Stepper Motor Control with an MC68HC11E9 Microcontroller

By Bob King and Edgar Saenz

1 Introduction

This note provides basic implementation details and procedural information to design and assemble a stepper motor system. The controller discussed here is the MC68HC11E9, an 8-bit Freescale microcontroller (MCU). There are many embedded control applications supported by the M68HC11 Family.

The note consists of a general description and gives highlights of implementing a basic stepper motor system application. A step-by-step hardware assembly section is included to promote ease of construction should one desire to build a similar system.

To simplify the application, the software was generated on the Freescale M68HC11EVM evaluation module (EVM). The program created with the EVM is shown in **6 Listing**. The program runs in addresses $C000 through $C1CC. It is meant to be used as a guide and can be modified to support a variety of stepper motor control applications. Some modules will require no changes for use. For convenience, a copy of the code is available through Freeware Data Services. The Freeware BBS can be accessed by modem at (512) 891-3733, or via the World Wide Web at *http://freeware.sps.mot.com*.

The EVM comes with an on-board monitor called EVMbug11 that supports software development. This evaluation system provides easy I/O interfacing to external hardware and offers the user an inexpensive programming solution for devices with OTP, EPROM and EEPROM non-volatile memory.

Evaluation of the A0, A1, A8, E0, E1, E9 or 811E2 versions of M68HC11 microcontroller devices is supported when using the EVM. The microcontroller that resides on the EVM for this application is the MC68HC811E1 version.

2 General System Information

Figure 1 shows basic system operation. R1 provides an analog input to the MCU which is converted to a digital value and used to determine the speed at which the motor turns. In this example, the resistance is being varied manually for the A/D input to the MCU. A feedback scheme from the motor back to the A/D input could be implemented to facilitate a closed loop system.

To support motor turn direction, one I/O port pin is used to determine clockwise or counter-clockwise rotation. The voltage applied to the pin is sampled each time the program cycles through the software routines. A manual switch controls the state of the I/O pin. Green and yellow LEDs illuminate to indicate the turn direction.

A seven-segment display shows the delay between steps when the stepper motor is driven, and indicates motor speed. A parallel port is used to send the appropriate character codes to the seven-segment display. Four LEDs form a second visual speed indicator. These LEDs are turned on in sequence as the respective coils of the stepper motor are activated. The activating pulse originates from an on-chip port. The pulse pattern displayed by the LEDs alternates according to the motor shaft turn direction.

