CSF241: Microprocessors Programming and Interfacing

Elevator Control

Design Project - Group 6

PROJECT REPORT SUBMITTED BY

2018B1A70609G	Taher Lilywala
2019A7PS0044G	Akhilesh Adithya
2019A7PS0131G	Aeishna Khaund
2019A7PS0021G	Arun Ganti
2019A7PS0079G	Harsha Vishak
2019A7PS0111G	Pranav Dargan

Contents:

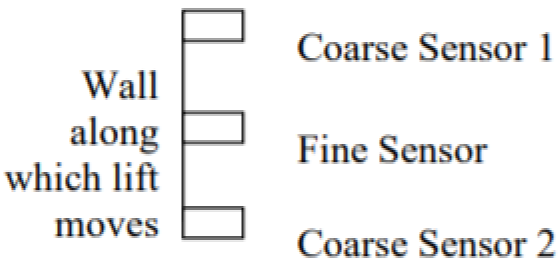
User Requirements & Technical Specifications

System Requirements

- The elevator operates along 4 floors.
- When not in use the elevator is always on the ground floor.
- The elevator can be called by pressing any one of two buttons available on each floor.
- One button is up and the other is down.
- Whether the elevator stops at the floor or not depends on the direction in which the lift moves. For eg. if the lift is moving in upward direction and the person on say the 2nd floor presses the down button; the lift will not stop in the current journey. When the lift reaches the 3rd floor and starts moving down then the lift will stop at the 2nd floor.
- At every floor there is a 7-segment display that indicates the floor in which the lift is right now. The display can be any value from 0 - 3. ‘0’ indicates the ground floor.
- Inside the lift - buttons are available for floor selection.
- The floor towards which the lift is moving is also displayed within the lift.
- Doors to the lift open and close automatically.
- When the lift reaches any floor where it has to stop it opens automatically, and it closes when a button called “Door Close” is pressed. Lift does not move until the door is closed.
- System runs from a standard power inlet.

System Specifications

- An Electro-magnetic system is used for open and close of the door. You just need to provide the on/off control. A heavy duty servo motor is used for lift movement. You just need to provide the input to the driver circuit.
- The inputs are direction (up/down) and a PWM input which controls the speed at which the lift moves. The duty cycle can vary from 20% to 50%.
- The frequency of the PWM signal is 10 Hz.
- For detecting whether the lift has reached a floor, the system has a set of three sensors – two 'coarse' sensors and a 'fine' sensor. All the sensors are contact switches (i.e) when the lift reaches the point where the sensors are placed, the contact switch gets pushed in. Output of contact switches are low when closed and high otherwise. The sensor arrangement is represented in the fig below



- On the ground floor – only Coarse Sensor q and Fine Sensor will be available. On the 3rd floor only Coarse Sensor 2 and the Fine Sensor will be available.
- When the lift starts at the ground floor it starts at a low speed gradually accelerating to the maximum speed. It should operate at maximum speed when it reaches 'Coarse Sensor 1". As the lift moves up if it has to stop at floor '1', when Coarse Sensor 2 is detected at that floor the lift starts moving at a low speed until it can stop when it reaches Fine sensor. When it starts again it moves at low speeds and reaches the maximum possible speed when it reaches the Coarse sensor 2. The same is done in the reverse direction with the appropriate sensors. Speed at which the lift moves is proportional to the duty cycle. For acceleration, duty cycle has to be gradually increased from 20 % to 50 %. And for deceleration, the duty cycle reduced from 50 % to 20 %. The increase is in steps of 10 %

Assumptions and Justifications

Assumption: A 0% duty cycle causes the lift to stop, with 20%, 30%, 40% and 50% allowing the lift to accelerate to a corresponding speed

Justification: The lift needs a set value to stop, which has been assumed to be a permanent off signal

Assumption: A person entering the lift does not take more than 5 seconds to press the floor they want to go to and press the close door button

Justification: There needs to be a fixed amount of delay before the lift moves back to ground floor, or to another floor where a button is pressed

Assumption: Every 7 segment display can be connected to the same output

Justification: The displays show the most recent floor the lift has visited, which is always the same for all the displays.

