

## LAB 5 STEPPER MOTOR AND A/D CONVERSION

### OBJECTIVE

The objective of this laboratory is to introduce the student to the use of:

1. Stepper motors under microprocessor control
2. A/D conversion with the 68HC11 microcontroller

The physical principles and the programming features of these devices will be studied and applied.

### PART I – STEPPER MOTORS

#### PREREQUISITES

Floppy disk with the asm codes for the programs:

- LASTNAME\_Firstname\_Stepper\_Motor.asm

Hard copy (printout) of Hmwk7 – Stepper Motor. When printing, use the 'pages per sheet' option in the lower right corner of the print dialog-box with settings of 4 or 2 (depending on your eyesight) to save paper. (We may want to experiment a little with this before printing the full document.)

#### PROCEDURE

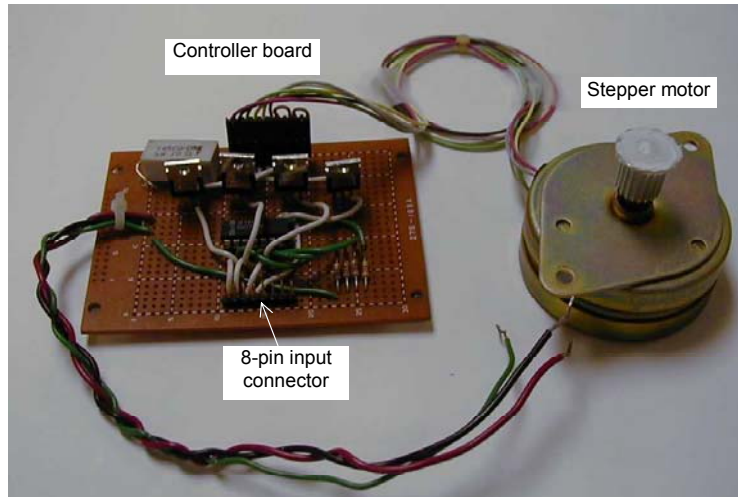
The students will utilize the asm code developed with the THRSim11 simulator for Hmwk 7 and Hmwk 8. The students will go through the printout of Hmwk7 step by step and will verify that the MCU responds to instructions as expected.

The lab is divided into sections. After completing each section, the student will ask the TA to check the student's work and make a check mark on that section.

The asm code is activated into the MCU following the standard procedure learned in Lab 1.

