PACEMAKER Design Specifications

October 25, 2020

-names of group members here-
- Alex Radikov - 400192066 -
- Muhammad Ali - 400201794 -
-Samuel de Haan - 400083906-

Contents

1 Introduction (brief summary of what we did)

The purpose of this documentation is to describe the functions, requirements and design that went into the Pacemaker. The simulink models, along with our design decisions and requirements will be explored as well as the results of our testing and experimentation. We decided to modularize our design to streamline the system and allow us to easily update the system in the future.

2 Requirements

2.1 Monitored and Controlled quantities

What are monitored variables, what are monitored and controlled quantities,
with suitable names, minimum values, limitations, What are the initialized variables, what are the limitations?

Monitored Variables

Variable Name	Units	Min-Max/Nominal Values	Description
Natural Atrium Pace	Beats per minute BPM	60-100	Natural Atrial pace detected from the heart
Natural Ventricular Pace	BPM	60-100 bpm	Natural Ventricular pace detected from the heart
Mode	-	0,1	0=Atrial, 1=Ventricular
Lower Rate Limit	BPM	60 nominal	Sets the lower rate limit for pacing
Upper Rate Limit	BPM	120 nominal	Sets the upper rate limit for pacing that the pacing rate cannot exceed
Ventricular Pulse	ms		Duration of time that the pacing

Width		0.2-0.6	capacitor discharges into the Ventricle
Atrial Pulse Width	ms	0.2 - 0.6	Duration of time that the pacing capacitor discharges into the Atrium
Ventricular Pulse Amplitude	V	0-5	The amount voltage that is charged to the pacing capacitor, represented as a duty cycle value
Atrial Pulse Amplitude	V	0-5	The amount voltage that is charged to the pacing capacitor, represented as a duty cycle value
Atrial Refractory Period (ARP)	ms	250	The amount of time allocated after an atrial event in which another atrial event will not inhibit or trigger pacing
Ventricle Refractory Period (VRP)	ms	250	The amount of time allocated after an Ventricle event in which another Ventricle event will not inhibit or trigger pacing

Controlled Variables

Variable Name	Units	Min-Max/Nominal Values	Description
Pace Charge Control	-	Low-High	Used to start and stop charging the lead capacitor (C22). When Low, PWM disconnects and capacitor no longer charges
Atrium Pace Control	-	Low-High	Used to discharge the lead capacitor into the Atrium
Ventricular Pace Control	-	Low-High	Used to discharge the lead capacitor into the Ventricle
Pace Ground Control	-	Low-High	To allow the current to flow in either the Atrium or Ventricle
Atrium Pace Ground Control	-	Low-High	Used to connect the Atrial ring to ground to not allow charge buildup
Ventricular Pace Ground Control	-	Low-High	Used to connect the Ventricle ring to ground to not allow charge buildup
Pacing PWM	V	0-5	Used to charge the lead capacitor (C22)
Impedance Atrium	-	Low-High	Allows impedance circuit to connect

Control			to the ring electrode of the Atrium
Impedance Ventricular Control	-	Low-High	Allows impedance circuit to connect to the ring electrode of the Ventricular

2.2 Requirements Likely to Change and results of those changes

Removing constant/preset inputs
In the future we plan to use the DCM as the tool to pick inputs. Therefore we plan to add the serial communication between the DCM and simulink to be a requirement. Hence we will need to replace the current inputs (currently just constant blocks).

Not implementing Upper Rate Limit
According to the pacemaker requirements, the upper rate limit is related to modes with ventricular pacing and atrial sensing. However none of our implemented 4 modes (AAI, VVI, VOO, AOO) pace the ventricle while sensing the atrial nodes. It is unclear in the Pacemaker requirements why URL is a programmable parameter for the 4 modes.

Not implementing PVARP
Similar to the reasons for not implementing upper rate limit, this feature also requires ventricular pacing and atrial sensing modes. However

Not implementing Rate Smoothing
This decision was taken because there was not enough information within the requirements and other pacemaker documents about what this exactly is. Hence it was unclear.

Adding Voltage Logic Level to Programmable Parameters/Monitored Variables
This variable was added in order to convert ventricular amplitude, ventricular sensitivity, atrial amplitude and atrial sensitivity from raw voltage (V) to a percentage within the range 0-100 (appropriate for PWM). More specifically the PWM was set to a percentage by dividing by the voltage logic level. For example if ventricular amplitude is set to 3.7V, and the voltage level is 5V, then the PWM is (3.7/5)*100, or 74.

New modes may be added in the future: DOO, AOOR, VOOR, DOOR, AAIR, VVIR
This was anticipated in advance and was taken into consideration during the design of the stateflow.

2.3 Design Decision Likely to Change and results of those changes

New modes may be added in the future: DOO, AOOR, VOOR, DOOR, AAIR, VVIR

Originally we planned to put all the state flows into the same module. However the model is beginning to get very cluttered. Hence dual chamber pacing should be implemented in its own module.

