University of California at Berkeley
College of Engineering
Department of Electrical Engineering and Computer Science

EECS 150 R. H. Katz
Spring 2001

FINAL EXAMINATION
Friday, 11 May 2001

INSTRUCTIONS—READ THEM NOW! All work is to be done in your blue books. Print and sign your name and Student ID Number on the cover of each book you use. Partial credit is given only if we can evaluate your approach: indicate your assumptions and ***write as neatly*** as possible. The examination is worth 150 points. You have three hours to complete your work. Spend 20-30 minutes thinking about the problem specification before you commence. This is an open note/open book examination.

It is a sad fact of life that cheating happens. It will not be tolerated. By signing your blue book, you assert that all of the work included therein is your own, and that you understand the harsh penalties that will be imposed should cheating be detected—a 0 on the examination, and a letter of reprimand to your file:

QUESTION	**POINTS ASSIGNED**
1	30
2	30
3	50
4	30
5	10
TOTAL	150

Design Specification: READ CAREFULLY! Please send 20-30 minutes studying the specification. This is factored into the point totals for the questions (they add up to 150 points rather than 180).

Professor Katz has always been fascinated by the things he meets in everyday life that never seem to operate as expected. It is difficult to understand why it is so hard to get the behavior of these things right until you try to design one yourself.

Motivated by the brain-dead elevators in Soda Hall, your task--should you agree to accept it, Mr. Phelps--is to design a sophisticated elevator controller. You will start with a single "conventional" elevator (i.e., simple up/down buttons to call the elevator) operating in a four-floor building. Then you will extend your design to two elevators. You might keep this in mind while working on the initial design.

The specification for the basic single elevator design is described in the sections below. The design questions are on the final two pages.

Passenger Interface: Lobby and Elevator Buttons and Lights

A passenger sees the following buttons and indicators. On each floor, there are up and down push buttons. When pressed, the button illuminates. When the elevator arrives at the floor in the indicated direction, the button's light is turned off. The floor the elevator most recently passed is also displayed. In addition, up and down arrows are illuminated. These display the current direction that the elevator is moving.

Inside the elevator, the passenger sees the four floor buttons. When pressed, these become illuminated. They stay lit until the elevator controller resets them. This happens when it stops on the indicated floor. Normally the doors stay open for 30 seconds. In addition, the elevator has a "door open" button, which keeps them open for an extra 10 seconds when pressed, but never more than 30 seconds. It also has a "door close" button, which causes them to close within 10 seconds, or sooner if less than 10 seconds remains.

More specifically, the buttons are:

- *Elevator Lobby (one per floor)*: Floors 2 and 3 of the building have up and down buttons. Floor 1 only has an up button while Floor 4 only a down button. These are identified as U_i and D_i, where i is the floor. When a button is pressed, it is illuminated, until the elevator arrives at the floor heading in the desired direction. Each floor has up and down lighted arrows to indicate the current travel direction of elevator, denoted LU_i and LD_i for floor i. Each floor also has a display, CF_i, indicating the last floor through which the elevator traveled.

