SE 3K04

Assignment 1

Assignment counts 15% of your final grade

Due Date: Oct 20, 2019 @ 23:59

Introduction

There are two major parts to this assignment. The first part involves creating real-time software on the hardware platform. The second part involves creating an initial version of the Device Controller–Monitor (DCM).

Prerequisites

1. Familiarity with the (natural language) PACEMAKER Requirements Document. Specific sections that you need to understand at this stage are:

1	Introduction (italics for sections additional to Assignment 1)
2.1 - 2.2	Overview/Components
3.1 - 3.2	DCM *(English only, 3.2.2, 3.2.3, 3.2.4 only 1 and 2, for 3.2.5 display egrams without markers)*
3.4	Pacing Pulses
3.5 - 3.6	Modes and States
4.7	*Real-Time Electrograms (egrams)*
5	Modes and Their Programmable Parameters
Appendix A	Programmable Parameters

   The document *srsVVI* rev 2 provides formal requirements for VVI mode. You should consult this document if you are unsure of the exact pacing requirements you will need in the future. Also, use this document for a *serial protocol for communications, and details regarding egrams (electrograms).*

2. Familiarity with pacemaker_shield_explained.pdf to understand how the pacemaker shield operates

3. Familiarity with software development on the hardware platform

Part 1 - Pacemaker Design

Use Simulink to implement stateflows for AOO, VOO, AAI, and VVI (Hint: Think of the labs and the charts on simulink). The stateflow should use the programmable parameters listed in the requirements document. You can find this in the 'Prerequisites' section above. Specifically of interest are the pulse characteristics (width, and amplitude), rate characteristics (limits, and delays) and what chamber(s) are being paced. See Part 3 for instructions regarding the pinmap for the two digital output pins (as parameters): one for atrial pacing and another for ventricular pacing. The stateflow should not change if the pinmap is altered, but rather the correct pin should map to its corresponding component.

When debugging it may be helpful to change the colours of the on-board LEDs to reflect the different states in the stateflow.

Part 2 - DCM Design

The DCM is detailed in the natural language PACEMAKER Requirements. There is also information in the *srsVVI* rev 2 document about what specific aspects of the DCM you should concentrate on. You can either use the virtal machines (i.e. VMware) or operate locally on your computers to design and operate the DCM. Your team have the option of programming the DCM in your language of choice. Hint: Python is a common choice to create a simple GUI (with the added bonus of a stable serial communication library, introduced in Assignment 2).

For this assignment, you are required to:

1. Develop an interface that includes a welcome screen, including the ability to register a new user (name and password), and to login as an existing user. A maximum of 10 users should be allowed to be stored locally.

2. Develop essential aspects of the user interface – with respect to 3.2.2 in PACEMAKER, you should include: 1, 2 (input buttons), 3, 4, and 7.

3. Develop interfaces for all of the pacing modes mentioned in Part 1.

4. Make provision for storing programmable parameter data for checking inputs – for the purposes of this assignment the parameters we want specifically are: Lower Rate Limit, Upper Rate Limit, Atrial Amplitude, Atrial Pules Width, Ventricular Amplitude, and Ventricular Pulse Width. The complete set is in PACEMAKER document on page 28.

Part 3 - Hardware Hiding

Each team is given a pacemaker board and shield set that allows you to see the signal as it is output from the pacemaker and into the 'heart' (the pacing output via the black connectors on the shield). We can see these signals using an oscilloscope. You are required to map the correct pins to the Simulink model through hardware hiding (Hint: use a Simulink Subsystem to do this). This will also help you in future labs where the models will increase in complexity. Rememeber that the idea is to abstract away the hardware from the design. Use the large table in *pacemaker_shield_explained* document to help with the mapping between hardware and design.

Attach the Pacemaker Shield to the board and perform VOO functionality, explained in Section 3.5 in the PACEMAKER document and in Table 2: "Bradycardia Operating Modes". Use an oscilloscope to view this waveform being output from the pacemaker (a part of the demo).

