CHOCOLATE VENDING MACHINE

Design Problem number-18
Submitted by Group-4

Group Members

1. Ashutosh Gupta 2019AAPS0223G
2. Pranav Goyal 2019A8PS0548G
3. Sanskar Jain 2019AAPS0333G
4. Siddharth Barnwal 2019A7PS0114G
5. Sparsh Kachhadiya 2019A8PS0491G
6. Tanmay Bhonsale 2018B5A70903G

Contents

User Requirements & Technical Specifications ---------- 3
Assumptions and Justification ---------- 3
Hardware Description ---------- 4
Address Map ---------- 8
Design (Also attached in Design.pdf) ---------- 9
Flowcharts ---------- 21
Variations in Proteus Implementation with justification ---------- 25
Firmware (Also attached in chocolate_vending.asm) ---------- 26
List of Attachments ---------- 37
References ---------- 37

User Requirements & Technical Specifications

Design a chocolate vending machine with following chocolates:

1. Perk - 5rs
2. Five Star - 10rs
3. Dairy Milk - 15rs

Technical Specifications:

1. Money is provided only in terms of 5rs coins
2. LEDS should be used to indicate whether a chocolate is available or not
3. Weight sensor must be used to check if the money provided matches with the price of chocolate

Assumptions and Justification

1. Maximum 100 chocolates of each type are available
2. In each transaction, the user can get only one chocolate of a particular type (i.e. dairy milk, perk, five star).
3. All the 5rs Coin weigh the same, i.e 6g.
4. Users can only press one button at a time.
5. Only one chocolate can be dispensed at a time.
6. Coins of wrong denomination/fake coins/extra coins are not returned back.
7. Full 360 degree rotation of the stepper motor will dispense only one chocolate.
8. When a certain transaction is being processed, the user cannot press another button.
9. There are three gates for each of the chocolates which meet into the same final dispensing slot.
10. All LEDS require 5V supply voltage.

Coin Accepting Mechanism

1. **Opening/Closing the coin slot**: A stepper motor is attached to a mechanism that covers or uncovers the coin slot when it rotates clockwise or anticlockwise, respectively.
2. **Measuring the coin weight**:
    a. Rs.5 coin is assumed to weigh 6g, and other coins are assumed to differ by at least 1g.
    b. The MPX5500DP pressure sensor is not sensitive enough to directly measure this weight, so a level+gear train mechanism is used to ‘amplify’ the coins weight by a factor of 884.7. This can be easily accomplished by using 3 gears of ratio 9.6 in sequence.
    c. According to the datasheet, the sensor output is given by $Vout = 5 * (0.0018 * P + 0.04)$. The pressure on the sensor is given by the equation:

$$P = CoinWeight \times GearRatio \times SensorSurfaceArea$$

d. Thus the pressure value for a Rs.5 coin is 31kPa, and the corresponding voltage is 0.425V. This translates to an ADC value of 00011001 in binary (25 in decimal)
e. Vref+ of ADC 0808 is connected to 5V and Vref- is connected to ground, therefore the resolution of ADC is $(5 - 0/256) = 0.01953$.

3. **Coin Evacuating Mechanism:** A solenoid pushed the coin out of the weight sensor and into the coin box.

Hardware Description

- 8284
- 8086

- Pressure Sensor: MPX5500 (Datasheet attached)
  - **Function:** This sensor will detect the weight of the number of 5 rupee coins inserted.
  - Pressure range 0-500 Kpa
  - Will be used with a force amplifying mechanism.
  - Voltage i/p 4.75V to 5.2V DC regulated with a typical voltage of 5V DC,10 mA
  - Six pin connector- Vout, Vcc, GND, V1, V2, VEX

Pin	Signal	Input/Output	Description
1	VOUT	Output	Voltage signal corresponding to the pressure applied on the sensor(0V dc min to 5Vdc max
2	GND	Input	Ground
3	VCC	Input	Voltage input
4	V1	NC	Internal pin
5	V2	NC	Internal pin
6	VEX	NC	Internal pin

- ADC0808
  - **Function:** Measure output of pressure sensor
  - **Configuration:** Only first input (IN0) is used, so all selector inputs are set LOW.

- 8255 (x2)
  - **Function:** Interface ADC0808, LEDs and stepper motors
  - **Configuration:**
    - Mode 0 and I/O mode (not BSR) is used everywhere
    - Port I/O settings:

Chip	Port A	Port B	Port C Lower	Port C Upper
1st 8255	Input	NC	Output	NC
2nd 8255	Output	Output	Output	NC

- 8254
  - **Function:** Used to generate 833kHz ADC Clock and 12.7Hz timer interrupt for measuring weight via ADC values.
  - **Configuration:**

Counter	Mode	Input Frequency	Count	Output Frequency	Control Word
counter0	mode 3 (square wave)	5MHz	06h	833kHz	00110110b
counter1	mode 2 (rate gen)	833kHz	FFFFh (65535d)	12.7Hz	01110100b

- 8259
  - **Function:**
    - To read buttons for selecting chocolate
    - Create timed interrupts for measuring ADC values
    - Execute adc_isr for getting ADC value when the end of conversion is reached.
  - **Configuration:**

ICW1	**00011011b**
ICW2	**01000000b**
ICW3	**00000001b**
ICW4	**No master slave**

OCW1	**00001111b**
OCW2	**10100XXXb**

- **Pin Connections:**

IR0	Perk button
IR1	Five Star button
IR2	Dairy milk button
IR3	ADC EOC
IR4	8254 OUT1 (via enabling logic)

- 2732
  - Two 2732 used
  - Smallest ROM chip available is 4K and as we need to have an even and an odd bank and IVT starts at 00000H

- 6116
  - Two 6116 used
  - Smallest RAM chip available is 2K and we need an odd and even bank. We need RAM for stack and temporary storage of data.

- ULN2003A
  - Four ULN2003A used
  - 16- pin IC
  - It has seven Darlington Pairs, which can drive loads upto 50V and 500mA and is used to drive the stepper motors.

- L293D
  - Dual H Bridge IC
  - Used to run solenoid that evacuates the coin from the pressure sensor and into the coin box.

- LS 138
  - 2 3x8 decoders

- Required Gates
    - LS 373, LS 245, LS 244

- Misclenlaneous
    - LED’s(3)
    - Switches(3)
    - Stepper Motors(4)
    - Solenoid (1)

Address Map

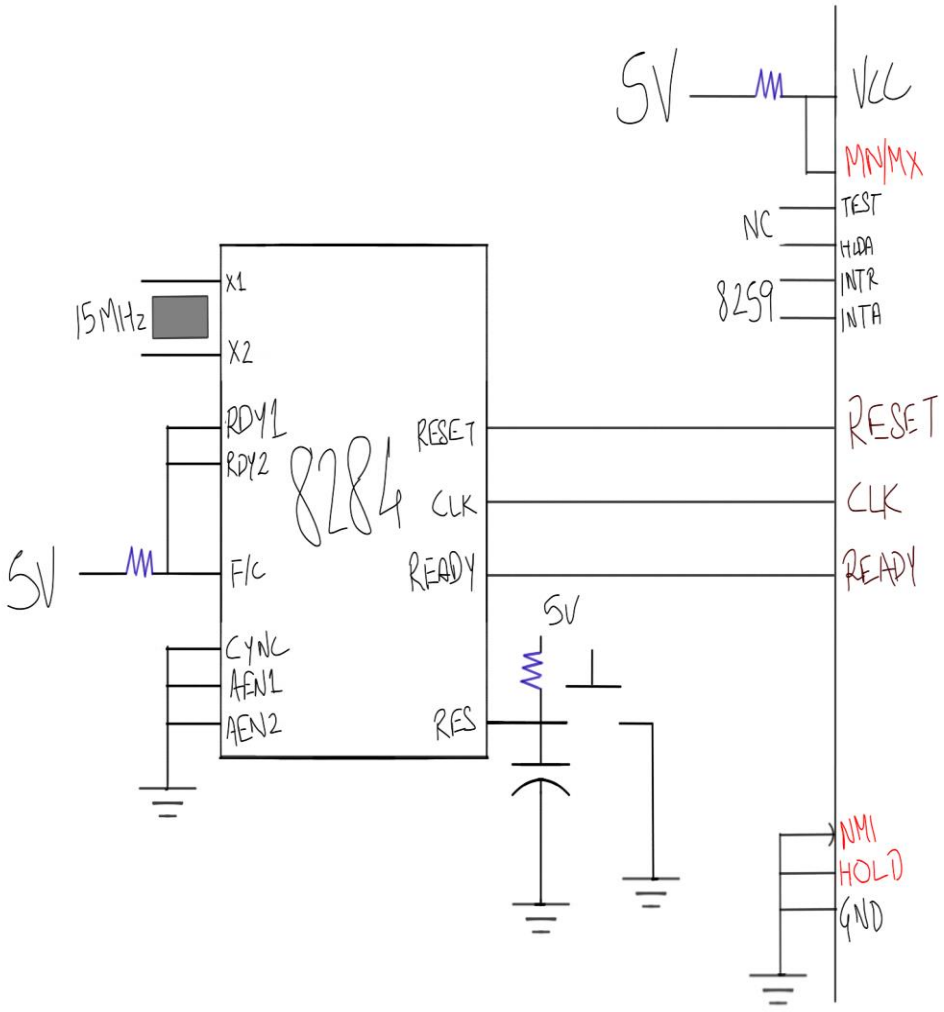
Memory Map

ROM – 00000H – 01FFFH → (8KB) with odd-even banking
RAM – 02000H – 02FFFH → (4KB) with odd-even banking

