Eastern Mediterranean University

Department Of Electrical & Electronics Engineering

Course: Microprocessor I

Instructor: *Prof. Dr. Hasan Demirel*

Project: Hardware Interfacing via parallel port

Students: Enes Ergul 110741, Mohamad Alhaddad 139098

Abstract:

The project main aim is interfacing hardware components with desktop PC processor through parallel port.

Components:

a) Hardware:

1) 7- segment display (used anode 7 segment display)
2) 7- segment display driver (BCD – TO – SEVEN – SEGMENT – DECODER), model: SN7446A
3) DC motor driver, model: L239D

b) Software:

EMU8086 Assembler

Steps:

User will be able to control the motor **speed** and **rotation** via keyboard buttons ('u','d') and mouse buttons (left, right) respectively, so program flow will be divided into 3 main sections as follows:

1) Handling inputs:

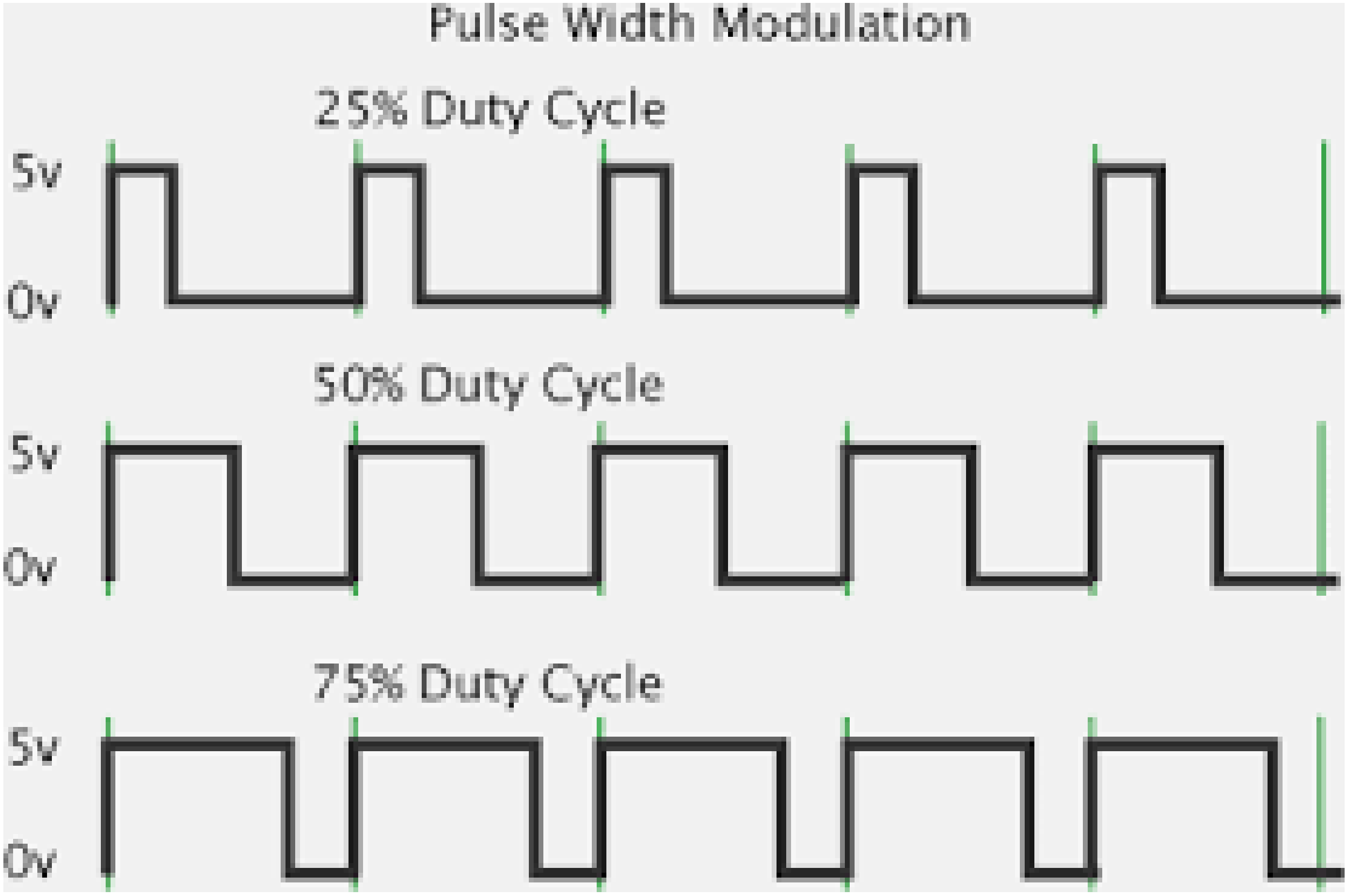
Since the user will input through standard input devices (mouse, keyboard) the program should handle this inputs and associate actions accordingly. This is achieved with the usage of "STDIO.mac" custom library that will run the appropriate interrupts through its functions to handle such inputs.

