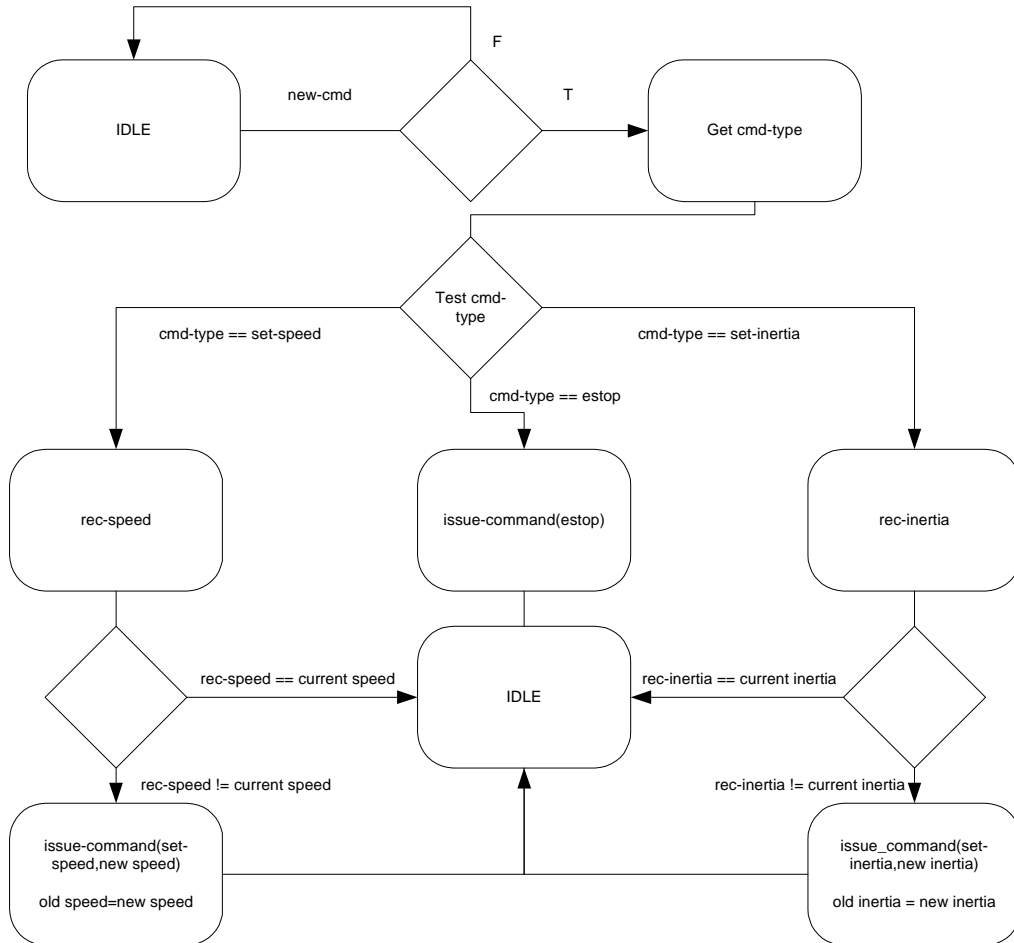


Answers

- Q1-1. The distinction between requirements and specifications stem from the fact that requirements are inherently an informal description of what the product should be. This description should be something that could be given to and understood by a marketing specialist and should include such items as size, weight, speed, UI, and performance. In contrast, specifications are a more formal, precise description of the product. This description should be written so as to allow the system's architects to implement the feature described in the requirements.
- Q1-2. The distinction between specification and architecture is that the specification is a detailed description of how a system behaves. In contrast, a system's architecture describes how a system is actually built. While in many systems a specification may lead directly to one type of architecture, ideally this is not the case.
- Q1-3. We would determine what type of CPU we would use in the architecture design stage of the design methodology
- Q1-4. We would choose a high-level programming language in the architecture design stage of the design methodology
- Q1-5. We would test our design for functional correctness in both the component design/building phase and the system integration phase of the design methodology