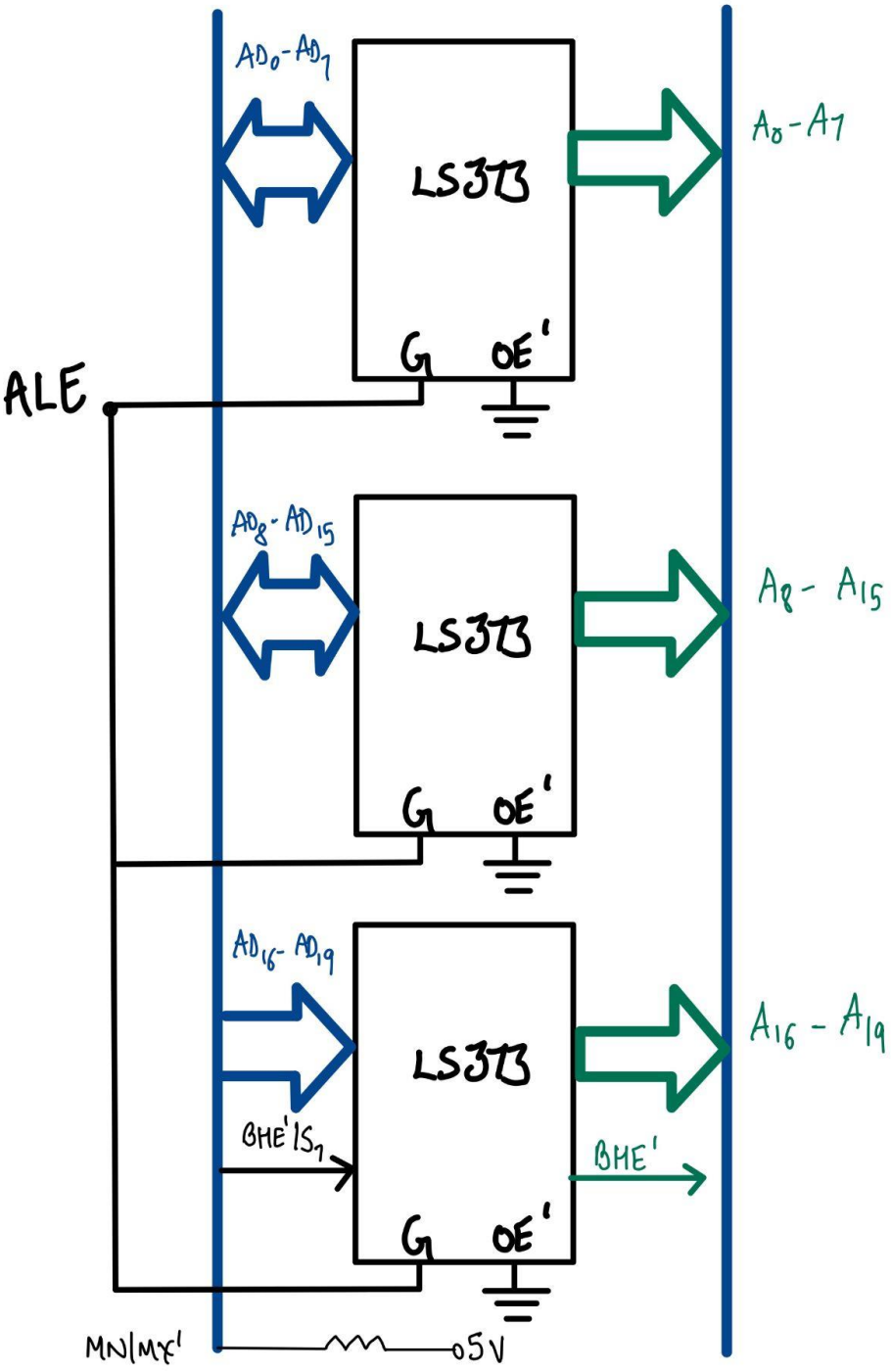
I/O Map

8255 (1) - 00H - 06H
8255 (2) - 08H - 0Eh
8254 - 10H - 16H
8259 - 18H - 1AH

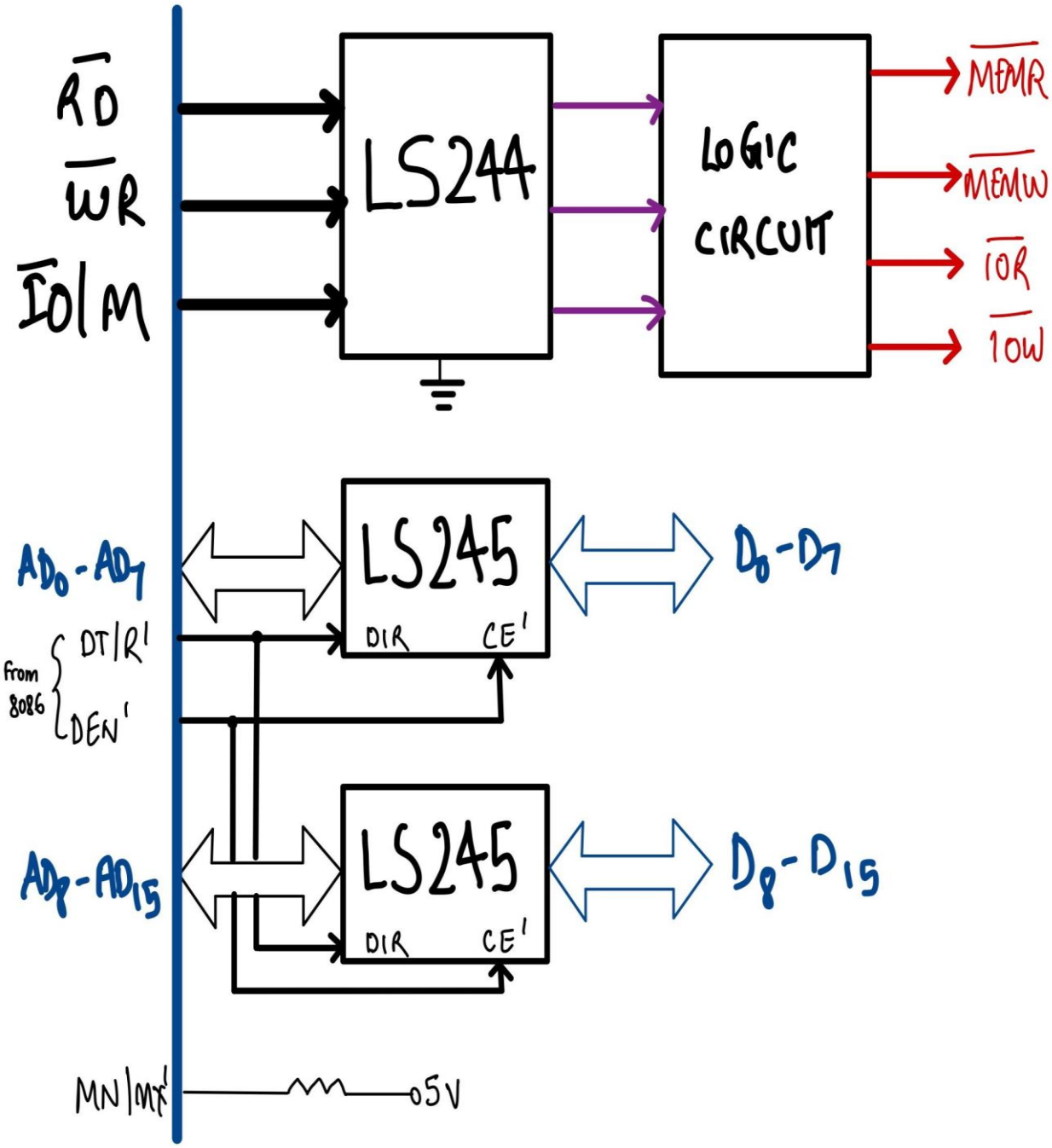
Design (Also attached in Design.pdf)



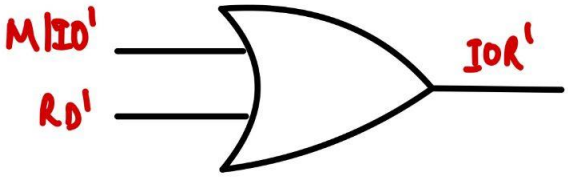
Inputs to 8086

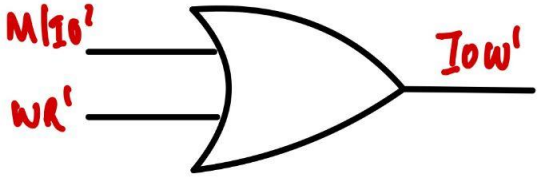


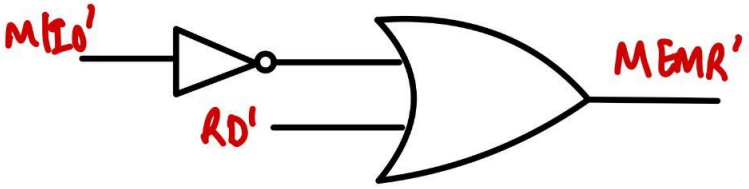
System Bus OF 8086 (Address)

