Birla Institute of Technology and Science,Pilani

A REPORT
ON

Smart Lighting System

SHAMBHAVI RANI 2016A7PS0131P
GAUTAM PATHAK 2016A7PS0134P
SARTHAK AGARWAL 2016A7PS0135P
AMIT BANSAL 2016A7PS0140P

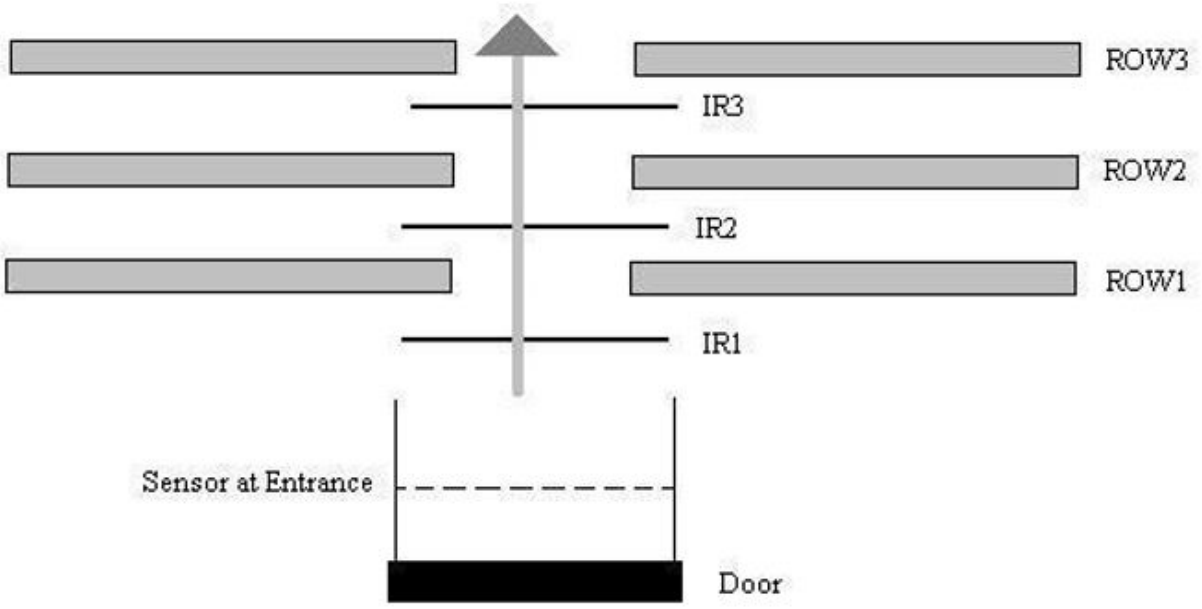
CONTENT

PROBLEM STATEMENT

System to be Designed: Smart Lighting System

Description: This is a lighting system for a conference room. On detection of a person the door is automatically opened/closed. As the seats get filled the light should be turned on. The rows are filled from row 1 onwards. There are 3 lights per row. As each row begins to get filled the lights get turned on as each rows empties completely the light gets turned off. You can assume there are at least 6 rows. The system should also display the number of people in the room on a LCD display.

System Details:



ASSUMPTIONS

1) There is one main entry to the conference room.
2) There are 2 sensors on either side of the door.
3) There are 2 sensors allotted for each row.
4) Each row has 3 lights.
5) People get seated from the first row onwards. So, if there are people in nth row then there must be people in (n-1)th row and thus lights would
be ON in all n rows.
6) No 2 sensors are activated at the exact same instant.
7) Only one person enters or exits the room at a particular time.
8) Maximum capacity of the conference room is 200.

SYSTEM DESCRIPTION

1) Input Devices:
    16 IR Sensors

2) Output Devices:
    18 LEDS

3) Two 8255(Programmable Peripheral Interface) chips interface to 8086.
    Port A: Input from IR sensors
    Port B: Input from IR sensors
    Port C: Output to LEDs

4) INTEL 8086 MICROPROCESSOR

HARDWARE DEVICES

CHIP NUMBER	CHIP	QUANTITY REQUIRED	USE
8086	MICROPROCESSOR	1	Central Processing Unit
6116	RAM 2K	2	Random access memory which contains DS,SS
2732	ROM 4K	2	Read only memory which contains entire code cs
74LS373	8 BIT LATCH	3	To latch address bus
74LS245	8 BIT BUFFER	2	To buffer data bus (bidirectional)
74LS138	3:8 DECODER	1	Used for select signals
8255	PROGRAMMABLE PERIPHERAL INTERFACE	2	Used for i/o
8284	CLOCK TIMER	1	For stable Clock signal
LED	COMMON CATHODE CONFIGURATION	18	For lighting
LM020L	LCD DISPLAY	1	For Display
L293D	STEPPER MOTOR	1	For opening/ closing the Door

MAPPING

Memory Organization:

The system uses 4KB of RAM and 8KB of ROM. RAM consists of two 2K chips and ROM consists of 4K chips. They are organized into odd and even bank to facilitate both byte and word size data transfers.

