```
ORG 0000H
MOV SP, #70H
MOV PSW, #00H
LED EQU P1.6 ; Define BUZZER pin as P1.5
LED25 EQU P0.0 ; for showing 25% cycle
LED50 EQU P0.1  ; for showing 50% cycle
LED75 EQU P0.2  ; for showing 75% cycle
LED100 EQU P0.3  ; for showing 100% cycle
;-------------------------------------------------------------
RS EQU P1.1
EN EQU P1.2
;-------------------------------------------------------------
INITIALIZATION:
MOV P0, #00H
MOV P1, #00H
MOV P2, #00H
MOV P3, #0FEH ;11111110 ; for keypad
;-------------------------------------------------------------

;--------------------------------------------------------------
LCD_INIT:
MOV R2, #38H ;INITIALIZING LCD, 2 LINES, 5X7 MATRIX
ACALL COMMAND ;CALLING COMMAND SUBROUTINE

MOV R2, #0EH ;DISPLAY ON, CURSOR ON
ACALL COMMAND

MOV R2, #01H ;CLEARING SCREEN
ACALL COMMAND
MOV A,#06H ;INCREMENT CURSOR POSITION
ACALL COMMAND
MOV R2, #80H ;CURSOR AT START OF 1ST LINE
ACALL COMMAND
MOV A,#3CH
ACALL COMMAND
MOV DPTR,#MSG ;Display Starting message
ACALL LCD_PRINT ;Subroutine for displaying in LCD
MOV DPTR,#MSG2
ACALL LCD_PRINT
MOV R2,#01H
ACALL COMMAND
MOV DPTR,#MSG3 ;Display Starting message
ACALL LCD_PRINT ;Subroutine for displaying in LCD
MOV DPTR,#MSG4
ACALL LCD_PRINT

MOV R2,#01H
ACALL COMMAND
```

```
MOV DPTR,#MSG5 ;Display Starting message
ACALL LCD_PRINT ;Subroutine for displaying in LCD
MOV DPTR,#MSG6
ACALL LCD_PRINT

MOV R2,#01H ;for removing all the prev messages and start new ones
ACALL COMMAND
MOV DPTR,#MSG7 ;Display Starting message
ACALL LCD_PRINT ;Subroutine for displaying in LCD
MOV DPTR,#MSG8
ACALL LCD_PRINT

MOV R2,#01H ;for removing all the prev messages and start new ones
ACALL COMMAND
MOV DPTR,#MSG9 ;Display Starting message
ACALL LCD_PRINT ;Subroutine for displaying in LCD
MOV DPTR,#MSG10
ACALL LCD_PRINT

MOV R2,#01H
ACALL COMMAND
MOV DPTR,#MSG11 ;Display Starting message
ACALL LCD_PRINT ;Subroutine for displaying in LCD
MOV DPTR,#MSG12
ACALL LCD_PRINT

MOV R2, #01H ; for removing all the prev messages and start new ones
ACALL COMMAND
MOV DPTR,#START1 ;Display Starting message
ACALL LCD_PRINT ;Subroutine for displaying in LCD
MOV DPTR,#START2
ACALL LCD_PRINT
MOV R2, #01H
ACALL COMMAND
MOV DPTR,#ACTIVE
ACALL LCD_PRINT
```

Upto this part we are actually printing the initial messages.

```
MSG:    DB 'EEE-4706',0

MSG2:   DB 'A1-GROUP-02',0

MSG3:   DB 'AFIF',0

MSG4:   DB 'ID:141',0

MSG5:   DB 'TASNIA',0
```

```
MSG6:   DB 'ID:113',0

MSG7:   DB 'FABBIHA',0

MSG8:   DB 'ID:105',0

MSG9:   DB 'FARUK',0

MSG10:   DB 'ID:133',0

MSG11:   DB 'SHERIFFO',0

MSG12:   DB 'ID:153',0
```

