Experiment #9

Flight86 Application II – Motor Control

9.0 Objectives:

The objective of this experiment is to control the operation and speed of a DC motor.

In this experiment, you will do the following:

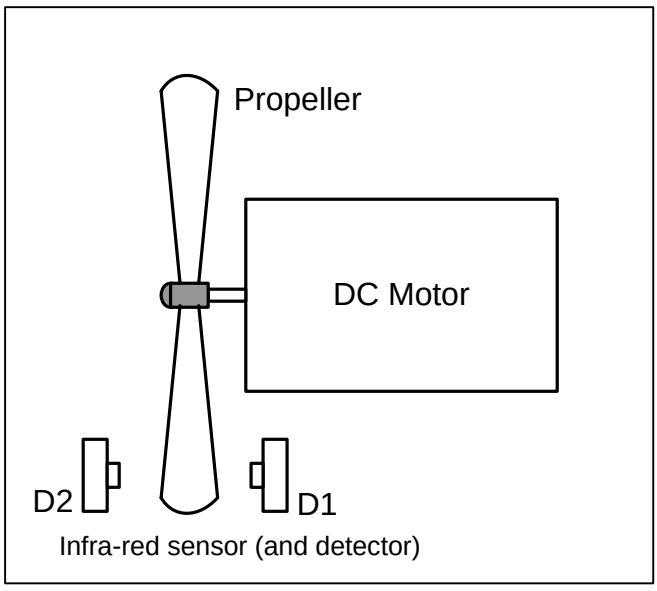
- Write a program to control the DC motor operation
- Write a program to vary the speed of the DC motor with a potentiometer
- Assemble, download, and test your programs on the trainer board

9.1 Equipment and Software

- Flight86 Trainer and Application Boards
- PC with Flight86 Monitor program
- Assembler and conversion utilities (exe2bin, bin2hex)

9.2 Introduction:

There is a small DC motor on the Application board. This motor with 3 bladed-propeller is limited to approx. 8000 RPM by the two current limiting resistors connecting it to the board. The polarity of the voltage applied to the motor and hence forward and reverse is selected by a relay. The motor is driven from the unregulated supply obtained from the mains adaptor (approx. 9V).



As the motor rotates, the propeller blades pass between D1, an infra-red source, and D2 an infra-red detector, the change in current through the detector provides a signal 3 times per revolution back to Port A bit 4. However, for this to function, SW4A must be set to 'SPEED'.

9.2.1 Motor Operation

Motor ON forward/reverse selection is by output bits 6 and 7 on Port B, their value is decoded by U4 (74LS139 – 2 x 4 decoder).

Port B		
Bit 7	**Bit 6**	**Motor Operation**
0	0	Stop
0	1	Forward motion
1	0	Reverse motion
1	1	Stop

Table 1: Motor Operation

To decode and drive any output of U4, first U4 must be enabled by placing SW2B in 'MOTOR' position.

In this experiment, we will write a program to control the operation of the Motor with two DIL switches available on the Applications Board.

9.2.2 Motor Speed Control

The speed controller works by varying the average voltage sent to the motor. It could do this by simply adjusting the voltage sent to the motor, but this is quite inefficient to do. A better way is to switch the motor's supply on and off very quickly. If the switching is fast enough, the motor doesn't notice it, it only notices the average effect.

Now imagine a light bulb with a switch. When you close the switch, the bulb goes on and is at full brightness, say 100 Watts. When you open the switch it goes off (0 Watts). Now if you close the switch for a fraction of a second, and then open it for the same amount of time, the filament won't have time to cool down and heat up, and you will just get an average glow of 50 Watts. This is how lamp dimmers work, and the same principle is used by speed controllers to drive a motor.

The speed of the Motor can be varied by turning the Motor *ON* and *OFF* very quickly.

If the supply voltage is switched fast enough, it won't have time to change speed much, and the speed will be quite steady. This is the principle of switch mode speed control. This principle of controlling the speed of the motor by turning the supply on and off very quickly is called Pulse Width Modulation (PWM).

As the amount of time that the voltage is *on* increases compared with the amount of time that it is *off*, the average speed of the motor increases.

