CS723 Assignment 2 Report

Group 17 - Beck Busch & Francis Cho

1. Abstract

For this assignment, we designed and implemented a simple cruise control system using the synchronous programming language Esterel. Our implementation consisted of 3 main sections; a system state controller, a cruise speed regulator, and a car speed controller. Alongside these main components, a head module runs everything in parallel, and auxiliary modules provide signals for speed and pedal detection. Rigorous testing using the executable reactive program was carried out to ensure all functional requirements were being met.

2. Introduction

Cruise control is a system that is relatively common in modern cars that allows the car to maintain a constant speed without any input from the driver. This is extremely useful for driving long distances on highways or country roads, as the driver no longer needs to manage the accelerator pedal and can drive more relaxed. Another benefit to cruise control systems is fuel usage, as the car's computer can often provide finer control over the engine than a human driver.

Cruise control systems need to balance comfort for the driver with maximum control over the car, as it can be hazardous to take control away from the driver at any time. This is done by creating a system where the driver can resume control of the car at any time simply by using the pedals, which is already the first step in avoiding a dangerous situation. This means that the driver can simply rely on their existing training and instincts, and the cruise control system will automatically yield control of the vehicle.

2.1. Learning objectives

The objectives for this assignment are to provide an introduction to high-level synchronous programming in Esterel. Focus is placed not only on the development of Esterel code but also on the preparation and creation of state machine diagrams to assist with the design.

3. Specification

Embedded systems software is often used for precise and safety-critical applications. Due to this nature, it is essential that the specifications of the system are carefully defined and adhered to.

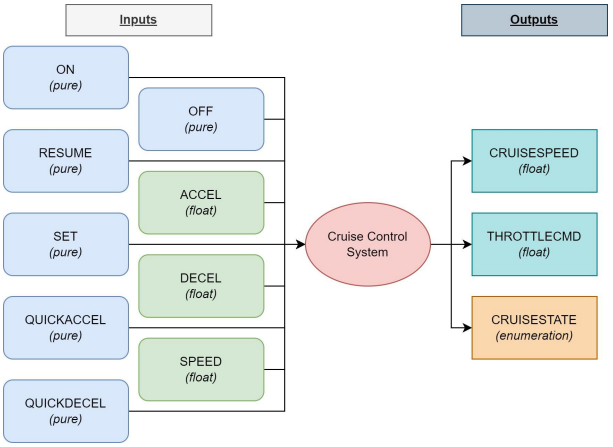
3.1. System design requirements

The requirements of this system outline seven parameters for managing the operation of the state machine and speed algorithms. These can be seen in Table 1. Along with these parameters, there are several inputs and outputs to interface the design with the real-world environment. The inputs take the form of buttons that the driver can use to manage the cruise control system, as well as inputs from the car describing the accelerator pedal, brake pedal, and car speed. The outputs from the cruise control system are used to set the speed of the car, as well as provide feedback to the driver. These interfaces are laid out with the top-level context diagram in Figure 1.

Table 1: Operation parameters

SpeedMin	30.0 km/h	Min speed for cc operation
SpeedMax	150.0 km/h	Max speed for cc operation
SpeedInc	2.5 km/h	Push button speed increase
Kp	8.113	Constants for car speed regulation algorithm and control
Ki	0.5	
ThrottleSatMax	45.0 %	
PedalsMin	3.0 %	Sensitivity of car pedals

