COMPSYS 723 CRUISE CONTROLLER

Akansha Aggarwal & Sophia Creak

Department of Computer Systems Engineering
University of Auckland, Auckland, New Zealand

Abstract

This report will be describing the main elements of creating a cruise control system for a vehicle, as was required for computer systems 723, assignment two. The system's functional specification was created using the synchronous programming language Esterel V5, resulting in an executable reactive program that meets the given requirements. This report will cover the design and specifications used to create the functional cruise control system.

1. Introduction

Cruise controllers are featured in most modern vehicles they allow drivers to maintain a constant speed without using the accelerator pedal. The system uses sensors and an actuator to control the throttle and regulate the vehicle's speed, ensuring it stays within a specified range. The driver can adjust the cruising speed, and the system can be deactivated to manually control the speed using the accelerator pedal. Cruise mode is automatically disabled or put on standby when the accelerator or brake pedals are pressed. To create this system, we used Esterel V5, a synchronous, system-level language for embedded systems. The project involves creating formal design specifications, performing dataflow and control-flow decomposition, and modeling the system's behavior with a concurrent finite state machine. The final implementation meets the specified requirements, ensuring accurate speed maintenance and passenger comfort.

2. Specifications and Design

It is important to fully understand a project before starting it, so time was taken to visualise and fully understand the project brief and design requirements.

2.1. Context Diagram

The overall input and output structure can be seen in the context diagram below, figure one.

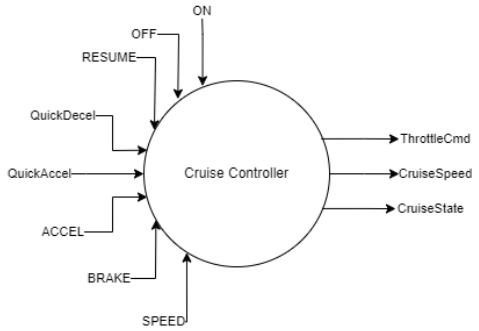


Figure One: Cruise Controller Context Diagram

The cruise controller takes in the following inputs:
1) On (pure): Enable the cruise control.
2) Off (pure): Disable the cruise control.
3) Resume (pure): Resume the cruise control after braking.
4) Set (pure): Set the current speed as the new cruise speed.
5) QuickDecel (pure): Decrease the cruise speed.
6) QuickAccel (pure): Increase the cruise speed.
7) Accel (float): Accelerator pedal sensor.
8) Brake (float): Brake pedal sensor.
9) Speed (float): Car speed sensor.

These inputs are used throughout the system to create the following outputs:

1) CruiseSpeed (float): Cruise speed value
2) ThrottleCmd (float): Throttle command
3) CruiseState (enumeration): State of the cruise control. It can be one of: OFF, ON, STDBY or DISABLE.

2.2. Low-Level Context Diagram

To fully understand the outputs the context state machine was broken up into a low-level version, figure two.

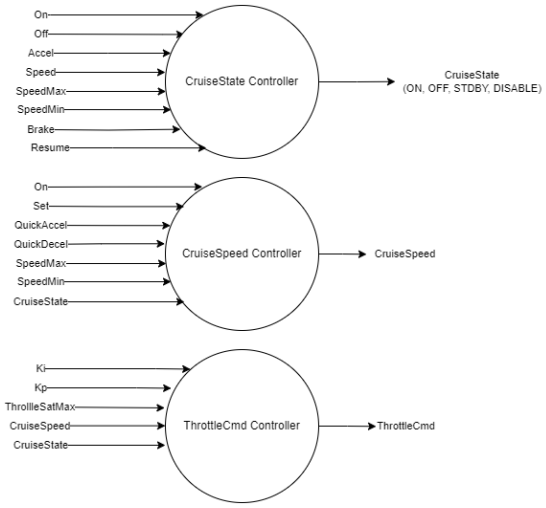


Figure Two: Low-Level Context Machine

This helped us understand the different controllers we would need to implement in Esteral. The cruise controller contains three controllers or modules each associated with a different output. This modular approach simplifies the implementation, allowing individual focus on each component's signal flow and timing. CruiseSpeedController module uses system inputs to create the system output CruiseState. CruiseState as explained above is an enumeration used to describe the state of the cruise controller. Each state has an attributed number; OFF (1), ON (2), STDBY (3), DISABLE (4).
CruiseSpeedController module uses the CruiseState and system inputs Set, Speed, QuickAccel and QuickDecel to create the system output CruiseSpeed. CruiseSpeed as described above is a float used to describe the cruise speed value. This is not the speed of the car (Speed) but what the cruise speed is currently set to.
ThrottleCmdController module uses CruiseState, CruiseSpeed and system inputs Speed and Accel to create the system output ThrottleCmd. ThrottleCmd as described above, is a float used to describe the amount of throttle needed to take the current speed up to the set cruise speed.