9.2.3 Speed Control with a Potentiometer

The Potentiometer is an analogue input which provides a linear voltage between 0 and 2.55V. We will convert this voltage to a digital value with the help of the Analog-to-Digital Converter (ADC) provided on the board. The specifications of this ADC are given below:

Clock rate 400 KHz
Conversion time 180 us
Input 0.00 V for 00 Hex output Input 2.50 V for 80 Hex output Input 5.00 V for FF Hex output

This is an 8-bit ADC, free running at a clock rate of approx. 400 KHz. As the ADC is free running it is completely asynchronous with any 'read' from the microprocessor training board (Controller board), which means if read just as the output of the ADC is being updated, false readings could be obtained. This can be overcome by reading twice or more times, only accepting a value when it is unchanged over two consecutive readings. For most microprocessors, this should provide no problem in taking two readings in between conversions of the ADC.

To obtain a reliable reading, take reading two or more times, only accepting a value when it is unchanged over two consecutive readings.

In this experiment, we will control the speed of the Motor with a Potentiometer provided on the Applications Board. However, we will not use the Potentiometer to vary the voltage supplied to the Motor but to determine how long the motor should be turned OFF. The Potentiometer value will be converted to a digital value which will then be used to index into a table. This table will contain delay values for the duration for which the running motor will be turned OFF.

The delay values will be stored in a decreasing order so that higher Potentiometer voltage corresponds to lower delay value, and vice-versa.

As we know, the speed of the Motor can be varied by turning the Motor ON and OFF very quickly, we will use ON/OFF duration in multiples of 1ms (milliseconds). This will ensure that the speed of the Motor will be quite steady.

To make things simple, the 'ON' duration of the Motor will be kept constant. Only the 'OFF' duration will change depending on the Potentiometer value.

As the Potentiometer is turned to produce increasing voltage (up to 2.55V), the 'OFF' delay value selected from the table will be smaller, and thus, the speed of the Motor will be increased. Similarly, as the Potentiometer is turned to produce decreasing voltage (down to 0V), the 'OFF' delay value selected from the table will be larger, and thus, the speed of the Motor will be decreased.

The ADC will produce values in the range 00H to 80H (0 to approx. 130) corresponding to Potentiometer values (0 to 2.55V). This range of ADC values (0 to approx. 130) requires a table of 130 elements. This is quite large and not practical for a small application like this. To make matters easy, we will scale down the ADC range. If we divide the ADC value by 13, our effective range is scaled down to (0 to 10).

The delay must be implemented in multiples of 1ms. The following code can be used to implement a delay of approx. 1ms.