Read Only Memory: Starting Address: 00000h, Ending Address: 01FFFh

Random Access Memory: Starting Address: 02000h, Ending Address: 02FFFh

CHIP	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
ROM :FROM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ROM :TO	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
RAM :FROM	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
RAM :TO	0	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1

I/O Mapping:

8255-0 (Connected to switches and LEDs)

PORT A –INPUT--00000H

PORT B –INPUT--00002H

PORT C –OUTPUT--00004H

CONTROL REGISTER –00006H

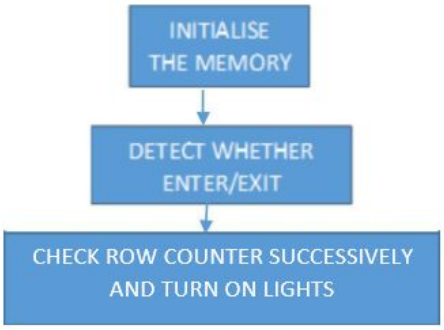
8255-1(Connected to motor and LCD)
LCD_DATA --OUTPUT-- 00008H

INPUTS – 00010H (Not used)

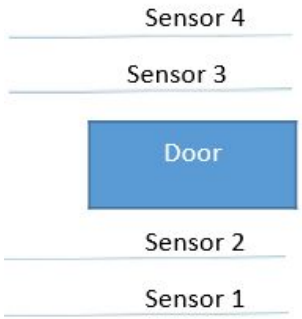
LCD_MOTOR_CONTROL – UPPER-INPUT, LOWER-OUTPUT-- 00012H

CONTROL REGISTER -- 00014H

FLOWCHART



THE LOGIC FOR OPENING AND CLOSING OF DOOR:



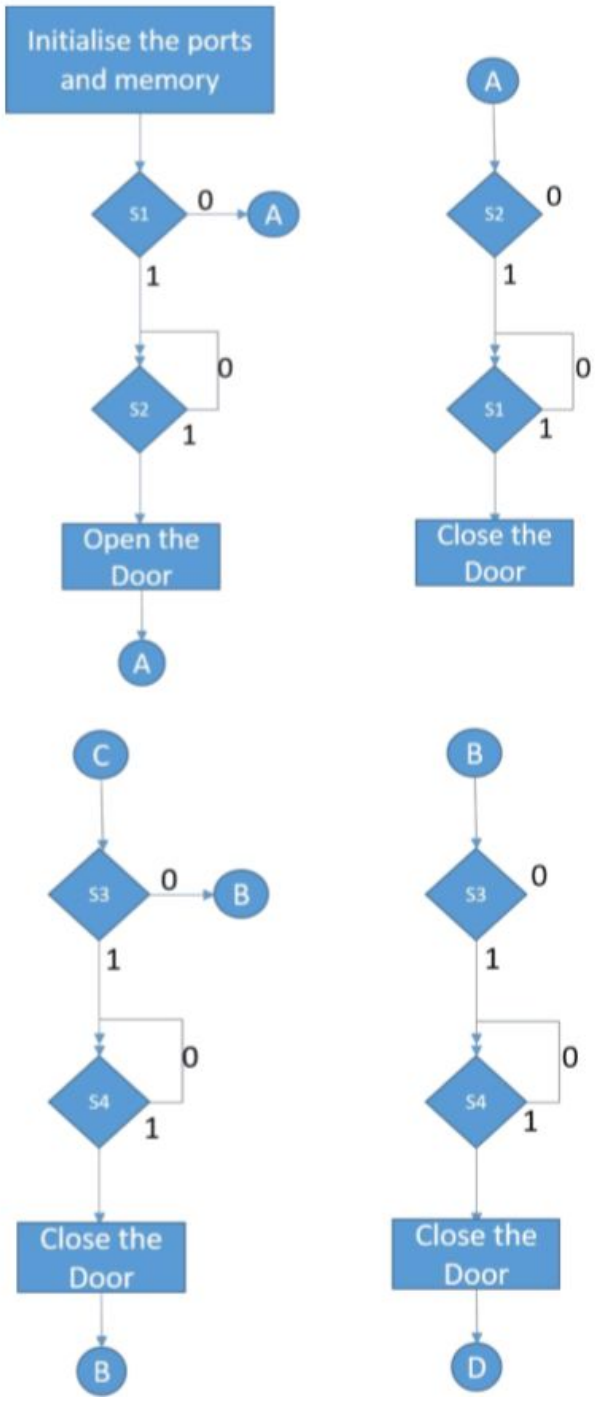
NOTE:

S1: Check sensor 1 for input

S2: Check sensor 2 for input

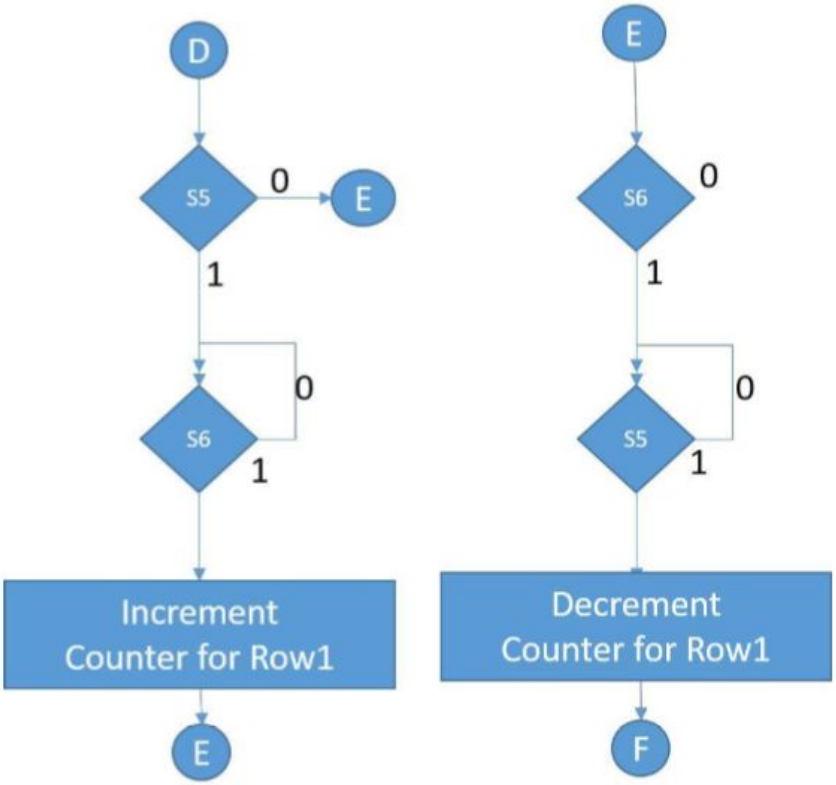
S3: Check sensor 3 for input

S4: Check sensor 4 for input

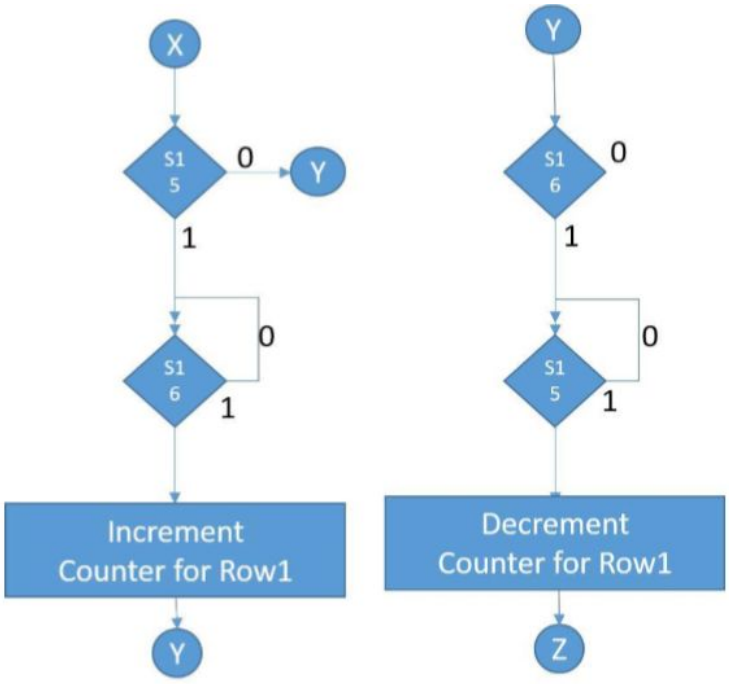

Initialise the ports and memory
S1
0
A
1
S2
0
1
Open the Door
A
A
S2
0
1
S1
0
1
Close the Door
C
S3
0
B
1
S4
0
1
Close the Door
B
B
S3
0
1
S4
0
1
Close the Door
D