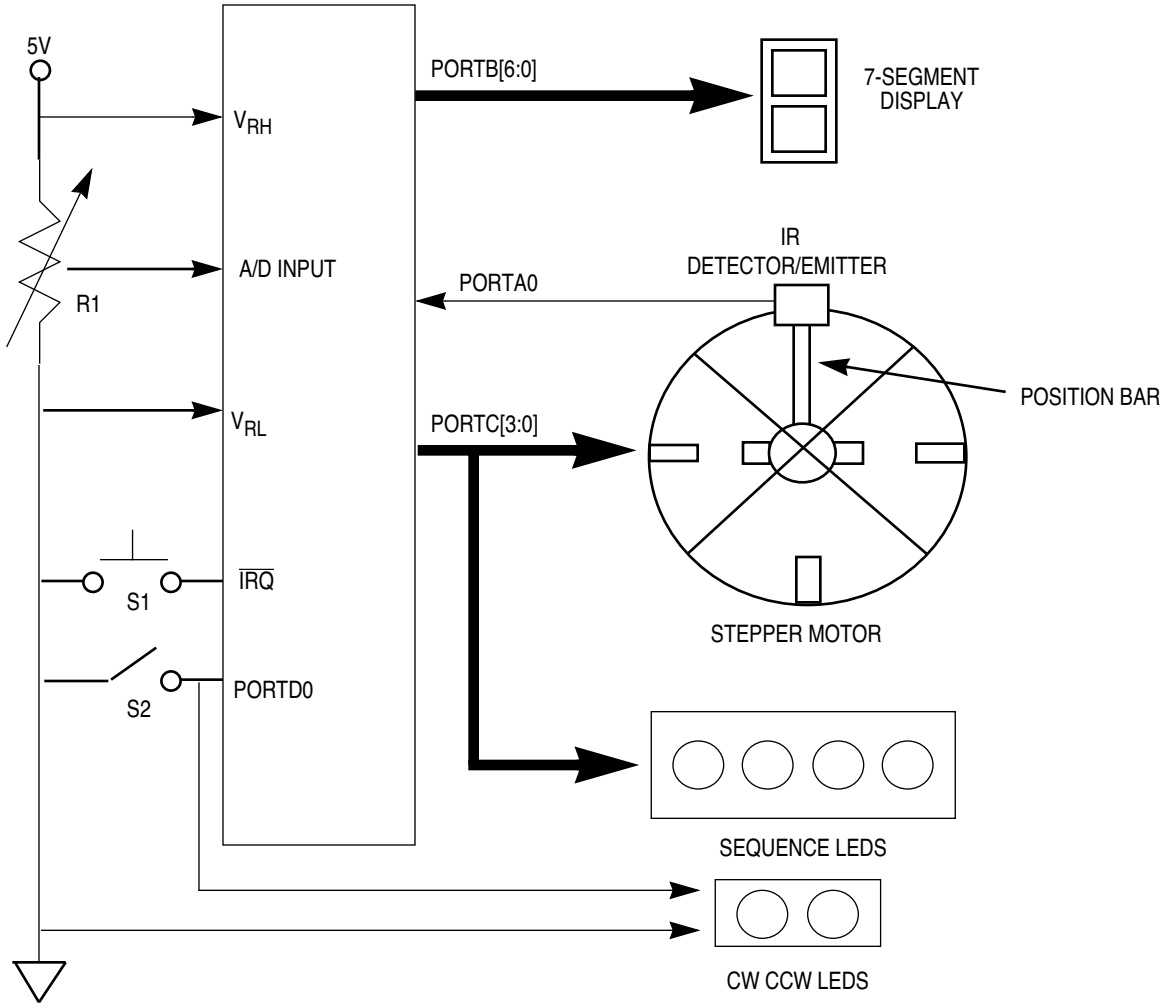


Figure 1 Basic Stepper Motor Operation

3 Hardware Development

3.1 Motor Description

The motor coil assemblies operate at voltage levels ranging from +5 to +24 volts. The motor has a single rotor which is connected to a shaft at the center of the assembly. There are multiple coils surrounding the rotor. A total of 100 steps are required for one complete revolution of the shaft. Each step increments by 3.6°. For this application, a wheel is attached to the shaft, but for other applications, the wheel could be replaced by a gear, a pulley, a belt or a timing mechanism.

3.2 Components

The hardware required to control the stepper motor varies significantly from one application to the next. Below is a list of components used to implement the system interfaced to an EVM:

1. One EVM;
2. One stepper motor;
3. One ULN-2075B motor driver IC (optional for enhanced drive);
4. One 25 KΩ potentiometer (A/D input control);
5. Two SPDT switches (power and CW or CCW turn control);
6. One SPDT switch MOM (position control and single-step);
7. One seven-segment display (display stepper motor delay $0–F);
8. One infrared detector and emitter (position control);
9. Seven LEDs (sequence, power and CW or CCW indicators);
10. Two 1-inch square mounting boards for the IR pair;
11. One project assembly board (4-inch x 6-inch);
12. One power terminal strip;
13. Two wirewrap sockets;
14. Four NPN and four PNP transistors (optional for increased motor drive).

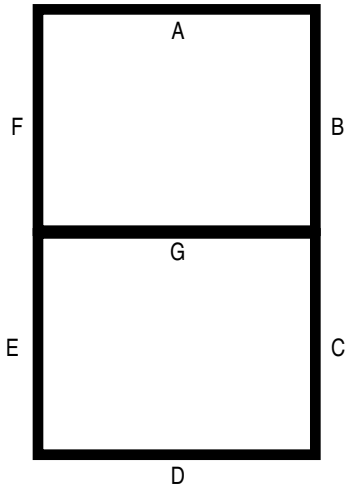
3.3 Assembly Procedures

Use the following sequence to assemble the project.

1. Lay out the positions of the various components that are located on the 4-inch x 6-inch project assembly board. A board of this size provides ample room for all hardware required to assemble this project.
2. Place the power terminal strip at one end of the project board. Connect +5 volts and ground connections from the EVM to the appropriate power terminal strip connection. An optional power supply can be used to provide increased power for driving the motor.
3. Connect the +5 volts from the power terminal strip to one side of the slide switch. Connect the other side of the slide switch to the main +5 volt power bus for distribution to components on the project board. The main ground bus on the project board also needs to be made available for distribution to the components on the project board.
4. Place the two wirewrap sockets at the opposite end of the project board from the power terminal strip. One socket is for the seven-segment display and the other is for the optional motor driver.
5. A 25 KΩ potentiometer is used for the analog input to the A/D converter. Connect one side of the potentiometer to +5 volts and the other side to ground. The center tap of the potentiometer is connected to PORTE bit 0 on the MCU. +5 volts is connected to the high reference voltage (V_{RH}) and ground is used to supply the low reference voltage (V_{RL}) MCU inputs.
6. The base portion of the stepper motor being used is a 1.5-inch cube. Place it securely in the center of the project board with the shaft and turning wheel at the top. Align the infrared (IR) emitter and detector to provide the best transmission and detection of the IR signal. Each IR component is mounted on a 1-inch square mounting board to aid with alignment.
7. Connect the emitter to +5 volts so that it continually emits a signal. Connect the detector to the MCU PORTA0 for sampling. The stepper motor has a narrow position bar located on the wheel. As the wheel turns, the position bar passes through the signal being sent and received by the IR pair.
8. The stepper motor has four coil wires that must be connected to MCU PORTC to promote the desired turning motion. PORTC[3:0] are used for this connection. The order of connecting these wires depends on the motor being used. A wiring diagram of the motor simplifies the connection process. The diagram is usually supplied with the motor.
9. Adding the optional ULN-2075B driver significantly enhances the performance of the system by providing increased drive capability. The ULN-2075B IC contains four individual driver circuits. These must be connected from the MCU to the input of the driver and from the output of the driver to the stepper motor coil connections. Another method of increasing the drive current to the stepper motor is to use a push-pull amplifier between each motor coil and PORTC. The

amplifier consists of one NPN and one PNP transistor. One side of the amplifier is connected to the optional +12 volts and the other side goes to ground.