Figure 1: High-level context diagram

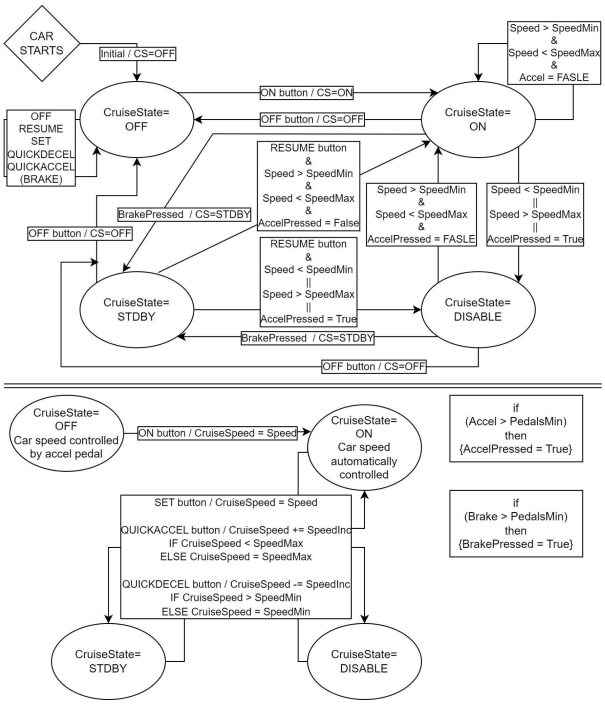


3.2. System design

The operation of our system is defined by the behavioral requirements. When the car starts, the cruise control system should be in the OFF state. In the off state, all inputs from the cruise control buttons should be ignored except for the on button. When the on button is pressed, the cruise state will be set to ON and remain there unless the cruise control operation requirements are broken. For the cruise control system to be managing the speed of the car, the accelerator and brake pedals must not be pressed, and the speed of the car must be within the defined range. If the brake pedal is pressed at any time, the cruise system will enter the standby state and wait to be resumed with the RESUME button. If the accelerator pedal is pressed or the speed ranges are

broken, the cruise system will enter the disabled state until these conditions are reversed. Of course, pressing the OFF button during any state will turn off the cruise control system. Our full-state machine diagram can be seen below in Figure 2 or full-sized in the appendix.

Figure 2: System FSM diagram



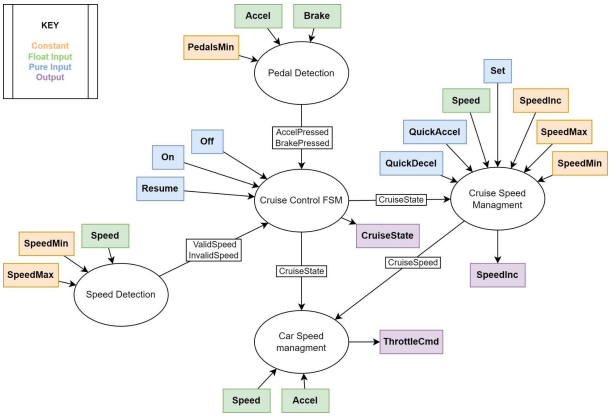
4. Design in Esterel

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc vitae nisl vel lacus fringilla suscipit. Pellentesque volutpat eget lorem nec euismod.

4.1. Design decisions

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc vitae nisl vel lacus fringilla suscipit. Pellentesque volutpat eget lorem nec euismod.

Figure 3: low-level context diagram



4.2. System design

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc vitae nisl vel lacus fringilla suscipit. Pellentesque volutpat eget lorem nec euismod.