- Q1-6. In a bottom-up design, we begin with individual components that could comprise a system and work our way up to eventual complete a full system. In contrast, in a top-down design, we begin with an abstract notion of a device and work our way down to the components that comprise the system. Of course, both methods are similar from the standpoint that the net result is a finished product. However, they both obviously achieve this result in completely different ways. Furthermore, both methods are related since it would be very difficult to do a good top down design without any knowledge of underlying components and vice versa.
- Q1-7. A great example of this is as follows: Suppose that the information obtained from the software programming phase indicates that the application being written does the majority of it's work in floating point. If this knowledge had been known during the architecture design phase, it would be possible to make the hardware more efficient when doing floating-point calculations.
- Q1-8. A great example of this is as follows: Suppose the data I/O design revealed that the device being made needed some high speed external data connection that required a higher bus bandwidth than a regular interrupt driven system can provide. If this info had been known during the architecture design phase, it might have been possible to add something like a DMA controller to interface directly with the data I/O device.

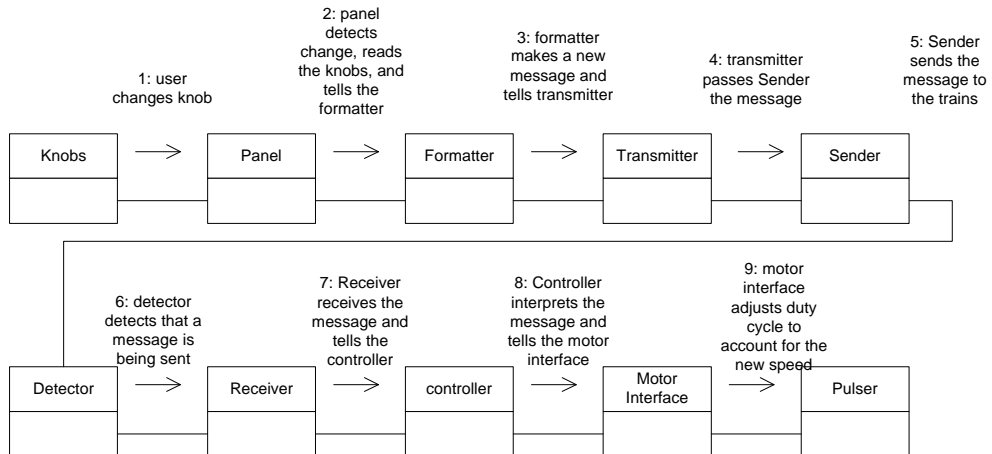
Q1-9. UML state diagram for the receive-message behavior of the controller class:

NOTE: The 2 IDLE states shown below are the same state.

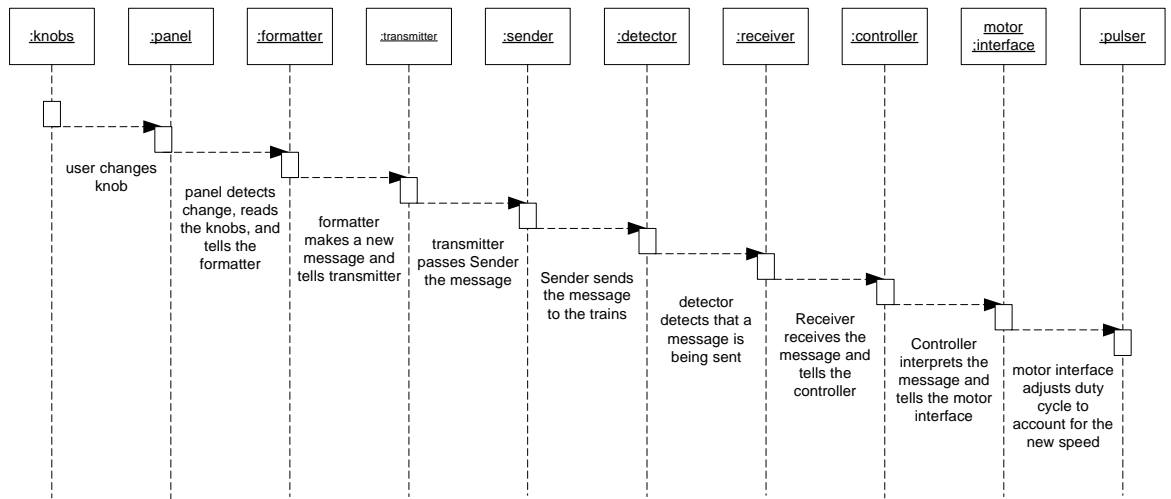


Q1-10.