THE LOGIC FOR CHECKING ROW COUNT:

For Row 1



Similarly for each row… Finally for Row 6



CODE

```
#make_bin#

#LOAD_SEGMENT=FFFFh#
#LOAD_OFFSET=0000h#

#CS=0000h#
#IP=0000h#

#DS=0000h#
#ES=0000h#

#SS=0000h#
#SP=FFFEh#

#AX=0000h#
#BX=0000h#
#CX=0000h#
#DX=0000h#
#SI=0000h#
#DI=0000h#
#BP=0000h#

VARS:

strlen DB 0
empty DB 'EMPTY'
full DB 'FULL'

;main program

counter dw 0
rcounter db 6 dup(0)
gate db 0

;8255-0
porta0 equ 00h
portb0 equ 02h
portc0 equ 04h
command_address0 equ 06h

;8255-1

inputs EQU 0Ah
lcd_data EQU 08h
```

```
lcd_motor_control EQU 0Ch
creg_io EQU 0Eh

jmp     st1
db     1024 dup(0)



st1:   cli
; intialize ds, es,ss to start of RAM
mov       ax,02000h
mov       ds,ax
mov       es,ax
mov       ss,ax
mov       sp,02FFEH

;intialise porta and portb as input & portc as output for 8255-0
mov     al,92h
out     command_address0,al

;initialise porta as output for lcd and portc upper as input and portc lower as output

MOV AL,10000000b
OUT creg_io,AL


;initialise hardware
; initialise the lcd
; check for busy status
; clear the screen
; display 'empty'
;writing on the command register for initialization


startup:
LCD_initialization
CALL update_the_LCD
CALL DELAY_1S
;calling lcd initialization

;;check for entry to door which triggers opening of door
x1: in al,02h
   and al,80H
   cmp al,80H
   jnz x2
y1: in al,02h
   and al,40h
   cmp al,40h
```

```
    jnz y1
    inc counter
    mov gate,1
call DELAY_1S
    motor_anticlockwise

;;check for exit from door which triggers closing of door
x2: in al,02h
    and al,40h
    cmp al,40h
    jnz x3
y2: in al,02h
    and al,80H
    cmp al,80H
    jnz y2
    dec counter
    mov gate,0
call DELAY_1S
;;check for entry on other side of door which triggers closing of door
x3: in al,02h
    and al,20h
    cmp al,20h
    jnz x4
y3: in al,02h
    and al,10h
    cmp al,10h
    jnz y3
    mov gate,0
call DELAY_1S
;;check for exit on other side of door which triggers opening of door
x4: in al,02h
    and al,10h
    cmp al,10h
    jnz x5
y4: in al,02h
    and al,20h
    cmp al,20h
    jnz y4
    mov gate,1
call DELAY_1S
motor_anticlockwise
;;check for entry in row1
x5: in al,02h
    and al,08h
    cmp al,08h
    jnz x6
y5: in al,02h
    and al,04h
```

```
    cmp al,04h
    jnz y5
    inc [rcounter+0]
call DELAY_1S
;;check for exit from row1
x6: in al,02h
    and al,04h
    cmp al,04h
    jnz x7
y6: in al,02h
    and al,08h
    cmp al,08h
    jnz y6
    dec [rcounter+0]
call DELAY_1S
;;check for entry in row2
x7: in al,02h
    and al,02h
    cmp al,02h
    jnz x8
y7: in al,02h
    and al,01h
    cmp al,01h
    jnz y7
    inc [rcounter+1]
call DELAY_1S
;;check for exit from row3
x8: in al,02h
    and al,01h
    cmp al,01h
    jnz x9
y8: in al,02h
    and al,02h
    cmp al,02h
    jnz y8
    dec [rcounter+1]
call DELAY_1S
;;check for entry in row3
x9: in al,00h
    and al,80H
    cmp al,80H
    jnz x10
y9: in al,00h
    and al,40h
    cmp al,40h
    jnz y9
    inc [rcounter+2]
call DELAY_1S
```

```
;;check for exit from row3
x10: in al,00h
   and al,40H
   cmp al,40H
   jnz x11
y10: in al,00h
   and al,80h
   cmp al,80h
   jnz y10
   dec [rcounter+2]
call DELAY_1S
;;check for entry in row4
x11: in al,00h
   and al,20h
   cmp al,20h
   jnz x12
y11: in al,00h
   and al,10h
   cmp al,10h
   jnz y11
   inc [rcounter+3]
call DELAY_1S
;;check for exit from row4
x12: in al,00h
   and al,10H
   cmp al,10H
   jnz x13
y12: in al,00h
   and al,20h
   cmp al,20h
   jnz y12
   dec [rcounter+3]
call DELAY_1S
;;check for entry in row5
x13: in al,00h
   and al,08h
   cmp al,08h
   jnz x14
y13: in al,00h
   and al,04h
   cmp al,04h
   jnz y13
   inc [rcounter+4]
call DELAY_1S
;;check for exit from row5
x14: in al,00h
   and al,04H
   cmp al,04H
```

```
    jnz x15
y14: in al,00h
    and al,08h
    cmp al,08h
    jnz y14
    dec [rcounter+4]
call DELAY_1S
;;check for entry in row6
x15: in al,00h
    and al,02h
    cmp al,02h
    jnz x16
y15: in al,00h
    and al,01h
    cmp al,01h
    jnz y15
    inc [rcounter+5]
call DELAY_1S
;;check for exit from row6
x16: in al,00h
    and al,01H
    cmp al,01H
    jnz x
y16: in al,00h
    and al,02h
    cmp al,02h
    jnz y16
    dec [rcounter+5]
call DELAY_1S
;;output for leds is made 1 wherever rcounter is 1

x:  mov al,00000000b
    mov bl,00000100b
    mov cx,6
    lea si,rcounter
y:  cmp [si],0
    jnz z
    jmp w
z:  or al,bl
    rol bl,1
w:  inc si
    loop y

    out 04h,al
    call DELAY_1S
    call DELAY_1S

disp: CALL update_the_LCD
```

