COMPSYS 723 ASSIGNMENT 2 REPORT

Sylvain Bechet, Mark Yep

Department of Electrical and Computer Engineering
University of Auckland, Auckland, New Zealand

Abstract

We present the design of a cruise control system in Esterel. Based on a set of requirements, we first developed a set of functional specifications for a cruise control system. We then implemented these specifications using the synchronous programming language Esterel. This allowed us to develop an executable reactive program that fulfils the given requirements for a cruise control system. We then tested our solution with a variety of inputs to ensure it was working as intended.

1. Introduction

This solution is based on an automobile cruise controller system with given specifications. The purpose of this system is to maintain a car at a constant speed without the driver pressing the accelerator or brake pedals. The system controls the car throttle command to maintain a constant speed under the conditions that it does not exceed a specific maximum speed value or fall below a minimum speed value. Furthermore, the system does not operate while either the brake or the accelerator pedals are pressed. This is to ensure that the user can manually speed up or slow down if required. This solution is implemented using the Esterel language as it offers many features which facilitate the development of the system, the prominent one being its ease of implementing concurrency. Esterel also provides a deterministic solution which ensures that we do not get unexpected output for a tested input sequence.

2. Development Process

The process of creating the cruise control system involved several stages of development. This followed an engineering design flow to ensure that the development was undertaken efficiently to create a solution which meets all the requirements specified. This report also follows the stages of development, in the order detailed below.

The first stage of this process was analysing the provided specifications. This ensured our development was aimed at meeting these specifications, so that minimal time would be wasted rewriting code due to inconsistencies in our resulting solution.

The second stage of the process was to split our development into tasks, and to define these tasks with diagrams. These diagrams are included in this report as figures and appendices. Diagrams provide an overview of the specifications and outline how our cruise control system will work.

The third stage of the process was to implement these diagrams in Esterel to create a reactive and deterministic solution. This involved researching methods to implement FSMs and flowcharts with modules. Also included in this stage was interlinking these modules so that they worked interdependently.

The final stage of the process was to test and verify our solution. This involved using a range of input combinations and analysing the output to ensure our system was working as intended and met the required specifications.

3. Specifications

The overall behaviour of the cruise controller is described using the context diagram shown in Figure 1.

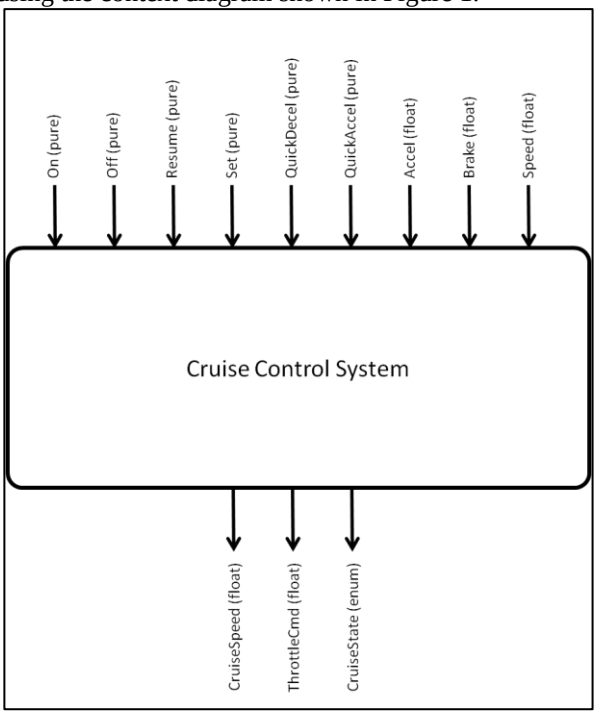


Figure 1: System Context Diagram

The system takes inputs from the driver through pedals, and buttons. The system also takes the cars current speed as input. After processing the inputs, the cruise controller then outputs three signals to other parts of the car.