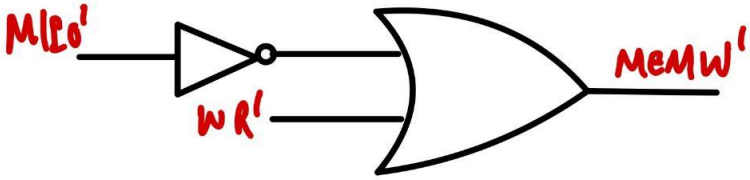


System Bus (Data + Control)



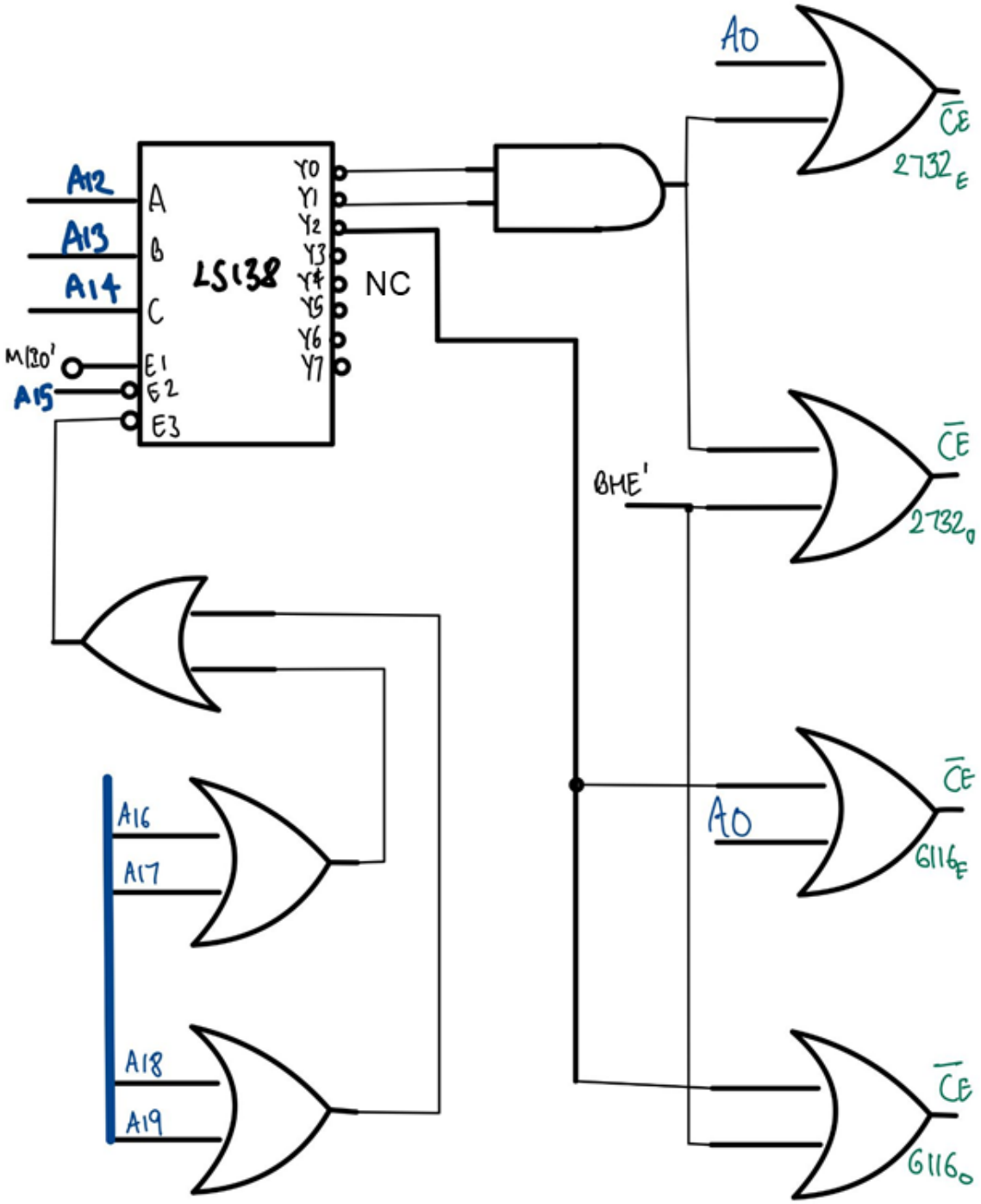




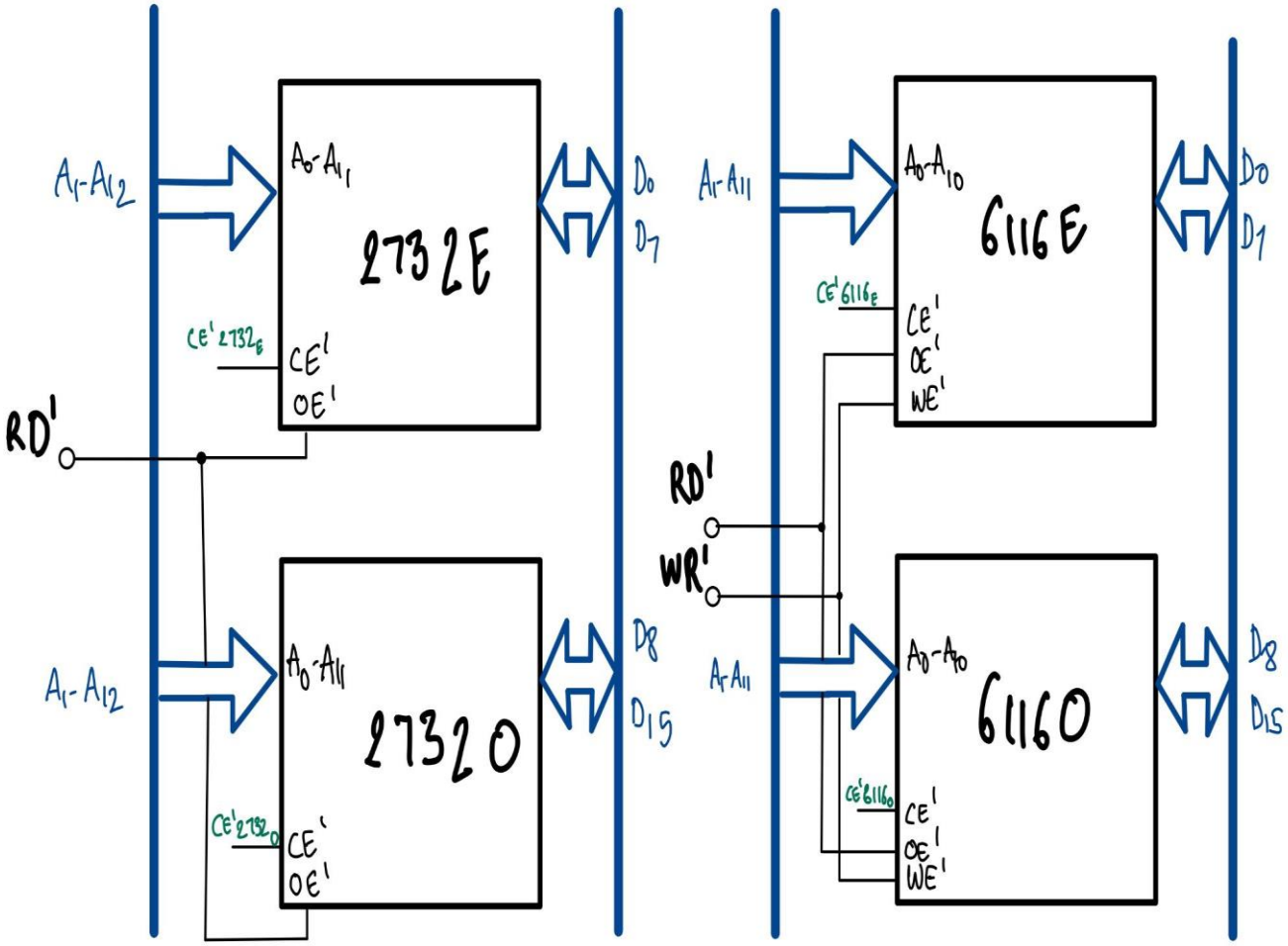


M/IO'	RD'	WR'	Bus Cycle
1	0	1	MEMR'
1	1	0	MEMW'
0	0	1	IOR'
0	1	0	IOW'