Bonus

Can you use the push button to inhibit a pace? Meaning that when an on-board push button is quickly pressed (like a pulse), the pacemaker accepts it as a natural pace. The pacemaker must then only pace after a the proper waiting time has passed.

Example: The pacing rate is 60 bpm (1 pace per second) and the button is pressed after 900 msec of an artificial pacemaker pace then the next pacemaker pace will only show up 1000 msec later (i.e. at 1900 msec after the artificial pace). This is analogous to when the pacemaker detects a natural heart pace and does not interfere with normal cardiac operation (in our case the natural pace is resembled by the button). You must do this without restarting the board, stateflow or model...

If your implemention works then document it! **No bonus without appropriate documentation.**

Deliverables

Submission: You are required to submit a PDF document for each part of the assignment (including bonus if completed). These files must be included in a .zip folder that is submitted to avenue. Your code for the DCM and Simulink programs should be located in a code folder which must also be included in the .zip submission folder.

`Part 1`	`part1_group#.pdf`
`Part 2`	`part2_group#.pdf`
`Part 3`	`part3_group#.pdf`
`Bonus`	`bonus_group#.pdf`
`Code Folder`	`code_group#`
`Submission Folder`	`assignment2_group#.zip`

Note: Each PDF file mentioned above requires a **title page, table of contents and a reference list (if necessary)** in addition to all other requirements mentioned below.

Remember that the purpose of documentation is not to just type it up, it is used to outline your design and help

others understand it. Please do not write excessively, **keep it as short and simple as possible!**

Part 1: For all implemented components you will need to document your design and implement the design in Simulink.

Document the design as follows:

1. List all requirements changes that are likely.
2. List all design decisions that are likely to change.
3. The Simulink diagram must include necessary annotation to understand the model. Provide an additional pdf document that describes design decisions. Provide an additional pdf document that describes testing you performed, and the results.

Part 2:

1. List all requirements changes that are likely.
2. List all design decisions that are likely to change.
3. For each module:
   (a) Describe the Purpose of the module.
   (b) Document the "secret" of the module – if there is one.
   (c) List public functions provided by the module – with their parameters.
   (d) Describe the black-box behaviour of each function (part of the Module Interface Specification).
   (e) Describe any global variables in the module that are within scope for all functions in the module – we call them state variables. You must describe the date structure where appropriate (part of the Module Internal Design).
   (f) List private functions in the module, if any (part of the Module Internal Design).
   (g) Describe the internal behaviour of each public and private function in enough detail that you can code from it with ease. Pay particular attention to how these functions maintain the state variables, or provide the values of state variables (part of the Module Internal Design).

   Describe testing and results of those tests.
   Note: Your documentation is a living document that evolves as you progress through the project. By the end of the project we should be able to see every step that was taken towards completion within the respective documentation.

Information regarding future assignments

1. Improve/finalize the DCM, including communications between DCM and Pacemaker.
2. Use the pacing functionality on the Pacemaker shield and demonstrate "correctness".
3. Implement VOOR mode.
4. Implement VVI mode as specified in *srsVVI*.
5. **Superbonus**: implementation of DDDR mode.

Design Principles

This is a course on software development, and design principles are a huge part of being able to achieve dependable, safe and secure applications. As such, you should pay particular attention to the following design principles as you develop this project in these 3 assignments:

1. Separation of concerns – in particular, effective modularity
2. Your designs must be robust with respect to anticipated changes – apply information hiding.
3. High cohesion and low coupling of modules

Deliverables

1. All design and code in a zip file, submitted on Avenue.
2. Demo in lab

Grading

1. Demo: **40 Marks**

Correctness	30
Discussion	10

2. Documentation: **60 Marks**

Correctness (includes testing)	20
Design Principles	30
Style	10

3. Bonus: **5 Marks**

"Style" includes aspects such as notation, readability, consistency, etc ...