SE 3K04
Dr. Bokhari
Group 10
November 29, 2020

SE 3K04: Assignment 2
Pacemaker Development

Old Information, New Information

Hamza Ashraf
Sumer Karir
Abigail Nevo
Graham Power
Stephanie Ralph

Revision History

Rev.	Description	Date (Y-M-D)
P1	Preliminary internal release for October 24, 2020, Internal Team Review. **Authors:** Hamza Ashraf, Sumer Karir, Abigail Nevo, Graham Power, Stephanie Ralph	2020-10-15
A	External document release for November 1, 2020, Assignment One Deadline **Authors:** Hamza Ashraf, Sumer Karir, Abigail Nevo, Graham Power, Stephanie Ralph	2020-11-01
B	External document release for November 29, 2020, Assignment Two Deadline **Authors:** Hamza Ashraf, Sumer Karir, Abigail Nevo, Graham Power, Stephanie Ralph	2020-11-29

Related Documents

Designator	Name	Revision
[1]	3K04-Pacemaker DCM (Group 10) HTML	B
[2]	3K04-Pacemaker DCM Testing (Group 10) HTML	B

Contents

Revision History **2**

Related Documents **2**

Contents **3**

On Academic Honesty **5**

Introduction **6**

Planning and Research **6**
- Pacemaker 6
- Device Controller-Monitor 6

Requirements and Specifications **9**

Design and Specification **11**
- Pacemaker Design 11
- DCM Design 12

Monitored Variables and Controlled Quantities **23**
- AOO 23
- VOO 24
- AAI 25
- VVI 27
- DOO 29
- Rate Adaptive Pacing 30
- Merged_Simulink_Modes 32
- Serial Communication 32

Decision Tables **34**
- Pacemaker 34
  - AOO & VOO 34
  - AAI 34
  - VVI 35
  - DOO 35
  - Merged_Simulink_Modes 35
- DCM 36
- Serial Communication 37

States **38**
- Pacemaker 38
  - Stateflow Chart 38
    - AOO 38

VOO 38
AAI 39
VVI 39
DOO 39
Rate Adaptive Pacing 40
Merged Modes 41
State Diagram 42
AOO 42
VOO 43
AAI 44
VVI 45
DOO 46
Rate Adaptive Pacing 47
Merged Modes 47
State Transition Table 49
AOO 49
VOO 50
AAI 51
VVI 52
DOO 53
Rate Adaptive Pacing 55
Serial Communication 56
Simulink Stateflow Tabular Expression 57
AOO 57
VOO 58
AAI 59
VVI 60
DOO 61
Rate Adaptive Pacing 62
States Descriptions 63
DCM 64

Design Decisions 65
AOO, VOO, AAI & VVI 65

Hardware Hiding 68
Merged Modes 69

Testing 69
Simulink 69
Python 74

On Academic Honesty

As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is my own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario. Submitted by:

[Hamza Ashraf, ashrah4, 400201553]:

[Sumer Karir, karirs, 400186680]:

[Abigail Nevo, nevoa, 400170276]:

[Graham Power, powerg, 400198753]:

[Stephanie Ralph, ralphs, 400171292]:

2020.11.29

Introduction

A pacemaker is a device used to pace, sense, and provide electrical response to the heart where needed. It generates electrical impulses delivered through electrodes to make the heart chambers contract and pump blood. This device in turn regulates the electrical conduction system of the heart to maintain a safe and healthy body.
We were instructed to design and develop the basis for our pacemaker using Simulink and any trouble shooting was done on Heartview to make sure that the stateflow diagrams were operating correctly.

A Device Controller-Monitor (DCM) Is a type of Graphical User Interface (GUI) capable of displaying information about the device it controls, as well as communicating with that device to pass control instructions. For this project, the DCM needs to store relevant pacemaker parameters for up to ten registered users, and allow logged in users to view and modify these parameters.

Planning and Research

Pacemaker

Prior to any coding and development, research was key to a successful design. Understanding all of the documents and making notes of the key takeaways was crucial from the starting. This made referencing and implementation of the design a lot easier. Once we understood what was asked of us, we prepared some requirements and specifications, in the following section, to have some clear tasks which needed to be completed. Assignment 1 was then divided amongst the group where three members would work on the pacemaker design and two on the DCM.
In the following sections, we go over what our initial design and thought process was when we were starting out and how it evolved as we progressed throughout this project.
We came up with a high level flowchart to provide an understanding of how our pacemaker would work and where the different segments we worked on would fit.