Components Used

Component	Name	Number used	Justification
8086	Microprocessor	1	-
8284	Clock signal generator	1	-
8253	Programmable Interval Timer		To generate the PWM signal for the motor
8255	Programmable Peripheral Interface Chip	1	To interface the the button inputs and the 7 segment displays
2716	2kBx8 EPROM	4	Smallest ROM chip available is 2K and as we need to have even and odd bank and ROM is required at reset address which is at FFFF0H and 00000H
6116	2kBx8 RAM	2	Smallest RAM chip available is 2 K and we need an odd and even bank. We need RAM for stack and temporary storage of data
74ALS138	1-of-8 Demultiplexer		To select the required I/O device based on the address provided
BUTTON	Single Pole Single Throw Button	21	To take inputs from the user and also to act as the coarse and fine sensors
74ALS245	Octal Bus Transceivers	2	-
74LS373	Octal D-Latch	3	-
74LS138	3-to-8 Decoder	3	For memory and I/O interfacing
74LS47	BCD-to-7 segment Decoder	1	Decoding the value for the display
7SEG-COM-ANODE	7 Segment Common Anode Display	5	To display the floor
SW-SPDT-MOM	Single Pole Double Throw - Momentary Action	1	For reset signal
L293D	Quadruple Half-H Drivers	1	The chip to control the motor by the PWM signal