Initial messages; repeating same block of code to print all messages.

```
;LCD display subroutine----------------------------------------
LCD_PRINT:
LOOP:
CLR A
MOVC A, @A+DPTR
MOV R2, A
JZ EXIT ;stop writing when null char is detected
ACALL DISPLAY ;main display part
ACALL DELAY ;giving a short delay for LCD
INC DPTR
LJMP LOOP
EXIT:
MOV R2, #0C0H ;go to the 2nd line of LCD
ACALL COMMAND
RET

DISPLAY:
MOV P2, R2 ;subroutine for displaying in LCD
SETB RS ;RW is grounded in Hardware
SETB EN
ACALL DELAY
CLR EN
RET
;-----------------------------------------------------------
COMMAND:
MOV P2, R2 ;giving instructions to LCD
CLR RS
SETB EN
ACALL DELAY
CLR EN
ACALL DELAY
RET
;-----------------------------------------------------------
DELAY:
```

```
MOV 52H, #100 ;50 ms delay (approx)
D1: MOV 51H, #255
DJNZ 51H, $
DJNZ 52H, D1
RET
```

This part is directly taken from lab-03 on lcd to create different subroutines for printing command.

Part-02:

```
DECISION:
LJMP MOTOR_ROTATION ;Start Motor part

;MOTOR PART-----------------------------------------------------
MOTOR_ROTATION:
SETB P1.0
SETB P1.3 ;Input to motor driver
CLR P1.4 ;Clockwise direction
```

From this part we dive into the starting of motor. We set P1.0 (enable pin of driver) to start the driver and our default direction of rotation is clockwise so we set IN1 high and IN2 low.(opposite for anticlockwise)

```
STARTUP_MOTOR:
MOV R2, #01H
ACALL COMMAND
MOV DPTR, #INT_MSG ;print 'ENTER RPM LEVEL'
LCALL LCD_PRINT
MOV R2, #01H
ACALL COMMAND
MOV DPTR, #LEVEL ;print 'LEVEL: 1 TO A'
LCALL LCD_PRINT
LCALL DELAY
MOV DPTR, #STOP ;print 'PRESS 0 TO STOP'
LCALL LCD_PRINT
MOV DPTR, #ACLCL
LCALL LCD_PRINT
```

This part of the code displays all the messages and options of controlling the operation of motor.

This includes-

INT_MSG: DB 'ENTER RPM LEVEL',0 ;INITIAL MESSAGE

LEVEL: DB 'LEVELS: 1 TO A',0

STOP: DB 'PRESS 0 TO STOP',0

ACLCL: DB 'ACL: * CL: #',0

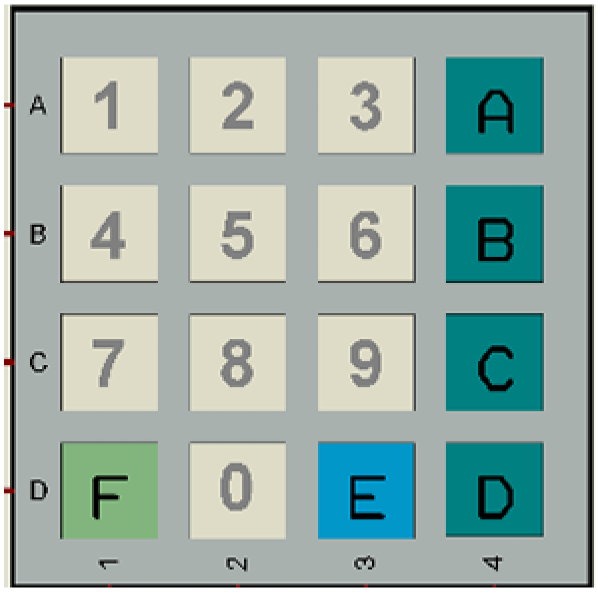
Part-03: Key board scanning

In this part we are not following the conventional method described in the lab sheet. In the lab sheet one port was used as input and another one was used as output. Here we are using the port 3 for both rows and columns of keypad. (P3.0-P3.3 for rows and P3.4-P3.7 for columns). At the initialization section of code we defined MOV P3,#0FEH it denotes that in the upper 4 bits(P3.4-3.7;columns) e have 1111 like the code in lab sheet and in the lower four bits(P3.0-P3.3;rows) we have 1110(to start checking the 1st row)

```
;SCANNING ALL ROWS -----------------------------------------
ML:
JNB P3.0, CC1
JNB P3.1, CC2
JNB P3.2, CC3
JNB P3.3, CC4
SJMP ML
```

Like I said earlier we already sent 1110 to the row so we will check whether any key was pressed in the column to get the corresponding digit from the keypad. For this we check if there is a zero value in the first row if so then we for the checking of corresponding column values .