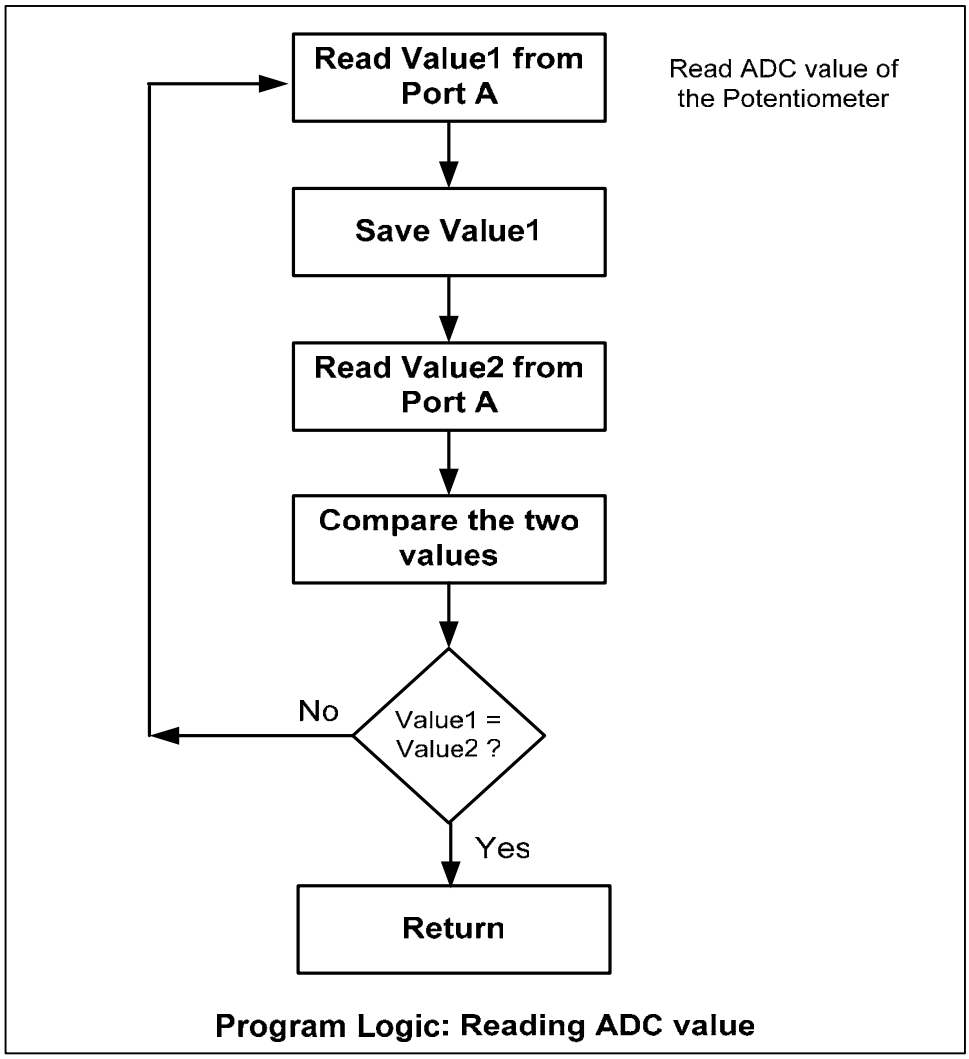
```
DELAY:  MOV CX, 270     ; Load CX with a fixed value
DEL1:   DEC CX          ; decrement CX
        JNZ DEL1        ; and loop if not zero
        RET             ; when CX=0, then exit
```

9.3 Pre-lab

1. Review the hardware specifications of the Flight86 system described in the experiment – **Introduction to Flight86 Trainer and Application Board**.
2. Read all the above sections of this experiment.
3. Write a subroutine to read the ADC value (which provides the Potentiometer value converted to a digital value) on Port A, when Switch SW2B is in Motor position and Switch SW3 is in VOLTS position. This subroutine will be used in Part 2 of the Lab Work of this experiment for Motor Speed Control.

To obtain a reliable reading, you program must take reading two or more times, only accepting a value when it is unchanged over two consecutive readings.

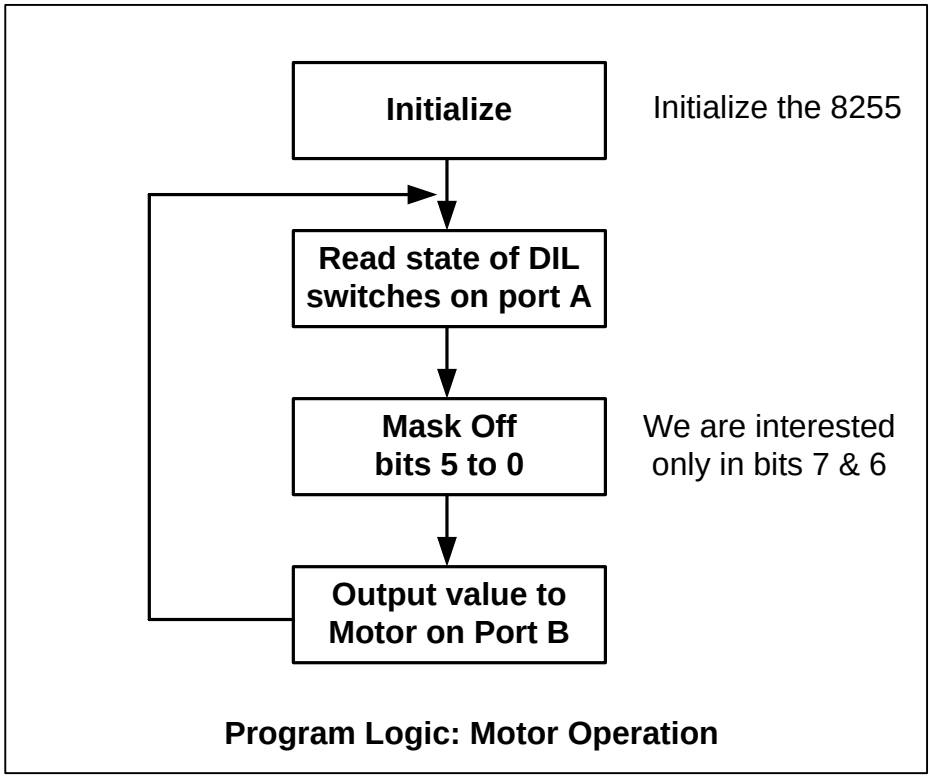
The logic for the read operation is illustrated in the flowchart below:



Program Logic: Reading ADC value

9.4 Lab Work – Part 1: Motor Operation

Write a program to control the operation of the DC Motor. The operation of the motor must be controlled by bits 7 and 6 of the DIL switches (according to Table 1 shown above). The logic of the program is illustrated by the flowchart shown below:



Program Logic: Motor Operation

Procedure:

1. Include the following initialization code at the beginning of your program.

```
INIT:
        MOV AL, 99          ; set up Port A IN, Port B OUT,  Port C IN
        OUT 07, AL          ; and output this word  to control Port
        MOV AL,0            ; data zero
        OUT 03,AL           ; output to Port B to ensure all LEDs off
```

2. Use an editor to write to your program. Name your file as ***motor_oper.asm***
3. Assemble and link your program using the TASM assembler to produce ***motor_oper.exe***.
4. Convert the ***motor_oper.exe*** file to binary format using the exe2bin.exe program by typing ***exe2bin motor_oper.exe*** at the DOS prompt.
5. Convert the ***motor_oper.bin*** file ***to motor_oper.hex*** using the bin2hex.exe program by typing ***bin2hex motor_oper.bin*** at the DOS prompt.

6. Start the Flight86 monitor program.
7. Download ***motor_oper.hex*** to the Flight86 controller board by typing at the '-' prompt

   : C:\FLIGHT86\ motor_oper.hex

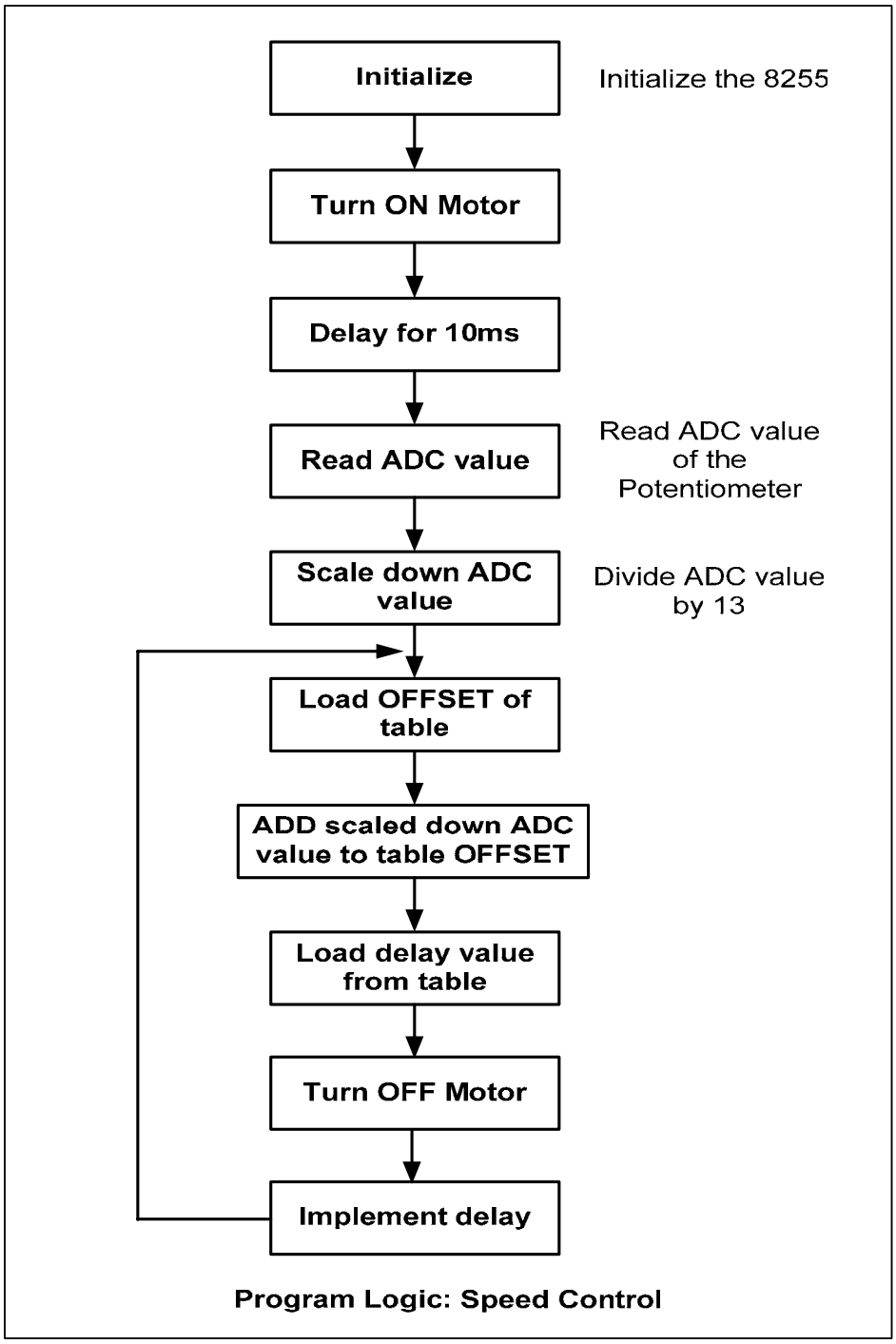
8. Before you run your program make sure the mode switches are in the correct position.