3.1. Inputs

The driver inputs to the cruise controller consist of the accelerator pedal, the brake pedal, and six different buttons – Off, On, Resume, Set, QuickAccel, and QuickDecel. The current speed of the car is also input to the cruise controller. The Off and On buttons are used to turn the system off and on respectively while the Set button takes the current speed of the car and sets it as the desired cruising speed. As the cruise controller is disabled after the accelerator is pressed, the Resume button is used to re-enable the control system without having to restart the system. This also conserves the previously saved cruise speed.

While the cruise controller is turned on, the driver has the possibility of pressing the QuickAccel or QuickDecel buttons. These buttons increment and decrement the current cruise speed by a set value respectively. If either the accelerator pedal or brake pedals are sufficiently pressed, they affect the state of operation of the system. The brake pedal will change the system to a standby state and the accelerator pedal will change the system to a disable state. These states are held until both pedals are released. Finally, the cars current speed is sampled when setting the cruising speed. This is also used to ensure that the system does not operate while outside a range of reasonable speeds. This implies that the cruise controller will be disabled if the car is above or below the threshold speed values.

3.2. Outputs

There are three signals which are outputs of the cruise controller – the cruise speed, the throttle command, and the cruise state. These outputs are used by other components of the car to apply the control functions of the cruise controller.

The cruise speed is used to give feedback to the driver about the currently set cruising speed of the cruise controller. The throttle command is related to the amount of fuel per unit time fed into the engine, while the cruise state informs the driver about which of the four states the cruise controller is in: off, on, disabled or standing by.

4. Esterel Overall Design

The top-level context diagram in Figure 1 can be further detailed into the lower-level context diagram shown in Appendix A: Cruise Controller Diagram to highlight how each module in the system relates with system inputs, outputs, and other modules. Each of the three modules handles the computation of a single output. These modules are dependent on the inputs to the system and on the outputs of concurrently running modules.

The cruise control system module is the only module which does not rely on other modules' outputs. Instead it only accesses the inputs On, Off, Resume, Accel, Brake, and Speed to determine in which state the cruise controller should be in. This system module operates according to a Mealy finite state machine (FSM) detailed in Appendix B: Cruise State FSM to determine the state of the cruise controller.

The cruise speed management module then uses the output calculated in the cruise control system as well as the inputs Set, Resume, QuickAccel, and QuickDecel to determine the value of the current cruising speed. This module is detailed in Figure 2. We have defined it as a flow chart as it processes information through a series of conditional statements to determine the cruise speed.

Finally, the car driving control module uses the output values of both previously described modules to calculate the throttle value to pass to the throttle command. It also uses the current speed of the car as an input. This module uses an externally defined function which calculates the required throttle value from the cruise speed and the current speed.

