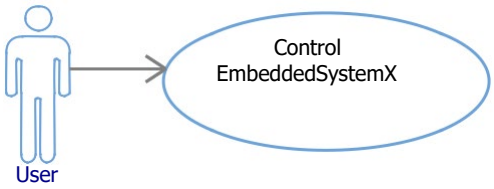
Embedded Real-time Systems
Exercise 1. Modeling and implementation of a system behavior using a State Pattern, Command pattern, Singleton pattern and Active Object Pattern.

Description:
In this exercise, you will implement a concurrent state machine where the control processing (events handling) is implemented using *GoF State Pattern, GoF Singleton Pattern, Command pattern* and *Active object pattern.*

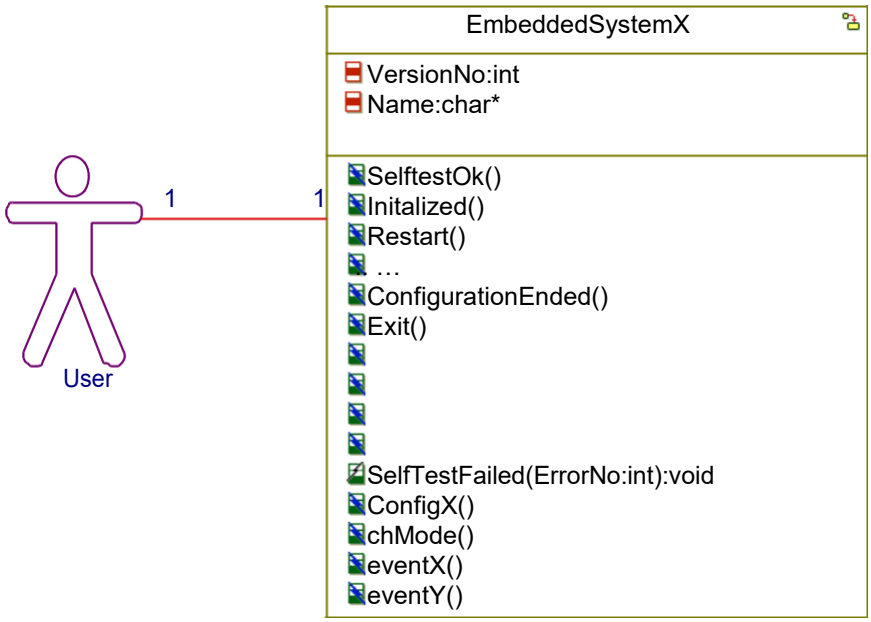
Goals:
When you have completed this exercise, you will
· have experience with implementing the State pattern for hierarchical/concurrent State Machine
· Implement an active object pattern where objects can be created for each request to the active object.
· Experience with implementing command pattern and interconnect it to the state pattern.
· have getting started with using Visual studio or an alternative UML tool to document your design

Exercise 1:

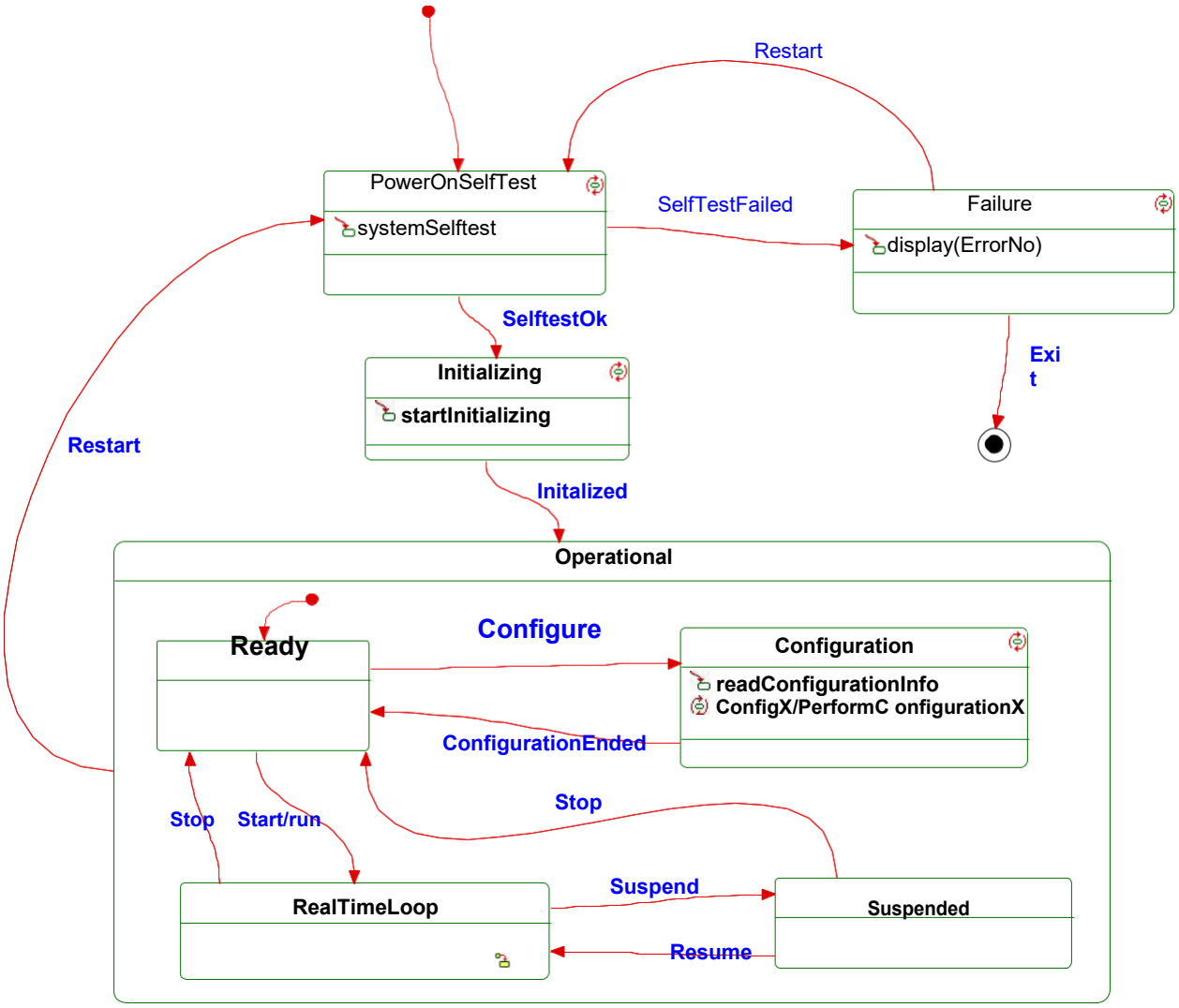
Use Case Diagram:



Class Diagram with event operations:

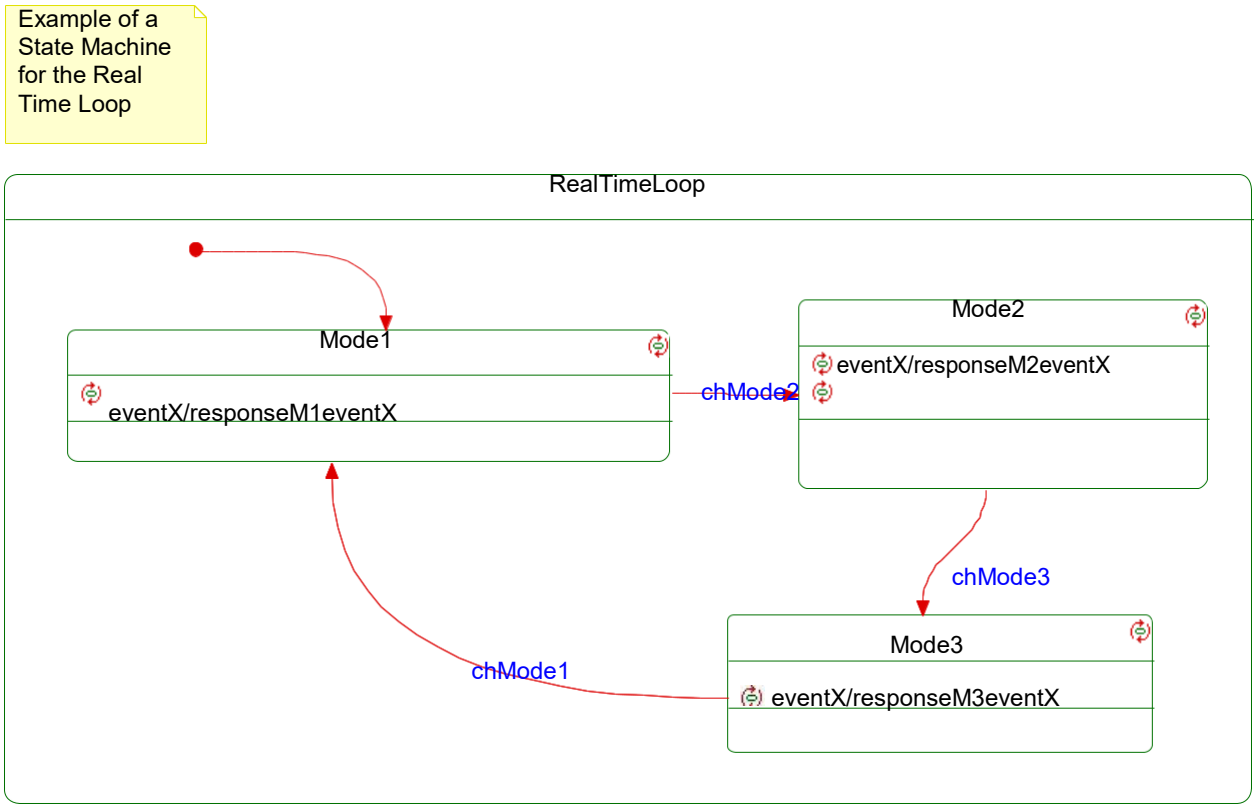


State Diagram of EmbeddedSystemX:



The implementation shall include the following concurrent state diagram for the RealTimeLoop Class.

Sub State Diagram of State RealTimeLoop:



1.1. Design a solution to implement the EmbeddedSystemX behavior as follows:

- Each state is implemented with the Singleton Pattern and the behavior (events processing) is implemented using the GoF state pattern, except the states/events below.

- Events *Resume* and *Stop* of state *Suspended* are commands implemented using the command pattern where each occurrence of such events correspond to create a request instance (event handler), this is basically empty except moving the control to the next state.

- Consider *RealTimeLoop* to be an active object (interface = {*chmode1*, *chmode2*, *chmode3*}) where for each *chmode* event a new request object is created. That request just calls the corresponding handler method from the source state.

1.2. Implement and test the design with an application implemented in C++. You can consider the hardware model provided (**Platform-model**) to run the application on the **Zybo** board (**optional task**).

1.2.1 Insert the class diagram for the solution in Visual studio or an alternative UML tool

1.2.2 Implement the Context Class (EmbeddedSystemX) with the shown event operations and add the necessary operations for implementing the State Pattern. Add a test operation for displaying the actual state.

1.2.4 Test the solution with a main program, where the user activates the public event operations and the actual state is displayed after each invocation.
1.2.5 Report about the challenges and surprises