4.3. Testing and verification

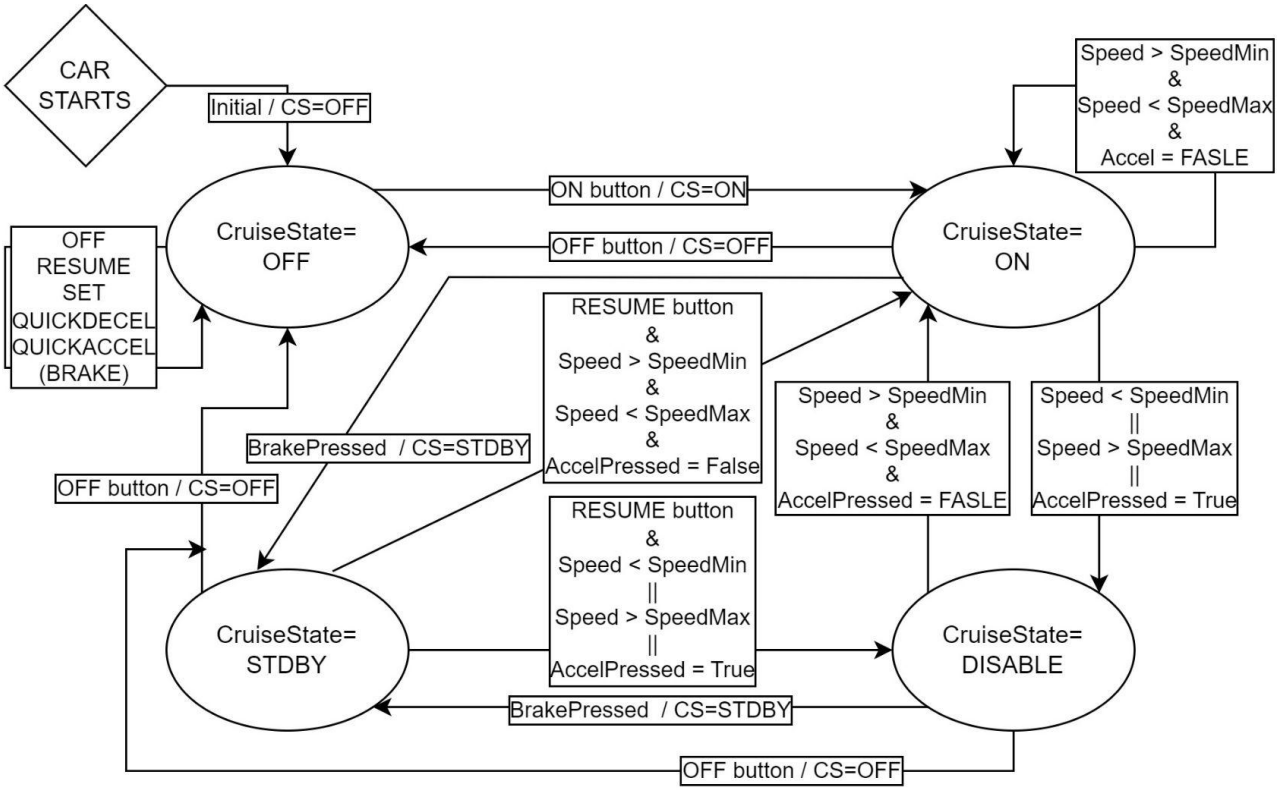
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc vitae nisl vel lacus fringilla suscipit. Pellentesque volutpat eget lorem nec euismod.