To mention the structure of our keypad is:



Last row we have *,0,=,D instead of(F,0,E,D)

```
;SCANNING COLUMN1 ---------------------------------------------
CC1:
JNB P3.4, JMP_TO_1
JNB P3.5, JMP_TO_2
JNB P3.6, JMP_TO_3
JNB P3.7, JMP_TO_A
```

```
SETB P3.0
CLR P3.1
SJMP ML
```

Therefore we check whether we are getting a zero value is C1 C2 C3 C4 or not and based on that we jump to four different levels referring the symbols as per our keypad design of 1st row.

This is for the 1st row(JNB P3.0 we go to scan column 1). Similarly for rest of the rows we scan rest of the columns labeled by CC2,CC3,CC4

```
CC2:
JNB P3.4, JMP_TO_4
JNB P3.5, JMP_TO_5
JNB P3.6, JMP_TO_6
JNB P3.7, JMP_TO_B

SETB P3.1
CLR P3.2
SJMP ML
;SCANNING COLUMN3 --------------------------------------------

CC3:
JNB P3.4, JMP_TO_7
JNB P3.5, JMP_TO_8
JNB P3.6, JMP_TO_9
JNB P3.7, JMP_TO_C

SETB P3.2
CLR P3.3
SJMP ML
;SCANNING COLUMN4 --------------------------------------------

CC4:
JNB P3.4, JMP_TO_F
JNB P3.5, JMP_TO_0
JNB P3.6, JMP_TO_E
JNB P3.7, JMP_TO_D

SETB P3.3
CLR P3.0
LJMP ML
```

After checking 1st column, if none of the keys in the column were pressed then we need to go to the second row and need to send 1101 to the ports. That's why we do –

SETB P3.0
CLR P3.1
SJMP ML .**Set the second row and first one 1 to send 1101 . Same for all other column operations.**

ASCII LOOK-UP TABLE FOR EACH ROW

KCODE0: DB '1','2','3','A'   ;ROW 0

KCODE1: DB '4','5','6','B'   ;ROW 1

KCODE2: DB '7','8','9','C'   ;ROW 2

KCODE3: DB '*','0','=','D'   ;ROW 3

Part-04:

```
JMP_TO_0: LJMP M_0 ;MOTOR OFF
JMP_TO_1: LJMP M_1 ;LEVEL 1
JMP_TO_2: LJMP M_2 ;LEVEL 2
JMP_TO_3: LJMP M_3 ;LEVEL 3
JMP_TO_4: LJMP ML ;NOT NEEDED
JMP_TO_5: LJMP ML ;continue scanning
JMP_TO_6: LJMP ML
JMP_TO_7: LJMP ML
JMP_TO_8: LJMP ML
JMP_TO_9: LJMP ML
JMP_TO_A: LJMP M_A ;LEVEL 4
JMP_TO_B: LJMP ML
JMP_TO_C: LJMP ML
JMP_TO_D: LJMP ML
JMP_TO_E: LJMP M_E ;CLOCKWISE
JMP_TO_F: LJMP M_F ;ANTI-CLOCKWISE
```

As mentioned earlier to take the input from the user to control the operation of motor we are using keypad. For that after getting an input from the keypad we jump to corresponding mode of operation we want to do. Here one thing to remember is that we are using only 1st and last row in our entire operation. Any input from 2nd and 3rd row will not affect the operation. That is why we have declared labels with the corresponding inputs from the 1st (1,2,3,A)and last rows (*,0,=,D) and for the rest of the rows we have used LJMP ML to go back to the initial stage of scanning keypad cause we donot want any input other than 1st and last row