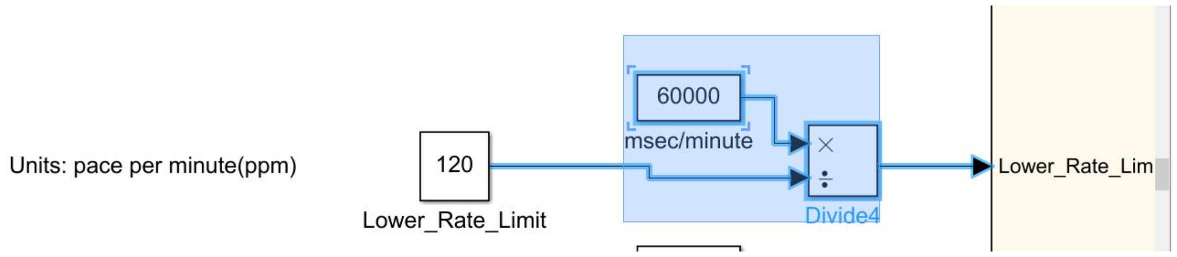
Adding rate modulation can be added by updating our model. Currently stateflow has 4 phases: “discharging C21 and charging C22”, “sensing and response to sensing”, and “pacing”. A 4th phase will need to be added to the model called “modulation”. This will give rate modulation to all 4 of our current modes.
That does not mean that we regret the design decision to make a single module for the 4 modes, as explained in the design decisions.

However in order to avoid complexity in our one module, dual modes will have to be implemented in a separate module from scratch. This is a drawback of using this design.

<u>Replacing the constant block inputs</u>
In the future we will add DCM serial communication, so the constant blocks will likely be removed.

1. Luckily we did not spend too much time organizing the input blocks, so we saved development time.
2. In our model, in order to implement a mode, all parameters must be set, even if they are not applicable to a specific mode. For example ventricular pulse width must be set for the mode AOO, which has nothing to do with the ventricular side of the heart. Fortunately the DCM will fix this problem, by showing only the applicable programmable parameters (given a mode) in the user interface. Hence a design problem will be fixed

<u>Replacing division and multiplication blocks for inputs, with matlab functions, or by performing the calculations within the DCM (with python)</u>
Currently to perform calculations on the inputs, most of them occur along the line from the constant block to the model input. An example is shown below, where we convert Lower_Rate_Limit from units ppm to units msec:



By using matlab/python functions we will increase modularization, and decrease clutter in the current model. This was not implemented currently because we don’t yet know how to interface matlab and simulink, and because DCM imulink communication has not been implemented yet.

3 Design

Reading and Understanding the Design

Modular Design
All of the 4 pacing modes are implemented in the same module, called Assignment1.slx. To use any of the states, simply change the first 3 inputs (Mode_Chamber_Paced, Mode_Chamber_Sensed, Mode_Response) to whatever valid mode you desire.

Reading stateflow vertically
The stateflow is organized in transitions from the top (starting state) to the bottom.
When reading from top to bottom there are roughly 4 phases, however there is no clear boundary between them. First we charge C22, then discharge the C21, sense the heart (if needed for the mode) and then pace.

Reading stateflow horizontally
States in the center column of the stateflow are those states that are shared by both all 4 modes (ventricular and atrial). Any state to the left (so states in the left column) correspond to states for atrial modes only. Any state to the right (so states in the right column) correspond to states for ventricular modes only.

Annotations and comments
We choose not to add a significant amount of comments and annotations. This is because we have over 10 states in the model, each with a few lines of code. Because each state requires a name in simulink, each of the states has a descriptive name for what it does. Hence the state names are the only annotations needed for understanding the model. Additional comments were not added to avoid too much clutter/words.

Units and names are provided beside each of the constant block inputs

Design

To better understand the role of each variable controlling the mode the heart is current pacing. This tabular expression is provided. Other variables do not control pacing modes and were left out to show how each specific mode is achieved.

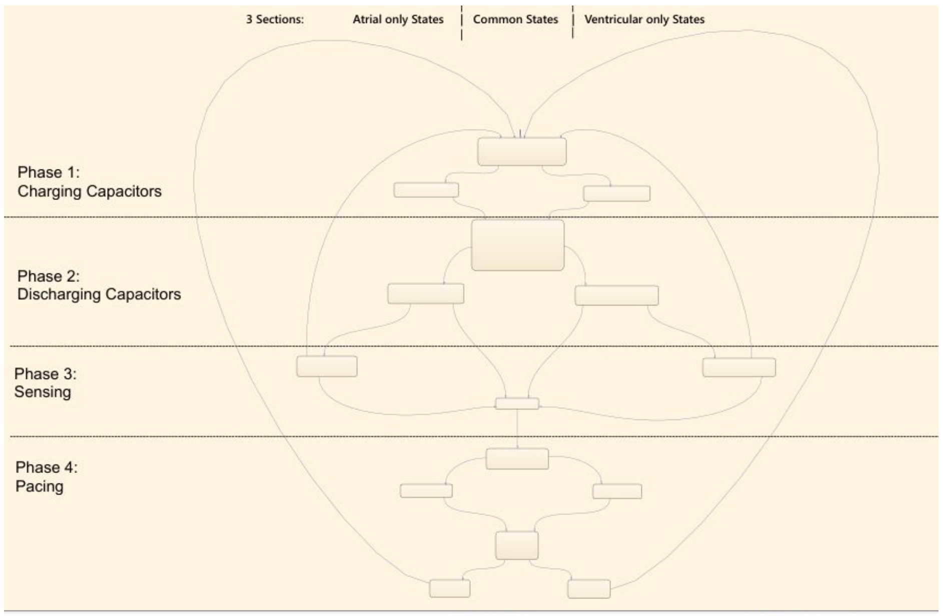
Assignment 1 Specifications Tabular Expression

