A Report
On
CHOCOLATE VENDING MACHINE
Hardware Design and ALP

Microprocessor Programming and Interfacing

Done By
Group No. 73 Project No. 19

Amitayush Thakur – 2012B4A7819P
Jaiwant Rawat – 2012B4A7714P
Prajakta K Joshi – 2012B1A3846P
Garima Dhanania - 2012B4A3381P

Problem Statement:

P19: System to be Designed : Chocolate Vending System

Description: This automatic machine vends three different types of chocolates.

Perk: Rs. 5.00

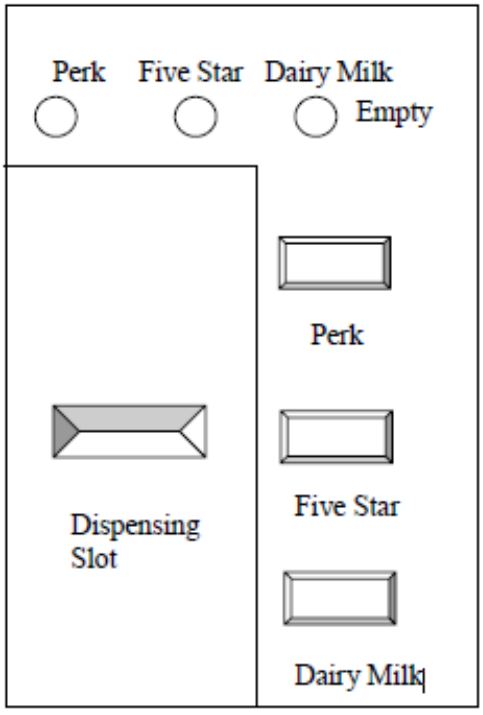
Five-Star: Rs10

Dairy Milk: Rs20.

The currency has to be given in terms of 5 Rupee coin. A weight sensor is used to detect whether the coin is an Rs5 coin or not. There are three buttons available for the selection of the chocolate. After the chocolate has been selected user has to put the correct currency into the coin slot. When the user has dropped the entire amount into the slot, the machine dispenses the correct chocolate.

LED's are used as indicators to show if any of the chocolates being vended are not available.

User Interface:



System to be designed: Chocolate Vending Machine

System Requirements:

- Coins denomination should be 5.
- System is a vending machine which give chocolates of three types i.e. Dairy Milk , Five Star and Perk.
- The prices of the chocolates are as follows:
  Dairy Milk – Rs. 15
  Five Star – Rs. 10
  Perk – Rs. 5
- User presses the button for chocolate selection and then puts in money (currency in terms of 5 Rupee coin only).

System Specifications:

- 4 LEDs are used to indicate if chocolate is available in the machine. Each LED is of 5 Volt.
- Motor is used to dispense the correct chocolate.
- Motor used is of 12V.
- Pressure Sensor (with conversion factor of 1KPa = 20 mV) is used to sense the pressure of the input coin.
- Analog To Digital Converter is used to digitize the reading taken by pressure sensor. The resolution of the ADC is 5V/256 = 19.53125 mV.
- Unipolar Stepper Motor is used to serve the purpose of the dispensing slot.

Assumptions:

- Maximum 100 Dairy Milk, Five Star and Perk are available (i.e the maximum capacity of the machine at any given time).
- If the user enters invalid number of coins then the coins are supposed to be returned automatically and the user is supposed to pick them up back.
- If there are insufficient chocolates in the machine, the machine indicates the user of it by glowing the corresponding and LED and the user is supposed to pick up the coins.
- At most user can put 14 coins at once.
- Only one type of chocolate is dispensed in one transaction. (More than one chocolate of the same typed can be dispensed in one transaction).
- The pressure of a 5 rupee coin is 1KPa which gives a 20mV voltage.

System Description:

This automatic machine vends three different types of chocolates. Perk: Rs. 5.00 , Five-Star: Rs10.00 , Dairy Milk: Rs20.00. The currency has to be given in terms of 5 Rupee coin. A weight sensor (by the means of a pressure sensor) is used to detect whether the coin is a Rs 5 coin or not. There are three buttons available for the selection of the chocolate. After the chocolate has been selected user has to put the correct currency into the coin slot. When the user has dropped the entire amount into the slot, the machine dispenses the correct chocolate and the current number depending upon the amount put in. LED's are used as indicators to show if any of the chocolates being vended are not available. If the input is invalid, the invalid LED glows to indicate this and the user needs to pick the coins back up.

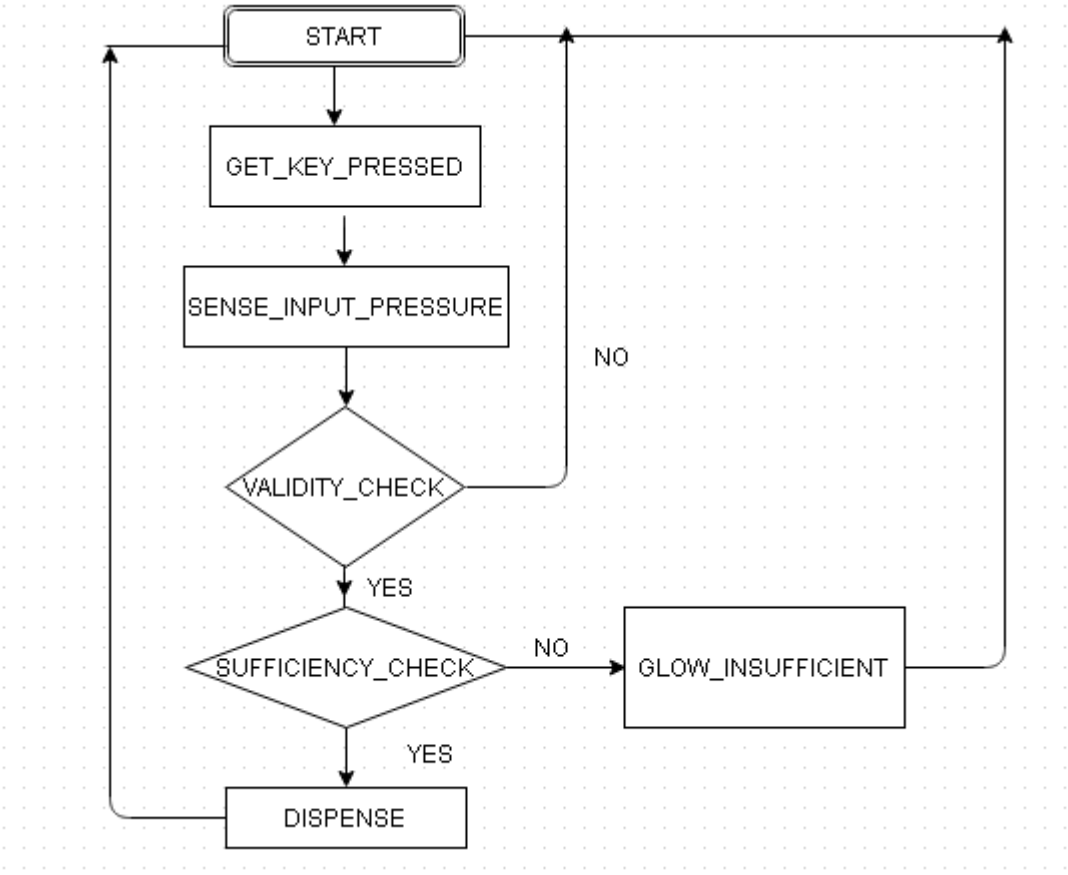
Hardware Description:

Name	Quantity
Intel 8086:Microprocessor (MPU)	1
Intel 8255A :Programmable Peripheral interface device (PPI)	1
74154 (Decoder IC)	1
74LS373(Latch)	3
74LS245 (Bidirectional Buffer)	2
NOR GATE	1
6116:2k RAM	2
2732:2k ROM	2
ADC0808(Analog to Digital Converter)	1
LED	4
Switches(SW-SPDT)	1
Buttons(SPST Push Button)	1
MPX 4250(Pressure Sensor)	1
Motor Stepper(Unipolar Stepper Motor)	1
OR GATE	1
AND Gate	1
Not Gate	1