a. Collaboration Diagram

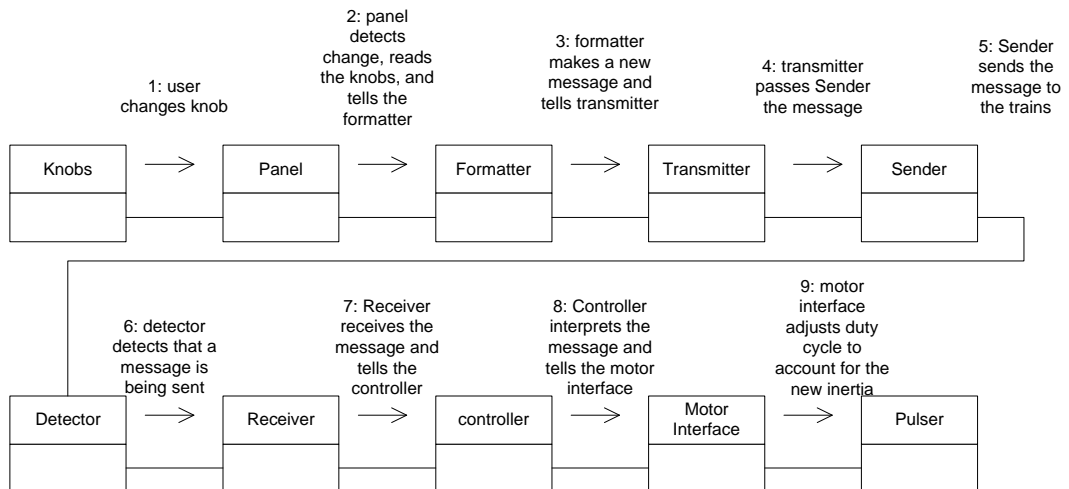


b. Sequence Diagram

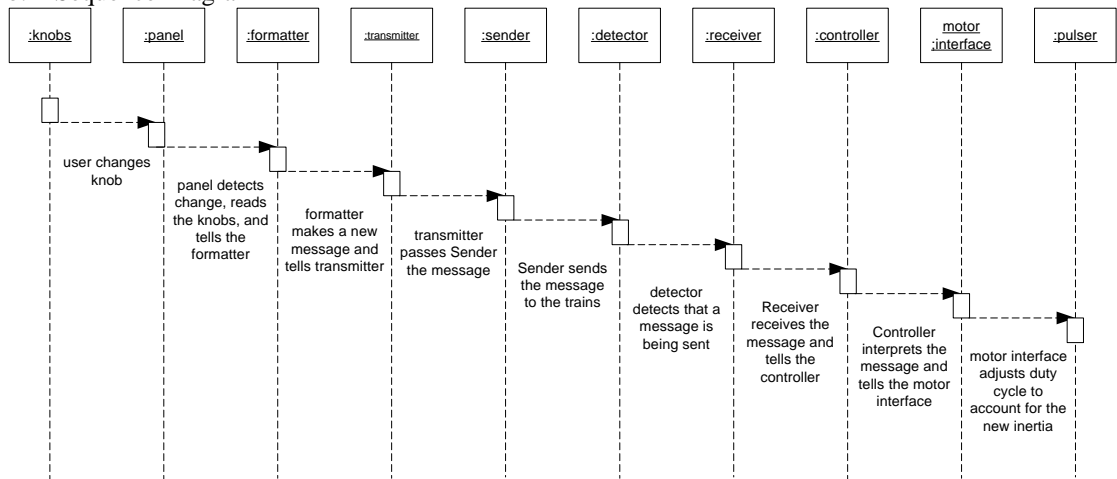


Q1-11.