Chamber Paced=1	Chamber Sensed=0	Inhibition Response=0	AOO Pacing

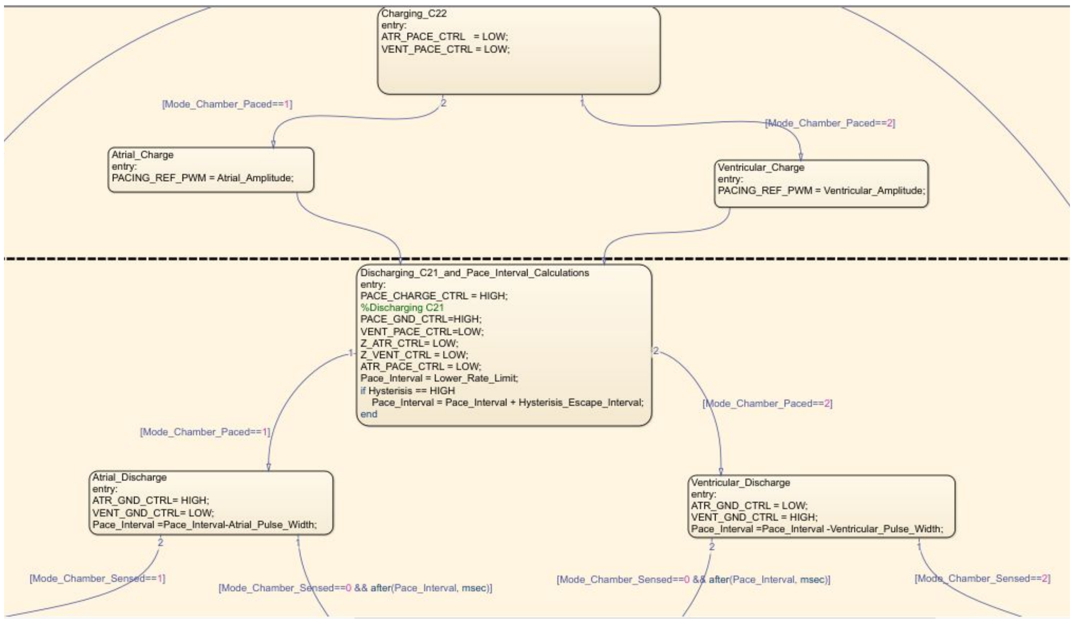
	Chamber Sensed=1	Inhibition Response=2	AAI Pacing
	Chamber Sensed=0	Inhibition Response=0	VOO Pacing
Chamber Paced=2	Chamber Sensed=2	Inhibition Response=2	VVI Pacing

A better understanding can be gained by looking at Assignment1.slx instead of the following screenshots.

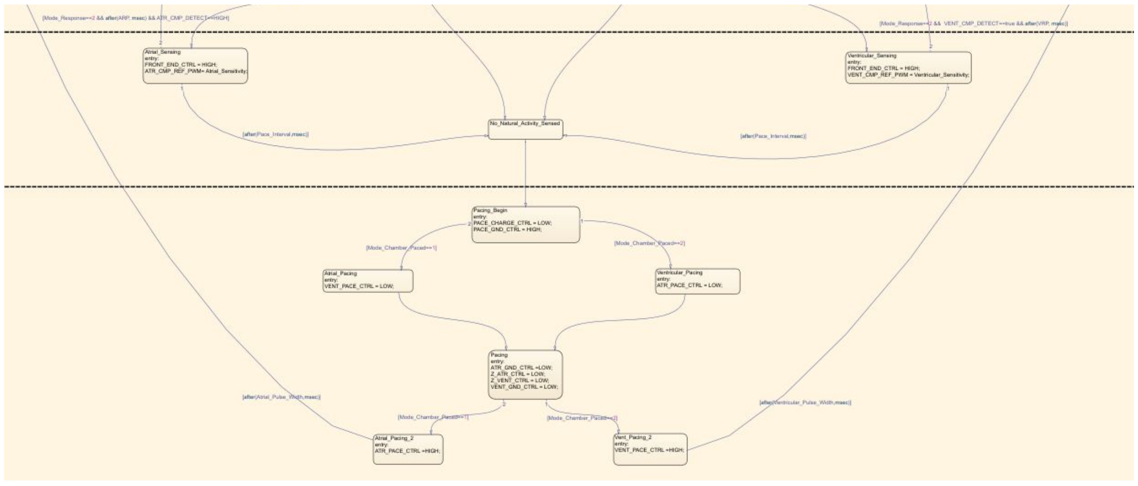
Overview:



Phase 1 and 2 close up:



Phase 3 and 4 close up:



State Transition Table

Attached in a separate excel spreadsheet and pdf document.

