Goal of HW2 is to familiarize with the concept of platform API, refinement, formal model for system design. General Comments: there is not right/wrong answer to a question, I would rather consider your homework as solutions to design problems. I could argue that your solution is inefficient and you could argue the same about mine. Grade is established mostly on the rationale you follow to answer questions. Hence it is in your interest to justify all your statements. You can use any kind of sources as long as you reference them all.

Problem description: We want to provide a system designer with an environment to describe a control system using codesign finite state machines (CFSM) and then go down to the implementation.

We consider the system as in homework 1, composed of a producer, a consumer, a protocol down-converter and a protocol up-converter.

QUESTION 1a: Describe the four FSMs using the mathematical definition of finite state machine showed in class: define the set of input, output, initial state, transition function and output function.

QUESTION 1b: Describe the four FSMs using the state diagram representation.

We have to provide the designer with a platform to describe codesign finite state machines. Communication services play an essential role here. Assume that there is a language, maybe a graphical language, to describe the behavior of a finite state machine. Designers can draw the state diagram to model the system behavior. We want to provide services for communication between state machines. This problem is similar to the yapi modeling showed in class where we provide the read/write functions as platform APIs.

QUESTION 2: Assuming that CFSMs communicate through a single place
buffer, provide a set of functions to emit an output signal, check if a signal is enabled, read an input signal value and whatever other services you think are needed to build a CFSM modeling environment. Please, comment on the semantics of this functions and on the implementation of them. Also describe the producer/consumer system using the functions you have defined.

**QUESTION 3:** Describe how a simulator will simulate your system on a single processor machine (remember the event driven nature of CFSM model of computation) and write a pseudo-code of the simulator. You may consider iterate between question 2 and question 3 since in order to simulate you may need to modify or add other API functions. Also make sure that the producer/consumer system described in question 2 using your APIs can be simulated with your simulator.

Notice that the simulation is an implementation of the model.

Now we have to march toward the implementation. We go down one more platform. The next platform is the abstract architecture platform. Assume you have a way of synthesizing one FSM. This is actually not difficult to do: there is a lot of literature and actually tools to write VHDL out of a state diagram. Also it is not difficult to write C code out of the state diagram description. Notice that this is different from compilation since we are moving from a formal description to a C or VHDL implementation of the model. The generated code has the same function calls that you defined in question 2. Now that we crossed one platform using synthesis for the single finite state machine, we need two important things: scheduling and communication. In particular we want to be able to synthesize both.

Let’s start with some consideration about interface synthesis. Based on the discussion above, a CFSM is an independent model that can be implemented in hardware or software. Consider the case where two finite state machines are implemented an a processor as two software tasks. You may have overlooked the important fact that emission of outputs and reading of inputs must be atomic operations.

**QUESTION 4** Assuming that there is a construct `ATOMIC{< statements >}` which guarantees the atomicity of the block of statements, write an implementation of the communication functions described in question 2 as C functions. Write them in the case of software-to-software communication.

The previous case was not difficult. Now let’s consider the case where one CFSM is implemented in hardware and the other is implemented in software. For sake of simplicity we consider only the case of software to hardware communication.
QUESTION 5 Write a software interface in C and hardware interface (as composition of latches and gates) that implement the functions in question 2 in the particular case that a software CFSM emits events and an hardware CFSM receives events.

It is also possible to synthesize a real time operating system (RTOS) that is an implementation of the simulator you have described in question 3 but it takes into account real time constraints coming from the upper platform. The last question is about RTOS. Considering your answers to question 4 and 5:

QUESTION 6 Describe a set of RTOS functions (or APIs) that are needed to implement communication between CFSMs and scheduling of them on a single processor.

This last paragraph will just make you reasoning about what you have done.

- Is it possible to automate the path from CFSM specification to implementation?

- If the process is totally automatic, and if the synthesis tools have no bugs, then this process is correct by construction. What happens if the specification of producer, for instance, changes?

- In question 6, you have defined a set of reasonable RTOS APIs. Another step down will implement those services using the services offered by a real RTOS (sometimes it is only a function renaming). What happens if the processor changes or if the RTOS changes?