2) Parallel port interfacing:

Since no inputs from the physical world into the program, we will only use the default *output* mode by using the instruction **OUT** given the first argument *DX* which contains the address of the port (0378H) and the second argument *AL* which contains a Byte of data to be sent through port.

3) Speed control:

We will implement Pulse Width Modulation (PWM) to control the speed of the motor, basically that technique depends on how much is the device **ON** & **OFF** per period cycle (figure 1).



The figure above gives a basic idea about what is PWM.

****For codes check appendix***

Schematic:

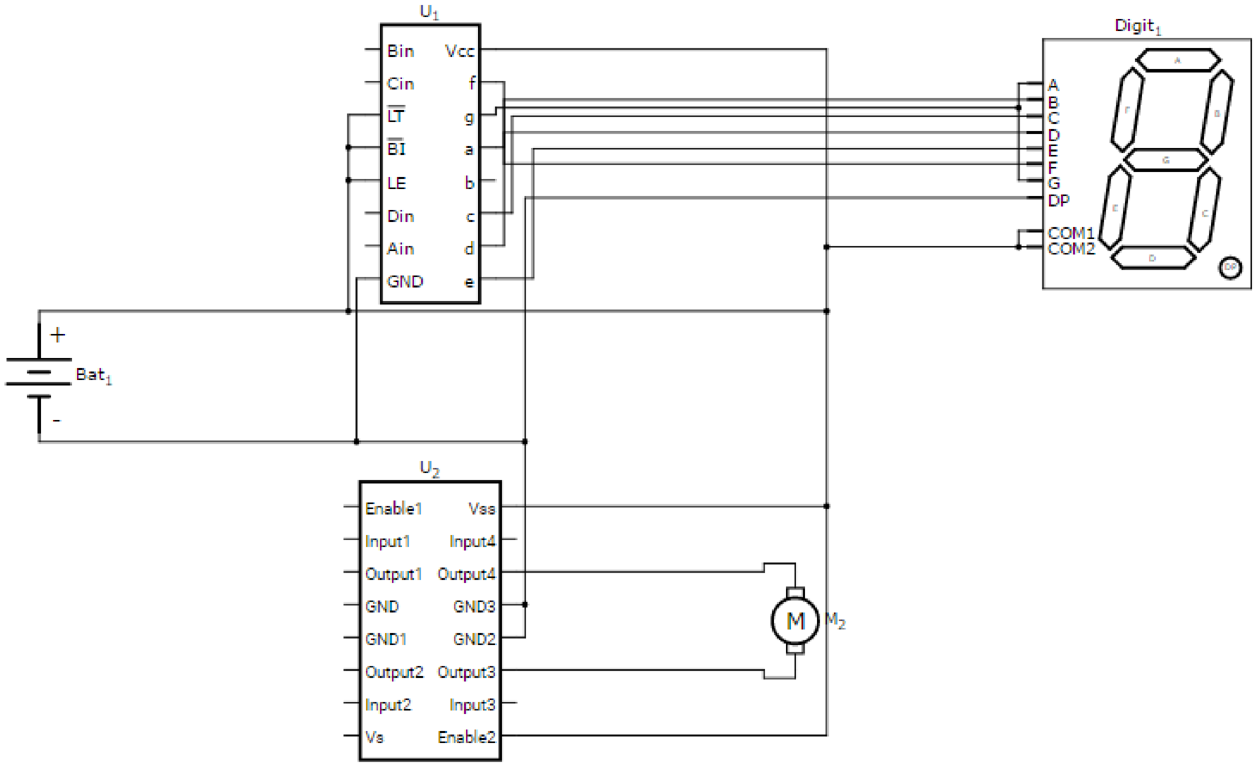


Figure2

Conclusion:

Controlling hardware components through the usage of software driven by user inputs, is called *Human Computer Interface* thus this project is pretty basic however, it's an implementation of such field of study with low level programming language (assembly) which adds some challenging to the implementation.

