Dalhousie University
Department of Electrical and Computer Engineering

ECED 4402 – Real Time Systems

Assignment 3: Real Time Programming

1 Objectives

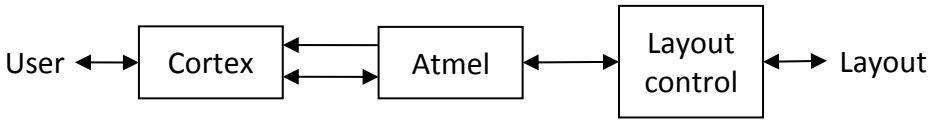
In this assignment, two "trains" are to run on the train set. The trains are to operate on the same loop of track, running in the same or opposite directions. Your train-control software must be written so that the two trains never collide and, when necessary, the "local train" must make way for the "express train".

A screen map, showing the train's state (location) and the state of the track (switches), is to be maintained and displayed on the user's computer.

This assignment uses the software developed in the previous assignments. It also requires the development of software to control the train set, to handle inter-machine communications, and, possibly, software to support a virtual train machine.

2 Physical Components

The system consists of the Cortex processor (the controller), an Atmel (the trainset), and proprietary layout control hardware. The components are structured as follows:



The Cortex (Tiva) sends and receives train set control signals via the second UART (UART 1).

The layout consists of two *sidings* and two *passing loops*. There are six *switches* (a switch is 'Y' in shape, with a straight section, A-B, and a curved section, A-C); each switch is in one of two states: *straight* (the locomotive passes from A-to-B or vice versa) or *diverted* (the locomotive passes from A-to-C or vice versa). The track has 24 Hall Effect sensors that indicate the location of a train. Each locomotive is associated with a *magnitude* and a *direction*.

Layout state and control information messages are exchanged between the Cortex and the Atmel in frames carrying data or acknowledgement packets. Data packets carry messages referring to:

Locomotive location. A locomotive's location on the layout is detected by the magnet on the locomotive triggering one of the 24 Hall sensors. The sensor number is queued by the trainset and sent to the controller in a message. The controller must acknowledge its receipt of the Hall sensor message to allow the trainset to reset the Hall sensor.

Locomotive speed and direction. The magnitude (speed) and direction of each locomotive is specified by the controller.

Switch state. Switch-throw messages are sent from the controller to the trainset when a switch is to be thrown to straight or diverted. After a switch is thrown, the trainset is to respond with a switch-thrown acknowledgement message.

Details of the trainset protocol are available on the course website.

3 Assignment overview

In this assignment, two trains, a fast express and a slow local, are to run on the trainset. The express (train 1) is to start on siding 23-24 (the number pairs refer to the Hall sensors between which the locomotive is to be placed), move onto the mainline, and then travel in a clockwise fashion around the layout without visiting the sidings or passing loops. The local (train 2) is to start between sensors 19-20 and visit siding 23-24, passing loop 17-18, siding 23-24 (again), and then return to passing-loop 19-20. It must stop on each passing loop or siding it visits.

The express can reduce its speed, but it should attempt maintain an average loop-speed of 5 and must remain on the mainline running in a clockwise direction only. The local just chugs along as best it can, averaging a loop-speed of 2.

3.1 Startup

The trains are to be placed on the layout in their starting sections of track (see above). The user is to inform the system of the initial configuration by entering data from the system window; for example:

Enter starting sections and speeds:
6 7 7 4 5 6

The first two values indicate where the first train is to be placed (between sensors 6 and 7), the third value, and its maximum speed (7 units). The second grouping of three indicates where the second train is to be placed (between sensors 4 and 5) and its maximum speed (6 units).

The maximum allowable speed for a locomotive travelling around the loop is the speed value supplied from the command line. A locomotive can stop itself, reverse its direction, and enter a siding.

3.2 Sections

You may want to implement some form of virtual structure over the layout, dividing it into (virtual) sections. This would allow a degree of control over when a train can safely enter a section without colliding with another. The next three subsections (3.2.1 to 3.2.3) were used in the previous offering of the course when the layout had physical sections.

Alternatively, you may simply institute a rule that ensures there are at least *N* Hall sensors between the front of the "following" train and the end of the "leading" train. Some form of handshaking between the train processes will probably be necessary.

3.2.1 Look-ahead

Before a locomotive can enter a section, it must know the number of the section in question. This can be achieved in a number of ways—a clean, simple technique is to use a look-ahead or

adjacency matrix (in this case, a vector). Each entry in the matrix indicates the next section that can be entered. This can be done using an array of the sections:

[1]	[2]	...	[15]	[16]
2	3	...	-1	-1

Using a locomotive's current section as a subscript, the locomotive's controlling process can determine the next section to enter:

next = look_ahead[current];

Note that the resulting "next" section is not necessarily a "safe" section (see below).

3.2.2 Rules for entering sections

The rules for entering sections are as follows:

- If the next section is empty, proceed.
- If the next section is occupied, stop until allowed to proceed.

3.2.3 Rules for leaving sections

When leaving a section, the exiting locomotive must determine whether the "other" locomotive was waiting to enter the section. If there was a waiting locomotive, the exiting locomotive must "get out of the way" by entering the next available siding.

If locomotive 'A' "got out of the way" for locomotive 'B', locomotive 'B' must indicate to locomotive 'A' when it is safe to return to the track. The same safety rules described above must be applied when locomotive 'A' resumes.

3.3 Hall sensors

When a locomotive passes over a Hall sensor, the process associated with the train must be informed that this has occurred. The process must determine which end of the locomotive triggered the sensor.

The process should voluntarily give up the processor while waiting for a sensor signal.

3.4 Idle process

When the processor ain't workin' on the railroad, it should be doing something useful. The idle process should display the locations of the trains and their status (i.e., moving or sitting on a siding).

3.5 Kernel

The kernel should handle all sensor interpretation and process management.

4 Train Machine

It is possible to write the solution to this assignment with a large number of "if" statements. Instead, what is recommended is the development of a general purpose *virtual machine* (or scripting language) that executes instructions. Possible instructions include "*Turn off section 'X'*",

"*Turn on section 'X'*", "*Switch 'X' straight*", "*Switch 'X' diverged*", "*Check switch 'X' straight*", "*Check switch 'X' diverged*", and "*Stop*". You may want others.

5 Track Display

The state of the track should be displayed graphically on the user's monitor. The location of the train should be highlighted in reverse video. The state of each switch should be clear to the person looking at the monitor. VT-100 escape sequences can be used for this.

6 Marking

This assignment will be marked as follows:

- Design.

  A detailed design document outlining your solution to this problem. Diagrams showing the various data and code structures are required.

  Total points: 7.

- Software.

  A fully commented, indented, magic-numberless, tidy piece of modular software that meets the requirements described above and meets the design description. Avoid "hard-coding" solutions into the application; the solution should be "general purpose", allowing the programmer to modify the system by changing the data, not the software.

  Total points: 13.

- Testing.

  Descriptions of sufficient tests to allow the implementation to be thoroughly exercised. The descriptions *must* include a detailed overview of what the test is attempting to achieve *and* the results of performing the test on your software.

  Total points: 5.

The demonstration must be completed successfully for the assignment to be marked.

7 Important Dates

Available: 5 November 2019

Design: 21 November 2019

Demonstration/Due: 4 December 2019 (Pending class agreement)

8 Miscellaneous

If you are having *any* difficulty with this assignment, *please* contact Dr. Hughes as soon as possible.

Due to the size of the assignment, teams of at most two members can work on a single assignment.

This is a non-trivial assignment.