Address Mapping

Memory Mapping

- ROM1 : 00000_H to $00FFF_H$
- RAM1 : 01000_H to $01FFF_H$
- ROM2 : $FF000_H$ to $FFFFF_H$

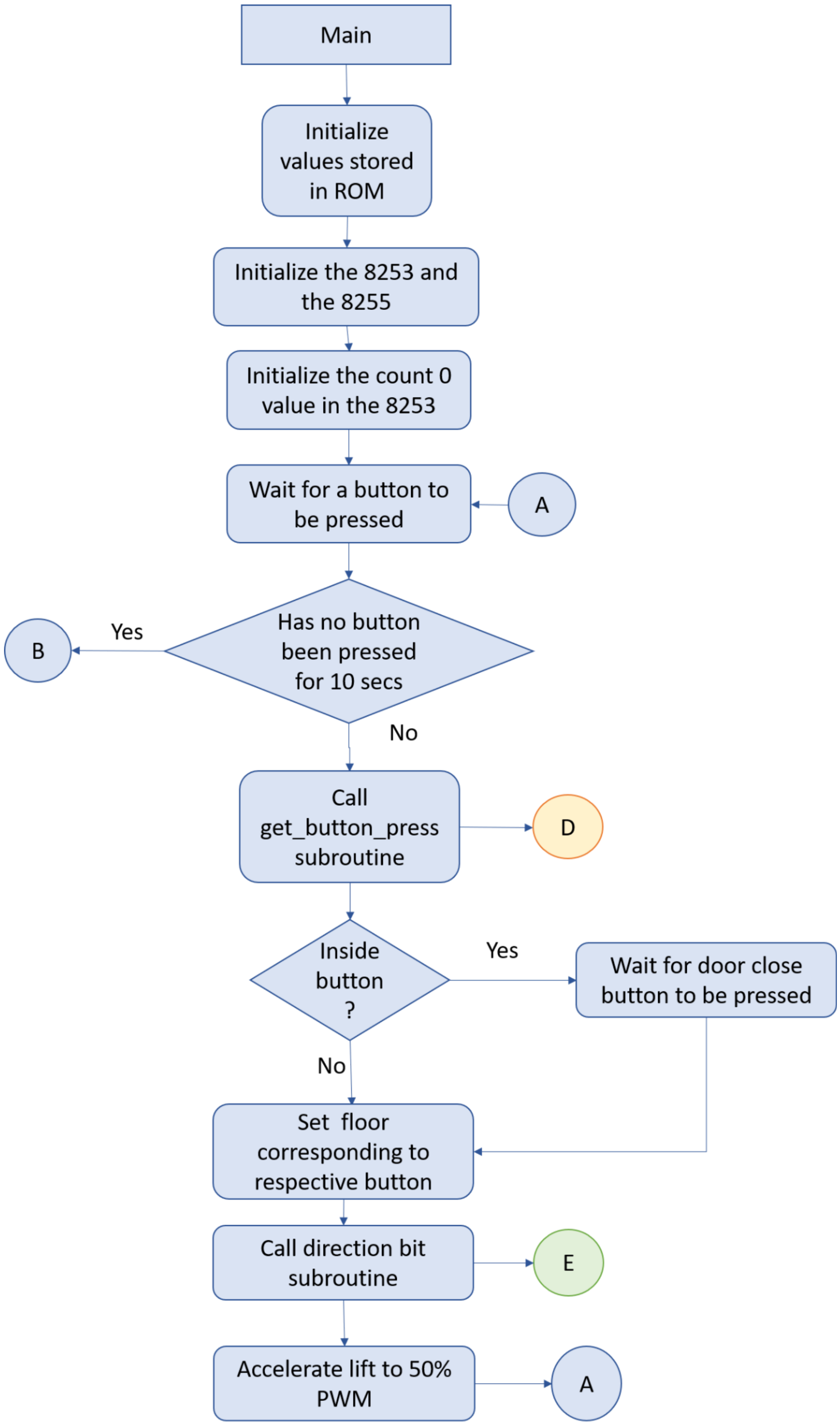
I/O Mapping

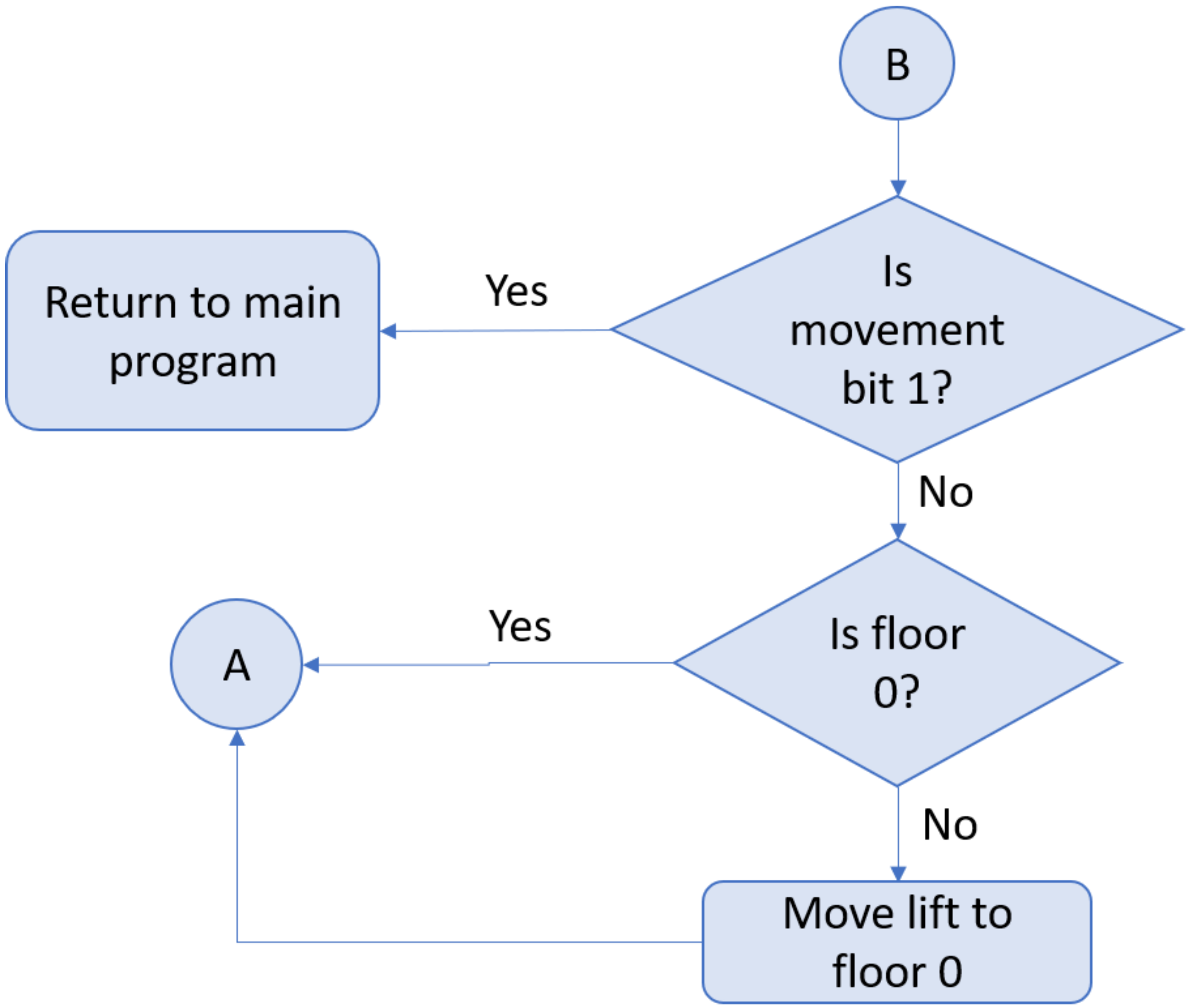
- 8254 : 00_H to 06_H
- 8255 : 08_H to $0E_H$