Appendix:

STDIO functions signatures:

- **INIT_MOUSE:** Initialize mouse object with program
- **SHOW_MOUSE:** will show mouse cursor on the screen
- **HIDE_MOUSE:** will hide mouse cursor..
- **HANDLE_MOUSE_CLICK (btn):** will detect which mouse button was clicked
- **DISPLAY (str):** display a string on screen
- **ASYNC_GETCHR:** will detect key press, and get the *ASCII* code of it (non-blocking)

STDIO.mac:

```
; Standard Input Output

INIT_MOUSE MACRO
        MOV AX, 0
        INT 33H
ENDM

SHOW_MOUSE MACRO
        MOV AX, 01H
        INT 33H
ENDM

HIDE_MOUSE MACRO
        MOV AX, 02H
        INT 33H
ENDM

HANDLE_MOUSE_CLICK MACRO BTN
    LOCAL LEFT_, RIGHT_, MIDDLE_
        MOV BX, 0
        MOV AX, 03H
        INT 33H

                        CMP BL, 00H
                        JE ENDIF;
        CMP BL, 01H
        JE LEFT_
        CMP BL, 02H
        JE RIGHT_
        CMP BL, 04H
        JE MIDDLE_
        JMP ENDIF

    LEFT_:    MOV BTN, 'l'
           JMP ENDIF
    RIGHT_:   MOV BTN, 'r'
           JMP ENDIF
    MIDDLE_:  MOV BTN, 'm'

    ENDIF:    ;
ENDM

DISPLAY         MACRO           STR
                MOV AH, 09H
                MOV DX, OFFSET STR
                INT 21H
ENDM

DISPLAY_N MACRO    STR
        DISPLAY STR
        DISPLAY_CHR 10
        DISPLAY_CHR 13
ENDM

DISPLAY_CHR MACRO CHAR
   MOV DL, CHAR
   MOV AH, 02H
   INT 21H
ENDM

GETSTR MACRO STR
ENDM
; detect key press from user blocking
SYNC_GETCHR         MACRO           TO
                MOV AH, 01H
                INT 21H
                MOV TO, AL
                AND AL, 0
ENDM
; detect key press from user non-blocking
ASYNC_KEY MACRO
        MOV AH, 01H
        INT 16H
ENDM
; get character from user non-blocking
ASYNC_GETCHR MACRO CHR
        LOCAL PASS
        MOV CHR, 0
        ASYNC_KEY
        JZ PASS ; no key pressed
        ; key pressed
        MOV AH, 0
```

```
          INT 16H
          MOV CHR, AL
          PASS: ;
ENDM
```