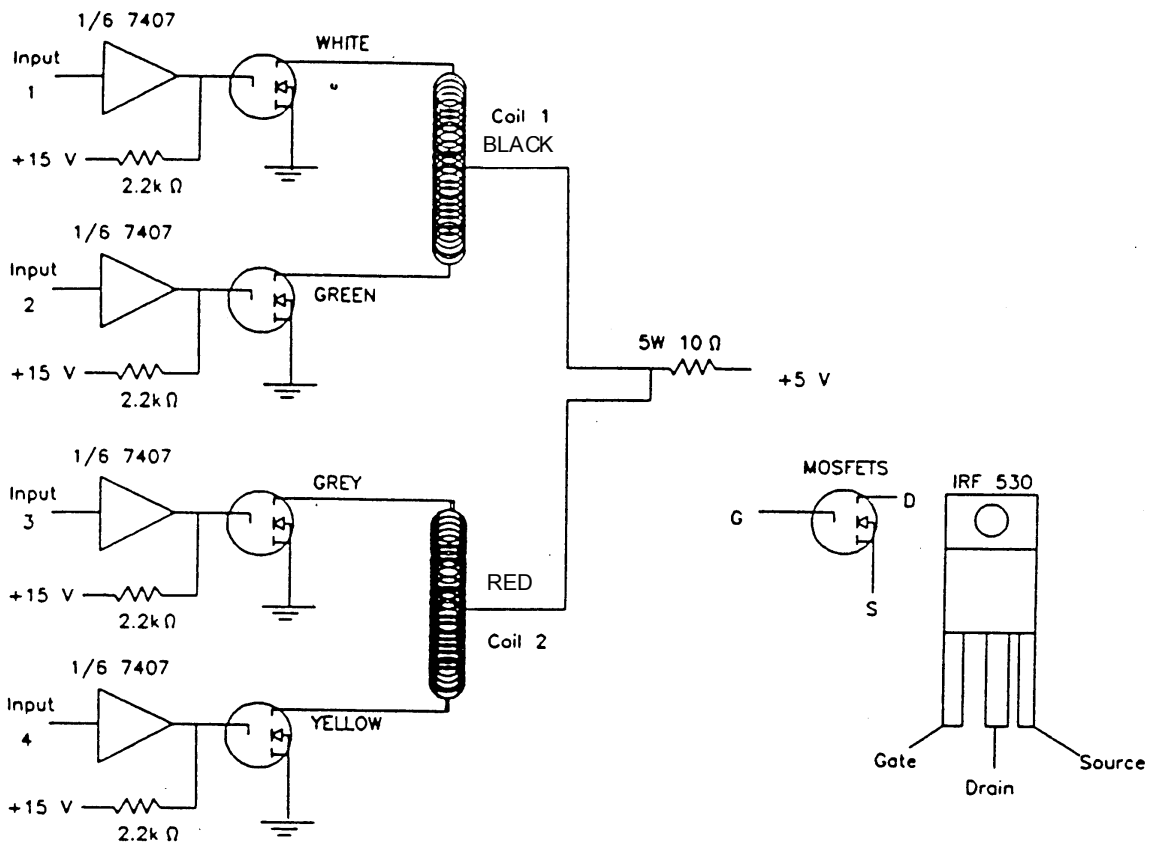
#### EXPERIMENTAL SETUP

The experimental setup for this experiment consists of a stepper motor (Figure 1) and its controller board (Figure 2). The stepper motor controller board is connected to the MCU port B. The MCU generates a sequence of binary patterns that are used by the controller board to energize the stepper motor coils and generate motion.



**Figure 1** Stepper motor and its controller board. Control signals to the controller board are sent through the 8-pin input connector.

*CIRCUIT DIAGRAM*



**Figure 2** Circuit diagram of the stepper motor controller board

*WIRING DIAGRAM*

Wire	Connection
------	------------

Green wire:	+15 V
Red wire	+5 V
Black wire	0 V (Ground)

The 8-pin input connector on the controller board is connected to MCU Port B. Make sure to respect the MSB convention.

### PRE-TEST PROCEDURE

Before starting your test, perform the following pre-test procedure to verify that your experimental set-up is performing correctly:

1. Check the correct wiring of the stepper motor controller board and input/output connectors:

Wire	Connection	Check mark
Green wire:	+15 V	
Red wire	+5 V	
Black wire	0 V (Ground)	
Input connector	Port B	
Output connector	Stepper motor connector	

2. Test hardware:

Use a removable marker to make two alignment marks, one on the stator, the other on the rotor of the stepper motor. Energize and establish communication.

Manually send the 2-hex numbers corresponding to the four full-step sequences of Table 1 to the parallel port B. Use a pencil to mark the position of your marking after each block of four full-step sequences has been entered. Fill in the answers below:

Estimate the angle of rotation after one complete block of four full-step sequences= \_\_\_\_\_

Count the number of blocks of four full-step sequences needed to achieve a complete rotation = \_\_\_\_\_

Calculate the angle of rotation corresponding to one full step: \_\_\_\_\_

Deduce the angle of rotation corresponding to a half step: \_\_\_\_\_

**Table 1 Stepper motor energizing patterns and their 2-hex equivalent value**

Sequence	Energizing pattern	8-bit 2-hex equivalence	Phase energizing type	Step type
S0	1000	\$08	1 phase	Half step
S1	1001	\$09	2 phase	Full step
S2	0001	\$01	1 phase	Half step

S3	0101	\$05	2 phase	Full step
S4	0100	\$04	1 phase	Half step
S5	0110	\$06	2 phase	Full step
S6	0010	\$02	1 phase	Half step
S7	1010	\$0A	2 phase	Full step

### STEPPER MOTOR CONTROL PROGRAM

Run the program of homework 7 using the commands >, <, +, -, S: Verify that it can be controlled to do forward and backward motion, slower and faster. Verify that it stops.

- |      |                                 |   |            |
|------|---------------------------------|---|------------|
| i)   | Move forward                    | > | Check mark |
| ii)  | Move backward                   | < | Check mark |
| iii) | Increase speed (decrease delay) | + | Check mark |
| iv)  | Decrease speed (increase delay) | - | Check mark |
| v)   | Stop program                    | S | Check mark |

When satisfied with the program operation, show it to your TA.

TA checkmark \_\_\_\_\_

## PART II – A/D CONVERSION

### PREREQUISITES

Floppy disk with the asm codes for the program:

- LASTNAME\_Firstname\_AD.asm

Hard copy (printout) of Hmwk8 – A/D Conversion. When printing, use the 'pages per sheet' option in the lower right corner of the print dialog-box with settings of 4 or 2 (depending on your eyesight) to save paper. (We may want to experiment a little with this before printing the full document.)

