Birla Institute of Technology and Science, Pilani

A REPORT

ON

Smart Lighting System

(Group 66 – Problem 14)

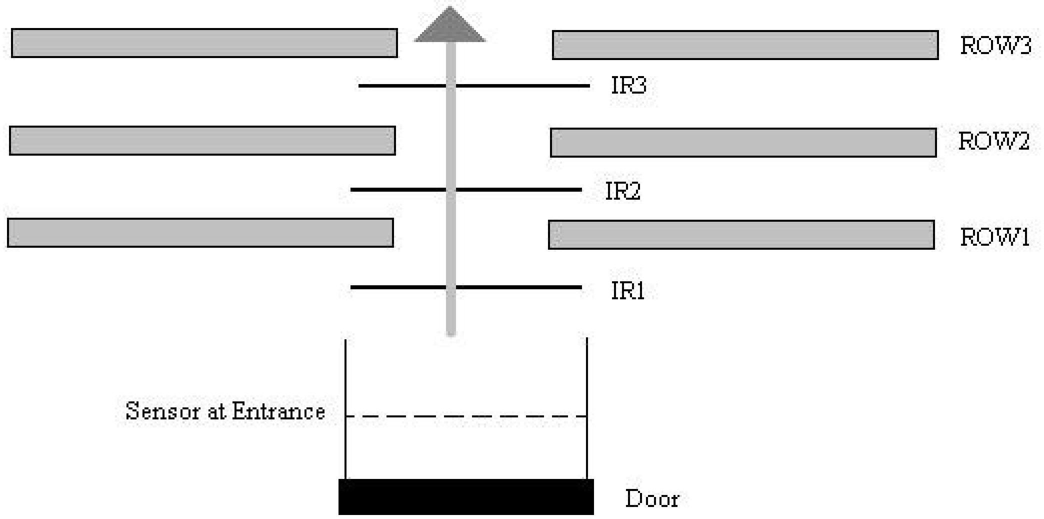
2017A3PS0365P Lakshya Kwatra
2017A3PS0366P Kanishk Singh Raghav
2017A3PS0370P Manit Baser
2017A3PS0372P Raghav Seth

INDEX

PROBLEM STATEMENT

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 row1 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 atleast 6 rows.

System Details:



ASSUMPTIONS

1) There is one main entry to the conference room

2) Only one person enters or exits the room at a time

3) There are 2 sensors on either side of the door

4) There are 2 sensors allotted for each row

5) No 2 sensors are activated at the exact same instant

6) Each row has 3 lights

7) 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

8) Maximum capacity of the conference room is 200

HARDWARE DEVICES

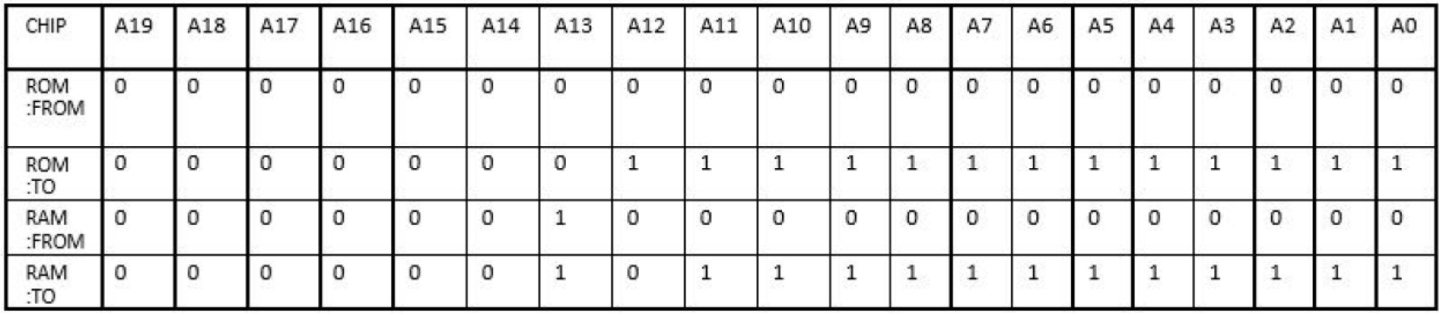
CHIP NUMBER	CHIP	QUANTITY REQUIRED	USE
8086	Microprocessor	1	Central Processing Unit
6116	RAM 2K	2	Random access memory which contains DS,SS
2732	ROM 4K	2	Read only memory which contains entire code (CS)
74LS373	8 Bit Latch	3	To latch address bus
74LS245	8 Bit Buffer	2	To buffer data bus (bidirectional)
74LS138	3:8 Decoder	1	Used for select signals
8255	Programmable Peripheral Interface	2	Input and Output ports
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 (2732): Starting Address: 00000h, Ending Address: 01FFFh
Random Access Memory (6116): 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:

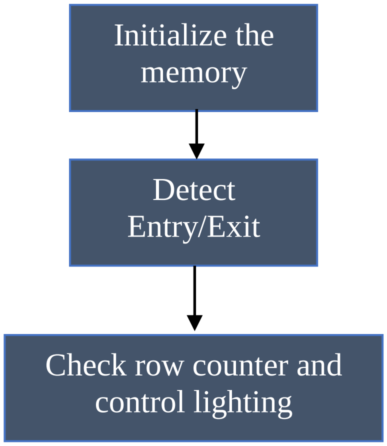
8255-0

Port	Address	Usage
Port A	00000H	Input from Sensors
Port B	00002H	Input from Sensors
Port C	00004H	Output to LEDs
CWR	00006H	Control Register

8255-1

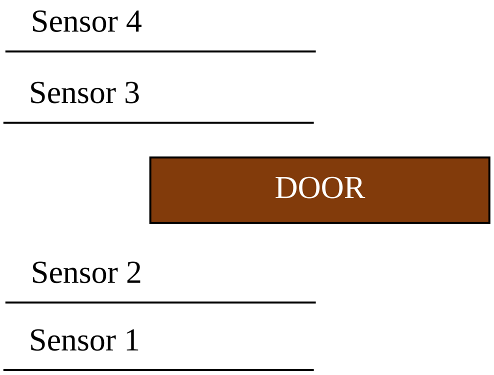
Port	Address	Usage
Port A	00008H	Output To LCD
Port B	0000AH	Unused
Port C	0000CH	Motor Control – Upper as Input, Lower as Output
CWR	0000EH	Control Register

