Pacemaker Firmware Documentation

MECHTRON 3K04 Assignment 2

Group 4

Andrew De Rango, Ali Hussin, Ethan Otteson,
Marco Tan, Rafael Toameh

Friday, November 29, 2024

Table of Contents

1 Scope

This document provides a comprehensive overview of the Pacemaker Firmware, an embedded system software designed to control the functionality and operation of pacemaker effectively. It outlines the firmware's requirements, functionality, implementation, design decisions, justifications, and testing made throughout the development process, enabling users and developers to have a clear understanding of the firmware's features and software architecture.

2 Terms and Definitions

Pacemaker Firmware - All software that is embedded on to the pacemaker microcontroller.

Off - Pacemaker mode in which no sensing or shocking occurs.

AOO - Pacemaker mode in which the atria are shocked rhythmically, without considering the natural atrial pulse.

VOO - Pacemaker mode in which the ventricles are shocked rhythmically, without considering the natural ventricular pulse.

AAI - Pacemaker mode in which the atria are shocked rhythmically unless inhibited by natural atrial pulses.

VVI - Pacemaker mode in which the ventricles are shocked rhythmically unless inhibited by natural ventricular pulses.

DDD - Pacemaker mode in which both the atria and ventricles are shocked rhythmically unless inhibited by natural atrial or ventricular pulses.

MSR - Maximum Sensor Rate, the heart rate which is the maximum at a specific activity level.

AOOR - Pacemaker mode operating the same as AOO but rhythmic shocks are rate adaptive.

VOOR - Pacemaker mode operating the same as VOO but rhythmic shocks are rate adaptive.

AAIR - Pacemaker mode operating the same as AAI but rhythmic shocks are rate adaptive.

VVIR - Pacemaker mode operating the same as VVI but rhythmic shocks are rate adaptive.

DDDR - Pacemaker mode operating the same as DDD but rhythmic shocks are rate adaptive.

ARP - Atrial refractory period

VRP - Ventricular refractory period

LRL - Lower rate limit

URL - Upper rate limit

HR - heart rate

RAPL - Rate adaptive pacing limit

BPM - Beats per minute

DCM - Device Controller-Monitor, see separate documentation for further details.

State - Block in Simulink which can set variables to value and acts as a state in a finite state machine.

Chart - Block in Simulink which contains states, acts as a finite state machine.

Subchart - Chart inside of another chart which contains states

HeartView - Software in which natural atrium and ventricle pulses can be created and natural and artificial pacemaker pulses can be visualized and monitored. Separate program provided to the project team.

Charging states - Variables that tell the firmware which pins should be active.

Electrogram - Term used for electrical activity of the heart recorded by the pacemaker and graphed. Also referred to as egram.

3 Requirements

The following outlines all the recognized Pacemaker Firmware requirements. These requirements come from either the PACEMAKER requirements document provided as part of the project instruction package or from the project team directly. Not all requirements presented in the PACEMAKER document are included as many were considered out of scope for the implementation required for this project. This section includes all of the requirements given in natural language with many also provided in tabular expression for clarity, and a note regarding the expected future requirement changes.

3.1 Natural Language Requirements

The Pacemaker Firmware should abide by the following natural language requirements. These requirements are provided for ease of understanding and complement the formal specifications provided below. If the natural language requirements and formal specifications do not agree, the formal specifications are given precedence.

1. Pacemaker Firmware should implement the 11 programmable modes Off, AOO, VOO, AAI, VVI, DDD, AOOR, VOOR, AAIR, VVIR, & DDDR.
2. The ability to toggle between modes should be available.
3. All modes except for Off should regulate the patient's heart rate with electrical pulses.
4. The following independent programmable parameters should be available:
   - 4.1. Pulse Amplitude should operate as follows:
     - 4.1.1. Two independent parameters, one for the atria and one for the ventricles.
     - 4.1.2. Pulse amplitude should be the voltage delivered to the heart when paced within a 12% tolerance.
     - 4.1.3. Atrial and ventricular pulse amplitudes should be between 0V and 5V.
   - 4.2. Pulse width should operate as follows:
     - 4.2.1. Two independent parameters, one for the atria and one for the ventricles.
     - 4.2.2. Pulse width should be the amount of time a pulse is delivered to the heart during a given pace within a 1 ms tolerance.
     - 4.2.3. Atrial and ventricular pulse width should be between 1 ms to 30 ms
   - 4.3. Refractory periods should operate as follows:
     - 4.3.1. Two independent parameters, one for the atria and one for the ventricles.
     - 4.3.2. No pulses should be delivered to the atria during the ARP and to the ventricles during the VRP following a sensed or delivered pulse of that chamber.
     - 4.3.3. Atrial and ventricular refractory periods should be between 150 ms and 500 ms.
   - 4.4. Lower rate limit should operate as follows:
     - 4.4.1. A single programmable parameter for both the atria and ventricles.
     - 4.4.2. If the heart rate is less than the lower rate limit then the heart is considered to be in bradycardia.

- 4.4.3. The lower rate limit should be between 30 to 175 BPM with a tolerance of 8 ms.
- 4.5. Upper rate limit should operate as follows:
  - 4.5.1. A single programmable parameter for both the atria and ventricles.
  - 4.5.2. Describes the maximum allowable pacing rate for rate adaptive modes, as such is only required for rate adaptive modes.
  - 4.5.3. The upper rate limit should be between 50 to 175 BPM with a tolerance of 8 ms.
- 4.6. Sensitivity should operate as follows:
  - 4.6.1. Two independent parameters, one for the atria and one for the ventricles.
  - 4.6.2. Sensitivity should be the reference voltage used to determine if a given sensed heart voltage is considered enough to count as sensed within a 2% tolerance.
  - 4.6.3. Atrial and ventricular sensitivities should be between 0V and 5V.
- 4.7. Atrioventricular delay should operate as follows:
  - 4.7.1. A single programmable parameter required for the DDD and DDDR pacing modes.
  - 4.7.2. Describes the amount of time between a sensed or delivered atrial pulse and delivered ventricular pulse, assuming a ventricular pulse is not inhibited.
  - 4.7.3. The AV delay should be between 30 ms and 300 ms with a tolerance of 8 ms.
- 4.8. Rate factor (or response factor) should operate as follows:
  - 4.8.1. A single programmable parameter required for the rate adaptive pacing modes.
  - 4.8.2. Describes how steep the curve of pacing rate versus activity is. A high rate factor will require lower activity to reach the maximum rate adaptive pacing rate compared to a lower rate factor.
  - 4.8.3. The rate factor should be an integer between 1 and 16.
- 4.9. Activity threshold should operate as follows:
  - 4.9.1. A single programmable parameter required for the rate adaptive pacing modes.
  - 4.9.2. Describes the threshold at which additional activity begins to increase the adaptive pacing rate. A lower threshold will require less activity to be considered active compared to a higher threshold.
  - 4.9.3. The activity threshold should have one of the following values: very low, low, medium low, medium, medium high, high, very high.
- 4.10. Reaction time should operate as follows:
  - 4.10.1. A single programmable parameter required for the rate adaptive pacing modes.
  - 4.10.2. Describes the amount of time required for the heart rate to reach the limit set by the rate adaptive pacing.
  - 4.10.3. The reaction time should be between 1 to 50 seconds.

- 4.10.4. Note that the allowable range has been modified from the provided PACEMAKER document to allow for ease of demonstration and testing.

4.11. Recovery time should operate as follows:

- 4.11.1. A single programmable parameter is required for the rate-adaptive pacing modes.
- 4.11.2. Describes the amount of time required for the heart rate to reach the lower rate limit from the current heart rate.
- 4.11.3. The recovery time should be between 1 to 240 seconds.
- 4.11.4. Note that the allowable range has been modified from the provided PACEMAKER document to allow for ease of demonstration and testing.

5. Off mode should operate as follows:
   - 5.1. Not sense or regulate any chambers of the heart.
6. AOO mode should operate as follows:
   - 6.1. Pace the atria periodically regardless of the patient's heart activity.
   - 6.2. Periodic pacing should be based on the lower rate limit provided.
   - 6.3. Does not affect the ventricles.
7. VOO mode should operate as follows:
   - 7.1. Pace the ventricles periodically regardless of the patient's heart activity.
   - 7.2. Periodic pacing should be based on the lower rate limit provided.
   - 7.3. Does not affect the atria.
8. AAI mode should operate as follows:
   - 8.1. Monitor the patient's heart rate for bradycardia and provide pacing if bradycardia is detected.
   - 8.2. If the sensed atrial heart rate is less than the lower rate limit then the mode should detect bradycardia and deliver a pacing pulse to the atria.
   - 8.3. Does not affect the ventricles.
9. VVI mode should operate as follows:
   - 9.1. Monitor the patient's heart rate for bradycardia and provide pacing if bradycardia is detected.
   - 9.2. If the sensed ventricular heart rate is less than the lower rate limit then the mode should detect bradycardia and deliver a pacing pulse to the ventricles.
   - 9.3. Does not affect the atria.
10. DDD mode should operate as follows:
    - 10.1. Provides dual chamber pacing and monitoring with the ability to inhibit both atrial and ventricular pulses or trigger ventricular pulses based on a sensed atrial pulse.
    - 10.2. If the sensed atrial heart rate is less than the lower rate limit then the mode should detect bradycardia or atrial malfunction and deliver a pacing pulse to the atria.
    - 10.3. If the sensed atrial heart rate is greater than the lower rate limit then the mode should not deliver any sort of atrial pacing.
    - 10.4. Following a delivered or sensed atrial pulse, if a ventricular pulse is not detected during the expected AV delay the pacemaker should trigger and deliver a pacing pulse to the ventricles.

   - 10.5. Following a delivered or sensed atrial pulse, if a ventricular pulse is detected during the expected AV delay the pacemaker should not deliver any sort of ventricular pacing.
   - 10.6. Pressing the button on the pacemaker board should always inhibit ventricular pacing in this mode and the mode's function should return to normal once the button is released.
11. Rate adaptive pacing should operate as follows:
   - 11.1. MSR should be set and modulated based on activity, and rate factor.
   - 11.2. An activity threshold should be set and when the activity of the user is greater than the threshold, MSR should be used, otherwise the MSR should be equal to the lower rate limit.
   - 11.3. The modulated rate should always be obtained gradually, and jumps should not occur.
   - 11.4. The parameters recovery and reaction time should dictate the time to reach the modulated rate, recovery for decreasing and reaction for increasing.
   - 11.5. The rate factor parameter should dictate the required activity for the rate adaptive pacing limit to reach the upper rate limit.
12. AOOR mode should operate as follows:
   - 12.1. Requirements are identical to 3.1.6 AOO mode substituting the lower rate limit for the rate adaptive pacing limit.
13. VOOR mode should operate as follows:
   - 13.1. Requirements are identical to 3.1.7 VOO mode substituting the lower rate limit for the rate adaptive pacing limit.
14. AAIR mode should operate as follows:
   - 14.1. Requirements are identical to 3.1.8 AAI mode substituting the lower rate limit for the rate adaptive pacing limit.
15. VVIR mode should operate as follows:
   - 15.1. Requirements are identical to 3.1.9 VVI mode substituting the lower rate limit for the rate adaptive pacing limit.
16. DDDR mode should operate as follows:
   - 16.1. DRequirements are identical to 3.1.10 DDD mode substituting the lower rate limit for the rate adaptive pacing limit.
17. Serial communication should be included and follow the following requirements:
   - 17.1. Should follow all requirements and design elements described in the "Serial Protocol Documentation" for this project.
   - 17.2. Should allow for the transmission and receival of information to and from the DCM when connected via a UART compatible cable.
   - 17.3. Should be able to send the pacemaker's serial number, current parameters, and electrogram data to the DCM.
   - 17.4. Should be able to receive commands to modify parameters and toggle the transmission of electrogram data from the DCM.

3.2 Formal Specifications

These formal specifications provided in tabular expression are intended to provide a clear, unambiguous, and precise description for many of the natural language requirements. The natural language requirement corresponding to a specific specification is noted in the title of the table. Not all requirements have a corresponding specification. If the natural language requirements and formal specifications do not agree, the formal specifications are given precedence.

Table 3.2.a: Formal specification of 3.1.1 describing the required modes.
**Note: A is for atrium, V is for ventricle, B is for both chambers, I is for inhibitor, and I+T is for inhibitor and trigger.*

	Pacemaker Firmware Modes										
Mode Behavior	**Off**	**AOO**	**VOO**	**AAI**	**VVI**	**DDD**	**AOOR**	**VOOR**	**AAIR**	**VVIR**	**DDDR**
Pace Chamber	N/A	A	V	A	V	B	A	V	A	V	B
Sense Chamber	N/A	N/A	N/A	A	V	B	N/A	N/A	A	V	B
Response to Sense	N/A	N/A	N/A	I	I	I+T	N/A	N/A	I	I	I+T
Rate Adaptive Pacing	N/A	N/A	N/A	N/A	N/A	N/A	Yes	Yes	Yes	Yes	Yes

Table 3.2.b: Formal specification of 3.1.4 describing the behaviour of the programmable parameter.

	Result
Condition	**Given Programmable Parameter**
Less than lower range limit	Round parameter to lower range limit
Within range	Parameter remains unchanged
Greater than upper range limit	Round parameter to upper range limit

Table 3.2.c: Formal specification of 3.1.5 describing the behaviour of the Off mode.

	Result	
Condition	**Pace Atria**	**Pace Ventricles**
Heart Rate < Lower Rate Limit	No	No
Heart Rate = Lower Rate Limit	No	No
Heart Rate > Lower Rate Limit	No	No

Table 3.2.d: Formal specification of 3.1.6 describing AOO mode behavior.

	Result	
Condition	**Pace Atria**	**Pace Ventricles**
Heart Rate < Lower Rate Limit	Yes, LRL after most recent delivered pulse for atrial pulse width time and with atrial pulse amplitude	No
Heart Rate = Lower Rate Limit	Yes, LRL after most recent delivered pulse for atrial pulse width time and with atrial pulse amplitude	No
Heart Rate > Lower Rate Limit	Yes, LRL after most recent delivered pulse for atrial pulse width time and with atrial pulse amplitude	No

Table 3.2.e: Formal specification of 3.1.7 describing VOO mode behavior.

	Result	
Condition	**Pace Atria**	**Pace Ventricles**
Heart Rate < Lower Rate Limit	No	Yes, LRL after most recent delivered pulse for ventricular pulse width time and with ventricular pulse amplitude
Heart Rate = Lower Rate Limit	No	Yes, LRL after most recent delivered pulse for ventricular pulse width time and with ventricular pulse amplitude
Heart Rate > Lower Rate Limit	No	Yes, LRL after most recent delivered pulse for ventricular pulse width time and with ventricular pulse amplitude

Table 3.2.f: Formal specification of 3.1.8 describing AAI mode behaviour.

	Result	
Condition	**Pace Atria**	**Pace Ventricles**
Heart Rate < Lower Rate Limit	Yes, LRL after most recent sensed or delivered pulse for atrial pulse width time and with atrial pulse amplitude	No
Heart Rate = Lower Rate Limit	No	No
Heart Rate > Lower Rate Limit	No	No

Table 3.2.g: Formal specification of 3.1.9 describing VVI mode behavior.

	Result	
Condition	**Pace Atria**	**Pace Ventricles**
Heart Rate < Lower Rate Limit	No	Yes, LRL after most recent sensed or delivered pulse for ventricular pulse width time and with ventricular pulse amplitude

Heart Rate = Lower Rate Limit	No	No
Heart Rate > Lower Rate Limit	No	No

Table 3.2.h: Formal specification of 3.1.10 describing DDD mode behavior.

**Note: The actual AV delay is considered to be the time between a sensed or delivered atrial pulse and the next ventricular pulse. This definition means that if the atria are not beating then the AV delay set in HeartView may not be equivalent to the actual AV delay, however, if the atria are beating then the AV delay set in HeartView is equivalent to the actual AV delay.*

	Result	
Condition	**Pace Atria**	**Pace Ventricles**
Atria not beating AND Ventricles not beating	Yes, LRL after most recent delivered pulse for atrial pulse width time and with atrial pulse amplitude	Yes, AV delay after the delivered atrial pulse for ventricular pulse width time and with ventricular pulse amplitude
Atria not beating AND Actual AV Delay > Programmed AV Delay	Yes, LRL after most recent delivered pulse for atrial pulse width time and with atrial pulse amplitude	Yes, AV delay after the delivered atrial pulse for ventricular pulse width time and with ventricular pulse amplitude
Atria not beating AND Actual AV Delay ≤ Programmed AV Delay	Yes, LRL after most recent delivered pulse for atrial pulse width time and with atrial pulse amplitude	No
Heart Rate < Lower Rate Limit AND Ventricles not beating	Yes, LRL after most recent sensed or delivered pulse for atrial pulse width time and with atrial pulse amplitude	Yes, AV delay after the sensed or delivered atrial pulse for ventricular pulse width time and with ventricular pulse amplitude
Heart Rate < Lower Rate Limit AND Actual AV Delay > Programmed AV Delay	Yes, LRL after most recent sensed or delivered pulse for atrial pulse width time and with atrial pulse amplitude	Yes, AV delay after the sensed or delivered atrial pulse for ventricular pulse width time and with ventricular pulse amplitude
Heart Rate < Lower Rate Limit AND Actual AV Delay ≤ Programmed AV Delay	Yes, LRL after most recent sensed or delivered pulse for atrial pulse width time and with atrial pulse amplitude	Yes only if AV delay after the delivered atrial pulse, for ventricular pulse width time and with ventricular pulse amplitude No, following a sensed atrial pulse
Heart Rate ≥ Lower Rate Limit AND Ventricles not beating	No	Yes, AV delay after the sensed atrial pulse for ventricular pulse width time and with ventricular pulse amplitude
Heart Rate ≥ Lower Rate Limit AND Actual AV Delay > Programmed AV Delay	No	Yes, AV delay after the sensed atrial pulse for ventricular pulse width time and with ventricular pulse amplitude
Heart Rate ≥ Lower Rate Limit AND Actual AV Delay ≤ Programmed AV Delay	No	No

Table 3.2.i: Formal specification of 3.1.11.1 describing MSR creation behaviour

Condition	***Result***

Activity < Activity Threshold	MSR = LRL
Activity > Activity Threshold	MSR = Programmer decided equation

Table 3.2.j: Formal specification of 3.1.11 describing rate adaptive pacing behaviour

Condition	***Result***
Activity < Activity Threshold AND HR = LRL	Keep HR at LRL
Activity < Activity Threshold AND LRL < HR	Gradually lower HR to LRL after recovery time
Activity > Activity Threshold AND HR < MSR	Gradually Raise HR to MSR after the reaction time
Activity > Activity Threshold AND HR > MSR	Gradually Lower HR to MSR after recovery time

Table 3.2.k: Formal specification of 3.1.12 describing AOOR mode behavior.

	Result	
Condition	**Pace Atria**	**Pace Ventricles**
Heart Rate < Rate Adaptive Pacing Limit	Yes, RAPL after most recent delivered pulse for atrial pulse width time and with atrial pulse amplitude	No
Heart Rate = Rate Adaptive Pacing Limit	Yes, RAPL after most recent delivered pulse for atrial pulse width time and with atrial pulse amplitude	No
Heart Rate > Rate Adaptive Pacing Limit	Yes, RAPL after most recent delivered pulse for atrial pulse width time and with atrial pulse amplitude	No

Table 3.2.l: Formal specification of 3.1.13 describing VOOR mode behavior.

	Result	
Condition	**Pace Atria**	**Pace Ventricles**
Heart Rate < Rate Adaptive Pacing Limit	No	Yes, RAPL after most recent delivered pulse for ventricular pulse width time and with ventricular pulse amplitude
Heart Rate = Rate Adaptive Pacing Limit	No	Yes, RAPL after most recent delivered pulse for ventricular pulse width time and with ventricular pulse amplitude
Heart Rate > Rate Adaptive Pacing Limit	No	Yes, RAPL after most recent delivered pulse for ventricular pulse width time and with ventricular pulse amplitude

Table 3.2.m: Formal specification of 3.1.14 describing AAIR mode behaviour.

	Result	
Condition	**Pace Atria**	**Pace Ventricles**
Heart Rate < Rate Adaptive Pacing Limit	Yes, RAPL after most recent sensed or delivered pulse for atrial pulse width time and with atrial pulse amplitude	No
Heart Rate = Rate Adaptive Pacing Limit	No	No
Heart Rate > Rate Adaptive Pacing Limit	No	No

Table 3.2.n: Formal specification of 3.1.15 describing VVIR mode behavior.

	Result	
Condition	**Pace Atria**	**Pace Ventricles**
Heart Rate < Rate Adaptive Pacing Limit	No	Yes, RAPL after most recent sensed or delivered pulse for ventricular pulse width time and with ventricular pulse amplitude
Heart Rate = Rate Adaptive Pacing Limit	No	No
Heart Rate > Rate Adaptive Pacing Limit	No	No

Table 3.2.o: Formal specification of 3.1.16 describing DDDR mode behavior.

**Note: The actual AV delay is considered to be the time between a sensed or delivered atrial pulse and the next ventricular pulse. This definition means that if the atria are not beating then the AV delay set in HeartView may not be equivalent to the actual AV delay, however, if the atria are beating then the AV delay set in HeartView is equivalent to the actual AV delay.*

	Result	
Condition	**Pace Atria**	**Pace Ventricles**
Atria not beating AND Ventricles not beating	Yes, RAPL after most recent delivered pulse for atrial pulse width time and with atrial pulse amplitude	Yes, AV delay after the delivered atrial pulse for ventricular pulse width time and with ventricular pulse amplitude
Atria not beating AND Actual AV Delay > Programmed AV Delay	Yes, RAPL after most recent delivered pulse for atrial pulse width time and with atrial pulse amplitude	Yes, AV delay after the delivered atrial pulse for ventricular pulse width time and with ventricular pulse amplitude
Atria not beating AND Actual AV Delay ≤ Programmed AV Delay	Yes, RAPL after most recent delivered pulse for atrial pulse width time and with atrial pulse amplitude	No
Heart Rate < Rate Adaptive Pacing Limit AND Ventricles not beating	Yes, RAPL after most recent sensed or delivered pulse for atrial pulse width time and with atrial pulse amplitude	Yes, AV delay after the sensed or delivered atrial pulse for ventricular pulse width time and with ventricular pulse amplitude
Heart Rate < Rate Adaptive Pacing Limit AND Actual AV Delay > Programmed AV Delay	Yes, RAPL after most recent sensed or delivered pulse for atrial pulse width time and with atrial pulse amplitude	Yes, AV delay after the sensed or delivered atrial pulse for ventricular pulse width time and with ventricular pulse amplitude
Heart Rate < Rate Adaptive Pacing Limit	Yes, RAPL after most recent sensed	Yes only if AV delay after the

AND Actual AV Delay ≤ Programmed AV Delay	or delivered pulse for atrial pulse width time and with atrial pulse amplitude	delivered atrial pulse, for ventricular pulse width time and with ventricular pulse amplitude No, following a sensed atrial pulse
Heart Rate ≥ Rate Adaptive Pacing Limit AND Ventricles not beating	No	Yes, AV delay after the sensed atrial pulse for ventricular pulse width time and with ventricular pulse amplitude
Heart Rate ≥ Rate Adaptive Pacing Limit AND Actual AV Delay > Programmed AV Delay	No	Yes, AV delay after the sensed atrial pulse for ventricular pulse width time and with ventricular pulse amplitude
Heart Rate ≥ Rate Adaptive Pacing Limit AND Actual AV Delay ≤ Programmed AV Delay	No	No

3.3 Completed and Expected Changes

The following section describes the changes to the requirements for the Pacemaker Firmware made between Assignment 1 and 2 and any expected future changes. While these future changes will not be implemented as this project will be complete upon this document's submission, we considered these future changes as either those we would want to make if we had more time and the changes which would be anticipated if this system was to be transformed into an actual medical pacemaker device.

3.3.1 Past Changes

Many of the requirement changes made between the two assignments were initially anticipated, including the following. Additional requirements were added to outline the behaviour of the new pacing modes including DDD, AOOR, VOOR, AAIR, VVIR, & DDDR. These requirement changes were necessary to differentiate the new modes from the previous ones. These requirements were used to direct the design process for developing these new modes. Requirements had to be added describing the new programmable parameters used for these new modes. These new parameters fall under two categories, the parameters required to facilitate rate adaptive pacing and the parameters required for dual chamber pacing. The new parameters which need requirements for the rate adaptive pacing were the upper rate limit, rate factor, activity threshold, reaction time, and recovery time. These parameters were previously not required as none of the previous modes implemented rate adaptive pacing. However, with the need to add rate adaptive pacing, it was important to outline the behaviour for these parameters to ensure the pacemaker only operates within safe and reasonable parameter conditions. This also ensured that each of the required parameters for rate adaptive pacing would be implemented and none of them would be missed or forgotten about. The only new parameter required to enable dual chamber pacing was the AV delay. For similar reasons, it was important to describe the expected behaviour of this parameter to ensure it is bound within safe and reasonable bounds and is used for its intended purpose. The final requirement change that we anticipated was adding requirements for the necessary serial communication between the Pacemaker Firmware and the DCM. These requirements are fairly simple, however, the most important of which was to require all the assignment specific behavior and require that the Pacemaker Firmware follows the serial protocol outlined in the separate Serial Protocol Documentation. This will ensure all necessary requirements are captured by the design process. There was one requirement change between assignments that we made but was not included in our initially anticipated changes. This was adding requirements for how the rate adaptive pacing rate should be determined. These requirements are important as they link the various parameters together to govern how this feature should work. We likely missed anticipating this because we did not have a complete understanding of what rate adaptive passing would entail prior to submitting Assignment 1. All of these requirement changes are contained within the new sections 3.1.4.5, 3.1.4.7-3.1.4.11, 3.1.10-3.1.17, and Tables 3.2.h-3.2.o. This has described all of the requirement changes and justification for these changes underwent between Assignment 1 and 2.

3.3.2 Future Changes

Although this project has reached its effective endpoint, it is still valuable to consider what requirement changes would be anticipated if this project were to continue. Firstly, we think it would be interesting to add more modes to the pacemaker, specifically those that respond to tachycardia as all of the current modes are responsible for handling bradycardia. Adding these new modes would come with new requirements describing the behaviour of these modes and with new parameter requirements describing the bounds and uses for these new required parameters. Considering what requirement changes may be necessary if this project were continued with the ultimate goal of creating a real pacemaker to be implanted in patients. Many additional requirements would need to be included specifically around safety so the device could ethically be considered safe. Following are some of the examples of these heightened requirements which would be required. Considering the highly time-sensitive nature of pacing the heart both ensuring that the pacing rate is consistent with its programmed behaviour and that the sensing activity is working exactly as expected most rigorous requirements would be needed to ensure the boundary cases which are currently raising concerns would be untolerated in an actual application. There would also be a need for requirements outlining exactly which tests and verification would need to be conducted and the tolerated results. It would be important to come up with a set of tests before beginning the design process to avoid issues such as confirmation bias. Many of these stricter and added requirements would be taken from the associated regulations, specifications, and standards outlined by the medical pacemaker industry. Finally, the requirements for how the DCM and Pacemaker Firmware communicate would need to be overhauled. This is because, in an actual implementation, the pacemaker would be embedded in the patient’s thoracic cavity without the ability to connect via a wire. This requirement change would likely see a migration to a wireless communication protocol over serial communication. These are the anticipated changes required for the Pacemaker Firmware requirements assuming the project were to continue, also including justification for each proposed change.

4 Design

The following outlines all design decisions that were made while creating the Pacemaker Firmware, including the functionality of every module, and the reasoning for the design decisions.

4.1 Overall Design

The Pacemaker Firmware was created using MATLAB Simulink and consists of four main sections, the input subsystem, the serial subsystem, the pacemaker controller chart, and the output subsystem. The input subsystem is where all the necessary inputs are taken in, checked to be within the correct range, and converted into the expected form. In the Serial subsystem, the values are taken from the DCM and passed into the input subsystem, and values are sent from the pacemaker to the DCM. In the pacemaker control chart, the behaviour is decided based on the given pacemaker mode. The output subsystem takes values from both the input subsystem and the pacemaker control chart, specific pin behaviour is then decided based on these input values. Finally, the serial subsystem handles all serial communication with the DCM.

The design of the Pacemaker Firmware ensures hardware hiding through the use of the 4 variable model by having direct access to board pins in only 2 locations, the input and output subsystem. This allows the charts to perform computation without direct access to the pins. This decomposition means that if an element related to the inputs is changed, such as connecting the Pacemaker Firmware to the DCM, only the input subsystem will need to be modified with the other sections remaining unchanged. Alternatively, if a different circuit board was used to represent the pacemaker with different sensing and pacing circuitry then the main logic of the pacemaker control chart could be left unchanged with the only modifications required being in the input and output subsystems.

The design does have some coupling in the output system however this is to minimize the duplication of code. The modules that do have problems of coupling are designed to work with the other modules as if the module they are coupled with were part of them. Because of the finite state machine design used in Simulink, only one of the coupled modules will ever be active at a time reducing the possible trouble that could be caused by the coupling. The finite state machine design also makes it as if the module using the code is working like a stand alone module, reducing the effects of coupling even more.

Simulink has increased safety compared to other programming languages due to the usage of finite state machines. A novel unexpected state could pose serious safety risks to the patient as the pacemaker could act unpredictably or incorrectly. Through the use of finite state machines, only specific states are representable, this allows only these states which are considered to be safe to have an effect on the user's heart.