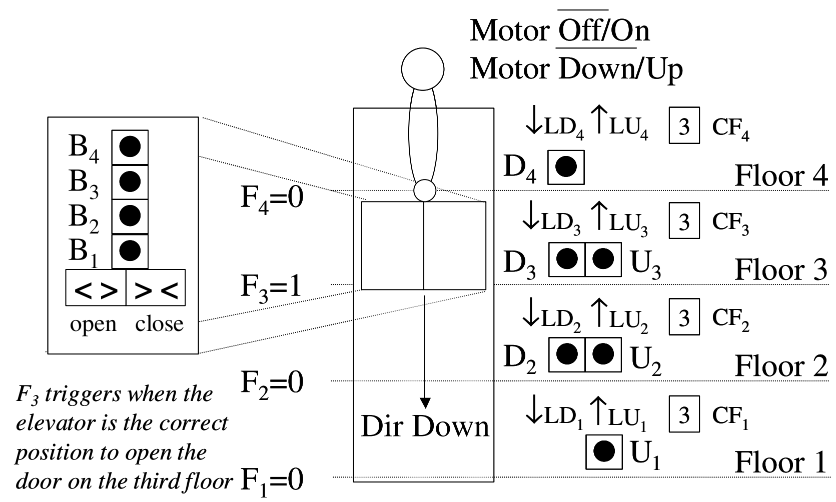
- *Inside the Elevator*: There are four floor buttons, denoted B_j. A passenger presses a button to request that the elevator stop at the indicated floor. The button is illuminated until it arrives at the given floor, when it is to be reset. In addition, there are *OpenDoor* and *CloseDoor* buttons. These are not illuminated when pressed. *OpenDoor* causes the door to remain open for an extra 10 seconds, but never more than 30 seconds if there are more than 20 seconds left on its "open" timer. *CloseDoor* reduces the open interval to 10 seconds, unless the remaining time is already less than 10 seconds.

See the figure at the top of the next page.

Information to Track/Initial Assumptions

- Electronic sensors are located on each of the four floors, denoted F_i. These become true just when the elevator reaches the floor and is in the correct position to open its doors. Don't worry about the precision of these sensors. *Dir* indicates the direction of the elevator: 0 is down and 1 is up.

- On reset, the elevator moves to the first floor (the controller knows this to be the case when F_1 becomes true), opens its doors, and the default direction is up. All buttons are initially reset/unlit. The elevator stays on Floor 1 with its doors open until someone calls the elevator to another floor (the doors should close after 30 seconds if nobody gets on).



F_3 triggers when the elevator is the correct position to open the door on the third floor

Sample Scenario to Illustrate the Elevator Behavior

The elevator is initially on the first floor with its doors open. The CF_i lights on all floors display a 1 and LU_i lights are illuminated. Alice presses U_1, and she steps inside. She wants to go to Floor 4. She presses B_4, which causes that button to be illuminated. At about the same time, Bob on the second floor presses D_2. The elevator starts to move up. As the elevator passes through the second floor, it does not stop because he wants to go down. As the elevator approaches the third floor, Carol presses U_3.

The way this elevator works, it sweeps as far as it can in the current direction until it has satisfied all floor requests. Then it changes directions and does the same. Requests come from each floor as well as from inside the elevator. The elevator knows it must visit Floor 2 (Bob is waiting to go down), Floor 3 (Carol is waiting to go up) and Floor 4 (the is Alice's destination). However, since it is on an upward sweep, it will only stop at the third and fourth floors.

When it reaches the third floor (indicated by F_3 becoming true), it stops, opens its doors, and waits for thirty seconds before closing them. Carol gets on, pressing B_4. The elevator continues on to the fourth floor (it knows it has arrived when F_4 becomes true). It opens its doors and waits 30 seconds while Alice and Carol get off.

The elevator changes its direction to down. It has one call now pending at Floor 2. Meanwhile Dave has arrived at Floor 3 and presses U_3. The elevator won't stop on the third floor because its direction is now down, and Dave wants to go up.

It moves to the second floor and stops when F_2 becomes true. It opens its door for 30 seconds, Bob gets on, and presses B_1. The doors close, and the elevator moves on the first floor. F_1 becomes true, the doors open, and, Bob gets off.

The elevator again changes its direction, this time to up. Now it recognizes that there is a pending "up" request on the third floor. After 30 seconds, the doors close, and the elevator moves to the third floor to pick up Dave.