10. The PORTC pins are also connected to four LEDs. As the motor turns, the LEDs indicate the sequence of motor coil activations, turn direction and speed of motor turn.
11. Wire the clockwise/counter clockwise slide switch and the respective LED indicators to the MCU. For this application, the switch is connected to PORTD0. One side of the switch is pulled high and the other side is pulled low. One LED is wired to illuminate when PORTD0 is high and the other LED illuminates when PORTD0 is low.
12. Wire MCU PORTB to the seven-segment display. **Figure 2** shows interfacing requirements for the seven-segment display.



SEGMENT	PORTB BIT #
A	6
B	5
C	4
D	3
E	2
F	1
G	0

Figure 2 Seven-Segment Display Interfacing

13. Connect the $\overline{\text{IRQ}}$ line to one side of a momentary switch. Connect the other side to ground. Activation of this switch causes instruction execution to resume after a WAI instruction has been executed. The switch is also used for motor single-stepping.

4 Software Development

All software routines for this application are implemented in assembly language. The EVM supports the software routine implementation. P&E Microsystems IASM11 software was used for code development, assembly, debug, and for programming the EEPROM memory.

Program execution occurs in the order shown in **Figure 3**. To take advantage of modular software techniques, several jump to subroutine (JSR) instructions are used. Each JSR has a corresponding return from subroutine (RTS) instruction. By using this method, the program has a smooth and efficient flow. Debugging software errors is simplified by using the modular approach as well.

The following paragraphs describe the subroutines that initialize the system, control the stepper movement, calculate speed, and output digital readouts. Each heading represents an individual stand-alone module.

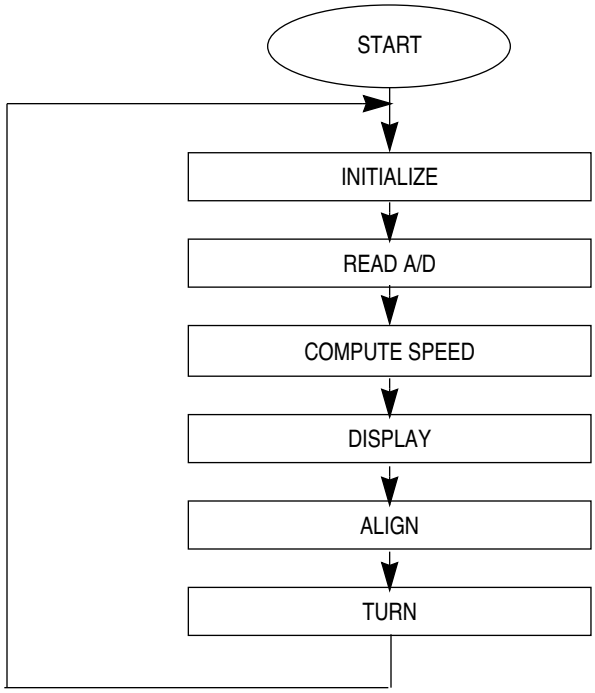


Figure 3 Stepper Motor Controller Program Flowchart

4.1 INIT

The initialization routine sets the base address for the MCU registers and establishes the constant port values used by the subroutines contained within the main program. In addition, the $\overline{IRQ}$ interrupt control bit in the CCR (I) is cleared. This allows the control bit to be set when the SWI instruction is executed.

A counter for turning the shaft 90°, 180°, or 360° is loaded during initialization. A delay timer to define the number of times each coil is activated is also set up. An alternate method of doing this would be to have a keyboard scan routine that accepts predetermined numbers or letters to control the degree of wheel turn as well as the number of coil activations.

4.2 DIRECTION

Following the initialization routine, the main program is entered. The first routine of the main program controls the shaft turning direction. The stepper motor can turn either clockwise or counterclockwise. A yellow LED and a green LED are used to indicate direction of turn. To change direction, a manual switch toggles the state of a single port pin. The state of this pin is stored at ATEMP. The status of the two LEDs is determined by the position of the switch.

4.3 READAD

A 25 KΩ potentiometer is used as the analog input to PORT E. The reference high and reference low inputs of the A/D are set at +5 V and 0 V respectively.

A value of $90 is written to the A/D OPTION register (OPTION), enabling the A/D power up (ADPU) and the delay (DLY) for crystal stabilization as shown below:

OPTION — System Configuration Options **$1039**

	BIT 7	6	5	4	3	2	1	BIT 0
	ADPU	CSEL	IRQE[1]	DLY[1]	CME	—	CR1[1]	CR0[1]
RESET:	0	0	0	0	0	0	0	0

NOTES:
1. Can be written only once in first 64 cycles out of reset in normal modes, or at any time in special modes.

A value of $A0 is written to the A/D control register (ADCTL), enabling the scan mode and setting the conversion complete flag, as shown below.

ADCTL — A/D Control/Status **$1030**

	BIT 7	6	5	4	3	2	1	BIT 0
	CCF	—	SCAN	MULT	CD	CC	CB	CA
RESET:	I	0	I	I	I	I	I	I

To simplify the design, keyboard inputs can be used to provide the necessary value directly to the A/D converter rather than using a potentiometer.

4.4 COMSPD

This routine reads A/D result register 1 (ADR1) to determine the speed value being input by the potentiometer. ADR1 is an 8-bit register that contains one of 256 possible values. The value in the register is complemented so that the highest value (F) represents the longest delay or slowest turning speed.

To obtain the highest resolution of result register content, four consecutive LSRA instructions are executed. Following these four shifts, a value in the range 0 to F remains in the lower nibble of accumulator A. The value is stored in ATEMP2 for later use. To insure that the latest converted value is represented, the ADCTL register is set up so that the result register is continuously being scanned during program execution.