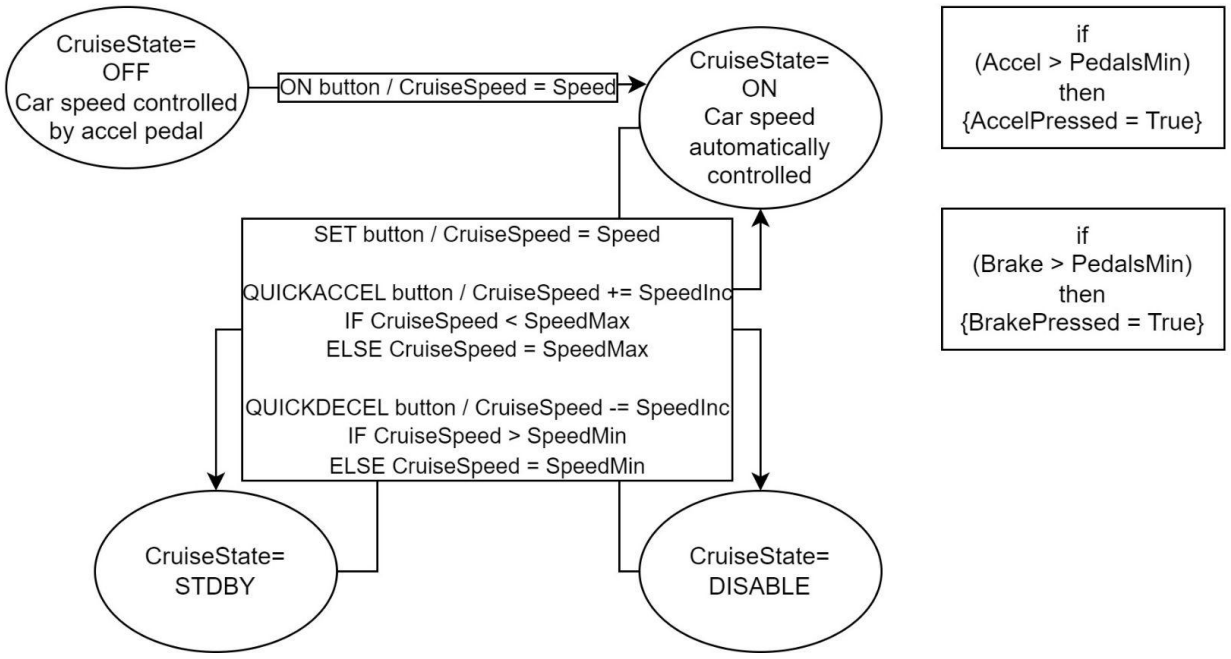
5. Conclusion

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc vitae nisl vel lacus fringilla suscipit. Pellentesque volutpat eget lorem nec euismod.

