UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

Student : Rogoz Bogdan Dorin

Group : 30433

Table of Contents

1. Introduction

The purpose of this document is to present the design elements of the CAN Bus simulation application, in order to provide an overview on the final implementation of the backend and user interface.

2. Scope

The elements described in this document are: functional requirements, non-functional requirements, entity modeling, user actions and graphical user interface.

3. Functional Requirements

The user should be able to:

- Add CAN controllers without affecting the other controllers' functionalities
- Remove CAN controllers without affecting the other controllers' functionalities
- Define the behavior of his controllers based on a template
- Load simulations
- Save simulations
- View the bus data and the timestamps

4. Non-functional Requirements

The following requirements should be satisfied:

- The bus and each controller should represent a different thread, in order to simulate the concurrency of the components
- The Observable pattern should be used, in order to simplify data transmission

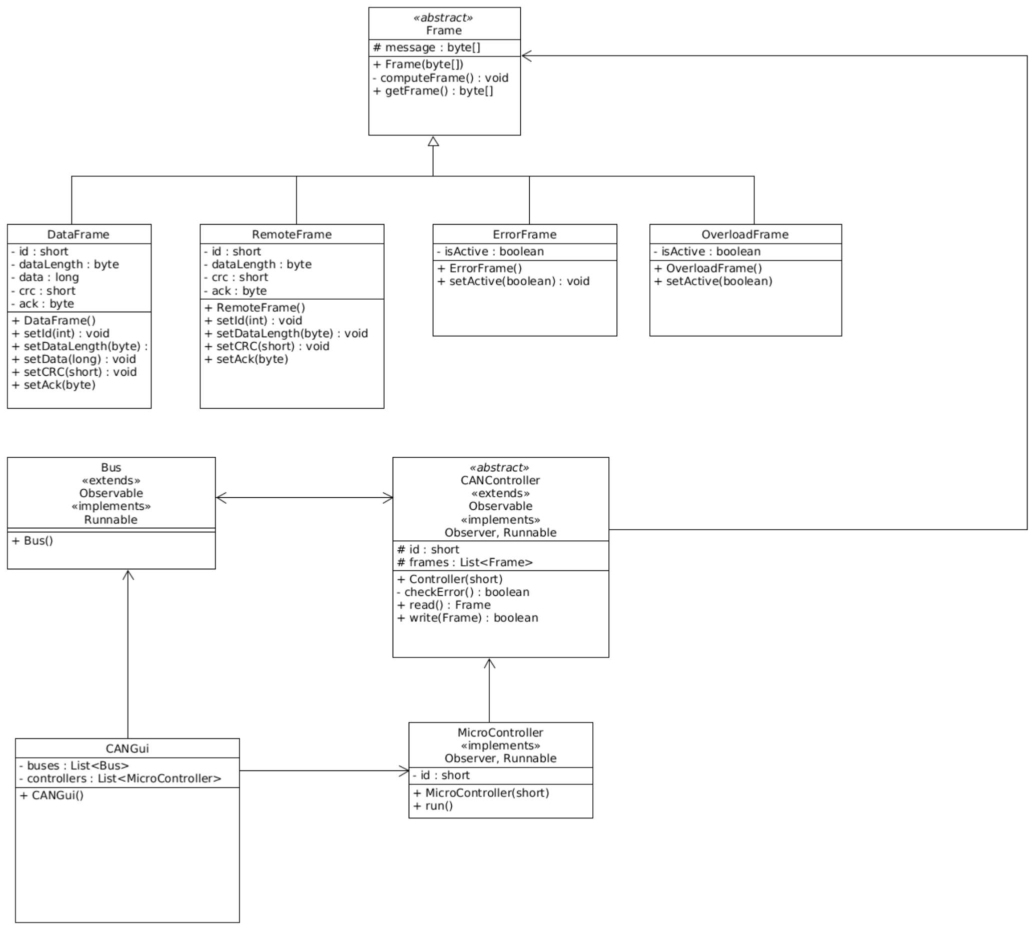
5. Models

The following models are implemented:

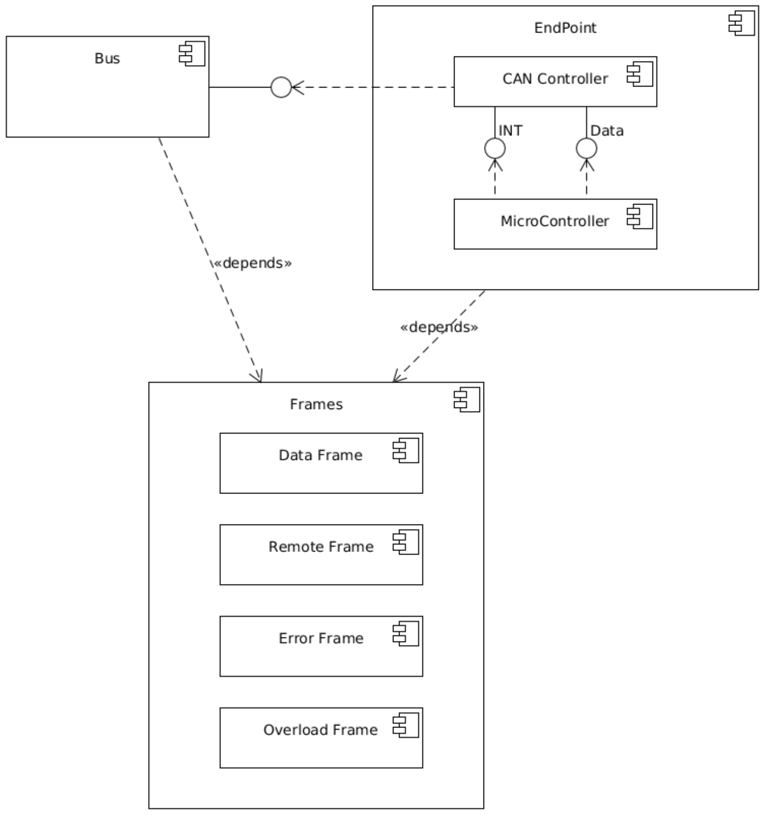
- Frame : represents a "builder" object which transforms a set of parameters into valid CAN frames; a frame can be of multiple types