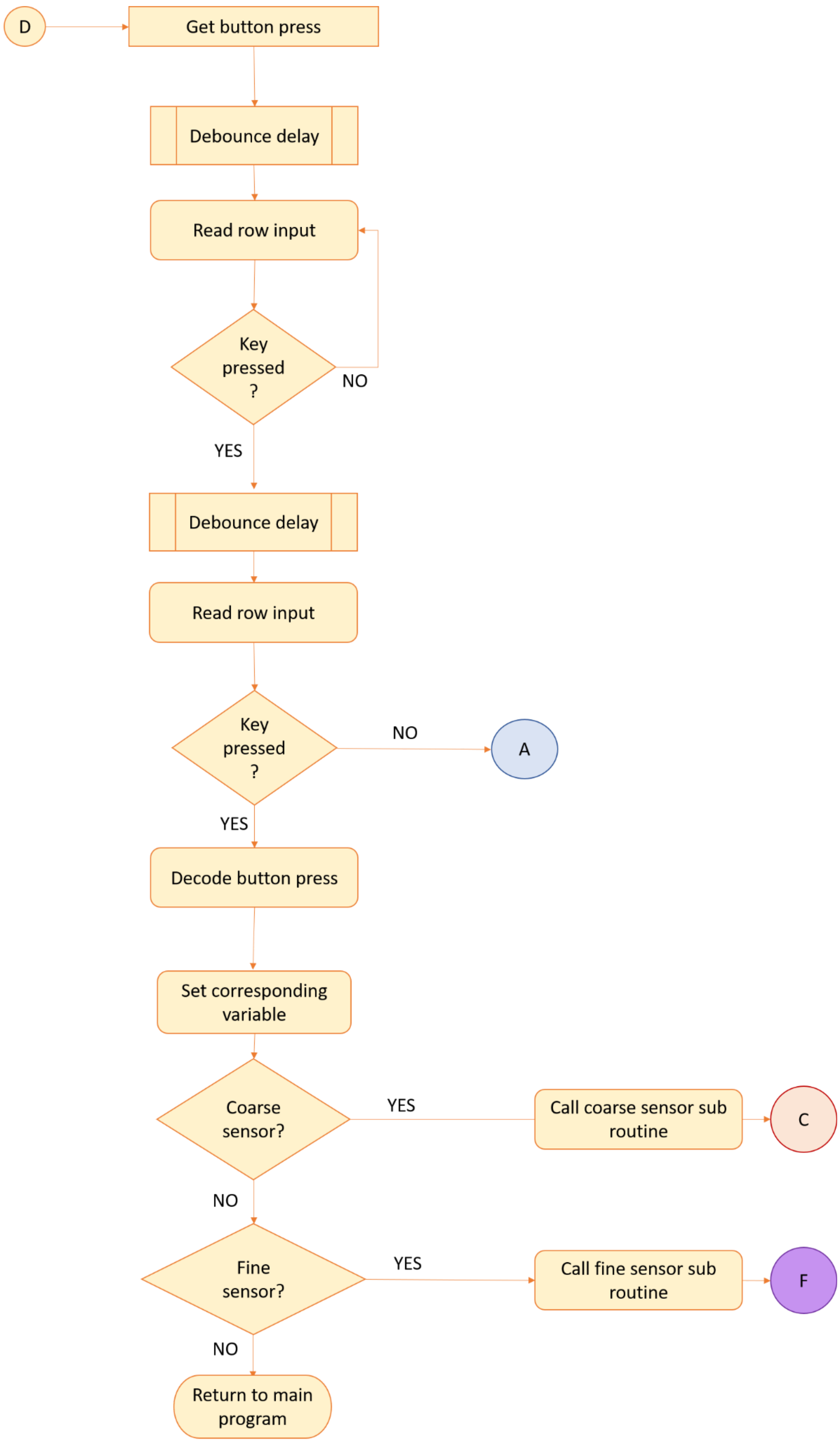
Design

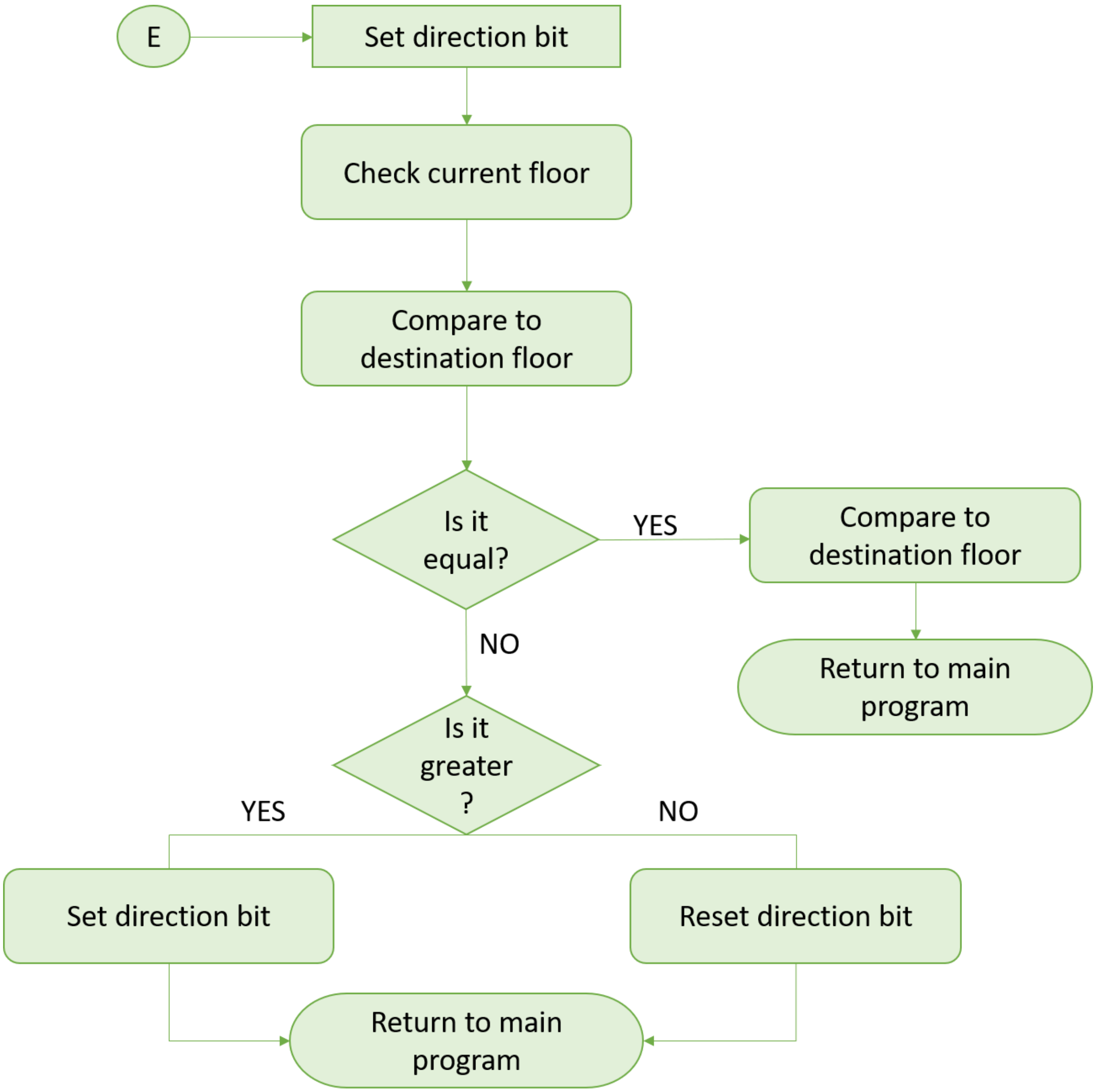
(Elevator control hardware diagrams have been attached as a pdf file)