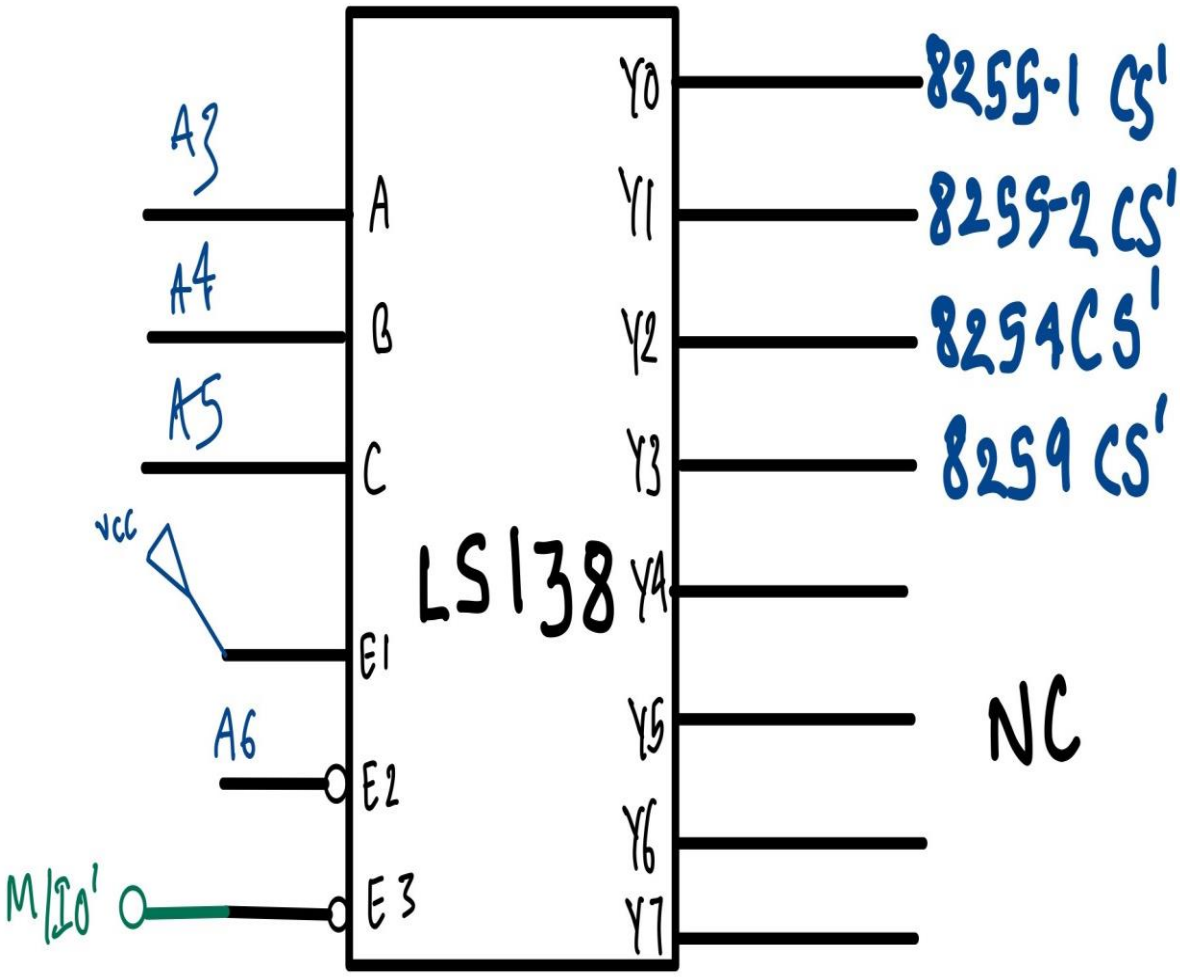
System Bus Logic



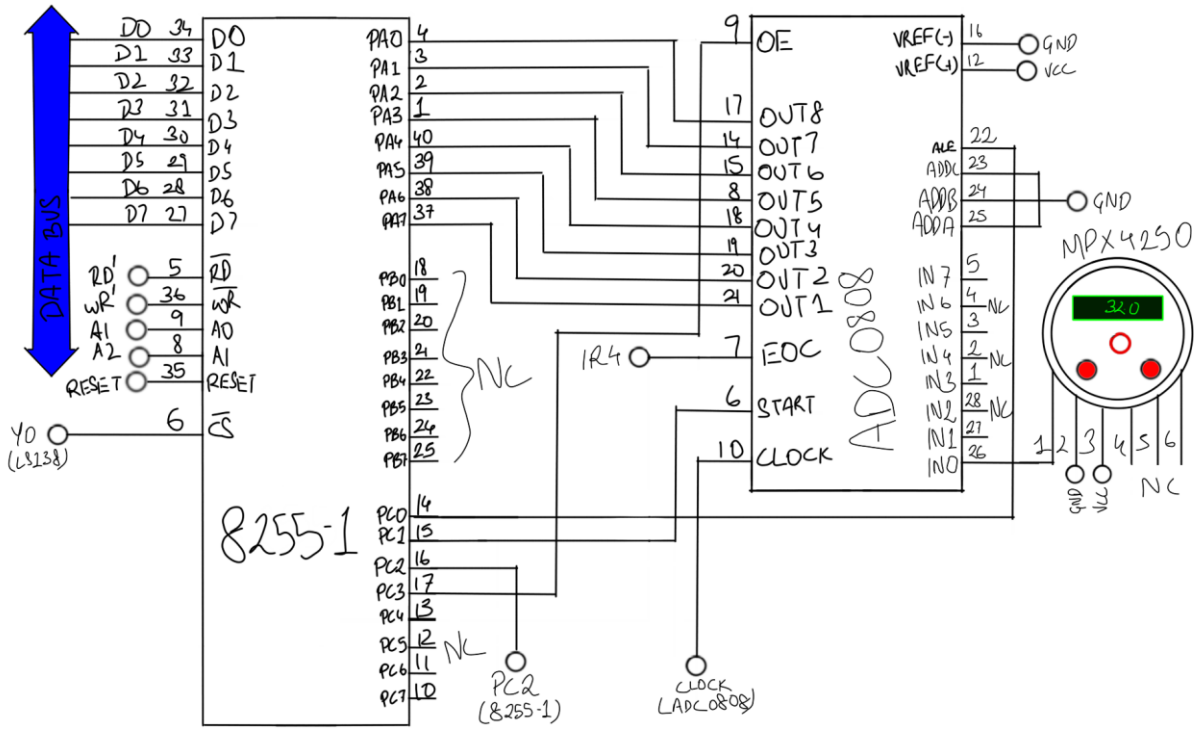
Memory Decoder



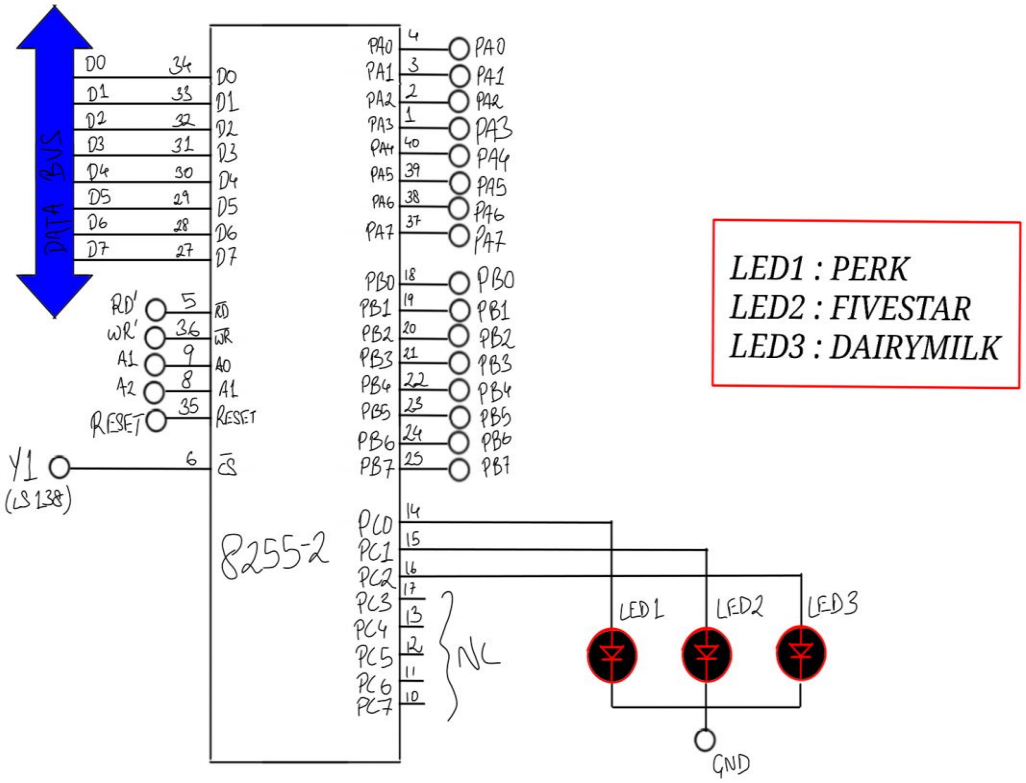
Memory Interfacing


A3
A4
A5
A6
M/IO'
Vcc
A
B
C
E1
E2
E3
LS138
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
8255-1 CS'
8255-2 CS'
8254 CS'
8259 CS'
NC