- DataFrame : represents the data frame used by the CAN protocol. It builds a message based on the fields’ values; it is used for sending data, usually as a response to a request
- RemoteFrame : represents the remote frame used by the CAN protocol; it is used for sending requests to remote controllers
- ErrorFrame : represents the error frame used by the CAN protocol; it is instatiated when an error is detected by the CAN controller
- OverloadFrame : represents the overload frame used by the CAN protocol.
- Bus : represents a CAN bus; multiple instances are possible
- CANController : represents an endpoint of the CAN bus; it reads from and writes to the CAN bus; it is also responsible for checking if the received messages are errorneous
- MicroController : represents a microcontroller which does a certain job; it communicates with the CAN bus through the CAN controller

6. UML Class Diagram



7. Component diagram

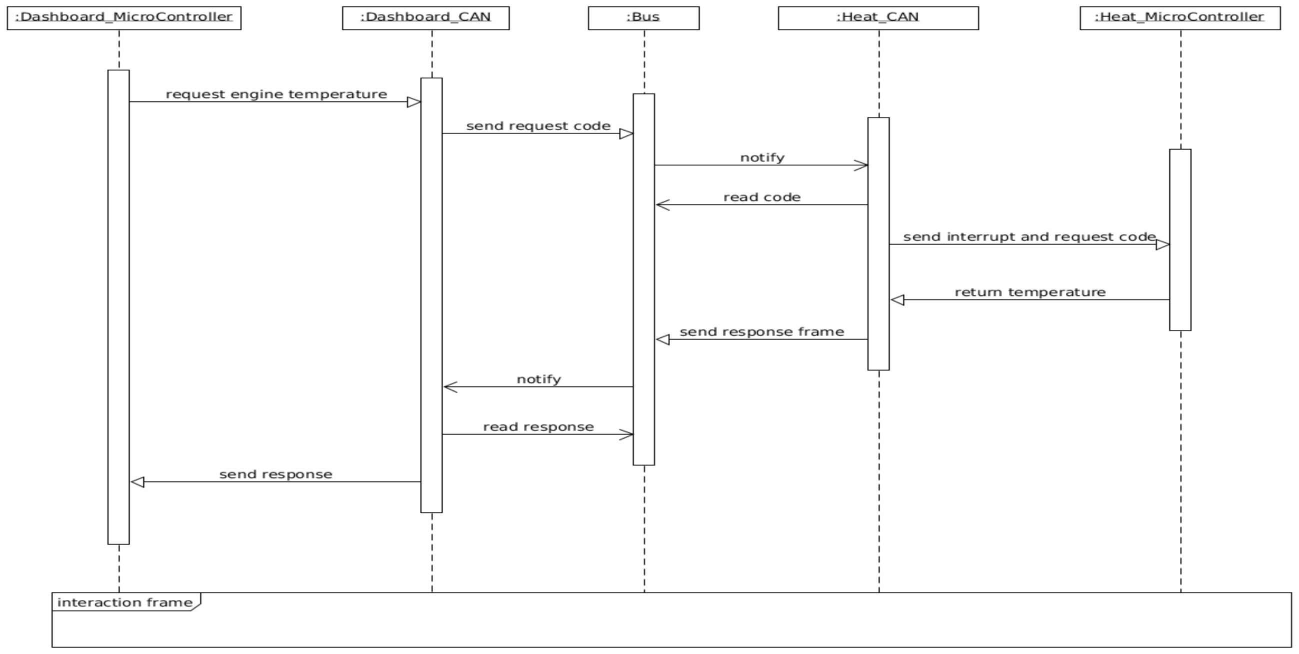


8. Scenarios

An example scenario would be consisting of the following components:

- One CAN bus
- 2 microcontrollers, each one having 1 CAN controller: a microcontroller for an engine heat sensor and the dashboard

At one point, the dashboard sends a message to the CAN bus, asking for the temperature measured by the heat sensor. The heat sensor’s CAN controller receives the message, sends the interrupt to its corresponding microcontroller, then sends the response back to the dashboard.



9. Graphical User Interface

The GUI should be simple, intuitive and responsive. The following elements are considered to be some basic ones:

- Load design button
- Save design button
- Design view, containing all the active components
- Component pane, containing all the user defined components
- Load example design menu, containing a list of predefined designs
- Create component button
- A message history pane, containing a list of all messages transmitted through the bus and their status (SUCCESS, ERROR etc.)

10. Bibliography

http://microcontroller.com/learn-embedded/CAN1_sie/CAN1big.htm

http://www.can-wiki.info/doku.php