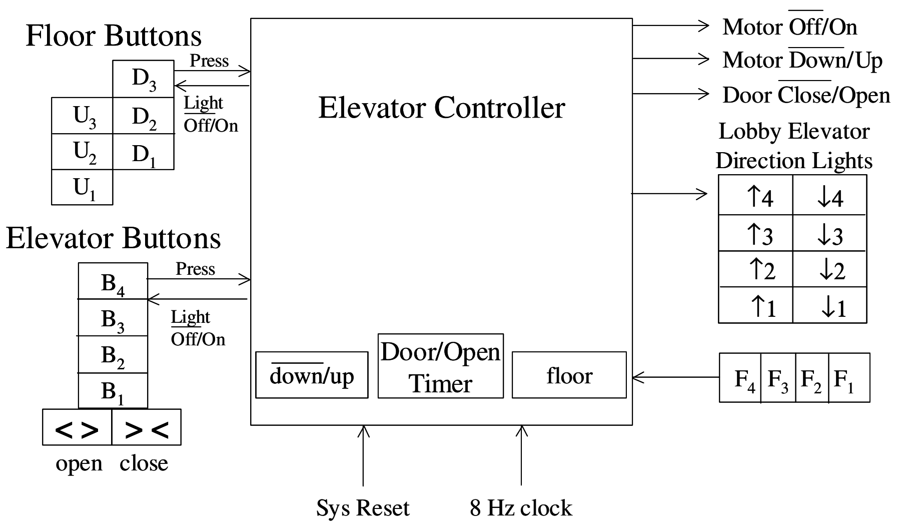
What if Bob got on with the elevator heading down, but pressed B_3 instead of B_1? In this case, the doors would close, the elevator would detect that it has no request to go anywhere below its current floor. It would then change its direction to up. Depending on your design, the doors may open again (remember that U_2 is still pressed at the point in time the elevator changes direction), and Bob may have to press B_3 again. But what does he expect wanting to go up on the down elevator!

What if Alice had wanted to get off on the third floor and Carol wanted to get on to the elevator on the fourth floor and go down? After Alice gets off, the elevator detects that there are no more "up" requests above its current floor. The right strategy is for the elevator to change its direction to down, but before it does that, we MUST proceed to the highest floor for which there is a pending down request. Thus it moves up to the fourth floor, opens its doors, changes the direction lights to down, and internally changes its direction to down. The same thing MUST happen when it changes from down from up. It should find the lowest floor with a pending up request, and move down to that floor before changing its direction to up.

Observations and Specifications Details

- Your controller must track the direction of the elevator.
- Your controller must track the pending elevator stops as indicated by calls for the elevators on the various floors ("elevator calls") as well as button presses inside the elevator ("elevator stops").
- Assume passengers will do silly things, like want to go up in an elevator moving down. Do something reasonable in theses cases (for example, ignore their request!).
- The elevator MUST FOLLOW a sweep algorithm. That is, it should process all calls and stops in one direction before turning around. However, consider the end cases when the elevator changes its direction. In particular, before it turns around, it has to continue in the same direction to the highest (going up to a request to go down) or lowest floor (going down for a request to go up) for which there is a pending request in the opposite direction.

The block diagram for the Basic Controller is shown below. It shows the input buttons on the left, which the controller can selectively reset. The system clock is 8 cycles per second, that is the clock period is 125 ms. The controller can assert the up/down lights per floor, can move the elevator up and down but starting the motor in the desired direction, and can open or close the elevator door.



Once you design the basic controller, you will extend it to two elevators. Your solution must avoid the annoying attribute of sending both elevators to service the same elevator call. In addition, you will add two inputs: MORNING (AM) and AFTERNOON (PM). These are never asserted at the same time. When AM is true, the controller should make sure that an elevator returns to the first floor whenever it has been idle for two minutes. When PM is true, the controller stages an idle elevator, defined as not having moved in two minutes, to either the first or fourth floor. It should never be the case that both elevators go to the same floor when idle.

Question 1. *Basic Elevator Controller High Level Description* (30 Points)

(i) To make sure that you understand the function of the basic elevator controller, write pseudocode that describes its behavior. Declare your variables and use the inputs and outputs as defined in the specification. Use a reasonable and consistent program syntax. Neatness counts!

(ii) Provide an annotated trace of the execution of your pseudocode, using as input the Alice-Bob-Carol-Dave scenario described on Page 3.