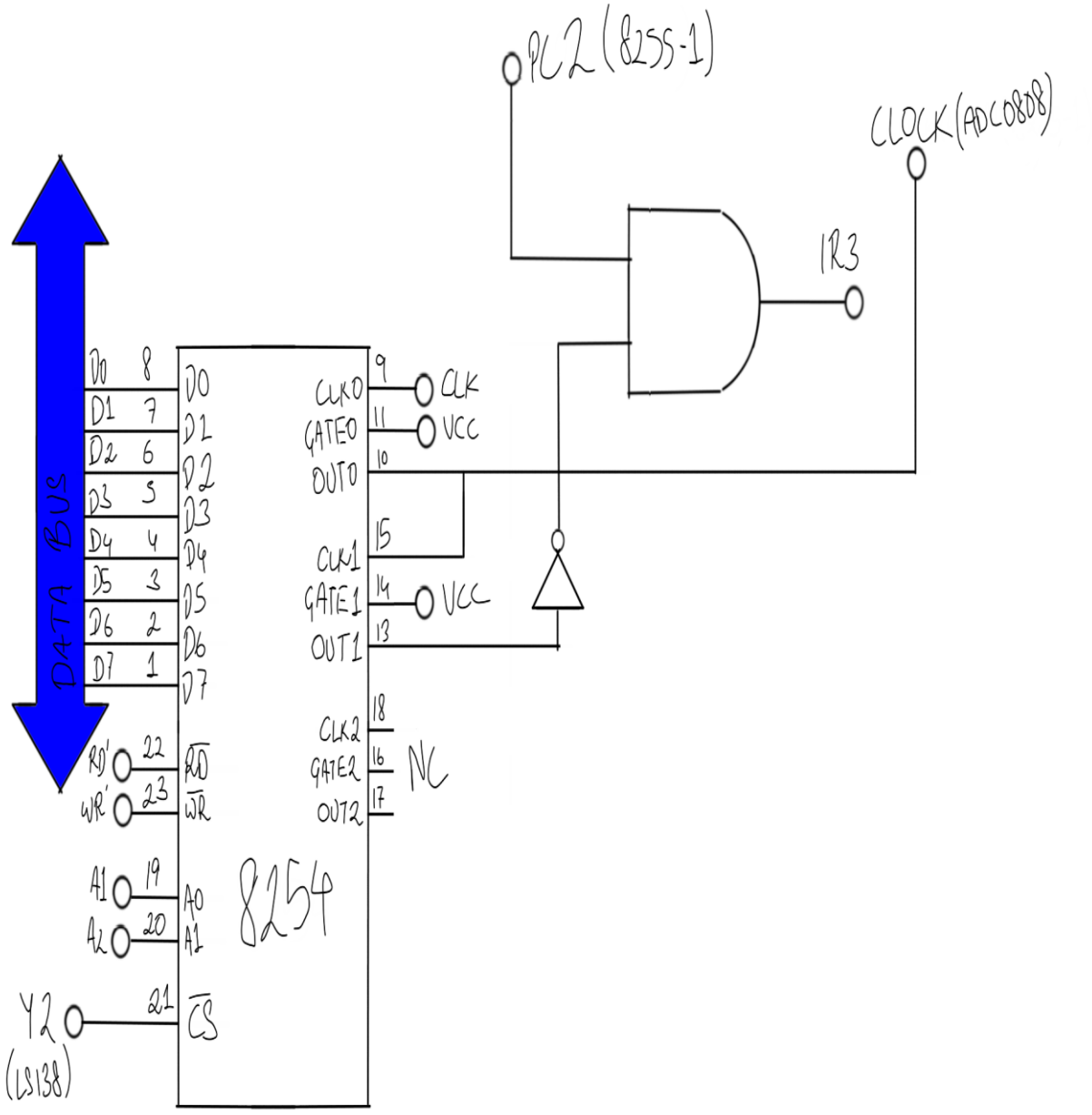
I/O Decoder


DATA BUS
D0
D1
D2
D3
D4
D5
D6
D7
RD'
WR'
A1
A2
RESET
Y0
(LS138)
8255-1
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
NC
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7
PC2
(8255-1)
IR4
CLOCK
(ADC0808)
OE
EOC
START
CLOCK
OUT8
OUT7
OUT6
OUT5
OUT4
OUT3
OUT2
OUT1
ADC0808
VREF(-)
VREF(+)
GND
VCC
ALE
ADDC
ADDB
ADDA
IN7
IN6
IN5
IN4
IN3
IN2
IN1
IN0
MPX4250
320
GND
VCC
NC

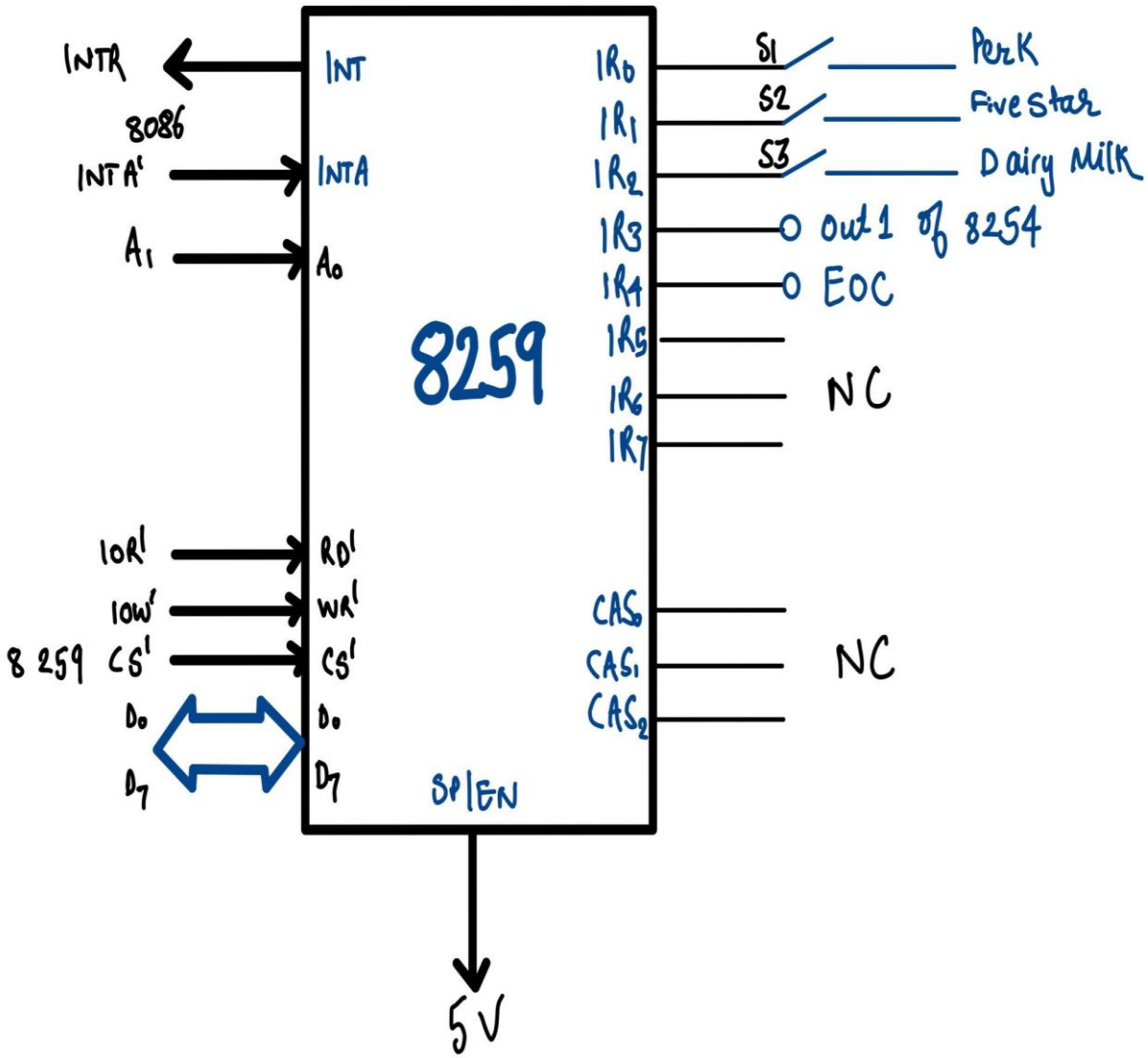
8255 -1


DATA BUS
D0
D1
D2
D3
D4
D5
D6
D7
RD'
WR'
A1
A2
RESET
Y1
(LS138)
8255-2
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7
NC
LED1
LED2
LED3
GND
LED1 : PERK
LED2 : FIVESTAR
LED3 : DAIRYMILK

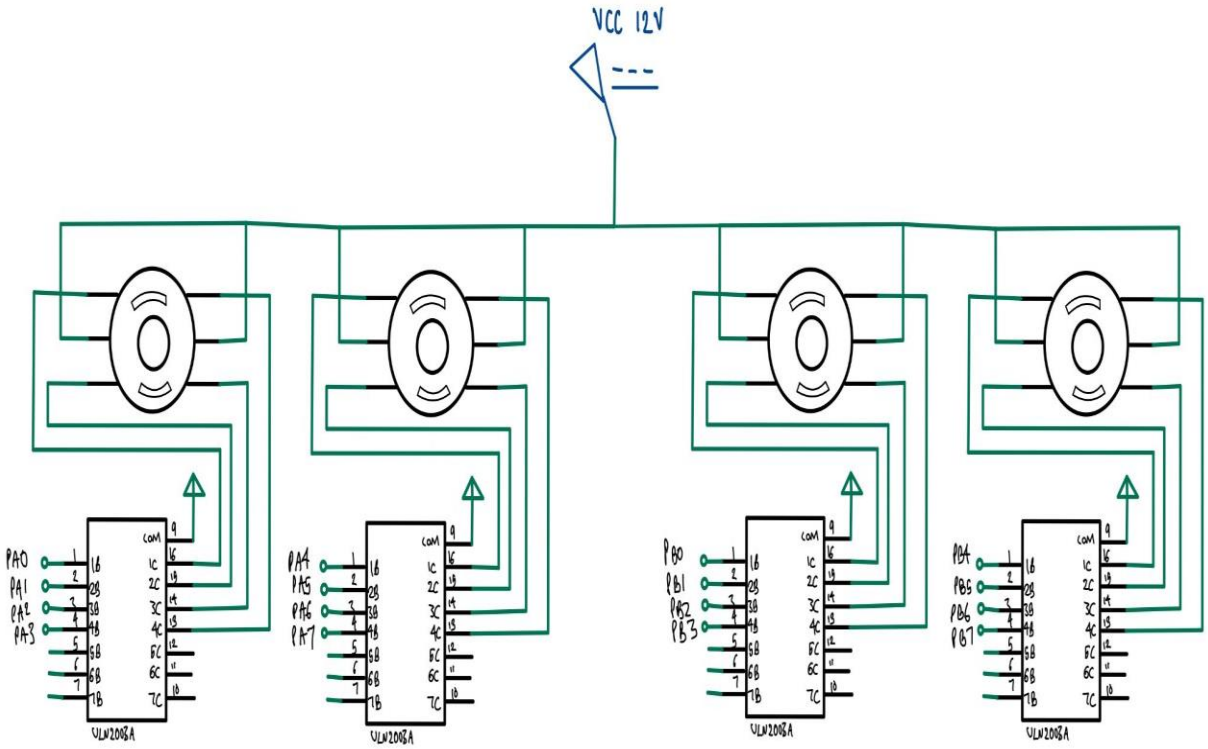
8255 - 2


PC2 (8255-1)
CLOCK (ADC0808)
IR3
DATA BUS
D0 8
D1 7
D2 6
D3 5
D4 4
D5 3
D6 2
D7 1
RD' 22
WR' 23
A1 19
A2 20
Y2 (LS138)
21
D0
D1
D2
D3
D4
D5
D6
D7
RD
WR
A0
A1
CS
8254
CLK0 9
GATE0 11
OUT0 10
CLK
VCC
CLK1 15
GATE1 14
VCC
OUT1 13
CLK2 18
GATE2 16
NC
OUT2 17