a. Collaboration Diagram

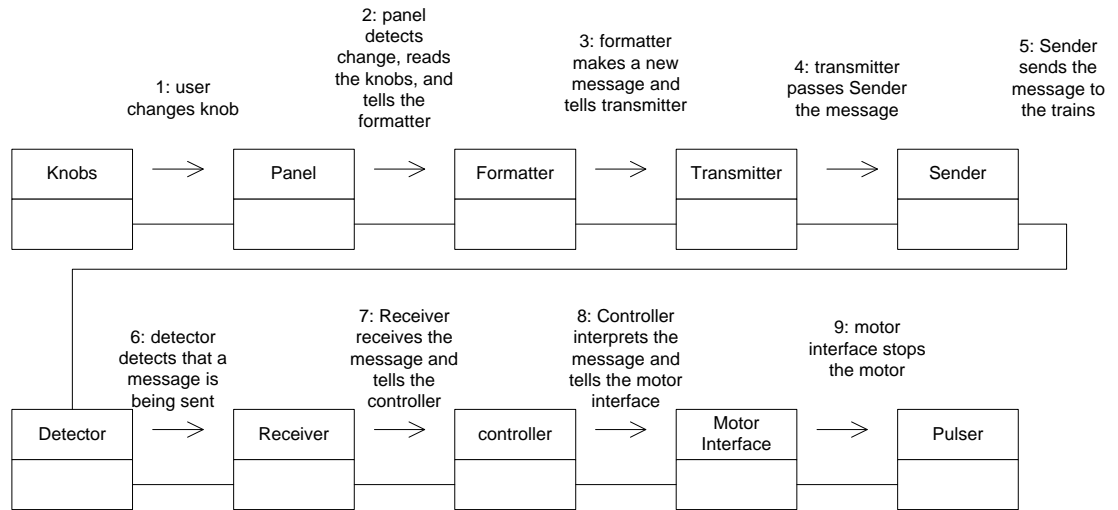


b. Sequence Diagram

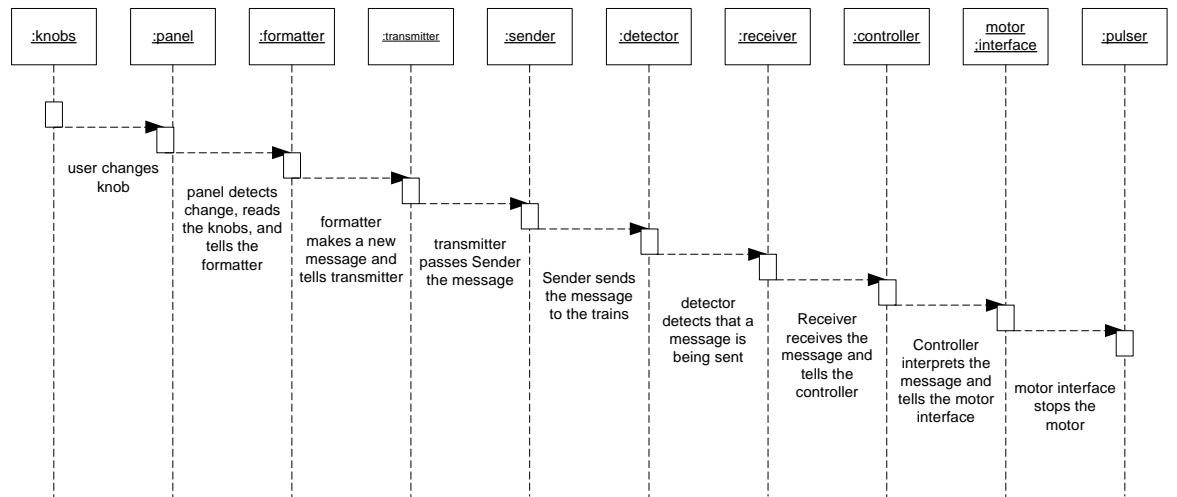


Q1-12.

