STUDENT NAME: **Student ID:**

ENGG4420: Real Time Systems Design
Instructor: Petros Spachos

Individual Assignment: Weight 10%

Date Issued: October 8, 2025; **Due Date**: November 14 (Friday), 2025, online by 11:59 pm.
Note 1: Students must provide independent work and research the solution without consultations with other students in the class. Also, no questions related to the assignment solutions will be allowed to the course instructor or to the TAs.
Note 2: The solution should be typed; provide answers to all the questions posed in the assignment. Also, you will need to draw some diagrams and present some uC/OS-III code as required.

ASSIGNMENT DESIGN PROBLEM [10 mark].
Please note the mark distribution described in the table.

Assignment Requirements [marks]
Requirement 1 [1 mark]
Requirement 2 [2 marks]
Requirement 3 [3 marks]
Requirement 4 [4 marks]
Total [10 marks]

Introduction.
In a conventional cruise control system, the car can maintain a steady state speed that the driver sets. Adaptive Cruise Control (ACC) is an intelligent system, an enhancement of the conventional cruise control system, which can keep the set cruising speed and match the car's speed in front of you if required. Various settings and challenges are coming with the ACC system. However, in this assignment, we will narrow the requirements of the design. Please consult the posted reference for this assignment (MCDW-Deeper-Learning-ACC.pdf) to understand the challenges and the basic functionality of the adaptive cruise control system. Of course, it is recommended that you research other documents related to this system.

Assignment Requirements.
The controls of the ACC are car-dependent, but in principle, the driver would set two system parameters: a cruising speed V_{cruise} and a minimum safe following distance X_{set} to the car ahead. In this assignment, you will study and design only specific components of an ACC system. Provide complete answers and solutions to all the requirements specified in this assignment. Upload a pdf file with the complete solutions to the Assignment dropbox on ENGG4420 CourseLink. Type your answers and provide legible schematics.

Requirement 1 [1 mark]. Study the posted reference document for the ACC system [MCDW-Deeper-Learning-ACC.pdf]. Select one of the 4 challenges presented in the document that you would be interested

in solving. Provide a short description with your solution to the questions posed in the challenge you choose. Specifically, address the real-time related issues.

Requirement 2 [2 marks]. The ACC system will require safety and fault tolerance support employing hardware and software. Study the reading material posted in the Assignment folder entitled "engg4420_FaultTolerance…" and research other reliability and safety topics online. Answering this requirement, discuss how you would improve the system (hardware and software) and what specific features you would implement to make the ACC system safe and reliable (fault tolerant). Present a general block diagram with the safety modules you would introduce and briefly explain their role.

Requirement 3 [3 marks]. In this requirement, you need to identify the tasks' type (hard, soft, cyclic, event; see slides of ENGG4420 lectures for a reference example). Your system should have, at minimum, the following tasks: setup, ISR, sensors, control, actuator, and display (See Figure 1 below). Of course, you can add more tasks if required, as these tasks outlined here would be only the core tasks.
In our ENGG4420 lectures (week 5 lecture notes), we discussed the topic of task synchronization and the topic of "Inter-Task Communication, Message Passing". Based on the examples presented in these sections you are required to present a functional task diagram of the ACC system, showing the main tasks and the data objects with synchronization and communication structures between these tasks. Your diagram should satisfy the functionality steps presented below. Precisely, our ACC system should synchronize with a timer interrupt that triggers the execution of the sensors' data acquisition task. The ISR routine will post to a semaphore object or flag signal (whichever you decide to use), indicating that the time for sensors' data acquisition and control has arrived. The timer interrupt frequency should be less than 100 ms, as the cars move at fast speeds so that we could have a good system response and have enough time to run all the critical tasks. The simplified overall functionality of the system should be as follows (however, you can improve this if you desire so):