Algorithm:

- First of all the user presses the button then GET_KEY_PRESSED procedure is called which save the KEY_PRESSED into the variable KEYPRESSED
- As soon as the key is pressed the user puts the coins which is sensed by the pressure sensor.
- Procedure SENSE_INPUT_PRESSURE is called to sense the pressure and find out number of coins placed and stores it in a variable called NO_OF_COINS. If the input is invalid the variable IS_VALID is set to 0 and NO_OF_COINS are made 0. Else the variable IS_VALID is set to 1.
- The next challenge is to find out whether the NO_OF_COINS is integral multiple of number of coins needed for the corresponding chocolate or not and whether the number of chocolates are sufficient or not. This is done using the VALIDITY_CHECK_AFTER_KEYPRESS.
- Next, the procedure SUFFICIENCY_CHECK checks if the input is valid in terms of chocolates available in the system and if it is so then the corresponding chocolates are reduced in the system.
- After that if the transaction is valid then the Stepper Motor is rotated as many times as many chocolates are to be dispensed. This is done using the procedure START_MOTOR.
- If the transaction is not valid then we glow BLUE LED using the procedure GLOW_LED.
- If the chocolates are not sufficient then the corresponding LED glows.

Flow Chart:



Legend for reading flowchart :

GET_KEY_PRESSED: Gets the key pressed by the user.

SENSE_INPUT_PRESSURE: Converts the input pressure to the number of coins.

VALIDITY_CHECK: Checks if the number of coins are integral multiple of the price for the chocolate for which the key was pressed.

SUFFICIENCY_CHECK: Checks if the selected chocolate is available in sufficient quantity or not.

GLOW_INSUFFICIENT: Glow the LED for the corresponding chocolate which is found insufficient.

DISPENSE:Uses motor to dispense the corresponding number of chocolates.

Memory addresses:

RAM: 02000H-02FFFH

ROM: 00000H-01FFFH

NAME OF THE PORT IN 8255A	ADDRESS OF THE PORT IN 8255A
PORT A	10H
PORT B	12H
PORT C	14H
CONTROL REGISTER	16H

ALP Program:

```
.Model Tiny

.data
ORG   00

; KEYPAD LOOKUP TABLE
KEYPAD_TABLE      DB    060H,050H,030H
KEYPAD_TABLE_LENGTH    EQU   3

; PORT ADDRESSES OF 8255
PORTA EQU 10H
PORTB EQU 12H
PORTC EQU 14H
CTRL_ADDR EQU 16H
IO_MODE EQU 80H

;DELAY AND KEYBOARD VARIABLES
KEYPRESSED DB    ?
DELAY20MSCOUNT   EQU   1000h

; KEY IDs
KEYID_DAIRY_MILK EQU   1
KEYID_FIVE_STAR        EQU         2
KEYID_PERK             EQU         3


; STACK
STACK1            DW    100 DUP(0)
TOP_STACK1  LABEL WORD

;PRESSURE SENSOR VARIABLES
IS_VALID db ?
NO_OF_COINS db ?
PRESSURE_OFFSET equ 13
PRESSURE_LIMIT equ 14
PRESSURE_LIMIT_PLUS_OFFSET equ 28    ;mAX
COINS=14,OFFSET=13,+1(IN-VALID INPUT FROM 15 COINS
ONWARDS)
```

```
;STATE VARIABLES
STATE_PORTA db ?
STATE_PORTB db ?
STATE_PORTC db ?
STATE_CONTROL_REGISTER db ?

;VALIDITY CONDITION VARIABLES
IS_INSUFFICIENT db 4 ; if set less than 4 then the
chocolate id is insufficient
COINS_FOR_DAIRY_MILK equ 3
COINS_FOR_FIVE_STAR equ 2
COINS_FOR_PERK equ 1
NUM_OF_CHOCS db 0
NUM_OF_DAIRY_MILK_LEFT db 100
NUM_OF_FIVE_STAR_LEFT db 100
NUM_OF_PERK_LEFT db 100

;STEPPER MOTOR ROTATION SEQUENCE VARIABLES
STEPPER_MOTOR_SEQUENCE1 EQU 00000100B              ;motor
sequence with PB2=1
STEPPER_MOTOR_SEQUENCE2 EQU 00001000B              ;motor
sequence with PB3=1
STEPPER_MOTOR_SEQUENCE3 EQU 00010000B              ;motor
sequence with PB4=1
STEPPER_MOTOR_SEQUENCE4 EQU 00100000B              ;motor
sequence with PB5=1


.code
.startup
            ;MOV AL,80H
            ;OUT CTRL_ADDR,AL
            ;MOV  AL, 06h                                          ; POR
            ;OUT  CTRL_ADDR,AL
            ;MOV AL,80H
            ;OUT CTRL_ADDR,AL
            ;MOV AL, 10000100b
            ;OUT PORTB,AL
            ;CALL GLOW_YELLOW
            ;CALL DELAY_20MS
            ;CALL GLOW_RED
            ;CALL DELAY_20MS
```

```
;CALL GLOW_BLUE
;CALL DELAY_20MS
;CALL GLOW_GREEN
MAIN1:
      ;set all ports to zero
      ;CALL RESTORE_PORTS
      CALL GLOW_NOTHING


      ;Get the key pressed in the variable
KEYPRESSED
      CALL GET_KEY_PRESSED

      CMP KEYPRESSED,KEYID_DAIRY_MILK
      JNZ X1
      CALL GLOW_RED
      JMP X3
      X1:
            CMP KEYPRESSED,KEYID_FIVE_STAR
            JNZ X2
            CALL GLOW_YELLOW
            JMP X3
      X2:
            CMP KEYPRESSED,KEYID_PERK
            JNZ X3
            CALL GLOW_GREEN
      X3:
            ;CALL DELAY_20MS
            CALL RESTORE_PORTS
            CALL GLOW_NOTHING

      ;Start sensing pressure
      CALL SENSE_INPUT_PRESSURE

      ;set all ports to zero
      ;CALL RESTORE_PORTS

      ;CALL GLOW_NOTHING
      ;CALL GLOW_GREEN
      ;CALL DELAY_20MS
      ;CALL DELAY_20MS
```

```
                    ;Check if the number of coins exceed or
not
                          CMP IS_VALID,00h
                          JZ MAIN1    ; if yes then discard
and start fresh
                          ; else go to MAIN2 where you see
the key press.
              MAIN2:




                    ;check for the validity as well as the
multiplicity
                    CALL VALIDITY_CHECK_AFTER_KEYPRESS

                    ;Check if the number of coins s integral
multiple or not
                          CMP IS_VALID,00h
                          JNZ MAIN3                    ; if
yes then discard and start fresh
MAIN1_BEFORE:       CALL GLOW_BLUE
                          CALL DELAY_20MS
                          JMP MAIN1
                          ; else go to start motor to
dispense the chocolates

              MAIN3:
                    ;if the chocolates are not sufficient
then go back
                          CMP IS_INSUFFICIENT,4
                          JGE MAIN4
                          CALL GLOW_INSUFFICIENT
                          JMP MAIN_END

              MAIN4:
                          CALL START_MOTOR

              MAIN_END:
                         JMP MAIN1
```

```
                    ;JNZ X1
                    ;CALL GLOW_RED
                    ;JMP X3
                    ;X1:
                    ;     CMP KEYPRESSED,KEYID_FIVE_STAR
                    ;     JNZ X2
                    ;     CALL GLOW_YELLOW
                    ;     JMP X3
                    ;X2:
                    ;     CMP KEYPRESSED,KEYID_PERK
                    ;     JNZ X3
                    ;     CALL GLOW_GREEN
                    ;X3:
                    ;     CALL DELAY_20MS
                    ;     CALL RESTORE_PORTS
                    ;     CALL DELAY_20MS


                    ;CALL STORE_STATE_OF_PORTS

.exit
GLOW_YELLOW PROC NEAR
            PUSHF
            PUSH  AX
            PUSH  BX
            PUSH  CX
            PUSH  DX


      ;SET PB1 TO 1 AND PB0 TO 1
            MOV AL,10011000b
            OUT CTRL_ADDR,AL
            MOV AL, 00000011b
            OUT PORTB,AL
            MOV AL,00001000b
            OUT PORTC,AL
```

```
            POP   DX
            POP   CX
            POP   BX
            POP   AX
            POPF

            RET

GLOW_YELLOW ENDP

GLOW_RED PROC NEAR
            PUSHF
            PUSH  AX
            PUSH  BX
            PUSH  CX
            PUSH  DX


      ;SET PB1 TO 1 AND PB0 TO 0
            MOV AL,10011000b
            OUT CTRL_ADDR,AL
            MOV AL, 00000010b
            OUT PORTB,AL
            MOV AL,00001000b
            OUT PORTC,AL

            POP   DX
            POP   CX
            POP   BX
            POP   AX
            POPF

            RET

GLOW_RED ENDP

GLOW_BLUE PROC NEAR
            PUSHF
            PUSH  AX
            PUSH  BX
            PUSH  CX
            PUSH  DX
```

```
        ;SET PB1 TO 0 AND PB0 TO 0
              MOV AL,10011000b
              OUT CTRL_ADDR,AL
              MOV AL, 00000000b
              OUT PORTB,AL
              MOV AL,00001000b
              OUT PORTC,AL

              POP   DX
              POP   CX
              POP   BX
              POP   AX
              POPF

              RET

GLOW_BLUE ENDP

GLOW_GREEN PROC NEAR
              PUSHF
              PUSH  AX
              PUSH  BX
              PUSH  CX
              PUSH  DX

        ;SET PB1 TO 0 AND PB0 TO 1
              MOV AL,10011000b
              OUT CTRL_ADDR,AL
              MOV AL, 00000001b
              OUT PORTB,AL
              MOV AL,00001000b
              OUT PORTC,AL

              POP   DX
              POP   CX
              POP   BX
              POP   AX
              POPF

              RET
```

```
GLOW_GREEN ENDP

DELAY_20MS  PROC  NEAR
            PUSHF
            PUSH  AX
            PUSH  BX
            PUSH  CX
            PUSH  DX

            MOV   CX, DELAY20MSCOUNT
X_DELAYLOOP:      NOP
            NOP
            NOP
            NOP
            NOP
            LOOP  X_DELAYLOOP

            POP   DX
            POP   CX
            POP   BX
            POP   AX
            POPF

            RET
DELAY_20MS  ENDP

GET_KEY_PRESSED PROC NEAR
            PUSHF
            PUSH  AX
            PUSH  BX
            PUSH  CX
            PUSH  DX

            ;Setting 8255 PC lower(0-3) is input and PC
upper(4-7) is output



            MOV AL,88h
            OUT CTRL_ADDR,AL
```

```
            ;check for key release
            GET_KEY_PRESSED1:

                                    MOV AL,88h
                                    OUT CTRL_ADDR,AL
                                    MOV AL,01110000b
;Here we can't set the inner ports they all will be one
regardless of whatever we move.
                                    OUT PORTC,AL
                                    ;;CALL DELAY_20MS
            GET_KEY_PRESSED2:

                                    MOV AL,88h
                                    OUT CTRL_ADDR,AL
                                     IN AL,PORTC
                                     AND AL,0F0h
                                     CMP AL,070h
                                     JNZ GET_KEY_PRESSED2
            ;;CALL DELAY_20MS

            ;CHECK FOR KEY PRESS
            MOV AL,88h
            OUT CTRL_ADDR,AL
            MOV AL,70h
            OUT PORTC,AL
            ;;CALL DELAY_20MS
            GET_KEY_PRESSED3:

                  MOV AL,88h
                  OUT CTRL_ADDR,AL
                  IN AL,PORTC
                  AND AL,0F0h
                  CMP AL,070h
                  JZ GET_KEY_PRESSED3
            ;;CALL DELAY_20MS

            MOV AL,88h
            OUT CTRL_ADDR,AL
            MOV AL,70h
            OUT PORTC,AL
            ;;CALL DELAY_20MS
            GET_KEY_PRESSED4:
```

```
                MOV AL,88h
                OUT CTRL_ADDR,AL
                IN AL,PORTC
                AND AL,0F0h
                CMP AL,070h
                JZ GET_KEY_PRESSED4

        MOV AL,88h
        OUT CTRL_ADDR,AL
        IN AL,PORTC
        AND AL,0F0H

        GET_KEY_PRESSED5:
                CMP AL,KEYPAD_TABLE[0]
                JNZ GET_KEY_PRESSED6
                MOV KEYPRESSED,KEYID_DAIRY_MILK
                JMP GET_KEY_PRESSED_END
        GET_KEY_PRESSED6:
                CMP AL,KEYPAD_TABLE[1]
                JNZ GET_KEY_PRESSED7
                MOV KEYPRESSED,KEYID_FIVE_STAR
                JMP GET_KEY_PRESSED_END
        GET_KEY_PRESSED7:
                CMP AL,KEYPAD_TABLE[2]
                JNZ GET_KEY_PRESSED_END
                MOV KEYPRESSED,KEYID_PERK
                JMP GET_KEY_PRESSED_END

        GET_KEY_PRESSED_END:
        CALL RESTORE_PORTS
        POP   DX
        POP   CX
        POP   BX
        POP   AX
        POPF

        RET
GET_KEY_PRESSED ENDP

RESTORE_PORTS PROC NEAR
```

```
            PUSHF
            PUSH  AX
            PUSH  BX
            PUSH  CX
            PUSH  DX

            MOV AL,80h
            OUT CTRL_ADDR,AL

            MOV AL,00h
            OUT PORTA,AL


            MOV AL,80h
            OUT CTRL_ADDR,AL
            MOV AL,00000000b
            OUT PORTB,AL


            MOV AL,80h
            OUT CTRL_ADDR,AL
            MOV AL,00000000b
            OUT PORTC,AL

            ;CALL DELAY_20MS
            POP   DX
            POP   CX
            POP   BX
            POP   AX
            POPF

            RET
RESTORE_PORTS ENDP

;call this procedure before any procedure or inside any
procedure that has a possibility of changing the state of
ports.
STORE_STATE_OF_PORTS PROC NEAR
            PUSHF
            PUSH  AX
            PUSH  BX
            PUSH  CX
```

```
            PUSH  DX

            IN AL,CTRL_ADDR
            MOV STATE_CONTROL_REGISTER,AL
            IN AL,PORTA
            MOV STATE_PORTA,AL
            IN AL,PORTB
            MOV STATE_PORTB,AL
            IN AL,PORTC
            MOV STATE_PORTC,AL

            POP   DX
            POP   CX
            POP   BX
            POP   AX
            POPF

            RET
STORE_STATE_OF_PORTS ENDP

;call this procedure after any procedure that has a
possibility of changing the state of ports.
REVERT_STATE_OF_PORTS PROC NEAR
            PUSHF
            PUSH  AX
            PUSH  BX
            PUSH  CX
            PUSH  DX

            MOV AL,STATE_CONTROL_REGISTER
            OUT CTRL_ADDR,AL
            MOV AL,STATE_PORTA
            OUT PORTA,AL
            MOV AL,STATE_PORTB
            OUT PORTB,AL
            MOV AL,STATE_PORTC
            OUT PORTC,AL

            POP   DX
            POP   CX
            POP   BX
            POP   AX
```

```
            POPF

            RET
REVERT_STATE_OF_PORTS ENDP

SENSE_INPUT_PRESSURE PROC NEAR
                  PUSHF
                  PUSH  AX
                  PUSH  BX
                  PUSH  CX
                  PUSH  DX



                                    ;send start of
conversion signal to ADC along with address of analog
input channel to activate
                                                MOV
AL,10011000B          ;setting PORTC upper(4-7) as input
and PORTC lower(0-3) as output,PORTA as input,PORTB as
output
                                                OUT
CTRL_ADDR,AL
                                                ;Making it
low to high

                                                MOV
AL,00000000B          ;setting PC1(SOC) to 0,PC0 to 0
                                                OUT PORTC,AL
                                                CALL
DELAY_20MS

                                                MOV
AL,10011000B          ;setting PORTC upper(4-7) as input
and PORTC lower(0-3) as output,PORTA as input,PORTB as
output
                                                OUT
CTRL_ADDR,AL
                                                MOV
AL,00000010B          ;setting PC1(SOC) to 1,PC0 to 0
                                                OUT PORTC,AL
```

```
                                        CALL
DELAY_20MS