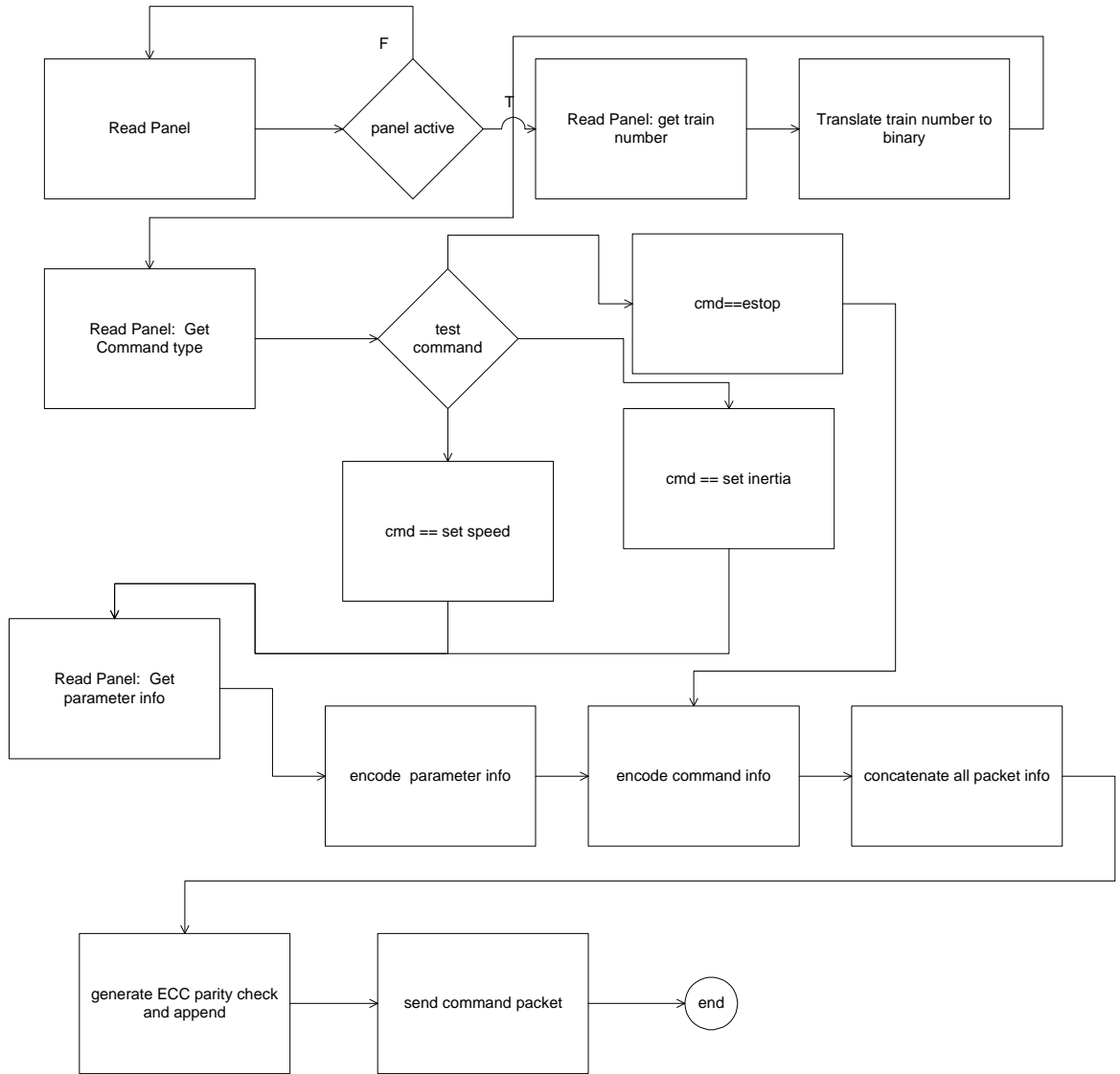
a. Collaboration Diagram



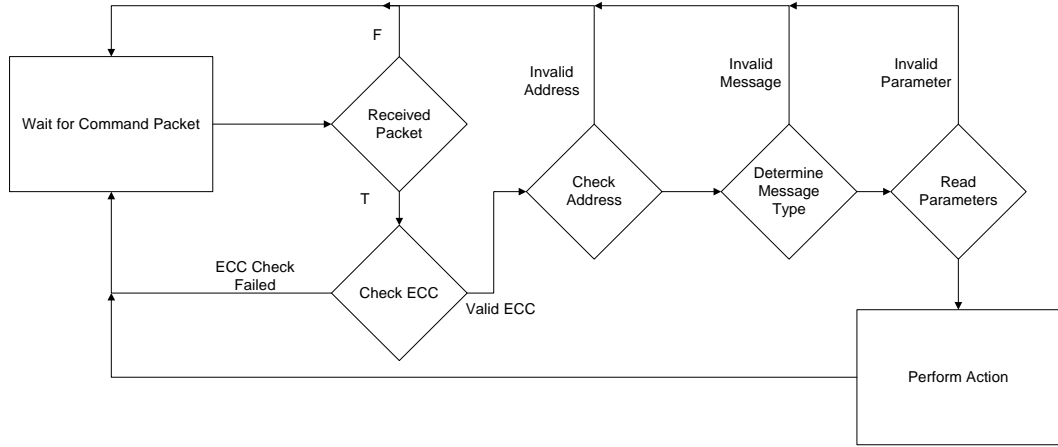
b. Sequence Diagram