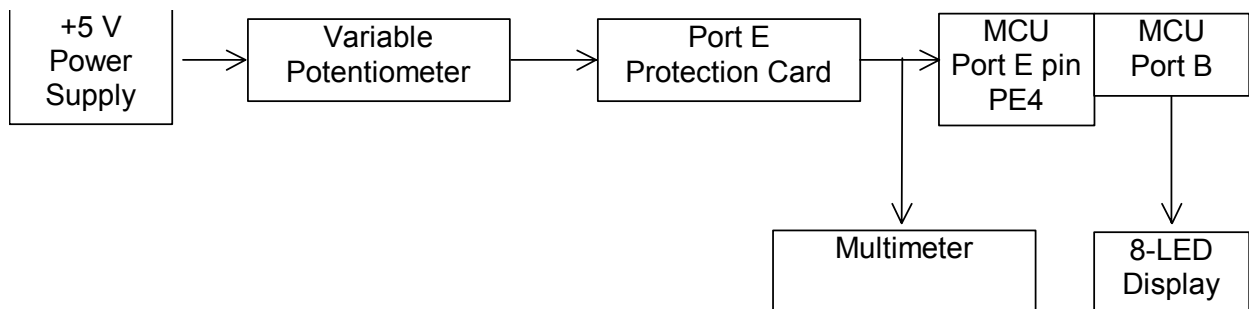
### PROCEDURE

The students will utilize the asm code developed with the THRSim11 simulator for Hmwk8. The students will go through the printout of Hmwk8 step by step and will verify that the MCU responds to instructions as expected.

The lab is divided into sections. After completing each section, the student will ask the TA to check the student's work and make a check mark on that section.

The asm code is activated into the MCU following the standard procedure learned in Lab 1.

### EXPERIMENTAL SETUP



**Figure 3 AD conversion experiment: block diagram**

The analog-to-digital (AD) converter function of the M68HC11 microprocessor will be used to record, as eight-bit digital data, the analog signals developed by a variable potentiometer and sent to the MCU through port E pin PE4.

The experimental setup (Figure 3) consists of a variable potentiometer, port E protection card, multimeter, the MCU port E, the MCU port B, and an 8-LED display. The potentiometer generates an adjustable voltage (analog signal) in the range 0 – 5 V. The analog signal is through the port E protection card to the MCU port E. The multimeter records the value (mV) of the analog signal that enters the MCU. Inside the MCU, the analog signal is converted to 8-bit digital. Conversion is effected continuously, in a round-robin fashion and stored internally in four variables (VAL1 – VAL4). The average of these four values is calculated by software and output through MCU port B for digital display on the 8-LED display.

**WIRING DIAGRAM**

Wire	Connection
Red wire	+5 V
Black wire	0 V (Ground)
Yellow wire	Potentiometer variable output (0 – 5 V)

**PRE-TEST PROCEDURE**

Before starting your test, perform the following pre-test procedure to verify that your experimental set-up is performing correctly:

1. Check the correct wiring of the potentiometer, port E protection card, port E pin PE4:

Wire	Connection	Check mark
Red wire	+5 V	
Black wire	0 V (Ground)	
Port E protection card and port E	MSB alignment	
Input connector	Port E protection card pin PE4	
Multimeter	Probe to PE4	
Output connector	8-LED display MSB alignment	

2. Connect multimeter to yellow wire. Adjust the potentiometer and verify that voltages in the range 0 – 5 V can be generated.
3. Set potentiometer to generate approx. 750 mV. Check with the multimeter probe that the generated voltage passes through the port E protection card and arrives at actual port E pin PE4.

**AD CONVERSION TEST PROCEDURE**

When you are satisfied that the hardware works satisfactorily, proceed with the actual AD conversion experiment. Table 2 list six target voltage values. With your program running, do the following for each target voltage:

1. Reading with the multimeter, adjust the potentiometer to within the target value, and, when the reading is steady, record it in the 'Actual' column of Table 2.
2. Calculate the predicted value resulting from AD conversion. Perform the calculation in hex, then convert to binary. Enter the values in the corresponding column under the heading 'Predicted'.
3. Call TA to verify.
4. Proceed to the next target voltage.

**Table 2**

#	Voltage at pin PE4 (mV)		Digital values from AD conversion			TA check mark
	Target	Actual	Predicted		Displayed	
			Hex	Binary		
1	750		\$	%	%	
2	1600		\$	%	%	
3	2200		\$	%	%	
4	3000		\$	%	%	
5	4000		\$	%	%	
6	5000		\$	%	%	