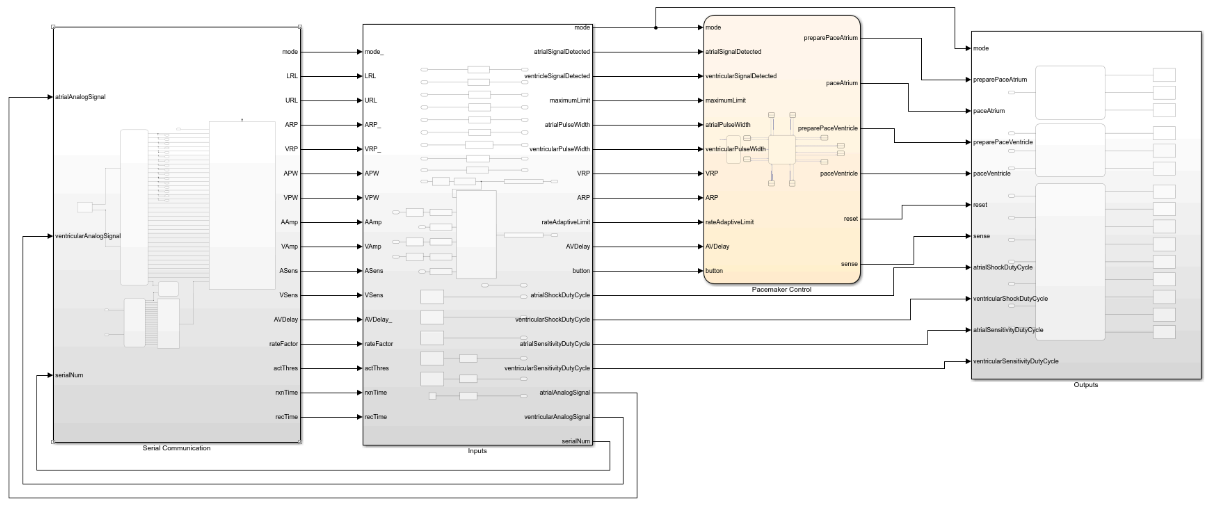


Figure 4.1.a: The entire Simulink model, moving from left to right is the serial communication subsystem, the input subsystem, the pacemaker control chart, and the output subsystem.

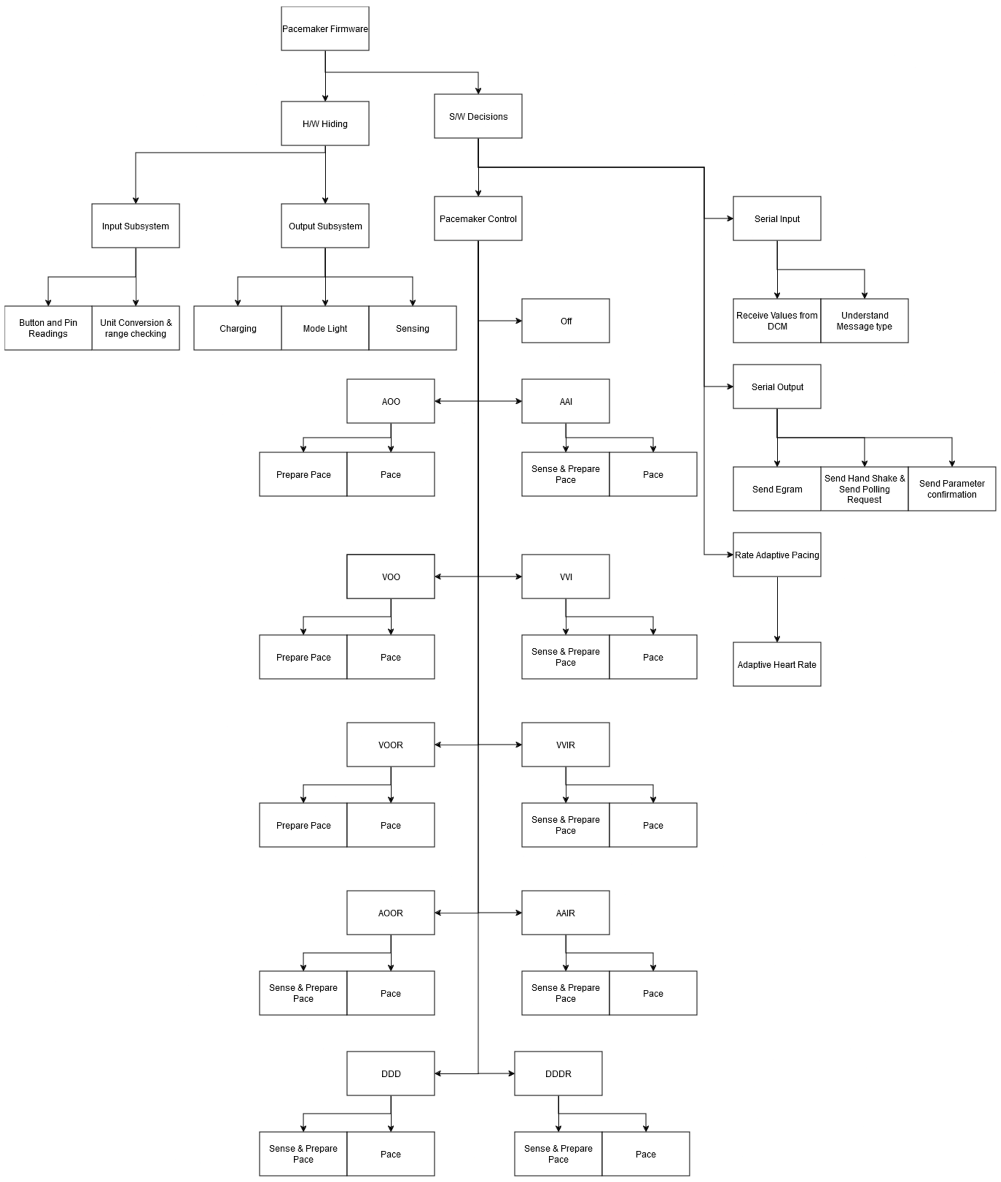


Figure 4.1.b: Module Decomposition of Pacemaker Firmware

At the very top level of the diagram is the Pacemaker Firmware, the app that controls all other decisions made. The second level is hardware hiding and the software decisions. Hardware hiding is implemented through the input and output subsystem. Software decisions are handled by the Pacemaker Control chart, Serial input and Serial Output, and Rate Adaptive Pacing.

In the Input Subsystem, button and pin readings, and unit conversion & range checking, this is what occurs in the input subsystem. Inputs are converted to a more useful form if required and then they are range checked to ensure they are in the required ranges. In the Output Subsystem there is charging, mode light, and sensing. These are the 3 things that occur in this subsystem. They all deal with setting the pins to the correct values, pacing circuitry, lighting up an indicator to signal the active mode, and setting the sensing pins to detect natural heart rhythms.

In Pacemaker Control there are ten main modes and Off. Under the ten main modes is what takes place in them, for AOO, VOO, VOOR, and AOOR there is Pace and Prepare Pace, under AAI, VVI, AAIR, VVIR, DDD, and DDDR are Sense & Prepare Pace and Pace. It is split up like this as All R modes do not care about what the value of the heart rate is and it is decided by a different module.

Under Serial Input there is receiving values from the DCM, specifically parameters, and understanding the messages that are sent to the pacemaker. This is all that is done in the serial input section when looked at in the abstract.

Under Serial Output there is sending egram data, and sending handshakes and poll requests, the parameter values that are sent to the pacemaker are sent back to the DCM for verification.

Under Rate adaptive pacing, there is the calculation of the rate adaptive heart rate, which is the sole responsibility of this module.

Multiple naming conventions were used in the Pacemaker Firmware to differentiate how the variables are used. The different naming convention makes it so that at a glance a programmer can get information about the variable's usage based on the way the name is written. See Table 4.1.a below for the naming conventions used.

Table 4.1.a: Naming convention for variables in Simulink

Variable Type	**Naming Convention**
External inputs and output, and variables that connect to pins	ALL_CAPS_SNAKE_CASE
Subsystems and chart names	Normal Sentences With First Letter Of Each Word Uppercase
Chart input and outputs	camelCase
State titles	CamelCaseWithCapitalFirstLetter

4.2 Input Subsystem

This subsystem is responsible for handling all inputs to the Pacemaker Firmware, that come from the serial subsystem, and the input pins. Any inputs from the DCM are passed from serial into here. Any input interactions with pins are also handled in this subsystem. Rate adaptive limit is also decided in this subsystem. Many values are transformed into more useful forms or are range-checked to ensure they are within the proper bounds before being sent to other parts of the system for use. This subsystem is also where the pacemaker mode is selected. Details for each input variable are discussed below.

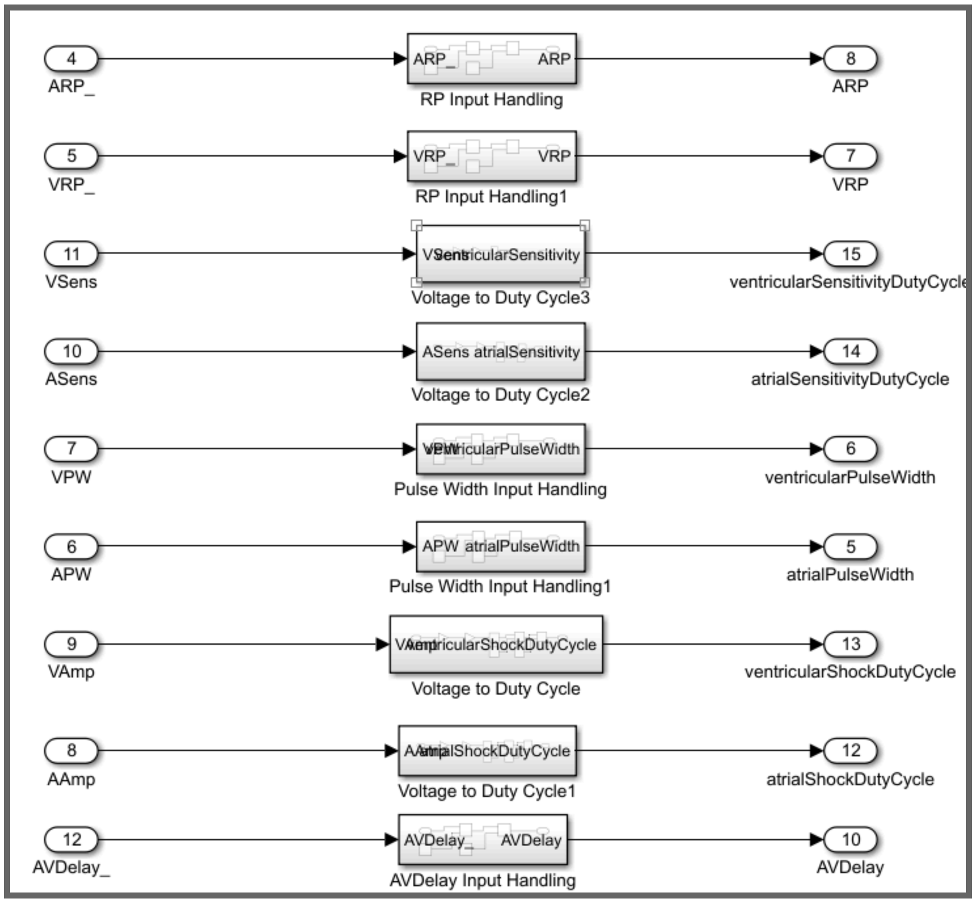


Figure 4.2.a: The top third of the input subsystem, all inputs in this figure are bounds checked and some are changed to a more useful form.

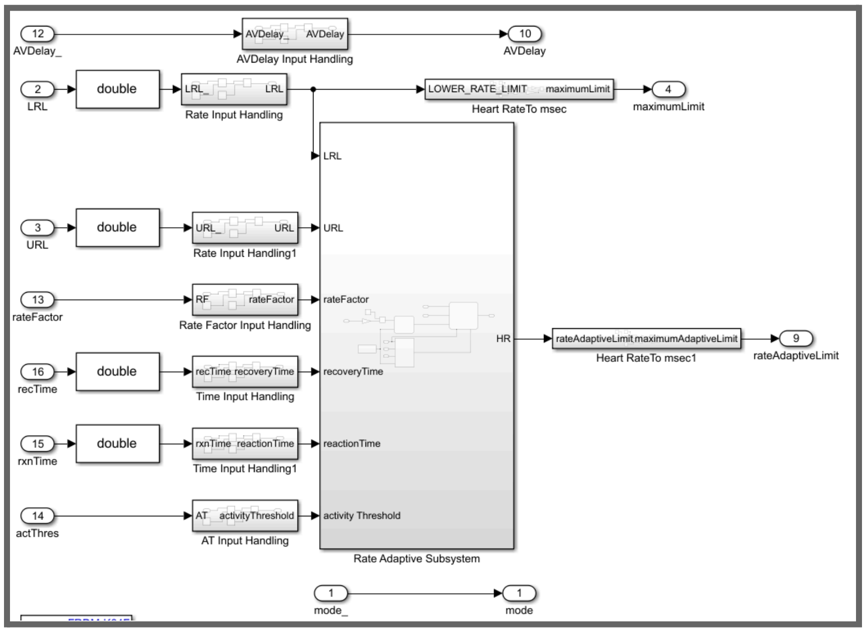


Figure 4.2.b: The middle third of the input subsystem, including the rate adaptive subsystem, mode selection and input handling and changing values to more useful forms.

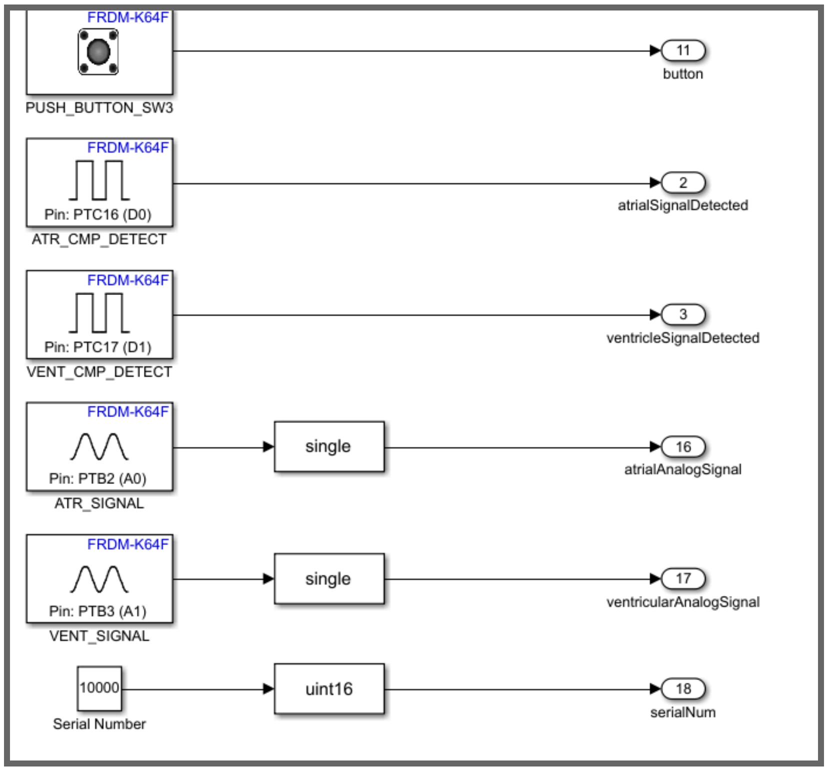


Figure 4.2.c: The bottom third of the input subsystem, which includes only values that have direct pin access and the serial number

4.2.1 Programmable Parameters

Although the programmable parameters are set through interfacing with the DCM, it is still important to collect them all in one place to ensure consistency and make sure the same procedure is followed for each variable. Additionally, adding a level of safety to ensure even if there is an issue with the DCM or serial communication, checking all of the parameters should ensure that the pacemaker does not exit safe operating limits. Each of the parameters is checked to ensure no issues arise from the parameters being outside of reasonable bounds and to ensure compliance with the requirements. In addition to being range checked, many of the parameters are either converted into a different form or have a correction factor applied. Details are covered in the following subsections.

4.2.1.1 Refractory Periods

The atrial refractory period is simply range checked to ensure the given value complies with the requirements. If it is out of range, the value will be rounded to the nearest value within the acceptable range.

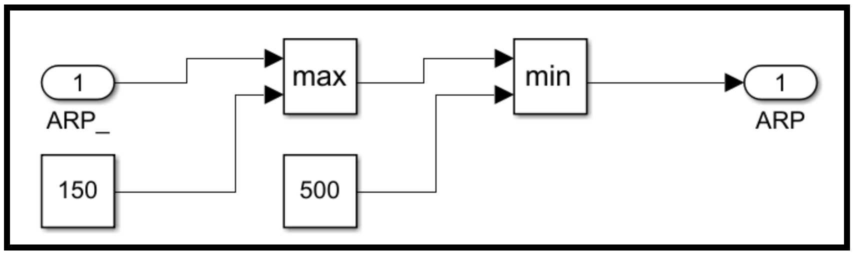


Figure 4.2.1.1.a: Subsystem which range-checks ARP.

Both the refractory period range checking subsystems operate in the same way with the same range. The given refractory period, XRP, is passed in from the serial communication subsystem and range checked between 150ms and 500ms. The variable from the subsystem, ARP or VRP, is used in the pacemaker control chart.

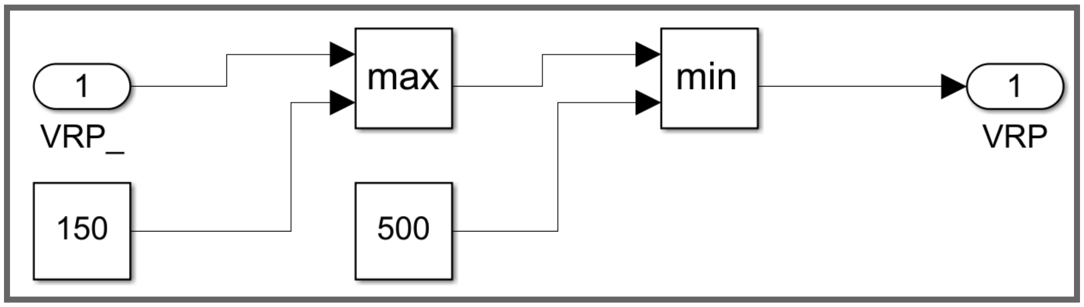


Figure 4.2.1.1.b: Subsystem which range-checks VRP.

4.2.1.2 Sensitivities

Although sensitivity was listed as a required parameter for this assignment, it is important to note that it is not recommended to use any value except 4V to avoid unexpected behavior. Through informal testing, it was determined that setting the sensitivity too low would cause the sensing pins constantly reading noise and outputting a value of true. Alternatively, when the sensitivity was too high it would have trouble detecting the natural heart pulses, particularly at higher heart rates. Either way, this would lead to unexpected behaviour within the pacemaker modes which utilize the sensing circuitry. Using a value of 4V for the sensitivity seemed to mitigate any issues.

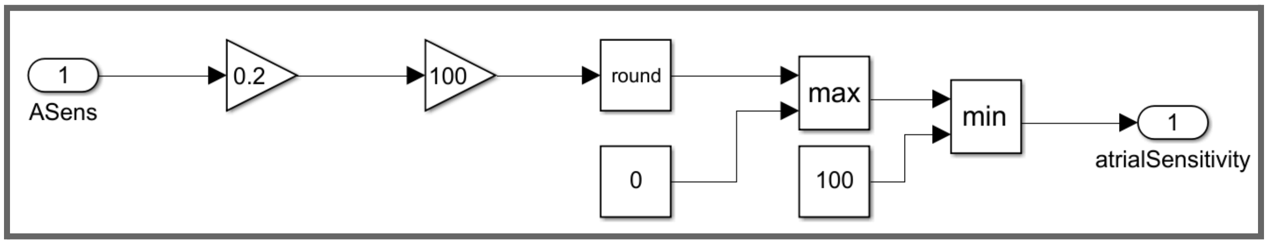


Figure 4.2.1.2.a: Subsystem which range checks and converts ASens.

The sensitivity range checking subsystem operates the same way for both variables. The value is first converted from a voltage into a duty cycle and then the bounds are checked to ensure it is a valid duty cycle. The sensitivities are provided by the serial communication subsystem as a voltage between 0 and 5 and then range checked and converted to a number between 0 and 100. The variable from the subsystem, atrialSensitivity or ventricularSensitivity, is used in the output subsystem to control the sensitivity of the sensing circuitry.

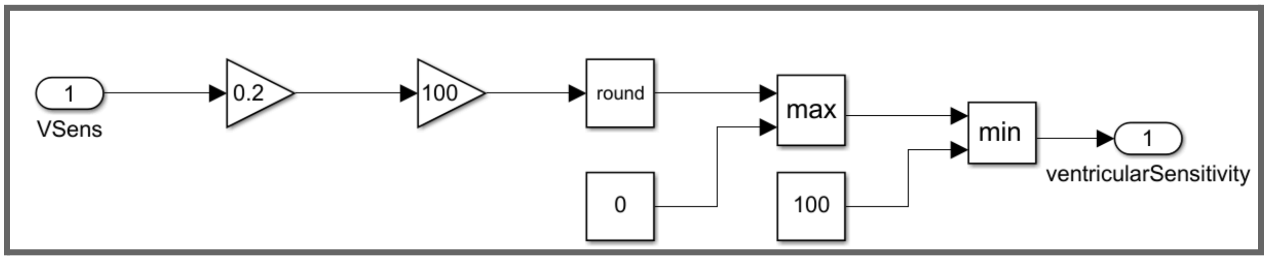


Figure 4.2.1.2.b: Subsystem which range checks and converts VSens.

4.2.1.3 Pulse Widths

The pulse width is the amount of time in milliseconds in which the capacitor delivering the pace to the heart will be given to discharge. This subsystem simply checks the bounds to ensure compliance with the requirements.

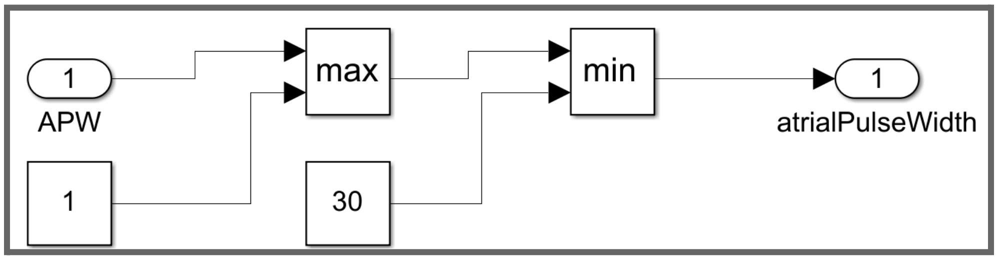


Figure 4.2.1.3.a: Subsystem which range checks APW.

Both pulse width variables are limited to the same range through identical subsystems. XPW is passed in from the serial communication subsystem. The value of XPW is range checked between 1ms and 30ms. The variable from the subsystem, atrialPulseWidth or ventricularPulseWidth, is used in the pacemaker control charge to determine how much time the pulse is delivered for in each mode.

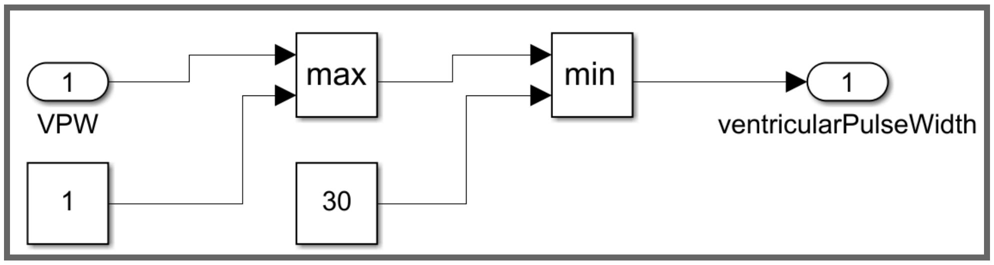


Figure 4.2.1.3.b: Subsystem which range checks VPW.

4.2.1.4 Pulse Amplitudes

The range checking for the pulse amplitudes works similarly to the sensitivities. First, ensure it is converted to a duty cycle, and then make sure the value is a valid duty cycle.

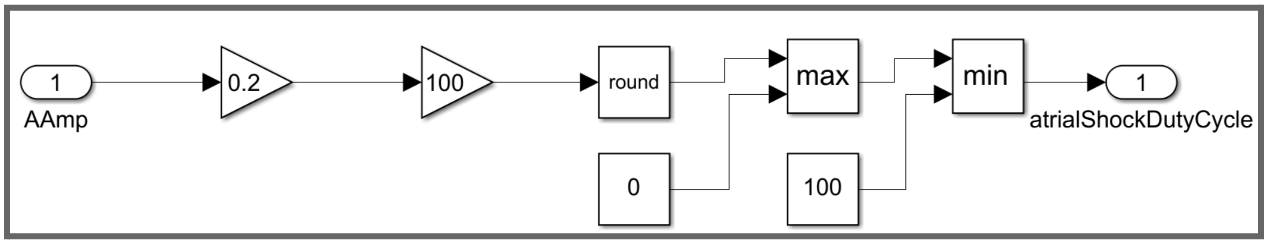


Figure 4.2.1.4.a: Subsystem which range checks and converts AAmp.

XAmp is passed in from the serial communication subsystem. In the associated subsystem, the value of XAmp is converted to duty cycle by multiplying by 20 and is range checked between 0 and 100. The variable from the subsystem, atrialShockDutyCycle or ventricularShockDutyCycle, is used in the output subsystem to control the PWM reference used for pacing the heart.

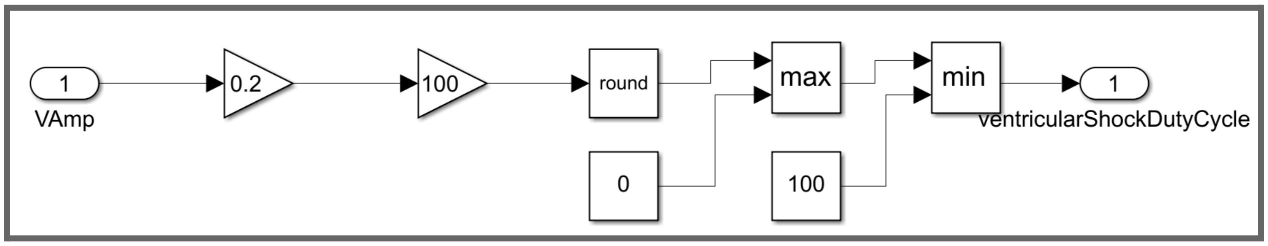


Figure 4.2.1.2.b: Subsystem which range checks and converts VAmp.

4.2.1.5 Atrioventricular Delay

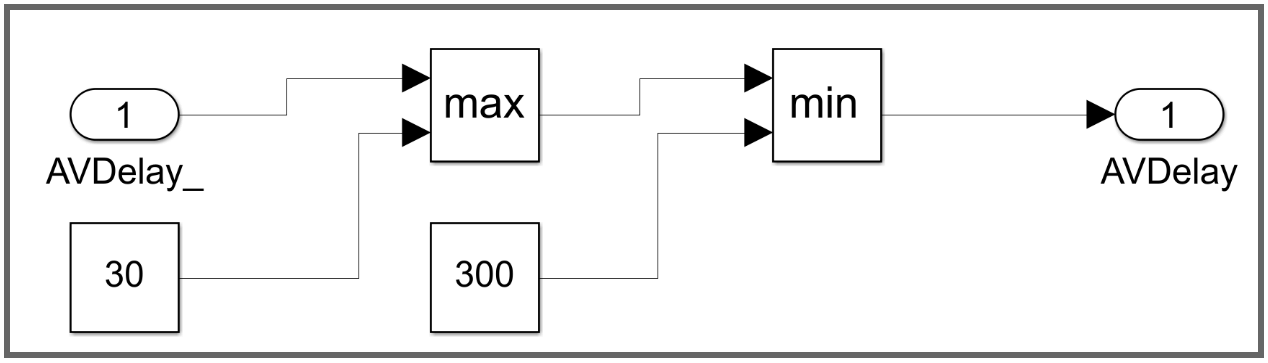


Figure 4.2.1.5.a: Subsystem which range checks AV delay

Atrioventricular delay (AV delay) is the amount of time in milliseconds between the atrium being pulsed and the ventricle being pulsed afterwards, this is used in the DDD and DDDR modes, and is ranged checked between 30 and 300 milliseconds.

4.2.1.6 Lower Rate Limit

The lower rate limit was considered to be the definition of bradycardia. It is the rate at which the AOO and VOO modes (and associated R modes) will pace and the rate at which the AAI, VVI, and DDD modes (and associated R modes) are detecting to determine if they should pace or not. This parameter is given in beats per minute (BPM) and converted into pulse period in milliseconds. Initially, this was all that this subsystem did, however, it was recognized that due to hardware limitations causing the artificial pulsing to be slightly below the given rate and because HeartView is slightly faster than the given rate, the artificial and natural heart rates would operate at slightly different rates when set to the same value causing the plots to shift in phase relative to one another over time. To correct for this the period of the natural and artificial pulses was measured at eight different heart rates. These measurements at each heart rate were an average of the measurement of several periods. The difference between the natural and artificial heart rates was calculated and plotted as a function of heart rate. The trendline was determined for this data and was best matched by a power function, specifically $correction\,factor\ =\ 94.28\cdot(heart\,rate)^{-0.686}$. This correction factor corrects the difference between the natural and artificial heart rate over the whole range to a maximum difference of about 1 ms, which is well within the 8 ms tolerance given by the requirements. This design was pursued as it allows for near-perfect alignment between what the Pacemaker Firmware and HeartView consider to be any given heart rate. This correction factor makes the lower rate limit the most complicated parameter processing for this system.