4.5 DISPLAY

The A/D conversion value is retrieved from location ATEMP2. DISPLAY calls the COMPDIS subroutine, which determines the number to be displayed. When the polling routine finds the appropriate match, the data to turn on the segments for that particular number is stored in PORT B. The information from PORT B is routed to the seven-segment display for visual monitoring.

4.6 ALIGN

The ALIGN routine controls the actual alignment of the motor wheel to a known starting point, as shown in **Figure 4**. This configuration illustrates an easy method of controlling wheel alignment.

An infrared (IR) emitter and detector are used to establish proper alignment of the wheel. The motor wheel has an extension connected to its topside. As the wheel turns, the extension breaks the invisible IR beam between the emitter and detector. When this occurs during the very first revolution, the interrupt flag (I) is set, the wheel stops turning and the letter ‘S’ for STOP is displayed on the seven-segment display. The wheel is now aligned to a known starting point, and the program is waiting for the interrupt to be serviced by the appropriate routine. When the program continues to cycle through the main subroutines, the ALIGN routine is bypassed.

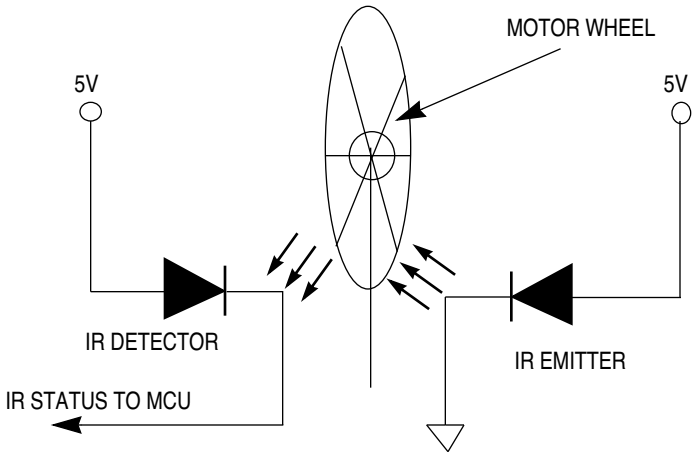


Figure 4 IR Emitter and Detector Stepper Motor Wheel Alignment

4.7 TURN

Now that the wheel is perfectly aligned to a known starting point, the TURN subroutine can be executed. To initiate the turn sequence, the interrupt from the ALIGN routine must be serviced. This is accomplished with a momentary switch S1 connected to the $\overline{IRQ}$ line.

The value from ATEMP is used to control whether to use the clockwise or counterclockwise subroutine. After the direction has been determined, the corresponding routine is entered and the coils energize in the proper sequence to cause the motor to turn. **Figure 5** shows the interface between the MCU and the stepper motor coils.

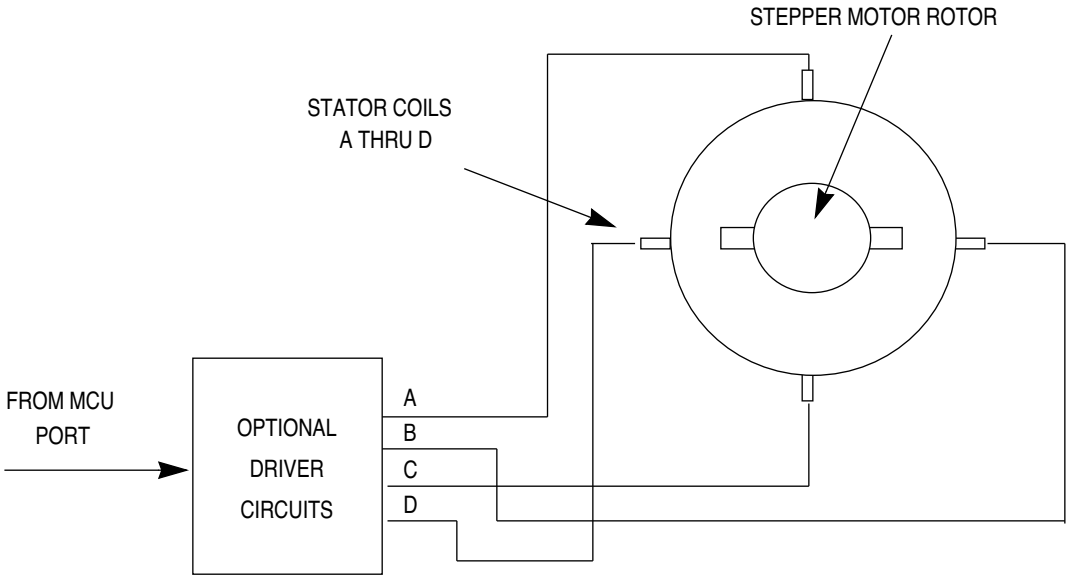


Figure 5 MCU Interface to Stepper Motor Coils

Table 1 shows the pattern used to energize the coils of the stepper motor for clockwise and counterclockwise rotation. Only four port pins are needed to control the pulses going to the motor. The letters A through D represent the inputs to the coils.