Question 2. *Design of Datapath Components* (30 Points)

(i) Design a button that when pressed for the first time will illuminate a light until subsequently reset by an external LIGHT RESET signal. (5 Points)

(ii) Design a counter subsystem that asserts "DONE" when it has counted 30 seconds since an input signal "START" has been asserted. If a "CLOSE" signal is asserted, the subsystem reduces the time left to indicate DONE by 10 seconds, unless there are less than 10 seconds left. If an "OPEN" signal is asserted, then DONE is delayed by a further 10 seconds, but never more than 30 seconds. CLOSE and OPEN are debounced signals with resets—you may assume that higher levels of control deal with repetitive pressing of the OPEN and CLOSE buttons. (15 Points)

(iii) Design a Prioritizer circuit with three data inputs, I_1, I_2, I_3, three outputs, O_1, O_2, O_3, and a control input Min/Max that works as follows. At any time at most one of the three outputs is asserted. When Min is asserted, if I_1 is true then O_1 is asserted. If I_1 is false but I_2 is true, then O_2 is asserted. If I_1 and I_2 are both false but I_3 is true, then O_3 is asserted. In other words, the output associated with the input that has the lowest index that is true will be true. With Max asserted, the outputs are defined analogously: the input with the highest index that is true will be true. Briefly describe how such a prioritizer can be used inside the elevator controller. Yes it is useful! (10 Points)

Question 3. *Basic Elevator Controller State Machine* (50 Points)

Organize your state machine into three sub-FSMs that communicate with each other: (1) the User Interface FSM that deals with the button presses and manages the button illuminations and displays, (2) the Elevator Move FSM that stops at an indicated floor and manages the opening and closing of the door, and (3) the Main FSM that determines, based on the inputs, where to send the elevator next.

(i) First, describe the method by which your FSMs will communicate with each other. What information needs to be exchanged between them to achieve the desired behavior?

(ii) Second, what registers or other kinds of datapath components are needed to make your decomposed FSM operate correctly?

(iii) Finally, provide complete state diagrams for each of the three FSMs implemented as Moore Machines. In each state, indicate in English what detailed control operations take place.

Question 4. *Extend Your Controller to Two Elevators* (30 Points)

(i) Identify how the controller inputs and outputs have to change to manage two elevators.

(ii) Partition your User Interface FSM into three communicating state machines: one for the Lobby I/Os, and one for each of the Elevator I/Os. Describe the necessary changes to your UI FSM. What are the needed changes to how this part of your design interfaces to the Main FSM?

(iii) Describe in English at a high level (or easy-to-understand pseudocode) your algorithm for managing the movement of the two-elevator system as described at the bottom of Page 4. Namely, (a) you must avoid sending both elevators to the same floor in response to an elevator call from that floor; (b) when AM is true, as each elevator becomes idle (it hasn't moved in two minutes), it should be sent to the first floor; and (c) when PM is true, as an elevator becomes idle, it moves to either the first or fourth floors. For Part (c), if one elevator is already on or moving to the first floor, the idle elevator should be directed to the fourth floor. The case is similar if there is an elevator on or moving to the fourth floor; the idle elevator is sent to the first floor. If the above rules don't apply, then an idle elevator on the second floor has a preference to move to the first floor. An idle elevator on the third floor has a preference to move to the fourth floor. You must avoid a design that sends two idle elevators to the first or fourth floors when PM is true.

(iv) Describe the necessary changes to your Main FSM to implement your algorithm in Part (iii), by showing the parts of the state diagram that need to change to accommodate the two elevators and their constrained behavior as described in Part (iii).

Question 5. *Implementation Methods* (10 Points)
Discuss how you would use a microprogrammed approach to implement your state diagram for Question 4, Part (iv)—the revised Main FSM for the two elevator system. Draw a block diagram of the controller organization, that is, the structure of the ROM, multiplexers, and state register. Describe how your controller implementation determines the next state: how it is encoded, how alternative next states are represented in the control ROM, and the method by which the correct next state is selected.