2.3. Finite State Machine

We then used a finite state machine (FSM) to help us understand how the logic worked within the CruiseStateContoller, figure three.

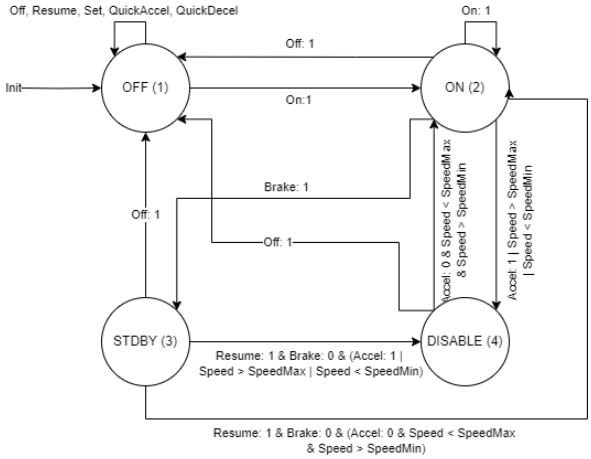


Figure Three: Cruise Controller Finite State Machine

OFF (1) is the initial state, it can only transition to ON (2) when the On button is pressed.
ON (2) is the cruise speed management state. If Brake is pressed it will transition to STDBY (3). If the Accel is pressed of Speed is outside of speed range, minimum 30km/h and maximum of 150km/h, then transition to DISABLE (4).
STDBY (3) is standby state, this state can only be left once Resume button is pressed. If Resume is pressed, Accel is low and Speed is within range state returns to ON (2). If Resume is pressed and Accel is pressed or Speed is outside range, then state transitions to DISABLE (4).
DISABLE (4) is the disabled state. This state can only transition to ON (2) once Accel is no longer being pressed on Speed is within range.
All states return to OFF (1) if the Off button is pressed.

2.4. Other Specifications

The other two modules are not as well explained in diagrams.
CruiseSpeedController emits CruiseSpeed in every loop and waits for CruiseState. If CruiseState is not off then it checks if either Set, QuickAccel or QuickDecel has been pressed. If Set is pressed; CruiseSpeed is set to current speed. If QuickAccel is pressed; CruiseSpeed is increased by a factor of SpeedInc (2.5). If QuickDecel is pressed; CruiseSpeed is decreased by a factor of SpeedInc (2.5). If CruiseSpeed goes outside set range then it is reset to maximum or minimum. If the state was OFF before this tick then CruiseSpeed is set to current speed. If state is OFF then CruiseSpeed is set to 0.0.
ThrottleCmdContoller emits ThrottleCmd conditionally in every loop and waits for tick. Inside the loop, a trap block named throttle is used to conditionally emit the

ThrottleCmd signal based on the value of State_1 and possibly its previous value. Depending on the state, it either directly emits the accelerator value or calculates the throttle using a regulateThrottle function with different parameters, then exits the trap.

3. Testing

To execute and test our code using the Graphical User Interface (GUI), we utilized specific commands in the terminal from the project directory:

- $ make cruiseregulation.xes
- $ ./cruiseregulation.xes

We manually verified the conditions provided in the vectors.in files and compared our outputs with those in the vectors.out files. The recorded results are presented in tables and screenshots found in Appendix A. Some extra test cases to show the functionality of the cruise controller can be found in Appendix B.

4. Conclusions

This project successfully designed and implemented a simplified cruise control system using the synchronous programming language Esterel V5. By developing formal design specifications, performing dataflow and control-flow decomposition, and modeling with a concurrent finite state machine, we created a functional system that meets all specified requirements.
The system's key components, including the CruiseSpeedController, CruiseStateController, and ThrottleCmdController, were developed modularly, ensuring precise control and easy verification. The use of Esterel V5 enabled a reactive and deterministic program suitable for real-time applications.
Testing confirmed that the system accurately maintained the car's speed and handled state transitions correctly based on user inputs and sensor data. Overall, the project provided valuable hands-on experience in synchronous programming and embedded system design, achieving accurate speed maintenance and passenger comfort.