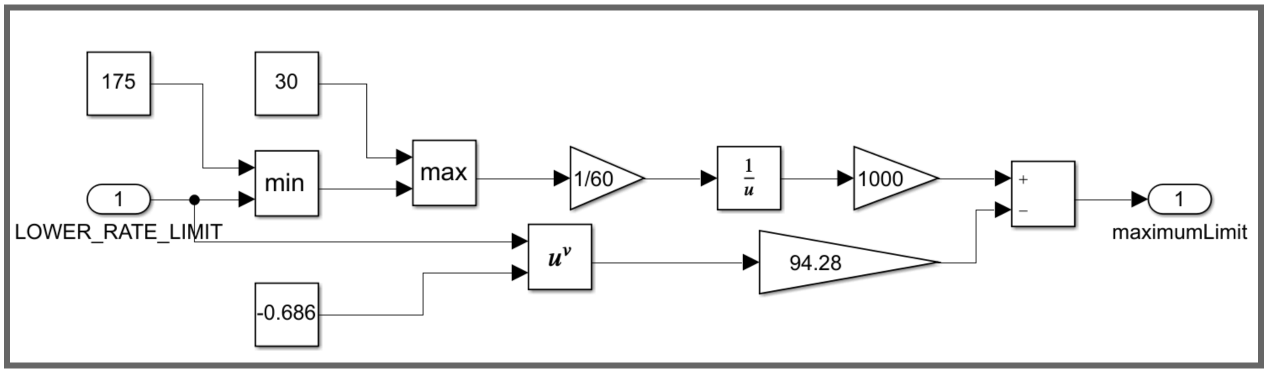


Figure 4.2.1.6.a: Subsystem which range checks and converts LRL

The subsystem works as follows to ensure bound checking and correction are implemented. LRL is passed in from the serial communication subsystem, converted to a double to facilitate the later power operation as part of the correction factor. The LRL is quickly checked to be within the appropriate bounds described by the requirements. From here the LRL is used both in the rate adaptive pacing subsystem and is kept as its own parameter for use in the pacemaker controller chart. For the value used in the pacemaker controller chart it is converted from bpm to milliseconds and range checked. The correction factor is also calculated in parallel and subtracted from the final value. The variable from the subsystem, maximumLimit, is used in the pacemaker control chart as the maximum amount of time the heart should go without a natural or artificial pulse.

4.2.1.7 Upper Rate Limit

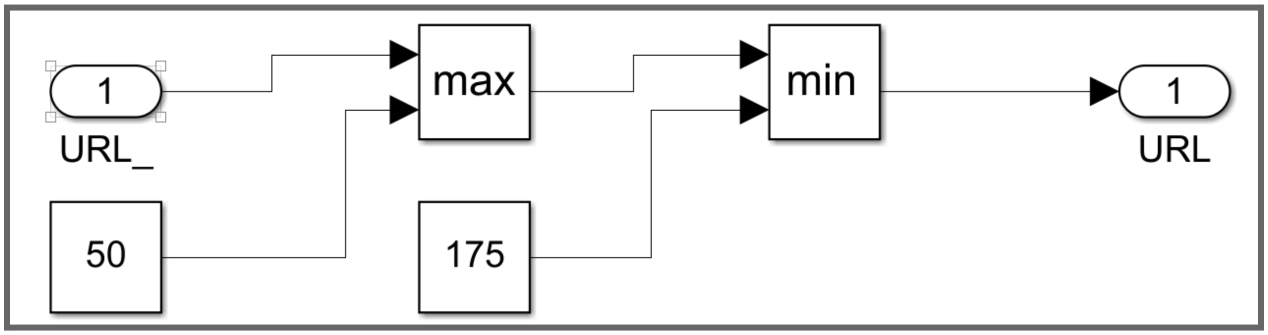


Figure 4.2.1.7.a: Subsystem which range checks Upper rate limit

The Upper rate limit is the maximum value that the heart rate can ever be at. It is checked to be between 50 and 175 bpm.

4.2.1.8 Rate Factor

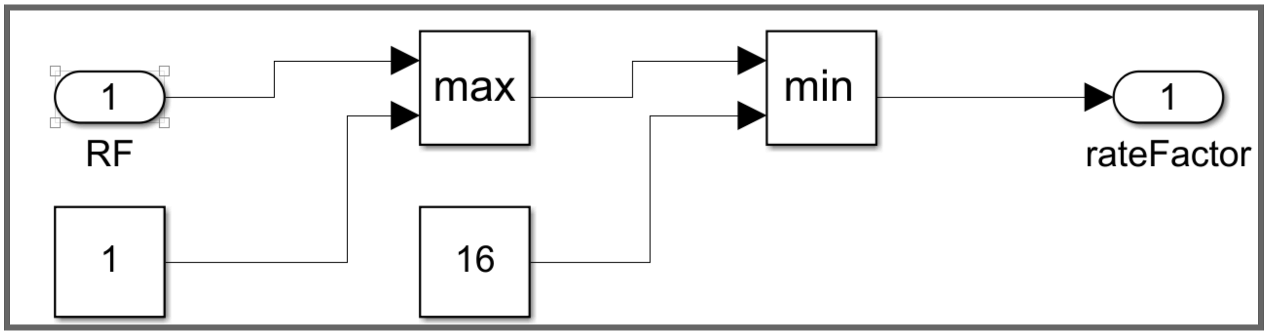


Figure 4.2.1.8.a: Subsystem which range checks rate factor

The rate factor is a multiplier that is applied to rate adaptive passing to change the amount of activity required to reach the upper rate limit. This value is range checked between 1 to 16 with 16 meaning that the MSR will reach the upper rate limit with less activity.

4.2.1.9 Recovery Time

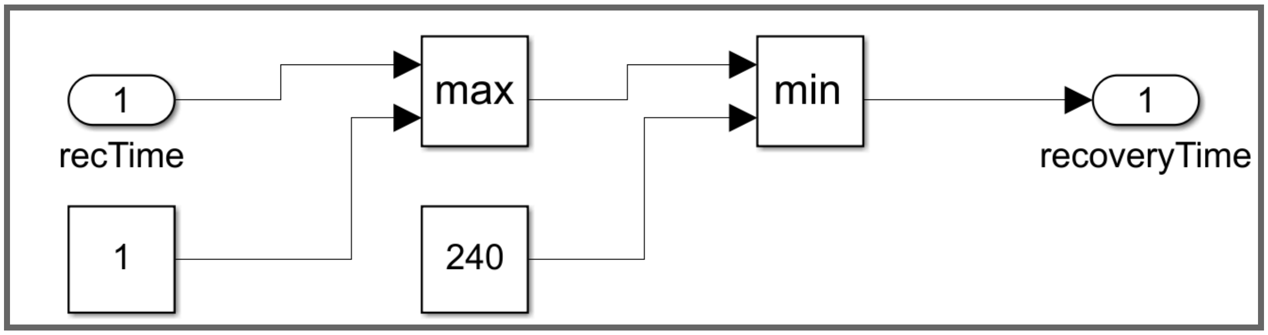


Figure 4.2.1.9.a: Subsystem which range checks recovery time

The recovery time is the amount of time required for the heart rate to change from its current position to the lower rate limit. This value is ranged checked to be between 1 and 240 milliseconds

4.2.1.10 Reaction Time

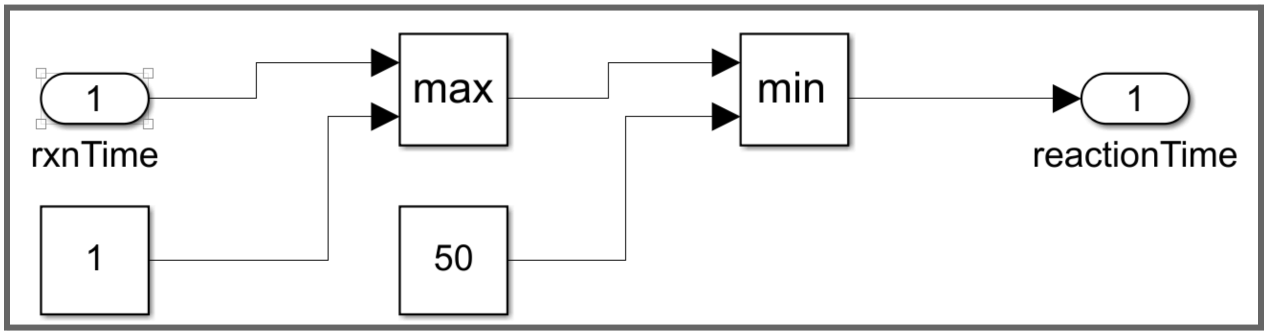


Figure 4.2.1.10.a: Subsystem which range checks reaction time

The reaction time is the amount of time required for the heart rate to increase from its current position to the required value. This value is ranged checked to be between 1 and 50 milliseconds

4.2.1.11 Activity Threshold

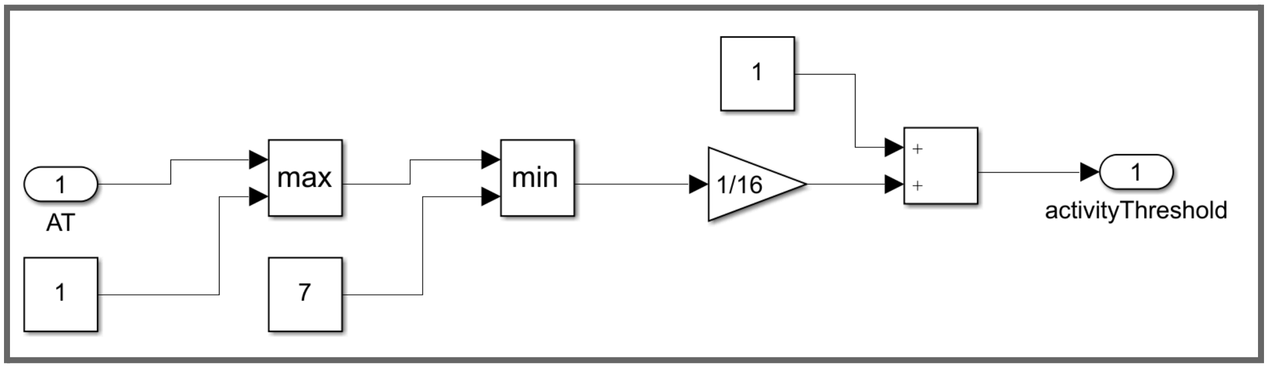


Figure 4.2.1.11.a: Subsystem which range checks activity threshold and modifies the value

The activity threshold is the minimum activity that needs to be achieved to cause a change in the heart rate. In the activity threshold, the value is range checked between 1 and 7 (very low and very high) and then made to be a smaller number so that a lower activity is required to reach the activity threshold, as an activity threshold of 7 is nearly impossible to reach.

4.2.2 Input Pins

ATR_CMP_DETECT and VENT_CMP_DETECT are direct readings from the pins D0 and D1, the readings are boolean values. The output from these pins is used in the pacemaker control chart, true means that a natural heart pulse is detected false means that no natural pulse was detected. To ensure effective use of the 4 variable model, the variables are converted to atrialSignalDetected and ventricularSignalDetected before leaving the input system so that if they are ever changed to a different method for detecting a natural pulse in the future only this section of code would need to be modified.

Other input pins include ATR_Signal, and Vent_Signal. These pins are direct readings from pins A0 and A1, these values are an analog voltage output that is sent directly to the serial output as egram data and are treated as data type single.

The serial number is a number specific to each pacemaker that is used as an identification. It is a UInt16, as it is not expected that there will be more than 65536 pacemakers being made.

4.2.3 Rate Adaptive Subsystem

A subsystem within the input subsystem is used to calculate the HR used for the rate adaptive pacing modes. The inputs to the subsystem are the LRL, URL, rateFactor, activity threshold, recovery and reaction time. The output is the heart rate. The goal of this subsystem is to generate a heart rate based on these inputs to be used in the rate adaptive variants of the modes. Activity level is based on an accelerometer built into the board. This subsystem is located inside of the input subsystem as it uses the accelerometer which is board specific.

The output heart rate is used only in rate adaptive modes, AOOR, VOOR, AAIR, VVIR, and DDDR.

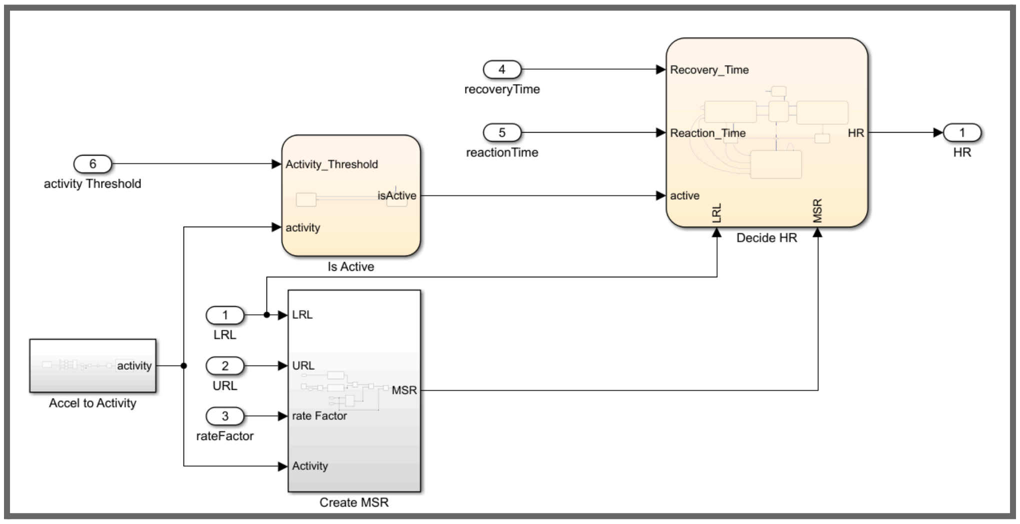


Figure 4.2.3: The rate adaptive subsystem

4.2.3.1 Accel To Activity

Accel to activity is where the accelerometer is used to calculate a current activity level.

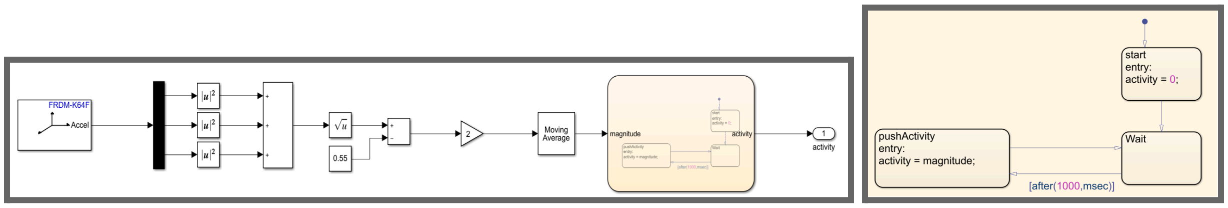


Figure 4.2.3.1.a: The accel to activity subsystem (left) and the activity emitter chart (right)

Acceleration is turned into a magnitude, by first being split into all 3 components, x, y, z, all 3 components are squared, summed and then square rooted. After, the magnitude is adjusted by subtracting 0.55 and doubling the value, this is done because when the pacemaker is not undergoing any movement, the pacemaker has a reading of 1g, to offset this, these calculations are performed. A moving average block is used to find the average value over one second, this is done so that a small spike in activity does not have a large effect on the overall heart rate, this is the activity value, there is a chart inside of this subsystem that makes it so that the activity value is only sent to the rest of the subsystem every second. One second was chosen so that the activity does not constantly change, allowing the values in following charts to have a set point to reach.

The pacemaker is set to only sample every 2ms, this is because if it is set to 1ms it causes all pacemaker firmware to run at half speed. To maintain the speed of the pacemaker firmware, which is needed for serial communication, this change is required.

4.2.3.2 Is Active Chart

This chart decides using the current activity level, if the pacemaker is active or not, and passes a boolean if it is considered to be active.

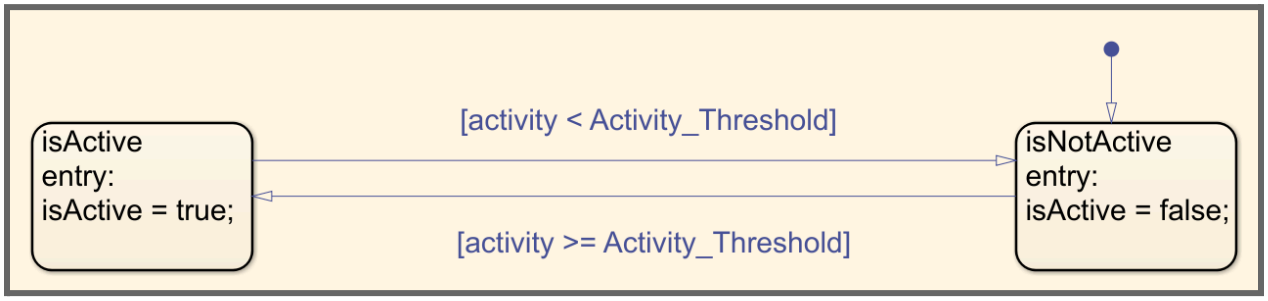


Figure 4.2.3.2.a: IsActive chart

Inside of this subsystem there is a chart there is 2 states, indicating if the activity threshold is exceeded, if the calculated activity is greater than the activity threshold, the isActive variable is true, otherwise it is false. The isActive value is passed into the decide HR chart.

4.2.3.3 Create MSR

In this subsystem activity, rate factor, URL, and LRL are used to create the MSR value, this functions are applied to create the following equation:

$$MSR = LRL + (URL - LRL) * \frac{ln(1+activity)}{ln(1+(16.1-rateFactor))}$$

Equation 4.2.3.3.a: equation used to calculate the MSR

This equation is used to ensure that the MSR value mimics how a real heart works. The rate factor is subtracted from 16.1, as the lower the denominator is the larger the MSR will be, by doing this, the higher rate factors increase MSR faster than lower rate factors.

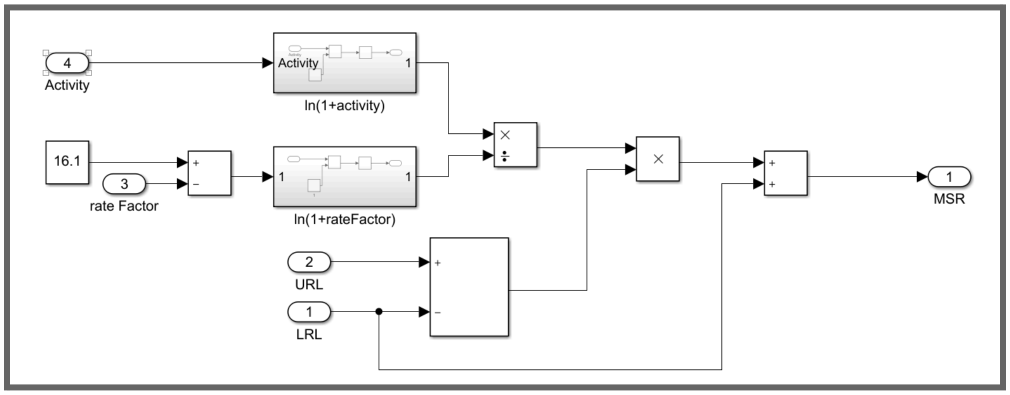
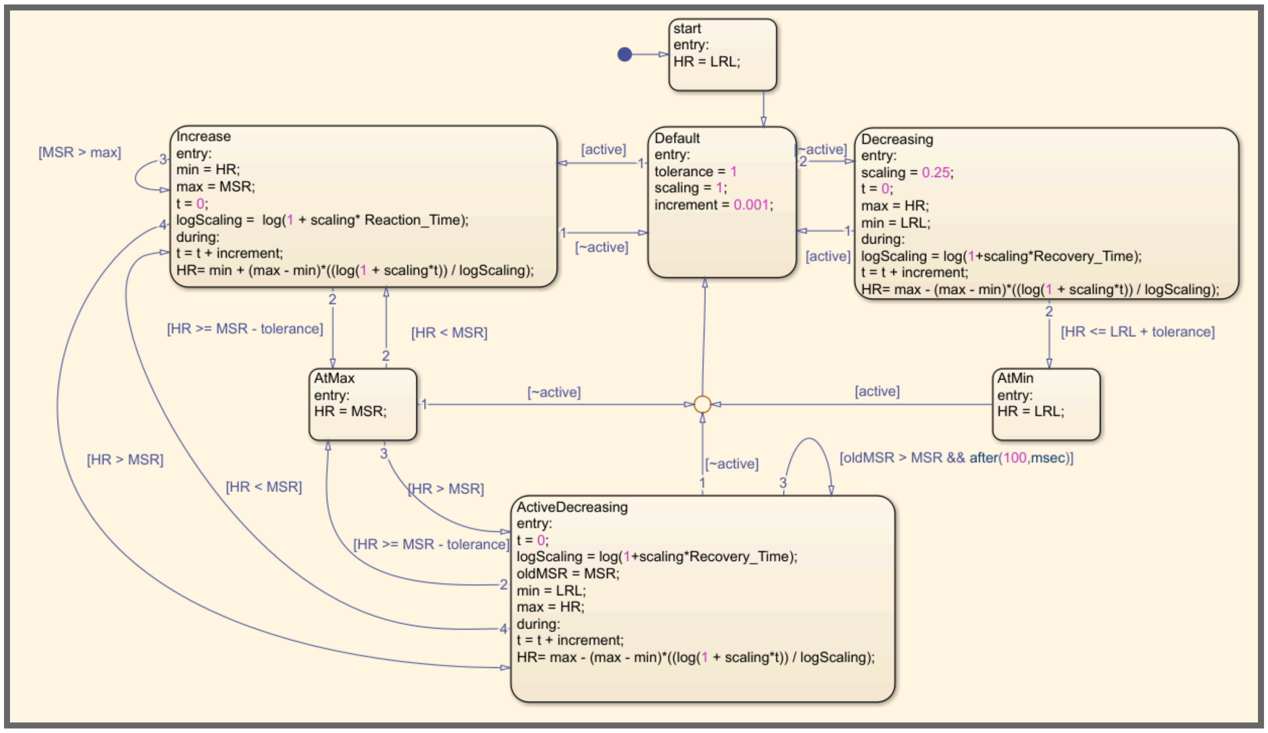


Figure 4.2.3.3.a: Subsystem used to calculate the MSR

4.2.3.4 Decide HR Chart

The Decide HR chart is where the values that will be used as the heart rate is calculated and then used in the rest of the pacemaker. The inputs to this chart are the LRL, MSR, recovery and reaction time, and if the pacemaker is active or not.

The values for reaction and recovery time are decreased significantly from what the values are said to be in the PACEMAKER document this is for both for testing and demoing purposes, as in the PACEMAKER document, the values are in the scale of minutes, instead of having the values in minutes, the values are instead in seconds, In a final product these values would be bounded by time in minutes.



```
start
entry:
HR = LRL;

Increase
entry:
min = HR;
max = MSR;
t = 0;
logScaling =  log(1 + scaling* Reaction_Time);
during:
t = t + increment;
HR= min + (max - min)*((log(1 + scaling*t)) / logScaling);

Default
entry:
tolerance = 1
scaling = 1;
increment = 0.001;

Decreasing
entry:
scaling = 0.25;
t = 0;
max = HR;
min = LRL;
during:
logScaling = log(1+scaling*Recovery_Time);
t = t + increment;
HR= max - (max - min)*((log(1 + scaling*t)) / logScaling);

AtMax
entry:
HR = MSR;

AtMin
entry:
HR = LRL;

ActiveDecreasing
entry:
t = 0;
logScaling = log(1+scaling*Recovery_Time);
oldMSR = MSR;
min = LRL;
max = HR;
during:
t = t + increment;
HR= max - (max - min)*((log(1 + scaling*t)) / logScaling);
```

Figure 4.2.3.4.a: Chart in which the heart rate is calculated

4.2.3.4.1 Active Is False

When this chart is first entered, the value is set to the LRL. This will only occur on pacemaker startup. When no active is false, the decreasing state is entered, this chart follows the following equation:

$$HR = max - (max - min) * \frac{ln(1+\frac{1}{4}t)}{ln(1+\frac{1}{4}*Recovery\,Time)}$$

Equation 4.2.3.4.1.a: Equation used to calculate the heart rate in the Decreasing state

Where max is the heart rate at entry, min is the lower rate limit, and t is a counter that counts in milliseconds. This equation allows for the heart rate to reach the required heart rate gradually mimicking a real heart.

When the heart rate value has reached the lower rate limit within a tolerance, the chart becomes AtMin where the heart rate is set to the lower rate limit. If at any point while in the Decreasing or AtMin state, when the pacemaker has activity exceeding the threshold, the increasing state is entered.

4.2.3.4.2 Active Is True

The Increasing state uses the following equation:

$$HR = min + (max - min) * \frac{ln(1+t)}{ln(1+Reaction\ Time)}$$

Equation 4.2.3.4.2.a: Equation used to calculate the heart rate in the Increasing state

Where min is the heart rate at entry, max is the MSR at entry, and t is a counter that counts in milliseconds. If the MSR increases, (this occurs when the activity increases beyond the activity at entrance), the state is reentered, setting tt back to 0 and setting a new min and max. This equation allows the heart rate to reach the MSR gradually.

If the MSR decreases to a value lower than the heart rate, the chart will enter the ActiveDecreasing state, which follows the following equation:

$$HR = max - (max - min) * \frac{ln(1+t)}{ln(1+Recovery\ Time)}$$

Equation 4.2.3.4.2.b: Equation used to calculate the heart rate in the ActiveDecreasing state

Where max is the heart rate at entry, min is the lower rate limit, and t is a counter that counts in milliseconds. This equation allows for the heart rate to reach the required heart rate gradually mimicking a real heart.

If the chart is in the ActiveDecreasing state, and the MSR increases beyond the heart rate, the chart will return to the ActiveIncreasing state

When the heart rate value has reached the MSR within a tolerance, the chart becomes AtMax where the heart rate is set to the MSR. If at any point while in the Increasing, ActiveDecreaing or AtMax state the activity decreases below the activity threshold, the Decreasing state will be entered.

4.2.3.4.3 After deciding heart rate

After the heart rate is decided, it is range checked, and converted to pulse period, and a correction factor is applied.

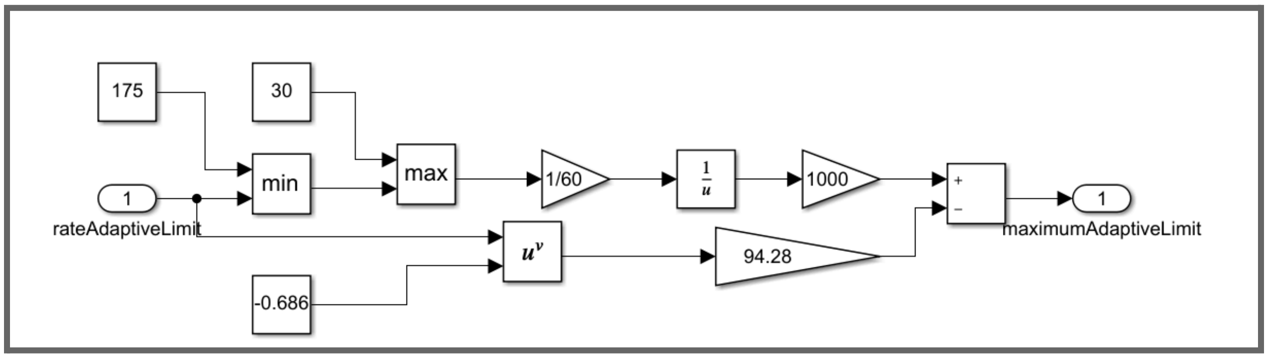


Figure 4.2.3.4.3.a: Subsystem in which the heart rate is bound checked, converted to pulse period, and a correction factor is applied.

4.3 Pacemaker Control Chart

The pacemaker control chart is the location where the logic of each mode is controlled and the logic of switching between the modes. The system functions by having the Off mode in the center and a branch to and from each of the states. This means that the states can not switch between each other directly and rather have to pass through the off safe state. This is beneficial in two ways as it prevents unexpected behaviour and eliminates the need to connect each mode to one another which would become a mess once the number of modes grows. The state for a specific mode will be entered once the mode is set to that mode and exited once it is no longer equal to that mode. To eliminate confusing state transitions which use “magic numbers,” an initialization state is created which will assign numeric values to each of the modes. This means that the transitions between states can be as simple as when equal to mode or when not equal to mode, rather than using seemingly random numbers for the transitions.