Device Controller-Monitor

Given the lenience in the DCM Requirement Specifications as to what technologies are to be used, all options must be considered before choosing an implementation to ensure the chosen technologies allow all requirements to be met. To assist in choosing these technologies, the DCM was split into three categories; GUI display, data management, and serial communication. Possible implementations of each category were compared and scored to determine their implementations.

Table #1: GUI Display Technology Comparison

Technology	Supported Implementation Languages	Comments	Overall Score (/10)
Tkinter	Python	● Language both developers are comfortable with ● Only allows for basic GUI's (not visually pleasing)	4
Web (HTML/CSS)	Python - Flask JavaScript Java - Spring MVC	● Allows for modern GUI design (visually pleasing) ● Using flask allow developers to use a language both are comfortable with ● Easily scalable	8
Swing	Java	● Platform agnostic ● Uses MVC paradigm, allowing for a more flexible UI ● Developers are not a string in Java as Python	5
Qt	C++	● Qt webkit allows for JavaScript to be connected to C++ code ● Comes with a GUI designer ● DB integration will be harder in C++ than Python	7

Table #2: Data Management Technology Comparison

Technology	Supported Implementation Languages	Comments	Overall Score (/10)
MongoDB	Python - MongoPy JavaScript - Mongoose	● Non-relational database with excellent documentation and an easy learning curve. ● Supporting packages allow for convenient integration with Flask applications. ● Developers have more experience using Mongo	5
Firebase	JavaScript - Firebase SDK	● Non-relational database with excellent documentation. ● Requires registering an account with Google Apps alongside payment information.	2
SQL	Postgres Sqlite3 - Python	● Relational database seems a better data representation for the data we need to store in this project ● Using sqlite3 allows for direct integration with a Python project	7

Table #3: GUI Serial Communication Comparison

Technology	Supported Implementation Languages	Comments	Overall Score (/10)
Node.js SerialPort	JavaScript - Node.js	● Allows access to serial ports ● Not widely used - issues are not well-documented on programming forums such as Stack Overflow.	4
PySerial	Python	● Popular python library with long term support and lots of documentation ● Very easy to implement	10

Using the scores from the above tables, the technologies to use in DCM development can be chosen. The highest scoring implementation from every table shared the Python programming language in common. So the technologies that will be used to create the DCM will be; Python, Flask, SQLite3, PySerial.

Requirements and Specifications

Here we have listed some of the key tasks which our design must accomplish and under which restraints it must operate under. For convenience, any requirements or specifications referenced later in this document have been provided with an identifying name.

Table #4: Implementation Requirements and Specifications

Pacemaker Implementation	
Requirements	**Specifications**

- Pacemaker must be able to emit an electrical stimuli which can pulse the heart safely
- Pacing modes to implement:
  - AOO
  - VOO
  - AAI
  - VVI
- The 'heart' and 'pacemaker' hardware must be able to communicate with each other and provide the appropriate output where needed.
- Implement Dual pacing mode where both atrium and ventricle can be paced simultaneously
- Implement Rate Adaptive Pacing modes which will increase the pulses sent by pacing maker depending on the movement it detects
- Link Simulink workflows to the DCM
  - Allows new pacing mode to be implemented without having to restart board
	- Pacing modes implemented must be precise in design and follow all safety guidelines (size of impulse, frequency of impulse, etc.)
	- Pacemaker modes must be designed using Simulink software and downloaded to NXP FRDM-K64F hardware where design is tested
	- Must utilize accelerometer to implement rate-adaptive pacing modes
	- Use serial communication ports to connect DCM to pacing modes developed in Simulink

DCM Implementation	
ID	**Requirements**
DCM-REQ-01	The user interface shall be capable of utilizing and managing windows for display of text and graphics

DCM-REQ-02	The user interface shall be capable of processing user positioning and input buttons
DCM-REQ-03	The user interface shall be capable of displaying all programmable parameters for review and modification
DCM-REQ-04	The user interface shall be capable of visually indicating when the DCM and the device are communicating
DCM-REQ-05	The user interface shall be capable of visually indicating when a different pacemaker device is approached than was previously interrogated
DCM-REQ-06	The user interface shall be capable of serial communication with the Pacemaker
DCM-REQ-07	The user interface shall be capable of limiting the range and step of all programmable parameters
DCM-REQ-08	The user interface shall be capable of displaying egram data from the pacemaker in real-time
DCM-REQ-09	The DCM shall record all actions and events taken in a detailed manner for debugging purposes.
ID	**Specifications**
DCM-SPEC-01	The DCM shall include a welcome screen, including the ability to register a new user and log in as an existing user
DCM-SPEC-02	Develop interfaces to present all of the pacing modes mentioned in ‘Pacemaker Implementation’
DCM-SPEC-03	Make provisions for storing programmable parameter data and checking inputs
DCM-SPEC-04	Develop interfaces to present the egram data live to the user
DCM-SPEC-05	Make provisions for programming the pacemaker through a serial connection and verifying the parameters were programmed correctly

Design and Specification

After understanding the requirements and specifications and having the appropriate amount of research done, we believed we were ready to begin designing the different components of the pacemaker.