Design Decisions

Single module for all 4 modes

Originally we planned to put all the state flows into the same module. This was because

1. We did not know many simulink constructs, especially about how to use library functions, integrate simulink with matlab and importing/exporting modules in simulink.
2. All 4 modes are very similar, hence we thought that to minimize coupling (inter module communication) we should put them in a single module.

3. A single module increases understanding of how the various modes relate, are similar and are different.
4. Minimize code duplication. A lot of the pacing modes contain the same similar code. Making a single module limits code duplication.
5. Making all of the models in a single module allowed easy change and verification. For example if we add hysteresis to the modules, then all 4 modes get hysteresis capability, rather than us having to change 4 modules.

   This was very helpful because we were trying things out, we often made incremental changes. Oftentimes we added a small change, finding out it doesn't work and then removing that change. If we have multiple modules we may have implemented something, then added them to all 4 modules, then if we found it was incorrect then we would have had to change all 4 modules again.
6. It forced us to think about the impact of our design decision on all of the states ("Design goal for family"), not just each of them at a time (Sequential completion).

Compile time: One drawback of having a single module is compile time (about 1.5 min). We should find a way to minimize compile time if we have time at the end of the project.

Safety Critical Design Decisions

Code Duplication: in order to minimize error, software engineer best practice is to minimize code duplication. As writing more repeated code increases the chance of human error in typing the code (as humans tend to have less focus on repeated tasks). Furthermore when code is reused, there is a greater chance of noticing an incorrectness (if there is any).

We made "minimized code duplication and maximize code reuse" a central pillar in our design. Code that is common to all states, is implemented in states located in the center column. Every transition away from the center eventually returns to the center. We have over 5 states in the center, so about half the states/code is shared and reused by all the modes! The other very obvious way we maximized code reuse is implementing all the modes in one module, so all 4 modes use the same states.

Readability and Understanding: In safety critical systems have a model that is easily understandable is important for verifying correctness. This is because it is easier to detect error when we understand the model.

Models that have *symmetry* allow developers to easily understand the relationships between the modes and the overall design strategy more easily. Symmetry helps in detecting errors, since if there is an error on one side it will likely appear on the other (symmetric) side as well.

We implemented *symmetry* by having atrial only states on the left side of the state flow, and ventricular only states on the right side of the stateflow. Each of the sides are symmetric!

Models that are more visual also help in reading/understanding the model. They also help to verify correctness, as bugs are harder to find within lines of code then they are visually.

We implemented this by adding many states, instead of programming constructs such as if statements and loops. We have only 1 if statement, which could easily be replaced with a state if we had more time.

4 Simulink Testing

The first 3 programmable parameters control the mode. They perform correctly, as they are implicitly tested in every test we do (every test has some pacing mode). According to the following tabular expressions, we can see what each controlling variable will do. Other Programmable parameters will be changed to test the versatility of the modes.

Mode_Chamber_Paced=1	Mode_Chamber_Sensed=0	Mode_Response=0	AOO Pacing
	Mode_Chamber_Sensed=1	Mode_Response=2	AAI Pacing
Mode_Chamber_Paced=2	Mode_Chamber_Sensed=0	Mode_Response=0	VOO Pacing
	Mode_Chamber_Sensed=2	Mode_Response=2	VVI Pacing

If a programmable parameter is not mentioned, then assume it is not applicable.

Test one and two will be used as the primary reference graphs for future tests.

<u>This table provides each tests parameters in tabular form</u>

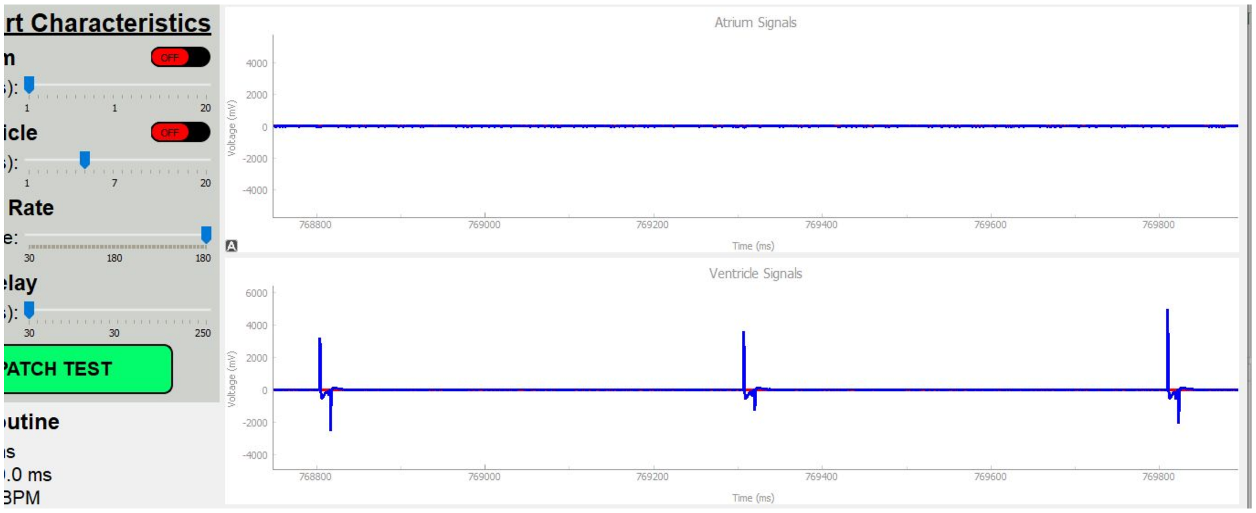
	Test1 VOO	Test2 AOO	Test3 AOO	Test4 VVI	Test5 VVI	Test6 AAI	Test7 AAI
Mode_Chamber_Paced	2	1	1	2	2	1	1
Mode_Chamber_Sensed	0	0	0	2	2	1	1