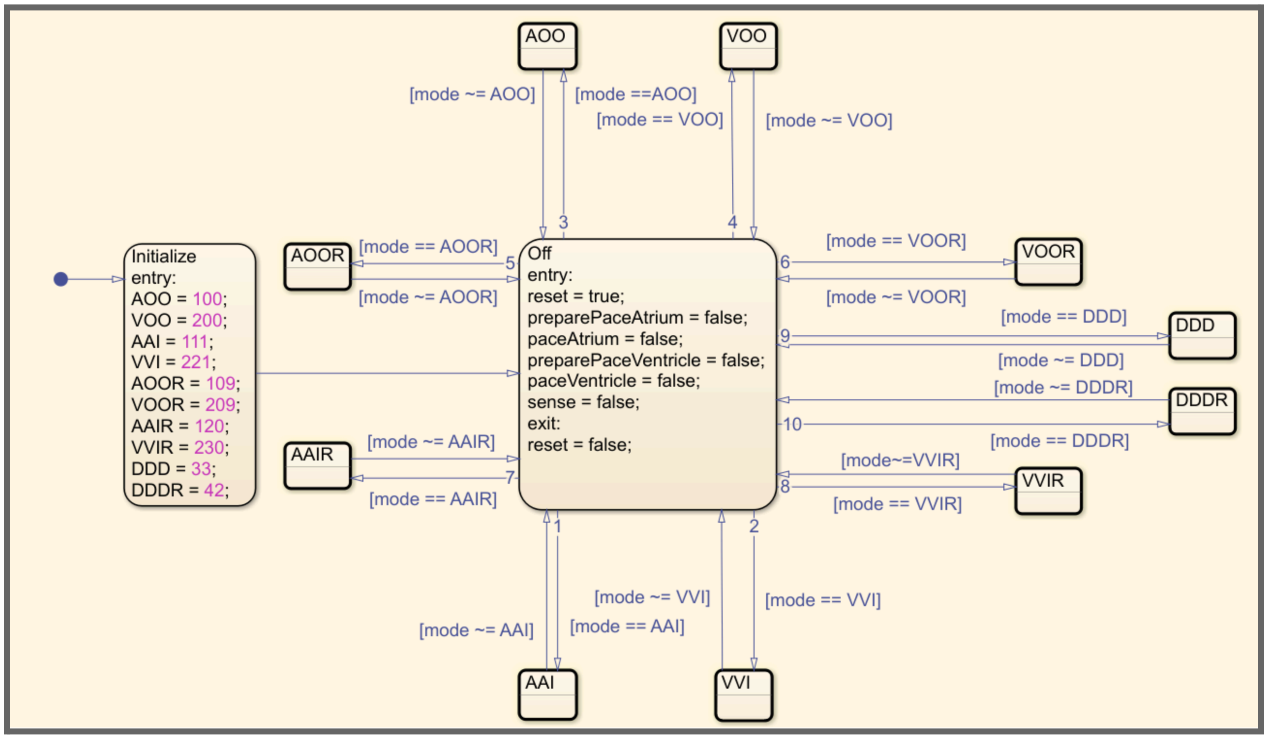


Figure 4.3.a: Inside of Pacemaker Control Chart

Within the Off mode are the many variables that will control states and pin behaviour in the output system. They are as follows:

Reset - Causes pin control to return to the off mode.

paceAtrium - Will turn on the necessary pins to pace the atrium.

paceVentricle - Will turn on the necessary pins to pace the ventricle.

preparePaceAtrium - Will turn on the necessary pins to prepare an atrium pace.

preparePaceVentricle - Will turn on the necessary pins to prepare an ventricle pace.

sense - Will turn on the necessary pins to sense the natural atrial pulses and ventricle pulses.

4.3.1 AOO Mode

The AOO subchart can only be active if the AOO mode is active. The subchart contains 2 states, the Prep state and the Shock state. The Prep state turns on the preparePaceAtrium charging state on entry. When exiting the state, preparePaceAtrium is turned off. The Shock state turns on the paceAtrium charging state on entry. When exiting the state, paceAtrium is turned off. To change the state from Prep to Shock, maximumLimit minus atrialPulseWidth amount of time must take place. To change the state from Shock to Prep, atrialPulseWidth amount of time must take place. Remember that the maximumLimit variable is equal to the pulse period, subtracting by the pulse width ensures that the period is met each cycle and that the mode is operating at the desired heart rate. The AOO subchart will be exited from when the mode is no longer AOO.

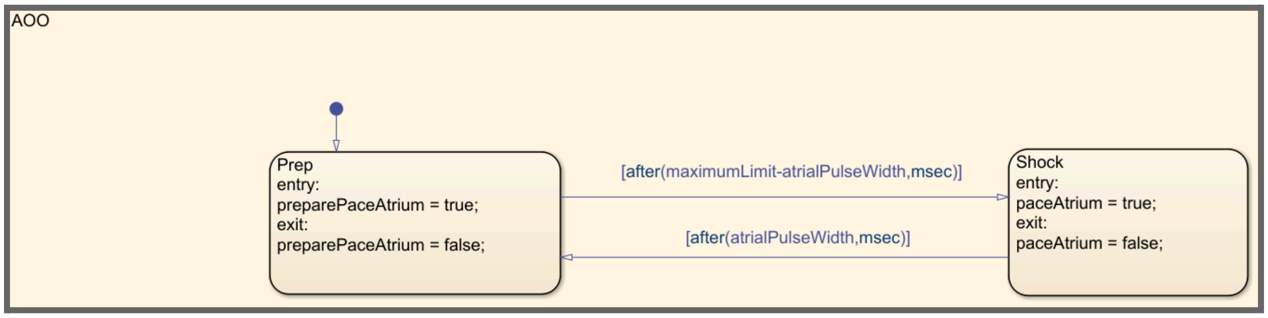

AOO
Prep
entry:
preparePaceAtrium = true;
exit:
preparePaceAtrium = false;
[after(maximumLimit-atrialPulseWidth,msec)]
[after(atrialPulseWidth,msec)]
Shock
entry:
paceAtrium = true;
exit:
paceAtrium = false;

Figure 4.3.1.a: Inside of AOO Subchart

4.3.2 VOO Mode

The VOO mode operates identically to AOO except pacing the ventricle instead. The VOO subchart can only be active if the VOO mode is active. The subchart contains 2 states, the Prep state and the Shock state. The Prep state turns on the preparePaceVentricle charging state on entry, which turns on all pins required for preparing to pace the ventricle . When exiting the state, preparePaceVentricle is turned off. The Shock state turns on the paceVentricle charging state on entry, which turns on all pins required to pace the ventricle when exiting the state, paceVentricle is turned off. To change the state from Prep to Shock, maximumLimit minus ventricularPulseWidth amount of time must take place. To change the state from Shock to Prep, ventricularPulseWidth amount of time must take place. Remember that the maximumLimit variable is equal to the pulse period, subtracting by the pulse width ensures that the period is met each cycle and that the mode is operating at the desired heart rate. The VOO subchart will be exited from when the mode is no longer VOO.

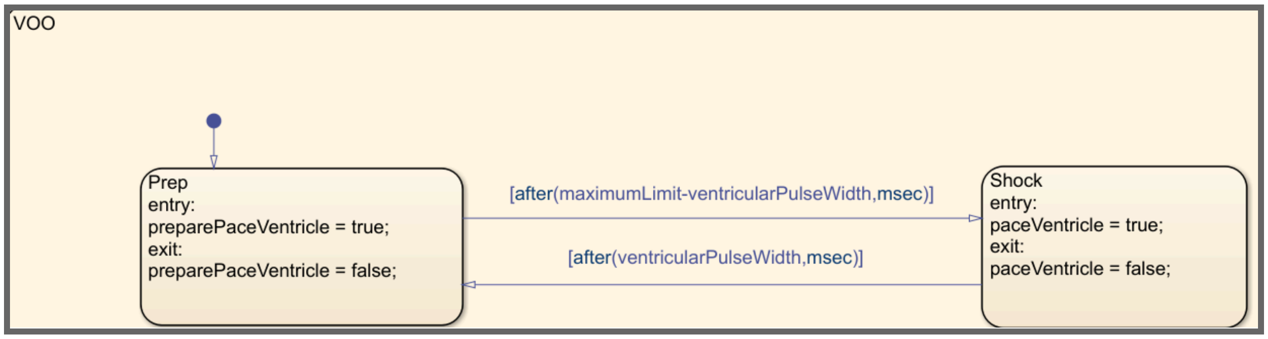

VOO
Prep
entry:
preparePaceVentricle = true;
exit:
preparePaceVentricle = false;
[after(maximumLimit-ventricularPulseWidth,msec)]
[after(ventricularPulseWidth,msec)]
Shock
entry:
paceVentricle = true;
exit:
paceVentricle = false;

Figure 4.3.2.a: Inside of VOO Subchart

4.3.3 AAI Mode

The AAI subchart is active if AAI mode is active. The subchart contains 3 states, the SenseDelay, SenseNoDelay, and the Pace state. The Sensing states turn on the preparePaceAtrium and the sense charging states on entry, when exiting, preparePaceAtrium, and sense are turned off. The Shock state turns on the paceAtrium charging state on entry, when exiting the state, paceAtrium is turned off. In both sensing states a maximum of the lower rate limit (converted to maximumLimit) amount of time of sensing takes place, if no natural pulses occur in that period, the pacemaker sends a pulse. When in the SenseNoDelay state if a natural pulse is detected the state will change to the SenseDelay state. When in the SenseDelay state a period of 100ms must be passed to loop back into itself. To change the state from Pace to SenseNoDelay, atrialPulseWidth amount of time must take place. This ensures that the pace lasts for the required amount of time. The AAI subchart will be exited from when the mode is no longer AAI.

This design decision was made to avoid either of the following cases. If only SenseDelay was an option then consider the case where the pacemaker pulses and immediately afterward the heart pulses. In this case, the natural pulse may not be detected and the next artificial pulse may be delivered too soon or unnecessarily. This raises the question of whether the 100 ms wait time on the cycle is even required. However, consider a case with a longer pulse rate. Without the counter, this longer natural pulse would continue to be sensed until its falling edge which would mean that the next possible artificial pulse would not be the lower rate limit later but rather the lower rate limit plus the duration of the natural pulse. Both of these cases caused problems for previous iterations of the Pacemaker Firmware, hence the current design which is able to detect natural pulses immediately following an artificial pulse while also only detecting the rising edge of the natural pulse and not the whole thing. The 100 ms wait time was settled on as it is sufficiently large to avoid detecting the same natural pulse multiple times while being smaller than the period between pulses of the fastest possible heart rate. Additionally, unlike AOO the wait before pacing is maximumLimit without subtracting the pulse width. This was done intentionally, as the objective is to sense a heart rate of the lower rate limit and it is impossible to both sense for and pace at the lower rate limit. It was decided that sense time was more important. Additionally, we are still within the tolerance of the lower rate limit as discussed earlier it is a maximum of about 1 ms off and adding the maximum of a 1.9 ms pulse width to this renders the error still within an 8 ms tolerance per the requirements.

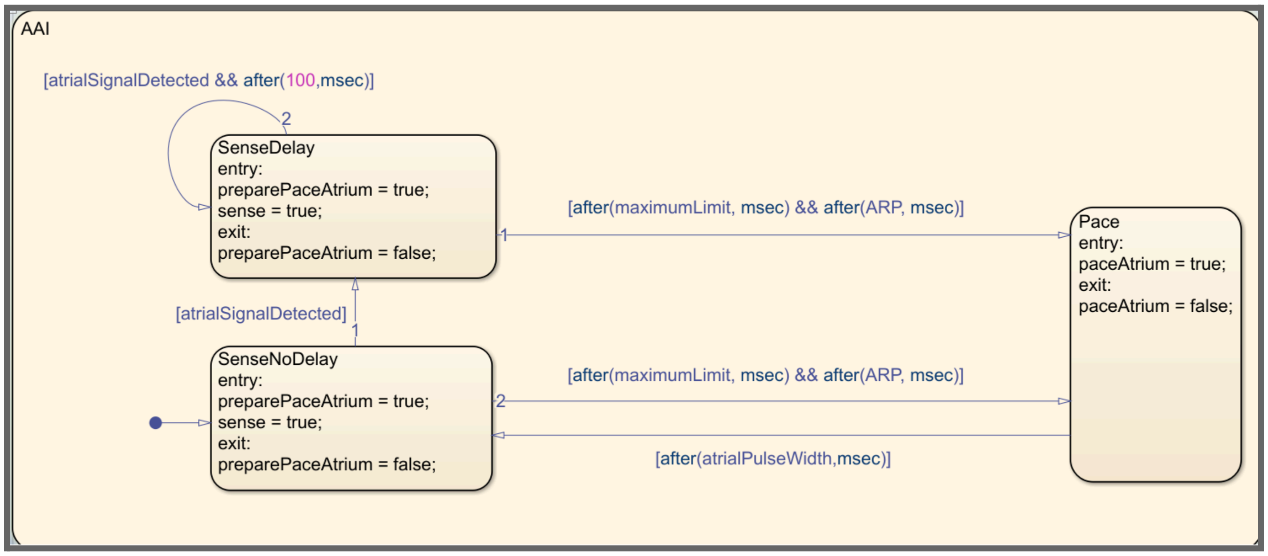

AAI
[atrialSignalDetected && after(100,msec)]
2
SenseDelay
entry:
preparePaceAtrium = true;
sense = true;
exit:
preparePaceAtrium = false;
[after(maximumLimit, msec) && after(ARP, msec)]
1
Pace
entry:
paceAtrium = true;
exit:
paceAtrium = false;
[atrialSignalDetected]
1
SenseNoDelay
entry:
preparePaceAtrium = true;
sense = true;
exit:
preparePaceAtrium = false;
[after(maximumLimit, msec) && after(ARP, msec)]
2
[after(atrialPulseWidth,msec)]

Figure 4.3.3.a: Inside of AAI Subchart

4.3.4 VVI Mode

The VVI subchart works exactly the same as the AAI subchart but using variables associated with the ventricle instead of the atrium, these variables have the same names but use ventricular instead of atrial and ventricle instead of atrium. All discussion of how this mode works and the justification of design decisions can be found in 4.3.3 AAI Mode.

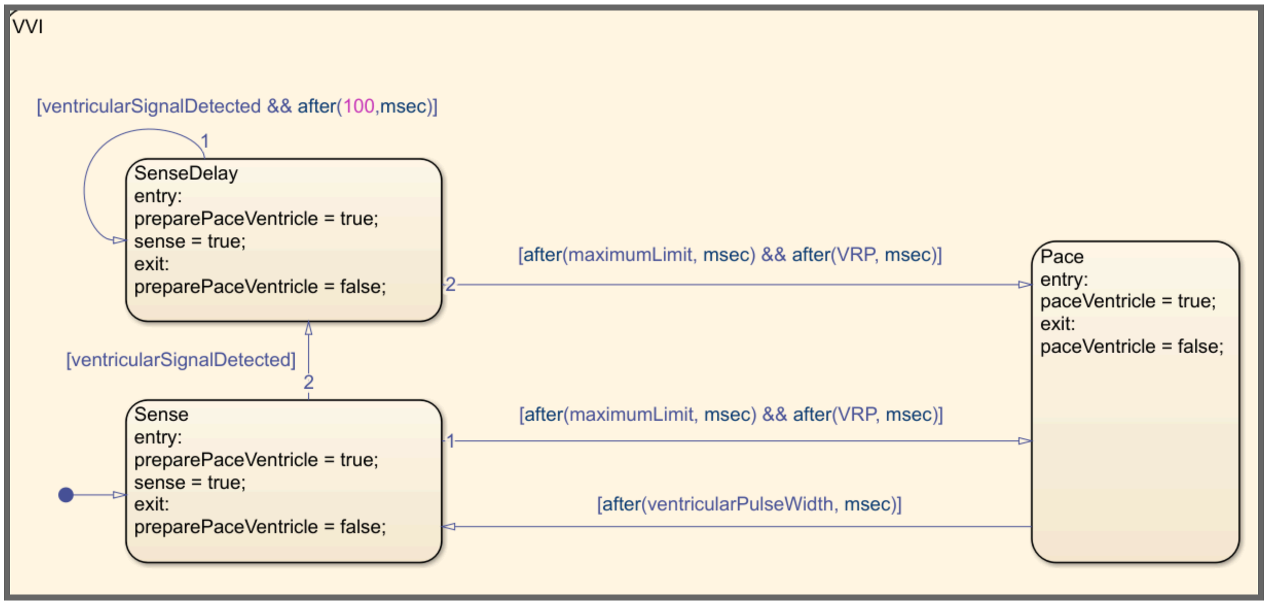

VVI
[ventricularSignalDetected && after(100,msec)]
1
SenseDelay
entry:
preparePaceVentricle = true;
sense = true;
exit:
preparePaceVentricle = false;
[after(maximumLimit, msec) && after(VRP, msec)]
2
Pace
entry:
paceVentricle = true;
exit:
paceVentricle = false;
[ventricularSignalDetected]
2
Sense
entry:
preparePaceVentricle = true;
sense = true;
exit:
preparePaceVentricle = false;
[after(maximumLimit, msec) && after(VRP, msec)]
1
[after(ventricularPulseWidth, msec)]

Figure 4.3.4.a: Inside of VVI Subchart

4.3.5 DDD Mode

The DDD subchart is active if the DDD mode is active. The DDD mode is significantly more complicated than the other modes discussed as it must be able to pace both chambers with aspects of inhibition and triggering built in. This means that the DDD mode has 12 total states, however, removing the reset states whose only purpose is to facilitate switching between pacing the different chambers, there are 9 other states. The DDD mode has four possible situations it can be operating in.

1) The heart is not beating or the heart is in bradycardia and the AV delay is too long. This mode is characteristic of the pacemaker pacing both the atria and ventricles. The pacemaker will start in the initial state of PrepAtrium, then after LRL minus the time it takes to complete the rest of the cycle, it will move to ShockAtrium where it will shock the atrium. It only shocked the atrium because it never detected a natural atrial pulse during the preparation period. After the APW the model will move to discharge blocking cap state, where it will begin to prepare another atrial pulse. This may seem odd as we still have not paced the ventricle, however, another pace must be prepared (but not delivered) to discharge the C21 capacitor within the pacing circuitry. This prevents a minor shock from shocking the atria when we later switch to pacing the ventricles. After discharging the blocking capacitor the reset state is entered which effectively turns the pacemaker to off mode for a brief millisecond. This is done to reset the pacing pin chart to the default safe state and avoid unexpected and dangerous pacing behaviour. The PrepVentricle state is then entered which prepares the ventricular pace. After AV delay - APW time the ventricle is shocked. Similar to switching from the atria to the ventricles, to switch the other way the C21 blocking capacitor must be discharged again and the reset state must be passed through. Returning to the initial state and preparing to pace the atrium again, the whole cycle takes LRL amount of time. This may not be evident from all of the times divided by two and added together but if the total of the whole journey is summed everything except LRL will cancel out.

2) The second situation is if the heart rate is above LRL and the AV delay is short enough. This will be characteristic of no pacing. The cycle will prepare the atrium to reset when the atrial signal is detected. There must be a reset here in case we do not end up detecting a ventricular pulse and are required to pace. However, in this case the sensed atrium state will be entered and once the ventricular pulse is detected the cycle will pass through another reset to prepare for atrial pacing and return to the initial prepare atrial pace state. This cycle detected both natural atrial and ventricular pulses and as such inhibited the artificial paces.

3) The next situation is one in which the atria is beating faster than LRL but either the ventricle is not beating or is taking longer than AV delay to beat. This means that after sensing an atrial pulse and it taking too long for the following natural ventricular pulse to occur, the pacemaker triggers an artificial ventricular pulse. This cycle begins the same as the second one, going from preparing the atrium to reset to sensed atrium, however, now the ventricular pulse is not detected in time and the cycle will deliver the ventricular pulse it has been preparing before discharging the blocking capacitor, resetting, and returning to the initial state. The wait time to detect the ventricular pulse is AV delay. This is because the rising edge of the atrial pulse is what is being detected and as such there is no need to subtract the atrial pulse width like was done in situation one. Notably, unlike the AAI and VVI modes which needed the 100 ms wait timer before detecting the same chamber again, because the maximum pulse width is 30 ms

and the minimum AV delay possible is 30 ms, there is no risk of the pacemaker detecting the same atrial pulse twice as was the issue with those other modes.

4) The final situation is one in which the atria is not beating or below LRL but the ventricle is beating less than AV delay after the delivered artificial atrial pulse. This is the situation in which the atrial pulse is being delivered but the artificial ventricular pace is being inhibited by a natural sensed one. This cycle begins the same way as the first situation did where it delivers an atrial pulse. At this point a ventricular pulse could be detected in the shock atrium state, the discharging the blocking capacitor state, or the preparing ventricle state. Note that the reset state is not included and this is simply because it the model only spends a millisecond here and it is not necessary to sense here as well when the heart (even when set to 1 ms) will take a minimum of slightly longer than a millisecond to deliver a natural pace. One may have noticed the t variable in any of the states which proceed a ventricular sense, this is used as a counter to determine how much time had elapsed in that state before the sense. This is necessary as even if we sense a ventricular pace during the shocking of the atrium or discharging of the blocking cap, those activities still need to be finished before we can move on. To do this, when the pace is detected they will pace to an identical state which lasts for the required time minus t. This ensures that even if the ventricular pulse is sensed as soon as the state is entered, there is still the opportunity to finish the atrial pulse or capacitor discharging. The final wait condition before returning back to the reset and initial states is another timer to ensure this whole cycle lasts a LRL amount of time.

The final notable thing about this mode is that the SW3 button on the pacemaker board can be pressed to inhibit ventricular pacing in this mode. This is done by making all the sensing ventricle conditions also include OR pushing the button. This will inhibit ventricular pacing when the button is pressed as mentioned in the requirements for this mode. The DDD subchart will be exited from when the mode is no longer DDD.

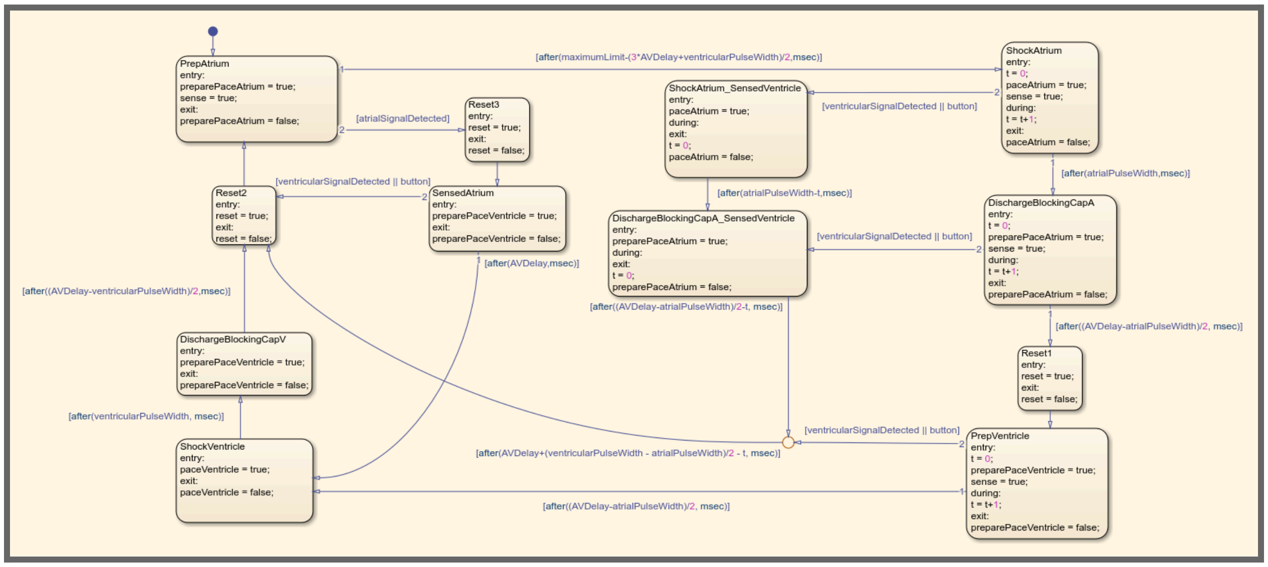

PrepAtrium
entry:
preparePaceAtrium = true;
sense = true;
exit:
preparePaceAtrium = false;
[after(maximumLimit-(3*AVDelay+ventricularPulseWidth)/2,msec)]
[atrialSignalDetected]
Reset3
entry:
reset = true;
exit:
reset = false;
ShockAtrium
entry:
t = 0;
paceAtrium = true;
sense = true;
during:
t = t+1;
exit:
paceAtrium = false;
ShockAtrium_SensedVentricle
entry:
paceAtrium = true;
during:
exit:
t = 0;
paceAtrium = false;
[ventricularSignalDetected || button]
Reset2
entry:
reset = true;
exit:
reset = false;
SensedAtrium
entry:
preparePaceVentricle = true;
exit:
preparePaceVentricle = false;
[after(AVDelay,msec)]
[after(atrialPulseWidth-t,msec)]
[after(atrialPulseWidth,msec)]
DischargeBlockingCapA_SensedVentricle
entry:
preparePaceAtrium = true;
during:
exit:
t = 0;
preparePaceAtrium = false;
DischargeBlockingCapA
entry:
t = 0;
preparePaceAtrium = true;
sense = true;
during:
t = t+1;
exit:
preparePaceAtrium = false;
[after((AVDelay-ventricularPulseWidth)/2,msec)]
[after((AVDelay-atrialPulseWidth)/2-t, msec)]
[after((AVDelay-atrialPulseWidth)/2, msec)]
DischargeBlockingCapV
entry:
preparePaceVentricle = true;
exit:
preparePaceVentricle = false;
Reset1
entry:
reset = true;
exit:
reset = false;
[after(ventricularPulseWidth, msec)]
ShockVentricle
entry:
paceVentricle = true;
exit:
paceVentricle = false;
[after(AVDelay+(ventricularPulseWidth - atrialPulseWidth)/2 - t, msec)]
PrepVentricle
entry:
t = 0;
preparePaceVentricle = true;
sense = true;
during:
t = t+1;
exit:
preparePaceVentricle = false;
[after((AVDelay-atrialPulseWidth)/2, msec)]

Figure 4.3.5.a: Inside of DDD Subchart

4.3.6 AOOR Mode

The AOOR subchart works exactly the same as the AOO subchart, but uses the rateAdaptiveLimit instead of the maximumLimit variable where applicable.

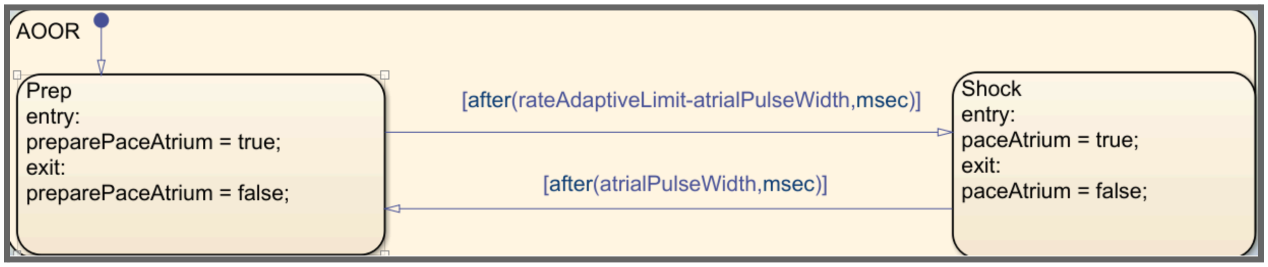

AOOR
Prep
entry:
preparePaceAtrium = true;
exit:
preparePaceAtrium = false;
[after(rateAdaptiveLimit-atrialPulseWidth,msec)]
[after(atrialPulseWidth,msec)]
Shock
entry:
paceAtrium = true;
exit:
paceAtrium = false;

Figure 4.3.6.a: Inside of AOOR Subchart

4.3.7 VOOR Mode

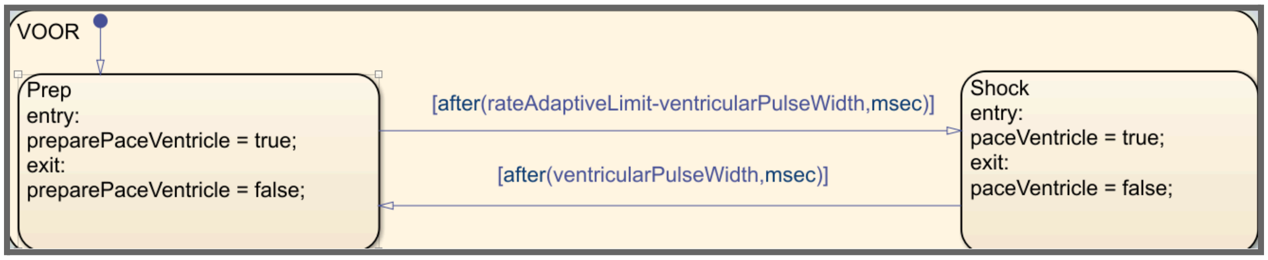

VOOR
Prep
entry:
preparePaceVentricle = true;
exit:
preparePaceVentricle = false;
[after(rateAdaptiveLimit-ventricularPulseWidth,msec)]
[after(ventricularPulseWidth,msec)]
Shock
entry:
paceVentricle = true;
exit:
paceVentricle = false;

Figure 4.3.7.a: Inside of VOOR Subchart

The VOOR subchart works exactly the same as the VOO subchart, but uses the rateAdaptiveLimit instead of the maximumLimit variable where applicable.