Table 1 Stepper Motor Coil Energizing Pattern for CW and CCW Rotation

CW Rotation			
D	C	B	A
0	0	0	1
0	0	1	1
0	0	1	0
0	1	1	0
0	1	0	0
1	1	0	0
1	0	0	0
1	0	0	1

CCW Rotation			
D	C	B	A
0	0	0	1
1	0	0	1
1	0	0	0
1	1	0	0
0	1	0	0
0	1	1	0
0	0	1	0
0	0	1	1

Four PORT C pins are used to drive the inputs to the stepper motor coils. Direct connection from the MCU to the motor is fine for applications that require minimal drive. But for applications that require increased current drive capabilities, enhanced circuits are necessary.

One way to increase drive current is to use a motor controller IC designed specifically for that purpose. An example of this device is the ULN2075B driver IC. Each IC contains four individual high-current Darlington switch and driver circuits. One IC satisfies the MCU to the stepper motor interfacing requirements.

Another approach is to use discrete components to form a push/pull amplifier. Each amplifier consists of an NPN and a PNP transistor. The amplifier is arranged to generate a 12 volt output pulse to one input of the motor in response to a +5 volt pulse coming from PORT C. **Figure 6** shows the amplifier configuration for one motor coil.

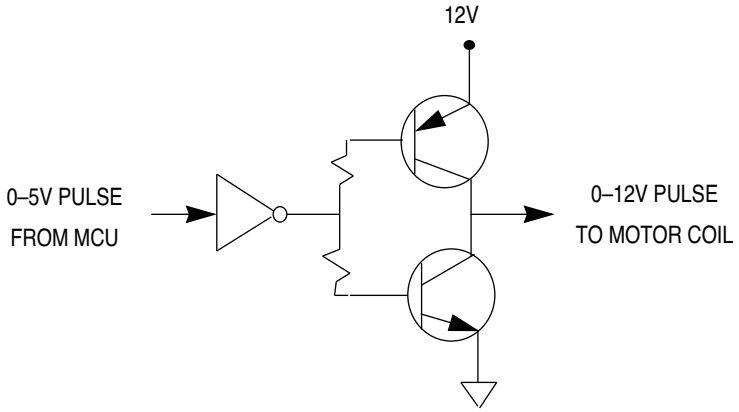


Figure 6 Amplifier Configuration for a Stepper Motor Coil

As the motor turns, the A/D input from the potentiometer is being scanned. When the value varies, the seven-segment display changes accordingly as does the speed of the motor.

The momentary switch connected to the $\overline{\text{IRQ}}$ line has an additional function. It is the input for the manual/single step control feature of this application. When the switch is engaged, the motor halts. When the switch is released, the motor resumes normal operation. Many applications can take advantage of the single step feature to monitor elapsed time or observe the status of activities that may be linked to the motor.

The IR emitter and detector have numerous potential uses in this and similar applications. A visual display can be used for each revolution of the wheel. This application displays a 'P' each time the IR emission path is broken. For enhanced applications, this same principal can be used to increment a counter each time there is a missing IR signal received by the MCU.

After the RTS instruction is executed at the end of the TURN subroutine, the program is directed with a BRA instruction to go to the label NEXT and sample the A/D input. From here, the program continues to cycle until the routine is forced to stop or until a predetermined count or time period has elapsed.

5 Conclusion and Summary

There are numerous stepper motor applications that can take advantage of the power, features and flexibility of the M68HC11 single-chip MCU. Applications would include robotics controllers, turning machine tools and other precise shaft positioning control environments. This example is a general solution that demonstrates the ease with which an MCU can be designed into a stepper motor control application.

Due to the types of applications supported, stepper motors operate at relatively low rotating speeds. The actual speed is controlled by varying the delay between coil activations. With this system application, the stepper motor converts binary input pulses coming from the MCU to rotary shaft movement on the stepper motor. The direction of turn is a function of the sequence in which the binary pulses are applied to the stepper motor.

In addition, the requirement for a digital-to-analog converter is eliminated when using stepper motors versus dc or ac motors in dc systems. Ac and dc motors provide continuous shaft rotation. However, stepper motors produce shaft rotation in precise steps or increments as the result of the applied binary pulses. This can be in the form of either half or full steps (step-angular sensitivity) depending on the sequence of coil activations.

It is noteworthy to mention that most stepper motors are used in applications with relatively small loads. An overload condition could result in a shaft slip. This undesirable condition could induce an error that might not be recognized and affect operating precision. To minimize the possibility of this occurring, buffer type amplifiers should be placed between the MCU and the stepper motor.

In terms of reliability, MCUs can operate problem-free in stepper motor applications for years if used within their specified limits.