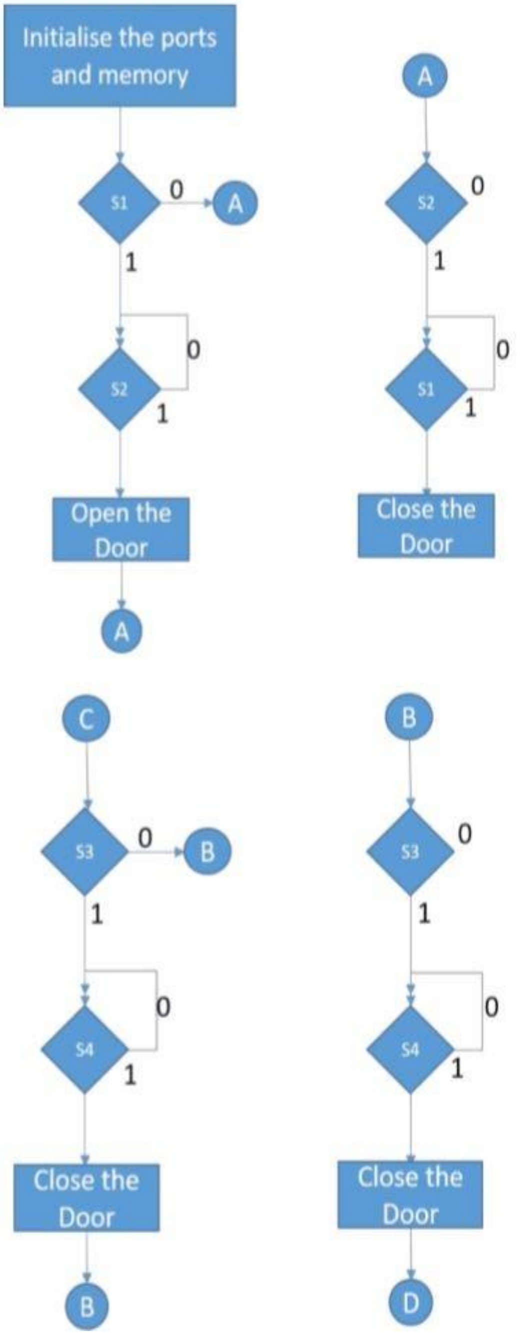
FLOWCHART



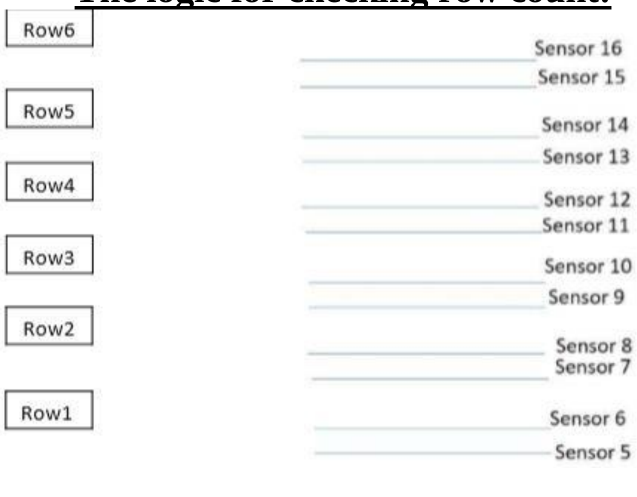
The logic for opening and closing of door:

Step 1: Check sensor 1 for input
Step 2: Check sensor 2 for input
Step 3: Check sensor 3 for input
Step 4: Check sensor 4 for input

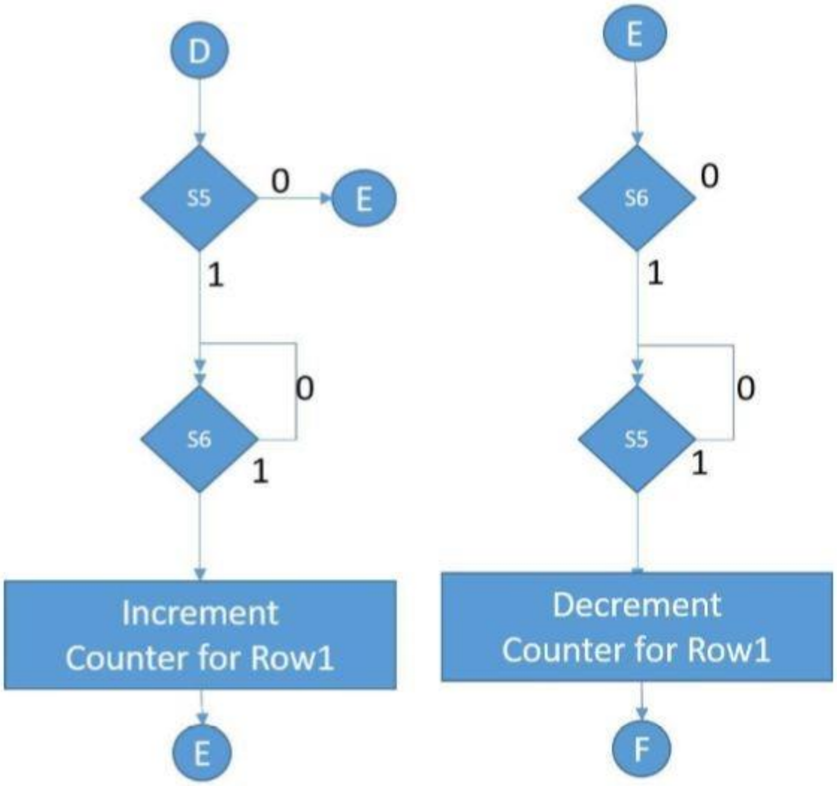




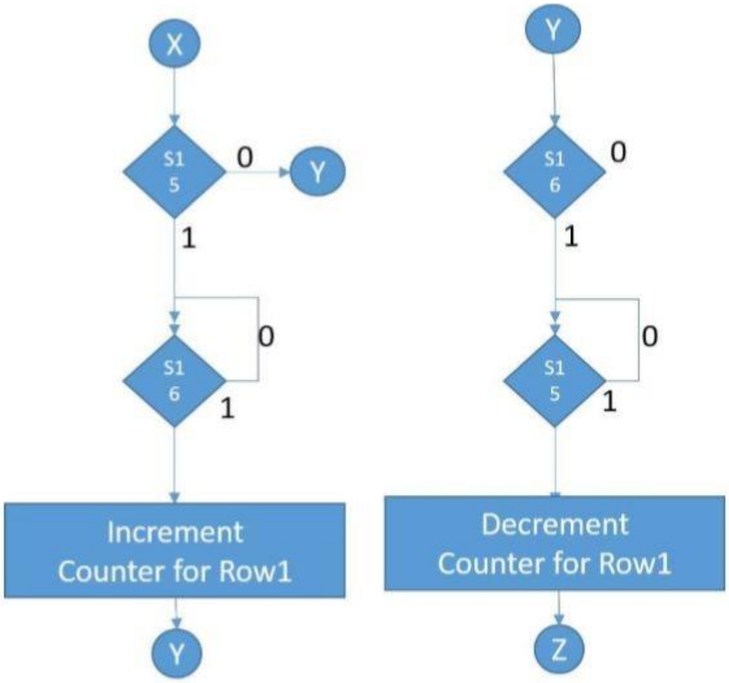
The logic for checking row count:



For Row 1



Similarly for each row… Finally for Row 6



CODE

```
vars:
;MAIN PROGRAM
counter dw 00h
rcounter db 00h,00h,00h,00h,00h,00h
gate db 0
strlen db 0
empty db 'empty'
full db 'full'
;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 1001 dup(0)
st1:
; 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
mov al,10000000b
out creg_io,al
mov al,00h
out 04h,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
call delay_1s
;STOPPING_THE_MOTOR
;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
add counter,1
mov gate,1
motor_clockwise
call delay_1s
stopping_the_motor
call delay_1s
call delay_1s
;;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
sub counter,1
mov gate,0
motor_anticlockwise
call delay_1s
stopping_the_motor
call delay_1s
call delay_1s
call delay
;;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
motor_anticlockwise
call delay_1s
stopping_the_motor
call delay_1s
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
motor_clockwise
call delay_1s
stopping_the_motor
call delay_1s
call delay_1s
call delay
;;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
add rcounter,1
call delay_1s
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
sub rcounter,1
call delay_1s
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
add rcounter+1,1
call delay_1s
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
sub rcounter+1,1
call delay_1s
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
add rcounter+2,1
call delay_1s
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
sub rcounter+2,1
call delay_1s
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
add rcounter+3,1
call delay_1s
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
```