4.3.8 AAIR Mode

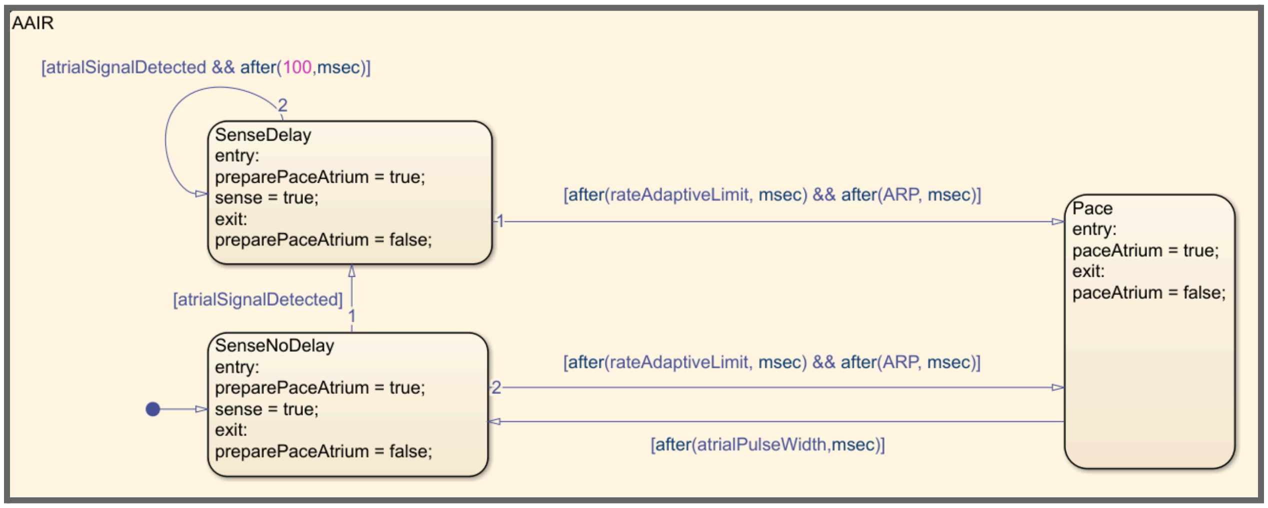

AAIR
[atrialSignalDetected && after(100,msec)]
2
SenseDelay
entry:
preparePaceAtrium = true;
sense = true;
exit:
preparePaceAtrium = false;
1
[after(rateAdaptiveLimit, msec) && after(ARP, msec)]
[atrialSignalDetected]
1
SenseNoDelay
entry:
preparePaceAtrium = true;
sense = true;
exit:
preparePaceAtrium = false;
2
[after(rateAdaptiveLimit, msec) && after(ARP, msec)]
[after(atrialPulseWidth,msec)]
Pace
entry:
paceAtrium = true;
exit:
paceAtrium = false;

Figure 4.3.8.a: Inside of AAIR Subchart

The AAIR subchart works exactly the same as the AAI subchart, but uses the rateAdaptiveLimit instead of the maximumLimit variable where applicable.

4.3.9 VVIR Mode

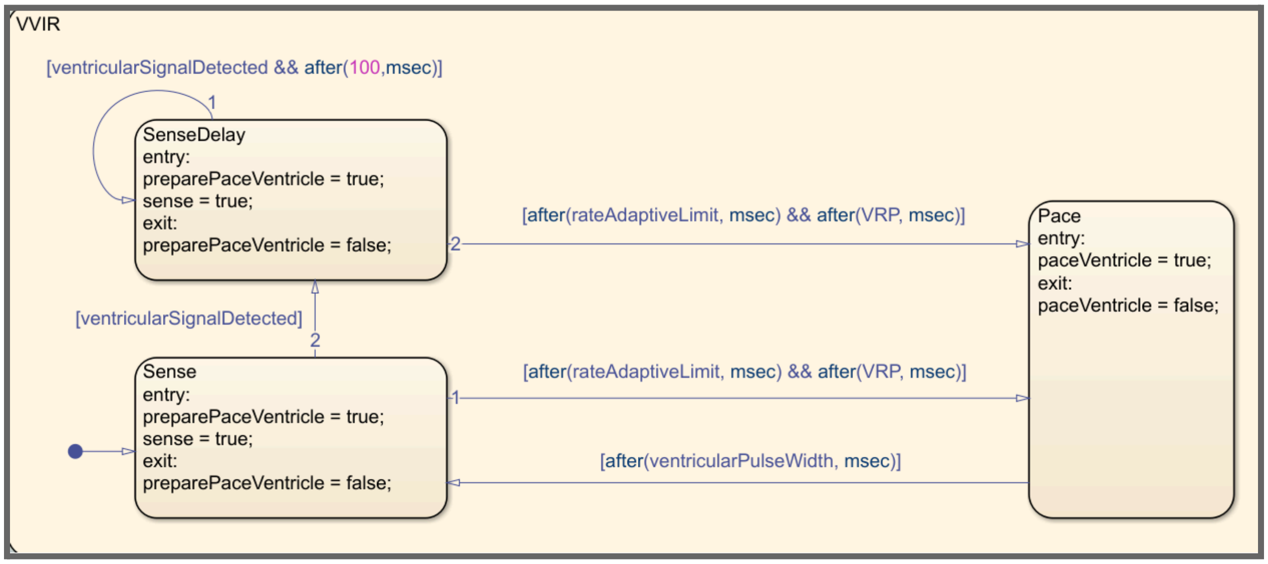

VVIR
[ventricularSignalDetected && after(100,msec)]
1
SenseDelay
entry:
preparePaceVentricle = true;
sense = true;
exit:
preparePaceVentricle = false;
2
[after(rateAdaptiveLimit, msec) && after(VRP, msec)]
Pace
entry:
paceVentricle = true;
exit:
paceVentricle = false;
[ventricularSignalDetected]
2
Sense
entry:
preparePaceVentricle = true;
sense = true;
exit:
preparePaceVentricle = false;
1
[after(rateAdaptiveLimit, msec) && after(VRP, msec)]
[after(ventricularPulseWidth, msec)]

Figure 4.3.9.a: Inside of VVIR Subchart

The VVIR subchart works exactly the same as the VVI subchart, but uses the rateAdaptiveLimit instead of the maximumLimit variable where applicable.

4.3.10 DDDR Mode

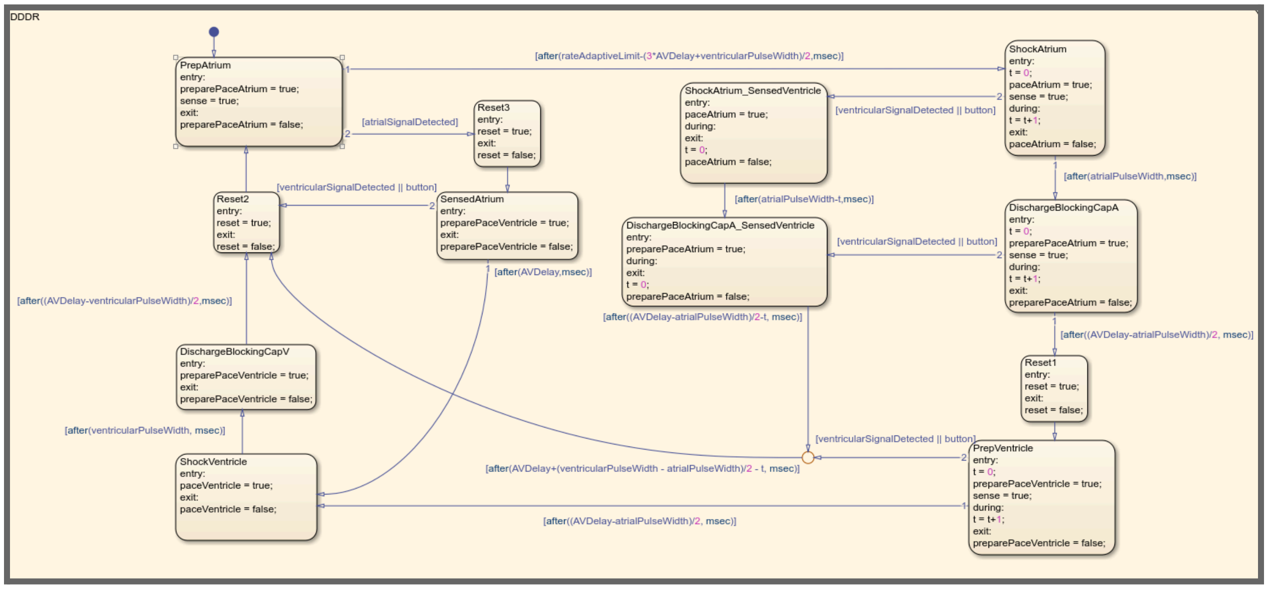

DDDR
PrepAtrium
entry:
preparePaceAtrium = true;
sense = true;
exit:
preparePaceAtrium = false;
1
[after(rateAdaptiveLimit-(3*AVDelay+ventricularPulseWidth)/2,msec)]
2
[atrialSignalDetected]
Reset3
entry:
reset = true;
exit:
reset = false;
ShockAtrium
entry:
t = 0;
paceAtrium = true;
sense = true;
during:
t = t+1;
exit:
paceAtrium = false;
ShockAtrium_SensedVentricle
entry:
paceAtrium = true;
during:
exit:
t = 0;
paceAtrium = false;
2
[ventricularSignalDetected || button]
1
[after(atrialPulseWidth,msec)]
[after(atrialPulseWidth-t,msec)]
[ventricularSignalDetected || button]
Reset2
entry:
reset = true;
exit:
reset = false;
2
SensedAtrium
entry:
preparePaceVentricle = true;
exit:
preparePaceVentricle = false;
1
[after(AVDelay,msec)]
DischargeBlockingCapA_SensedVentricle
entry:
preparePaceAtrium = true;
during:
exit:
t = 0;
preparePaceAtrium = false;
[ventricularSignalDetected || button]
2
DischargeBlockingCapA
entry:
t = 0;
preparePaceAtrium = true;
sense = true;
during:
t = t+1;
exit:
preparePaceAtrium = false;
[after((AVDelay-ventricularPulseWidth)/2,msec)]
[after((AVDelay-atrialPulseWidth)/2-t, msec)]
1
[after((AVDelay-atrialPulseWidth)/2, msec)]
DischargeBlockingCapV
entry:
preparePaceVentricle = true;
exit:
preparePaceVentricle = false;
Reset1
entry:
reset = true;
exit:
reset = false;
[after(ventricularPulseWidth, msec)]
[ventricularSignalDetected || button]
2
[after(AVDelay+(ventricularPulseWidth - atrialPulseWidth)/2 - t, msec)]
PrepVentricle
entry:
t = 0;
preparePaceVentricle = true;
sense = true;
during:
t = t+1;
exit:
preparePaceVentricle = false;
ShockVentricle
entry:
paceVentricle = true;
exit:
paceVentricle = false;
1
[after((AVDelay-atrialPulseWidth)/2, msec)]

Figure 4.3.10.a: Inside of DDDR Subchart

The DDDR subchart works exactly the same as the DDD subchart, but uses the rateAdaptiveLimit instead of the maximumLimit variable where applicable.

4.4 Output Subsystem

The goal of the output subsystem is to handle all outputs and ensure that only valid outputs occur. This is done using inputs from the previous two systems. This again is in support of the 4 variable model as the interaction with hardware is separate from the decisions being made based on the mode and operation. If the hardware was to change, for instance using an actual pacemaker the sensing and pacing circuitry would likely be different with a different number of switches and capacitors which may need to be interacted with in different orders. Rather than having to go and implement those changes throughout the model multiple times it would need to only be done once in the output system. The layout of the output system also helps to minimize code duplications as every mode uses the same pacing and sensing code.

4.4.1 LED Pin Control

Although the mode is no longer switched with the onboard button, it can still be valuable to be able to know which mode the pacemaker is in to facilitate testing. Having a visual indicator of the mode without needing to launch HeartView can be helpful as for instance it shows if a parameter change request sent from the DCM has been implemented. Additionally, it was useful during the demo as the mode was switched then a discussion would take place and it would be useful to easily come back to the pacemaker later and know what the expected behaviour is based on the colour of the LED without necessarily knowing the previous parameters sent by the DCM. To solve this issue the LED on the pacemaker board is used to indicate the mode. The operation of this is controlled by the LED Pin Control chart within the output system. Inside the chart there are states that associate a mode with a colour using the variables r, b, and g for red, blue, and green, see Table 4.4.1.a and Figure 4.4.1.a and 4.4.1.b.

To limit the use of “magic numbers” within the code, numeric values are associated with each mode at initialization so that the transition conditions within the chart can use the names of the modes rather than seemingly random numbers. Additionally, assigned in this initialization state is the period of time in seconds which the blinking of the LED for the rate adaptive modes will take.

Table 4.4.1.a: Pacemaker modes and associated LED colors.

Mode	Off	AOO	VOO	AAI	VVI	DDD	AOOR	VOOR	AAIR	VVIR	DDDR
Associated LED Color/Pattern	Solid White	Solid Red	Solid Green	Solid Purple	Solid Blue	Solid Cyan	Blink Red	Blink Green	Blink Purple	Blink Blue	Blink Cyan

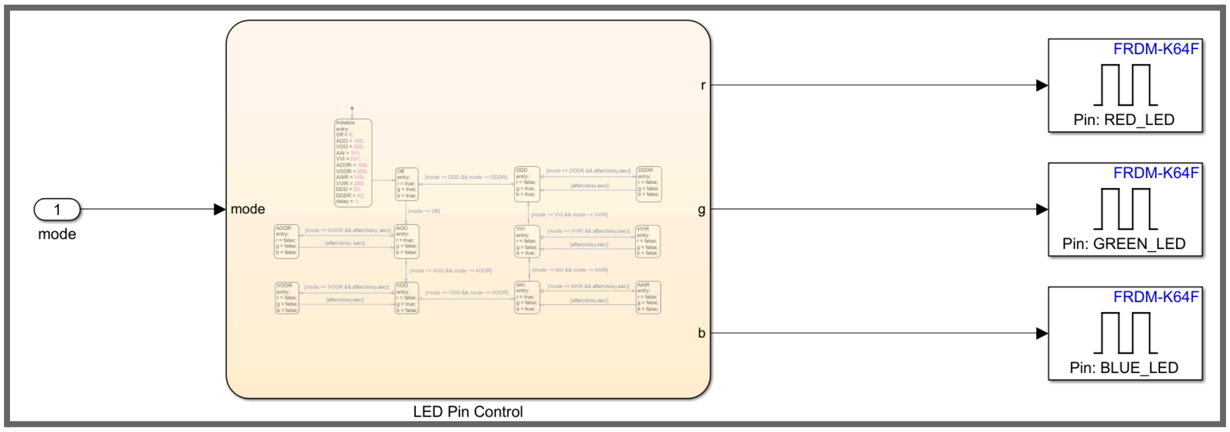


Figure 4.4.1.a: Outside of LED Pin Control Chart

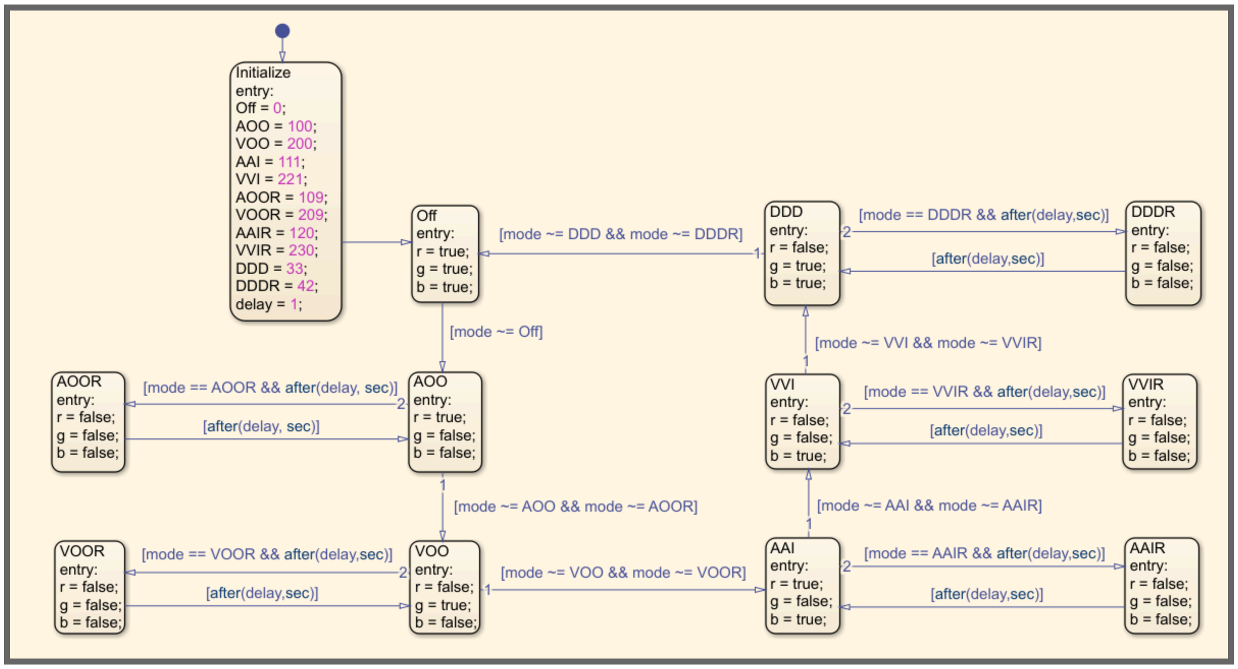


Figure 4.4.1.b: Inside the LED Pin Control Chart. The chart will cycle through each of the modes counter clockwise until it finds the correct state for the given mode.

4.4.2 Sensing Pin Control

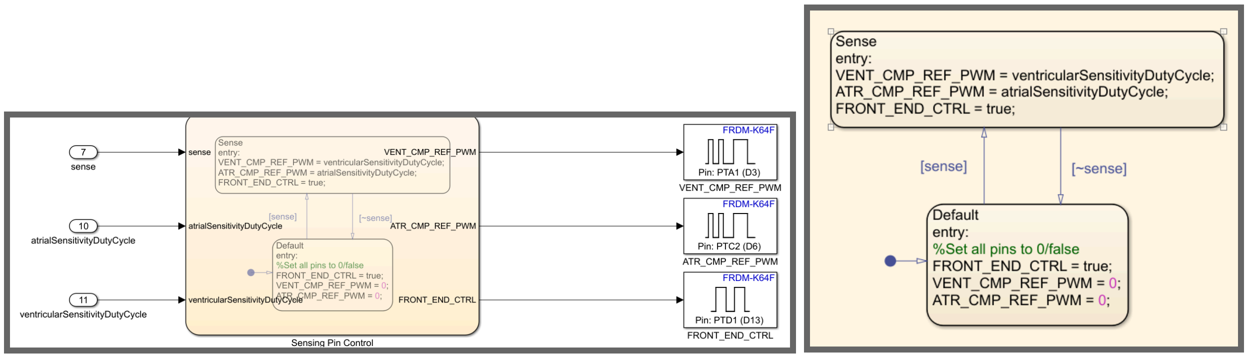
The Sensing Pin Chart is used to turn on pins required for sensing natural heartbeats. In the Sensing Pin Control chart, the inputs are the variables from the pacemaker control chart sense, and the inputs atrial and ventricular sensitivity duty cycle. The outputs from the chart are the values of the 3 pins required for sensing natural heartbeats. When the charging state is sense, the Sense state is active. When sense is not active, no sensing is done. See Table 4.4.2.a to see the pins that states turn on. Note that the FRONT_END_CTRL pin is always high as there are no risks to having it set as high but if it were to ever be low this would inhibit both digital and importantly analog sensing. If analog sensing is ever not active this causes unexpected problems with the egram data sent to and plotted by the DCM. As such it is always high in every state.

Table 4.4.2.a: Pins turned on by the charging states involved in sensing.

State	Pin Values

Sense	FRONT_END_CTRL = true
	VENT_CMP_REF_PWM = ventricularSensitivityDutyCycle
	ATR_CMP_REF_PWM = atrialSensitivityDutyCycle

Default	FRONT_END_CTRL = true
	VENT_CMP_REF_PWM = 0
	ATR_CMP_REF_PWM = 0



```
Sense
entry:
VENT_CMP_REF_PWM = ventricularSensitivityDutyCycle;
ATR_CMP_REF_PWM = atrialSensitivityDutyCycle;
FRONT_END_CTRL = true;
```

```
Default
entry:
%Set all pins to 0/false
FRONT_END_CTRL = true;
VENT_CMP_REF_PWM = 0;
ATR_CMP_REF_PWM = 0;
```

Figure 4.4.2.a: Outside (left) and inside (right) of Sensing Pin Control chart.

4.4.3 Pacing Pin Control

The Pacing Pin Control chart turns on pins required for pacing both the atrium and ventricles. The inputs to the chart are the variables from the pacemaker control chart: both preparePaceAtrium/Ventricle, both PaceAtrium/Ventricle, and reset, and both atrial/ventricularShockDutyCycle. The appropriate state will only be active when the associated charging state is on due to the pacemaker control chart deciding behaviour for the modes.

The way the chart is designed allows code reuse to be maximized, as both AOO and AAI use the states for preparing and pacing the atrium and both VOO and VVI use the states for preparing and pacing the ventricle.

Another approach is to have the charging states directly linked to the states to pace and sense this would decrease coupling but would increase code duplication drastically with VOO and VVI both having their own set of the preparePaceVentricle and paceVentricle. However, as only 1 of these states is active at a time there is no need to have multiple copies of these states. Splitting up the pin controls would also significantly decrease the cohesion of dealing with these pins as many activities involving these pins would be spread throughout the model and not concentrated in one place. See Table 4.4.3.a to see the pins that states turn on.

Table 4.4.3.a: Pins turned on by the charging states involved in pacing.

State	Pin Values

PreparePaceAtrium	VENT_PACE_CTRL = false
	ATR_PACE_CTRL = false
	PACING_REF_PWM = atrialShockDutyCycle
	PACE_CHARGE_CTRL = true
	PACE_GND_CTRL= true
	Z_VENT_CTRL = false
	Z_ATR_CTRL = false
	ATR_GND_CTRL = true
	VENT_GND_CTRL = false

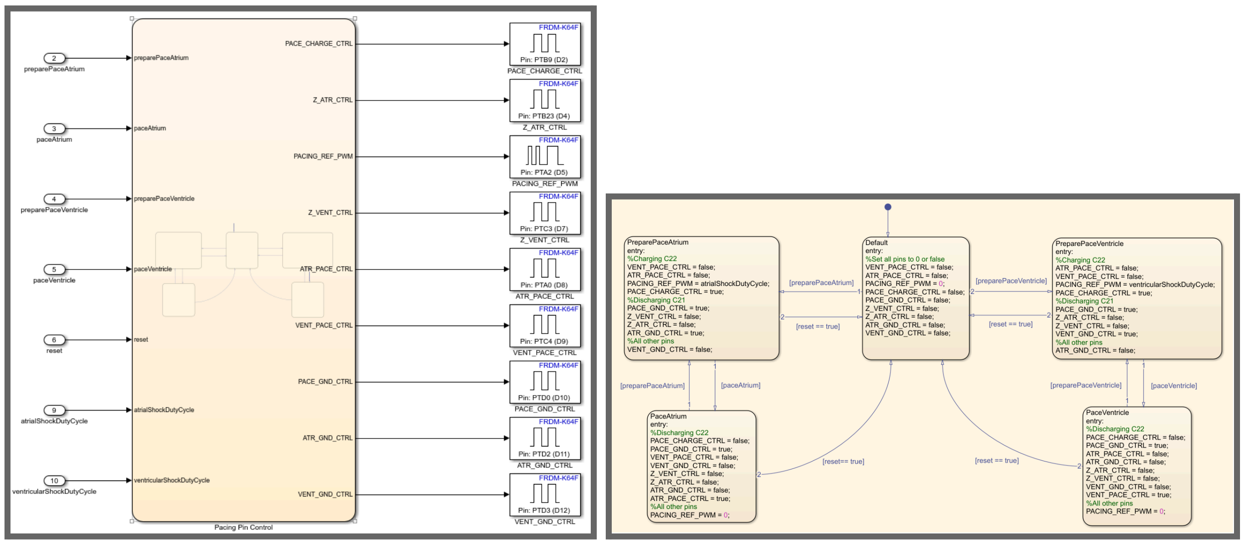
PreparePaceVentricle	ATR_PACE_CTRL = false
	VENT_PACE_CTRL = false
	PACING_REF_PWM = ventricularShockDutyCycle
	PACE_CHARGE_CTRL = true
	PACE_GND_CTRL= true
	Z_ATR_CTRL = false
	Z_VENT_CTRL = false
	VENT_GND_CTRL = true
	ATR_GND_CTRL = false

PaceAtrium	PACE_CHARGE_CTRL = false
	PACE_GND_CTRL= true
	VENT_PACE_CTRL = false
	VENT_GND_CTRL = false
	Z_VENT_CTRL = false
	Z_ATR_CTRL = false
	ATR_PACE_CTRL = true
	ATR_GND_CTRL = false
	PACING_REF_PWM = 0

PaceVentricle	PACE_CHARGE_CTRL = false
	PACE_GND_CTRL= true
	ATR_PACE_CTRL = false
	ATR_GND_CTRL = false
	Z_ATR_CTRL = false
	Z_VENT_CTRL = false
	VENT_GND_CTRL = false
	VENT_PACE_CTRL = true
	PACING_REF_PWM = 0

Default	VENT_PACE_CTRL = false
	ATR_PACE_CTRL = false
	PACING_REF_PWM = 0
	PACE_CHARGE_CTRL = false

	PACE_GND_CTRL= false
	Z_VENT_CTRL = false
	Z_ATR_CTRL = false
	ATR_GND_CTRL = false
	VENT_GND_CTRL = false


2 preparePaceAtrium
3 paceAtrium
4 preparePaceVentricle
5 paceVentricle
6 reset
9 atrialShockDutyCycle
10 ventricularShockDutyCycle
Pacing Pin Control
FRDM-K64F Pin: PTB9 (D2) PACE_CHARGE_CTRL
FRDM-K64F Pin: PTB23 (D4) Z_ATR_CTRL
FRDM-K64F Pin: PTA2 (D5) PACING_REF_PWM
FRDM-K64F Pin: PTC3 (D7) Z_VENT_CTRL
FRDM-K64F Pin: PTA0 (D8) ATR_PACE_CTRL
FRDM-K64F Pin: PTC4 (D9) VENT_PACE_CTRL
FRDM-K64F Pin: PTD0 (D10) PACE_GND_CTRL
FRDM-K64F Pin: PTD2 (D11) ATR_GND_CTRL
FRDM-K64F Pin: PTD3 (D12) VENT_GND_CTRL
[preparePaceAtrium]
[reset == true]
[paceAtrium]
[preparePaceVentricle]
[paceVentricle]
[reset== true]

```
PreparePaceAtrium
entry:
%Charging C22
VENT_PACE_CTRL = false;
ATR_PACE_CTRL = false;
PACING_REF_PWM = atrialShockDutyCycle;
PACE_CHARGE_CTRL = true;
%Discharging C21
PACE_GND_CTRL = true;
Z_VENT_CTRL = false;
Z_ATR_CTRL = false;
ATR_GND_CTRL = true;
%All other pins
VENT_GND_CTRL = false;
```

```
Default
entry:
%Set all pins to 0 or false
VENT_PACE_CTRL = false;
ATR_PACE_CTRL = false;
PACING_REF_PWM = 0;
PACE_CHARGE_CTRL = false;
PACE_GND_CTRL = false;
Z_VENT_CTRL = false;
Z_ATR_CTRL = false;
ATR_GND_CTRL = false;
VENT_GND_CTRL = false;
```

```
PreparePaceVentricle
entry:
%Charging C22
ATR_PACE_CTRL = false;
VENT_PACE_CTRL = false;
PACING_REF_PWM = ventricularShockDutyCycle;
PACE_CHARGE_CTRL = true;
%Discharging C21
PACE_GND_CTRL = true;
Z_ATR_CTRL = false;
Z_VENT_CTRL = false;
VENT_GND_CTRL = true;
%All other pins
ATR_GND_CTRL = false;
```

```
PaceAtrium
entry:
%Discharging C22
PACE_CHARGE_CTRL = false;
PACE_GND_CTRL = true;
VENT_PACE_CTRL = false;
VENT_GND_CTRL = false;
Z_VENT_CTRL = false;
Z_ATR_CTRL = false;
ATR_GND_CTRL = false;
ATR_PACE_CTRL = true;
%All other pins
PACING_REF_PWM = 0;
```

```
PaceVentricle
entry:
%Discharging C22
PACE_CHARGE_CTRL = false;
PACE_GND_CTRL = true;
ATR_PACE_CTRL = false;
ATR_GND_CTRL = false;
Z_ATR_CTRL = false;
Z_VENT_CTRL = false;
VENT_GND_CTRL = false;
VENT_PACE_CTRL = true;
%All other pins
PACING_REF_PWM = 0;
```

Figure 4.4.3.a: Inside and outside the Pacing Pin Control chart.

4.5 Serial Communication Subsystem

The serial communication subsystem is responsible for handling all serial communication with the DCM. It is set up this way separate from the rest of the model to facilitate information hiding of how the communication works from the reset of the model. It also functions to not give the DCM direct access to the operational components of the Pacemaker Firmware. The serial communication subsystem is split into three parts. The COM IN chart is responsible for handling all the serial communication message decoding logic. The send messages function handles sending specific messages to the DCM when called. It is set up as a function so that it can be run from any of the states which require a message to be sent. The other three subsystems/charts are used for handling egram data. Since the egram data needs to operate asynchronously from the rest of the communication when it is toggled, these states are responsible for doing that. These different components allow for separation of concern within the serial communication. The following sections will describe the operation of each of these subsystems and charts.