                                        MOV
AL,10011000B            ;setting PORTC upper(4-7) as input
and PORTC lower(0-3) as output,PORTA as input,PORTB as
output
                                        OUT
CTRL_ADDR,AL
                                        MOV AL,00H
                                        OUT PORTB,AL
                                ;giving conversion time
to ADC
                                ;right now giving a
longer delay(20ms) rather than only conversion time of
ADC(100us)
                                ;check for end of
conversion signal from ADC


EOC_CHECK:
                                        MOV
AL,10011000B            ;setting PORTC upper(4-7) as input
and PORTC lower(0-3) as output,PORTA as input,PORTB as
output
                                        OUT
CTRL_ADDR,AL
                                        IN AL,PORTC
                                        MOV BL,AL
                                        AND
BL,10000000b
                                        CMP BL,00H
                                        JZ EOC_DONE
                                        JMP
EOC_CHECK

                                ;Conversion complete
move to taking input from ADC
EOC_DONE:                               MOV AL,10011000B
                                        OUT
CTRL_ADDR,AL
                                        IN AL,PORTA
```

```
                                    ;To check validity of
input by examining D0-D7 sequence
                                    MOV AL,10011000B            ;sett
                                    OUT CTRL_ADDR,AL
                                    IN AL,PORTA                 ;taki


                                    CMP
AL,PRESSURE_LIMIT_PLUS_OFFSET
                                    JGE
PRESSURE_LIMIT_EXCEED
                                    CMP AL,00H
                                    JE
PRESSURE_LIMIT_FALL_SHORT
                                    MOV IS_VALID,01H
                                    MOV BL,AL
                                    SUB BL,PRESSURE_OFFSET
                                    MOV NO_OF_COINS,BL
                                    JMP PRESSURE_FINISH
PRESSURE_LIMIT_EXCEED: MOV IS_VALID,00H
                                    MOV NO_OF_COINS,00H
                                    JMP PRESSURE_FINISH
PRESSURE_LIMIT_FALL_SHORT: MOV IS_VALID,00H
                                    MOV NO_OF_COINS,00H
PRESSURE_FINISH:

;NO_COIN_INPUT_CHECK:  MOV BL,AL
;                                   CMP BL,00H
;                                   JNZ D7_CHECK
;                                   MOV IS_VALID,00h
;                                   JMP FINISH
;D7_CHECK:                    MOV BL,AL
;                                   AND BL,10000000b
;                                   CMP BL,01H
;                                   JNZ D6_CHECK
;                                   MOV IS_VALID,00h
;                                   JMP FINISH
;D6_CHECK:                    MOV BL,AL
;                                   AND BL,01000000b
;                                   CMP BL,01H
;                                   JNZ D5_CHECK
```

```
;                                       MOV IS_VALID,00h
;                                       JMP FINISH
;D5_CHECK:                        MOV BL,AL
;                                       AND BL,00100000b
;                                       CMP BL,01H
;                                       JNZ SET_FINAL_IS_VALID
;                                       MOV BL,AL
;                                       AND BL,00010000b
;                                       CMP BL,01H
;                                       JNZ D3_CHECK
;                                       MOV IS_VALID,00H
;                                       JMP FINISH
;D3_CHECK:                        MOV BL,AL
;                                       AND BL,00001000b
;                                       CMP BL,01H
;                                       JNZ SET_FINAL_IS_VALID
;                                       MOV BL,AL
;                                       AND BL,00000111b
;                                       CMP BL,00H
;                                       JZ SET_FINAL_IS_VALID
;                                       MOV IS_VALID,00H
;                                       JMP FINISH
;SET_FINAL_IS_VALID:              MOV IS_VALID,01H
;FINISH                    :

                                        ;if input is valid->
set number of coins based on range of D0-D7

            POP   DX
            POP   CX
            POP   BX
            POP   AX
            POPF

            RET
SENSE_INPUT_PRESSURE ENDP

VALIDITY_CHECK_AFTER_KEYPRESS PROC NEAR
            PUSHF
            PUSH  AX
            PUSH  BX
            PUSH  CX
```

```
            PUSH  DX

            ; get number of coins for the key pressed
            DAIRY_MILK_PRESSED:
                  CMP KEYPRESSED,KEYID_DAIRY_MILK
                  JNZ FIVE_STAR_PRESSED
                  MOV CL,COINS_FOR_DAIRY_MILK
                  MOV AL,NO_OF_COINS
                  MOV CH,00h
                MOV AL,NO_OF_COINS
                MOV AH,00h

                 ;check if the number of coins is the
integral multiple of the key pressed
                  DIV CL              ; al = ax/cl, ah =
remainder

            ;compare the remainder
                CMP AH,00h
                  JZ SET_IT_VALID1
                  MOV IS_VALID,00h
                  JMP VALIDITY_END

                  SET_IT_VALID1:
                  MOV IS_VALID,01h
                  MOV NUM_OF_CHOCS,AL
                  SUB NUM_OF_DAIRY_MILK_LEFT,AL
                  CMP NUM_OF_DAIRY_MILK_LEFT,00h
                  JGE   VALIDITY_END
                  MOV IS_INSUFFICIENT,KEYID_DAIRY_MILK
                  ADD NUM_OF_DAIRY_MILK_LEFT,AL
                  JMP VALIDITY_END


            FIVE_STAR_PRESSED:
                  CMP KEYPRESSED,KEYID_FIVE_STAR
                  JNZ PERK_PRESSED
                  MOV CL,COINS_FOR_FIVE_STAR
                  MOV AL,NO_OF_COINS
                  MOV AL,NO_OF_COINS
                  MOV CH,00h
                MOV AL,NO_OF_COINS
```

```
                MOV AH,00h

                 ;check if the number of coins is the
integral multiple of the key pressed
                DIV CL                  ; al = ax/cl, ah =
remainder

            ;compare the remainder
                CMP AH,00h
                  JZ SET_IT_VALID2
                  MOV IS_VALID,00h
                  JMP VALIDITY_END

                  SET_IT_VALID2:
                  MOV IS_VALID,01h
                  MOV NUM_OF_CHOCS,AL
                  SUB NUM_OF_FIVE_STAR_LEFT,AL
                  CMP NUM_OF_FIVE_STAR_LEFT,00h
                  JGE   VALIDITY_END
                  MOV IS_INSUFFICIENT,KEYID_FIVE_STAR
                  ADD NUM_OF_FIVE_STAR_LEFT,AL
                  JMP VALIDITY_END

            PERK_PRESSED:
                  MOV IS_VALID,00h
                  CMP KEYPRESSED,KEYID_PERK
                  JNZ VALIDITY_END
                  MOV CL,COINS_FOR_PERK
                  MOV AL,NO_OF_COINS
                  MOV AL,NO_OF_COINS
                  MOV CH,00h
                MOV AL,NO_OF_COINS
                MOV AH,00h

                 ;check if the number of coins is the
integral multiple of the key pressed
                DIV CL                  ; al = ax/cl, ah =
remainder

            ;compare the remainder
                CMP AH,00h
                  JZ SET_IT_VALID3
```

```
                MOV IS_VALID,00h
                JMP VALIDITY_END
                SET_IT_VALID3:
                MOV IS_VALID,01h
                MOV NUM_OF_CHOCS,AL
                SUB NUM_OF_PERK_LEFT,AL
                CMP NUM_OF_PERK_LEFT,00h
                JGE   VALIDITY_END
                MOV IS_INSUFFICIENT,KEYID_PERK
                ADD NUM_OF_PERK_LEFT,AL
                JMP VALIDITY_END
            AFTER_CHOCOLATE_SELECTED:
                MOV IS_VALID,01h

            MOV CH,00h
            MOV AL,NO_OF_COINS
            MOV AH,00h

            ;check if the number of coins is the integral
multiple of the key pressed
            DIV CL              ; al = ax/cl, ah =
remainder

            ;compare the remainder
            CMP AH,00h
            JZ SET_IT_VALID
            MOV IS_VALID,00h

            SET_IT_VALID:
            MOV IS_VALID,01h
            MOV NUM_OF_CHOCS,AL
            VALIDITY_END:

            POP   DX
            POP   CX
            POP   BX
            POP   AX
            POPF

            RET

VALIDITY_CHECK_AFTER_KEYPRESS ENDP
```

```
START_MOTOR PROC NEAR

            PUSHF
            PUSH  AX
            PUSH  BX
            PUSH  CX
            PUSH  DX

            ; now dummy glow led the number of time the
chocolates are ordered
            MOV CL,NUM_OF_CHOCS
            MOV CH,00h

            START_MOTOR1:
                  CALL STEPPER_MOTOR_OPEN
                  CALL DELAY_20MS
                  CALL STEPPER_MOTOR_CLOSE
                  LOOP START_MOTOR1

MOTOREND:


            POP   DX
            POP   CX
            POP   BX
            POP   AX
            POPF

            RET
START_MOTOR ENDP

GLOW_NOTHING PROC NEAR

            PUSHF
            PUSH  AX
            PUSH  BX
            PUSH  CX
            PUSH  DX

            MOV AL,80h
            OUT CTRL_ADDR,AL
```

```
            MOV AL,00000000b
            OUT PORTC,AL

            POP   DX
            POP   CX
            POP   BX
            POP   AX
            POPF

            RET

GLOW_NOTHING ENDP

GLOW_INSUFFICIENT PROC NEAR

            PUSHF
            PUSH  AX
            PUSH  BX
            PUSH  CX
            PUSH  DX

            CALL GLOW_NOTHING

            CMP IS_INSUFFICIENT,KEYID_DAIRY_MILK
            JNZ GLOW_INSUFFICIENT1
            CALL GLOW_RED
            JMP GLOW_INSUFFICIENT_END

            GLOW_INSUFFICIENT1:
            CMP IS_INSUFFICIENT,KEYID_FIVE_STAR
            JNZ GLOW_INSUFFICIENT2
            CALL GLOW_YELLOW
            JMP GLOW_INSUFFICIENT_END

            GLOW_INSUFFICIENT2:
            CMP IS_INSUFFICIENT,KEYID_PERK
            JNZ GLOW_INSUFFICIENT_END
            CALL GLOW_GREEN
            JMP GLOW_INSUFFICIENT_END

            GLOW_INSUFFICIENT_END:
            CALL DELAY_20MS
```

```
            CALL DELAY_20MS
            CALL GLOW_NOTHING

            POP   DX
            POP   CX
            POP   BX
            POP   AX
            POPF

            RET
GLOW_INSUFFICIENT ENDP

STEPPER_MOTOR_OPEN PROC NEAR
;give the sequence to stepper motor such that at a time
one input is 1,others are 0.
;clockwise rotation is taken as opening of motor slot.

            PUSHF
            PUSH  AX
            PUSH  BX
            PUSH  CX
            PUSH  DX

            MOV AL,10011000B            ;setting PORTC
upper(4-7) as input and PORTC lower(0-3) as output,PORTA
as input,PORTB as output
            OUT CTRL_ADDR,AL
            ;to disable the decoder putting PC3=0
            IN AL,PORTC
            MOV DL,AL
            MOV BL,DL
            AND BL,11110111B
            MOV AL,BL
            OUT PORTC,AL
            MOV AL,10011000B            ;setting PORTC
upper(4-7) as input and PORTC lower(0-3) as output,PORTA
as input,PORTB as output
            OUT CTRL_ADDR,AL
            MOV AL,STEPPER_MOTOR_SEQUENCE1
            OUT PORTB,AL
            CALL DELAY_20MS
            COMMENT @
```

```
            MOV AL,10011000B            ;setting PORTC
upper(4-7) as input and PORTC lower(0-3) as output,PORTA
as input,PORTB as output
            OUT CTRL_ADDR,AL
            MOV AL,STEPPER_MOTOR_SEQUENCE2
            OUT PORTB,AL
            CALL DELAY_20MS
            MOV AL,10011000B            ;setting PORTC
upper(4-7) as input and PORTC lower(0-3) as output,PORTA
as input,PORTB as output
            OUT CTRL_ADDR,AL
            MOV AL,STEPPER_MOTOR_SEQUENCE3
            OUT PORTB,AL
            CALL DELAY_20MS
            MOV AL,10011000B            ;setting PORTC
upper(4-7) as input and PORTC lower(0-3) as output,PORTA
as input,PORTB as output
            OUT CTRL_ADDR,AL
            MOV AL,STEPPER_MOTOR_SEQUENCE4
            OUT PORTB,AL
            CALL DELAY_20MS
            @
            ;setting everything to 0 again,may have to be
removed if not required.
            ;MOV AL,10011000B            ;setting PORTC
upper(4-7) as input and PORTC lower(0-3) as output,PORTA
as input,PORTB as output
            ;OUT CTRL_ADDR,AL
            ;MOV AL,00H
            ;OUT PORTB,AL
            ;CALL DELAY_20MS

            ;restore state of PORTC
            MOV AL,10011000B            ;setting PORTC
upper(4-7) as input and PORTC lower(0-3) as output,PORTA
as input,PORTB as output
            OUT CTRL_ADDR,AL
            MOV AL,DL
            OUT PORTC,AL

            POP   DX
            POP   CX
```