Mode_Response	0	0	0	2	2	2	2
Lower_Rate_Limit	120	60	60	120	120	60	60
Atrial_Amplitude	-	1.5	1.5	-	-	4	4
Atrial_Pulse_Width	-	20	20	-	-	10	10
Atrial_Sensitivity	-	-	-	-	-	3.5	4
Ventricular_Amplitude	4	-	-	4	4	-	-
Ventricular_Pulse_Width	10	-	-	10	10	-	-
Ventricular_Sensitivity	-	-	-	4	4	-	-
Natural_Heart_Rate	-	-	180	180	119	119	119
ARP	-	-	-	-	-	250	250
VRP	-	-	-	250	250	-	-
Hysterisis	-	-	-	0	0	0	1
Hysterisis_Escape_Interval	-	-	-	-	-	-	10

VOO

Test 1

Goal: To test if the VOO mode functions with the correct. Also to check if the ventricular amplitude, pulse width, and lower rate limit work correctly.


rt Characteristics

n OFF

;): 1 | 1 | 20

icle OFF

;): 1 | 7 | 20

Rate

e: 30 | 180 | 180

elay

;): 30 | 30 | 250

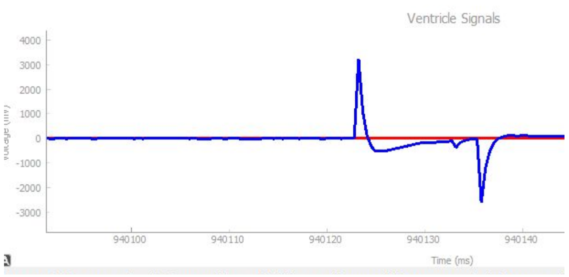
ATCH TEST

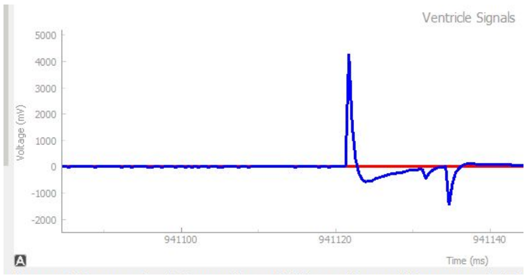
utine

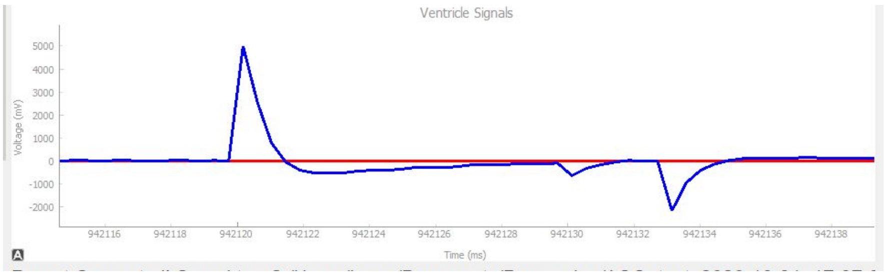
ıs

.0 ms

BPM







Given the lower rate limit chosen, the difference between successive paces should be 60000/lower_rate_limit=500 msec.This looks accurate with regards, at least in the time scale provided by heart view.

The amplitude was supposed to be 4000 mV, the three pulses are close, the first pace is about 3500 mV, 4000 mV and 5000 mV. Hence although the average is close to 4000 mV there is significant variance, so the voltage amplitude is imprecise.

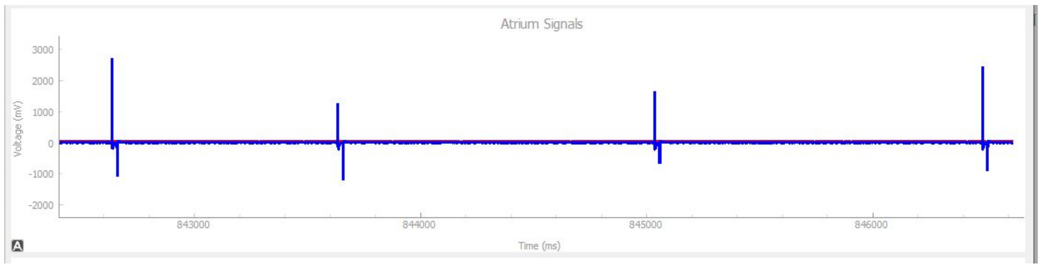
The pulse width looks to be a little bit longer than the correct value 10 msec, closer to 12 msec.