Full description here

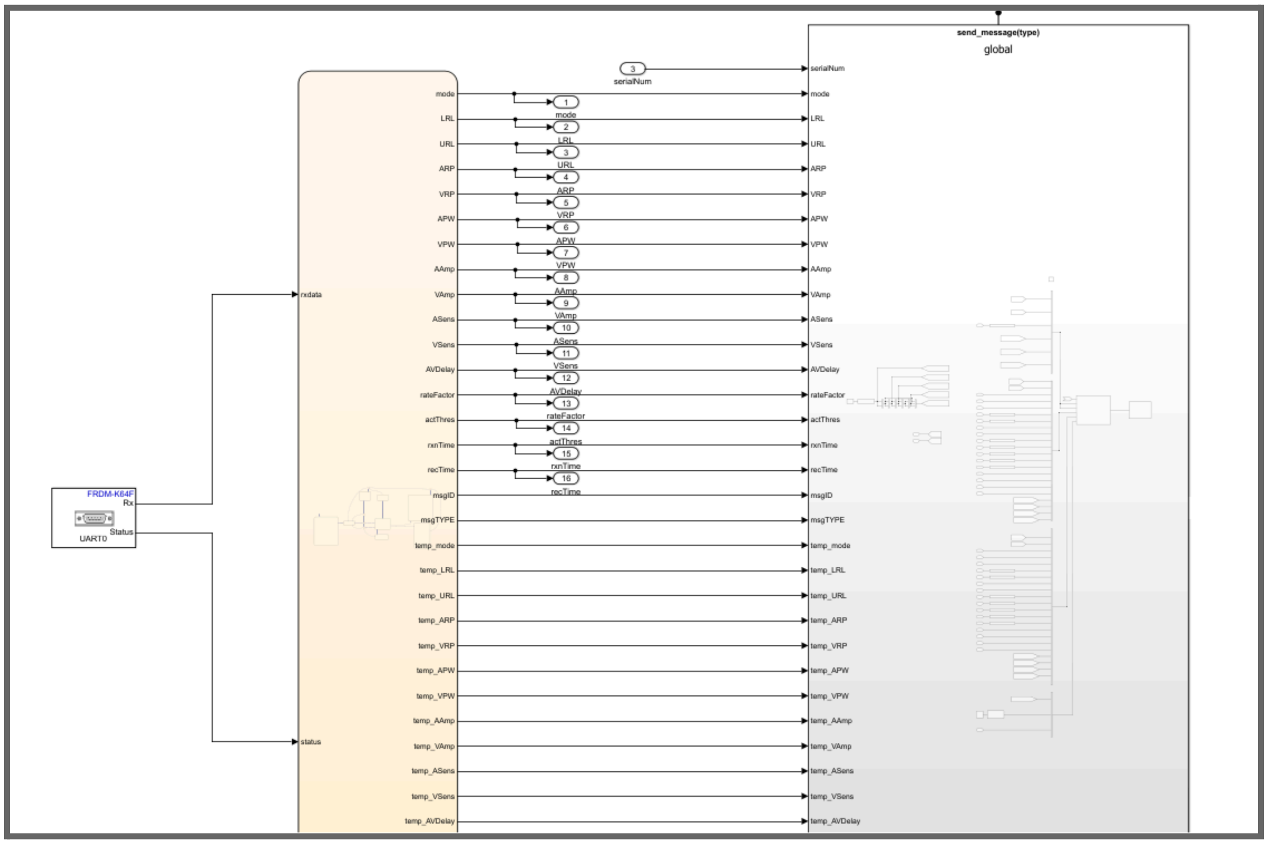


Figure 4.5.a: Top section of the serial communication subsystem, was too large for one screenshot.

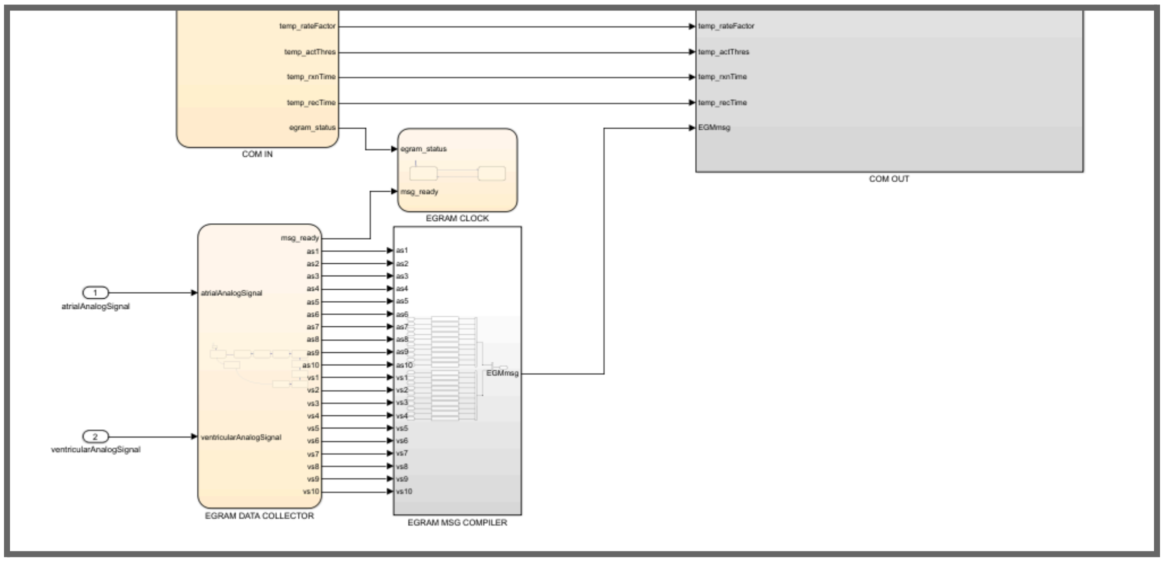
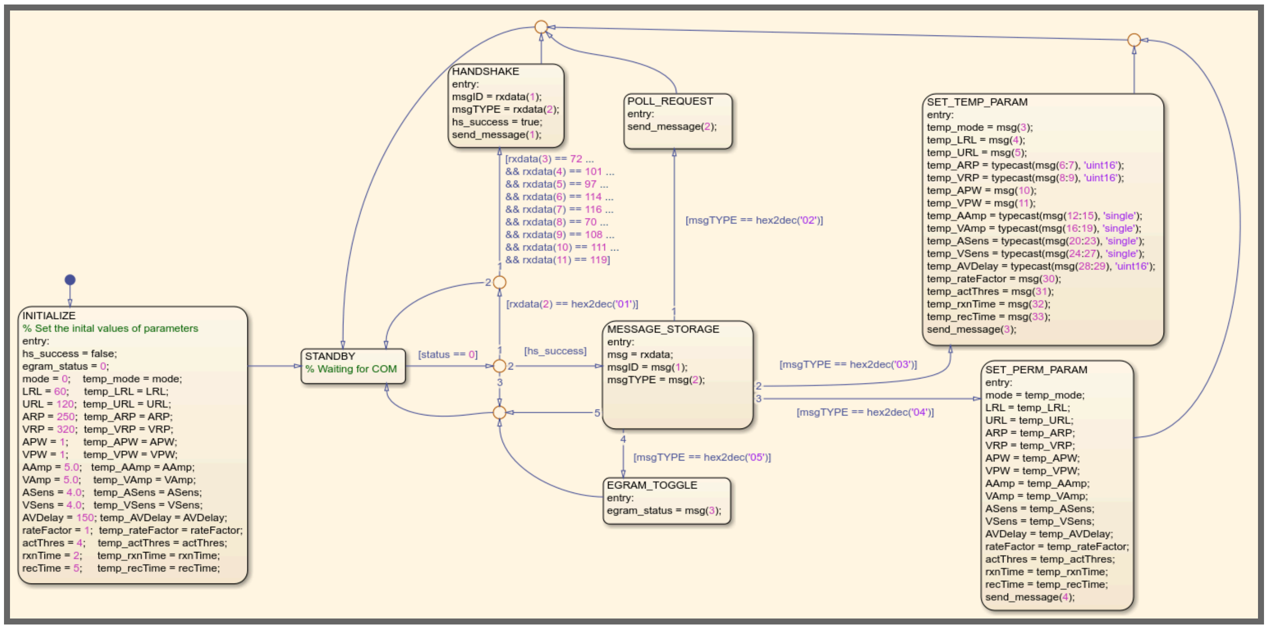


Figure 4.5.b: Bottom section of the serial communication subsystem, was too large for one screenshot.

4.5.1 Serial Communication Logic Chart

The serial communication logic chart, labelled as COM IN in the Simulink model, is responsible for handling all incoming messages from the DCM. The initial state sets each of the parameters to their initial values. Note that the mode is initially set to Off so although these parameters are set to be the nominal safe operating conditions, since the Off mode is the default safe state they should not be required. It is still valuable to have them set though in case something goes wrong and the mode randomly switches then at least the model should be operating at its safe nominal values. Following the initialization the default state which the model will always return to is the standby state. This is where the chart is waiting to receive any serial communication. When serial communication is received, the model will move forward to the circle to the right. From here it can only move to the rest of the message types if the handshake has been successfully completed. As per the requirements the handshake must be completed before any other message can be interpreted. If the message type provided at this point isn't the handshake then it will return to standby. But if it is the handshake message type then it will move up. Here the validation bytes are checked for the word HeartFlow. If the validation bytes are not correct, the model will return to standby, but if the validation bytes are correct then the handshake is considered correct and a response is provided before returning to the standby state. The handshake state will set the variable of message ID and type so they can be echoed back in the response. The handshake will be marked as correct, and the reply will be sent.

Now assuming the handshake is done and a different message type is sent it will move to the message storage state. This state is important as it stores the message variables so that if a different message is sent during the following couple milliseconds it will not mess up the communication and can potentially cause an error. After the message is saved, this saved message will be referred to for the rest of the chart until it returns back to stand by. If the message type is for a polling request the model will go up and send back a polling reply. If the message type is for egram toggle then the model will go down and start or stop the cycle for sending egram data. If the message is for requesting a parameter change then all of the parameters sent will be stored as temporary variables and sent back to the DCM to ensure they were correct and a mistake was not made during transmission or interpretation of the message. Finally, if the message type is confirming the parameters then all of the parameters will be set to the values of the temporary parameters and a message will be sent back to the DCM telling it what values it implemented for the parameters. That covers each state and their function. The logic is set up this way to follow the serial communication protocol described in a separate document as per the requirements.



```
INITIALIZE
% Set the inital values of parameters
entry:
hs_success = false;
egram_status = 0;
mode = 0;    temp_mode = mode;
LRL = 60;     temp_LRL = LRL;
URL = 120;  temp_URL = URL;
ARP = 250;  temp_ARP = ARP;
VRP = 320;  temp_VRP = VRP;
APW = 1;     temp_APW = APW;
VPW = 1;     temp_VPW = VPW;
AAmp = 5.0;   temp_AAmp = AAmp;
VAmp = 5.0;    temp_VAmp = VAmp;
ASens = 4.0;   temp_ASens = ASens;
VSens = 4.0;   temp_VSens = VSens;
AVDelay = 150; temp_AVDelay = AVDelay;
rateFactor = 1;  temp_rateFactor = rateFactor;
actThres = 4;    temp_actThres = actThres;
rxnTime = 2;     temp_rxnTime = rxnTime;
recTime = 5;     temp_recTime = recTime;
```

```
STANDBY
% Waiting for COM
```

```
HANDSHAKE
entry:
msgID = rxdata(1);
msgTYPE = rxdata(2);
hs_success = true;
send_message(1);
```

```
POLL_REQUEST
entry:
send_message(2);
```

```
MESSAGE_STORAGE
entry:
msg = rxdata;
msgID = msg(1);
msgTYPE = msg(2);
```

```
EGRAM_TOGGLE
entry:
egram_status = msg(3);
```

```
SET_TEMP_PARAM
entry:
temp_mode = msg(3);
temp_LRL = msg(4);
temp_URL = msg(5);
temp_ARP = typecast(msg(6:7), 'uint16');
temp_VRP = typecast(msg(8:9), 'uint16');
temp_APW = msg(10);
temp_VPW = msg(11);
temp_AAmp = typecast(msg(12:15), 'single');
temp_VAmp = typecast(msg(16:19), 'single');
temp_ASens = typecast(msg(20:23), 'single');
temp_VSens = typecast(msg(24:27), 'single');
temp_AVDelay = typecast(msg(28:29), 'uint16');
temp_rateFactor = msg(30);
temp_actThres = msg(31);
temp_rxnTime = msg(32);
temp_recTime = msg(33);
send_message(3);
```

```
SET_PERM_PARAM
entry:
mode = temp_mode;
LRL = temp_LRL;
URL = temp_URL;
ARP = temp_ARP;
VRP = temp_VRP;
APW = temp_APW;
VPW = temp_VPW;
AAmp = temp_AAmp;
VAmp = temp_VAmp;
ASens = temp_ASens;
VSens = temp_VSens;
AVDelay = temp_AVDelay;
rateFactor = temp_rateFactor;
actThres = temp_actThres;
rxnTime = temp_rxnTime;
recTime = temp_recTime;
send_message(4);
```

```
[status == 0]
[hs_success]
[rxdata(2) == hex2dec('01')]
[rxdata(3) == 72 ...
&& rxdata(4) == 101 ...
&& rxdata(5) == 97 ...
&& rxdata(6) == 114 ...
&& rxdata(7) == 116 ...
&& rxdata(8) == 70 ...
&& rxdata(9) == 108 ...
&& rxdata(10) == 111 ...
&& rxdata(11) == 119]
[msgTYPE == hex2dec('02')]
[msgTYPE == hex2dec('03')]
[msgTYPE == hex2dec('04')]
[msgTYPE == hex2dec('05')]
```

Figure 4.5.1.a: Inside the serial communication logic chart.

4.5.2 Send Message Function

The send message function is used to send a serial message to the DCM from anywhere within the serial communication subsystem. The function takes as input the message type that it should respond with, which are based on the message types described by the serial protocol documentation. On the far left is a set up which creates empty arrays of bytes of different sizes which then go into go to blocks. This is useful as most of the message types are not completely full of data but the message still has to be 82 bytes long. Using go to blocks cleans up the wiring of this chart significantly, increasing clarity. The four mux blocks are used to follow each of the five separate message types, going from highest to lowest handshake and polling request (because they have the same reply), message interpretation reply, message implementation reply, and egram data. The order of bytes in each of these messages is based on the order required by the serial communication protocol created. Note that the justification for using the given data types for each particular variable is provided in the serial communication protocol documentation as well. Finally, each one of these messages enters a switch which based on the function input, which is the message type that needs to be sent, will choose which one of the messages to send and write out to the DCM. All the messages need to be included in this one function because the serial output block can only be in the model once. Additionally, the serial output block must be within a function so it is only written to when the function is called and not constantly being written to.

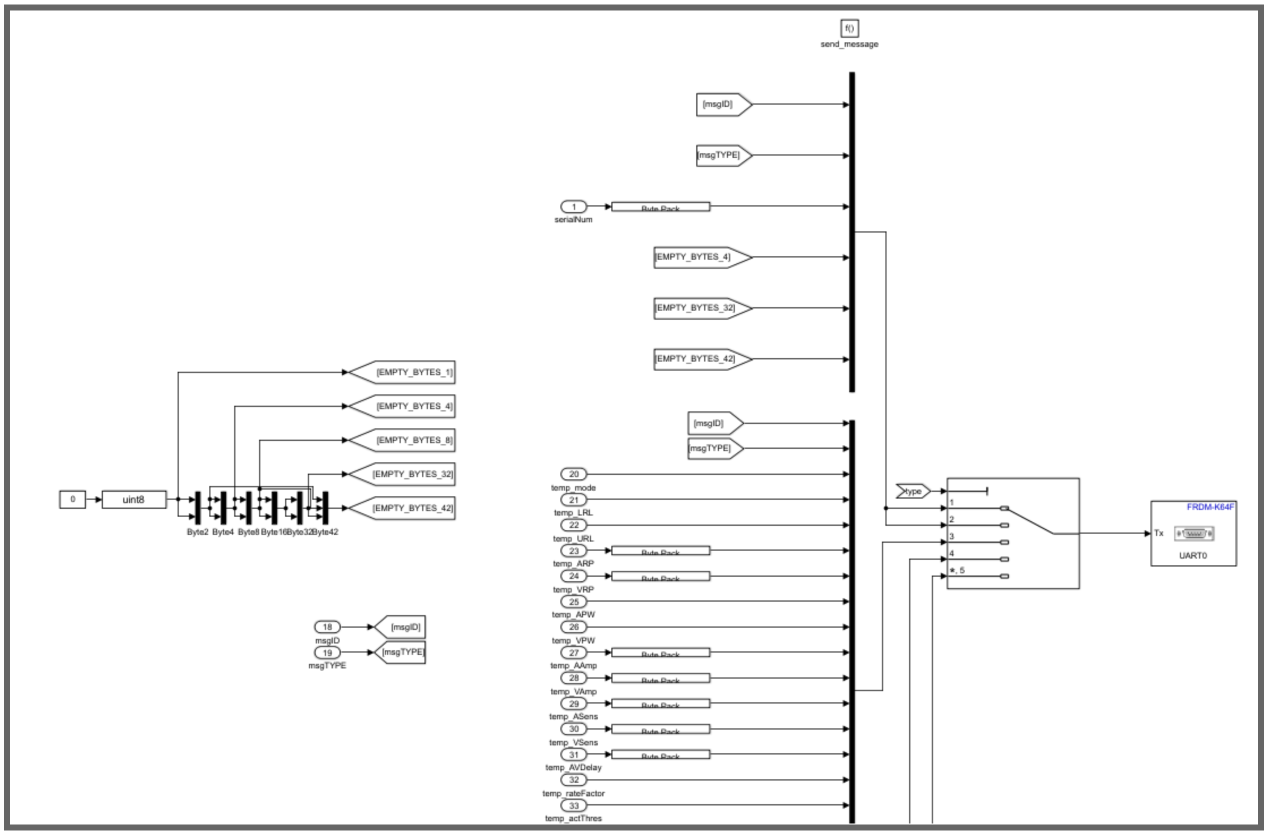


Figure 4.5.2.a: Inside the sense message function. Top sections, too large for one screenshot.

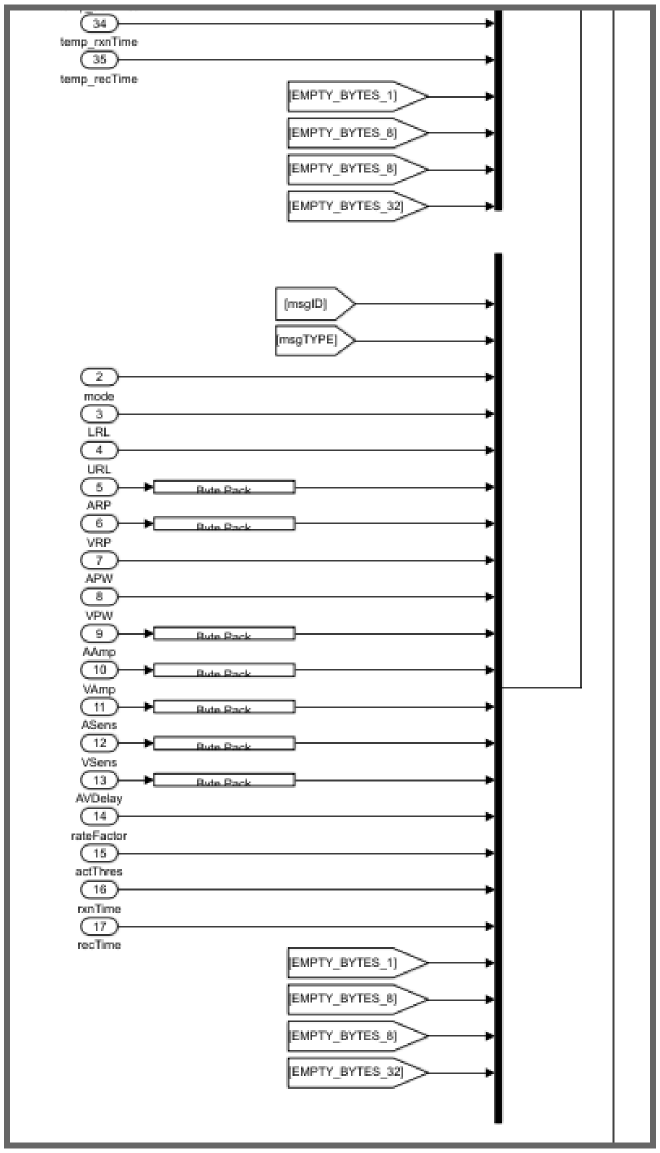


Figure 4.5.2.b: Inside the sense message function. Middle sections, too large for one screenshot.

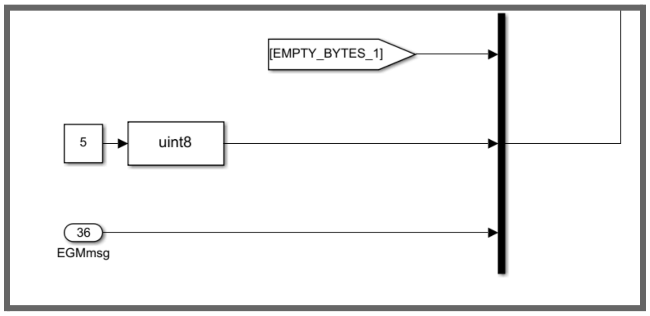


Figure 4.5.2.c: Inside the sense message function. bottom sections, too large for one screenshot.

4.5.3 Electrogram Data Collector

The electrogram data collector chart is responsible for collecting and storing the different analog heart signals detected at each time step. The reason that multiple time steps must be collected and sent in one message rather then just sent one time step each time is because doing so would cause one of two problem if egram messages are being sent every millisecond or two which would congest the serial communication or messages need to be sent less frequently which would limit the resolution of the electrogram plot and may lead to missing high frequency components of the signal introducing aliasing. To avoid this problem but still keep total message size small, ten time step readings are sent each message. This time step samples are taken at a sampling rate of 2 ms, set by the variable in the chart called wait. This was done because it allowed for high resolution in the electrogram plot but made the plot loading in the DCM much faster as it only had to load half the data points of a certain time compared to having a sample rate of 1 ms. The chart works by cycling through each of the ten time steps, recording the value of the analog sensing pins to a variable as it enters each state. The final state of the cycle will set a variable called msg_ready to true which will cause the message to be sent by a different subsystem if the egram is currently toggled on. The reason this chart is always running and does not have a stop condition that would cause it only to run when egram is enabled and the reason that it does not call the send message function directly is because we got weird circular dependency errors from Simulink when we tried to do either of those things. This was the solution which allowed us to implement the serial communication protocol we came up with in Simulink, we tried many different more elegant methods beforehand but this was the only one we could get to not throw an error when complying to the board.

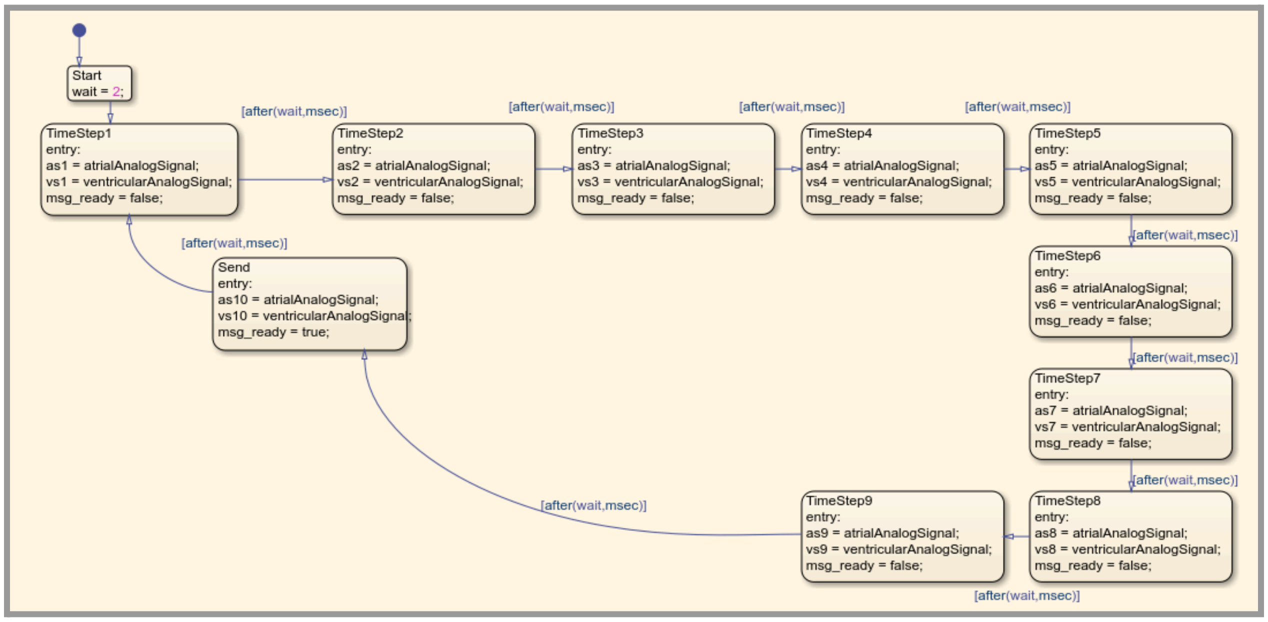

Start
wait = 2;
TimeStep1 entry: as1 = atrialAnalogSignal; vs1 = ventricularAnalogSignal; msg_ready = false;
TimeStep2 entry: as2 = atrialAnalogSignal; vs2 = ventricularAnalogSignal; msg_ready = false;
TimeStep3 entry: as3 = atrialAnalogSignal; vs3 = ventricularAnalogSignal; msg_ready = false;
TimeStep4 entry: as4 = atrialAnalogSignal; vs4 = ventricularAnalogSignal; msg_ready = false;
TimeStep5 entry: as5 = atrialAnalogSignal; vs5 = ventricularAnalogSignal; msg_ready = false;
TimeStep6 entry: as6 = atrialAnalogSignal; vs6 = ventricularAnalogSignal; msg_ready = false;
TimeStep7 entry: as7 = atrialAnalogSignal; vs7 = ventricularAnalogSignal; msg_ready = false;
TimeStep8 entry: as8 = atrialAnalogSignal; vs8 = ventricularAnalogSignal; msg_ready = false;
TimeStep9 entry: as9 = atrialAnalogSignal; vs9 = ventricularAnalogSignal; msg_ready = false;
Send entry: as10 = atrialAnalogSignal; vs10 = ventricularAnalogSignal; msg_ready = true;
[after(wait,msec)]

Figure 4.5.3.a: Inside the electrogram data collector chart.

4.5.4 Electrogram Message Compiler

The electrogram message compiler is a very simple subsystem. All it does is take all of the analog time step outputs for both the ventricle and atrium from the Electrogram Data Collector chart, pack each one into binary bytes, and combine them all together into one variable. This is simply done to limit the number of inputs to the send message function so that it does not have an additional 20 inputs which could simply be one input like this. The order of bytes is determined by the serial communication protocol created and documented separately.

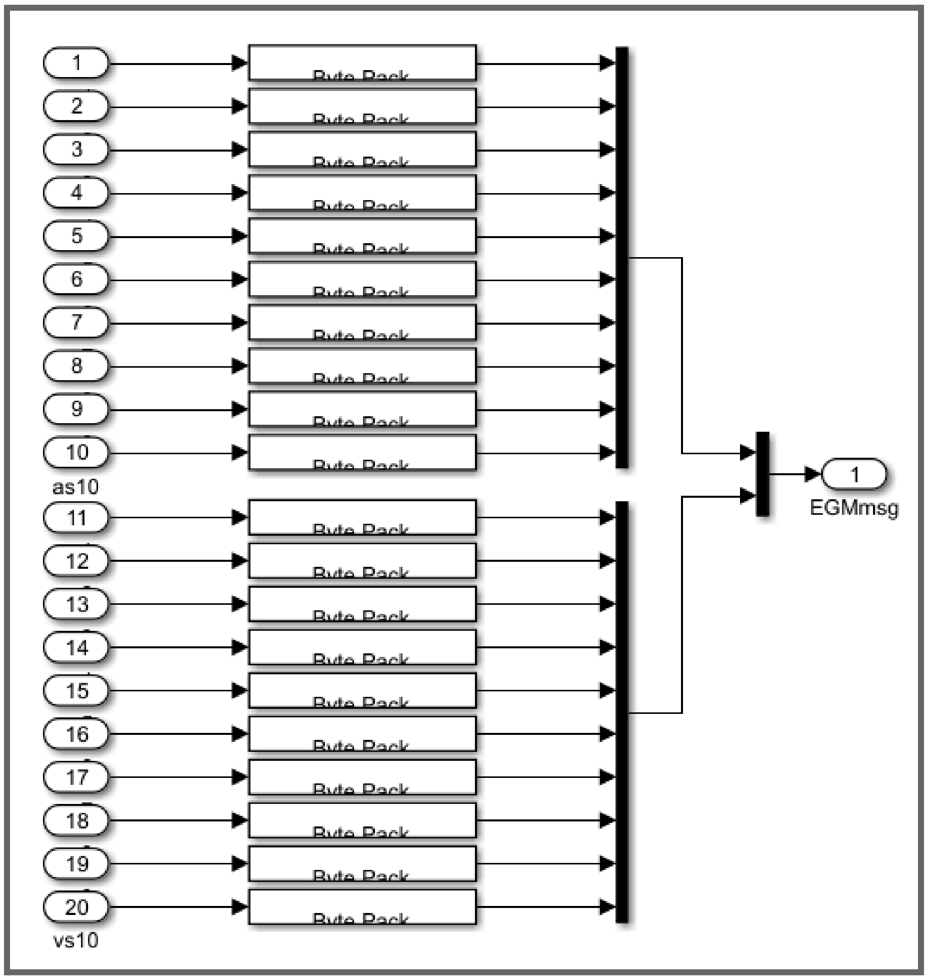


Figure 4.5.4.a: Inside the electrogram message compiler subsystem.