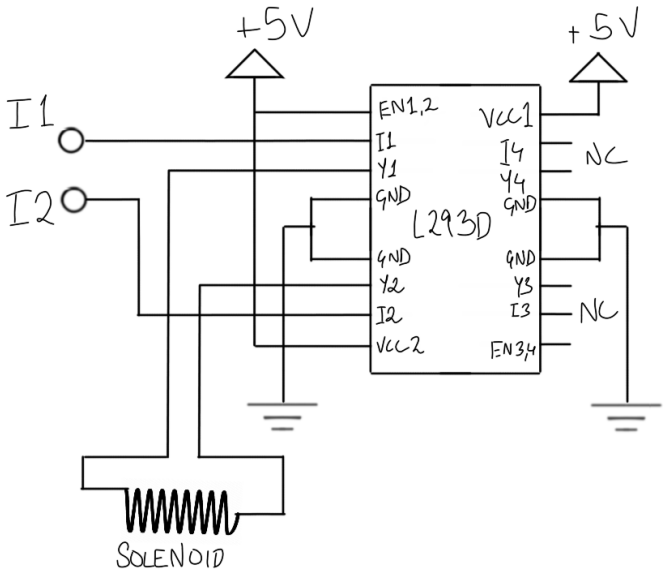
8254


INTR
INT
8086
INTA'
INTA
A1
A0
8259
IR0
IR1
IR2
IR3
IR4
IR5
IR6
IR7
S1
S2
S3
Perk
Five star
Dairy Milk
out 1 of 8254
EOC
NC
IOR'
RD'
IOW'
WR'
8259 CS'
CS'
D0
D0
D7
D7
SP/EN
CAS0
CAS1
CAS2
NC
5V

8259



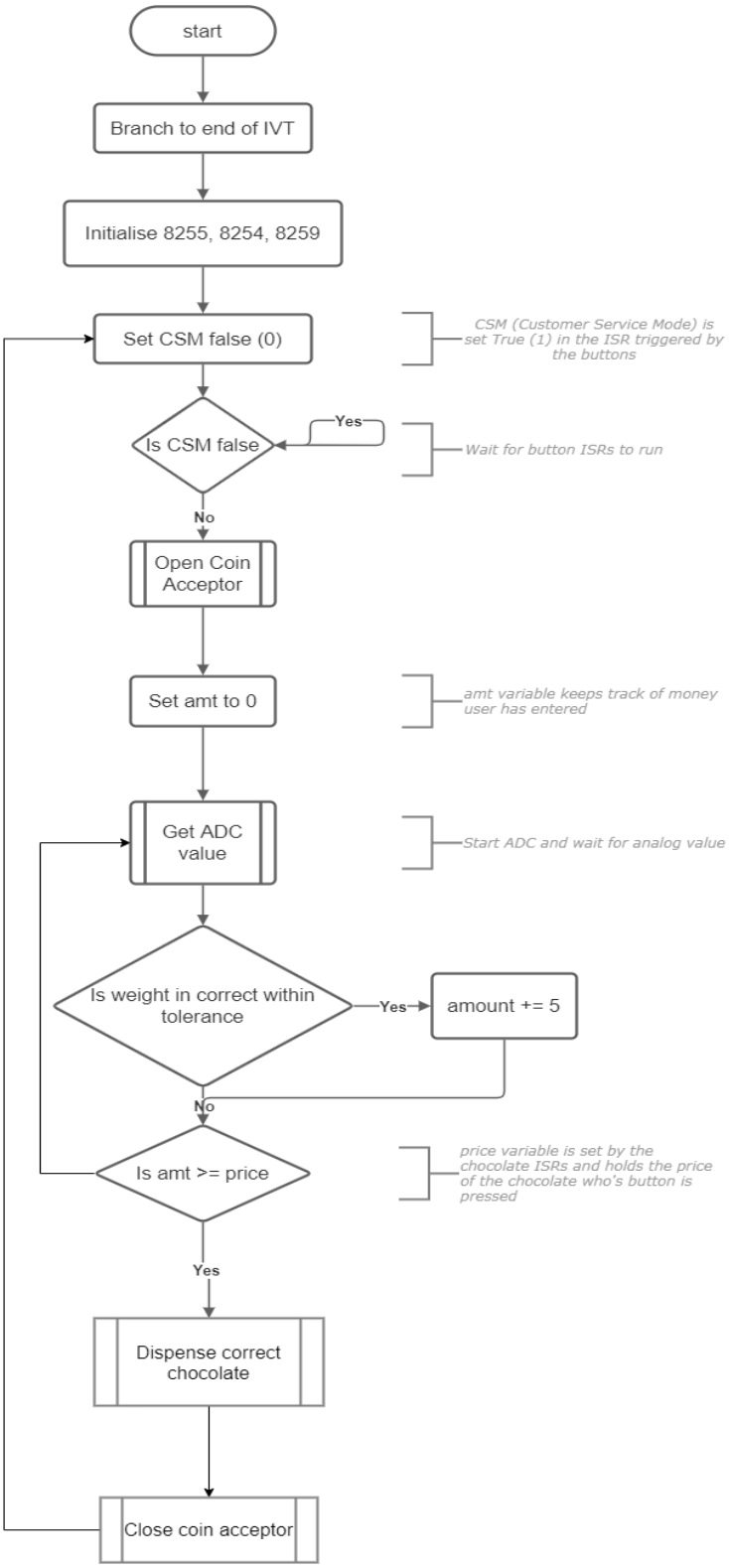
Stepper Motors



Solenoid
(connected from Port B of 8255-1)