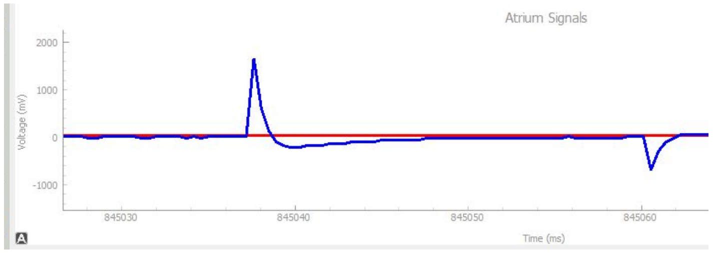
<u>Verdict: Pass // however significant imprecision was detected</u>

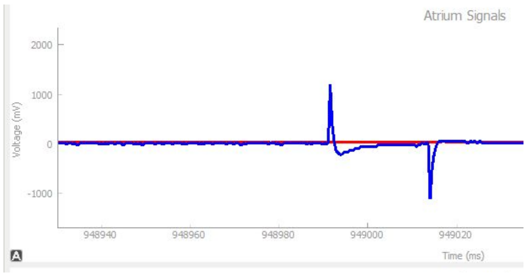
AOO

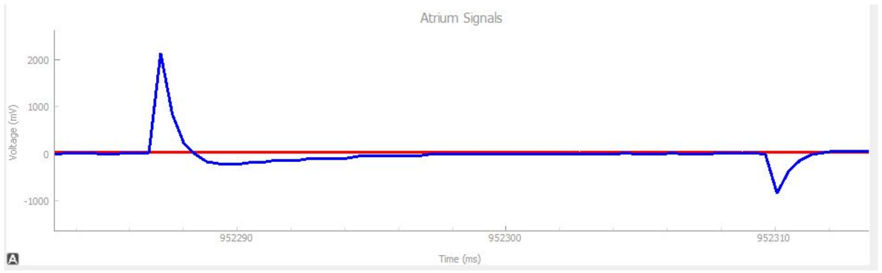
Test 2

Goal: To test if the AOO mode functions with the correct. Also to check if amplitude values, pulse width, and rate can be modified (i.e. use different input values then test 1).









The lower rate limit, or the time interval between pulses is close to 60000/lower_rate_limit = 1000 msec, at least with the accuracy of heartview.

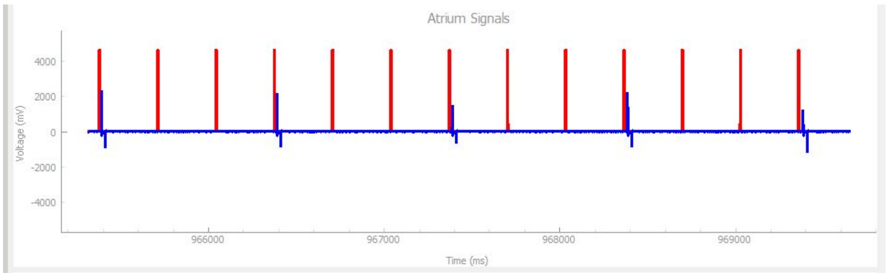
The amplitude is inconsistent, one pace being 1000 mV, 1500 mV and 2000 mV. Hence although the average is correctly 1500 mV, the amplitude is very imprecise.

The pulse width is slightly longer than the desired 20msec, closer to 23 msec.

<u>Verdict: Pass ; accurate but with significant imprecision.</u>

Test 3

Goal: Check if there is no response to natural pace, given that we are in the no response mode.



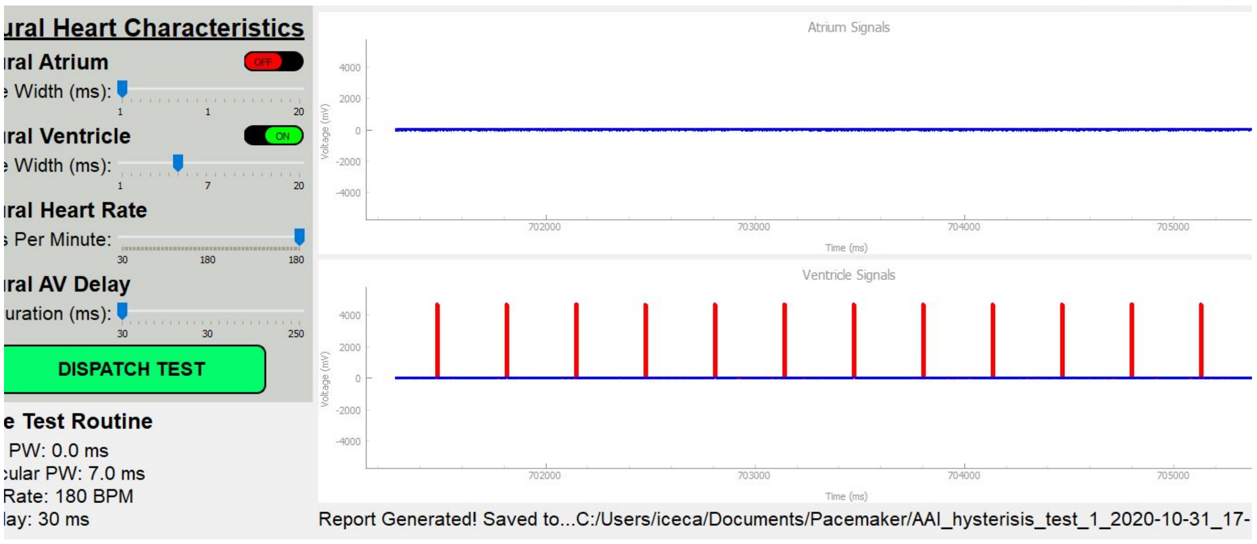
Even with 3 natural paces (every 333 msec) between successive paces (every 1000 msec), the results are the same as the from Test 2. Hence the pacemaker ignored the natural heart activity.