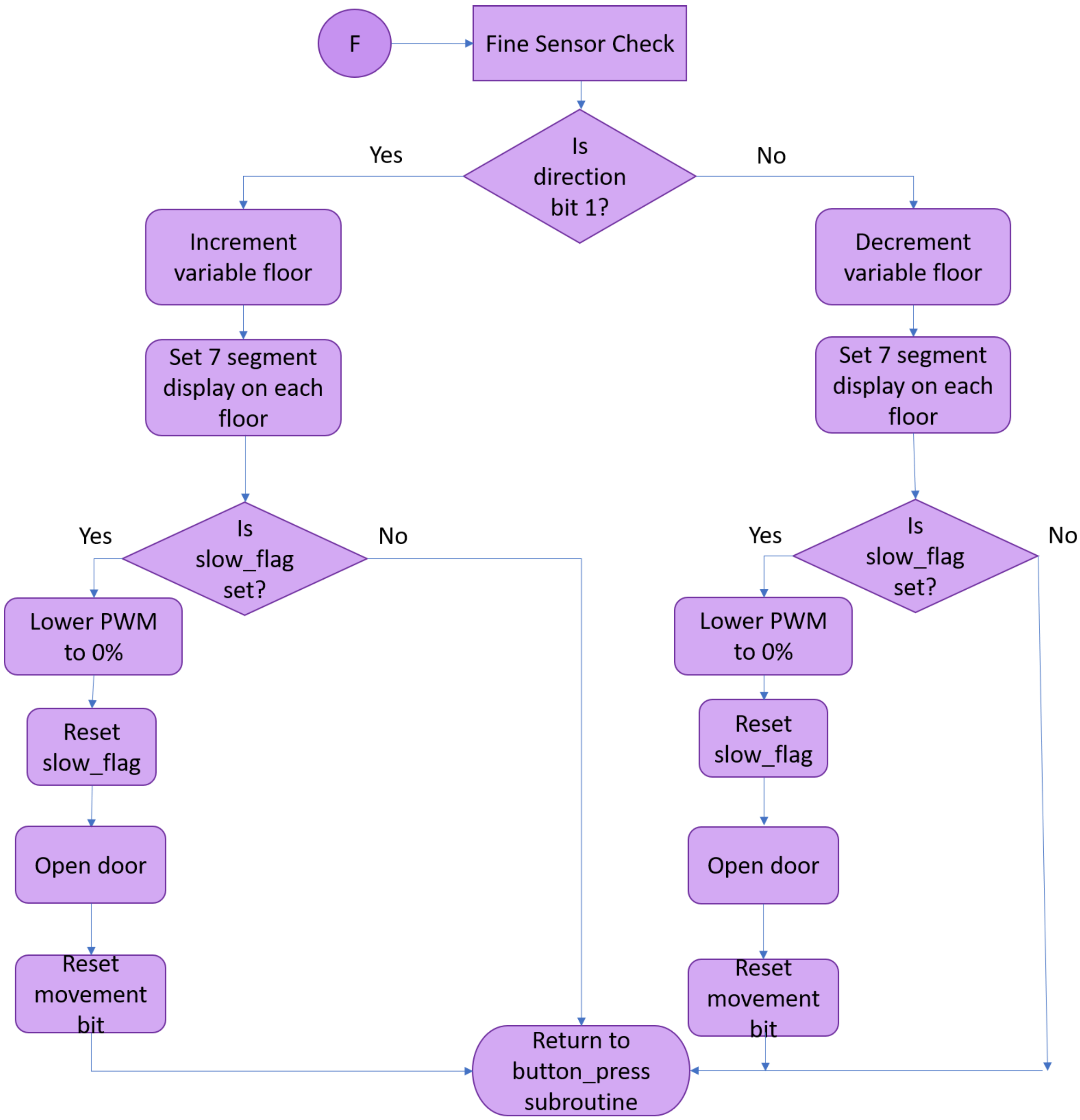
Flowcharts


Main
Initialize values stored in ROM
Initialize the 8253 and the 8255
Initialize the count 0 value in the 8253
Wait for a button to be pressed
A
Has no button been pressed for 10 secs
Yes
B
No
Call get_button_press subroutine
D
Inside button ?
Yes
Wait for door close button to be pressed
No
Set floor corresponding to respective button
Call direction bit subroutine
E
Accelerate lift to 50% PWM
A