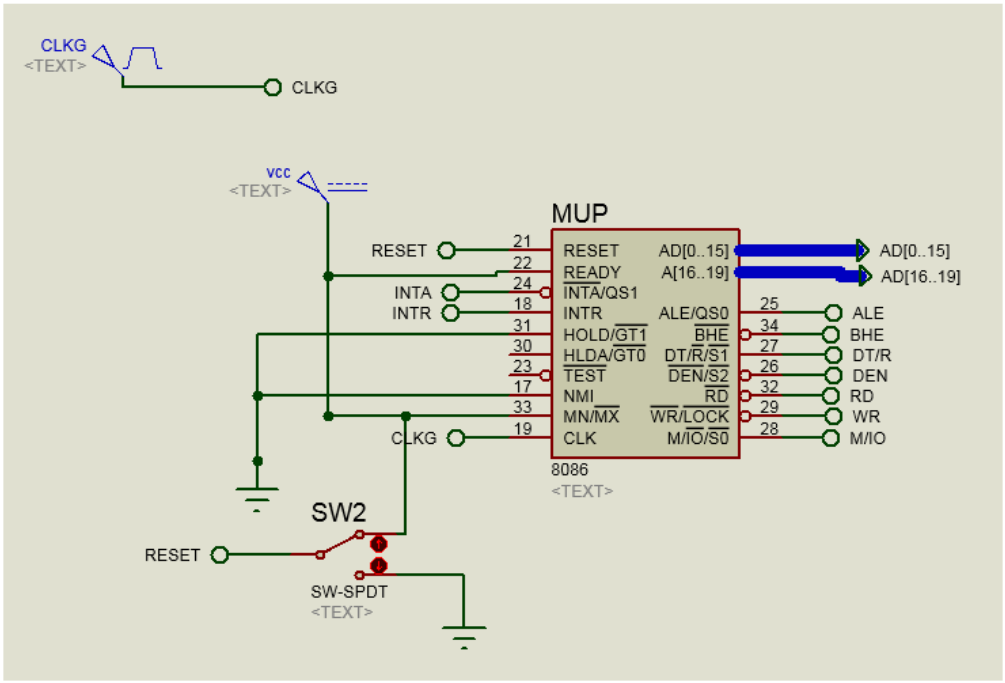
```
            POP   BX
            POP   AX
            POPF

            RET

STEPPER_MOTOR_OPEN ENDP

STEPPER_MOTOR_CLOSE PROC NEAR
;give the sequence to stepper motor such that at a time
one input is 1,others are 0.
;anti-clockwise rotation is taken as closing of motor
slot.

            PUSHF
            PUSH  AX
            PUSH  BX
            PUSH  CX
            PUSH  DX

            MOV AL,10011000B            ;setting PORTC
upper(4-7) as input and PORTC lower(0-3) as output,PORTA
as input,PORTB as output
            OUT CTRL_ADDR,AL
            ;to disable the decoder putting PC3=0
            IN AL,PORTC
            MOV DL,AL
            MOV BL,DL
            AND BL,11110111B
            MOV AL,BL
            OUT PORTC,AL
            MOV AL,10011000B            ;setting PORTC
upper(4-7) as input and PORTC lower(0-3) as output,PORTA
as input,PORTB as output
            OUT CTRL_ADDR,AL
            MOV AL,STEPPER_MOTOR_SEQUENCE3
            OUT PORTB,AL
            CALL DELAY_20MS
            COMMENT @
            MOV AL,10011000B            ;setting PORTC
upper(4-7) as input and PORTC lower(0-3) as output,PORTA
as input,PORTB as output
```

```
            OUT CTRL_ADDR,AL
            MOV AL,STEPPER_MOTOR_SEQUENCE1
            OUT PORTB,AL
            CALL DELAY_20MS
            MOV AL,10011000B            ;setting PORTC
upper(4-7) as input and PORTC lower(0-3) as output,PORTA
as input,PORTB as output
            OUT CTRL_ADDR,AL
            MOV AL,STEPPER_MOTOR_SEQUENCE4
            OUT PORTB,AL
            CALL DELAY_20MS
            MOV AL,10011000B            ;setting PORTC
upper(4-7) as input and PORTC lower(0-3) as output,PORTA
as input,PORTB as output
            OUT CTRL_ADDR,AL
            MOV AL,STEPPER_MOTOR_SEQUENCE3
            OUT PORTB,AL
            CALL DELAY_20MS
            @
            ;setting everything to 0 again,may have to be
removed if not required.
            ;MOV AL,10011000B            ;setting PORTC
upper(4-7) as input and PORTC lower(0-3) as output,PORTA
as input,PORTB as output
            ;OUT CTRL_ADDR,AL
            ;MOV AL,00H
            ;OUT PORTB,AL
            ;CALL DELAY_20MS

            ;restore state of PORTC
            MOV AL,10011000B            ;setting PORTC
upper(4-7) as input and PORTC lower(0-3) as output,PORTA
as input,PORTB as output
            OUT CTRL_ADDR,AL
            MOV AL,DL
            OUT PORTC,AL

            POP   DX
            POP   CX
            POP   BX
            POP   AX
            POPF
```