Verdict: Pass

VVI

Test 4

Goal: to test if inhibition works. This will be done by putting in an extreme natural heart rate, and hence every pace should be inhibited.



e Test Routine
PW: 0.0 ms
cular PW: 7.0 ms
Rate: 180 BPM
lay: 30 ms

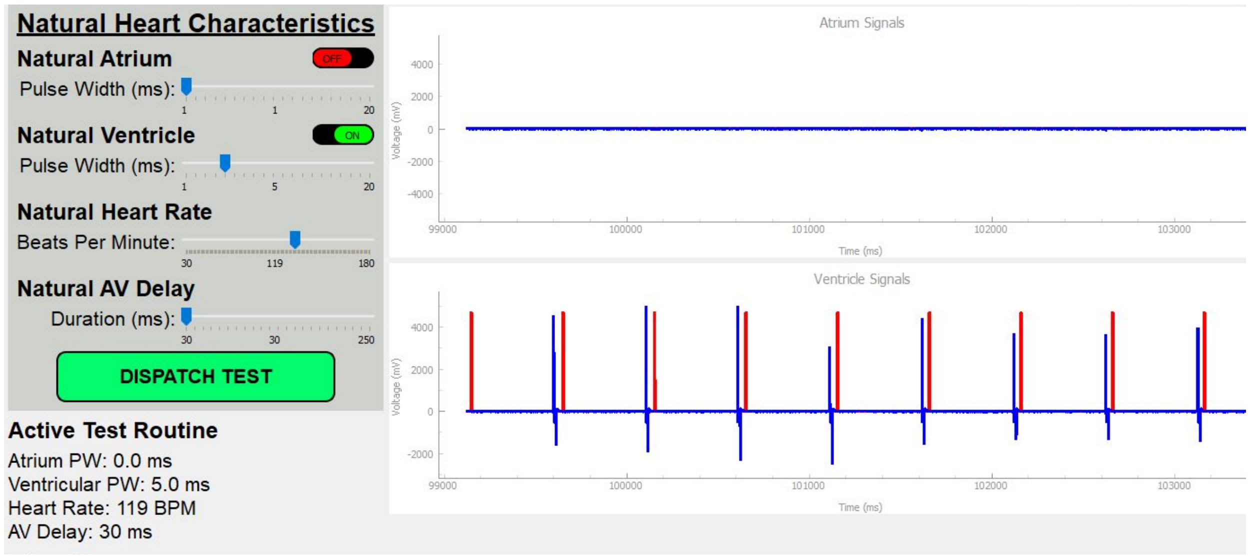
Report Generated! Saved to...C:/Users/iceca/Documents/Pacemaker/AAI_hysterisis_test_1_2020-10-31_17-

Because the pacing occurs every 333 msec, there is almost always a pace between the refractory period on the lower rate limit (between 250 and 500 msec), and hence every potential pace is inhibited. This is seen evidently in the graph.

Verdict: Pass

Test 5

Goal: to test if the refractory period is working, that is if a natural pulse is sensed within a pacemaker pulse and the ventricular refractory period, then the pacemaker should not inhibit.


