In yet another attempt to turn the university into a world-class construction pork center, state officials have announced the construction of a horizontal transportation system at UConn to be named the “Husky Interbuilding Transportation” system (colloquially referred to as “HIT”). This system is small-capacity above-ground monorail that will have stops at the North Parking Garage, Engineering II, the ITE building, and the UConn Coop. In related news, the subcontract (described as “not necessarily illegal” by sources close to the Governor) for the control system has been awarded to Robertics, Inc., whose motto “If you can’t afford the best, you could do worse” has raised a few eyebrows. It is rumored that the bid selection was based on free construction work done by Robertics, Inc., but the CEO has denied this (“That construction work was just a gift between friends. We were the low bid on this project—after all, we are using unpaid student labor.”).

The figure below illustrates the path through the campus.

![HIT layout diagram](image)

Figure 1: HIT layout.

Generally speaking, the control system is responsible for moving the vehicle (or “pod”) from stop to stop based on the requests of the passenger or potential passengers, and for opening and closing the door at appropriate times. The pod responds to one request at a time: if there is a passenger on board who has requested a location, it goes to that location and lets the passenger out. If there is no passenger who has requested a location, and there has been a request for service at one of the stations, then the pod goes to the requesting location and opens its door so the requestor can get on board. Naturally, the system should only move with its door shut.

Users request service via small paper cards with a magnetic strip; each card is encoded with a single location on the route. When a user puts a card in the slot inside the pod, the pod 1) closes the door, 2) drives the vehicle to the appropriate station, 3) turns on the “at station” light, and 4) opens the door so the user can disembark when the user removes the card from the reader. A user requests service at any station by inserting a card into the station card reader. The car responds (if it does not already have a passenger) by driving to the location of the request, then opening its door. When the passenger gets in and inserts his or her card, then the pod behaves as above.
As a safety feature, there is a 3 second delay between the system signalling the door to close and the door actually closing (A similar delay takes place when signalling the door to open.).

Your job is to build a control circuit that has the following characteristics:

1. The vehicle follows the internal request if there is one,
2. The vehicle follows the external request if there is no internal request,
3. The vehicle does not move when the door is not closed, and
4. All vehicle movement should be in the direction of the currently active request.

The Stations and Sensory Information

The four locations each are referred to by a 2-bit ID: 00 for the parking lot, 01 for E-II, 10 for ITE, and 11 for the COOP. These locations are encoded on the request cards (so if a card is in the reader, the reader provides that value as input to your system). These IDs are used in other data as well: the NEXT_STATION signals are IDs, as is the SR_LOCATION signal (more about these below). The other signal generated by the position of the vehicle is the AT_STATION signal, true if the vehicle is at a station (in position to open door).

The Inputs

The inputs to the control system are:

- **DOOR_CLOSED** true when door fully closed,
- **AT_STATION** true when vehicle at a station,
- **SERVICE_REQUESTED** true when someone is requesting service from a station,
- **SR_LOCATION** the ID of the station where service is being requested (if more than one station has a request, the one that has been requesting longest is the value provided),
- **CARD_IN_READER** true if there is a card in the internal reader, i.e. there is an internal request,
- **IR_LOCATION** ID of destination requested by card in internal reader,
- **NEXT_STATION_FWD** ID of next station in the forward direction (toward Gampel); if at a station, the ID of the current location,
- **NEXT_STATION_BWD** ID of next station in the backward direction (toward Parking Garage); if at a station, the ID of the current location

The Outputs

You are to provide the following outputs.

- **DRIVE** When true, causes motors to run, driving vehicle.
- **FWD/BACK** When true, DRIVE causes motion in forward direction; when false, DRIVE causes motion in backward direction.
- **OPEN/CLOSE_DOOR** When true, causes door to open; when false, causes door to close, and
- **DESTINATION_LIGHT** turns on the ’at destination’ light when it is time for passenger to disembark.
Behavioral concerns

How will it work? What if the SR_LOCATION given was that of the closest requesting station?

What about security? Suppose a person with evil intent boarded the system and removed his or her card. What would this mean for the next rider? Any ideas on how this might be dealt with?

For extra credit, try one or more of these

- Convert system to a loop arrangement, so the pod moves toward the destination in closest direction (direction with fewest intervening stations).

- Suppose we try to have more passengers: if a car with passengers is going past a station with a request to go in the same place as the car is already going, then it should stop and allow these passengers on. This would require more input, which you can specify.

Deadlines

1. 3 February Descriptive: You should have decided on all of the discrete steps necessary to complete the whole design. You should list the steps to indicate your design philosophy and present a 1 page (or so) description of your solution with appropriate simple diagrams. This description will allow your TA to help you and provide feedback for the next step. This report is required to be turned in but will not otherwise be graded.

2. 10 February Functional. By now you should have all of the logical details sorted out. You should know how to implement all of the steps listed on the initial document. You should be able to show how the problem is decomposed into small sections. You should also support this implementation with logical formulas and truth tables. The primary reason for this stage of the design is for you to get feedback so it will be weighted lightly.

3. 17 February Complete. Your report should contain, at least: a title page with your name, course number, section, and date, objectives, procedure or strategy for solving the problem, functional description of decomposed blocks, diagrams and circuits, test cases and results, discussion, conclusions. In addition to a hard copy, you will submit (via email or on a disk), a zipped set of files corresponding to the report plus any LogicWorks circuits. See “Project Information” at the web page for more information. We give you a grade sheet corresponding to these items before the project is due.

For any help, ask Robert or Gamal. Email is fairly effective, but do not always expect an immediate response, especially close to deadlines or early in the morning.