6 Listing

```
0000                    1  ADCTL     EQU      $30
0000                    2  PORTA     EQU      $0
0000                    3  PORTB     EQU      $4
0000                    4  PORTCDR   EQU      $7
0000                    5  PORTC     EQU      $3
0000                    6  PORTDDR   EQU      $9
0000                    7  PORTD     EQU      $8
0000                    8  RESREG    EQU      $31
0000                    9  ADON      EQU      $39
0000                   10  ATEMP     RMB      1
0001                   11  ATEMP2    RMB      1
0002                   12  ATEMP3    RMB      1
0003                   13  ATEMP4    RMB      1
0004                   14  ATEMP5    RMB      1
0005                   15  TIMER     RMB      1
0006                   16  COUNTER   RMB      1
0007                   17  FLGIRQ    RMB      1
                       18
C000                   19            ORG  $C000
                       20
C000 BDC017            21  START     JSR INIT            ;INITALIZE ROUTINE
C003 BDC034            22  NEXT      JSR DIRECTION       ;DIRECTION ROUTINE
C006 BDC03D            23            JSR READAD          ;READ A/D ROUTINE
C009 BDC04E            24            JSR COMSPD          ;COMPUTE SPEED ROUTINE
C00C BDC058            25            JSR DISPLAY         ;7-SEGMENT DISPLAY ROUTINE
C00F BDC0D2            26            JSR ALIGN           ;POSITION CONTROL ROUTINE
C012 BDC0EF            27            JSR TURN            ;STEPPER MOTOR TURN ROUTINE
C015 20EC              28            BRA NEXT
                       29
                       30              ;INITALIZE ROUTINE
C017 CE1000            31  INIT      LDX #$1000
C01A 86FF              32            LDAA #$FF           ;SET PORTC FOR OUTPUT
C01C A707              33            STAA PORTCDR,X      ;TO TURN MOTOR
C01E 8600              34            LDAA #$00           ;SET PORTD FOR INPUT
C020 A709              35            STAA PORTDDR,X      ;TO CONTROL MOTOR TURN
                                                                 ;DIRECTION
C022 C605              36            LDAB #5             ;SET TIMER FOR # OF TIMES TO
C024 D705              37            STAB TIMER          ;ACTIVATE EACH COIL
C026 8610              38            LDAA #10            ;SET COUNTER FOR # OF STEPS
C028 9706              39            STAA COUNTER        ;20 = 1 REVOLUTION
C02A 8600              40            LDAA #$00           ;SET PORTA FOR
C02C A700              41            STAA PORTA,X        ;IR EMITTER DETECTOR
C02E 9704              42            STAA ATEMP5
C030 9707              43            STAA FLGIRQ
C032 0E                44            CLI
C033 39                45            RTS
                       46
                       47              ;DIRECTION ROUTINE - CLOCKWISE OR COUNTER CLOCKWISE
C034 A608              48  DIRECTION LDAA PORTD,X        ;READ BIT 0 OF PORTD
C036 8401              49            ANDA #$01           ;MASK PORTD BITS 1-7
C038 A708              50            STAA PORTD,X        ;WRITE TO PORTD BIT 0 FOR CW OR CCW
C03A 9700              51            STAA ATEMP          ;STORE DIRECTION AT ATEMP
C03C 39                52            RTS
                       53
                       54              ;SET UP A/DCTL REGISTER (READ VARIABLE RESISTOR
                                       ;THROUGH PORT E)
                       55
C03D 8690              56  READAD       LDAA #$90       ;OPTION REG SET A/D PWR ON & DELY
C03F A739              57            STAA ADON,X     ;ENABLED FOR XTAL STABLIZATION ($1039)
C041 86A0              58            LDAA #$A0       ;SET A/D CONTROL WORD FOR SCAN MODE
C043 A730              59            STAA ADCTL,X    ;& CONVERSION COMPLETE FLAG SET ($1030)
C045 18CE0026          60            LDY #$26
C049 1809              61  DELAY     DEY
C04B 26FC              62            BNE   DELAY
C04D 39                63            RTS
```

```
                    64
                    65                                  ;READ CONTENTS OF THE RESULT
                                                        ;REGISTER TO COMPUTE TURN SPEED
                    66
 C04E A631          67  COMSPD    LDAA RESREG,X   ;READ RESULT REGISTER ($1031)
 C050 43            68             COMA            ;COMP SO HIGH # = LONGEST DELAY
 C051 44            69             LSRA            ;SHIFT 'A' 4 TIMES FOR 0 - F COUNT
 C052 44            70             LSRA            ;SHIFT
 C053 44            71             LSRA            ;SHIFT
 C054 44            72             LSRA            ;SHIFT
 C055 9701          73             STAA ATEMP2     ;STORE SHIFTED # AS FINAL SPEED CONTROL
                    74
 C057 39            75             RTS
                    76
                    77               ;DISPLAY SPEED ON 7-SEGMENT READOUT
 C058 9601          78  DISPLAY    LDAA ATEMP2     ;READ ATEMP2 # TO BE DISPLAYED
 C05A BDC060        79             JSR COMPDIS     ;JUMP TO ROUTINE TO COMPUTE DISPLAY
 C05D A704          80             STAA PORTB,X    ;DISPLAY 0-F THROUGH PORTB (7-SEG)
 C05F 39            81             RTS
                    82
 C060 8100          83  COMPDIS      CMPA #$00              ;COMPARE A = 0
 C062 273E          84             BEQ DOWN0
 C064 8101          85             CMPA #$01              ;COMPARE A = 01
 C066 273D          86             BEQ DOWN1