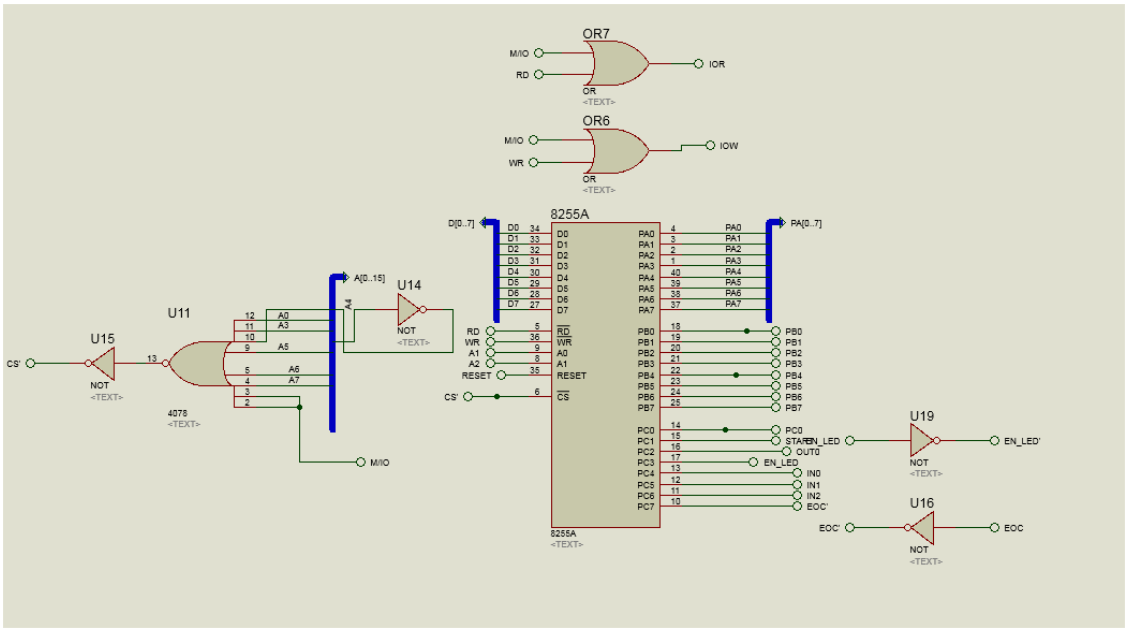
```
            RET

STEPPER_MOTOR_CLOSE ENDP

END
```

HARDWARE DESIGN

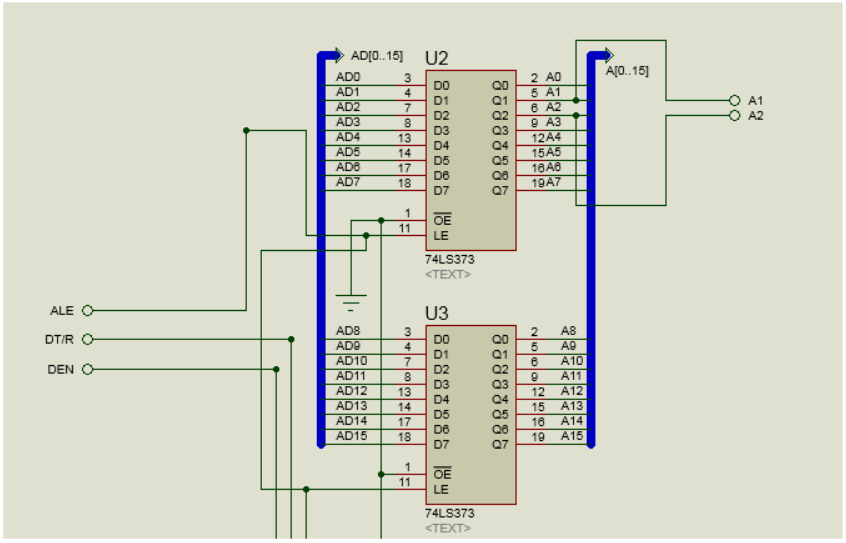
MICROPROCESSOR 8086:

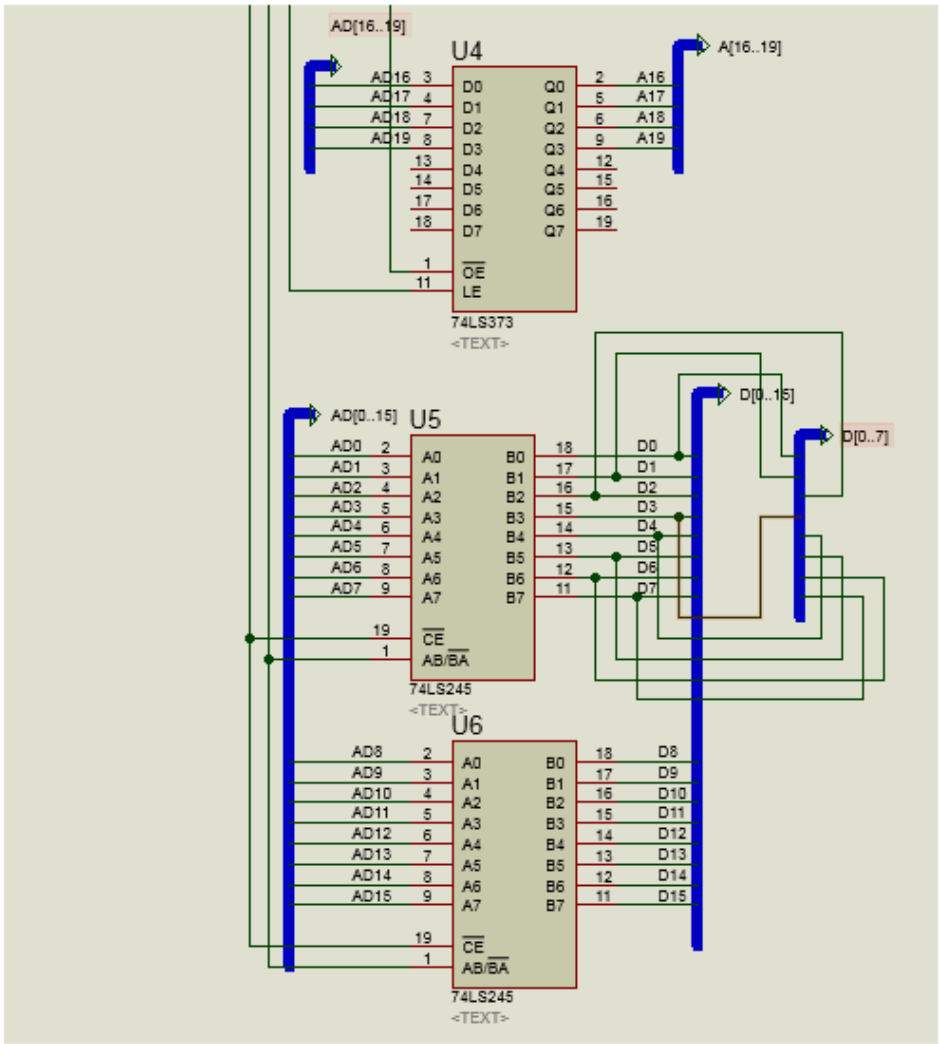


8255 INTERFACING CIRCUIT:

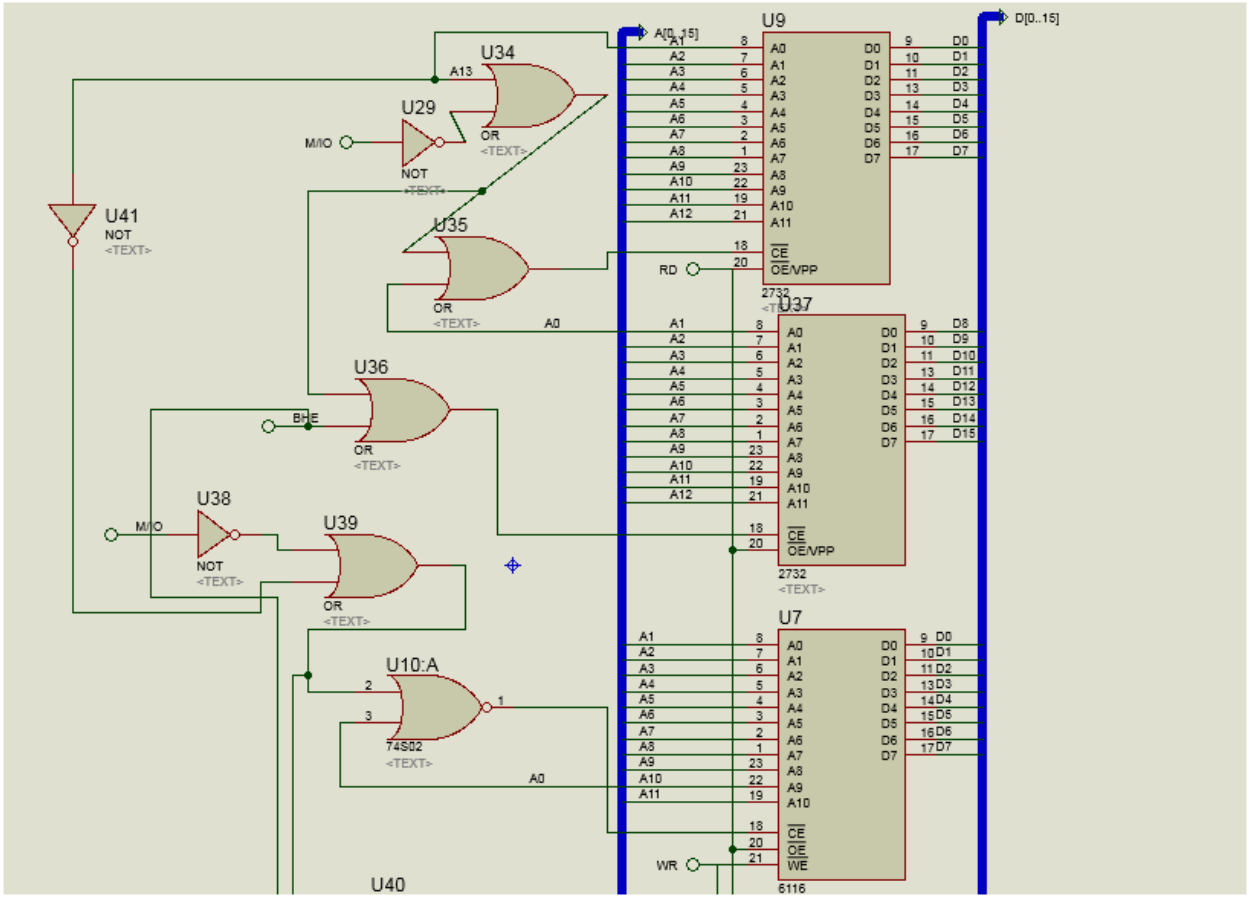


ADDRESS BUS AND DATA BUS CONNECTIONS:

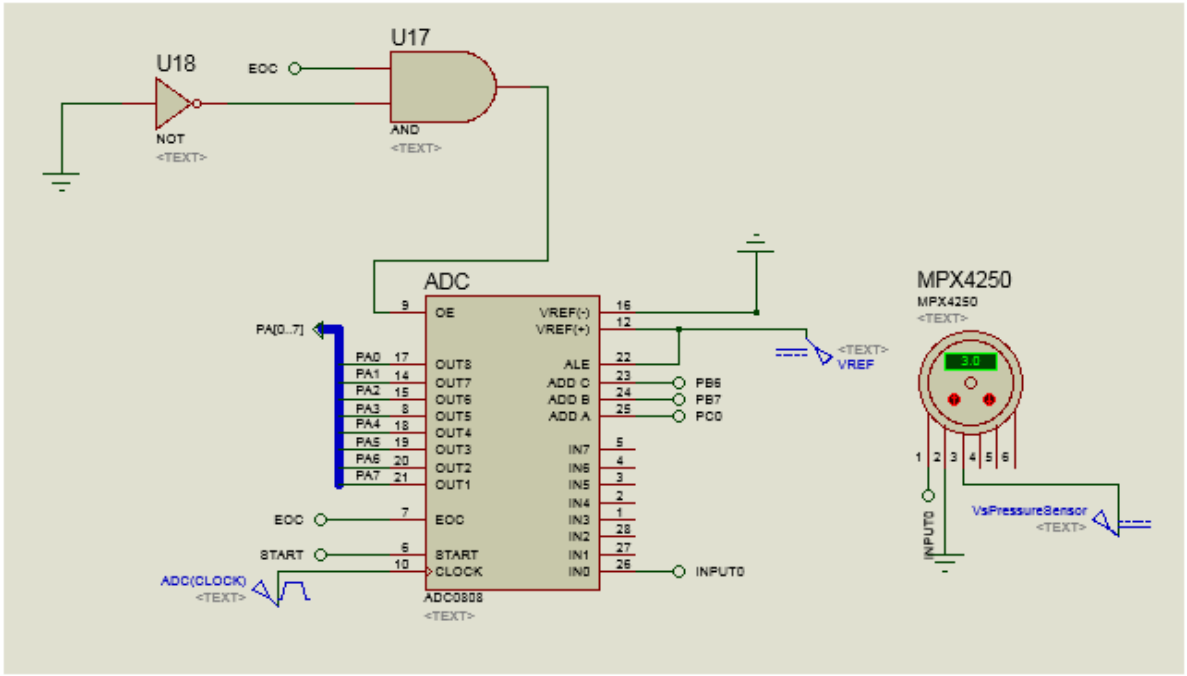




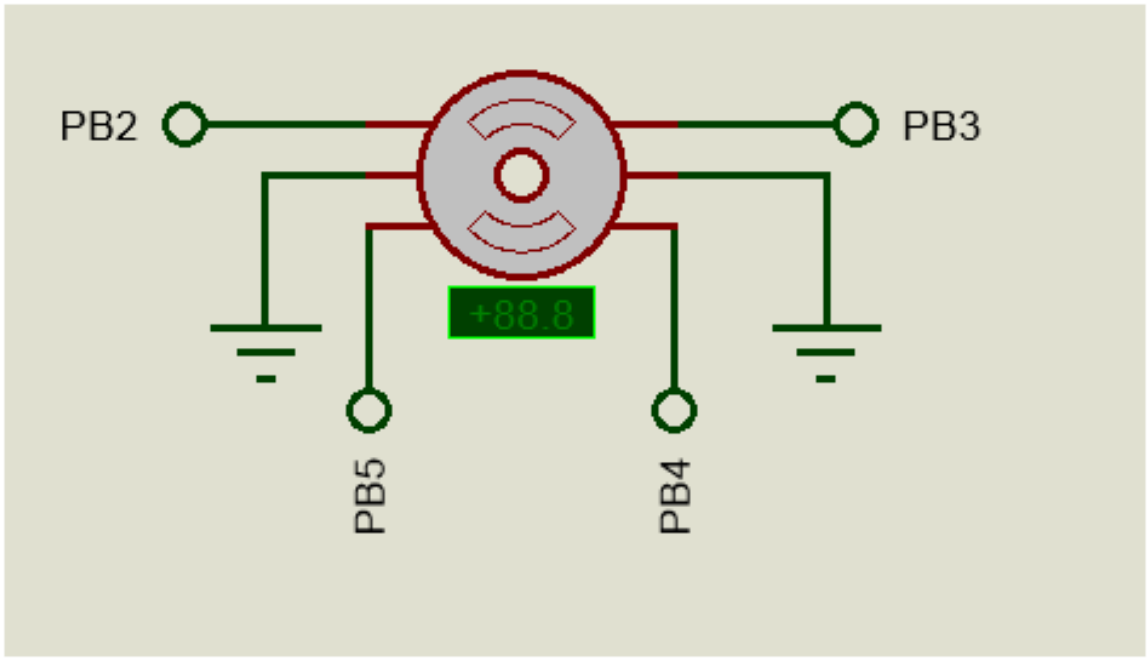
MEMORY INTERFACING:



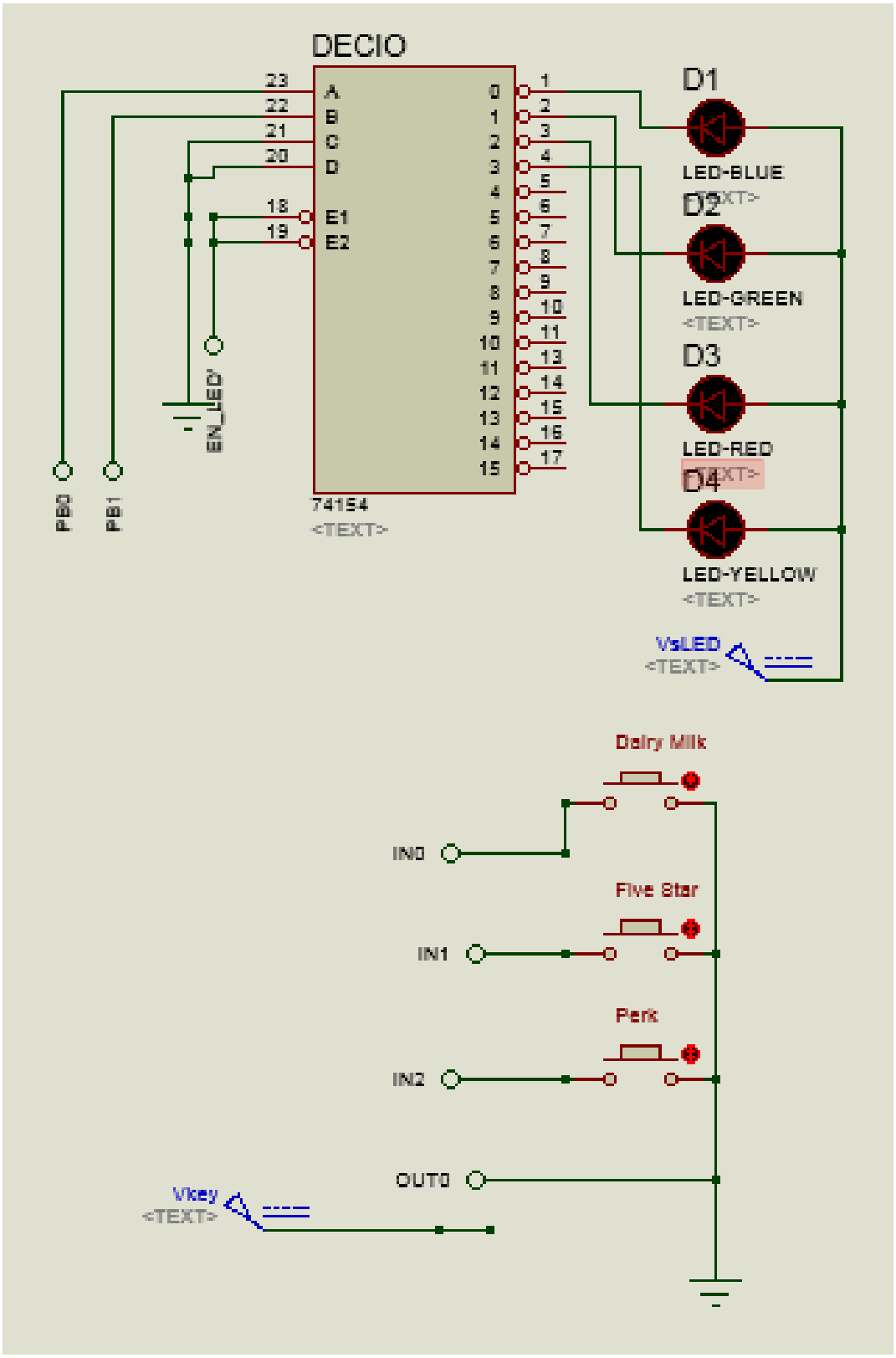
ADC AND PRESSURE SENSOR:



STEPPER MOTOR:



LEDS AND BUTTONS:



REFERENCES:

1. Datasheet of MPX4250 (Pressure Sensor):

   http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MPX4250&tab=Documentation_Tab&pspll=1&SelectedAsset=Documentation&ProdMetaId=PID/DC/MPX4250&fromPSP=true&assetLockedForNavigation=true&componentId=2&leftNavCode=1&pageSize=25&Documentation=Documentation/00610Ksd1nd``Data%20Sheets&fpsp=1&linkline=Data%20Sheets

2. Stepper Motor reference:

   http://www.theengineeringprojects.com/2013/06/stepper-motor-drive-circuit-in-proteus-isis.html