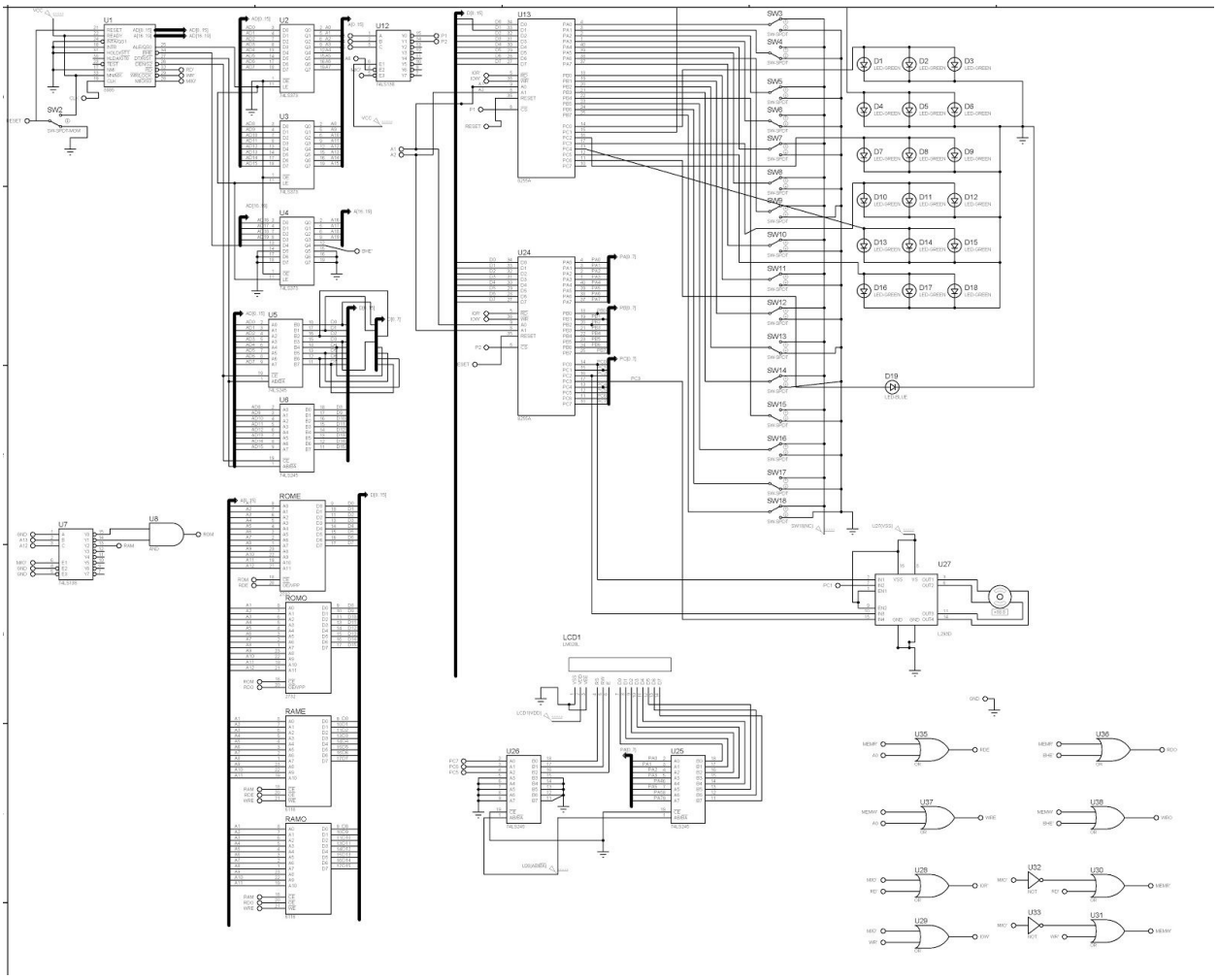
```
    jmp x1

;;;;;UPTIL HERE ALL LEDS IN THE ROWS WITH NON ZERO COUNT WILL GLOW
DELAY_1S PROC
   mov cx,50
   t: CALL DELAY
      loop t
   ret
DELAY_1s endp

DELAY PROC
   MOV CX, 1325 ;1325*15.085 usec = 20 msec
   W1:
      NOP
      NOP
      NOP
      NOP
      NOP
   LOOP W1
   RET
DELAY ENDP

macros:

set_the_LCD_mode MACRO
                  IN AL, lcd_motor_control
                  AND AL, 00011111b
                  OR AL, BL
                  OUT lcd_motor_control, AL
ENDM

LCD_initialization MACRO
                  MOV AL, 00001111b
                  OUT lcd_data, AL
                  MOV BL, 00100000b
set_the_LCD_mode
                  MOV BL, 00000000b
set_the_LCD_mode
ENDM

lcd_clear MACRO
                  MOV AL, 00000001b
OUT lcd_data, AL
                  MOV BL,00100000b
set_the_LCD_mode
                  MOV BL,00000000b
set_the_LCD_mode
ENDM
```

```
lcd_putch MACRO
                PUSH AX
                OUT lcd_data,AL
                call DELAY_1S
                MOV BL,10100000b
set_the_LCD_mode
                MOV BL,10000000b
set_the_LCD_mode
                POP AX
                ENDM

putstring_on_LCD MACRO
                MOV CH,0
                MOV CL, strlen
putting:
                MOV AL, [DI]
lcd_putch
                INC DI
                LOOP putting
ENDM

lcd_bcd MACRO
                MOV AX, counter
                MOV CX, 0
converting:
                MOV BL, 10
                DIV BL
                ADD AH, '0'
                MOV BL, AH
                MOV BH, 0
                PUSH BX
                INC CX
                MOV AH, 0
                CMP AX, 0
JNE converting
printing:
POP AX
lcd_putch
LOOP printing
ENDM

procs:
                update_the_LCD PROC NEAR
                lcd_clear
                MOV AL, ' '
                lcd_putch
                CMP counter, 0
```