4.5.5 Electrogram Clock

The final state involved in the electrogram component of the serial communication subsystem is the electrogram clock. As spoken about in section 4.5.3, other methods of implementing the egram part of the serial protocol caused us to have many circular dependency errors. But doing it this way seems to resolve those issues. Essentially how it works is that when egram data is toggled on by the serial communication (the toggle byte in the last egram toggle message is not equal to zero) and the message is ready to send (determined by the Electrogram Data Collector chart) then it will switch to the send egram state. In this state it calls the send message function with a message type of egram data. Then it immediately returns to waiting. This means that a message will not be sent when the egram data collected is not complete and will not be sent if the egram data has been toggled off by the DCM.

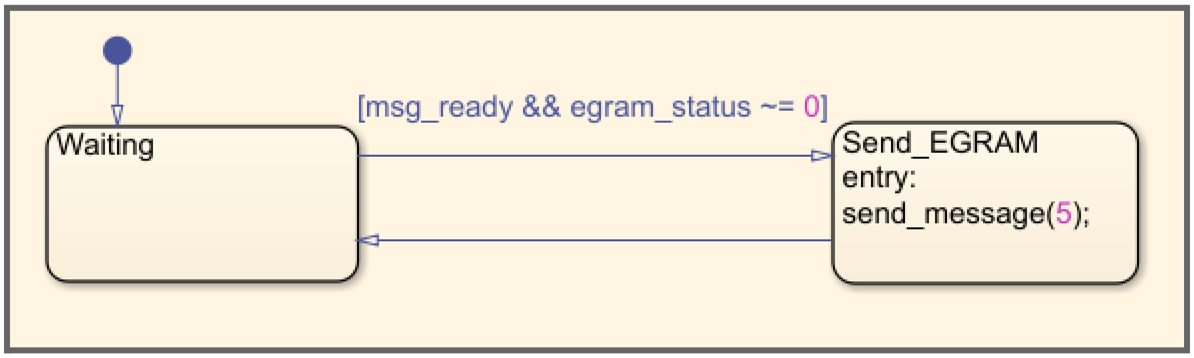


Figure 4.5.5.a: Inside the electrogram clock chart.

4.6 Completed and Expected Changes

4.6.1 Past Changes

Input Subsystem Changes:

- In the input subsystem all values which used to be programmable parameters became values which are sent to the DCM through serial connection. This was a known change, and was easy to adapt to.
- Mode selection was also changed to be sent from the DCM. It used to be tied to a button, and with more mores being added new values were needed.
- New parameter inputs were required to implement the rate adaptive modes

Pacemaker Chart Changes:

- When more modes were added more subchart needed to be added to the pacemaker chart, however no changes were needed to be made to the existing modes.
- New parameter inputs were required to implement the rate adaptive modes

Output Subsystem Changes:

- The sensing chart was redesigned, originally it was believed that the sensing chart required 3 states, with 1 sensing atrium, 1 sensing ventricle, and 1 not sensing. It was later learned that only 1 state was needed to do sensing, and both atrial and ventricular sensing could occur at the same time.
- With new modes added to the pacemaker, the LED indicator was updated with blinking lights for the rate adaptive modes.
- To remove magic numbers, variable names were tied to numbers instead of using the numbers in the Simulink.

Other Changes:

- Rate adaptive modes were added, and with it a new subsystem inside of the input subsystem.

- Serial communication was added, allowing the DCM and the simulink to send values between each other.

4.6.2 Future Changes

In the future, more changes are expected to be made the expected changes are listed below:

- In the serial subsystem the egram data is expected to send the data that the pacemaker is pacing, currently, the pacemaker only sends the values that are being read off of the on board heart, ideally, the egram data being sent through serial would also include the pacemaker paces.
- In a final product version of a pacemaker, tachycardia would need to be managed, as of right now the pacemaker is only capable of managing bradycardia, but to be able to tachycardia would require a large amount of work.

5 Validation and Verification

The following outlines the justification of Pacemaker Firmware requirements and all testing of the Pacemaker Firmware.

5.1 Validation

The requirements that are created for the Pacemaker Firmware are based on the PACEMAKER requirement documentation given to the project team. Based on sections 3.4 which specifies programmable parameters of pulse amplitude and pulse width, 3.5 lists required modes, and 5 has a table of required parameters for the modes. Section 3.6.5 gives values for the refractory period used in the requirements. Table A lists value ranges for lower rate limits, refractory periods, pulse amplitudes, and pulse widths. However, these values are not specific to the hardware that we are using, causing the values to possibly be slightly different for the requirements than the values that are in the document. The provided Assignment 2 documentation addresses this concern by providing new range restrictions specific to the hardware platform used. These modified parameters were the sensitivities, the amplitudes, and the pulse widths. Related requirements for these parameters have been sufficiently updated.

Additional requirements for the implementation of rate adaptive pacing were obtained from the provided Tutorial4 and Rate_Adaptive_Pacing documents. These documents outlined each of the required programmable parameters, their role within rate adaptive pacing, and the overall concept of rate adaptive pacing. A notable deviation from the provided requirements for rate adaptive pacing, was that the recovery and reaction times were significantly reduced from the provided expected ranges. This was done to facilitate improved testing and demo conditions. The Tutorial3 document provided and internal team discussions generated the requirements for the serial communication protocol. Finally, the specifics used to implement the DDD and DDDR modes were gathered from the previously mentioned PACEMAKER document and the Pacing_Modes document provided.

Most requirements were generated from these formal specifications making the needed modifications when necessitated by the specifics of this assignment. Considering the modifications required to complete the assignment, comparing the requirements with the formal specifications it can be said that the requirements are valid for creating the Pacemaker Firmware and what was created is in relative compliance with the formal specifications.

5.2 Verification

The following provides verification of each of the requirements outlined separately in this document. These tests ensure that the Pacemaker Firmware operates as intended and meets all requirements. Requirements are referred to by the numbering present in section 3 Requirements. Unless otherwise mentioned each one of these tests uses a blackbox testing methodology, in which the inputs are set and the outputs are observed. This approach was taken as it allows for testing to take place using exclusively the final version of the Pacemaker Firmware avoiding the risk that an unknown change was made between unit testing and the final implementation. Using the final system as a whole for testing in this manner allows for increased confidence that the system is safe because of the global property of safety. Just because the AOO and VOO modes are safe when tested in separate implementations does not mean they will integrate well and still remain safe. The same can be said regarding the DCM and Pacemaker Firmware. For these reasons all testing is performed using a blackbox testing approach and conducted using by passing the programmable parameter inputs from the DCM, unless otherwise mentioned.

5.2.1 Implicit Testing

Requirements 1-3 do not require formal testing as the design directly implements these requirements. As previously discussed, the eleven modes Off, AOO, VOO, AAI, VVI, DDD, AOOR, VOOR, AAIR, VVIR, & DDDR have been added and can be changed through the interface with the DCM with the LED colour indicating the current mode. Specific testing for each mode is below. Additionally, it can be inferred that the microcontroller representing the heart is being shocked using electrical pulses given that the paces are generated by charging and discharging a capacitor. The remaining requirements are all far more specific and testable and as such formal tests and their results follow.

5.2.2 Programmable Parameter Testing

The programmable parameters outlined by requirement 4 are tested below. These tests were done using the various pacemaker modes with different input parameters. Results were measured using the HeartView monitoring tool. Though many of these tests are unsuccessful, it is believed that the programmable parameters are implemented correctly and hardware errors and limitations are causing many of the discrepancies. Note that for any of the parameter testing which tests if the range correction is working properly, the default parameters when the pacemaker is powered on had to be modified because range limits within the DCM would not allow for out of range values to be entered. Any tests using values within the acceptable range are tested using the sending parameters functionality of the DCM.

5.2.2.1 Refractory Periods

The following test cases described in Table 5.2.2.1.a are intended to test the refractory period programmable parameters. These tests are conducted using the HeartView monitoring tool and the AAI, VVI, AAIR, and VVIR modes. The other Pacemaker Firmware modes do not need to be included in the test as the refractory period does not affect these modes. Interesting test results are further discussed in the table.

Table 5.2.2.1.a: Refractory period test cases verifying requirement 4.1.

Test	Expected Result	Pass/Fail	Comment
No natural atrial pulsing when ARP is set to 500 ms and the lower rate limit is set higher than 120 BPM	Atrial artificial pacing at approximately 120 BPM	Pass	Indicates that pacing is inhibited following the refractory period.
No natural atrial pulsing when ARP is set to 1000 ms and the lower rate limit is set higher than 120 BPM	Atrial artificial pacing at approximately 120 BPM	Pass	Indicates that the upper range limit is functioning and limiting ARP to 500 ms.
No natural ventricular pulsing when VRP is set to 500 ms and the lower rate limit is set higher than 120 BPM	Ventricular artificial pacing at approximately 120 BPM	Pass	Indicates that pacing is inhibited following the refractory period.
No natural ventricular pulsing when VRP is set to 1000 ms and the lower rate limit is set higher than 120 BPM	Ventricular artificial pacing at approximately 120 BPM	Pass	Indicates that the upper range limit is functioning and limiting VRP to 500 ms.
Natural atrial pulsing is higher than 120 BPM when ARP is set to 500 ms and the lower rate limit is set higher than the natural heart rate	No pacing output	Pass	Indicates that the refractory period is not inhibiting natural pulse sensing.

Natural ventricular pulsing is higher than 120 BPM when VRP is set to 500 ms and the lower rate limit is set higher than the natural heart rate	No pacing output	Pass	Indicates that the refractory period is not inhibiting natural pulse sensing.

As indicated by all of these tests the refractory periods are functioning properly. The upper range limit for the refractory periods is verified as an input value of 500 ms and 1000 ms renders the same resulting maximum refractory period of 500 ms. The lower range limit is not testable as the heart rate cannot reach a high enough value in which the period between pulses is less than 150 ms and the minimum refractory period would make a difference. When the refractory period is greater than the lower rate limit, it limits pacing to the refractory period as intended. Finally, during our project demonstration, it was instructed that the Pacemaker Firmware should continue to sense natural pacing during the refractory period which is verified as being true. As all tests verify the expected behavior, the refractory period programmable parameters are considered to be correct.

5.2.2.2 Sensitivities

Although sensitivity is offered as a programmable parameter, it is highly suggested to always set the value to 4V when testing is performed using HeartView. This value was determined by informal testing and has been dialed into this specific value to work most effectively to ensure it can detect the natural pulses without detecting noise. Further testing to determine if the sensitivity is working properly is difficult as the HeartView monitoring tool does not provide the ability to vary the natural pulse amplitude and as such various test cases to see if the sensitivity is changed can not be thoroughly completed. As such no further testing for the sensitivities is provided.

5.2.2.3 Pulse Widths

The following test cases outlined in Table 5.2.2.3.a are intended to test the pulse width programmable parameters. These tests are conducted using the HeartView monitoring tool and the AOO and VOO modes. These modes are used because they allow for testing of both the atrial and ventricular pulse widths and utilize identical pacing codes to all the other modes. Pulse width is measured as the time between the rising edge of the pulse and the initial falling edge as it crosses 0V. Interesting test results are further discussed in the table.

Table 5.2.2.3.a: Pulse width test cases verifying requirement 4.2.

Test	Expected Result	Pass/Fail	Comment
The pulse width is set below 1 ms	The pulse width is the same as setting it to 1 ms	Pass	Tested with both AOO and VOO. Indicates that the lower range limit is functioning and limiting the pulse width to 1 ms.

The pulse width is set above 30 ms	The pulse width is the same as setting it to 30 ms	Pass	Tested with both AOO and VOO. Indicates that the upper range limit is functioning and limiting the pulse width to 30 ms.
The atrial pulse width is set between 1 ms and 30 ms	The atrial pulse width measured in HeartView is approximately its set value	Fail	Tested at 1, 5, 10, 15, 20, 25, and 30ms. All measured to either approximately 1.0 or 1.5 ms.
The ventricular pulse width is set between 1 ms and 30 ms	The ventricular pulse width measured in HeartView is approximately its set value	Fail	Tested at 1, 5, 10, 15, 20, 25, and 30ms. All measured to either approximately 1.0 or 1.5 ms.

This set of tests yielded interesting and unexpected results. The first two tests passed because they yielded the same measured values as the parameters they were expected to be rounded to. However, the other two tests both failed as all the parameters seemed to produce a measured value of either 1.0 or 1.5 ms, whichever was closer. This is likely due to hardware errors and limitations regarding the time required to discharge the capacitor. Assuming this is what is causing the error, changing the parameter does slightly modify the pulse width meaning that the parameter is doing its function and it is likely implemented correctly.

5.2.2.4 Pulse Amplitudes

Formal testing of the pulse amplitudes is difficult due to large amplitude inconsistencies between pulses. This is considered to be a limitation of either the hardware or HeartView monitoring tool and as such will eliminate the possibility for formula testing. It can be shown that an amplitude of 4V is relatively larger than an amplitude of 1V. However, there is no way to repetitively and empirically test this. Considering this issue is a result of hardware error and changing the pulse amplitude does cause a visible relative change within HeartView, the pulse amplitude programmable parameters are considered correct.

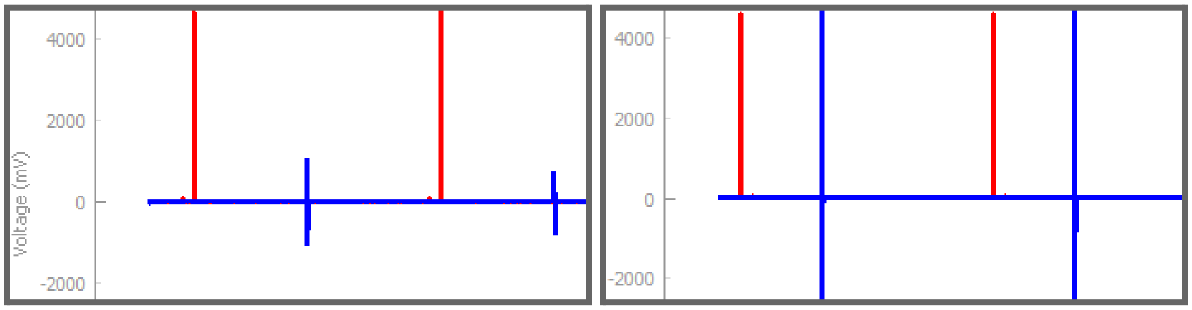


Figure 5.2.2.4.a: Comparison of the 1V (left) and 4V (right) pulse amplitude in HeartView.

5.2.2.5 Lower Rate Limit

The following test cases outlined in Table 5.2.2.5.a are intended to test the lower rate limit programmable parameter. These tests are conducted using the HeartView monitoring tool and the AOO and VOO modes. These modes are used because they are not affected by the natural pulsing of the heart and the natural and artificial beats can be set to the same frequency to observe any changes in phase shift over time. Interesting test results are further discussed following the table.

Table 5.2.2.5.a: Lower rate limit test cases verifying requirement 4.4.

Test	Expected Result	Pass/Fail	Comment
Natural pulsing is set to 30 BPM and the lower rate limit is set below 30 BPM.	Natural and artificial pulsing remain at the same phase shift over time.	Pass	Tested with both AOO and VOO. Indicates that the lower range limit is functioning and limiting the lower rate limit to 30 BPM.
Natural pulsing is set to 175 BPM and the lower rate limit is set above 175 BPM.	Natural and artificial pulsing remain at the same phase shift over time.	Pass	Tested with both AOO and VOO. Indicates that the upper range limit is functioning and limiting the lower rate limit to 175 BPM.
Natural atrial pulsing is set between 30 and 175 BPM and the lower rate limit is set to the same value.	Natural and artificial atrial pulsing remain at the same phase shift over time.	Pass	Tested at 30, 31, 47, 50, 60, 70, 90, 120, 150, 170, and 175 BPM. Indicates that natural and artificial atrial pulsing are consistently at the same rate.
Natural ventricular pulsing is set between 30 and 175 BPM and the lower rate limit is set to the same value.	Natural and artificial ventricular pulsing remain at the same phase shift over time.	Pass	Tested at 30, 31, 47, 50, 60, 70, 90, 120, 150, 170, and 175 BPM. Indicates that natural and artificial ventricular pulsing are consistently at the same rate.

All of the tests yield the expected results as anticipated because of the correction factor implemented. Older versions of the Pacemaker Firmware would not have achieved these results which is why the correction factor was implemented. Further details regarding this are provided in 4.2.1.5 Lower Rate Limit. It is important to note that given enough time there will be a change in phase shift as the two rates are not identical, however, over a reasonable amount of time they appear to be the same. Due to the successful testing, the lower rate limit is considered correct.

5.2.2.6 Upper Rate Limit

The following test cases outlined in Table 5.2.2.6.a are intended to test the upper rate limit programmable parameter. These tests are conducted using the HeartView monitoring tool and the AOOR and VOOR modes. These modes are used because they are not affected by the natural pulsing of the heart and the natural and artificial beats can be set to the same frequency to observe any changes in phase shift over time. To test the upper rate limit, the rate factor is set to 16 and the activity threshold is set very low so that shaking the pacemaker will cause the rate adaptive pacing limit to reach the upper rate limit and verify results. Interesting test results are further discussed following the table.

Table 5.2.2.6.a: Upper rate limit test cases verifying requirement 4.5.

Test	Expected Result	Pass/Fail	Comment
Natural pulsing is set to 50 BPM and the upper rate limit is set below 50 BPM. Pacemaker is shaken to maximize rate adaptive pacing limit.	Natural and artificial pulsing remain at the same phase shift over time.	Pass	Tested with both AOO and VOO. Indicates that the lower range limit is functioning and limiting the lower rate limit to 30 BPM.
Natural pulsing is set to 175 BPM and the upper rate limit is set above 175 BPM. Pacemaker is shaken to maximize rate adaptive pacing limit.	Natural and artificial pulsing remain at the same phase shift over time.	Pass	Tested with both AOO and VOO. Indicates that the upper range limit is functioning and limiting the lower rate limit to 175 BPM.
Natural atrial pulsing is set between 50 and 175 BPM and the upper rate limit is set to the same value. Pacemaker is shaken to maximize rate adaptive pacing limit.	Natural and artificial atrial pulsing remain at the same phase shift over time.	Pass	Tested at 50, 60, 70, 90, 120, 150, 170, and 175 BPM. Indicates that natural and artificial atrial pulsing are consistently at the same rate.
Natural ventricular pulsing is set between 50 and 175 BPM and the upper rate limit is set to the same value. Pacemaker is shaken to maximize rate adaptive pacing limit.	Natural and artificial ventricular pulsing remain at the same phase shift over time.	Pass	Tested at 50, 60, 70, 90, 120, 150, 170, and 175 BPM. Indicates that natural and artificial ventricular pulsing are consistently at the same rate.

All of the tests yield the expected results as anticipated because of the correction factor implemented. Older versions of the Pacemaker Firmware would not have achieved these results which is why the correction factor was implemented. Further details regarding this are provided in 4.2.1.5 Lower Rate Limit (same correction factor applied as LRL). It is important to note that given enough time there will be a change in phase shift as the two rates are not identical, however, over a reasonable amount of time they appear to be the same. Due to the successful testing, the upper rate limit is considered correct.

5.2.2.7 Atrioventricular Delay

The following test cases outlined in Table 5.2.2.7.a are intended to test the AV delay programmable parameter. These tests are conducted using the HeartView monitoring tool and the DDD mode. This mode is used because it is the only mode which uses the AV delay and is not affected by rate adaptive pacing to ensure each test case remains consistent. To test the AV delay all parameters are set to reasonable values, the natural atrium and ventricle are set to off, and the AV delay is varied. The AV delay is measured by subtracting the timestamp of the rising edge or the ventricular pace from the rising edge of the atrial pulse. Interesting test results are further discussed following the table.

Table 5.2.2.7.a: AV delay test cases verifying requirement 4.7.

Test	Expected Result	Pass/Fail	Comment
AV delay is set below 30 ms	The AV delay is the same as setting it to 30 ms	Pass	Tested with DDD with the natural heart disabled. Indicates that the lower range limit is functioning and limiting the AV delay to 30 ms.
AV delay is set above 300 ms	The AV delay is the same as setting it to 300 ms	Pass	Tested with DDD with the natural heart disabled. Indicates that the upper range limit is functioning and limiting the AV delay to 300 ms.
AV delay is set between 30 ms and 300 ms	The AV delay measured in HeartView is approximately its set value	Pass	Tested at 30, 100, 200, and 300ms.

All tests passed with the expected results. As the values are bounded within the pacemaker firmware, all of these values were testable and yielded the results that were expected. The AV delay value that was on the board is seen in HeartView and is within milliseconds of the expected value. AV delay is considered to be correctly implemented.

5.2.2.8 Rate Factor

The following test cases outlined in Table 5.2.2.8.a are intended to test the rate factor programmable parameter. These tests are conducted using the HeartView monitoring tool and any rate adaptive mode. These modes are used as they require the rate factor.

Table 5.2.2.8: Rate factor test cases verifying requirement 4.8

Test	Expected Result	Pass/Fail	Comments
Rate factor is set below 1	Rate factor value is set to 1	Pass	Tested by setting value to -1 and shaking pacemaker the heart rate required large amount of shaking to reach URL
Rate Factor is set above 16	Rate factor is set to 16	Pass	Tested by setting value to 20 and shaking pacemaker the heart rate required a small amount of shaking to reach URL
Rate Factor is set between 1 and 16	Rate factor is unchanged	Pass	Tested by setting value to all values between 1 and 16 and shaking pacemaker the heart rate different amount of shaking to reach URL

All tests pass with the expected results, as the values are bounded within the pacemaker firmware. All of these values were testable and yielded the results that were expected, with higher rate factors, having less shaking to be done to achieve the highest possible heart rate. The rate factor is considered to be correctly implemented.

5.2.2.9 Activity Threshold

The following test cases outlined in Table 5.2.2.9.a are intended to test the activity threshold programmable parameter. These tests are conducted using the HeartView monitoring tool and any rate adaptive mode. These modes are used as they require the activity threshold.

Table 5.2.2.9.a: Activity threshold test cases verifying requirement 4.9.

Test	Expected Result	Pass/Fail	Comment
Activity threshold is set below 1	Activity threshold is set to 1	Pass	Tested by setting value to -1, and setting rate factor to 16, when shaking a small amount of shaking was required to overcome activity threshold.
Activity threshold is set above 7	Activity threshold is set to 7	Pass	Tested by setting value to 20, and setting rate factor to 16 when shaking a large amount of shaking was required to overcome activity threshold.

Activity threshold is set between 1 and 7	Activity threshold is unchanged	Pass	Tested by setting value to all values between 1 and 7, and setting rate factor to 16 when shaking a variable amount of shaking was required to overcome activity threshold.

All tests pass with the expected results, as the values are bounded within the pacemaker firmware. All of these values were testable and yielded the results that were expected, with higher lower activity thresholds, having less shaking to be done to achieve a change in heart rate. Activity threshold is considered to be correctly implemented.

5.2.2.10 Reaction Time

The following test cases are outlined in Table 5.2.2.10.a are intended to test the reaction time programmable parameter. These tests are conducted using the HeartView monitoring tool and any rate adaptive mode. These modes are used because they require reaction time.

Table 5.2.2.10.a: Reaction time test cases verifying requirement 4.10.

Test	Expected Result	Pass/Fail	Comment
Reaction time is set to be less than 1	Reaction time is set to be 1	Pass	Tested by shaking with low activity value and rate factor is 16, when shaken the heart rate reaches the MSR quickly.
Reaction time is set to be more than 50	Reaction time is set to be 50	Pass	Tested by shaking with low activity value and rate factor is 16, when shaken the heart rate reaches the MSR quickly.
Reaction time is set to be between 1 and 50	Reaction time is unchanged	Pass	Tested by shaking with low activity value and rate factor is 16 at several different reaction times, when shaken the time reaches the MSR depending on the input.

All tests passed with the expected results, as the values are bounded within the pacemaker firmware. All of these values were testable and yielded the results that were expected, with higher reaction times, having more time to be able to reach the MSR. Reaction time is considered to be correctly implemented.

5.2.2.11 Recovery Time

The following test cases are outlined in Table 5.2.2.11.a are intended to test the recovery time programmable parameter. These tests are conducted using the HeartView monitoring tool and any rate adaptive mode. These modes are used as they require recovery time.

Table 5.2.2.11.a: Recovery time test cases verifying requirement 4.11.

Test	Expected Result	Pass/Fail	Comment
Recovery time is set to be less than 1	Recovery time is set to be 1	Pass	Tested by shaking with a low activity value and rate factor is 16, when shaking stops, the heart rate reaches the LRL quickly.
Recovery time is set to be less than 240	Recovery time is set to be 240	Pass	Tested by shaking with low activity value and rate factor is 16, when shaking stops, the heart rate reaches the LRL Slowly.
Recovery time is set to be between 1 and 240	Recovery time is unchanged	Pass	Tested by shaking with low activity value and rate factor is 16 at several different reaction times, when shaken the time reaches the LRL depending on the input.

All tests pass with the expected results, as the values are bounded within the pacemaker firmware. All of these values were testable and yielded the results that were expected, with higher recovery times, having more time to be able to achieve the lower rate limit. Recovery time is considered to be correctly implemented.

5.2.3 Testing Pacemaker Modes

Each of the eleven pacemaker modes is tested in the following section. This testing is previously mentioned and will be done using a blackbox testing approach. The inputs will be assigned using the DCM and the outputs will be measured using the HeartView monitoring tool. Each test set is based on covering each branch of the implemented mode and covering each outcome specified by that mode's requirements.

5.2.3.1 Off Mode Testing

The following test cases described in Table 5.2.3.1.a are intended to test the Pacemaker Firmware Off mode and requirement 5. These tests are to be conducted using the HeartView monitoring tool. References to low and high pulse rates refer to the tested pulse rate being lower or higher than the lower rate limit. Interesting test results are further discussed following the table.

Table 5.2.3.1.a: Off mode test cases verifying requirement 5.

Test	Expected Result	Pass/Fail	Comment
No natural atrial pulsing	No pacing output	Pass	No comment

Low natural atrial pulsing	No pacing output	Pass	No comment
High natural atrial pulsing	No pacing output	Pass	No comment
No natural ventricular pulsing	No pacing output	Pass	No comment
Low natural ventricular pulsing	No pacing output	Pass	No comment
High natural ventricular pulsing	No pacing output	Pass	No comment

As expected Off mode never paces the heart despite any possible heart activity. All tests returned the expected results and Off mode is considered correct.

5.2.3.2 AOO Mode Testing

The following test cases described in Table 5.2.3.2.a are intended to test the Pacemaker Firmware AOO mode and requirement 6. These tests are to be conducted using the HeartView monitoring tool. References to low and high pulse rates refer to the tested pulse rate being lower or higher than the lower rate limit. Interesting test results are further discussed following the table.

Table 5.2.3.2.a: AOO mode test cases verifying requirement 6.

Test	Expected Result	Pass/Fail	Comment
No natural atrial pulsing	Atrial pacing at the lower rate limit	Pass	No comment
Low natural atrial pulsing	Atrial pacing at the lower rate limit	Pass	No comment
High natural atrial pulsing	Atrial pacing at the lower rate limit	Pass	No comment
No natural ventricular pulsing	Atrial pacing at the lower rate limit	Pass	No comment
Low natural ventricular pulsing	Atrial pacing at the lower rate limit	Pass	No comment
High natural ventricular pulsing	Atrial pacing at the lower rate limit	Pass	No comment

As expected AOO mode always paces the atrium at the lower rate limit ignoring heart activity. All tests returned the expected results and AOO mode is considered correct.

5.2.3.3 VOO Mode Testing

The following test cases described in Table 5.2.3.3.a are intended to test the Pacemaker Firmware VOO mode and requirement 7. These tests are to be conducted using the HeartView monitoring tool. References to low and high pulse rates refer to the tested pulse rate being lower or higher than the lower rate limit. Interesting test results are further discussed following the table.

Table 5.2.3.3.a: VOO mode test cases verifying requirement 7.

Test	Expected Result	Pass/Fail	Comment
No natural atrial pulsing	Ventricular pacing at the lower rate limit	Pass	No comment
Low natural atrial pulsing	Ventricular pacing at the lower rate limit	Pass	No comment
High natural atrial pulsing	Ventricular pacing at the lower rate limit	Pass	No comment
No natural ventricular pulsing	Ventricular pacing at the lower rate limit	Pass	No comment
Low natural ventricular pulsing	Ventricular pacing at the lower rate limit	Pass	No comment
High natural ventricular pulsing	Ventricular pacing at the lower rate limit	Pass	No comment

As expected VOO mode always paces the ventricle at the lower rate limit ignoring heart activity. All tests returned the expected results and VOO mode is considered correct.

5.2.3.4 AAI Mode Testing

The following test cases described in Table 5.2.3.4.a are intended to test the Pacemaker Firmware AAI mode and requirement 8. These tests are to be conducted using the HeartView monitoring tool. References to low and high pulse rates refer to the tested pulse rate being lower or higher than the lower rate limit. Interesting test results are further discussed following the table.

Table 5.2.3.4.a: AAI mode test cases verifying requirement 8.

Test	Expected Result	Pass/Fail	Comment
No natural atrial pulsing	Atrial pacing at the lower rate limit	Pass	Tested at a lower rate limit of 31, 60, 120, 175 BPM.
Low natural atrial pulsing	Atrial pacing lower rate limit time after the previous pulse either natural or artificial	Pass	Tested at a lower rate limit of 31, 60, 120, 175 BPM.
Natural atrial pulsing one BPM below the lower rate limit	One artificial pace immediately before each natural pulse	Pass	Tested at a lower rate limit of 31, 60, 120, 175 BPM. At 120 and 175 BPM the artificial pace comes ~6 and ~10 ms after the natural pulse but still one artificial pace per natural pulse.
Natural atrial pulsing at the lower rate limit	No pacing output	Fail	Tested at a lower rate limit of 31, 60, 120, 175 BPM. Only at 30 BPM was every artificial pace consistently inhibited.
Natural atrial pulsing one BPM above the lower rate limit	No pacing output	Pass	Tested at a lower rate limit of 31, 60, 120, 175 BPM.
High natural atrial pulsing	No pacing output	Pass	Tested at a lower rate limit of 31, 60, 120, 175 BPM.
No natural ventricular pulsing	Atrial pacing at the lower rate limit	Pass	Tested at a lower rate limit of 31, 60, 120, 175 BPM.
Low natural ventricular pulsing	Atrial pacing at the lower rate limit	Pass	Tested at a lower rate limit of 31, 60, 120, 175 BPM.
High natural ventricular pulsing	Atrial pacing at the lower rate limit	Pass	Tested at a lower rate limit of 31, 60, 120, 175 BPM.