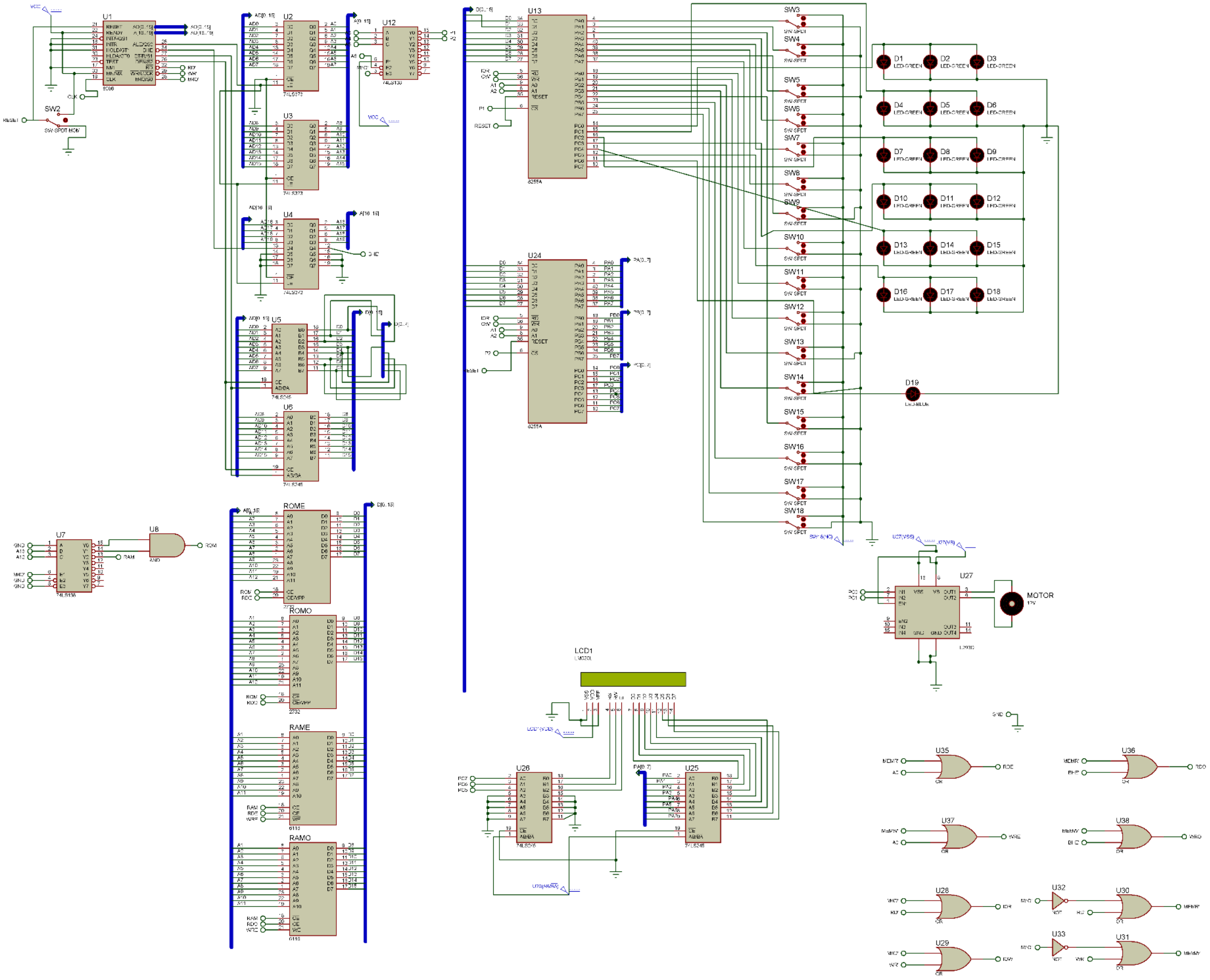
```
cmp al,20h
jnz y12
sub rcounter+3,1
call delay_1s
call delay_1s
;;CHECK FOR ENTRY IN ROW5
x13: in al,00h
and al,08h
cmp al,08h
jnz x14
y13: in al,00h
and al,04h
cmp al,04h
jnz y13
add rcounter+4,1
call delay_1s
call delay_1s
;;CHECK FOR EXIT FROM ROW5
x14: in al,00h
and al,04h
cmp al,04h
jnz x15
y14: in al,00h
and al,08h
cmp al,08h
jnz y14
sub rcounter+4,1
call delay_1s
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
add rcounter+5,1
call delay_1s
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
sub rcounter+5,1
call delay_1s
call delay_1s
;;OUTPUT FOR LEDS IS MADE 1 WHEREVER RCOUNTER IS 1
x: mov al,00000000b
mov bl,00000001b
mov cx,06h
lea si,rcounter
y: mov dl,[si]
cmp dl,00h
jnz z
jmp w
z: or al,bl
w: rol bl,1
add si,1
dec cx
cmp cx,00h
jnz y
out 04h,al
call delay_1s
call delay_1s
call delay_1s
call delay_1s
call delay_1s
call delay_1s
call delay_1s
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
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
call delay
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:
motor_anticlockwise macro
in al, lcd_motor_control
and al, 11111100b
or al, 00000010b
out lcd_motor_control, al
endm
motor_clockwise macro
in al, lcd_motor_control
and al, 11111100b
or al, 00000001b
out lcd_motor_control, al
endm
stopping_the_motor macro
in al, lcd_motor_control
and al, 11111100b
or al, 00000000b
out lcd_motor_control, al
endm
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
```