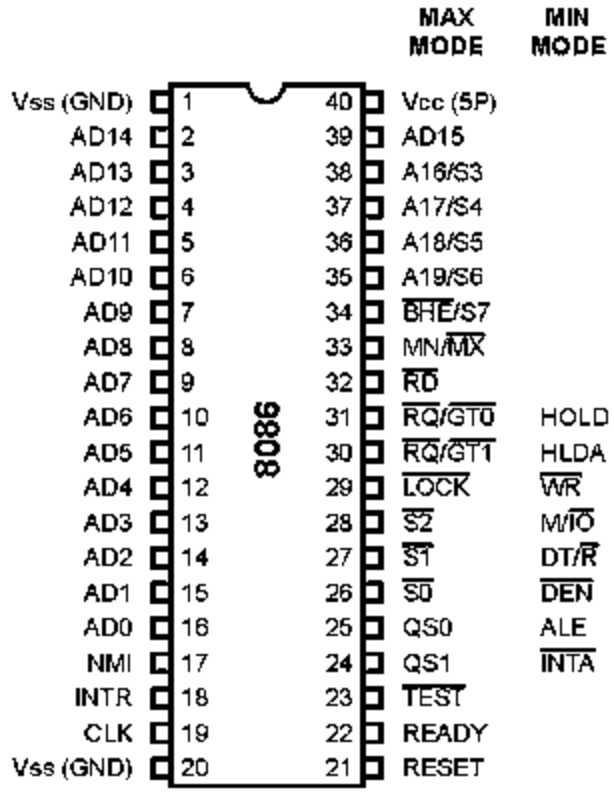
```
        JNZ notempty
        LEA DI, empty
        MOV strlen, 5
        JMP loaded
        notempty:
        CMP counter,2000
        JL notfull
        LEA DI, full
        MOV strlen, 4
        JMP loaded
        notfull:
        lcd_bcd
        RET
        loaded:
        call DELAY_1S
        putstring_on_LCD
                call DELAY_1S
        RET
        update_the_LCD ENDP



motor_anticlockwise MACRO
IN AL, lcd_motor_control
AND AL, 11111100b
OR AL, 00000010b
OUT lcd_motor_control, AL
call DELAY_1S
   call DELAY_1S
   call DELAY_1S
   call DELAY_1S
ENDM

motor_clockwise MACRO
IN AL, lcd_motor_control
AND AL, 11111100b
OR AL, 00000001b
OUT lcd_motor_control, AL
call DELAY_1S
   call DELAY_1S
   call DELAY_1S
   call DELAY_1S
ENDM

stopping_the_motor MACRO
IN AL, lcd_motor_control
AND AL, 11111100b
OR AL, 00000000b
```

```
OUT lcd_motor_control, AL
call DELAY_1S
   call DELAY_1S
   call DELAY_1S
   call DELAY_1S
ENDM
```