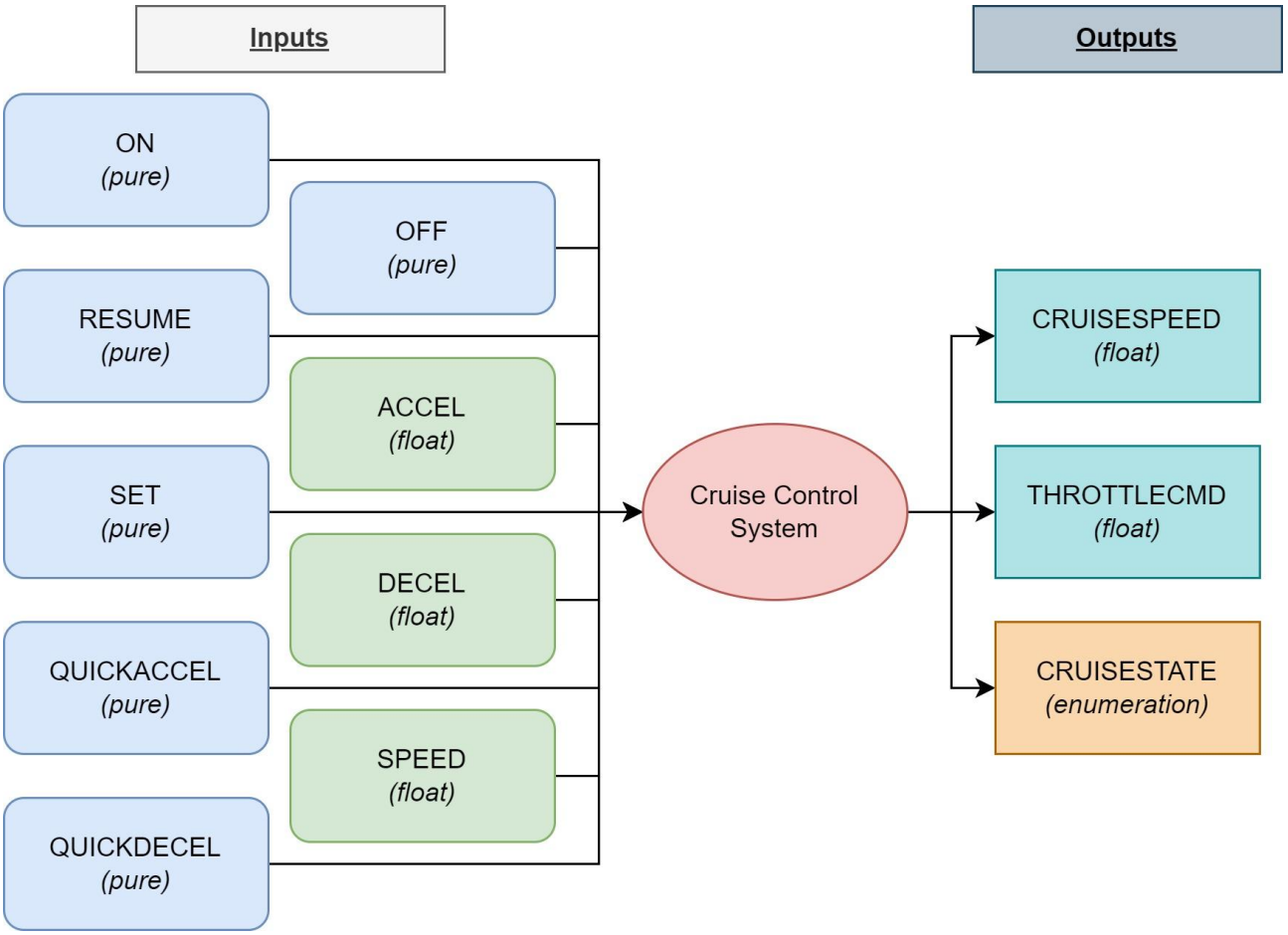
6. Appendix

6.1. Full-size FSM diagram

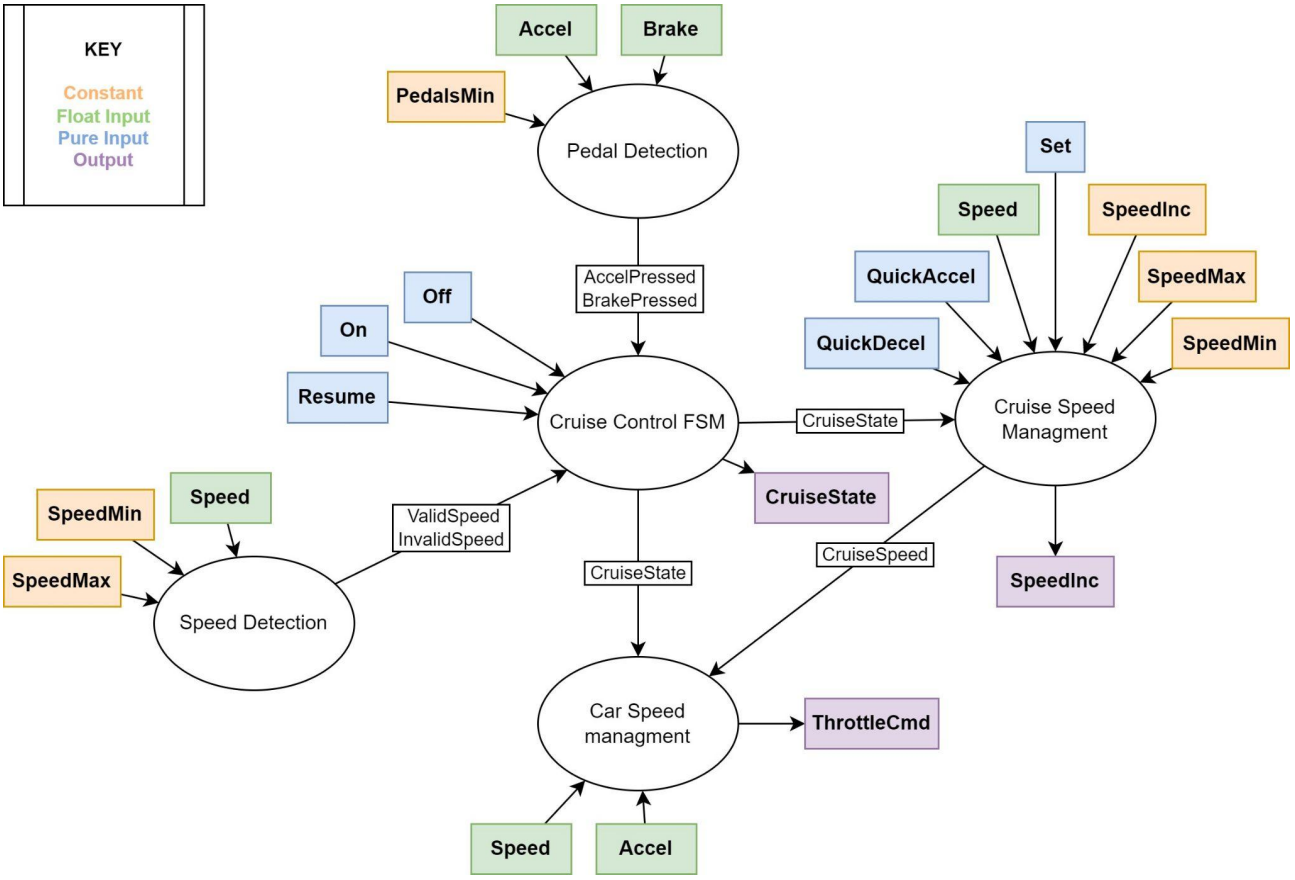




6.2. Full-size top-level context diagrams