Now we will define different mode of operation of the motor

```
M_0:
JNB P3.5,M_0 ;wait for key release
MOV R2, #01H
ACALL COMMAND ;clear LCD screen
MOV DPTR, #OFF_MSG ;display 'Motor off'
ACALL LCD_PRINT
```

```
LJMP MOTOR_0 ;MOTOR OFF STATE
CLR LED ; Turn off LED when '0' is pressed
```

This label is used whenever we press 0 in the keyboard it will display the message :

OFF_MSG: DB 'MOTOR OFF',0

And to show that it will send command to LCD. After that it will go to the corresponding MOTOR_0 level to execute the task of stopping the motor. Similarly for the rest of the 4 steps.

```
M_1:
JNB P3.4,M_1
MOV R2, #01H
ACALL COMMAND
MOV DPTR, #LVL1
ACALL LCD_PRINT
MOV DPTR, #STOP
ACALL LCD_PRINT
LJMP MOTOR_1 ;RPM LEVEL 1

M_2:
JNB P3.5,M_2
MOV R2, #01H
ACALL COMMAND
MOV DPTR, #LVL2
ACALL LCD_PRINT
MOV DPTR, #STOP
ACALL LCD_PRINT
LJMP MOTOR_2 ;RPM LEVEL 2

M_3:
JNB P3.6,M_3
MOV R2, #01H
ACALL COMMAND
MOV DPTR, #LVL3
ACALL LCD_PRINT
MOV DPTR, #STOP
ACALL LCD_PRINT
LJMP MOTOR_3 ;RPM LEVEL 3

M_A:
JNB P3.7,M_A
MOV R2, #01H
ACALL COMMAND
MOV DPTR, #LVL4
ACALL LCD_PRINT
MOV DPTR, #STOP
ACALL LCD_PRINT
```

```
LJMP MOTOR_4 ;RPM LEVEL 4
M_F:
JNB P3.4,M_F
MOV R2, #01H
ACALL COMMAND
MOV DPTR, #ANTICLK ;Display 'Anticlockwise'
ACALL LCD_PRINT
MOV DPTR, #CLM
ACALL LCD_PRINT
SETB P1.0
SETB P1.4 ;anticlockwise rotation
CLR P1.3
LJMP ML ;scan for new input

M_E:
JNB P3.6,M_E
MOV R2, #01H
ACALL COMMAND
MOV DPTR, #CLKWISE
ACALL LCD_PRINT
MOV DPTR, #CLM
ACALL LCD_PRINT
SETB P1.0
SETB P1.3 ;clockwise rotation (In1=1,In2=0)
CLR P1.4
LJMP ML
```

LVL1: DB 'L1:75 RPM,25%PW',0

LVL2: DB 'L2:150 RPM,50%PW ',0

LVL3: DB 'L3:225 RPM,75%PW ',0

LVL4: DB 'L4:300 RPM,100PW',0

ANTICLK: DB 'ANTI-CLOCKWISE',0

CLM: DB 'CLK   : PRESS #',0

CLKWISE: DB 'CLOCKWISE',0

ACLM: DB 'ANTICLK: PRESS *',0

And in each level we are always printing the message: how to stop the motor. That is why it is always incorporated in each level (MOV DPTR, #STOP
ACALL LCD_PRINT)

Part-05:

```
MOTOR_0:
CLR P1.0 ;MOTOR OFF
CLR P1.3
CLR P1.4
JNB P3.4,LL ;JNB IS SHORT JUMP,
JNB P3.5,LL ;SO LL IS USED AS AN INTERMEDIATE STAGE
JNB P3.6,LL
JNB P3.7,LL
SETB P3.0
CLR P3.3 ;CHECK ROW 4 (* 0 # D)
JNB P3.4,LC ;'*' PRESSED
JNB P3.5,LC ;'0' PRESSED
JNB P3.6,LC ;'#' PRESSED
SETB P3.3
CLR LED ; Turn off LED when '0' is pressed
CLR LED25
CLR LED50
CLR LED75
CLR LED100
CLR P3.0 ;AGAIN CHECK ROW 1(1 2 3 A)
SJMP MOTOR_0
```