Pacemaker Design

Understanding the anatomy of the heart and it's electrical conduction system was a key starting area, as we had to understand where the electrical stimuli would be applied. We also had to learn of the different pacing modes and what their particular specifications are so that they can be programmed accordingly. For assignment 1 we were required to implement 4 different modes which were AOO, VOO, AAI, and VVI. Each has a distinctly different operation which it conducts, as described below.

AOO:

- Atrial pacing, no sensing, atrial asynchronous pacing at lower programmed pacing rate

VOO:

- Ventricular pacing, ventricular sensing, sensed intrinsic QRS inhibits ventricular pacing

AAI:

- Atrial pacing, atrial sensing, intrinsic P wave inhibits atrial pacing

VVI:

- Ventricular pacing, ventricular sensing, sensed intrinsic QRS inhibits ventricular pacing

DOO:

- Pacing in atrium and ventricle, intrinsic P wave and QRS do not affect pacing
- Asynchronous Pacing - always pace at lower pace rate

Rate Adaptive Pacing Modes

- Rate Adaptive pacing was designed to increase pacing when the accelerometer detected activity greater than the activity threshold, and decrease pacing back to the Lower Rate Limit when activity is less than the threshold.

DCM Design

To aid in the development of the DCM, it was split into different modules. Each module was designed to be independent of one another and handle different tasks for the DCM. The description of the four modules, their requirements, secrets, and implementations are described in the charts below. To ensure the DCM development was productive and did not include any redundant or counter-productive features or code, Each DCM module must be able to prove it fulfils only the scope of its module specific requirements and that those requirements work towards fulfilling the DCMs overall requirements. Each module specific requirement must be theoretically testable and provide a description of how those tests will be administered. For a full description of all code implementations please refer to the Code Documentation [1].