1. ACC monitors the ACC_on_off switch through "Setup" task and when "on" it performs the setup steps for speed, distance, etc. When "off," the task cycles and continues polling the switch.
2. When the timer interrupt is asserted, the system executes the "IRQ_sensors" routine.
    a. The ISR's main functionality is to post to a semaphore or a flag signal to notify the sensors' data acquisition task to perform sensors reading.
3. The sensors' data acquisition task "Sensors" pends on the semaphore or the flag signal (whichever you used) to wait for performing the reading of the system's sensors.
    a. When the semaphore (or flag) is posted, the sensors' data acquisition task executes and reads the distance sensor, the speed sensor, and all the other sensors, shown in Figure 1, as appropriate. The current cycle readings from these sensors X(n) and V(n) (distance and speed, respectively) are stored in a parameter memory block (note that you might need to use a mutex to protect access to this block). This task can update the memory block and signal to the control task that new data is posted or use a queue to communicate the new data to the control task. After collecting the sensors' readings and communicating to the control task, this task waits for the next collection cycle.
4. When signaled, the control task "Control" uses a velocity control algorithm equation (See Requirement 4) to calculate the manipulated variable for the actuators and communicates to the actuator task the new calculation result.
    a. The actuator task "Actuator" sends the new control value to the ACC actuators.
5. The display task "Display" updates the distance and speed on an LCD every 2 seconds. Also, this task displays messages and ACC status as ON/OFF.

STUDENT NAME: **Student ID:**

Requirement 4 [4 marks]. Software implementation in uC/OSS-III. In this requirement, you are required to present a general uC/OS-III code for the ACC system, taking as reference the example illustrated in ENGG4420 lectures. Specifically, the data structure and the main tasks are shown in Figure 1. However, you are only required to implement in detail the control task. All other tasks, including the main program, should only show pseudo-code and the main uC/OS-III call functions that you might use to achieve the proper functionality of the system. For example, your main program should include the OSInit (), OSStart(), and all necessary functions call (initializations, objects creation, task creation, etc.) for multitasking. Similarly, for the ISR routine, you need to present only pseudo-code and show the post function you utilize. Note that the data structure of Figure 1 contains a mixture of variables and parameters with their significance outlined below. You also need to choose appropriate priorities for your tasks and briefly justify your choices.

The control task must implement the function as described below:

- The "Control" task must wait for the new control cycle (n^{th} cycle) with new data, calculate the new manipulated variable dM(n) and store it, then post to the "Actuator" task.
- The "Control" task must compute the dM(n) value, given by equation (3), where n represents the current sample time (the n^{th} interrupt cycle). The dM(n) is also called the manipulated variable and is used to control the engine speed actuators. The dM(n), X(n), V(n), etc., variables and calculations should be treated as floating point variables. The control task calculation requirements are:
    - If the current value, Xn, for cycle interrupt n, is greater than equal to Xset, then the controller performs the calculations given by equations (1), (2), and (3), in this order, and stores the updated value of Vset and the calculated value of dMn in the parameter memory block.
    - If the current value Xn, for the cycle interrupt n, is less than Xset, then the controller routine performs the calculations given by equations (4), (2), and (3), in this order, and updates the calculated values of Vset and dMn in the parameter memory block.
    - After these calculations, the task must post to the "Actuator" task.

$$Vset = Vcruise; \quad (1)$$

$$e(n) = Vset - Vn; e(n-1) = Vset - Vn1; e(n-2) = Vset - Vn2 \quad (2)$$

$$dM(n) = K1 \times e(n) + K2 \times e(n-1) + K3 \times e(n-2); \quad (3)$$

$$Vset = Vset - deltaV \quad (4)$$

Where, e(n), e(n-1), and e(n-2) represent the error values at interrupt cycles n, n-1, and n-2, respectively. And, K1, K2, K3, and deltaV are the controller's parameters.