Main Program:

```
org 100h


                                  -
; Trigger direction of rotation
SET_DIRECTION MACRO DIRECTION
   MOV BL, DIRECTION
   MOV MOTOR_DIRECTION, BL
ENDM

INIT_DIRECTION MACRO
   SET_DIRECTION 0
ENDM


; Run the motor
RUN_MOTOR MACRO
    LOCAL PASS1
   ; Get motor parameters
      ; Get the direction of rotation
      MOV AL, MOTOR_DIRECTION
      ; Get the speed
      MOV BL, CURRENT_SP
      ; Get Port
      MOV DX, PORT_


   CMP BL,0 ; CHECK IF SPEED IS 0, SO ITS OFF
   JE PASS1

   ; Turn On
   OR AL, BL
   OUT DX, AL

   SET_DELAY BX
   ;JMP EXIT
   ;
PASS1:
   ; Turn Off
   AND AL, 0FH
   OUT DX, AL

   ; EXTRACT DIFFERENCE BETWEEN
   ; MAX SPEED AND CURRENT SPEED
   ; AND SET THE DELAY ACCORDINGLY
   ; ASSUMING THE DELAY IS ABOUT 1 SECOND
   ; SO THE DELAY WILL BE DIVIDED INTO MAXIMUM SPEED OF SEGMENTS

   MOV BH, MOTOR_MAX_SP
   SUB BH, BL
   MOV BL, BH
   AND BH,0
   SET_DELAY BX


   EXIT:;
   ;

ENDM


; NOT READY
UPDATE_SPEED MACRO DIRECTION
   LOCAL P1, P2, EXIT
   MOV DX, PORT_
   MOV AL, CURRENT_SP
   MOV BL, DIRECTION
   ;
   CMP BL, 'u'
   JE P1
   CMP BL, 'd'
   JE P2
   ;

   P1:
     CMP AL, 09H
     JE EXIT
     ;
     INC AL
     JE EXIT
   P2:
     CMP AL, 00H
     JE EXIT
     DEC AL
     ;
     JE EXIT

   EXIT:
      MOV CURRENT_SP, AL
      OUT DX, AL

ENDM


DEC_SPEED MACRO
   LOCAL EXIT
   MOV BL, CURRENT_SP
   CMP BL, 0
   JE EXIT
   DEC BL
   MOV CURRENT_SP, BL
   EXIT: ;
ENDM
```

```
INC_SPEED MACRO
   LOCAL EXIT
   MOV BL, CURRENT_SP
   CMP BL, MOTOR_MAX_SP
   JE EXIT
   INC BL
   MOV CURRENT_SP, BL
   EXIT: ;
ENDM

INIT_DISPLAY MACRO
   MOV DX, PORT_
   IN AL, DX
   AND AL, 00001111B
   OUT DX, AL
ENDM



DELAY MACRO
   LOCAL w1

   MOV CX, BASE_DELAY_AMNT

   ;PUSH AX

   w1:
      IN AL, 61H
      AND AL, 10H
      CMP AL, AH
      JE w1
      MOV AH, AL
      LOOP w1
  ; POP AX

ENDM


SET_DELAY   MACRO DTIME
   LOCAL w, PASS1
   ;
   PUSH AX
   MOV CX, DTIME
   CMP CX, 0
   JE PASS1
   w:
      PUSH CX
      DELAY
      ;DISPLAY _DELAYIN_
      POP CX
      LOOP w

   PASS1:;
   POP AX
```

```
ENDM


RUN MACRO
   LOCAL RP, PASS1, PASS2, EXIT
   RP:     HANDLE_MOUSE_CLICK BL
      CMP BL,0
      JE PASS1
      CMP BL,'l'
      JE LFT
      CMP BL, 'r'
      JE RIT
      CMP BL, 'm'
      JE MID
      JMP PASS1
  LFT: DISPLAY LFT_
      SET_DIRECTION ROT_L
      JMP PASS1
      ;
  RIT: DISPLAY RIT_
      SET_DIRECTION ROT_R
      JMP PASS1
      ;
  MID: DISPLAY MID_
      JMP PASS1

  PASS1: ;
      ASYNC_GETCHR BL
      CMP BL, 0
      JE PASS2
      CMP BL, 'q'
      JE EXIT
      CMP BL, 'u'
      JE UP
      CMP BL, 'd'
      JE DOWN
      JMP PASS2
   UP:     DISPLAY UP_
         INC_SPEED
         JMP PASS2
   DOWN:   DISPLAY DOWN_
         DEC_SPEED
         JMP PASS2

  PASS2:

     RUN_MOTOR

  ;
                  JMP RP ; LOOP BACK
   EXIT:;

ENDM
```

```
    INCLUDE "STDIO.mac"
            .MODEL SMALL
            .STACK 64
            ;
            .DATA
            ;
PORT_       DW  0378H
;
; Strings
LFT_        DB  'CCW Rotation','$'
RIT_        DB  'CW Rotation','$'
UP_         DB  'Accelerating','$'
DOWN_         DB  'Deccelerating','$'
MID_        DB  'Middle','$'
STOP         DB  'System Terminated','$'
;
; Student Info.
STUDENTS      DB  'Mohamad & Enes','$'
COURSE       DB  'Microprocessor I','$'
;_DELAY_      DB  'Delay','$'
;_DELAYIN_     DB  'Delay_IN','$'

; Motor parameters
ROT_R             DB  01000000B
ROT_L             DB  10000000B
INIT_            DB  0
CURRENT_SP         DB  0
BASE_DELAY_AMNT DW  0FFFFH
MOTOR_DIRECTION DB  0
MOTOR_MAX_SP    DB  9
            ;
            .CODE
MAIN        PROC FAR
            MOV AX, @DATA
            MOV DS, AX
            ;
    ;
    DISPLAY_N COURSE
    DISPLAY_N STUDENTS

            ;
            ;SET_DIRECTION 0
            INIT_DIRECTION
            INIT_DISPLAY
            ;
    INIT_MOUSE
    SHOW_MOUSE
    ;
    RUN
            ;
            DISPLAY STOP
            ;
            MOV AH, 4CH
```

```
                INT 21H
MAIN            ENDP
                ;
                END MAIN
ret
```