Out: Tuesday, March 8, 2022

Due: Thursday, March 17, 2022

This lab is to become further familiar with real-time operating system kernel objects and services. You will learn how to use semaphore, mailbox and message queue for inter-task communication and synchronization by adding more features to the vehicle model that was developed in the previous lab.

To test the vehicle model, assume perception-reaction time is 0. Also, assume the vehicle will repeatedly travel the same 1.8 km test track. The details of the test track are described below. Due to uphill and downhill, it is necessary to track the vehicle location on the track in order to determine the retardation force.

The µC/OS-II port for Nios II that is distributed as part of Nios II Embedded Design Suite (EDS) is used for this lab. Nios II EDS can compile, debug, load and run µC/OS-II programs. The hardware to be used in this lab is a prebuilt DE-series computer system. Quartus Programmer is used for downloading the DE-series computer system to Intel FPGA board.

A number of system requirements are given below. Your mark will be predominantly evaluated based on the satisfaction of the **newly added** system requirements. Read through all the requirements carefully prior to design and implementation. The tasks to be modified are I/O task and vehicle task. Newly added constraints are highlighted in yellow. Deleted constraints are highlighted in grey.

All files should be kept in a folder named **cme332_ex4_ucosii_sync**. Create this folder before you proceed with the lab.

Design Constraints

1. The vehicle model must meet the following **key requirements**. (Note: Failure to comply with any of the key requirements will be graded zero)

1.1. The vehicle model must be implemented using µC/OS-II real-time operating system.

1.2. The vehicle model must run on a DE-series board.

1.3. There must be at least three tasks.

1.4. Nios II Embedded Design Suite must be used. The project and its BSP must be named as ex4_ucosii_sync and ex4_ucosii_sync_bsp, respectively.

1.5. Interrupts must not be used. That is, the only interrupt must come from µC/OS-II.

1.6. Shared resources must be guarded by semaphores.

1.7. At least one mailbox and one message queue must be used for inter-task communications.

1.8. In a text file named ucosii_kernel_objects_list.txt, note down information of semaphores, mailboxes and message queues that are used in the design. This file should be stored in the cme332_ex4_ucosii_sync folder.

2. General design requirements are listed below

2.1. There should be at least three periodic tasks: an engine task, an I/O task, and a vehicle task.

2.2. Engine task is responsible for adjusting the throttle based on input status and vehicle speed.

2.3. I/O task is responsible for handling inputs to the system, and displaying outputs to the user. This task does not modify any of the properties of the vehicle directly.

2.4. Vehicle task modifies the acceleration and speed of the vehicle based on the input and external factor as described below. This task also modifies the location of the vehicle according to the acceleration and speed.

2.5. The vehicle model must be tested using the given test track.

3. Constraints on the implementation the engine task are listed below

3.1. The engine must be turned on in order to generate force to accelerate the vehicle. Note that engine status and gear position are handled by the I/O task.

3.2. If accelerator pedal is depressed, a constant throttle of 6 is produced. Note that the throttle signal must be guarded by a semaphore or sent to the vehicle task through a mailbox or a message queue.

3.3. Throttle response time should be made minimal.

3.4. To start the engine: (1) key bob is inside the vehicle; (2) Step on brake; and (3) press and hold the engine start/stop button for longer than 1 second.

3.5. To turn the engine off: (1) gear is in park; (2) press the engine start/stop button.

3.6. To keep the vehicle model simple, engine cannot be turned off during driving.

4. Constraints on the implementation the vehicle task are listed below.

4.1. If brake pedal is not depressed, input acceleration of the vehicle is equal to

$$acceleration = 0.5 \times throttle$$

where throttle is from the engine task. For example, when the throttle value is 6, the equivalent acceleration is 3 m/s^2.

4.2. If brake pedal is depressed, it overrides the accelerator and the input acceleration is set to –6 m/s^2 until the speed reaches 0.

4.3. The vehicle speed can be calculated based on the acceleration:

$$new\_speed = speed + acceleration \times \Delta time$$

where speed is in (m/s), acceleration is in (m/s^2), and $\Delta time$ is in (s). $\Delta time$ is the amount of time since speed was last updated. Note that the vehicle speed must be guarded by a semaphore or sent/received through a mailbox or a message queue.

~~4.4. To keep the vehicle model simple, vehicle speed is unchanged if neither of accelerator and brake is depressed.~~

4.5. The acceleration is also subject to two more factors, the drag and the slope retardation. The net acceleration of the vehicle can be calculated by subtracting the drag and the slope retardation from the input acceleration as follows:

$$retardation = drag + slope\_retardation$$

$$net\ acceleration = 0.5 \times throttle - retardation$$

4.6. If neither pedal is depressed, the drag is proportional to the speed of the vehicle and is defined as

$$drag = speed^2 / 1000$$

where the units of drag are (m/s^2) and the units of speed are (m/s).

4.7. The slope retardation is chosen based on the location of the vehicle on the test track. The track is divided into 300m sections, each with the following slope retardation values due to varied terrain:

Location (m)	[0, 300)	[300, 600)	[600, 900)	[900, 1200)	[1200, 1500)	[1500, 1800)
Slope Retardation (m/s^2)	0	0.3	0.5	0	-0.4	-0.2

4.8. The vehicle location on the track can be determined using the following formula:

$$new\_location = location + speed \times \Delta time + 0.5 \times acceleration \times (\Delta time)^2$$

If the vehicle passed the 1.8 km line, reset the vehicle location.

5. Constraints on the implementation the I/O task are listed below.

5.1. The I/O task records the status of the following inputs for the engine task to reference to.

Switches / Buttons	SW9	SW1	SW0	KEY3	KEY2	KEY0
Functions	Key Fob	Gear Position On (Drive) / Off (Neutral)	Gear Position On (Park) / Off	Brake Pedal	Accelerator Pedal	Engine Start/Stop
Indicator	LEDR9	LEDR1	LEDR0	LEDG3	LEDG2	LEDG0

5.2. The I/O task updates vehicle speed (from the vehicle task) on HEX2-HEX0. Note that the speed is calculated and stored in m/s, but must be displayed in km/h.

5.3. The I/O task turns on a corresponding LEDR according to the vehicle location on the test track which is 1.8 km long,

Location (m)	[0, 300)	[300, 600)	[600, 900)	[900, 1200)	[1200, 1500)	[1500, 1800)
Indicator	LEDR3	LEDR4	LEDR5	LEDR6	LEDR7	LEDR8

Note that the location information from the vehicle task must be guarded by a semaphore or received through a mailbox or a message queue.

Deliverables:

Hand in a single compressed file to Canvas before the due date specified.

1. All files must be kept in cme332_ex4_ucosii_sync folder and organized as shown below.

cme332_ex4_ucosii_sync
cme332_ex4_ucosii_ccs/ex4_ucosii_sync
cme332_ex4_ucosii_ccs/ex4_ucosii_sync_bsp
cme332_ex4_ucosii_ccs/ucosii_kernel_objects_list.txt

2. The cme332_ex4_ucosii_sync folder must be cleaned up before compression; e.g. reproducible and unrelated files must be deleted.

3. Compress cme332_ex4_ucosii_sync folder into a single zip file named cme332_ex4_ucosii_sync.zip. **Other formats (.Z .gz, .7z, .rar, etc.) will be marked zero**.

Demonstration may be required when deemed necessary.