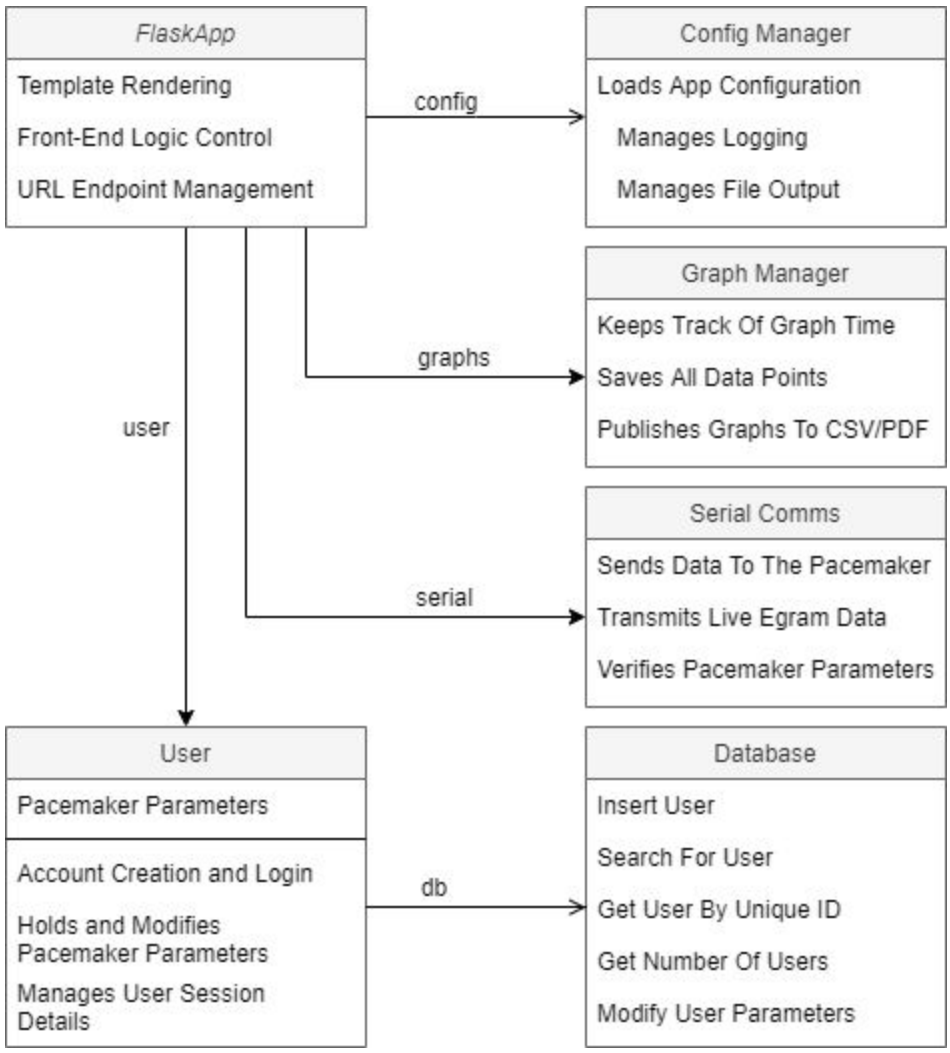


Figure #1: DCM Modular Design

Table #5: FlaskApp Module Details, Requirements, and Testing Plans

FlaskApp Module Details	
Module Description:	This module is designed to manage the top level functionality of the DCM. It handles the rendering of the DCM as well as submitting data from the DCMs various forms to the backend modules
Module Secrets:	[1] - How content is rendered to the screen [2] - How Egram data is rendered to the screen [3] - How user input is collected and managed [4] - How different parts of the UI are restricted

FlaskApp Module Requirements			
ID	**Module Specific Requirement**	**Testing Plan**	**DCM Requirement or Specification This Helps Fulfil**
DCM-APP-REQ-01	The FlaskApp shall be capable of providing accessible endpoints for the following user windows: login page, user specific page, user parameters page, user pacemaker connection page, user history page, user egram page, and logout page.	Each page of the FlaskApp will be checked to ensure it returns a “200 OK” response.	**DCM-REQ-01**
DCM-APP-REQ-02	The FlaskApp shall be capable of properly displaying all required windows from DCM-APP-REQ-01	Each page of the FlaskApp will be rendered and have their contents compared with their expected contents to ensure they have rendered correctly. Pictures will also be taken of each rendered page for visual verification.	**DCM-REQ-01** **DCM-REQ-03**
DCM-APP-REQ-03	The FlaskApp shall be capable of generating a post request when any button is pressed	Every button in the FlaskApp will be activated and the FlaskApp will be checked to ensure that a post request was generated.	**DCM-REQ-02**
DCM-APP-REQ-04	The FlaskApp shall display an icon in the	The user specific page of the FlaskApp will be rendered	**DCM-REQ-04**

	top-right to indicate whether or not it is connected to a pacemaker	before and after a pacemaker connection is made to ensure the icon changes from a no connection icon to a connection icon. Pictures will also be taken for visual verification.	
DCM-APP-REQ-05	The FlaskApp shall provide a popup notification telling the user they have connected a different pacemaker than before and ask them to log in again	**NOT POSSIBLE** Since all team members are working remotely no one has access to two different pacemakers, so this requirement must go untested.	**DCM-REQ-05**
DCM-APP-REQ-06	The FlaskApp's user parameters and user pacemaker connection page shall not allow the user to submit any parameter changes outside of their restricted values or steps	A parameter will be initialized with a known range and step and an attempt will be made to enter values both over and under its range as well as outside its step. The form will be monitored to ensure this attempt fails. Pictures will also be taken for visual verification.	**DCM-REQ-07**
DCM-APP-REQ-07	The FlaskApp shall be capable of requesting data points from the Graph Manager and rendering those points to a graph in real time	The user egram page will be loaded and requests for data points from the Graph Managers test function will be made. The egram page will be monitored to ensure the atrium and ventricle lines form sine and cosine waves respectively. Pictures will also be taken for visual verification.	**DCM-REQ-08** **DCM-SPEC-04**
DCM-APP-REQ-08	The FlaskApp shall be capable of rendering a login screen, allowing existing users to login or new users to be created	The FlaskApp will be initialized with a known user and an attempt to login with that user's credentials will be made to ensure it succeeds. An attempt to login with a series of random credentials not matching the known users will also be made to	**DCM-SPEC-01**

		ensure it fails. Pictures will also be taken for visual verification.	
DCM-APP-REQ-09	The FlaskApp shall limit the number of users created to a number specified by the Config Manager	The FlaskApp will be initialized with a full amount of users, and an attempt to create another user will be made to ensure it fails. Pictures will also be taken for visual verification.	**DCM-SPEC-01**
DCM-APP-REQ-10	The user parameters page shall display all parameters listed by the Config Manager	The user parameters page will be rendered and its list of parameters compared to the list provided by the Config Manager to ensure all parameters are listed correctly. Pictures will also be taken for visual verification.	**DCM-SPEC-02**

