CIS541: Embedded and Cyber Physical Systems

Spring 2010

Course Project – Milestone 3: Implementation

Due Apr 12, 2010

In this milestone, you will implement the pacemaker controller with the AVR/Butterfly board. A detailed lab manual as well as reference materials will be provided to get you started with the programming environment.

The pacemaker can be functionally viewed as a controller which monitors the heart activities, and responds by stimulating the heart if the heart itself fails to beat at the designated time. To describe the function of such a pacemaker controller, a set of timed automata (which you designed and verified in Milestone 1 & 2) can be used. Implementing the pacemaker controller consists of implementing these timed automata, accepting heart events and sending simulating events to the heart if necessary.

In this project, the heart signals are generated from a simulator running on a separate AVR/Butterfly board. The heart simulator will simulate all possible behaviors of a heart which vary from a normally functional heart which beats regularly to a problematic heart which beats randomly, within specified bounds. For example, suppose the parameters of lower and upper bounds for the next beat interval are 100ms and 1900ms, respectively. Here, whenever a heart beats itself or is paced (at time *t*), the next spontaneous beat will happen after time randomly chosen between (*t+100*)ms and (*t+1900*)ms. Also, the heart simulator will respond to pacing signals from the pacemaker. Once a pacing signal has been received by the heart simulator, there will be two alternatives that we can implement:

1. we assume the pacing has been performed successfully, and the heart simulator will not respond to the pacing signal (sending acknowledgement back immediately), or
2. we assume the pacing has been performed successfully , but will not send acknowledgement.

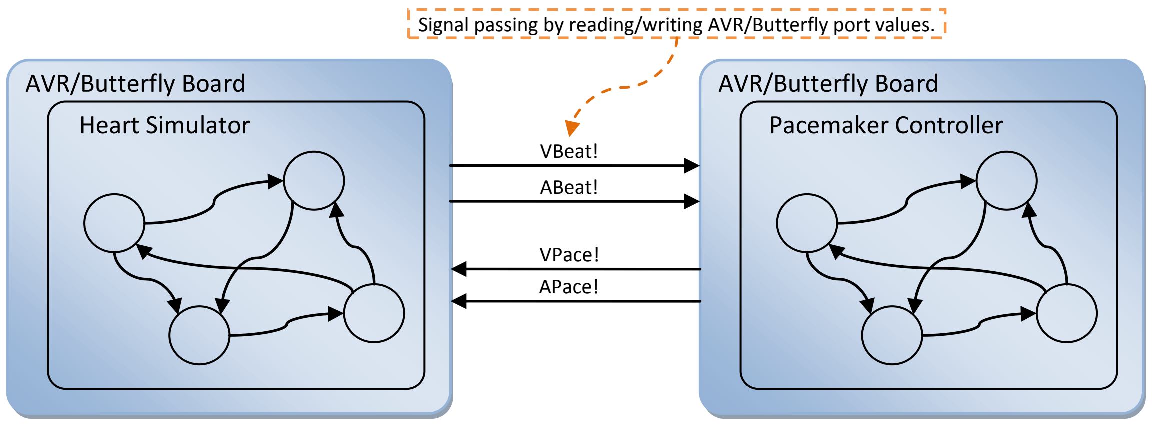
Either of these two assumptions are valid choices, you may choose one and state in your document which you chose.

Note that due to the limited pins the communicating ports have, this signal may not be distinguishable from one representing a spontaneous heart beat signal.

When the heart simulator runs on one AVR/Butterfly board, it outputs signals if the heart beats itself or is paced. To detect this event from the pacemaker controller, either polling or interrupts can be used. With polling, the pacemaker controller periodically checks the communicating ports to see if the port values are changed; with interrupts, the pacemaker controller is interrupted to handle this signal and then resume its execution.

Upon the detection of a heart signal, the pacemaker controller will decide whether (or when) to send a pacing signal to the heart, according to the pacemaker specification. The pacemaker controller will be able to communicate with the heart simulator with two input pins and two output pins. Details are specified in the lab manual.

An illustrational drawing of the architecture is as follows:



Your job in this milestone is to implement the heart simulator and also the pacemaker controller. Though the aforementioned scheme may be one reasonable way, you are not limited to this scheme but feel free to be creative.

To submit your work, write a simple document describing the main logic of your code, as well as the necessary information of how to connect up the two AVR/Butterfly boards provided to you. Zip this document and your project files (source code, hex files, etc.) and email them to Professor Lee (lee@cis.upenn.edu) and Shaohui “Vincent” Wang (shaohui@seas.upenn.edu).