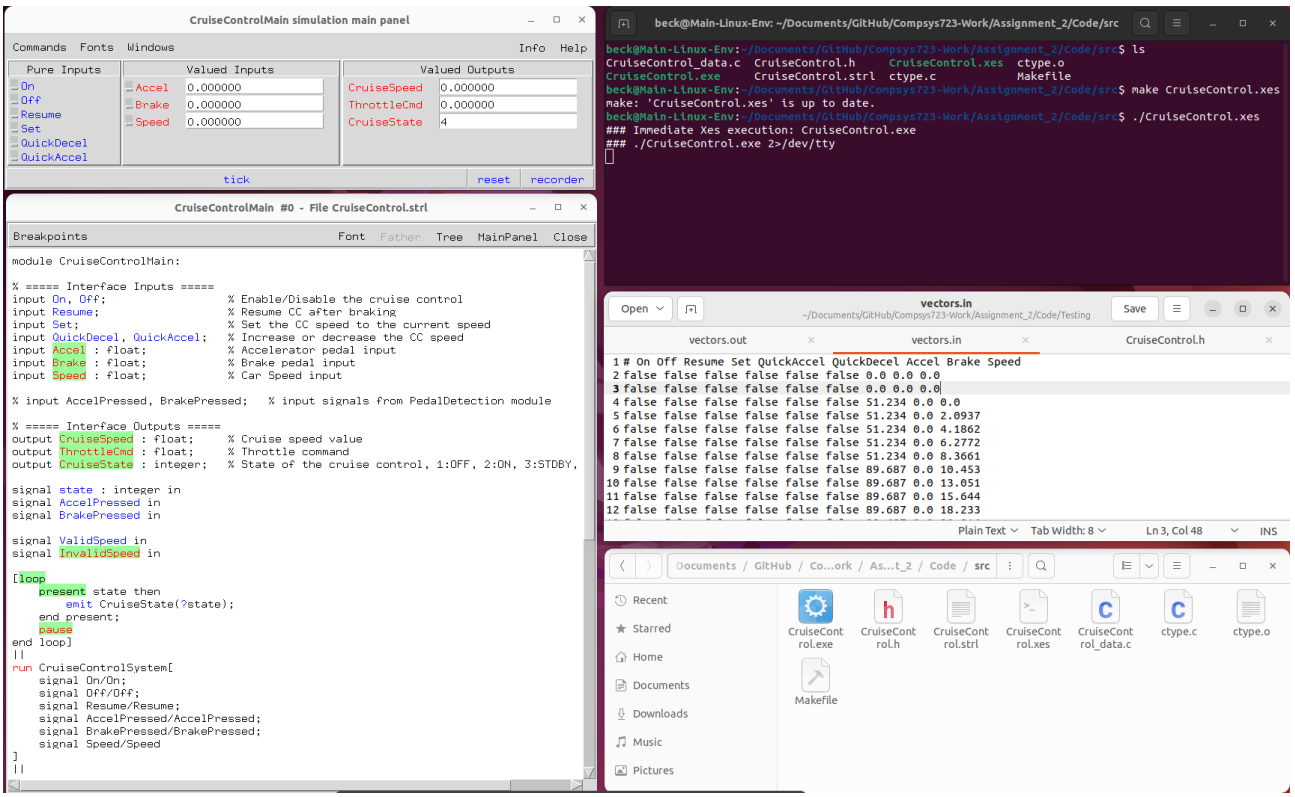


6.3. Full-size low-level context diagram



6.4. Testing Screenshots

6.4.1. Linux testing environment



6.4.2. Testing inputs and results