Table #6: Config Manager Module Details, Requirements, and Testing Plans

Config Manager Module Details			
	Module Description:	This module is designed to manage all variable parameters for the application in a way which allows them to be easily changed by users with limited or no coding knowledge, even after the code has been compiled. It also attempts to collect all application parameters into one convenient place.	
	Module Secrets:	[1] - How parameters are stored and loaded [2] - How .ini files are parsed into python variables	
Config Manager Module Requirements			
ID	**Module Specific Requirement**	**Testing Plan**	**DCM Requirement or Specification This Helps Fulfil**
DCM-CFG-REQ-01	The Config Manager shall store all pacemaker parameters and modes to be used in the DCM as well as lists	The Config Manager will be loaded with a known list of parameters, modes, and combinations and will be checked to ensure those	**DCM-REQ-03**

	of which parameters correspond to which modes. This information shall be made readily available to the rest of the application	variables are both correct and available to the entire application.	
DCM-CFG-REQ-02	The Config Manager shall store any limits/steps to pacemaker parameters as well as whether or not they can be disabled	The Config Manager will be loaded with a known list of parameters, limits, and steps and will be checked to ensure those variables are both correct and available to the entire application.	**DCM-REQ-07**
DCM-CFG-REQ-03	The Config Manager shall store parameters detailing how the egram data should be displayed to the user and how often this data should be updated	The Config Manager will be loaded with a known list of graphing variables and will be checked to ensure those variables are both correct and available to the entire application.	**DCM-REQ-08**
DCM-CFG-REQ-04	The Config Manager shall store parameters detailing log level and log message format.	The Config Manager will be loaded with a known list of logging variables. The Logger will be initialized and have known messages at various levels posted to it to ensure it is behaving as expected given the known Config Manager variables.	**DCM-REQ-09**

Table #7: User Module Details, Requirements, and Testing Plans

User Module Details	
Module Description:	This module is designed to manage account creation and login, as well as individual users' pacemaker

	parameters during runtime.
Module Secrets:	[1] - How a user is logged in [2] - How a new account is created [3] - How pacemaker parameters are stored and modified at runtime [4] - How to communicate with the Database module

User Module Requirements			
ID	**Module Specific Requirement**	**Testing Plan**	**DCM Requirement or Specification This Helps Fulfil**
DCM-USR-REQ-01	The User shall store all programmable parameters at runtime and provide them to the FlaskApp on request	A user will be created with known parameters and a request for those parameters will be made by the FlaskApp to ensure those parameters are both provided on request and correct.	**DCM-REQ-03**
DCM-USR-REQ-02	The User shall allow parameter modification by the FlaskApp at runtime	A user will be created with known parameters, those parameters will be changed by the FlaskApp. Then a request for user parameters will be made to ensure the change was successful.	**DCM-REQ-03**
DCM-USR-REQ-03	The User shall store all parameter limits/steps at runtime and provide them to the FlaskApp on request	A user will be created with known parameter limits/steps and a request for those limits/steps will be made by the FlaskApp to ensure those limits/steps are both provided on request and correct.	**DCM-REQ-07**
DCM-USR-REQ-04	The User shall be capable of logging in users on request by the FlaskApp	A user with known login credentials will be created and a login attempt will be made by the FlaskApp to ensure the User can perform a login request successfully.	**DCM-SPEC-01**
DCM-USR-REQ-05	The User shall be capable of creating a	A series of user creation requests will be made by the	**DCM-SPEC-01**

	new user	FlaskApp to ensure new users can be successfully created and that user creation will be denied if a user with similar credentials already exists.	
DCM-USR-REQ-06	The User shall be capable of storing parameter data internally at runtime and commanding the Database to make permanent changes to its parameters to match that of the Users	A series of parameter set and get requests will be made to ensure the user is capable of internally storing and modifying these parameters. The application will then be restarted and a parameter get request will be made to ensure the parameters values persist between runs of the application.	**DCM-SPEC-03**

Table #8: Database Module Details, Requirements, and Testing Plans

Database Module Details	
	This module is designed to store all user credentials

Module Description:	as well as all users current and past parameters and values. This storage is designed to persist between runs of the application to ensure all actions by the users are remembered indefinitely.
Module Secrets:	[1] - Where the user credentials and parameters are stored [2] - How the variables are retrieved, set, and modified during runtime