B
Is movement bit 1?
Yes
Return to main program
No
Is floor 0?
Yes
A
No
Move lift to floor 0


D
Get button press
Debounce delay
Read row input
Key pressed ?
NO
YES
Debounce delay
Read row input
Key pressed ?
NO
A
YES
Decode button press
Set corresponding variable
Coarse sensor?
YES
Call coarse sensor sub routine
C
NO
Fine sensor?
YES
Call fine sensor sub routine
F
NO
Return to main program

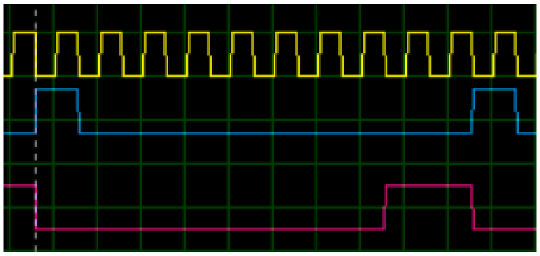

E
Set direction bit
Check current floor
Compare to destination floor
Is it equal?
YES
Compare to destination floor
Return to main program
NO
Is it greater ?
YES
NO
Set direction bit
Reset direction bit
Return to main program


F
Fine Sensor Check
Is direction bit 1?
Yes
No
Increment variable floor
Decrement variable floor
Set 7 segment display on each floor
Set 7 segment display on each floor
Is slow_flag set?
Yes
No
Is slow_flag set?
Yes
No
Lower PWM to 0%
Reset slow_flag
Open door
Reset movement bit
Lower PWM to 0%
Reset slow_flag
Open door
Reset movement bit
Return to button_press subroutine

Explanation of PWM Generation

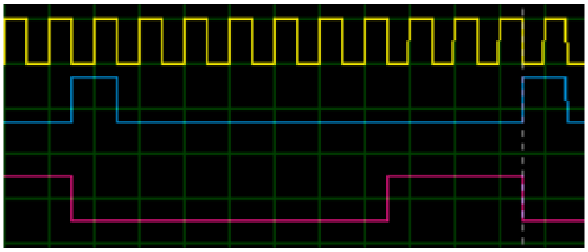
We use two counters of the 8253 chip to generate the PWM signal. We use counter 0 in modes 2 and counter 1 in mode 1. The inverted output of counter 0 is fed as the gate input to counter 1.

Say we want to generate a pulse with a duty cycle of 20%. We provide a count of 9 to counter 0 and of 2 to counter 1. Counter 0 is high for 9 clock cycles and low for 1 and will continue to repeat in this form. The inverted signal is the gate for counter 1, hence after every 10 clock cycles it is reset. The output will be low for 2 cycles and high for 8, giving a duty cycle of 20%.
Below are screenshots of a digital oscilloscope.

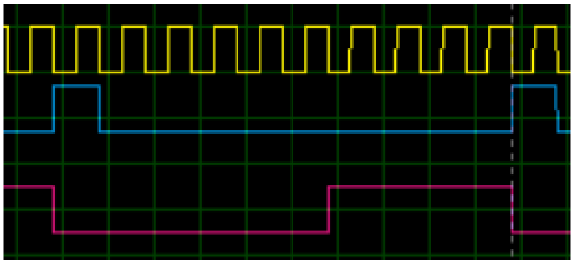


The signal in yellow is the clock signal. In blue is the inverted output of counter 0 and in red is the output of counter 1 which has a duty cycle of 80%. Hence for each case we set the count 0 value as 9. Varying the value of the counter from 2 to 5 depending on the required duty cycle.

30%
Count as 7



40%
Count as 6



Variations in Proteus Implementation with Justification

1. 8284 chip model not available in Proteus so the pulse generator is used
2. Using 2732 (4k) chips instead of 2716 (2k) chips for ROM
3. 8253 used instead of 8254, as 8254 is not available.

Firmware

(Code.asm file has been attached)

List of Attachments

1. Proteus file (DSN file)
2. EMU 8086 ASM file (ASM file)
3. EMU 8086 BIN file (BIN file)
4. Hardware Design (PDF file)
5. Datasheets
    a. 7447 (PDF)
    b. L293D (PDF)