In this part to halt the motor we need to set 0 to IN1 and IN2(CLR P1.3 and P1.4). Then we check if any key is pressed from row 1 or last row(motor will again start). To do so we at first check if any column is pressed then go to the start of the keypad to check which data is pressed and do the corresponding operation. To check whether there is any input from the last row we set P3.0 to 1 and P3.3 to 0 and check whether *,0,# which one is pressed. Then it will jump to LC label.

```
LC: MOV P3, #0FEH ;INITIAL CONDITION

LJMP ML ;SCAN KEYPAD AGAIN

;------------------------------------------------------------

LL: LJMP ML ;JUMP TO L1

;------------------------------------------------------------
```

It initializes the port and jump to ML from where we start scanning the keypad to check after pressing 0 whether we want to start our motor or not. If after stopping the motor we press any keys from the 1st row (1,2,3,A) or last row (*,#) motor will again start.

```
MOTOR_1: ;25% DUTY CYCLE
SETB LED25
CLR LED50
CLR LED75
CLR LED100
CLR LED; buzzer
SETB P1.0
SETB P1.3
CLR P1.4
ACALL DELAYM
CLR P1.3
CLR P1.4
ACALL DELAYM
ACALL DELAYM
ACALL DELAYM
ACALL DELAYM
JNB P3.4,LL ;CHECK FOR NEW INPUT
JNB P3.5,LL ;KEY 1,2,3,A FOR LEVEL 1-4
JNB P3.6,LL
JNB P3.7,LL
SETB P3.0
CLR P3.3
JNB P3.4,ACL1_1 ;'*' FOR ANTI-CLOCKWISE
JNB P3.5,LC ;GO OUT WHEN '0' IS PRESSED
JNB P3.6,CL1_1 ;# FOR CLOCKWISE
SETB P3.3
CLR P3.0
;JNB P1.0, LIM1
LJMP MOTOR_1
```

There are 4 different labels of operation. Level-1,(25%PW),2-50%,3-75% and 4=100%. Motor_1 refers to the 25% PW level. So whenever we are operating at 25% PW level corresponding led25 will be turned on and other leds will be turned off. Default rotation is clockwise so we set IN1 (P1.3) to high value and IN2 (P1.4) to low value then call a delay. Then clear IN1 and IN2 and call delay 4 times so (1/4*100=25%). Then we check for new input from the first row to shift to another rpm level. If so we jump to LL and start scanning the keypad again as stated earlier. Now if we want to rotate the motor anticlockwise then we need to scan for any input from the last row. If 0 is pressed then we will go out of the loop and jump to Motor_0 to halt the operation of the motor. If any key other than 0 is pressed then we will go to the corresponding ACL_1 or CL_1 level. If no key is pressed it will remain in MOTOR_1 state.

```
ACL1_1:LJMP ACL1
CL1_1: LJMP CL1
```

An intermediate stage from which we will jump to ACL1 and CL1.

```
CL1:ACALL CL_MAIN ;FOR CLOCKWISE ROTATION
LJMP MOTOR_1 ;CONTINUE SAME RPM LEVEL
```

```
ACL1:ACALL ACL_MAIN
LJMP MOTOR_1_ACL
;-----------------
```

Based on my input (whether want clockwise or anticlockwise) after coming to this label it will direct my operation. If we press clockwise(= from keypad) then it will just simply send the command to MOTOR_1 since default direction of rotation was clockwise. If * is pressed then it will go to ACL_MAIN.

```
ACL_MAIN:
MOV R2, #01H
ACALL COMMAND
MOV DPTR, #ANTICLK
ACALL LCD_PRINT
MOV DPTR, #CLM
ACALL LCD_PRINT
RET
```

Here it will just print the message on the LCD and return back to ACL_1 main.