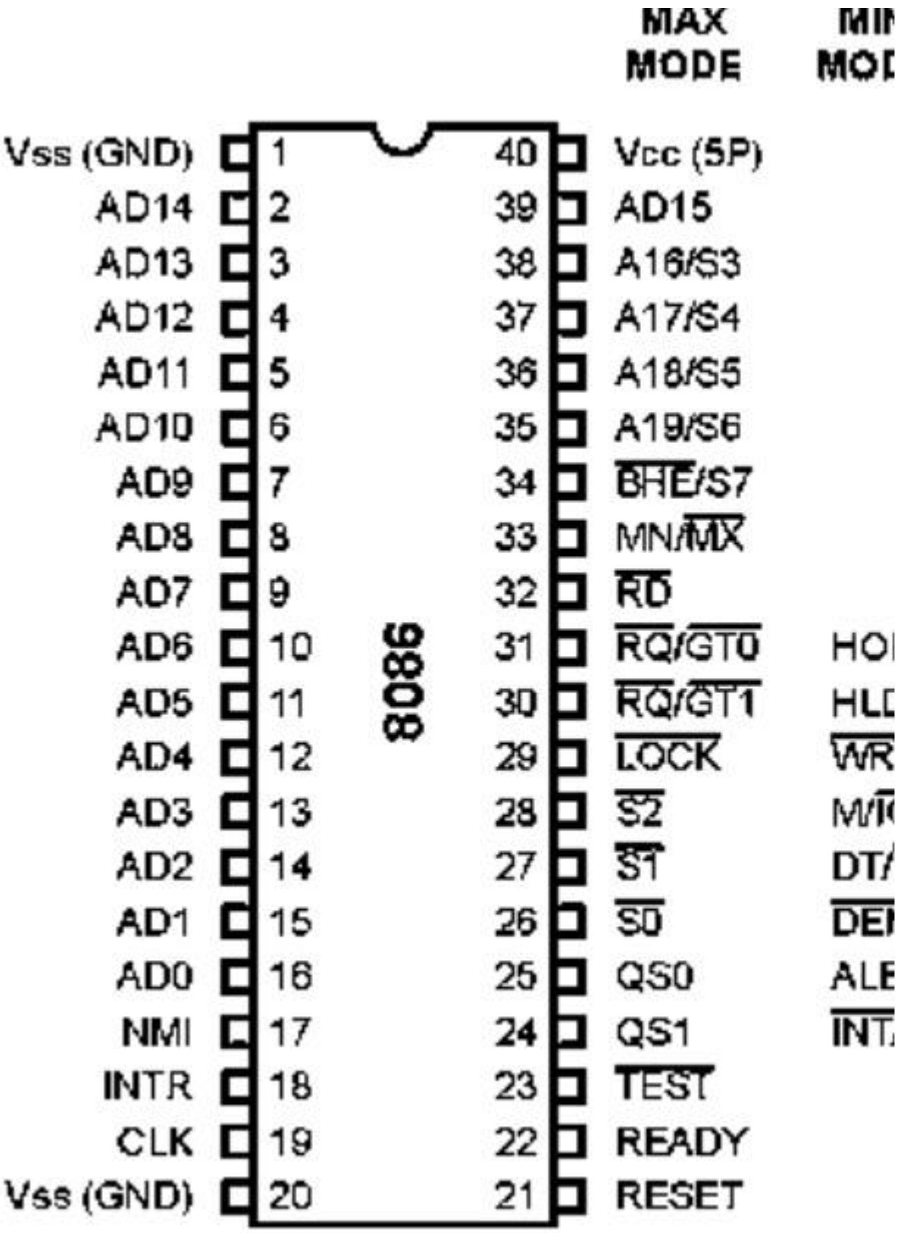
```
pop ax
endm
putstring_on_lcd macro
mov ch,00h
mov cl, strlen
putting:
mov al, [di]
lcd_putch
inc di
loop putting
endm
lcd_bcd macro
mov ax, counter
mov cx, 0
converting:
mov bl, 10
div bl
add ah, '0'
mov bl, ah
mov bh, 0
push bx
inc cx
mov ah, 0
cmp ax, 0
jne converting
printing:
pop ax
lcd_putch
loop printing
endm
procs:
update_the_lcd proc near

lcd_clear
mov al, ' '
lcd_putch
cmp counter,00h
jnz notempty
lea di, empty
mov strlen, 05h
jmp loaded
notempty:
cmp counter,200
jl notfull
lea di, full
mov strlen, 04h
jmp loaded
notfull:
lcd_bcd
ret
loaded:
;call delay_1s
```

```
putstring_on_lcd
call delay_1s
; call delay_1s
ret
update_the_lcd endp
```

CIRCUIT DIAGRAM



REFERENCES



MAX MODE
MI MOD
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
8086
40 Vcc (5P)
39 AD15
38 A16/S3
37 A17/S4
36 A18/S5
35 A19/S6
34 $\overline{BHE}$/S7
33 MN/$\overline{MX}$
32 $\overline{RD}$
31 $\overline{RQ}$/$\overline{GT0}$ HO
30 $\overline{RQ}$/$\overline{GT1}$ HL
29 $\overline{LOCK}$ $\overline{WR}$
28 $\overline{S2}$ M/I
27 $\overline{S1}$ DT/
26 $\overline{S0}$ $\overline{DE}$
25 QS0 ALE
24 QS1 $\overline{INT}$
23 $\overline{TEST}$
22 READY
21 RESET

PIR Sensor Module by Parallax Inc. 55-28027

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.

For more information about this sensor please refer to the file 555-28027-PIR-Sensor-Product-Guide-v2.3.pdf