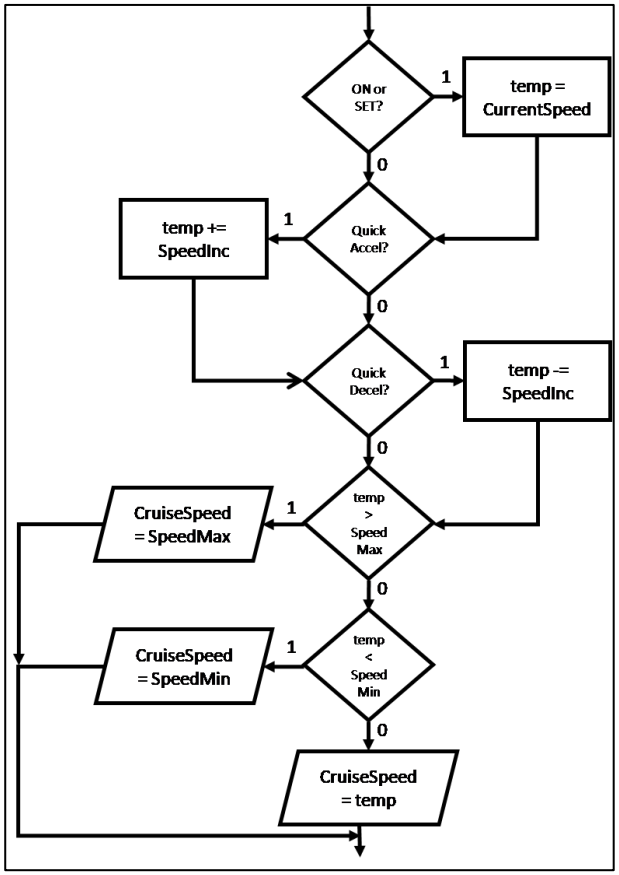


Figure 2: Cruise Speed Flowchart

5. Esterel Module Design

The cruise controller system has been designed based on the specifications mentioned in the previous section using the Esterel language. Three modules are used to define the behaviour of the system – the CruiseStateFSM module, the CruiseSpeedFlow module, and the CarSpeedControl module. These modules are encapsulated under the top-level CruiseController module.

5.1. CruiseStateFSM

The cruise control system is defined in the CruiseStateFSM module. This module is defined as an FSM that handles the inputs On, Off, Resume, Accel, Brake and Speed to determine the state in which the cruise controller should be in. As described in Appendix B: Cruise State FSM, the system can either be off, on, disabled or standing by. This results in an FSM which contains four states. To minimise unexpected behaviour, each state was assigned a relative priority. The off state has the highest priority, followed by the standby state, the disable state, and then the on state.

The off state was given the highest priority because turning off the system should be an immediate and non-interruptible action. We then decided that the standby state should have a higher priority than the disable state. This is because the standby state is dependent on the brake pedal being pressed while the disable state is dependent on the accelerator pedal being pressed. We decided that any actions relating to the brake should have a higher priority than any actions relating to the accelerator, for obvious automobile safety reasons. Finally, the on state was given the lowest priority as the system should only be active if it is safe for it to be on.

The off state is the initial state because when the car starts up the system will initially be off. When the on button is pressed then the system moves into the on state, signifying cruise control has started. If the brake pedal is pressed then the system moves into the standby state where it will pause itself until the brake pedal is released. This is to ensure that the car isn't trying to speed up in the case of the driver needing to slow down. Also from the on state, if the accelerator pedal is pressed or the car speed is beyond the minimum or maximum values, then the controller will enter the disable state. This ensures that the car does not try to regulate while the driver is attempting to go faster.

5.2. CruiseSpeedFlow

The cruise speed management system is defined in the CruiseSpeedFlow module. This module is detailed in Figure 2. It follows a flow chart design, processing information through a series of conditional statements to determine the cruise speed. It uses some of the state of the system calculated in CruiseStateFSM as well as the inputs Set, Resume, QuickAccel and QuickDecel to determine the value of the cruising speed. This module only runs when the cruise control system model is in any state other than the off state.

This module works by either setting the cruise speed to the current speed, incrementing the cruise speed, decrementing the cruise speed, or keeping the cruise speed unchanged. If the cruise speed is changed then it is checked against the threshold minimum and maximum values for the cruise speed. If it is outside this range then it is set to be the corresponding maximum or minimum value.

5.3. CarSpeedControl

The car driving control module is defined in the CarSpeedControl module. This module is used to calculate the throttle value to pass to the throttle command. It uses the state output from the cruise control system module to select whether the throttle command output is dependent on the accelerator pedal or on the cruise controller value. It then uses the cruise speed output from the cruise speed management module for determining the throttle value if required. This value and the current car speed is passed into the external saturateThrottle and regulateThrottle functions, which are defined in the corresponding C code. These define an integral based control system to regulate the car throttle.

The system will calculate throttle values from the car speed and the current cruise speed using the integral control system if the current state is on. If the current state is anything other than on then the throttle value will be a direct function of the accelerator pedal, which is the same behaviour as a regular automobile.

5.4. CruiseController

The cruise controller module is defined in the CruiseController module. This is the top-level module which runs the CruiseStateFSM, CruiseSpeedFlow and CarSpeedControl modules concurrently, while also linking all the signals correctly. This module acts as the interface to the system and handles all the external input and output values.

