LDA A   DIGIT
00271 F984 39                 RTS
00272                 *
00273 F985 B6 A056    SUB     LDA A   NUM       SUBTRACT CONSTANTS
00274 F988 F6 A057            LDA B   NUM+1     GET BINARY DIGITS
00275 F98B E0 01      SUB1    SUB B   1,X       ~ SUBTRACT CONSTANTS
00276 F98D A2 00              SBC A   0,X
00277 F98F 25 16              BCS     SUB3      TOO MANY SUBTRACTIONS
00278 F991 7D A05A            TST     DFLAG     ADD TO WHICH DIGIT
00279 F994 27 05              BEQ     SUB2
00280 F996 7C A059            INC     DIGIT+1   ADD TO LOWER DIGIT
00281 F999 20 F0              BRA     SUB1
00282 F99B 36         SUB2    PSH A
00283 F99C 86 10              LDA A   #$10      ADD TO UPPER DIGIT
00284 F99E BB A059            ADD A   DIGIT+1
00285 F9A1 B7 A059            STA A   DIGIT+1
00286 F9A4 32                 PUL A
00287 F9A5 20 E4              BRA     SUB1
00288 F9A7 EB 01      SUB3    ADD B   1,X       UNDO LAST SUBTRACTION
00289 F9A9 A9 00              ADC A   0,X
```

```
00290 F9A5 B7 A056          STA A  NUM      SAVE BINARY DIGIT
00291 F9A8 F7 A057          STA B  NUM+1
00292 F9AB 39               RTS
00293                  *
00294 F9AC 03E8     CONST   FDB    1000
00295 F9AE 0064             FDB    100
00296 F9B0 000A             FDB    10
00298 A050                  ORG    $A050
00299 A050 0001     CHAN    RMB    1
00300 A051 0002     COUNT   RMB    2
00301 A053 0001     MINUS   RMB    1
00302 A054 0001     SAVEA   RMB    1
00303 A055 0001     BINUPP  RMB    1
00304 A056 0002     NUM     RMB    2
00305 A058 0002     DIGIT   RMB    2
00306 A05A 0001     DFLAG   RMB    1
00307                  *
00308                       END

TOTAL ERRORS 00000

!
```

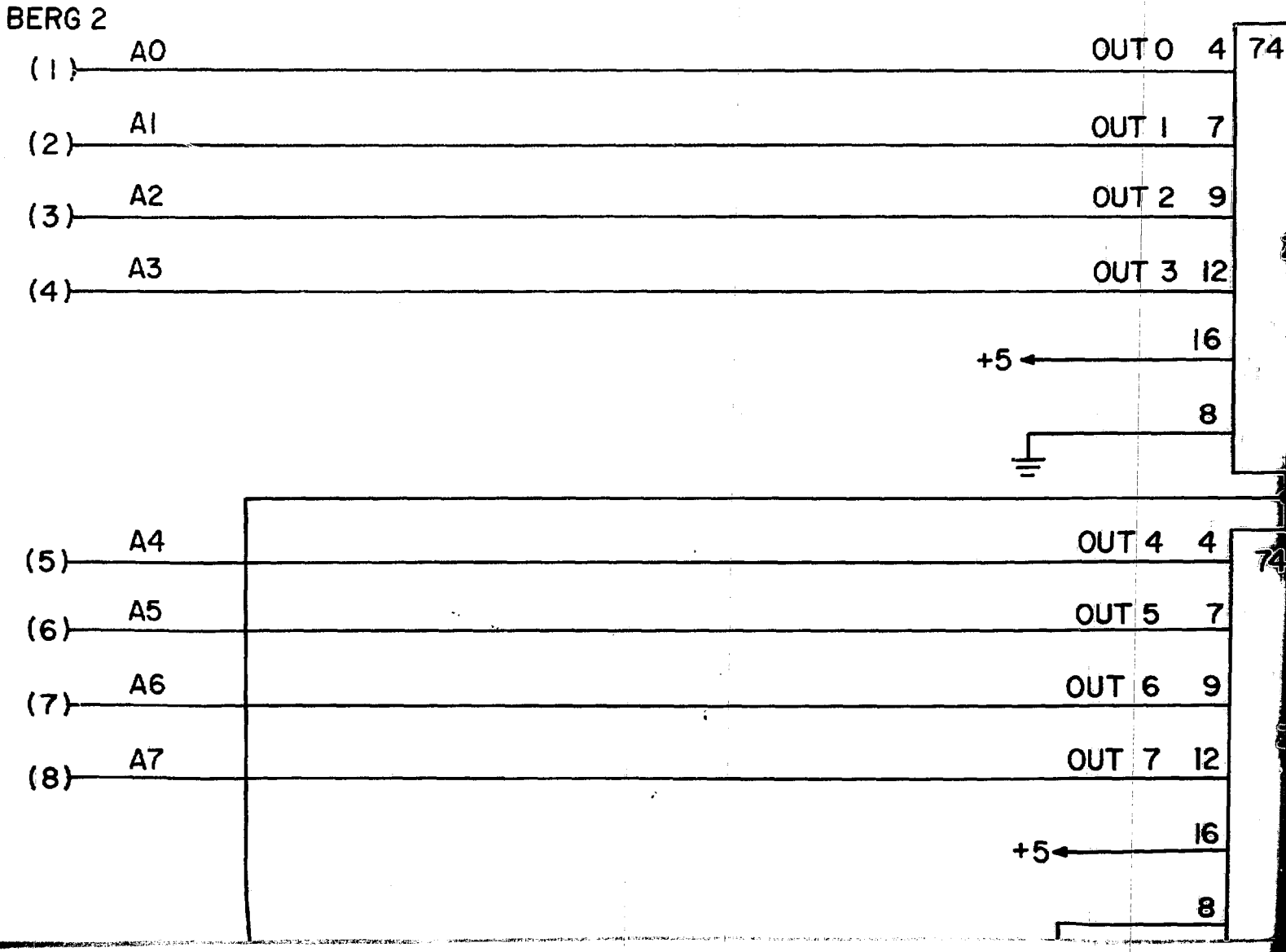

☆U.S. Government Printing Office: 1979 — 677—115/207

ANALOG

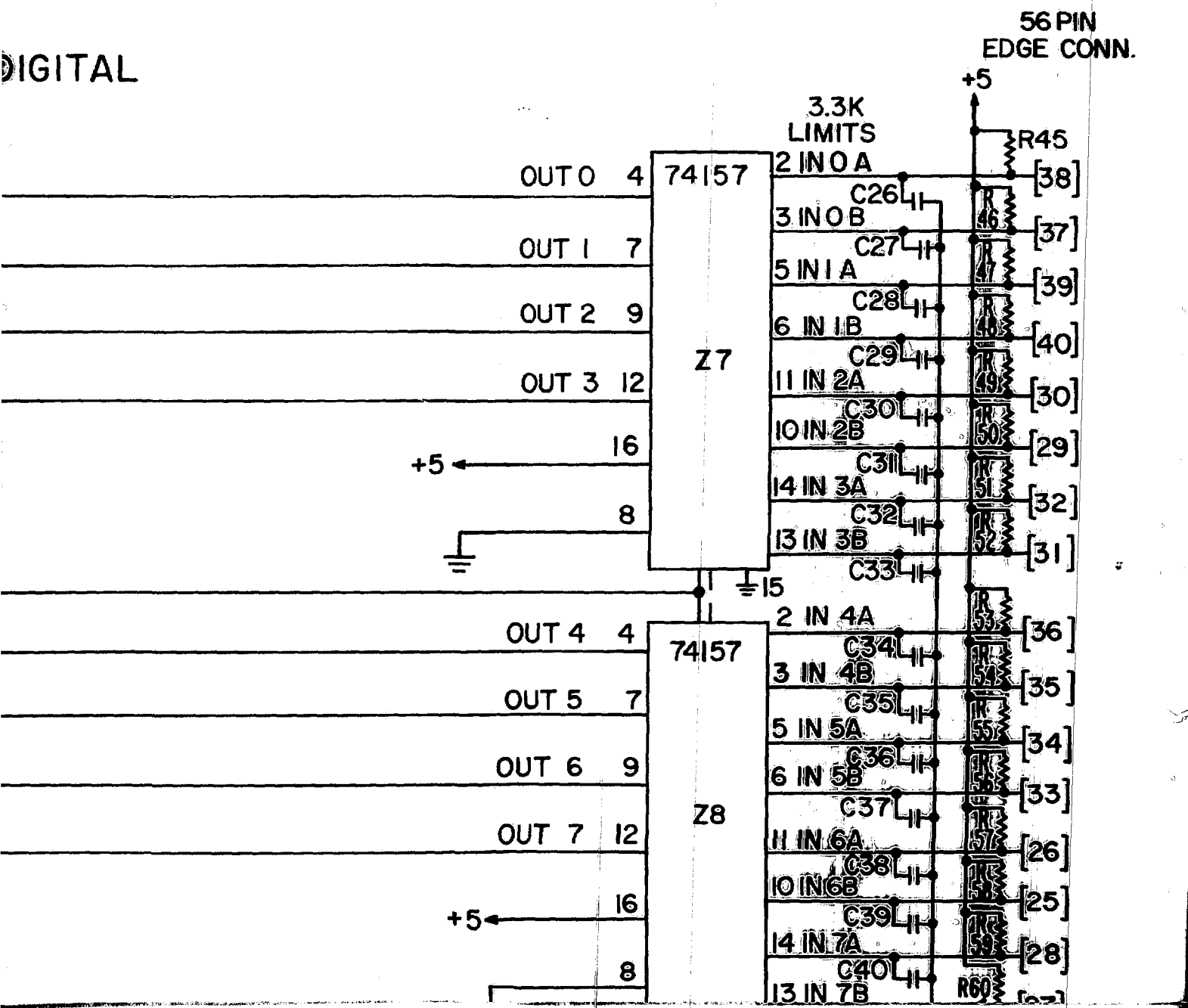

BERG 1
(9) B0
(10) B1
(11) B2
(13) B4
(14) B5
(15) B6
(16) B7
1 BIT 6
2 BIT 5
3 BIT 4
4 BIT 3
5 BIT 2
6 BIT 1(MSB)
7
8 BIT 1 (MSB)
9
ADC 80
32 BIT 7
31 BIT 8
30 BIT 9
29 BIT 10
28 BIT 11
27 BIT 12 (LSB)
26 N/C
25
24 N/C
N/C

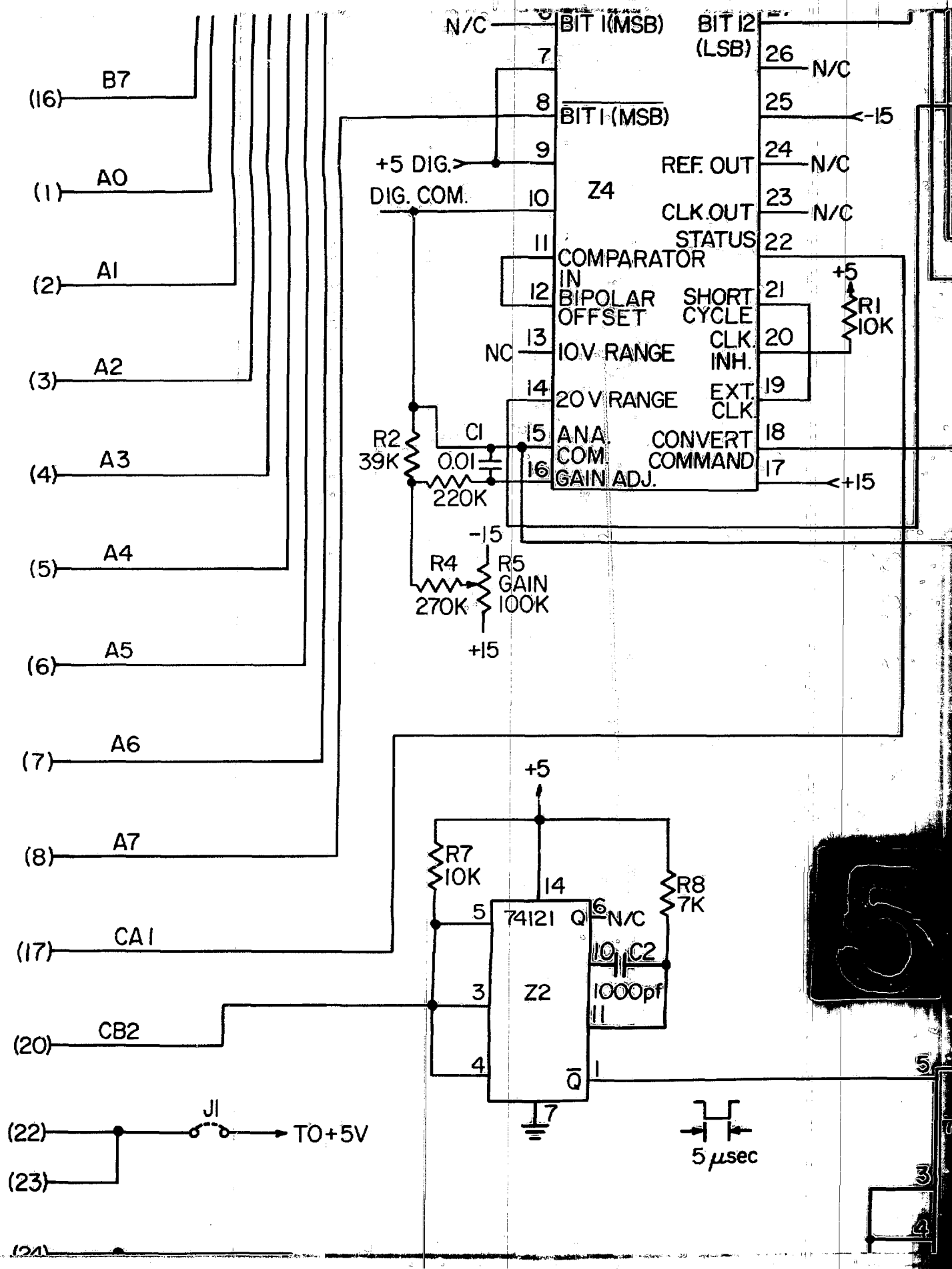

56 PIN
EDGE CONN.
10K R10 C18
POSITION
LM348
10K
S1 HI
[7]
Z1
R11
10K R12
S1 LO
[13]
10K
R13
C4
10K
C19
R14
10K
S2 HI
[9]
Z1
R15
10K R16
S2 LO
[4]
10K
R17 C5
10K
C20
R18
+15
10K
S3 HI
[1]
Z5
R19
10K
S3 LO
[22]
-15
R20
R21 10K
C6
10K
C21
R22
S4 HI
[15]
Z5
10K R23
10K R24
S4 LO
[19]
10K
R25
C7
10K R26
C22
10K
S5 HI
[8]
+15
-15
LF13058
IN0
IN1
IN2
IN3
IN4
OUT
Z3
N/C
-15

DIGITAL


BERG 2
(1) A0
OUT 0 4
74
(2) A1
OUT 1 7
(3) A2
OUT 2 9
(4) A3
OUT 3 12
+5
16
8
(5) A4
OUT 4 4
74
(6) A5
OUT 5 7
(7) A6
OUT 6 9
(8) A7
OUT 7 12
+5
16
8

DIGITAL


56 PIN
EDGE CONN.
+5
3.3K
LIMITS
R45
74157
Z7
OUT 0 4
OUT 1 7
OUT 2 9
OUT 3 12
16
+5
8
15
2 IN 0 A
3 IN 0 B
5 IN 1 A
6 IN 1 B
11 IN 2A
10 IN 2B
14 IN 3A
13 IN 3B
C26
C27
C28
C29
C30
C31
C32
C33
R46
R47
R48
R49
R50
R51
R52
[38]
[37]
[39]
[40]
[30]
[29]
[32]
[31]
74157
Z8
OUT 4 4
OUT 5 7
OUT 6 9
OUT 7 12
16
+5
8
2 IN 4A
3 IN 4B
5 IN 5A
6 IN 5B
11 IN 6A
10 IN 6B
14 IN 7A
13 IN 7B
C34
C35
C36
C37
C38
C39
C40
R53
R54
R55
R56
R57
R58
R59
R60
[36]
[35]
[34]
[33]
[26]
[25]
[28]


(16) B7
(1) A0
(2) A1
(3) A2
(4) A3
(5) A4
(6) A5
(7) A6
(8) A7
(17) CA1
(20) CB2
(22)
(23)
J1
TO +5V
N/C
BIT 1 (MSB)
BIT 12 (LSB)
N/C
BIT 1 (MSB)
-15
+5 DIG.
REF. OUT
N/C
DIG. COM.
Z4
CLK. OUT
N/C
STATUS
COMPARATOR IN
BIPOLAR OFFSET
SHORT CYCLE
+5
R1 10K
NC
10V RANGE
CLK. INH.
20V RANGE
EXT. CLK.
R2 39K
C1 0.01
ANA. COM
CONVERT COMMAND
GAIN ADJ.
220K
+15
-15
R4 270K
R5 GAIN 100K
+15
+5
R7 10K
R8 7K
74121
N/C
C2 1000pf
Z2
5 μsec

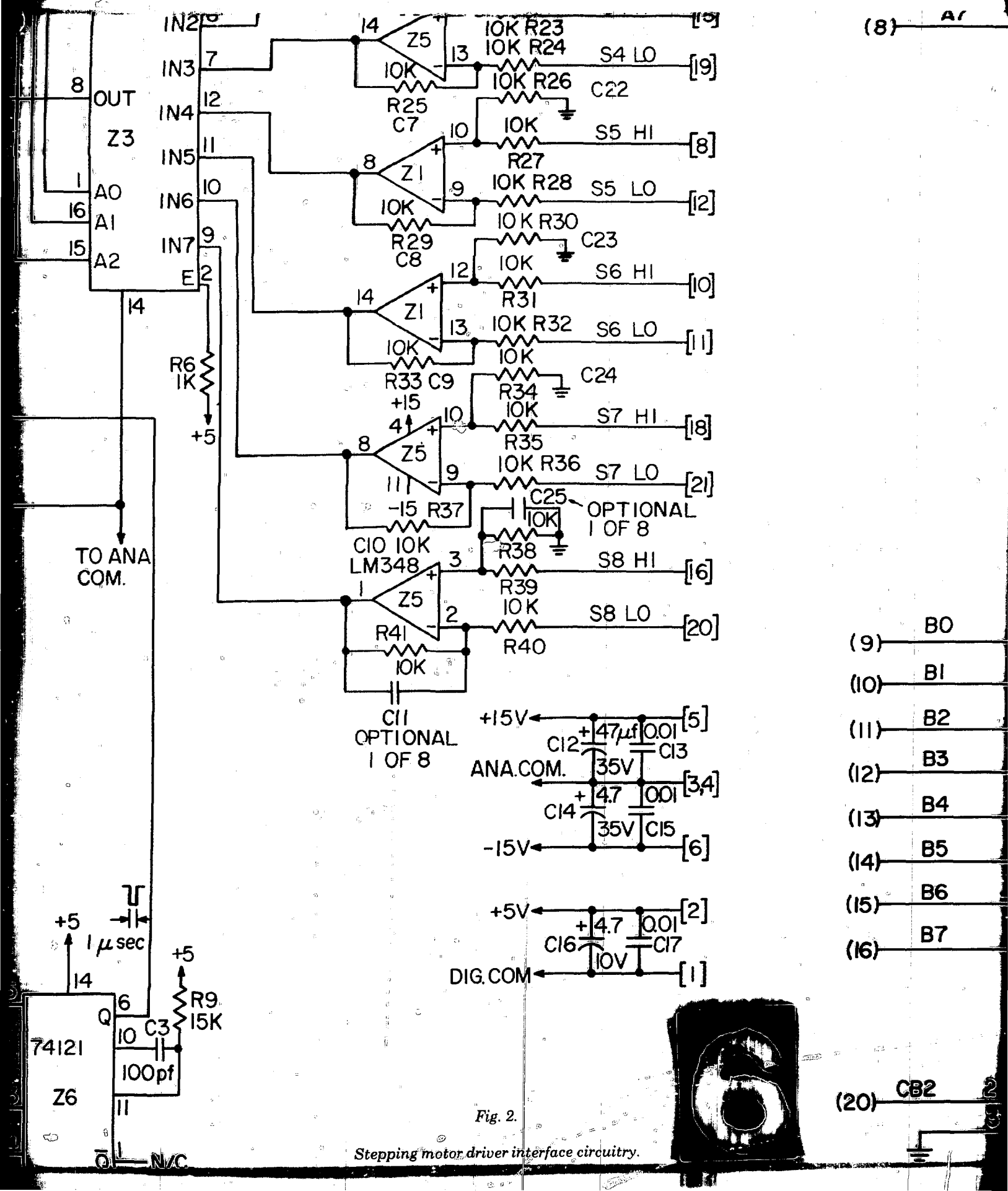
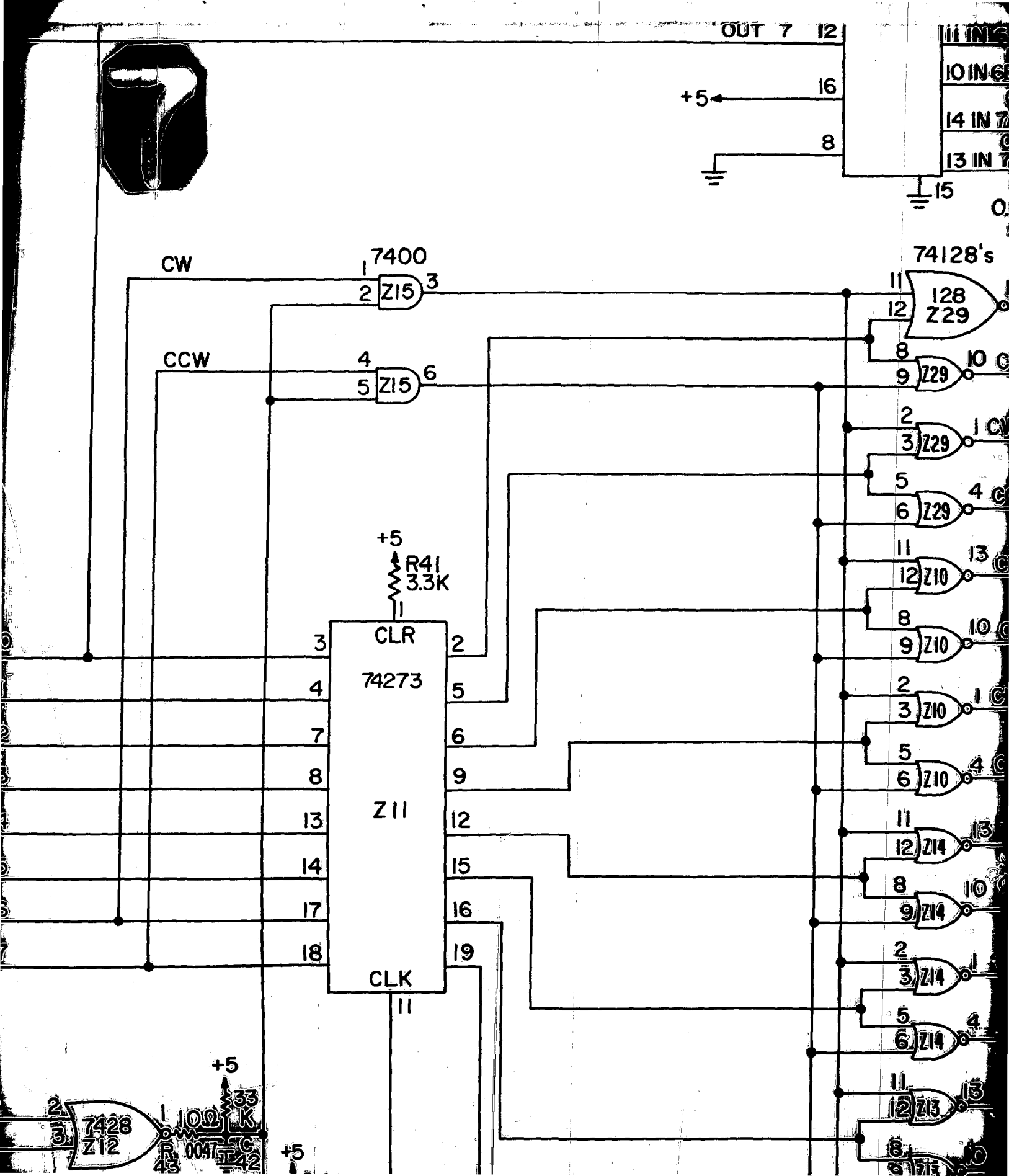
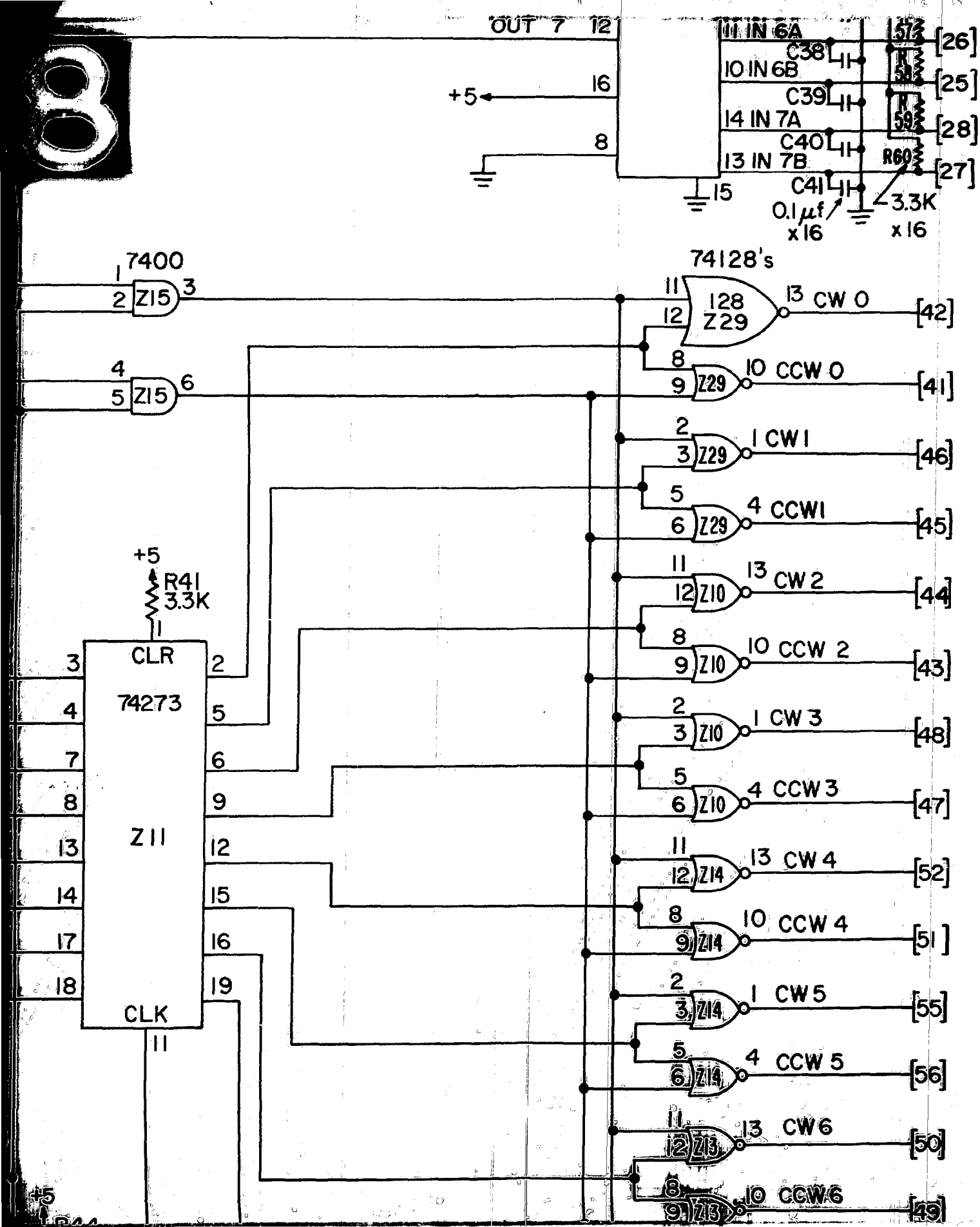
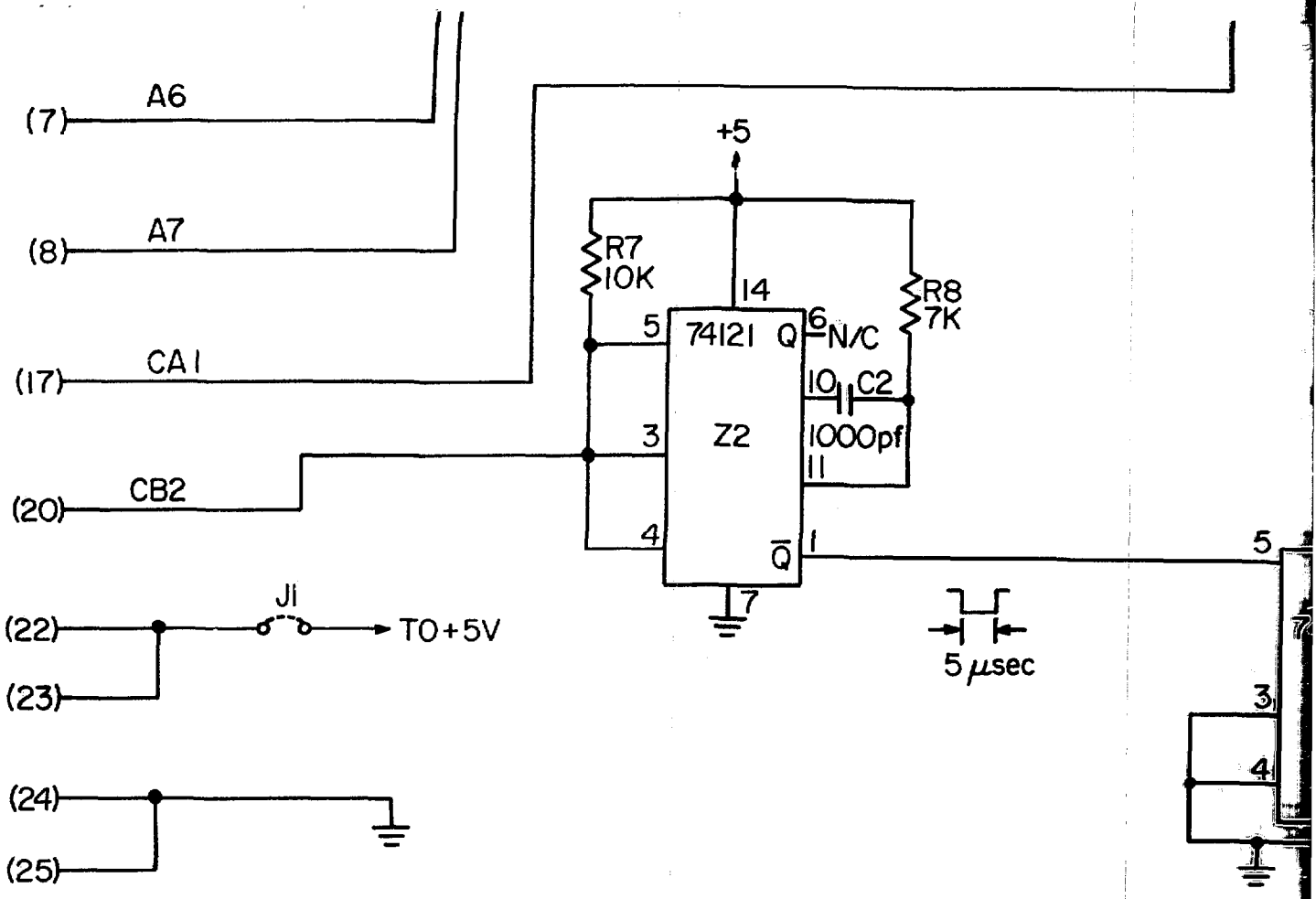


Fig. 2.

Stepping motor driver interface circuitry.


OUT 7
12
+5
16
8
15
CW
CCW
7400
Z15
74128's
128
Z29
Z10
Z14
Z13
+5
R41
3.3K
CLR
74273
Z11
CLK
7428
Z12
10Ω
33K
.0047
C42
R43
+5


OUT 7
12
+5
16
8
15
11 IN 6A
C38
10 IN 6B
C39
14 IN 7A
C40
13 IN 7B
C41
0.1 µf
x16
R60
3.3K
x16
[26]
[25]
[28]
[27]
7400
Z15
74128's
128
Z29
13 CW 0
[42]
10 CCW 0
[41]
1 CW 1
[46]
4 CCW 1
[45]
+5
R41
3.3K
CLR
74273
Z11
CLK
Z10
13 CW 2
[44]
10 CCW 2
[43]
1 CW 3
[48]
4 CCW 3
[47]
Z14
13 CW 4
[52]
10 CCW 4
[51]
1 CW 5
[55]
4 CCW 5
[56]
Z13
13 CW 6
[50]
10 CCW 6
[49]


(7) A6
(8) A7
(17) CA1
(20) CB2
(22)
(23)
J1
TO +5V
(24)
(25)
+5
R7
10K
R8
7K
14
5
74121
Q
6
N/C
10
C2
1000pf
11
3
Z2
4
Q
1
7
5 μsec
5
3
4

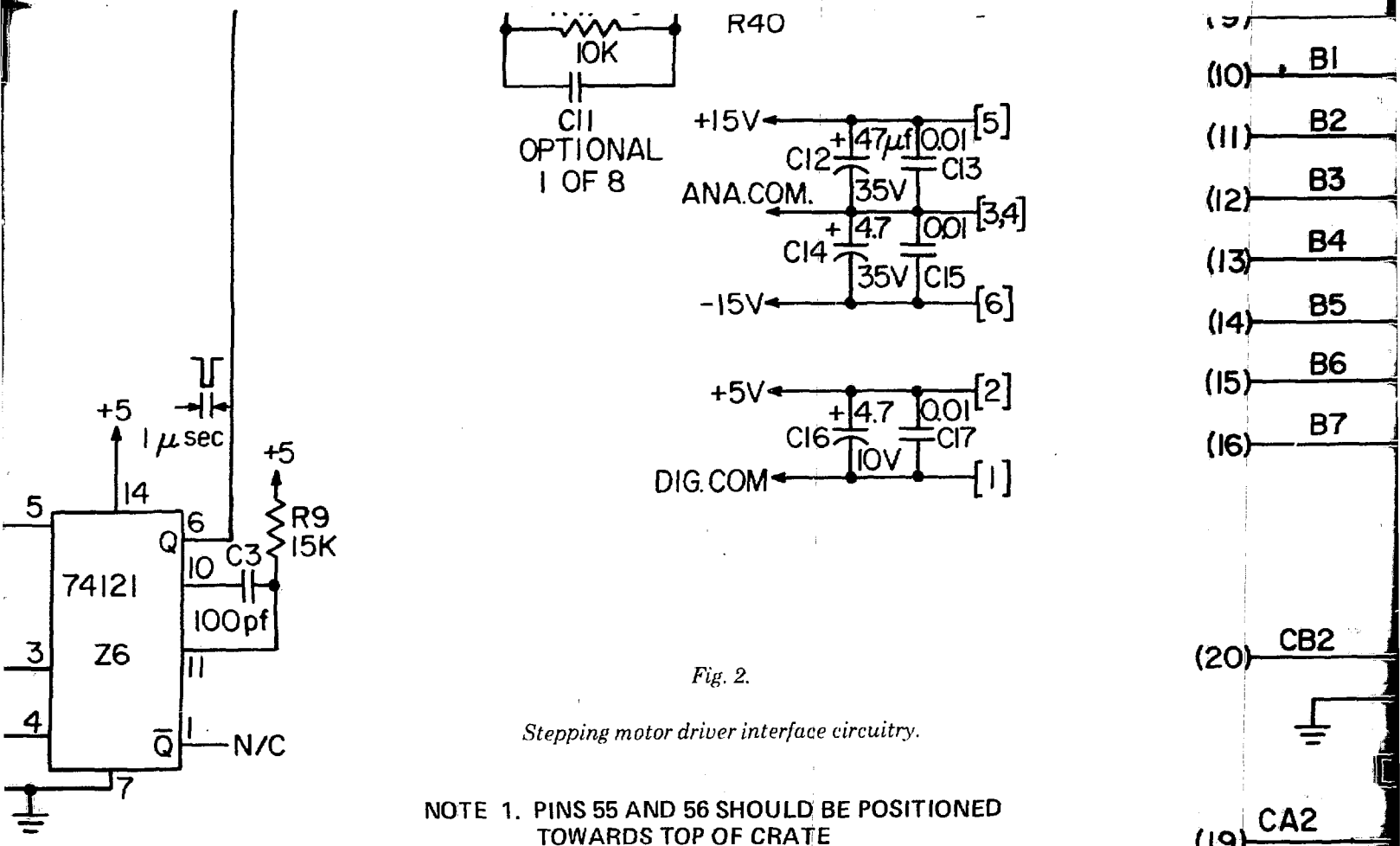


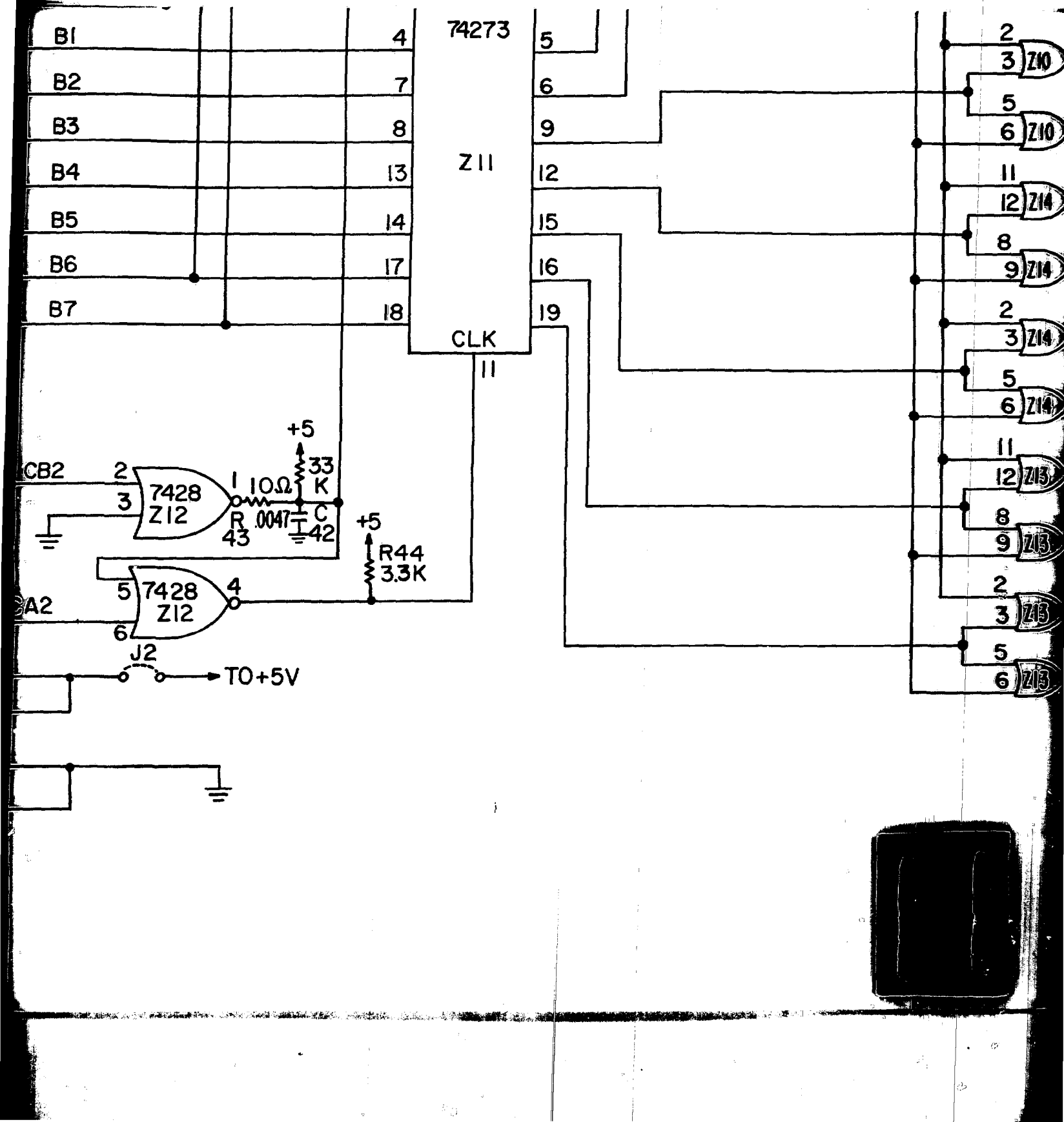
Fig. 2.