ANTICLK: DB 'ANTI-CLOCKWISE',0

CLM: DB 'CLK   : PRESS #',0

Similar for clockwise if = was pressed.

```
CL_MAIN:
MOV R2, #01H
ACALL COMMAND
MOV DPTR, #CLKWISE
ACALL LCD_PRINT
MOV DPTR, #ACLM
ACALL LCD_PRINT
RET
```

CLKWISE: DB 'CLOCKWISE',0

ACLM: DB 'ANTICLK: PRESS *',0

After coming back to ACL_1 main it will now jump to MOTOR_1_ACL where same 25% operation of motor is defined for anticlockwise direction.

```
MOTOR_1_ACL: ;25% DUTY CYCLE
SETB LED25
CLR LED50
CLR LED75
CLR LED100
CLR LED
SETB P1.0
SETB P1.4
CLR P1.3
ACALL DELAYM
CLR P1.4
CLR P1.3
ACALL DELAYM
ACALL DELAYM
ACALL DELAYM
ACALL DELAYM
JNB P3.4,LL_2 ;CHECK FOR NEW INPUT
JNB P3.5,LL_2 ;KEY 1,2,3,A FOR LEVEL 1-4
JNB P3.6,LL_2
JNB P3.7,LL_2
SETB P3.0
CLR P3.3
JNB P3.4,ACL1 ;'*' FOR ANTI-CLOCKWISE
JNB P3.5,LC1_2 ;GO OUT WHEN '0' IS PRESSED
JNB P3.6,CL1 ;# FOR CLOCKWISE
SETB P3.3
CLR P3.0
SJMP MOTOR_1_ACL
```

We start in the same way turn on the led25 for 25% PW operation and turn off the all other leds. Then set enable and set IN2 to 1 and IN1 to zero for anticlockwise rotation. Call delays (1:4) times for 25% cycle and check whether we want to stay in this loop or go for any other rpm level. If we want to check direction back to clockwise then it will jump to CL1 and then go back to MOTOR_1 for clockwise rotation. If no key is pressed then it will stay in this loop.

If this part is clear then we will do the same for 50%,75% and 100% PW

```
MOTOR_2: ;50% DUTY CYCLE
CLR LED25
CLR LED75
CLR LED100
CLR LED
SETB LED50
SETB P1.0
SETB P1.3
CLR P1.4
ACALL DELAYM
ACALL DELAYM
CLR P1.3
CLR P1.4
```

```
ACALL DELAYM
ACALL DELAYM
JNB P3.4,LL_1
JNB P3.5,LL_1
JNB P3.6,LL_1
JNB P3.7,LL_1
SETB P3.0
CLR P3.3
JNB P3.4,ACL2_1 ;'*' pressed
JNB P3.5,LC1_1
JNB P3.6,CL2_1
SETB P3.3
CLR P3.0
;JNB P1.0, LIM2
SJMP MOTOR_2
```

```
ACL2_1: LJMP ACL2
CL2_1: LJMP CL2

CL2: ACALL CL_MAIN
LJMP MOTOR_2
ACL2: ACALL ACL_MAIN ;rotation anticlockwise
LJMP MOTOR_2_ACL ;continue with the same RPM


ACL_MAIN:
MOV R2, #01H
ACALL COMMAND
MOV DPTR, #ANTICLK
ACALL LCD_PRINT
MOV DPTR, #CLM
ACALL LCD_PRINT
RET

MOTOR_2_ACL: ;50% DUTY CYCLE
CLR LED25
CLR LED75
CLR LED100
CLR LED
SETB LED50
SETB P1.0
SETB P1.4
CLR P1.3
ACALL DELAYM
ACALL DELAYM
```

```
CLR P1.4
CLR P1.3
ACALL DELAYM
ACALL DELAYM
JNB P3.4,LL_12
JNB P3.5,LL_12
JNB P3.6,LL_12
JNB P3.7,LL_12
SETB P3.0
CLR P3.3
JNB P3.4,ACL2 ;'*' pressed
JNB P3.5,LC1
JNB P3.6,CL2
SETB P3.3
CLR P3.0
;JNB P1.0, LIM2A
SJMP MOTOR_2_ACL
```