Database Module Requirements			
ID	**Module Specific Requirement**	**Testing Plan**	**DCM Requirement or Specification This Helps Fulfil**
DCM-DBS-REQ-01	The Database shall be capable of creating, storing, and retrieving user login credentials within a SQL table	A test database will be created and new users will be created and stored into the database. Those users will then be retrieved and compared with their original to ensure the database correctly stored and returned their values. Pictures will also be taken for visual verification.	**DCM-SPEC-03**
DCM-BDS-REQ-02	The Database shall be capable of storing each users pacemaker parameters and mode in a user specific SQL table	A user will be created and a sequence of pacemaker parameter changes will be made. The final parameter values will then be retrieved from the database to ensure they are correct.	**DCM-SPEC-03**
DCM-DBS-REQ-03	The Database shall be capable of creating and storing a timestamped history of every change made to the pacemaker parameters as the changes occur	A user will be created and a sequence of pacemaker parameter changes will be made. A history of all changes will then be retrieved from the database to ensure they are correct.	**DCM-SPEC-03**

Table #9: Graph Manager Module Details, Requirements, and Testing Plans

Graph Manager Module Details

Module Description:	This module is designed to manage the live egrams displayed to the user. It provides the DCM with live point data from both the atrium and ventricle to graph and keeps track of all points graphed even after they are moved off the live graph. It can also publish a complete graph to both csv and pdf forms.
Module Secrets:	[1] - How the point data is retrieved and formatted [2] - How the point data history is stored [3] - How the point data is published to csv and pdf

Graph Manager Module Requirements			
ID	**Module Specific Requirement**	**Testing Plan**	**DCM Requirement or Specification This Helps Fulfil**
DCM-GPH-REQ-01	The Graph Manager shall be capable of providing real time data on request by the FlaskApp	Requests for both real time data from the pacemaker and test data generated by the Graph Manager will be made and graphed by the FlaskApp to ensure data can both be provided, and that the data provided is correct. Pictures will also be taken for visual verification.	**DCM-REQ-08**
DCM-GPH-REQ-02	The Graph Manager shall be capable of storing all past data from the most recent live graph	The Graph Manager will be polled periodically throughout a live graping session of known test data to ensure that all past data has been properly kept and is still correct.	**DCM-REQ-08**
DCM-GPH-REQ-03	The Graph Manager shall be capable of publishing all past data from the most recent live graph in both csv and pdf formats upon request	A publish request will be made to the Graph Manager during a live graping session of known test data to ensure the data is properly published to both a csv and pdf format and the data appearing in the published documents is still correct. Pictures will also be taken for visual verification.	**DCM-REQ-08**

Table #10: Serial Communication Module Details, Requirements, and Testing Plans

Serial Communication Module Details			
Module Description:		This module is designed to manage serial communication between the DCM and the Pacemaker.	
Module Secrets:		[1] - How the DCM and Pacemaker receive and transmit information. [2] - How the information is converted to data packets as per the UART protocol. [3] - How the information from the data packets is unpacked and interpreted.	
Serial Communication Module Requirements			
ID	**Module Specific Requirement**	**Testing Plan**	**DCM Requirement or Specification This Helps Fulfil**
DCM-SRL-REQ-01	The pacemaker shall be capable of receiving data from the DCM through serial communication.	A known data packet will be serially communicated from the DCM to the Pacemaker. The data packet shall switch between different modes. The LED functionality for each mode is unique and will be used to recognize the transfer of data.	**DCM-REQ-06**
DCM-SRL-REQ-02	The pacemaker shall be able to interpret the received data and implement the new parameter values.	A known data packet will be serially communicated from the DCM to the Pacemaker. The data packet shall provide a set of parameters, and the functionality will then be verified through Heartview.	**DCM-REQ-06** **DCM-SPEC-05**
DCM-SRL-REQ-03	The DCM shall be capable to receiving data from the pacemaker through serial communication	A known data packet will be serially communicated from the Pacemaker to the DCM. The received packet will then be printed to ensure that serial communication was successful.	**DCM-REQ-06**
DCM-SRL-REQ-04	The DCM shall be capable of interpreting	A known data packet will be serially communicated from	**DCM-REQ-06** **DCM-SPEC-04**

	the egram data received from the pacemaker, and to display it.	the Pacemaker to the DCM. The received data packet shall be unpacked and the contents will be compared to the contents that were sent. The received egram data shall be displayed on the DCM.	**DCM-SPEC-05**

