SE 3K04

Assignment 2

Assignment counts 20% of your final grade

Due Date: Nov. 29, 2020 @ 23:59

Introduction

There are two major parts to this assignment. The first part will build on pacemaker functionality implemented in Assignment 1. The second part requires you to extend the functionality of the Device Controller-Monitor (DCM) and introduces Electrograms.

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 to operate the pacemaker shield.

3. Familiarity with software development on the hardware platform.

Part 1 - Pacemaker Design

This assignment uses the model developed in the previous assignment and adds on to its functionality. You are required to implement three different functionalities:

1. Have a working implementation of the following modes using all meaningful programmable parameters: AOO, VOO, AAI, VVI, DOO, AOOR, VOOR, AAIR, VVIR, DOOR.

2. Rate adaptive modes must track activity using the on-board accelerometer. The pulse rate (LRL) should increase once sufficient activity is detected. It is the responsibility of each group to design and justify the rate at which the LRL changes.

3. Implement DCM capability to dynamically change between modes without restarting the device.

Demo scenario for Rate Adaptivity: The pacemaker will be shaken by hand to simulate various speeds of activity; walking, jogging, and running. The device's rate should change accordingly.

Part 2 - DCM Design

The DCM is described in some detail 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.

For this assignment, you should:

1. Expand the DCM to include all required modes and parameters.

2. Implement serial communication to transmit and receive information between the DCM and the Pacemaker.

3. The system will be able to set, store, transmit programmable parameter data , and verify it is stored correctly on the Pacemaker device. A minimal set of the parameter data is specified in the *srsVVI rev 2* document in section 3.1); the complete set is in PACEMAKER, Table 7.

   The following requirement changes are introduced to reflect the specifications of our system and to facilitate demonstrative purposes. Accommodate these changes in your implementation.

Parameter	**Programmable Values**	**Increment**	**Nominal**	**Tolerance**
A or V Pulse Amplitude Regulated	Off, 0.1-5.0V	0.1V	5V	± 12%
A or V Pulse Width	1-30 ms	1 ms	1 ms	1 ms
A or V Sensitivity	0-5V	0.1V	-	±2%

4. The system will be able to display egram data when the user chooses to do so (for either ventricle, atrium, or both). The DCM must receive the egram data from the Pacemaker over the serial communication link in order to display it.

5. Document how programmable parameters originate at the DCM and are implemented in the device. Show how you can ensure the parameters stored in the Pacemaker are what the doctor input on the DCM. Also justify your choice of the data types used to represent parameters data.

What is an egram?

An egram or Electrogram is a visual output of the electrical activity inside each chamber of the heart. For our hardware, pins VENT_SIGNAL and ATR_SIGNAL are electrically connected to the analog egram signals for each chamber.

Part 3 - Testing

The Heartview testing environment allows you to build test cases by configuring natural atrial activity, ventricle activity, heart rate and AV delay to perform functional testing on the pacemaker. You are required to prove correctness in functionality by designing sufficient test-sets for your implementation. When documenting the testing portion provide the following:

1. Purpose/Test Justification, very brief explanation
2. System Input
3. Expected Output
4. Actual Output
5. Result (Pass/Fail)

Bonus

When the pacemaker is performing the DDD operating mode both chambers should be receiving a pace. But remember, both chambers cannot pace at the same time! There must be an **AV delay** between their paces to allow the blood to flow from the atrium to the ventricle.

To complete the bonus you will have to implement two functionalities:

1. DDDR mode with the proper AV delay (don't forget to make it rate adaptive).
2. Inhibit only ventricular pacing when holding down the pushbutton (i.e. atrial functionality is unaffected). Once the pushbutton is released then ventricular pacing will appear.

Deliverables

Submission: You are required to submit documentation as a PDF format. You may either submit separate documents for the DCM and Simulink, or combine the documentation of both software components into one PDF. The file(s) must be included in a .zip folder that is submitted to Avenue by any team member. 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.

`Documentation Folder`	`docs_group#`
`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!**

Simulink Software Documentation: 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 section in the PDF document that describes design decisions. Provide an additional section in the PDF document that describes testing you performed, and the results. If you have completed the bonus, include a section for it in this part.

DCM Software Documentation:

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.

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 (includes testing)	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 ...