 C068 8102          87             CMPA #$02              ;COMPARE A = 02
 C06A 273C          88             BEQ DOWN2
 C06C 8103          89             CMPA #$03              ;COMPARE A = 03
 C06E 273B          90             BEQ DOWN3
 C070 8104          91             CMPA #$04              ;COMPARE A = 04
 C072 273A          92             BEQ DOWN4
 C074 8105          93             CMPA #$05              ;COMPARE A = 05
 C076 2739          94             BEQ DOWN5
 C078 8106          95             CMPA #$06              ;COMPARE A = 06
 C07A 2738          96             BEQ DOWN6
 C07C 8107          97             CMPA #$07              ;COMPARE A = 07
 C07E 2737          98             BEQ DOWN7
 C080 8108          99             CMPA #$08              ;COMPARE A = 08
 C082 2736         100             BEQ DOWN8
 C084 8109         101             CMPA #$09              ;COMPARE A = 09
 C086 2735         102             BEQ DOWN9
 C088 810A         103             CMPA #$0A              ;COMPARE A = 0A
 C08A 2734         104             BEQ DOWNA
 C08C 810B         105             CMPA #$0B              ;COMPARE A = 0B
 C08E 2733         106             BEQ DOWNB
 C090 810C         107             CMPA #$0C              ;COMPARE A = 0C
 C092 2732         108             BEQ DOWNC
 C094 810D         109             CMPA #$0D              ;COMPARE A = 0D
 C096 2731         110             BEQ DOWND
 C098 810E         111             CMPA #$0E              ;COMPARE A = 0E
 C09A 2730         112             BEQ DOWNE
 C09C 810F         113             CMPA #$0F              ;COMPARE A = 0F
 C09E 272F         114             BEQ DOWNF
 C0A0 20BE         115             BRA COMPDIS            ;END OF POLL ROUTINE
                   116
 C0A2 86C0         117  DOWN0     LDAA #$C0              ;DISPLAY VALUE ON 7-SEG DISPLAY IF
MATCH
                                                         ;VALUE = 0
 C0A4 39           118             RTS
 C0A5 86CF         119  DOWN1     LDAA #$CF              ;VALUE = 1
 C0A7 39           120             RTS
 C0A8 8692         121  DOWN2     LDAA #$92              ;VALUE = 2
 C0AA 39           122             RTS
 C0AB 8686         123  DOWN3     LDAA #$86              ;VALUE = 3
 C0AD 39           124             RTS
 C0AE 868D         125  DOWN4     LDAA #$8D              ;VALUE = 4
 C0B0 39           126             RTS
```

```
C0B1 86A4       127  DOWN5     LDAA #$A4               ;VALUE = 5
C0B3 39         128            RTS
C0B4 86A1       129  DOWN6     LDAA #$A1               ;VALUE = 6
C0B6 39         130            RTS
C0B7 86CE       131  DOWN7     LDAA #$CE               ;VALUE = 7
C0B9 39         132            RTS
C0BA 8680       133  DOWN8     LDAA #$80               ;VALUE = 8
C0BC 39         134            RTS
C0BD 868C       135  DOWN9     LDAA #$8C               ;VALUE = 9
C0BF 39         136            RTS
C0C0 8688       137  DOWNA     LDAA #$88               ;VALUE = A
C0C2 39         138            RTS
C0C3 8680       139  DOWNB     LDAA #$80               ;VALUE = B
C0C5 39         140            RTS
C0C6 86F0       141  DOWNC     LDAA #$F0               ;VALUE = C
C0C8 39         142            RTS
C0C9 8683       143  DOWND     LDAA #$83               ;VALUE = D
C0CB 39         144            RTS
C0CC 86B0       145  DOWNE     LDAA #$B0               ;VALUE = E
C0CE 39         146            RTS
C0CF 86B8       147  DOWNF     LDAA #$B8               ;VALUE = F
C0D1 39         148            RTS
                149
                150              ;ALIGN WHEEL FOR POSITION CONTROL BETWEEN THE
                                 ;R EMITTER AND DETECTOR
                151
C0D2 A600       152  ALIGN     LDAA PORTA,X
C0D4 8401       153            ANDA #$01
C0D6 8100       154            CMPA #$00
C0D8 2702       155            BEQ NEX1
C0DA 2613       156            BNE TURN
C0DC 9607       157  NEX1      LDAA FLGIRQ             ;CHECK IRQ FLAG
C0DE 8100       158            CMPA #$00
C0E0 2702       159            BEQ WAIT                ;IF = 0 GOTO WAIT
C0E2 260B       160            BNE TURN                ;IF NOT = 0 BRANCH TO TURN
C0E4 86A4       161  WAIT      LDAA #$A4               ;DISPLAY 'S' FOR STOP
C0E6 A704       162            STAA PORTB,X            ;AT PORTB AND WAIT FOR
C0E8 8610       163            LDAA #10
C0EA 9706       164            STAA COUNTER
C0EC 0E         165            CLI
C0ED 3E         166            WAI                     ;INTERRUPT TO BE SERVICED
C0EE 39         167            RTS
                168
                169              ;STEPPER MOTOR TURN ROUTINE
C0EF 7A0006     170  TURN      DEC COUNTER
C0F2 9606       171            LDAA COUNTER
C0F4 8100       172            CMPA #00
C0F6 2702       173            BEQ BB
C0F8 2607       174            BNE BBB
C0FA 8610       175  BB        LDAA #10
C0FC 9706       176            STAA COUNTER
C0FE 2001       177            BRA BBB
C100 39         178            RTS
                179
C101 9600       180  BBB       LDAA ATEMP              ;GET STORED DIRECTION
C103 2653       181            BNE   CCW               ;IF NOT =, TURN CCW
C105 2700       182            BEQ   CW                ;ELSE TURN CW
                183
                184              ;CLOCKWISE TURN ROUTINE
C107 D605       185  CW        LDAB TIMER
C109 8601       186  CW1       LDAA #$01               ;COIL VALUE FOR POSITION 1
C10B BDC1A9     187            JSR DELAY1
C10E 5A         188            DECB
C10F 26F8       189            BNE CW1
C111 D605       190            LDAB TIMER
C113 8603       191  CW3       LDAA #$03               ;COIL VALUE FOR POSITION 2
```

```
C115 BDC1A9     192             JSR DELAY1
C118 5A         193             DECB
C119 26F8       194             BNE CW3
C11B D605       195             LDAB TIMER
C11D 8602       196  CW2        LDAA #$02               ;COIL VALUE FOR POSITION 3
C11F BDC1A9     197             JSR DELAY1
C122 5A         198             DECB
C123 26F8       199             BNE CW2
C125 D605       200             LDAB TIMER
C127 8606       201  CW6        LDAA #$06               ;COIL VALUE FOR POSITION 4
C129 BDC1A9     202             JSR DELAY1
C12C 5A         203             DECB
C12D 26F8       204             BNE CW6
C12F D605       205             LDAB TIMER
C131 8604       206  CW4        LDAA #$04               ;COIL VALUE FOR POSITION 5
C133 BDC1A9     207             JSR DELAY1
C136 5A         208             DECB
C137 26F8       209             BNE CW4
C139 D605       210             LDAB TIMER
C13B 860C       211  CWC        LDAA #$0C               ;COIL VALUE FOR POSITION 6
C13D BDC1A9     212             JSR DELAY1
C140 5A         213             DECB
C141 26F8       214             BNE CWC
C143 D605       215             LDAB TIMER
C145 8608       216  CW8        LDAA #$08               ;COIL VALUE FOR POSITION 7
C147 BDC1A9     217             JSR DELAY1
C14A 5A         218             DECB
C14B 26F8       219             BNE CW8
C14D D605       220             LDAB TIMER
C14F 8609       221  CW9        LDAA #$09               ;COIL VALUE FOR POSITION 8
C151 BDC1A9     222             JSR DELAY1
C154 5A         223             DECB
C155 26F8       224             BNE CW9
C157 39         225             RTS
                226
                227               ;COUNTER CLOCKWISE ROUTINE
C158 D605       228  CCW        LDAB TIMER
C15A 8601       229  CCW1       LDAA #$01               ;COIL VALUE FOR POSITION 9
C15C BDC1A9     230             JSR DELAY1
C15F 5A         231             DECB
C160 26F8       232             BNE CCW1
C162 D605       233             LDAB TIMER
C164 8609       234  CCW9       LDAA #$09               ;COIL VALUE FOR POSITION 8
C166 BDC1A9     235             JSR DELAY1
C169 5A         236             DECB
C16A 26F8       237             BNE CCW9
C16C D605       238             LDAB TIMER
C16E 8608       239  CCW8       LDAA #$08               ;COIL VALUE FOR POSITION 7
C170 BDC1A9     240             JSR DELAY1
C173 5A         241             DECB
C174 26F8       242             BNE CCW8
C176 D605       243             LDAB TIMER
C178 860C       244  CCWC       LDAA #$0C               ;COIL VALUE FOR POSITION 6
C17A BDC1A9     245             JSR DELAY1
C17D 5A         246             DECB
C17E 26F8       247             BNE CCWC
C180 D605       248             LDAB TIMER
C182 8604       249  CCW4       LDAA #$04               ;COIL VALUE FOR POSITION 5
C184 BDC1A9     250             JSR DELAY1
C187 5A         251             DECB
C188 26F8       252             BNE CCW4
C18A D605       253             LDAB TIMER
C18C 8606       254  CCW6       LDAA #$06               ;COIL VALUE FOR POSITION 4
C18E BDC1A9     255             JSR DELAY1
C191 5A         256             DECB
C192 26F8       257             BNE CCW6
```

```
C194 D605       258             LDAB TIMER
C196 8602       259  CCW2       LDAA #$02               ;COIL VALUE FOR POSITION 3
C198 BDC1A9     260             JSR DELAY1
C19B 5A         261             DECB
C19C 26F8       262             BNE CCW2
C19E D605       263             LDAB TIMER
C1A0 8603       264  CCW3       LDAA #$03               ;COIL VALUE FOR POSITION 2
C1A2 BDC1A9     265             JSR DELAY1
C1A5 5A         266             DECB
C1A6 26F8       267             BNE CCW3
C1A8 39         268             RTS
                269
                270               ;DELAY ROUTINE
C1A9 A703       271  DELAY1     STAA PORTC,X
C1AB 9601       272             LDAA ATEMP2
C1AD 9702       273             STAA ATEMP3
C1AF 18DE02     274             LDY ATEMP3
C1B2 1809       275  COUNT      DEY                     ;DELAY PER LOADED VALUES
C1B4 26FC       276             BNE COUNT
C1B6 A600       277             LDAA PORTA,X
C1B8 8401       278             ANDA #$01
C1BA 8101       279             CMPA #$01               ;COMPARE VALUE TO OUTPUT TO DISPLAY
C1BC 270E       280             BEQ NOSEG
C1BE 2600       281             BNE SEG
C1C0 8698       282  SEG        LDAA #$98               ;DISPLAY 'P' FOR POSITION
C1C2 A704       283             STAA PORTB,X
C1C4 18CE0FFF   284             LDY #$FFF
C1C8 1809       285  ZZ         DEY
C1CA 26FC       286             BNE ZZ
C1CC 39         287  NOSEG        RTS
                288
                289
                290             ;INTERRUPT ROUTINE FOR POSITION CONTROL
FFF2            291           ORG $FFF2                 ;VECTOR FOR IRQ
FFF2 FFF4       292           FDB IRQHND
                293
FFF4 7C0007     294  IRQHND     INC FLGIRQ
FFF7 3B         295           RTI
                296
                297
                298
                299
```

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com