Circuit Diagram



REFERENCES


MAX MODE
MIN MODE
8086
Vss (GND) 1
AD14 2
AD13 3
AD12 4
AD11 5
AD10 6
AD9 7
AD8 8
AD7 9
AD6 10
AD5 11
AD4 12
AD3 13
AD2 14
AD1 15
AD0 16
NMI 17
INTR 18
CLK 19
Vss (GND) 20
40 Vcc (5P)
39 AD15
38 A16/S3
37 A17/S4
36 A18/S5
35 A19/S6
34 BHE/S7
33 MN/MX
32 RD
31 RQ/GT0 HOLD
30 RQ/GT1 HLDA
29 LOCK WR
28 S2 M/IO
27 S1 DT/R
26 S0 DEN
25 QS0 ALE
24 QS1 INTA
23 TEST
22 READY
21 RESET

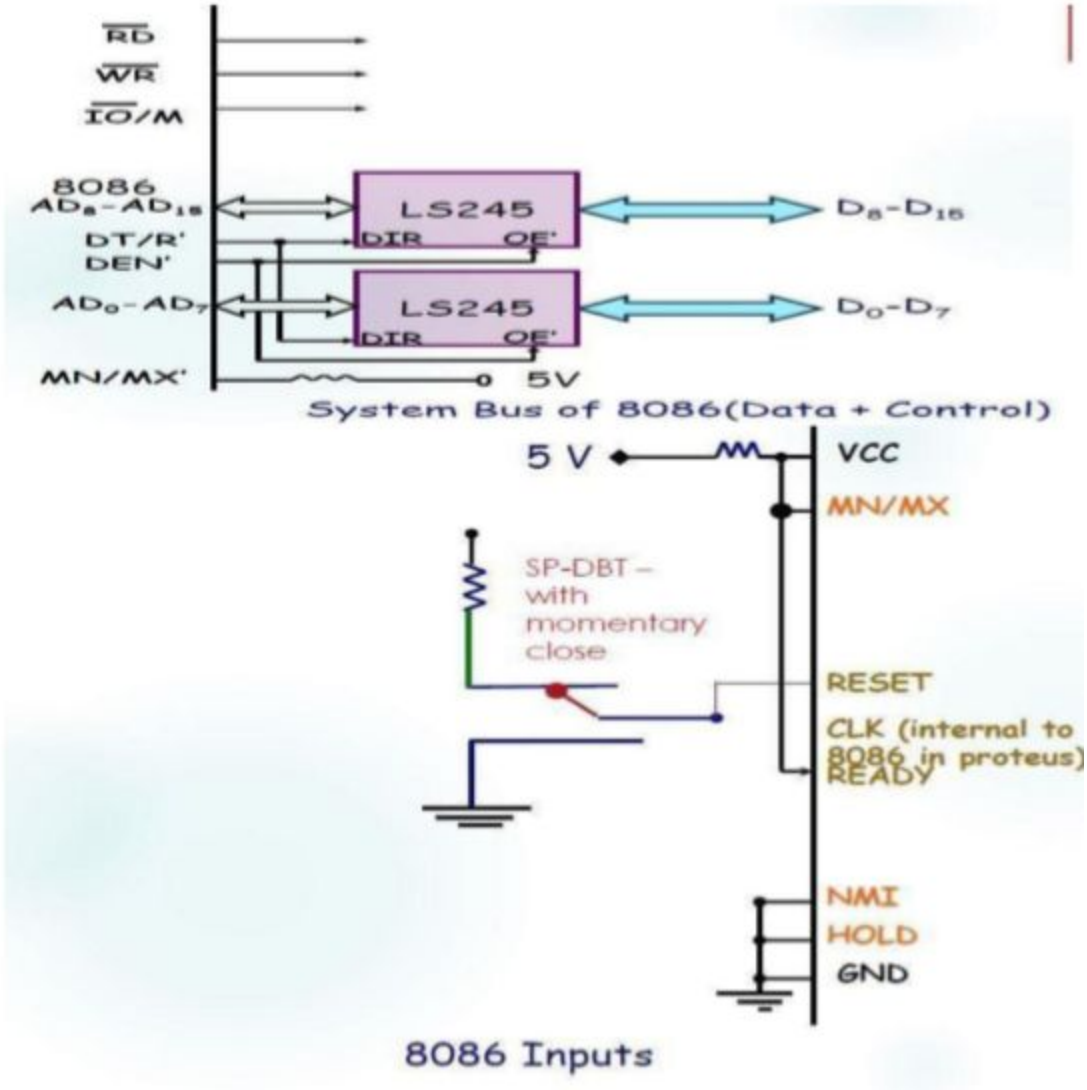
PIR Sensor Module by Parallax Inc. #555-28027

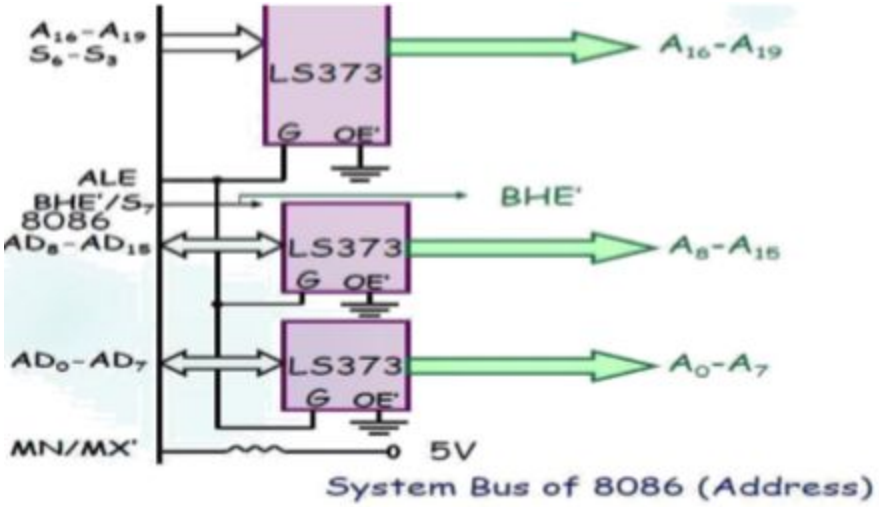
PIRs can detect levels of
ambient infrared radiation. Everything emits some low level radiation, and the hotter something is, the more radiation is emitted. The sensor in a motion detector is split in two halves. If one half sees more or less IR radiation than the other, the output will swing high or low.

Along with the pyroelectric sensor is a bunch of supporting circuitry, resistors and capacitors. The module here uses the BISS0001 ("Micro Power PIR Motion Detector IC"). This chip takes the output of the sensor and does some minor processing on it to emit a digital output pulse from the analog sensor.

Parts of Circuit

Memory Interfacing with 8086:





ADDRESSING 8255:

00 h – 06h 8255

MEMORY INTERFACING

RAM – 6116 (2 chips) (RAM (Even) AND RAM (0DD))
ROM – 2732(2 chips) (ROM (Even) AND ROM (ODD))
ROM 00000H - 01FFFH

RAM 02000H – 02FFFH