COMPSYS 723 ASSIGNMENT 2 REPORT

Matthew Taylor, Maryam Raza, David Su

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

Abstract

This report details the design and implementation of a cruise control system using the Esterel programming language. Using a defined set of requirements, we developed and implemented the functional specifications for the system using Esterel, resulting in an executable reactive program. Testing with various inputs was conducted to ensure the system's proper functionality and performance.

1. Introduction

This project focuses on the design and implementation of a cruise control system using the Esterel programming language. It provided a hands-on approach to implementing embedded software through a model-based approach. Starting from a clearly defined set of requirements, we crafted functional specifications for the cruise control system. These specifications were subsequently implemented using Esterel, allowing us to create an executable reactive program that satisfies the given requirements. Comprehensive testing with various inputs was performed to verify the system's functionality and performance. The project aims to enhance understanding and skills in model-based design and embedded software development, highlighting the complete process from specification to implementation and testing.

2. Specifications

The overall input-output behaviour of the cruise control system is described using the context diagram shown in Figure 1. The cruise control system reads the user inputs through the vehicle controls. This consists of the 'On' and 'Off' controls to enable and disable cruise control respectively. This also includes 'Resume', to resume cruise control after braking, 'Set', to set the current speed as the new cruise speed, 'QuickAccel', and 'QuickDecel' to increase and decrease the cruise speed respectively. Other inputs to the cruise control are the sensors for the acceleration pedal 'Accel', brake pedal 'Brake', and speedometer of the vehicle 'Speed'.

The outputs from the cruise control system are the cruise speed output 'CruiseSpeed', the throttle command output 'ThrottleCmd', and cruise state output 'CruiseState' which indicates the state of the cruise control—OFF, ON, STDBY, or DISABLE.

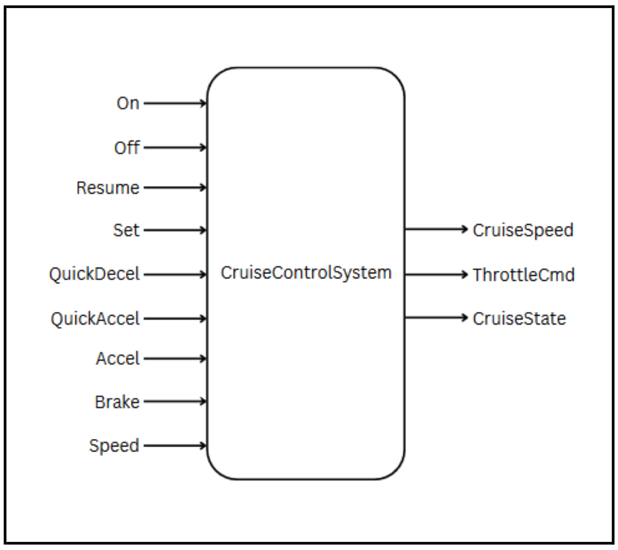

Figure 1. Context Diagram of Cruise Control System

The top-level context diagram can be further refined into the next level diagram as shown in Figure 2. Here the cruise control functionality is further partitioned into a state controller, cruise speed controller, and throttle controller.

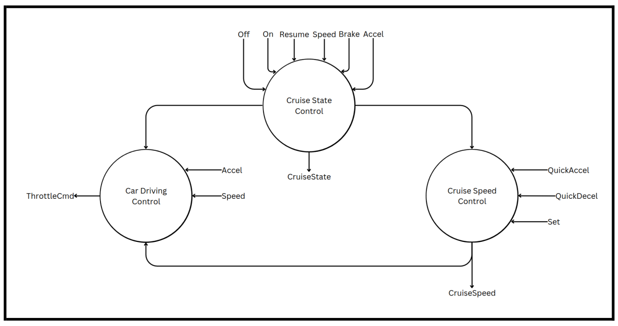

Figure 2. First-level Refinement

3. Design in Esterel

The cruise control system has been designed based on the specifications using the Esterel programming language. Three modules are used to define the behaviour of the system—the StateMachine module, the

CarDrivingControl module, and the CruiseSpeedManagement module.