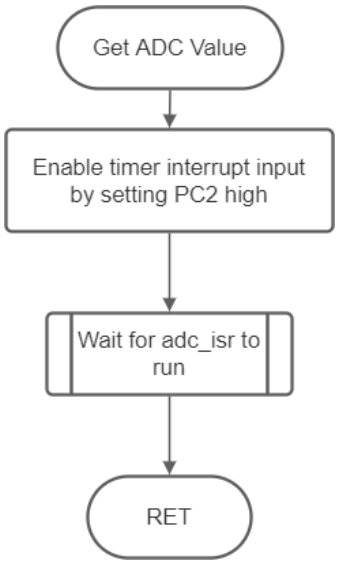
Flowcharts

Main Program

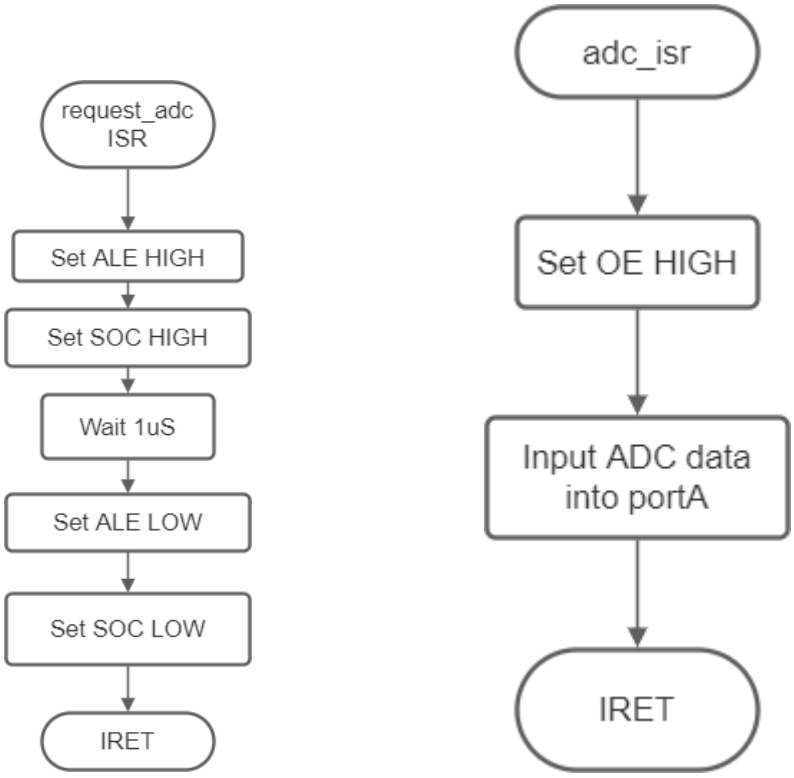


ADC functions

Subroutine that reads and stores ADC value

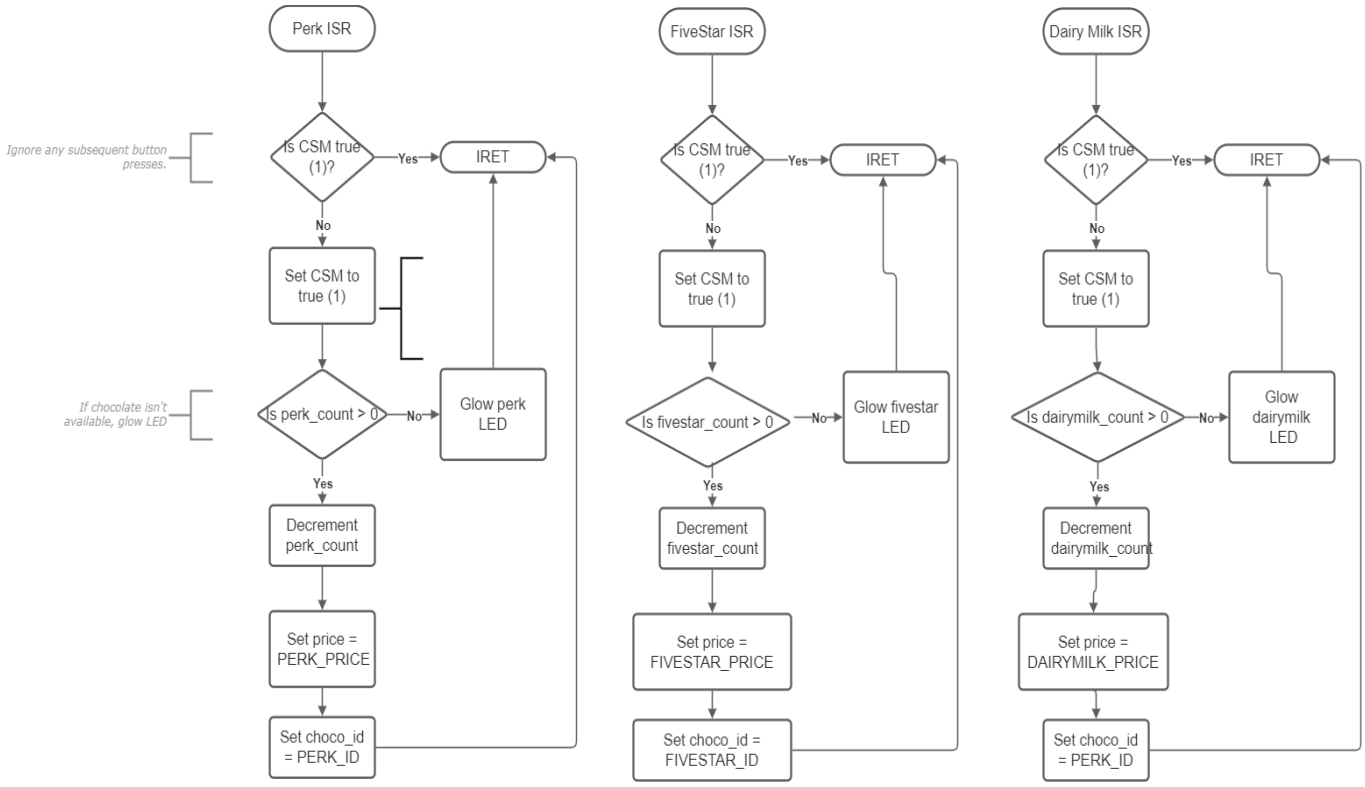


ISRs that request ADC to start conversion, and read data on end-of-conversion, respectively

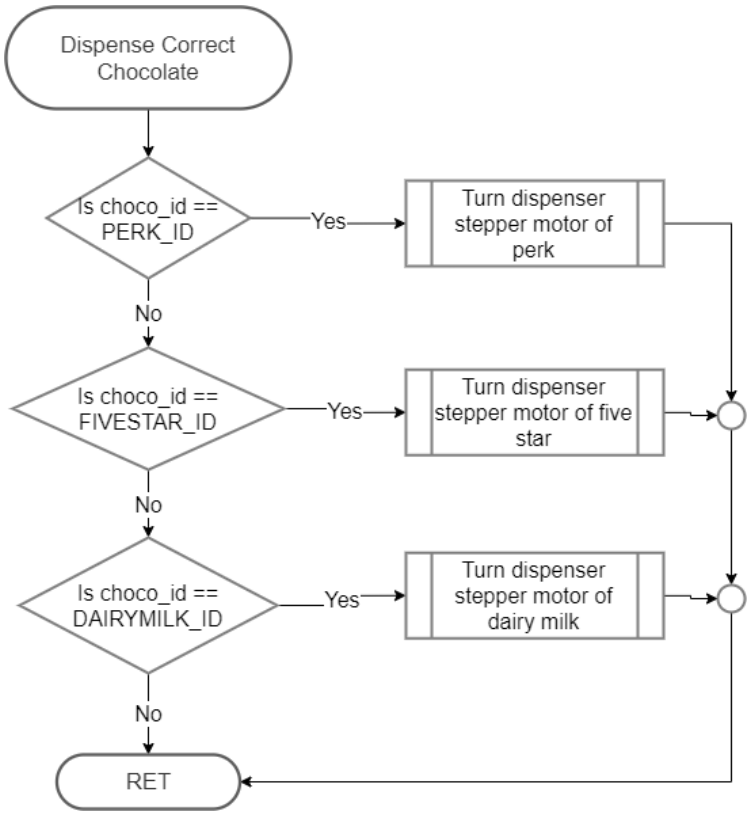


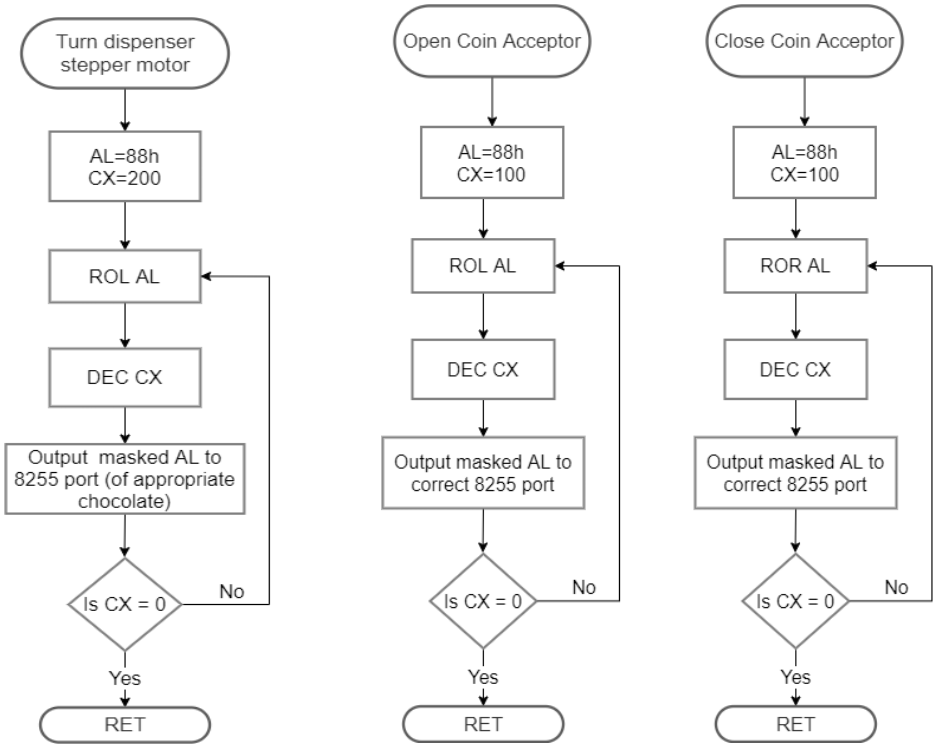
Chocolate Button ISRs

3 ISRs for each chocolate button that run when they are pressed



Dispense chocolate related functions





Variations in Proteus Implementation with justification

1. In Proteus, we have used polling for switches instead of interrupts because of the unavailability of 8259.
2. We have not used 8254 in proteus because it doesn't work properly for multiple counters, thus for ADC Clock we have used a Clock generator of proteus at 833KHz.
3. Since 8254 is not working properly in Proteus, we are not creating any timing signal for ADC checking instead we are just giving SOC pulse with delay in code itself. Therefore as compared to real life design where we will just measure the change in weight sensor every 100ms and compare with 5rs coin weight, in Proteus we are just comparing the exact value of money needed to be entered.
4. Since 8259 is not available we are directly polling EOC using 8255.
5. Instead of a pressure sensor we are giving direct DC voltage corresponding to different chocolates with the help of a 4 way rotary switch as input to the ADC, because all sensors are not available on Proteus.
6. In real life design we are also using a solenoid to clear the coins placed on the sensor to a money box when the desired price has been achieved Proteus does not have a solenoid.
7. ROM is only 00000 – as proteus allows to change reset address.