Switch SW2A – Switch position
Switch SW2B – Motor position
All other switches - OFF

9. Now enter **G 0050:0100** at the '-' prompt to run the program.
10. Check the operation of the Motor by changing the state of the two leftmost switches. Make sure the Motor operates according to Table 1.

9.5 Lab Work – Part 2: Motor Speed Control

Write a program to control the speed of the Motor. The speed of the motor must be controlled by the Potentiometer. The logic of the program is illustrated by the flowchart shown below:



Program Logic: Speed Control

Procedure:

1. Include the initialization code at the beginning of your program.
2. For your program, write a subroutine to read the ADC value. This subroutine can then be called using the CALL instruction. (Refer to section "**Speed Control with a Potentiometer**" for more information).
3. Place this table after the last instruction of your program.

```
TABLE   DB 0AH      ; 10 ms delay
        DB 09H      ; 9 ms delay
        DB 08H      ; 8 ms delay
        DB 07H      ; 7 ms delay
        DB 06H      ; 6 ms delay
        DB 05H      ; 5 ms delay
        DB 04H      ; 4 ms delay
        DB 03H      ; 3 ms delay
        DB 02H      ; 2 ms delay
        DB 01H      ; 1 ms delay
```

4. Use an editor to write to your program. Name your file as ***motor_speed.asm***
5. Assemble and link your program using the TASM assembler to produce ***motor_speed.exe***.
6. Convert the ***motor_speed.exe*** file to binary format using the exe2bin.exe program by typing ***exe2bin motor_speed.exe*** at the DOS prompt.
7. Convert the ***motor_speed.bin*** file ***to motor_speed.hex*** using the bin2hex.exe program by typing ***bin2hex motor_speed.bin*** at the DOS prompt.
8. Start the Flight86 monitor program.
9. Download ***motor_speed.hex*** to the Flight86 controller board by typing at the '-' prompt

   : C:\FLIGHT86\ motor_speed.hex

10. Before you run your program make sure the mode switches are in the correct position.

Switch SW2B – Motor position
Switch SW3 – VOLTS
All other switches – OFF

11. Now enter **G 0050:0100** at the '-' prompt to run the program.
12. Check the speed control of the Motor by turning the Potentiometer both ways.