Monitored Variables and Controlled Quantities

AOO

Table #11: AOO Controlled Quantities

Name	Function	Initialization Values	Limitations
Controlled Quantities			

PACING_REF_PWM_D5	Charges Primary Capacitor	Amp = 100	Is set to a constant value to make sure that the pacemaker is functioning under all conditions.
PACE_CHARGE_CTRL_D2	Starts and Stops PACING_REF_PWM_D5 ie. the charging of the capacitor	1	Is set to low if the pace control (pin D8) is high, meaning that a pulse is being emitted to the atrium
ATR_PACE_CTRL_D8	Discharges the primary capacitor through the atrium	0	Set to low in charging state, and high in pacing state. This indicates when the capacitors charge will be released to emit a pulse.
ATR_GND_CTRL_D11	Stops charge buildup during primary capacitor discharge	1	Capacitor has a charge limit.
PACE_GND_CTRL_D10	Controls current flow to the tip of the atrium from the ring	0	Set to low in charging state, and high in pacing state to prevent charge overflow to heart.
Z_ATR_CTRL_D4	Controls and analyzes impedance at the atrial electrode, and the connection between the atrial electrodes and the atrium	0	Always kept to low to prevent the pacemaker circuit from short circuiting.
PacePeriod	Converts BMP to total time from end of one pulse to the end of another pulse in msec	60000ms/45 (45 BPM)	Inputted into larger state flow. Dictates transition timing between charging and pacing state.
PulseWidth	Length of pulse, how long the capacitor is discharging in msec	20	

VOO

Table #12: VOO Controlled Quantities

Name	Function	Initialization Values	Limitations

Controlled Quantities			
PACING_REF_PWM_D5	Charges Primary Capacitor	P_Amp	Set to a constant value to ensure the pacemaker is functioning.
PACE_CHARGE_CTRL_D2	Starts and Stops PACING_REF_PWM_D5 ie. the charging of the capacitor	1	Is set to high in charging and low in pacing. This prevents the capacitor from charging while it is emitting the charge as a pulse to the ventricle.
VENT_PACE_CTRL_D9	Discharges the primary capacitor through the ventricle	0	Set to low in charging and high in pacing state. This allows the capacitor to build up sufficient charge.
VENT_GND_CTRL_D12	Stops charge buildup during primary capacitor discharge	1	Set to high in charging, and low in pacing state. Prevents over charging of capacitor.
PACE_GND_CTRL_D10	Controls current flow to the tip of the ventricle from the ring	0	Set to low in charging and high in pacing. Prevents the ventricle from receiving too much charge from the pulse.
Z_VENT_CTRL_D7	Connects the impedance circuit and the ventricle ring electrode	0	Always set to low to keep the circuit closed. Prevents it from short circuiting.
PacePeriod	Converts BMP to total time from end of one pulse to the end of another pulse in msec	60000/60 (60BPM)	Inputted into larger state flow. Dictates transition timing between charging and pacing state.
PulseWidth	Length of pulse, how long the capacitor is discharging in msec	20	

AAI

Table #13: AAI Monitored Variable and Controlled Quantities

Name	Function	Initialization	Limitations

		Values	
Monitored Variables			
ATR_CMP_DETECT_D0	Atrium Sensing	1	
PushButton	Pushed outputs true to inhibit the pulse Not pushed outputs false and the pace is not inhibited	0	
Controlled Quantities			
PACING_REF_PWM_D5	Charges Primary Capacitor	P_Amp	Set to a constant value to ensure the pacemaker is functioning.
PACE_CHARGE_CTRL_D2	Starts and Stops PACING_REF_PWM_D5 ie. the charging of the capacitor	1	Cannot be high if ATR_PACE_CTRL_D8 is high or the patient's atrium could be connected directly to the PWM signal
ATR_PACE_CTRL_D8	Discharges the primary capacitor through the atrium	0	Cannot be set to high or the patient's atrium could be connected directly to the PWM signal
ATR_GND_CTRL_D11	Stops charge buildup during primary capacitor discharge	1	Capacitor has a charge limit.
PACE_GND_CTRL_D10	Controls current flow to the tip of the atrium from the ring	0	Has to be high as it controls the switch which follows the tip
Z_ATR_CTRL_D4	Controls and analyzes impedance at the atrial electrode, and the connection between the atrial electrodes and the atrium	0	Always kept to low to prevent the pacemaker circuit from short circuiting.
FRONTEND_CTRL_D13	Starts sensing circuitry	1	
ART_CMP_REF_PWM_D6	Limit for when atrial action potential should be sensed	S_Amp	

PacePeriod	Converts BMP to total time from end of one pulse to the end of another pulse in msec	60000/45	Inputted into larger state flow. Dictates transition timing between charging and pacing state.
PulseWidth	Constant, represents length of pulse, how long the capacitor is discharging in msec	20	
P_Amp	Constant, represents the pacing PWM output that controls the amplitude of the pace	100	0 to 100
S_Amp	Constant, represents the sensing PWM threshold in the sensing circuit	80	0 to 100
ARP	Constant, represents the Atrial Refractory Period in msec	150	