The figure below shows the parameter memory block structure where the data parameters for the ACC system are stored. This memory block is updated accordingly by the sensors task and other tasks in the system. Therefore, you must manage the memory block accordingly using appropriate uC/OS-III objects. The label designation of the parameters' memory block is "Parameters." The parameters' significance is as follows:

- ACC01: ACC-on-off.
- K1, K2, K3: controller equation parameters.
- Vcruise: the vehicle cruise set speed.
- Vset: the current cycle value for the vehicle speed reference.
- Xset: the minimum safe distance to the car in the front set by the driver.
- Xn: the current distance read by the distance sensor at the n^{th} interrupt cycle.

- Vn: the current vehicle speed read by the speed sensor at the n^{th} interrupt cycle.
- Vn1: the vehicle speed that was stored one cycle before the current cycle, specifically, at cycle (n-1).
- Vn2: the vehicle speed that was stored two cycles before the current cycle, specifically, at cycle (n-2).
- dMn: the manipulated variable calculated by the control algorithm for the vehicle speed feedback control loop.

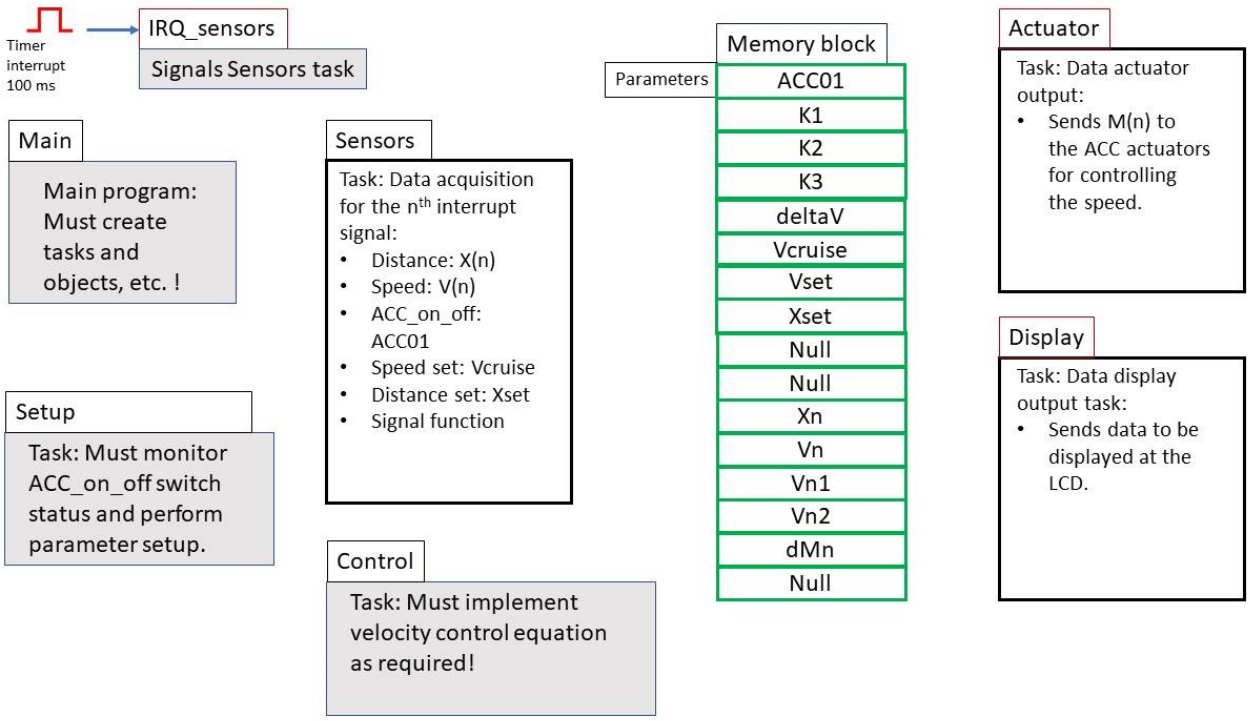

Figure 1. Tasks and data structure for the ACC system. You must also choose appropriate task priorities for your system and justify your choices.