For the most part, the AAI mode passes each test at the lower rate limits tested. The significant failure of this set of tests is that the Pacemaker Firmware generates inconsistent pacing when the atrial heart rate in HeartView is set to the lower rate limit. The only tested heart rate that passes this specific test is a lower rate limit of 30 BPM. The project team suspects this is the case because at greater heart rates minor errors become more significant due to the shorter period between pulses. As a result, very minor differences in the generated natural heart rate due to hardware errors associated with the heart microcontroller may cause the Pacemaker Firmware to sometimes pick up the natural pulse and other times not. More regarding why this design decision was made is spoken about in 4.3.3 AAI Mode. However, as this inconsistent pacing is happening effectively simultaneously with the natural heartbeats, this should not cause an issue and is therefore not considered to be a complete failure of the mode. Additionally, it is interesting that at higher rates the artificial pace comes slightly after the natural pulse when testing at one BPM less than the lower rate limit. This is believed to be the case because of the incredibly short time between the Pacemaker Firmware finishing waiting for a natural pulse to begin generating an artificial pulse and the heart generating a natural pulse. During this small time period after the Pacemaker Firmware has finished waiting for a natural pulse and begun to send the artificial pulse, the natural pulse occurs. As the heart rate increases this time period becomes even shorter and this is suspected to be the reason that 175 BPM has a greater delay than 120 BPM. As the artificial and natural paces effectively happen at the same time this delay is not considered harmful. Increasing the HeartView pulse width parameter during these tests causes all of the minor delays discussed to happen during this longer natural pulse. Finally, as intended the ventricular heart activity does not affect the AAI mode. Considering all that is discussed here the AAI mode is considered to be correct for this assignment.

5.2.3.5 VVI Mode Testing

The following test cases described in Table 5.2.3.5.a are intended to test the Pacemaker Firmware VVI mode and requirement 9. These tests are to be conducted using the HeartView monitoring tool. References to low and high pulse rates refer to the tested pulse rate being lower or higher than the lower rate limit. Interesting test results are further discussed following the table.

Table 5.2.3.5.a: VVI mode test cases verifying requirement 9.

Test	Expected Result	Pass/Fail	Comment
No natural ventricular pulsing	Ventricular pacing at the lower rate limit	Pass	Tested at a lower rate limit of 31, 60, 120, 175 BPM.
Low natural ventricular pulsing	Ventricular pacing lower rate limit time after the previous pulse either natural or artificial	Pass	Tested at a lower rate limit of 31, 60, 120, 175 BPM.
Natural ventricular pulsing one BPM below the lower rate limit	One artificial pace immediately before each natural pulse	Pass	Tested at a lower rate limit of 31, 60, 120, 175 BPM. At 60, 120, and 175 BPM the artificial pace comes ~2, ~16, and ~22 ms after the natural pulse but still one artificial pace per natural pulse.
Natural ventricular pulsing at the lower rate limit	No pacing output	Fail	Tested at a lower rate limit of 31, 60, 120, 175 BPM. Only at 30 BPM was every artificial pace consistently inhibited.
Natural ventricular	No pacing output	Pass	Tested at a lower rate limit of 31,

pulsing one BPM above the lower rate limit			60, 120, 175 BPM.
High natural ventricular pulsing	No pacing output	Pass	Tested at a lower rate limit of 31, 60, 120, 175 BPM.
No natural atrial pulsing	Ventricular pacing at the lower rate limit	Pass	Tested at a lower rate limit of 31, 60, 120, 175 BPM.
Low natural atrial pulsing	Ventricular pacing at the lower rate limit	Pass	Tested at a lower rate limit of 31, 60, 120, 175 BPM.
High natural atrial pulsing	Ventricular pacing at the lower rate limit	Pass	Tested at a lower rate limit of 31, 60, 120, 175 BPM.

Testing the VVI mode yields nearly identical results to testing the AAI mode, as such the explanation of interesting results can be found in 5.2.3.4 AAI Mode Testing. The only significant difference between the VVI mode testing is that at one below the lower rate limit of 60 BPM the artificial pace is delayed compared to the natural pulse. However, for the same reasons as the AAI mode, this is not considered to be a test failure. As the AAI mode is considered correct for this assignment, and the VVI mode renders similar testing results then the VVI mode is also considered correct for this assignment.

5.2.3.6 DDD Mode Testing

The following test cases described in Table 5.2.3.6.a are intended to test the Pacemaker Firmware DDD mode and requirement 10. These tests are to be conducted using the HeartView monitoring tool. References to low and high pulse rates refer to the tested pulse rate being lower or higher than the lower rate limit. Interesting test results are further discussed following the table.

Table 5.2.3.6.a: DDD mode test cases verifying requirement 10.
**Note: The actual AV delay is considered to be the time between a sensed or delivered atrial pulse and the next ventricular pulse. This definition means that if the atria are not beating then the AV delay set in HeartView may not be equivalent to the actual AV delay, however, if the atria are beating then the AV delay set in HeartView is equivalent to the actual AV delay.*

Test	Expected Result	Pass/Fail	Comment
Atria not beating AND Ventricles not beating	Atrial pacing at LRL and ventricular pacing programmed AV delay after atrial pace.	Pass	Tested at a lower rate limit of 31, 60, 120, 175 BPM and AV Delay of 150 ms.
Atria not beating AND Actual AV Delay > Programmed AV Delay	Atrial pacing at LRL and ventricular pacing programmed AV delay after atrial pace.	Pass	Tested at a lower rate limit of 31, 60, 120, 175 BPM and AV Delay of 150 ms.
Atria not beating AND Actual AV Delay =	Atrial pacing at LRL and no ventricular pacing.	Pass	Tested at a lower rate limit of 31, 60, 120, 175 BPM and AV Delay of 150 ms.

Programmed AV Delay			
Atria not beating AND Actual AV Delay < Programmed AV Delay	Atrial pacing at LRL and no ventricular pacing.	Pass	Tested at a lower rate limit of 31, 60, 120, 175 BPM and AV Delay of 150 ms.
Heart Rate < Lower Rate Limit AND Ventricles not beating	Atrial pacing at LRL after most recently sensed or delivered atrial pulse and ventricular pacing programmed AV delay after sensed or delivered atrial pace.	Pass	Tested at a lower rate limit of 31, 60, 120, 175 BPM and AV Delay of 150 ms.
Heart Rate < Lower Rate Limit AND Actual AV Delay > Programmed AV Delay	Atrial pacing at LRL after most recently sensed or delivered atrial pulse and ventricular pacing programmed AV delay after sensed or delivered atrial pace.	Pass	Tested at a lower rate limit of 31, 60, 120, 175 BPM and AV Delay of 150 ms.
Heart Rate < Lower Rate Limit AND Actual AV Delay = Programmed AV Delay	Atrial pacing at LRL after most recently sensed or delivered atrial pulse and ventricular pacing only following a delivered atrial pace.	Pass	Tested at a lower rate limit of 31, 60, 120, 175 BPM and AV Delay of 150 ms.
Heart Rate < Lower Rate Limit AND Actual AV Delay < Programmed AV Delay	Atrial pacing at LRL after most recently sensed or delivered atrial pulse and ventricular pacing only following a delivered atrial pace.	Pass	Tested at a lower rate limit of 31, 60, 120, 175 BPM and AV Delay of 150 ms.
Heart Rate = Lower Rate Limit AND Ventricles not beating	No atrial pacing and ventricular pacing programmed AV delay after sensed atrial pace.	Pass	Tested at a lower rate limit of 31, 60, 120, 175 BPM and AV Delay of 150 ms.
Heart Rate > Lower Rate Limit AND Ventricles not beating	No atrial pacing and ventricular pacing programmed AV delay after sensed atrial pace.	Pass	Tested at a lower rate limit of 31, 60, 120, 175 BPM and AV Delay of 150 ms.
Heart Rate > Lower Rate Limit AND Actual AV Delay > Programmed AV Delay	No atrial pacing and ventricular pacing programmed AV delay after sensed atrial pace.	Pass	Tested at a lower rate limit of 31, 60, 120, 175 BPM and AV Delay of 150 ms.
Heart Rate > Lower Rate Limit AND Actual AV Delay = Programmed AV Delay	No atrial pacing and no ventricular pacing.	Pass	Tested at a lower rate limit of 31, 60, 120, 175 BPM and AV Delay of 150 ms.
Heart Rate > Lower Rate Limit AND Actual AV Delay < Programmed AV Delay	No atrial pacing and no ventricular pacing.	Pass	Tested at a lower rate limit of 31, 60, 120, 175 BPM and AV Delay of 150 ms.

Testing DDD mode gives the expected results for all tests. Some tests were difficult to conduct, as the requirements state that the ventricular pulse should only be sensed within the AV delay amount of time. This is difficult to do as 150ms, the value the AV delay is set to is very small and making a natural ventricular pulse within that range in HeartView is very difficult. However, in HeartView it can be seen that the mode can catch the ventricular pulse on occasion.

5.2.3.7 AOOR Mode Testing

The following test cases described in Table 5.2.3.7.a are intended to test the Pacemaker Firmware AOOR mode and requirement 12. These tests are to be conducted using the HeartView monitoring tool. References to low and high pulse rates refer to the tested pulse rate being lower or higher than the lower rate limit. Interesting test results are further discussed following the table.

Table 5.2.3.7.a: AOOR mode test cases verifying requirement 12.

Test	Expected Result	Pass/Fail	Comment
No natural atrial pulsing and no shaking	Atrial pacing at the lower rate limit	Pass	No comment
No natural atrial pulsing and light shaking	Atrial pacing at the current HR	Pass	No comment
No natural atrial pulsing and heavy shaking	Atrial pacing at the upper rate limit	Pass	No comment
Low natural atrial pulsing and no shaking	Atrial pacing at the lower	Pass	No comment
Low natural atrial pulsing and light shaking	Atrial pacing at the current HR	Pass	No comment
Low natural atrial pulsing and heavy shaking	Atrial pacing at upper rate limit	Pass	No comment
High natural atrial pulsing and no shaking	Atrial pacing at the lower rate limit	Pass	No comment
High natural atrial pulsing and light shaking	Atrial pacing at the current HR	Pass	No comment
High natural atrial pulsing and heavy shaking	Atrial pacing at the upper rate limit	Pass	No comment
No natural ventricular pulsing and no shaking	Atrial pacing at the lower rate limit	Pass	No comment
No natural ventricular pulsing and light shaking	Atrial pacing at the current HR	Pass	No comment
No natural ventricular pulsing and heavy shaking	Atrial pacing at the upper rate limit	Pass	No comment
Low natural ventricular pulsing and no shaking	Atrial pacing at the lower rate limit	Pass	No comment

Low natural ventricular pulsing and light shaking	Atrial pacing at the current HR	Pass	No comment
Low natural ventricular pulsing and heavy shaking	Atrial pacing at the upper rate limit	Pass	No comment
High natural ventricular pulsing and no shaking	Atrial pacing at the lower rate limit	Pass	No comment
High natural ventricular pulsing and light shaking	Atrial pacing at the current HR	Pass	No comment
High natural ventricular pulsing and heavy shaking	Atrial pacing at the upper rate limit	Pass	No comment

As expected AOOR mode always paces the atrium at the calculated heart rate between the lower and upper rate limit ignoring heart activity. All tests returned the expected results and AOOR mode is considered correct.

5.2.3.8 VOOR Mode Testing

The following test cases described in Table 5.2.3.8.a are intended to test the Pacemaker Firmware VOOR mode and requirement 13. These tests are to be conducted using the HeartView monitoring tool. References to low and high pulse rates refer to the tested pulse rate being lower or higher than the lower rate limit. Interesting test results are further discussed following the table.

Table 5.2.3.8.a: VOOR mode test cases verifying requirement 13.

Test	Expected Result	Pass/Fail	Comment
No natural atrial pulsing and no shaking	Ventricular pacing at the lower rate limit	Pass	No comment
No natural atrial pulsing and light shaking	Ventricular pacing at the current HR	Pass	No comment
No natural atrial pulsing and heavy shaking	Ventricular pacing at the upper rate limit	Pass	No comment
Low natural atrial pulsing and no shaking	Ventricular pacing at the lower	Pass	No comment
Low natural atrial pulsing and light shaking	Ventricular pacing at the current HR	Pass	No comment
Low natural atrial pulsing and heavy shaking	Ventricular pacing at upper rate limit	Pass	No comment
High natural atrial pulsing and no shaking	Ventricular pacing at the lower rate limit	Pass	No comment
High natural atrial pulsing and light shaking	Ventricular pacing at the current HR	Pass	No comment
High natural atrial pulsing and heavy shaking	Ventricular pacing at the upper rate limit	Pass	No comment
No natural ventricular pulsing	Ventricular pacing at the	Pass	No comment

and no shaking	lower rate limit		
No natural ventricular pulsing and light shaking	Ventricular pacing at the current HR	Pass	No comment
No natural ventricular pulsing and heavy shaking	Ventricular pacing at the upper rate limit	Pass	No comment
Low natural ventricular pulsing and no shaking	Ventricular pacing at the lower rate limit	Pass	No comment
Low natural ventricular pulsing and light shaking	Ventricular pacing at the current HR	Pass	No comment
Low natural ventricular pulsing and heavy shaking	Ventricular pacing at the upper rate limit	Pass	No comment
High natural ventricular pulsing and no shaking	Ventricular pacing at the lower rate limit	Pass	No comment
High natural ventricular pulsing and light shaking	Ventricular pacing at the current HR	Pass	No comment
High natural ventricular pulsing and heavy shaking	Ventricular pacing at the upper rate limit	Pass	No comment

As expected AOOR mode always paces the atrium at the calculated heart rate between the lower and upper rate limit ignoring heart activity. All tests returned the expected results and AOOR mode is considered correct.

5.2.3.9 AAIR Mode Testing

As AAIR works the same as the AAI mode, the tests will mainly focus on the rate adaptive pacing. The following test cases described in Table 5.2.3.9.a are intended to test the Pacemaker Firmware AAIR mode and requirement 14. These tests are to be conducted using the HeartView monitoring tool.

Table 5.2.3.9.a: AAIR mode test verifying requirements 14

Test	Expected Result	Pass/Fail	Comment
Natural Atrium = LRL, no shaking	No atrial pacing should be seen	Pass	Test was done with LRL of 60 and URL of 175
Natural Atrium = LRL, Shaking	No atrial pacing should be seen while shaking	Pass	Test was done with LRL of 60 and URL of 175
Natural Atrium = URL, no Shaking	No atrial pacing should be seen	Pass	Test was done with LRL of 60 and URL of 175
Natural Atrium = URL, Shaking	No atrial pacing should be seen	Pass	Test was done with LRL of 60 and URL of 175

As expected AAIR mode paces when the HR is greater than the natural atrium rhythm, and does not pulse when the natural atrium rhythm is greater than the HR. This indicates that both the rate-adaptive pacing and the AAIR mode work as intended.

5.2.3.10 VVIR Mode Testing

As VVIR works the same as the VVI mode, the tests will mainly focus on the rate adaptive pacing. The following test cases described in Table 5.2.3.9.a are intended to test the Pacemaker Firmware VVIR mode and requirement 15. These tests are to be conducted using the HeartView monitoring tool.

Table 5.2.3.10.a: VVIR mode test verifying requirements 15

Test	Expected Result	Pass/Fail	Comment
Natural Atrium = LRL, no shaking	No ventricular pacing should be seen	Pass	Test was done with LRL of 60 and URL of 175
Natural Atrium = LRL, Shaking	ventricular pacing should be seen while shaking	Pass	Test was done with LRL of 60 and URL of 175
Natural Atrium = URL, no Shaking	No ventricular pacing should be seen	Pass	Test was done with LRL of 60 and URL of 175
Natural Atrium = URL, Shaking	No ventricular pacing should be seen	Pass	Test was done with LRL of 60 and URL of 175

As expected VVIR mode paces when the HR is greater than the natural ventricular rhythm and does not pulse when the natural ventricular rhythm is greater than the HR. This indicates that both the rate-adaptive pacing and the VVIR mode work as intended.

5.2.3.11 DDDR Mode Testing

As DDDR works exactly the same as the DDD mode, the tests will mainly focus on the rate adaptive pacing. The following test cases are described in Table 5.2.3.10.a are intended to test the Pacemaker Firmware DDDR mode and requirement 16. These tests are to be conducted using the HeartView monitoring tool.

Table 5.2.3.10.a: DDDR mode test verifying requirements 16

Test	Expected Result	Pass/Fail	Comment
Natural Atrium = LRL, AV delay > programmed AVDelay, no shaking	No atrial pacing should be seen, no ventricular pacing should be seen	Pass	Test was done with LRL of 60, URL of 175, and AV delay of 150ms
Natural Atrium = LRL, AV delay > programmed AVDelay, shaking	Atrial pacing should be seen, no ventricular pacing should be seen	Pass	Test was done with LRL of 60, URL of 175, and AV delay of 150ms
Natural Atrium = LRL, AV delay > programmed AVDelay, no shaking	No Atrial pacing, ventricular pacing occurs after ventricular natural pulse.	Pass	Test was done with LRL of 60, URL of 175, and AV delay of 150ms
Natural Atrium = LRL, AV delay > programmed AVDelay, shaking	Atrial pacing should be seen, ventricular pacing occurs after ventricular natural pulse.	Pass	Test was done with LRL of 60, URL of 175, and AV delay of 150ms
Natural Atrium = URL, AV delay < programmed AVDelay, no shaking	No atrial pacing should be seen, no ventricular pacing should be seen	Pass	Test was done with LRL of 60, URL of 175, and AV delay of 150ms

Natural Atrium = URL, AV delay < programmed AVDelay, shaking	No atrial pacing should be seen, no ventricular pacing should be seen	Pass	Test was done with LRL of 60, URL of 175, and AV delay of 150ms
Natural Atrium = URL, AV delay > programmed AVDelay, no shaking	No atrial pulses should be seen. Ventricular pacing occurs after ventricular natural pulse.	Pass	Test was done with LRL of 60, URL of 175, and AV delay of 150ms
Natural Atrium = URL, AV delay > programmed AVDelay, shaking	No atrial pulses should be seen. Ventricular pacing occurs after ventricular natural pulse.	Pass	Test was done with LRL of 60, URL of 175, and AV delay of 150ms

As expected DDDR mode paces when the HR is greater than the natural atrium rhythm, and does not pulse when the natural atrium rhythm is greater than the HR. It also pulses the ventricle when there are no natural ventricle signals in the AV delay time. This indicates that both the rate-adaptive pacing and the DDDR mode work as intended.

5.2.4 Serial Communication Testing

The serial communication protocol was the only major component in the Pacemaker Firmware which was tested using a whitebox testing methodology and specific unit testing, rather than formal integration testing at the end. To be clear, each of these tests has been reconducted after the DCM was complete using a blackbox testing methodology to ensure integration testing. However, the serial communication was initially tested separately from the DCM with a different set of Python code which is not provided as the final DCM code uses a similar logic. Each serial conversation type is tested below. For specifics regarding the requirements and expected behaviour of the serial protocol see separate Serial Protocol Documentation.

The serial communication tested in the DCM Documentation provides sufficient evidence that the integration testing of the serial communication between Pacemaker Firmware and DCM is done correctly. The fact that all other testing is completed by using the DCM to Pacemaker Firmware serial interface and these tests produce the expected results is in essence blackbox testing which proves that the serial communication is working. However, prior to integration with the DCM, the Pacemaker Firmware's serial communication was tested with the following tests.

5.2.4.1 Handshake Process

The important parts to test with the handshake process are that the handshake will only be recognized if it is sent with the correct format, the pacemaker should respond to a handshake message with a message containing its serial number, and all other message types should be ignored until the handshake has been completed. As shown in Table 5.2.4.1.a, each of these requirements is tested and returns the expected results. As such it can be said that the serial communication is working correctly.

Table 5.2.4.1.a: Test set used to verify the handshake process is correct.

Test	Expected Result	Pass/Fail	Comment

Handshake sent with correct message type and validation bytes.	Handshake reply echos the message ID and message type, and attaches the correct serial number.	Pass	No comment
Handshake is sent with incorrect message type but correct validation bytes.	No response	Pass	Tested by sending each of the other four message types and ten random message types.
Handshake is sent with correct message type but incorrect validation bytes.	No response	Pass	Tested by sending messages which one validation byte was incorrect and by sending messages which the validation bytes were all random.
Handshake is sent with correct message type but incorrect validation bytes, then sent again with correct message type and validation bytes using a different message ID than the first message.	Pacemaker replies only once and the handshake reply echos the second message ID and message type, and attaches the correct serial number.	Pass	Tested by sending messages which one validation byte was incorrect and by sending messages which the validation bytes were all random.
Non-handshake message type is sent prior to handshake completion.	No response	Pass	Tested by sending each of the other four message types and ten random message types.
Non-handshake message type is sent prior to handshake completion, then handshake sent with correct message type and validation bytes using a different message ID than the first message.	Pacemaker replies only once and the handshake reply echos the second message ID and message type, and attaches the correct serial number.	Pass	Tested by sending each of the other four message types and ten random message types before the handshake message.

5.2.4.2 Polling Process

Testing the polling process is very similar to testing the handshake process as they have the same response message type. However, the big difference between the two is that the handshake is only completed once and requires the DCM to send validation bytes within the message, but the polling request is done repeatedly to ensure the DCM and pacemaker are still connected and does not require validation bytes. As such the polling request should be tested to ensure that it responds when prompted and with the correct message type, message ID, and serial number. Note that the handshake is completed prior to any of the test cases being executed. Each of the test cases provided in Table 5.2.4.2.a pass and as such the polling request system is determined to be working correctly.

Table 5.2.4.2.a: Test set used to verify the polling process is correct.

Test	Expected Result	Pass/Fail	Comment
Polling request sent with correct message type.	Polling reply echos the message ID and message type, and attaches the correct serial number.	Pass	No comment

Non-polling request message type is sent.	No response unless it is another valid message type. Then it should respond with that reply not a polling reply.	Pass	Tested by sending each of the other four message types and ten random message types.
Non-polling request message type is sent, followed by correct polling request.	First message: no response unless it is another valid message type then it should respond with that reply not a polling reply. Second message: polling reply echos the message ID and message type, and attaches the correct serial number.	Pass	Tested by sending each of the other four message types and ten random message types.

5.2.4.3 Parameter Change Process

The process for changing parameters is slightly more complicated than the two previous processes as it requires two message types and a minimum of four messages to complete. So testing is a little more involved. At this stage, the expected results were based on the replies provided and the colour of the LED indicating the mode. The other parameters, while changed, were not necessarily verified at this stage. However, it can still be said that this is completed correctly as all of the previous tests involving the parameters and modes returned the expected results and as such imply that the serial communication used to change the parameters is correct. Each of the test cases is described in Table 5.2.4.3.a and it is important to note that a successful handshake was completed before running the test case and that each parameter change request involves changing the mode so the LED should change once the parameters are implemented into the pacemaker's operation. These results indicate that the parameter change process is correct as all tests pass.

Table 5.2.4.3.a: Test set used to verify the parameter change process is correct.

Test	Expected Result	Pass/Fail	Comment
Parameter change request sent with correct message type. Then parameter confirmation is sent with the correct message type.	First message: parameter interpretation reply echos the message ID and message type, and returns all the correct parameters sent. The LED does not change. Second message: parameter implementation reply echos the message ID and message type, and returns all the correct parameters. LED changes.	Pass	Tested for each mode.
Non-parameter change request message type is sent.	No response unless it is another valid message type. Then it should respond with that reply.	Pass	Tested by sending each of the other four message types and ten random message types.
Correct parameter change request sent. Parameter confirmation never sent.	Parameter interpretation reply echos the message ID and message type, and returns all the correct parameters sent. The LED does not change.	Pass	No comment

Correct parameter change request sent. Then non-parameter confirmation was sent.	First message: parameter interpretation reply echos the message ID and message type, and returns all the correct parameters sent. The LED does not change. Second message: no response unless it is another valid message type then it should respond with that reply.	Pass	Tested by sending each of the other four message types and ten random message types.
Correct parameter change request sent. Then non-parameter confirmation was sent. Then correct parameter confirmation.	First message: parameter interpretation reply echos the message ID and message type, and returns all the correct parameters sent. The LED does not change. Second message: no response unless it is another valid message type then it should respond with that reply. Third message: parameter implementation reply echos the message ID and message type, and returns all the correct parameters. LED changes.	Pass	Tested by sending each of the other four message types and ten random message types.
Correct parameter confirmation sent.	Parameter implementation reply echos the message ID and message type, and returns all the default parameters. The LED does not change.	Pass	No comment
Correct parameter change request sent. Then correct parameter confirmation. Then correct parameter confirmation sent again.	First message: parameter interpretation reply echos the message ID and message type, and returns all the correct parameters sent. The LED does not change. Second message: parameter implementation reply echos the message ID and message type, and returns all the correct parameters. LED changes. Third message: parameter implementation reply echos the message ID and message type, and returns all the previous parameters. The LED does not change.	Pass	No comment

Correct parameter change request sent. Then correct parameter confirmation. Then non-parameter related messages are sent. Then correct parameter confirmation sent again.	First message: parameter interpretation reply echos the message ID and message type, and returns all the correct parameters sent. The LED does not change. Second message: parameter implementation reply echos the message ID and message type, and returns all the correct parameters. LED changes. Third message: no response unless it is another valid message type then it should respond with that reply. Fourth message: parameter implementation reply echos the message ID and message type, and returns all the previous parameters. The LED does not change.	Pass	Tested by sending each of the other four message types and ten random message types.

5.2.4.4 Egram Process

Finally, the egram process can be tested. This process is unique as it is the only message type in which the Pacemaker Firmware sends a message periodically and not just in response to a DCM message like all of the other message types. The general shape provided by the egram should be similar to the shape seen in HeartView of the pacemakers paces. This is because the board which represents the heart presumably paces similarly to the pacemaker by discharging capacitors. The egram data should change when the heart rate or pulse width of the heart is changed in the HeartView monitoring tool. The following test cases, provided in Table 5.2.4.4.a, test to ensure the DCM can toggle on and off the Pacemaker Firmware sending egram data, and tests if the egram data sent seems reasonable. Note that a successful handshake is completed before each test case is run. Each of these test cases passes and as such it is responsible to say the egram process is working correctly.

Table 5.2.4.4.a: Test set used to verify the egram process is correct.

Test	Expected Result	Pass/Fail	Comment
Toggle on egram message request sent with correct message type.	Pacemaker sends egram data periodically forever.	Pass	The pacemaker was allowed to send constant data for 30 minutes at which point it was considered “forever”.
Non-toggle egram message message type is sent.	No response unless it is another valid message type. Then it should respond with that reply.	Pass	Tested by sending each of the other four message types and ten random message types.
Non-toggle egram message type is sent, followed by a correct toggle egram message.	First message: no response unless it is another valid message type then it should respond with that reply. Second message: pacemaker sends egram data periodically forever.	Pass	Tested by sending each of the other four message types and ten random message types. The pacemaker was allowed to send constant data for 2 minutes at which point it was considered “forever”.

Correct toggle on egram message sent. Then after a minute waiting period, correct toggle off the egram message sent.	First message: pacemaker sends egram data periodically for one minute. Second message: pacemaker stops sending egram data.	Pass	No comment
Correct toggle on egram message sent. Then a minute waiting period in which other non-toggle egram messages are sent. Then correct toggle off the egram message sent.	First message: pacemaker sends egram data periodically for one minute. Intermediate messages: no response unless it is another valid message type then it should respond with that reply. Final message: pacemaker stops sending egram data	Pass	Tested by sending each of the other four message types and ten random message types.
Correct toggle egram message is sent. Then the natural heart rate is increased.	Pacemaker sends egram data periodically and the period between peaks decreases as the heart rate changes.	Pass	Tested between minimum and maximum HeartView heart rate values.
Correct toggle egram message is sent. Then the natural heart rate is decreased.	Pacemaker sends egram data periodically and the period between peaks increases as the heart rate changes.	Pass	Tested between minimum and maximum HeartView heart rate values.
Correct toggle egram message is sent. Then natural heart pulse width is increased.	Pacemaker sends egram data periodically and the width of the pulses increases as the pulse width changes.	Pass	Tested between minimum and maximum HeartView heart rate values.
Correct toggle egram message is sent. Then natural heart pulse width is decreased.	Pacemaker sends egram data periodically and the width of the pulses decreases as the pulse width changes.	Pass	Tested between minimum and maximum HeartView heart rate values.
Correct toggle egram message is sent. Both the natural atrium and ventricle are off.	Pacemaker sends egram data periodically but it’s a flat line.	Pass	No comment
Correct toggle egram message is sent. Natural atrium is on and the ventricle is off.	Pacemaker sends egram data periodically atrium is pulsing but ventricle is a flat line.	Pass	No comment
Correct toggle egram message is sent. Natural atrium is off and the ventricle is on.	Pacemaker sends egram data periodically, the ventricle is pulsing but the atrium is a flat line.	Pass	No comment