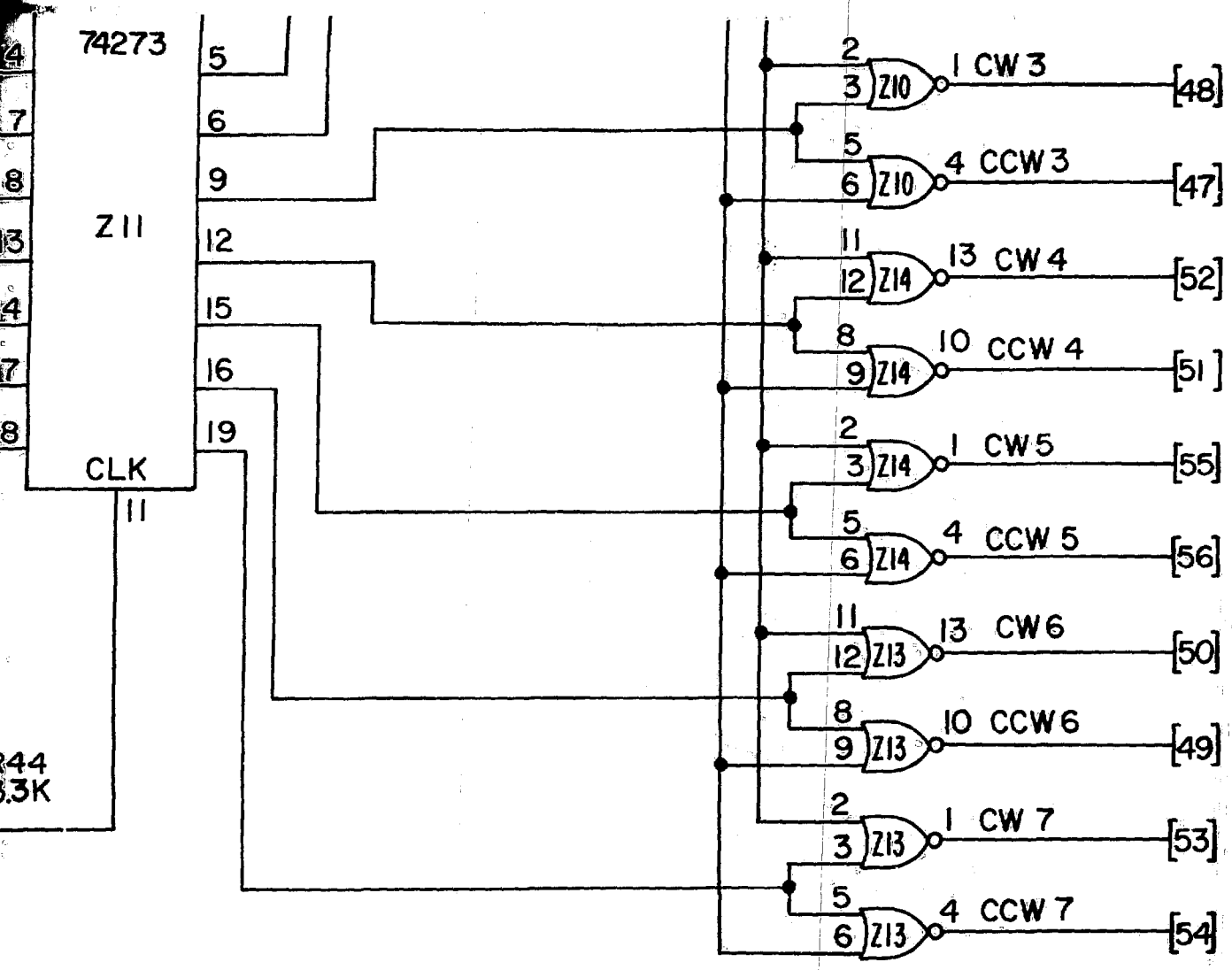
Stepping motor driver interface circuitry.

NOTE 1. PINS 55 AND 56 SHOULD BE POSITIONED TOWARDS TOP OF CRATE

2. BANDWIDTH LIMITING CAPACITORS ON LM 348 ARE OPTIONAL. HOLES WILL BE PROVIDED ON BOARD FOR 0.1 μf MONOLITHIC CERAMIC CAPS, PADS ON 0.2-in. CENTERS AND 0.15-in. THICKNESS

3. PROVIDE BY-PASS OF ± SUPPLY ON ANALOG COMPONENTS C43 - C50


B1
B2
B3
B4
B5
B6
B7
4
7
8
13
14
17
18
74273
Z11
5
6
9
12
15
16
19
CLK
11
+5
33 K
10Ω
R 43
.0047
C 42
CB2
2
3
7428
Z12
1
+5
R44
3.3K
A2
5
6
7428
Z12
4
J2
TO+5V
2
3
Z10
5
6
Z10
11
12
Z14
8
9
Z14
2
3
Z14
5
6
Z14
11
12
Z13
8
9
Z13
2
3
Z13
5
6
Z13


74273
Z11
CLK
11
5
6
9
12
15
16
19
2
3
Z10
1 CW 3
[48]
5
6
Z10
4 CCW 3
[47]
11
12
Z14
13 CW 4
[52]
8
9
Z14
10 CCW 4
[51]
2
3
Z14
1 CW 5
[55]
5
6
Z14
4 CCW 5
[56]
11
12
Z13
13 CW 6
[50]
8
9
Z13
10 CCW 6
[49]
2
3
Z13
1 CW 7
[53]
5
6
Z13
4 CCW 7
[54]