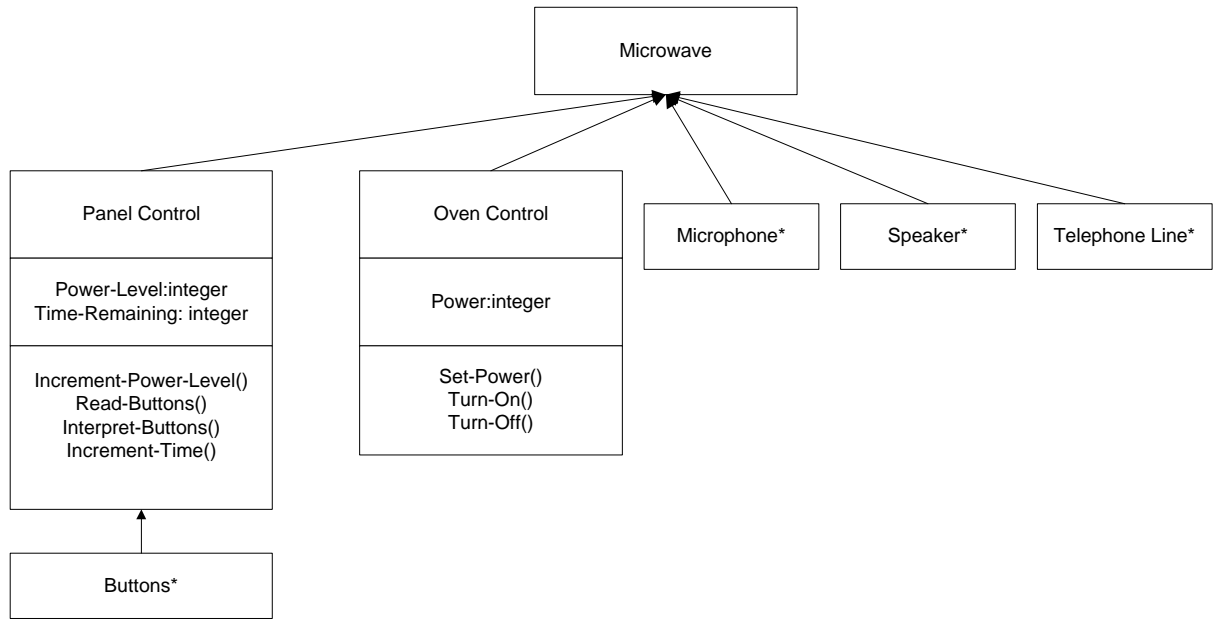
Q1-13. UML state diagram:



Q1-14. State Diagram:



Q1-15. Class Diagram



Q1-16. Collaboration Diagram:

