Microcontroller System Laboratory

Experiment 6 - Motor

Utkarsh Patel 18EC35034

Part 1 – Motor rotation display on 7-segment display

Objective

Rotate motor in clockwise direction and display the number of revolutions on Display 0 (7-segment display). Change the direction of motor rotation from clockwise to anti-clockwise after every 5 revolutions. Display the direction of rotation on the display

Circuit Diagram

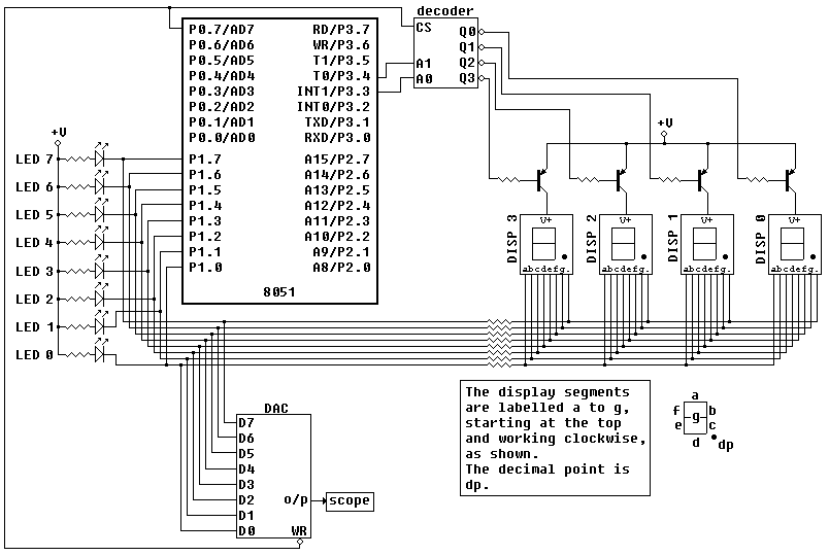

Fig. 1. Circuit diagram

Code

```
; Run this file with update frequency 10

start:
       MOV TMOD, #50H   ; put timer 1 in event counting mode
       SETB TR1         ; start timer 1

       MOV DPL, #LOW(LEDcodes)      ; | put the low byte of the start address of the
                                    ; | 7-segment code table into DPL

       MOV DPH, #HIGH(LEDcodes)     ; put the high byte into DPH

       CLR P3.4                     ; |
       CLR P3.3                     ; | enable Display 0

again:
       CALL setDirection     ; set the motor's direction
       MOV A, TL1            ; move timer 1 low byte to A
       CJNE A, #5, skip      ; if the number of revolutions is not 10 skip next instruction
       JMP changeDir1        ; if the number of revolutions is 10, reset timer 1
```

```
skip:
       MOVC A, @A+DPTR       ; | get the codes for 7-seg display
       MOV C, F0             ; move motor direction value to the carry
       MOV ACC.7, C          ; | and from there to ACC.7
       MOV P1, A             ; | move number of revolutions and motor direction
       JMP again             ; do it all again

setDirection:
       PUSH ACC              ; save value of A on stack
       PUSH 20H              ; | save value of location 20H (first bit-addressable
                             ; |    location in RAM) on stack
       CLR A                 ; clear A
       MOV 20H, #0           ; clear location 20H
       MOV C, P2.0           ; put SW0 value in carry
       MOV ACC.0, C          ; then move to ACC.0
       MOV C, F0             ; move current motor direction in carry
       MOV 0, C              ; and move to LSB of location 20H (which has bit address 0)

       CJNE A, 20H, changeDir; compare SW0 (LSB of A) with F0 (LSB of 20H)
       JMP finish            ; motor's direction does not need to be changed

changeDir:
       CLR P3.0              ; |
       CLR P3.1              ; | stop motor

       CALL clearTimer       ; reset timer 1
       MOV C, P2.0           ; move SW0 value to carry
       MOV F0, C             ; and then to F0 - this is the new motor direction
       MOV P3.0, C           ; move SW0 value (in carry) to motor control bit 1
       CPL C                 ; invert the carry
       MOV P3.1, C           ; | and move it to motor control bit 0

finish:
       POP 20H               ; get original value for location 20H from the stack
       POP ACC               ; get original value for A from the stack
       RET                   ; return from subroutine

clearTimer:
       CLR A                 ; reset revolution count in A to zero
       CLR TR1               ; stop timer 1
       MOV TL1, #0           ; reset timer 1 low byte to zero
       SETB TR1              ; start timer 1
       RET                   ; return from subroutine

changeDir1:
       CPL P2.0
       JMP again

LEDcodes:; | this label points to the start address of the 7-segment code table which is
         ; | stored in program memory using the DB command below
       DB 11000000B, 11111001B, 10100100B, 10110000B, 10011001B, 10010010B, 10000010B,
11111000B, 10000000B, 10010000B
```

Part 2 – Motor rotation display on LCD

Objective

Vary the speed of the motor manually (using the slider to the right of the motor). Display the revolutions count and direction on LCD, stop the motor rotation after the max. count that can be displayed.

Circuit Diagram

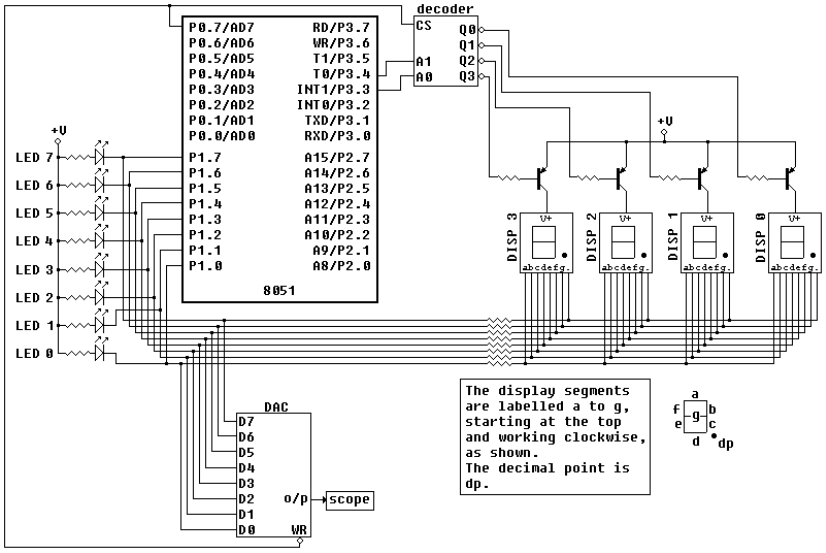

Fig. 2. Circuit diagram

Code

```
; Run this file with update frequency 100

ORG 0000H

MOV 50H, #'0'
MOV 51H, #'A'
MOV 52H, #'C'

MOV TMOD, #50H ; put timer 1 in event counting mode
SETB TR1               ; start timer 1

CLR P1.3               ; clear RS - indicates that instructions are being sent to the module

; function set
CLR P1.7               ; |
CLR P1.6               ; |
SETB P1.5              ; |
CLR P1.4               ; | high nibble set

SETB P1.2              ; |
CLR P1.2               ; | negative edge on E

CALL delay             ; wait for BF to clear

SETB P1.2              ; |
CLR P1.2               ; | negative edge on E

SETB P1.7              ; low nibble set (only P1.7 needed to be changed)

SETB P1.2              ; |
CLR P1.2               ; | negative edge on E
```

```
CALL delay            ; wait for BF to clear

; entry mode set
; set to increment with no shift
CLR P1.7              ; |
CLR P1.6              ; |
CLR P1.5              ; |
CLR P1.4              ; | high nibble set

SETB P1.2             ; |
CLR P1.2              ; | negative edge on E

SETB P1.6             ; |
SETB P1.5             ; |low nibble set

SETB P1.2             ; |
CLR P1.2              ; | negative edge on E

CALL delay            ; wait for BF to clear

; display on/off control
; the display is turned on, the cursor is turned on and blinking is turned on
CLR P1.7              ; |
CLR P1.6              ; |
CLR P1.5              ; |
CLR P1.4              ; | high nibble set

SETB P1.2             ; |
CLR P1.2              ; | negative edge on E

SETB P1.7             ; |
SETB P1.6             ; |
SETB P1.5             ; |
SETB P1.4             ; | low nibble set

SETB P1.2             ; |
CLR P1.2              ; | negative edge on E

CALL delay            ; wait for BF to clear

again:
       CALL setDirection     ; set the motor's direction
       MOV A, TL1                    ; move timer 1 low byte to A

       PUSH ACC

       ; converting revolution count to BCD
       MOV B, #10
       DIV AB
       MOV R2, B
       MOV B, #10
       DIV AB
       MOV R1, B
       SETB P1.3
       ADD A, 50H
       CALL sendChar
       MOV A, R1
       ADD A, 50H
       CALL sendChar
       MOV A, R2
       ADD A, 50H
       CALL sendChar
       MOV A, #0
       MOV C, F0
       MOV ACC.0, C
       MOV R1, A
```

```
        MOV A, #51H
        ADD A, R1
        MOV R1, A
        MOV A, @R1
        CALL sendChar
        CALL delay

        ; reset display to make room for next value
        CLR P1.3        ; lcd instruction mode on

        CLR P1.7        ; |
        CLR P1.6        ; |
        CLR P1.5        ; |
        CLR P1.4        ; | higher nibble value
        CALL pass       ; negative edge on enable
        SETB P1.4       ; | lower nibble value
        CALL pass       ; negative edge on enable
        CALL delay1

        POP ACC
        JMP again             ; do it all again

setDirection:
        PUSH ACC              ; save value of A on stack

        PUSH 20H              ; | save value of location 20H (first bit-addressable
                                    ; |    location in RAM) on stack

        CLR A                 ; clear A
        MOV 20H, #0           ; clear location 20H
        MOV C, P2.0           ; put SW0 value in carry
        MOV ACC.0, C   ; then move to ACC.0
        MOV C, F0             ; move current motor direction in carry
        MOV 0, C              ; and move to LSB of location 20H (which has bit address 0)

        CJNE A, 20H, changeDir              ; | compare SW0 with F0
        JMP finish

changeDir:
        CLR P3.0              ; |
        CLR P3.1              ; | stop motor

        CALL clearTimer       ; reset timer 1 (revolution count restarts when motor
direction changes)
        MOV C, P2.0           ; move SW0 value to carry
        MOV F0, C             ; and then to F0 - this is the new motor direction
        MOV P3.0, C           ; move SW0 value (in carry) to motor control bit 1
        CPL C                 ; invert the carry

        MOV P3.1, C           ; | and move it to motor control bit 0 (it will therefore have
the opposite
                                    ; | value to control bit 1 and the motor will start
                                    ; | again in the new direction)
finish:
        POP 20H               ; get original value for location 20H from the stack
        POP ACC               ; get original value for A from the stack
        RET                         ; return from subroutine

clearTimer:
        CLR A                 ; reset revolution count in A to zero
        CLR TR1               ; stop timer 1
        MOV TL1, #0           ; reset timer 1 low byte to zero
        SETB TR1              ; start timer 1
        RET                         ; return from subroutine

pass:               ; negative edge on enable
```

```
  SETB P1.2
  CLR P1.2
  MOV R7, #50     ; small delay for lcd buffer
  DJNZ R7, $
  RET

sendChar:    ; send data in accumlator to current address of DDRAM in LCD to display it
       MOV C, ACC.7   ; |
       MOV P1.7, C           ; |
       MOV C, ACC.6   ; |
       MOV P1.6, C           ; |
       MOV C, ACC.5   ; |
       MOV P1.5, C           ; |
       MOV C, ACC.4   ; |
       MOV P1.4, C           ; | high nibble set

       SETB P1.2             ; |
       CLR P1.2              ; | negative edge on E

       MOV C, ACC.3   ; |
       MOV P1.7, C           ; |
       MOV C, ACC.2   ; |
       MOV P1.6, C           ; |
       MOV C, ACC.1   ; |
       MOV P1.5, C           ; |
       MOV C, ACC.0   ; |
       MOV P1.4, C           ; | low nibble set

       SETB P1.2             ; |
       CLR P1.2              ; | negative edge on E

       CALL delay            ; wait for BF to clear
       RET

delay:
  MOV R7, #50
  DJNZ R7, $
  RET

delay1:
  MOV R7, #255
  DJNZ R7, $
  MOV R7, #255
  DJNZ R7, $
  MOV R7, #255
  DJNZ R7, $
  MOV R7, #255
  DJNZ R7, $
  MOV R7, #255
  DJNZ R7, $
  MOV R7, #255
  DJNZ R7, $
  RET
```

Discussion

1 Motor Module

- This program exercises the motor. The motor is rotated in a clockwise direction and the number of revolutions is displayed on Display 0 (the 7-segment display). The display only shows up to nine revolutions and then resets. The motor sensor is connected to P3.5, which is the external clock source for timer 1. Therefore, timer 1 is put into event counting mode. In this way, the the timer increments once every motor revolution.
- The value in timer 1 low byte is moved to A and this value together with the data pointer (DPH and DPL) are used to get the 7-segment code from program memory. The code is then sent to P1 to put the appropriate number on the Display 0.
- The motor can be changed from clockwise to anti-clockwise by pressing SW0 (on P2.0). The motor direction is stored in F0 (1 for clockwise, 0 for anti-clockwise). The value in F0 is sent to Display 0's decimal point (P1.7). This indicates the motor's direction - if the decimal point is lit, the motor is rotating anti-clockwise, while if it is not lit the motor is rotating clockwise.
- The value in F0 is compared with the value of SW0. If they are the same the motor direction does not need to be changed. If they are not the same it means the user has pressed SW0 and the motor direction must be reversed. When this happens the new motor direction is then stored in F0.

2 7-Segment Module

- To select multiplexed 7-segment displays as current display, P0.7 must be set to logic high, otherwise DAC will be the current display. By default, P0.7 is set to logic high.
- For this part, the mobile number is displayed on the display #3.
- To select display #3 as current display, P3.3 and P3.4 must be set to logic high.
- To display any character on 7-segment display, P1 must be set to corresponding value using (dp)gfedcba format. Logic high corresponds to OFF and logic low corresponds to ON.
- As my mobile number is 9547621111, the code for each character is as follow:

Character	Code
9	`10010000`
5	`10010010`
4	`10011001`
7	`11111000`
6	`10000010`
2	`10100100`
1	`11111001`
NULL	`11111111`

3 LCD Module

- The LCD module consists of 16 rows and 2 columns of 5x8 dot matrices.
- Name of the pins and their corresponding functions are as follows:

Pin Name	Function
VSS	Must be grounded
VCC	5V DC power supply
RS	Register Selection
R/W	Read/write
E	Enable
DB[7:0]	Data

- From the circuit diagram, it can be observed that P1.3 is the RS of the LCD display, and P1.2 is the E of the LCD display.
- To execute any set of command, a negative edge has to be generated by E, i.e., set P1.2 to logic high and then to logic low and add some delay

- There are two types of register modes:
    - **Command mode**: Indicates flow of instruction to the LCD module, RS (P1.3) must be set to logic low to select this register mode
    - **Data mode**: Indicates flow of data to the LCD module, RS (P1.3) must be set to logic high to select this register mode
- Every operation linked with LCD display has a unique hexadecimal code. Some of them are:

Code (in hexadecimal)	Operation
0F	LCD ON, cursor ON, blinking ON
01	Clear screen
02	Return home
04	Decrement cursor
06	Increment cursor
0E	Display ON, cursor OFF
80	Force cursor to the beginning of 1st line
C0	Force cursor to the beginning of 2nd line
38	Use 2 lines and 5x7 matrix
83	Cursor line 1 position 3
3C	Activate second line
08	Display OFF, cursor OFF
C1	Jump to second line, position 1
C2	Jump to second line, position 2
0C	Display ON, cursor OFF

- To perform any operation with code, say 0x75 = 01110101B, we divide the instruction into high nibble set (0111 in this case) and low nibble set (0101 in this case). Then we do:

```
CLR P1.7     ;  0|
SETB P1.6    ;  1|
SETB P1.5    ;  1|
SETB P1.4    ;  1| high nibble set
SETB P1.2    ;   |
CLR P1.2     ;   | negative edge on E
CLR P1.7     ;  0|
SETB P1.6    ;  1|
CLR P1.5     ;  0|
SETB P1.4    ;  1| low nibble set
SETB P1.2    ;   |
CLR P1.2     ;   | negative edge on E
CALL delay
```

- sendChar module is used for displaying the current character on the LCD display