3.1. Cruise State Control module

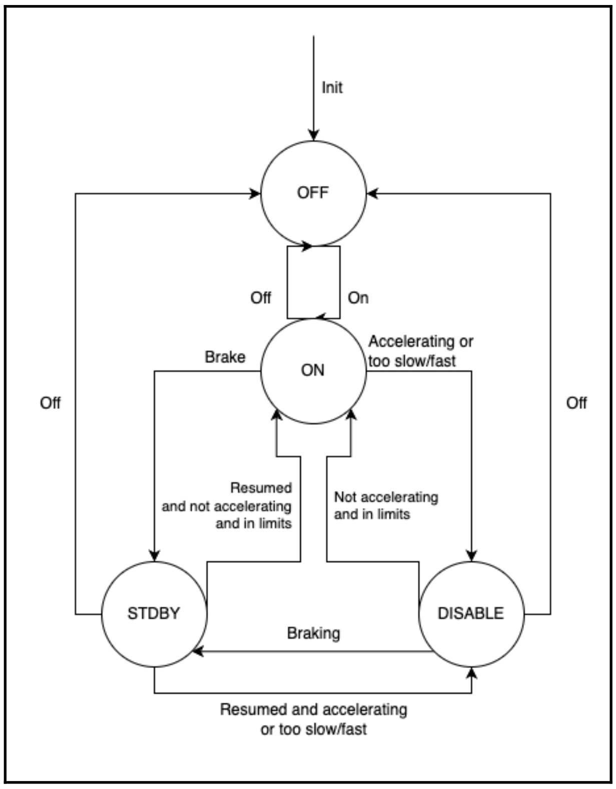
Init
OFF
Off
On
ON
Brake
Accelerating or too slow/fast
Off
Off
Resumed and not accelerating and in limits
Not accelerating and in limits
STDBY
DISABLE
Braking
Resumed and accelerating or too slow/fast

Figure 3. Cruise State Control FSM

The state module acts as a central control unit for the entire system, switching between off (OFF), on (ON), standby (STDBY), and disabled (DISABLE) modes. It defaults to an off state, utilising an output signal CruiseState1 (mapped to CruiseState in the top level module) to share the state with the rest of the system, and utilises a loop to process the state each tick.

Inside the loop is a trap statement, allowing the program to exit execution for the tick once a state has been changed. Going through a series of conditional statements, which evaluate whether certain buttons have been pressed (on, off, resume, accelerate) and current readings (brake value and speed), it determines whether the system should switch states.

If the on button is pressed while off, the state machine will always transition to the ON state. Similarly, if the off button is pressed at any time, the system will always transition to the OFF state. These transitions both have the highest priority in every state. When the system is on, but the speed reading is outside the speed limits or the accelerator has been pressed, the system will transition to the DISABLE state. It may exit this state when the accelerator is released, and the vehicle becomes within the speed limit. However, when the cruise control is in the ON or DISABLE state, it will transition into the STDBY state whenever the brake is pressed. This state can only be exited when the resume button is pressed, and will either transition to ON or STDBY depending on whether the accelerator is being pressed and if the car is within the speed limits.

3.2. Cruise Speed Management module

The Cruise Speed management function is handled and executed as a module within the Esterel code, using the corresponding name, CruiseSpeedManagement. The module uses inputs QuickAccel, QuickDecel, Set and State, to alter a temporary variable according to several requirements through the input signals, before emitting it to the output Speed signal.

Once cruise control mode is initiated, the cruise speed is set to, and held at the current speed value, unless it is either above the maximum speed value, or below the minimum speed value, in which case it will be set as the maximum or minimum speeds respectively. This functionality is replicated whilst the Set button is pressed.

Upon pressing the QuickAccel and QuickDecel buttons, the speed is either incremented or decremented by the value of variable SpeedInc. Actions carried out by buttons QuickAccel, QuickDecel and Set are placed into Present states for instantaneous detection and execution, due to the critical time constraints for the outcomes expected from the pressing of these buttons.

3.3. Car Driving Control module

The car driving control module is defined in the CarDrivingControl module. This module is used to calculate the throttle value to pass to the throttle command ‘ThrottleCmd’. It reads the state output ‘CruiseState’ from the cruise state control module ‘StateMachine’ to determine whether the throttle command output is dependent on the accelerator pedal or should be regulated by the cruise control system. If cruise state ‘CruiseState’ is ‘ON’, then it reads the cruise speed output from the cruise speed management module ‘CruiseControl’ to determine the throttle value by passing a boolean, whether or not the cruise state is transitioning from ‘OFF’ to ‘ON’, the cruise speed, and vehicle speed into the external c function regulateThrottle() defined in cruiseControl_data.c. The regulation is done using a proportional and integral algorithm, with *Kp* and *Ki* factors. To protect against the overshoot of its integral part: the integral action will be reset when the cruise control is going on, and frozen when the throttle output is saturated by using an external utility c function defined in cruiseControl_data.c. The throttle command ‘ThrottleCmd’ will also be saturated at ‘ThrottleSatMax’ when automatically regulating, in order to limit the car acceleration for passenger comfort.

If the cruise state 'CruiseState' is anything other than 'ON', then the throttle value will be directly mapped to the accelerator pedal value 'Accel', in other words operating like a regular vehicle.

3.4. Cruise Control

The CruiseControl module is the top-level module which runs the Cruise State Control module 'StateMachine', Cruise Speed Management module 'CruiseSpeedManagement', and the Car Control module 'CarDrivingControl' concurrently, while mapping their corresponding signals. This module interfaces with the system and handles the input and output values described in the top-level diagram in Figure 1.

4. Esterel Testing and Validation

After designing and integrating the modules, the system was using the given test files 'vectors.in' and 'vectors.out'. See Appendix A for xes recording of validation testing using test files. To validate our design, a range of input values were used to determine if our system responded according to the design requirements.

Because the provided vector files mostly only tested the system while the cruise control was off, we also validated the system manually with a variety of scenarios and edge cases. This included testing that CruiseSpeed was set when turned on, and when the set button was pressed, testing that the CruiseSpeed would never exceed the limits, that the pedal values were ignored when equal or less than 3%, and many other cases.

5. Conclusions

This technical report details the design and implementation process of a simple cruise control system based on a functional specification. These specifications are implemented in our design detailed in this report using the synchronous programming language Esterel . This report is aimed to give an understanding of our design, implementation, and validation process throughout this assignment.

6. Appendix A: Validation XES Recording (.esi)

```
Accel="0" Brake="0" Speed="0" ;
% Outputs: CruiseSpeed("0.000000") ThrottleCmd("0.000000") CruiseState("1")
%
Accel="0.000000" Brake="0.000000" Speed="0.000000" ;
% Outputs: CruiseSpeed("0.000000") ThrottleCmd("0.000000") CruiseState("1")
%
Accel="51.234" ;
% Outputs: CruiseSpeed("0.000000") ThrottleCmd("51.234001") CruiseState("1")
%
Accel="51.234001" Speed="2.0937" ;
% Outputs: CruiseSpeed("0.000000") ThrottleCmd("51.234001") CruiseState("1")
%
Speed="4.1862" ;
% Outputs: CruiseSpeed("0.000000") ThrottleCmd("51.234001") CruiseState("1")
%
Speed="6.2772" ;
% Outputs: CruiseSpeed("0.000000") ThrottleCmd("51.234001") CruiseState("1")
%
Speed="8.3661" ;
% Outputs: CruiseSpeed("0.000000") ThrottleCmd("51.234001") CruiseState("1")
%
Accel="89.687" Speed="10.453" ;
% Outputs: CruiseSpeed("0.000000") ThrottleCmd("89.686996") CruiseState("1")
%
Speed="13.051" ;
% Outputs: CruiseSpeed("0.000000") ThrottleCmd("89.686996") CruiseState("1")
%
Speed="15.644" ;
% Outputs: CruiseSpeed("0.000000") ThrottleCmd("89.686996") CruiseState("1")
%
Speed="18.233" ;
% Outputs: CruiseSpeed("0.000000") ThrottleCmd("89.686996") CruiseState("1")
%
Speed="20.816" ;
% Outputs: CruiseSpeed("0.000000") ThrottleCmd("89.686996") CruiseState("1")
%
Speed="23.393" ;
% Outputs: CruiseSpeed("0.000000") ThrottleCmd("89.686996") CruiseState("1")
%
Speed="25.962" ;
% Outputs: CruiseSpeed("0.000000") ThrottleCmd("89.686996") CruiseState("1")
%
Speed="28.524" ;
% Outputs: CruiseSpeed("0.000000") ThrottleCmd("89.686996") CruiseState("1")
%
Speed="31.078" ;
% Outputs: CruiseSpeed("0.000000") ThrottleCmd("89.686996") CruiseState("1")
%
Speed="33.623" ;
% Outputs: CruiseSpeed("0.000000") ThrottleCmd("89.686996") CruiseState("1")
%
Accel="0" Speed="36.158" ;
% Outputs: CruiseSpeed("0.000000") ThrottleCmd("0.000000") CruiseState("1")
%
```

```
On Speed="36.049" ;
% Outputs: CruiseSpeed("36.049000") ThrottleCmd("0.000000") CruiseState("2")
%
Speed="35.94" ;
% Outputs: CruiseSpeed("36.049000") ThrottleCmd("0.938827") CruiseState("2")
%
```