Natural Heart Characteristics

Natural Atrium OFF

Pulse Width (ms): 1

Natural Ventricle ON

Pulse Width (ms): 5

Natural Heart Rate

Beats Per Minute: 119

Natural AV Delay

Duration (ms): 30

DISPATCH TEST

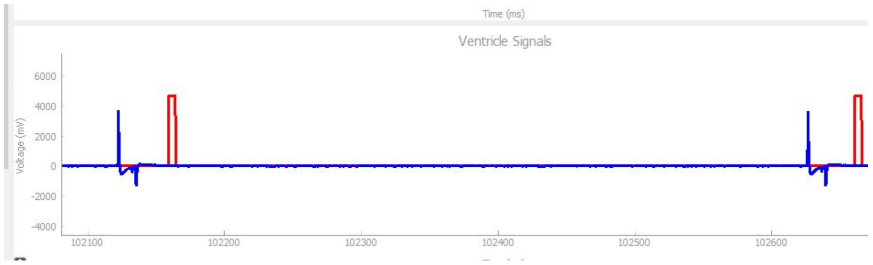
Active Test Routine

Atrium PW: 0.0 ms

Ventricular PW: 5.0 ms

Heart Rate: 119 BPM

AV Delay: 30 ms



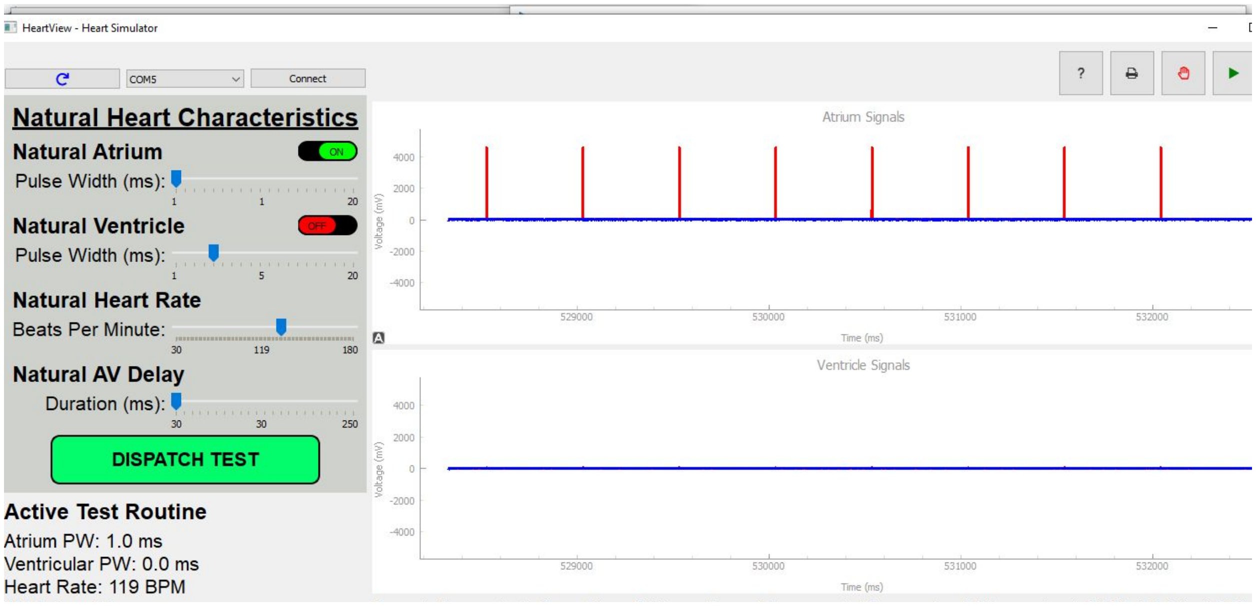
Clearly there is a natural heart pace sent after each pacemaker heart pace, as indicated by the graph. Using the inputs of this test, each natural pace occurs 4 msec after each pacemaker pace. Because the refractory period was set to 250 msec, the natural pace (4 msec) occurs within the refractory period (0-250 msec). During the refractory period the natural pace should be ignored. The graph shows no difference with regards to the reference graph from test 1, so the paces are being ignored.

Verdict: Pass

AAI

Test 6

Goal: The previous test demonstrated that a natural pulse just 4msec after the pacemaker pulse did not cause the pacemaker to inhibit its pulse (due to refractory period). In order to sense that natural pulse we will add hysteresis for 10msec. The pacemaker should now inhibit every pace. Also test is if the paces can be sensed with a lower sensitivity value.


HeartView - Heart Simulator

COM5 Connect

Natural Heart Characteristics

Natural Atrium ON

Pulse Width (ms): 1

Natural Ventricle OFF

Pulse Width (ms): 5

Natural Heart Rate

Beats Per Minute: 119

Natural AV Delay

Duration (ms): 30

DISPATCH TEST

Active Test Routine

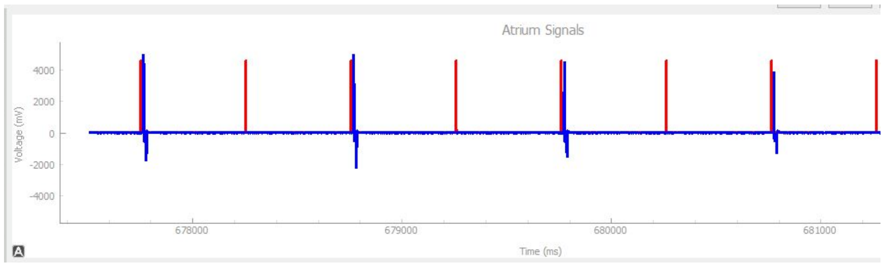
Atrium PW: 1.0 ms
Ventricular PW: 0.0 ms
Heart Rate: 119 BPM

Because we extended the lower rate limit by 15 msec through hysteresis, each of the natural paces now occur between the refractory period and the lower rate limit (250 - 515 msec). Hence all paces are inhibited.

<u>Verdict: Pass</u>

Test 7

Goal: This test will test the timing accuracy of the pacemaker. Rather than a hysteresis escape interval of 15 msec we will try 10 msec. The results should be the same. For context the natural pace occurs about 4.2 msec after the lower rate limit, but by adding an escape interval of 10msec, the lower rate limit should also be extended by 10 msec. So the pace should still be sensed (like test 6), and hence inhibited.



Some paces are inhibited, while some are not (about half and half). Hence the system is not correct.

<u>Verdict: Fail</u>

Conclusion

The pacemaker does function correctly (accuracy is high) and functions in all the pacing modes for various programmable values (robustness) but it is very imprecise (as indicated by test 1,2,7). However it is important testing cannot prove correctness, only robustness and lack of bugs.