Firmware (Also attached in chocolate_vending.asm)

```
#make_bin#

; BIN is plain binary format similar to .com format, but not limited to 1 segment;
; All values between # are directives, these values are saved into a separate .binf file.
; Before loading .bin file emulator reads .binf file with the same file name.

; All directives are optional, if you don't need them, delete them.

; set loading address, .bin file will be loaded to this address:
#LOAD_SEGMENT=FFFFh#
#LOAD_OFFSET=0000h#

; set entry point:
#CS=0000h#    ; same as loading segment
#IP=0000h#    ; same as loading offset

; set segment registers
#DS=0000h#    ; same as loading segment
#ES=0000h#    ; same as loading segment

; set stack
#SS=0000h#    ; same as loading segment
#SP=FFFEh#    ; set to top of loading segment

; set general registers (optional)
#AX=0000h#
#BX=0000h#
#CX=0000h#
#DX=0000h#
#SI=0000h#
#DI=0000h#
#BP=0000h#

jmp     st1
nop

db    1024 dup(0)
; Main program
```

```
st1:    cli
; intialize ds,es,ss to start of RAM
mov    ax,0200h
mov    ds,ax
mov    es,ax
mov    ss,ax
mov    sp,0FFFEH
mov si,0000


; DATA
JMP START
PORTA1 EQU    00h
PORTB1 EQU    02h
PORTC1 EQU    04h
CREG1  EQU    06h
PORTA2 EQU  08h
PORTB2 EQU  0ah
PORTC2 EQU  0ch
CREG2  EQU  0eh
CNT0   EQU  10H
CREG3  EQU  16H
STEPPER_MOTOR EQU 88H

PERKC DB 100
FIVEC DB 100
DMC   DB 100

PERKID EQU 36
FIVEID EQU 61
DMID   EQU 86
PRICE  DB  ?




START:
; Initialize 8255A
```

```
; portA1 as input, portB1 is NC, portC1 lower as output and portc1 upper as input.

MOV    AL, 9AH ;10011010b
OUT    CREG1, AL
; portA2 as output, portB2 as output, portC2 lower as output and portc2 upper as input
MOV AL, 88H ;10001000b
OUT CREG2, AL

; initialise all ouput as 0
MOV AL,00
OUT PORTC1,AL
MOV AL,00H
OUT PORTA2,AL
MOV AL,00H
OUT PORTB2,AL



Main:

;first making sure that all keys are released
x1 : IN AL,PORTC2
CMP AL,70H
JNZ X1

;checking for a key press
x2:  IN AL,PORTC2
AND AL,70H
CMP AL,60H
JZ  PERK
CMP AL,50H
JZ  FIVESTAR
CMP AL,30H
JZ  DM
JMP X2  ; loop back if no button pressed

PERK:

; checking count of available Perk chocolates
CMP PERKC,0
```

```
JZ LED_GLOW_PERK

; if available then process starts
CALL ACCEPT_COIN
MOV PRICE,PERKID
CALL PRICE_INITIATE
CALL DELAY_20MS
CALL DISPENSE_PERK
CALL DELAY_20MS
DEC PERKC
CALL CLOSE_COIN
JMP START


FIVESTAR:

; checking count of available Five Star chocolates
CMP FIVEC,0
JZ LED_GLOW_FIVE

; if available then process starts
CALL ACCEPT_COIN
MOV PRICE,FIVEID
CALL PRICE_INITIATE
CALL DISPENSE_FIVE
DEC FIVEC
CALL CLOSE_COIN
JMP START


DM:

; checking count of available Dairy Milk chocolates
CMP DMC,0
JZ LED_GLOW_DM

; if available then process starts
CALL ACCEPT_COIN
MOV PRICE,DMID
CALL PRICE_INITIATE
```

```
CALL DISPENSE_DM
DEC DMC
CALL CLOSE_COIN
JMP START


LED_GLOW_PERK:
; to glow LED red indicating no Perk chocolate available
;PC0 IS HIGH FOR PERK
MOV AL,01H
OUT PORTC2,AL


LED_GLOW_FIVE:
; to glow LED red indicating no Five Star chocolate available
;PC1 IS HIGH FOR FIVESTAR
MOV AL,02H ;00000010B
OUT PORTC2,AL


LED_GLOW_DM:
; to glow LED red indicating no Dairy Milk chocolate available
;PC2 IS HIGH FOR DAIRYMILK
MOV AL,04H ;00000100B
OUT PORTC2,AL


hlt


ACCEPT_COIN PROC NEAR
; moves the stepper motor-4 to open the coin aceptance flap
PUSHF
PUSH AX
PUSH BX
PUSH CX

MOV AL,STEPPER_MOTOR
MOV CX,50      ; 50 is equivalent to 180 Deg rotation
```

```
ROT_MOTOR_4_CLKWISE: ; rotates the motor clockwise
MOV     BL,AL
AND     AL,0F0H
OUT     PORTB2,AL
CALL    DELAY_20MS
MOV     AL,BL
ROR     AL,01
DEC     CX
JNZ     ROT_MOTOR_4_CLKWISE

; shut off motor
MOV AL,00H
OUT PORTB2,AL

POP CX
POP BX
POP AX
POPF
RET
ACCEPT_COIN ENDP


PRICE_INITIATE PROC NEAR
; takes ADC input and waits until it becomes equal to required coin weight
PUSHF
PUSH AX
PUSH BX
PUSH CX

mov cl,PRICE

;ale activated
X8:
mov AL,01H ;00000001B
OUT PORTC1,AL

;soc high
mov AL,03H ;00000011B
OUT PORTC1,AL
```

```
; waiting
nop
nop
nop
nop

;ale low
and AL,11111110b
OUT PORTC1,AL
;soc low
and AL,11111101b
OUT PORTC1,AL

X7: ; checking for EOC high
IN AL,PORTC1
AND AL,10H
JZ X7

; OE high
MOV AL,04H
OUT PORTC1,AL

; taking ADC input
IN AL,PORTA1
CMP AL,CL   ; comparing to pre-defined coin weight required for the selected chocolate
JNZ X8  ; looping back to take another input from ADC if weight not matched

POP CX
POP BX
POP AX
POPF
RET
PRICE_INITIATE ENDP


DISPENSE_PERK PROC NEAR
; rotates the motor-1 to dispense Perk Chocolate
PUSHF
PUSH AX
PUSH BX
```

```
PUSH CX

MOV AL,STEPPER_MOTOR
MOV CX,100      ;100 IS EQUIVALENT TO 360 DEG ROTATION
ROT_MOTOR_1: MOV     BL,AL
AND     AL,0FH
OUT     PORTA2,AL
CALL    DELAY_20MS
MOV     AL,BL
ROL     AL,01
DEC     CX
JNZ     ROT_MOTOR_1

MOV AL,00
OUT PORTA2,AL

POP CX
POP BX
POP AX
POPF
RET
DISPENSE_PERK ENDP


DISPENSE_FIVE PROC NEAR
; rotates the motor-2 to dispense Five Star Chocolate
PUSHF
PUSH AX
PUSH BX
PUSH CX

MOV AL,STEPPER_MOTOR
MOV CX,100      ;100 IS EQUIVALENT TO 360 DEG ROTATION

ROT_MOTOR_2: MOV     BL,AL
AND     AL,0F0H
OUT     PORTA2,AL
CALL    DELAY_20MS
MOV     AL,BL
ROL     AL,01
```

```
DEC     CX
JNZ     ROT_MOTOR_2

MOV     AL,00
OUT     PORTA2,AL

POP CX
POP BX
POP AX
POPF
RET
DISPENSE_FIVE ENDP


DISPENSE_DM PROC NEAR
; rotates the motor-3 to dispense Dairy Milk Chocolate
PUSHF
PUSH AX
PUSH BX
PUSH CX

MOV AL,STEPPER_MOTOR
MOV CX,100      ;100 IS EQUIVALENT TO 360 DEG ROTATION

ROT_MOTOR_3: MOV     BL,AL
AND     AL,0FH
OUT     PORTB2,AL
CALL    DELAY_20MS
MOV     AL,BL
ROL     AL,01
DEC     CX
JNZ     ROT_MOTOR_3

MOV     AL,00
OUT     PORTB2,AL

POP CX
POP BX
POP AX
POPF
```

```
RET
DISPENSE_DM ENDP



CLOSE_COIN PROC NEAR
; moves the stepper motor-4 to close the coin aceptance flap
PUSHF
PUSH AX
PUSH BX
PUSH CX

MOV AL,STEPPER_MOTOR
MOV CX,50      ;50 IS EQUIVALENT TO 180 DEG ROTATION

ROT_MOTOR_4_ANTICLKWISE: MOV     BL,AL
AND     AL,0F0H
OUT     PORTB2,AL
CALL    DELAY_20MS
MOV     AL,BL
ROL     AL,01
DEC     CX
JNZ     ROT_MOTOR_4_ANTICLKWISE

MOV AL,00
OUT PORTB2,AL

POP CX
POP BX
POP AX
POPF
RET
CLOSE_COIN ENDP



DELAY_20MS PROC NEAR
; general delay function
PUSHF
PUSH AX
PUSH BX
PUSH CX
```

```
PUSH DX

NOP
NOP
NOP
NOP
NOP
NOP

POP DX
POP CX
POP BX
POP AX
POPF
RET
DELAY_20MS ENDP
```

List of Attachments

1. Complete Hardware Real World design - Design.pdf
2. Proteus file - chocolate_vending.DSN
3. EMU 8086 file - chocolate_vending.asm
4. Binary File after after assembly - chocolate_vending.bin
5. Manuals:
    - MPX5500DP
    - STEPPER MOTOR NEMA 17
    - ULN2003A
    - ADC0808

References

1. https://books.google.co.in/books?id=KJNpD2KimEsC&pg=PA228&lpg=PA22#v=onepage&q&f=false (for Servo Motors)
2. https://technobyte.org/8051-stepper-motor-interfacing/ (for ULN2003A)