75% PW

```
MOTOR_3: ;75% DUTY CYCLE
CLR LED
CLR LED25
CLR LED50
CLR  LED100
SETB LED75
SETB P1.0
SETB P1.3
CLR P1.4
;SETB LED75
ACALL DELAYM
ACALL DELAYM
ACALL DELAYM
CLR P1.3
CLR P1.4
ACALL DELAYM
ACALL DELAYM
ACALL DELAYM
ACALL DELAYM
JNB P3.4,LL1_12
JNB P3.5,LL1_12
JNB P3.6,LL1_12
JNB P3.7,LL1_12
SETB P3.0
CLR P3.3
JNB P3.4,ACL3_1
JNB P3.5,LC1
JNB P3.6,CL3_1
SETB P3.3
CLR P3.0
;JNB P1.0, LIM3
SJMP MOTOR_3
```

```
ACL3_1:LJMP ACL3
CL3_1:LJMP CL3


CL3:ACALL CL_MAIN
LJMP MOTOR_3
ACL3:ACALL ACL_MAIN
LJMP MOTOR_3_ACL


MOTOR_3_ACL: ;75% DUTY CYCLE
CLR LED
CLR LED25
CLR LED50
CLR  LED100
SETB LED75
SETB P1.0
SETB P1.4
CLR P1.3
ACALL DELAYM
ACALL DELAYM
ACALL DELAYM
CLR P1.4
CLR P1.3
ACALL DELAYM
ACALL DELAYM
ACALL DELAYM
ACALL DELAYM
JNB P3.4,LL1_1
JNB P3.5,LL1_1
JNB P3.6,LL1_1
JNB P3.7,LL1_1
SETB P3.0
CLR P3.3
JNB P3.4,ACL3
JNB P3.5,LC11_1
JNB P3.6,CL3
SETB P3.3
CLR P3.0
;JNB P1.0, LIM3A
```

```
LJMP MOTOR_3_ACL
```

100%

```
MOTOR_4: ;KEY 'A'
CLR LED25
CLR LED50
CLR LED75
SETB LED100
SETB P1.0 ;100% DUTY CYCLE
SETB P1.3
CLR P1.4
SETB LED ; Turn on LED when motor reaches 3000 RPM

ACALL DELAYM
ACALL DELAYM
ACALL DELAYM
ACALL DELAYM
JNB P3.4,LL1_2
JNB P3.5,LL1_2
JNB P3.6,LL1_2
JNB P3.7,LL1_2
SETB P3.0
CLR P3.3
JNB P3.4,ACL4_1
JNB P3.5,LC111_1
JNB P3.6,CL4_1
SETB P3.3
CLR P3.0
;JNB P1.0, LIM4
CLR P1.0
SJMP MOTOR_4
```

```
ACL4_1:LJMP ACL4
CL4_1: LJMP CL4


CL4:ACALL CL_MAIN
```

```
LJMP MOTOR_4
ACL4:ACALL ACL_MAIN
SJMP MOTOR_4_ACL


OTOR_4_ACL: ;KEY 'A'
CLR LED25
CLR LED50
CLR LED75
SETB LED100
SETB P1.0 ;100% DUTY CYCLE
SETB P1.4
CLR P1.3
ACALL DELAYM
ACALL DELAYM
ACALL DELAYM
ACALL DELAYM
JNB P3.4,LL1
JNB P3.5,LL1
JNB P3.6,LL1
JNB P3.7,LL1
SETB P3.0
CLR P3.3
JNB P3.4,ACL4
JNB P3.5,LC111_2
JNB P3.6,CL4
SETB P3.3
CLR P3.0
;JNB P1.0, LIM4A
CLR P1.0
LJMP MOTOR_4_ACL
```

That is all.

For overload protection when the motor is operating at 25% PW meaning that load is excess and speed is very low. In this regard what we can do is whenever we are setting the rpm level to 25% PW after a few seconds of operation the enable pin of the driver can be set to 0 so that the motor is disconnected. So we need to change the part of code MOTOR_1 and MOTOR_1 ACL a bit to do this task.