Appendix A

The results of the test cases provided in vectors.in with output compared to vectors.out file, referred to as in section 3 of this report are located here:

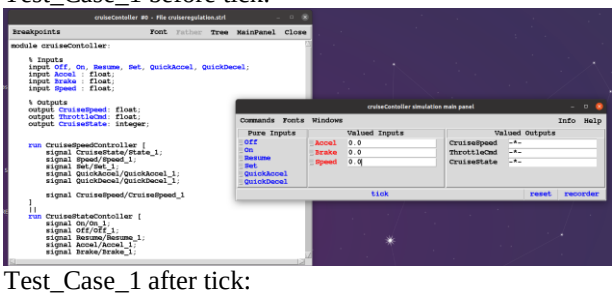
vectors.in

```
# On Off Resume Set QuickAccel QuickDecel Accel Brake Speed
false false false false false false 0.0 0.0 0.0
false false false false false false 0.0 0.0 0.0
false false false false false false 51.234 0.0 0.0
false false false false false false 51.234 0.0 2.0937
false false false false false false 51.234 0.0 4.1862
false false false false false false 51.234 0.0 6.2772
false false false false false false 51.234 0.0 8.3661
false false false false false false 89.687 0.0 10.453
false false false false false false 89.687 0.0 13.051
false false false false false false 89.687 0.0 15.644
false false false false false false 89.687 0.0 18.233
false false false false false false 89.687 0.0 20.816
false false false false false false 89.687 0.0 23.393
false false false false false false 89.687 0.0 25.962
false false false false false false 89.687 0.0 28.524
false false false false false false 89.687 0.0 31.078
false false false false false false 89.687 0.0 33.623
false false false false false false 0.0 0.0 36.158
true false false false false false 0.0 0.0 36.049
false false false false false false 0.0 0.0 35.94
```

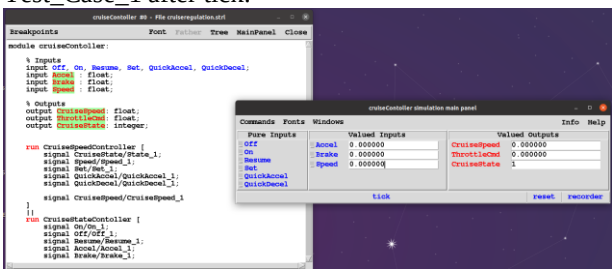
vectors.out

```
# CruiseSpeed ThrottleCmd CruiseState(OFF-1 ON-2 STDBY-3 DIS-4)
0.000000 0.000000 1
0.000000 0.000000 1
0.000000 51.234001 1
0.000000 51.234001 1
0.000000 51.234001 1
0.000000 51.234001 1
0.000000 51.234001 1
0.000000 89.686996 1
0.000000 89.686996 1
0.000000 89.686996 1
0.000000 89.686996 1
0.000000 89.686996 1
0.000000 89.686996 1
0.000000 89.686996 1
0.000000 89.686996 1
0.000000 89.686996 1
0.000000 0.000000 1
36.049000 0.000000 2
36.049000 0.938827 2
```

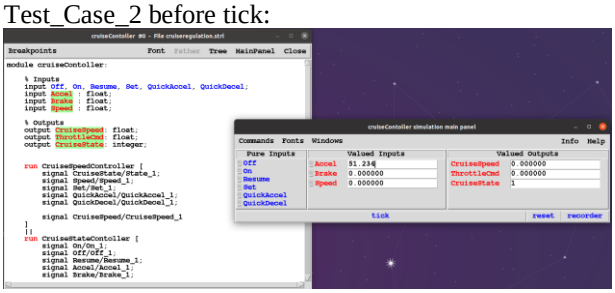
Test_Case_1 before tick:



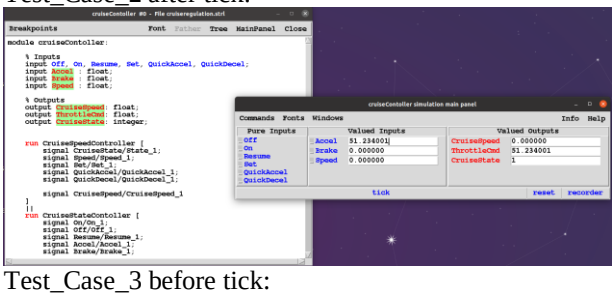
Test_Case_1 after tick:



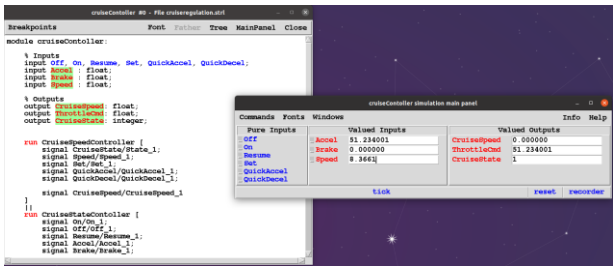
Test_Case_2 before tick:



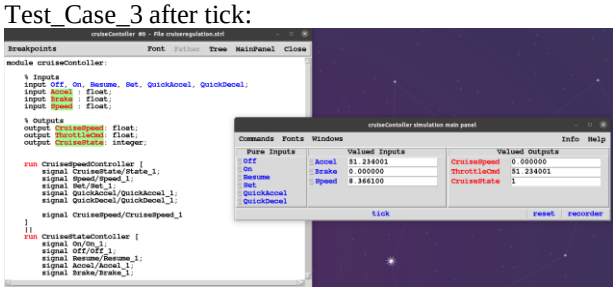
Test_Case_2 after tick:



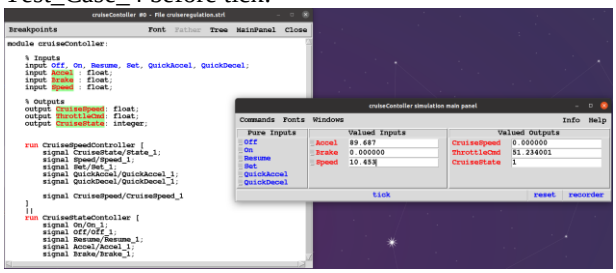
Test_Case_3 before tick:



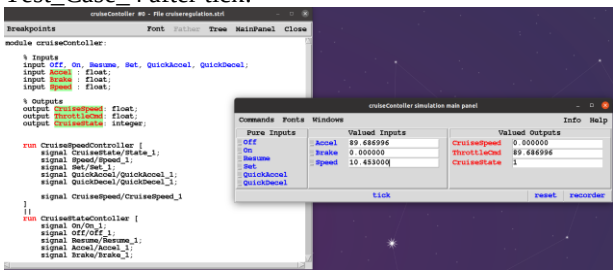
Test_Case_3 after tick:



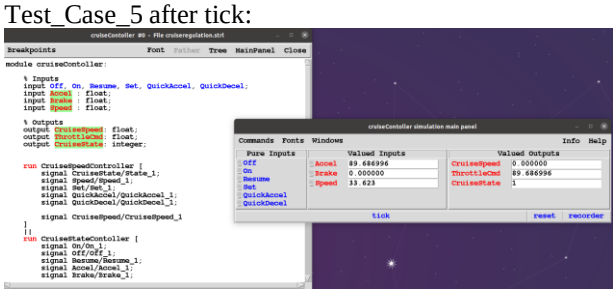
Test_Case_4 before tick:



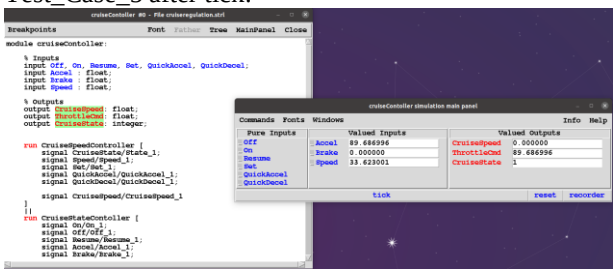
Test_Case_4 after tick:



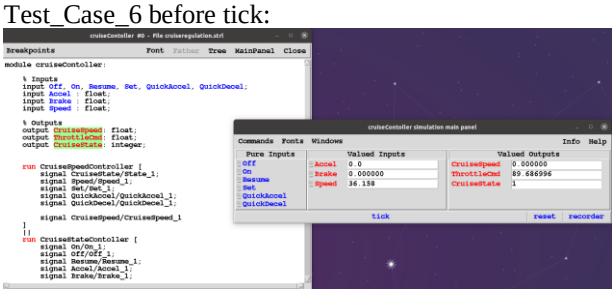
Test_Case_5 after tick:



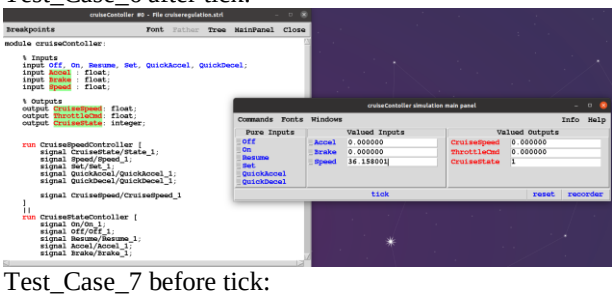
Test_Case_5 after tick:



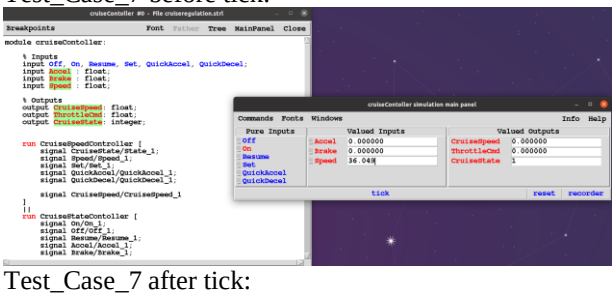
Test_Case_6 before tick:



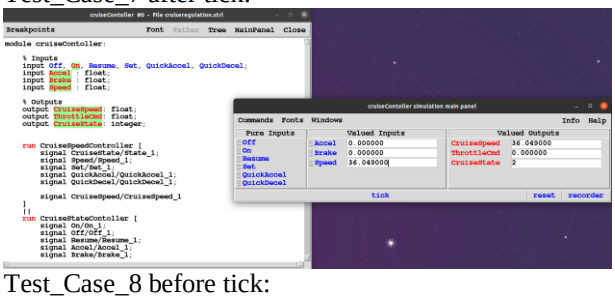
Test_Case_6 after tick:



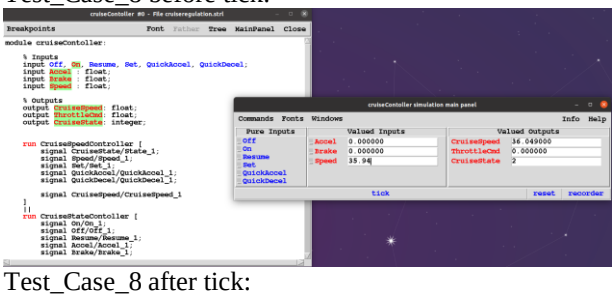
Test_Case_7 before tick:



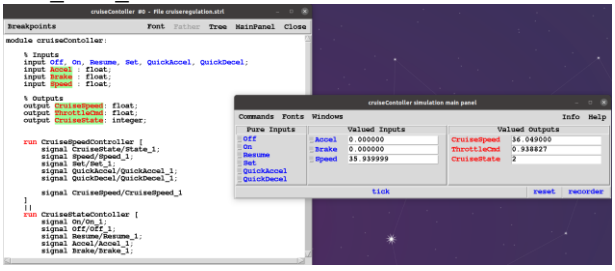
Test_Case_7 after tick:



Test_Case_8 before tick:



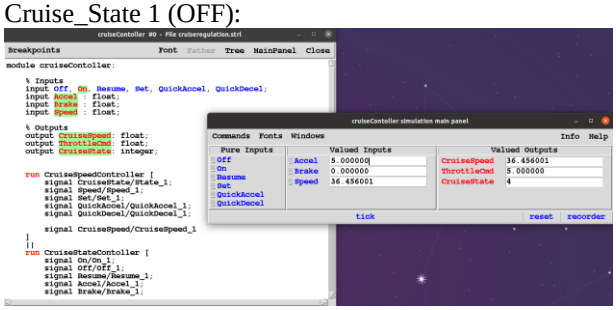
Test_Case_8 after tick:



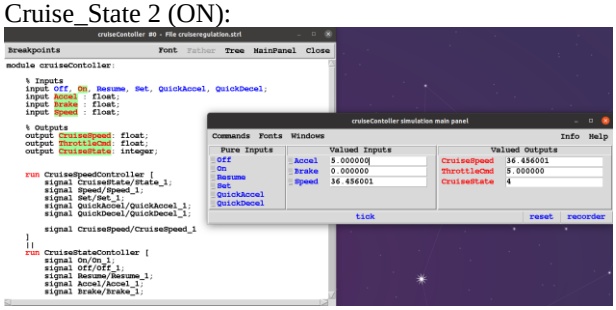
Appendix B

Some extra test cases to show the functionality of the cruise controller, referred in section 3 are located here:

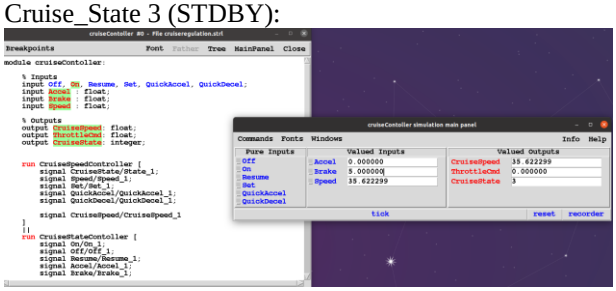
Cruise_State 1 (OFF):



Cruise_State 2 (ON):



Cruise_State 3 (STDBY):



Cruise_State 4 (DISABLE):