6. Esterel Testing and Validation

After development, the system was tested using the supplied test file. This accounted for a range of input values to determine if our system provided the correct output or not. Additionally, the system was tested using our own custom values to ensure we had tested as many input combinations as possible.

This black-box testing is sufficient for our purposes as Esterel is a deterministic language. This means for a

specific input combination, there will be a single output combination observed.

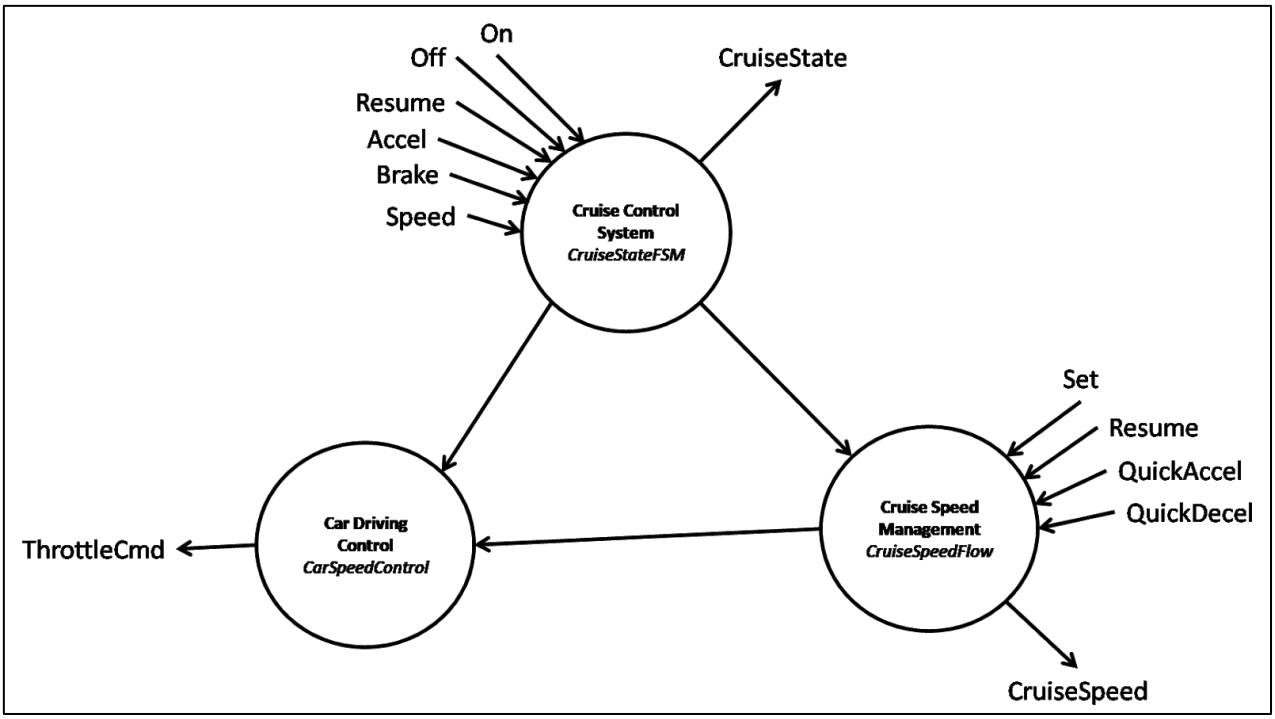
7. Conclusions

This report details the implementation of a car cruise controller from given specifications. These specifications are detailed in this report and have shaped the design of the cruise control system in Esterel. This report is aimed to give an understanding of the design, development, and validation process undergone to create the cruise control system.

Acknowledgements

We would like to thank the lecturer Avinash Malik for his clear and concise lecturing style, and the teaching assistants Nathan Allen and Keith Ren for sharing their knowledge with us.

8. Appendix A: Cruise Controller Diagram



9. Appendix B: Cruise State FSM