VVI

Table #14: VVI Monitored Variable and Controlled Quantities

Name	Function	Initialization Values	Limitations
Monitored Variables			
VENT_CMP_DETECT_D1	Ventricular Sensing	1	
PushButton	Pushed outputs true to inhibit the pulse Not pushed outputs false	0	

	and the pace is not inhibited		
Controlled Quantities			
PACING_REF_PWM_D5	Charges Primary Capacitor	P_Amp	Set to a constant value to ensure the pacemaker is functioning.
PACE_CHARGE_CTRL_D2	Starts and Stops PACING_REF_PWM_D5 ie. the charging of the capacitor	1	Cannot be high if VENT_PACE_CTRL_D9 is high or the patient's ventricle could be connected directly to the PWM signal
VENT_PACE_CTRL_D9	Discharges the primary capacitor through the ventricle	0	Cannot be set to high or the patient's ventricle could be connected directly to the PWM signal
VENT_GND_CTRL_D12	Stops charge buildup during primary capacitor discharge	1	Set to high in charging, and low in pacing state. Prevents over charging of capacitor.
PACE_GND_CTRL_D10	Controls current flow to the tip of the ventricle from the ring	0	Has to be high as it controls the switch which follows the tip
Z_VENT_CTRL_D7	Connects the impedance circuit and the ventricle ring electrode	0	Always set to low to keep the circuit closed. Prevents it from short circuiting.
FRONTEND_CTRL_D13	Starts sensing circuitry	1	
VENT_CMP_REF_PWM_D3	Establishes the ventricular action threshold for when sensing should occur	S_Amp	
PacePeriod	Converts BMP to total time from end of one pulse to the end of another pulse in msec	60000/60 = 1000	Inputted into larger state flow. Dictates transition timing between charging and pacing state.
PulseWidth	Constant, represents length of pulse, how long the capacitor is	20	

	discharging in msec		
VRP	Constant, represents time interval, to signify a length of time after an ventricular event, when ventricular sensing will not inhibit or trigger pacing in msec	150	
P_Amp	Constant, represents the pacing PWM output that controls the amplitude of the pace	100	0 to 100
S_Amp	Constant, represents the sensing PWM threshold in the sensing circuit	80	0 to 100

The LED States were used purely for testing but are not connected to the functionality of the pacemaker so they are neither monitored variables or controlled quantities.

All controlled quantities mentioned in the Design Decisions were changed to Monitored Variables.

DOO

Table #15: VVI Monitored Variable and Controlled Quantities

Name	Function	Initialization Values	Limitations
Monitored Variables			
PACING_REF_PWM_D5	Charges Primary Capacitor	Data	Set to a constant value to ensure the pacemaker is functioning.
Pace_Period	Converts BMP to total time from end of one	60000(msec) /BPM	Inputted into larger state flow. Dictates transition

	pulse to the end of another pulse in msec	Values are selected through DCM	timing between charging and pacing state.
Pulse_Width	Constant, represents length of pulse, how long the capacitor is discharging in msec	0 - 20 (msec)	Value cannot exceed 20msec due to restrictions of AV delay
AV_Delay	The delay between pacing the atrium and ventricle since one chamber must retract before the other is paced.	Avg of 30msec	AV delay on average is 30msec and this influences the pulse-width values for safety.
Controlled Quantities			
PACE_CHARGE_CTRL_D2	Starts and Stops PACING_REF_PWM_D5 ie. the charging of the capacitor	1	Cannot be high if VENT_PACE_CTRL_D9 is high or the patient's ventricle could be connected directly to the PWM signal
ATR_PACE_CTRL_D8	Discharges the primary capacitor through the atrium	0	Cannot be set to high or the patient's atrium could be connected directly to the PWM signal
ATR_GND_CTRL_D11	Stops charge buildup during primary capacitor discharge	1	Capacitor has a charge limit.
PACE_GND_CTRL_D10	Controls current flow to the tip of the ventricle from the ring	0	Has to be high as it controls the switch which follows the tip
Z_ATR_CTRL_D4	Controls and analyzes impedance at the atrial electrode, and the connection between the atrial electrodes and the atrium	0	Always kept to low to prevent the pacemaker circuit from short circuiting.
VENT_PACE_CTRL_D9	Discharges the primary capacitor through the ventricle	0	Cannot be set to high or the patient's ventricle could be connected directly to the PWM signal

VENT_GND_CTRL_D12	Stops charge buildup during primary capacitor discharge	1	Set to high in charging, and low in pacing state. Prevents over charging of capacitor.
Z_VENT_CTRL_D7	Connects the impedance circuit and the ventricle ring electrode	0	Always set to low to keep the circuit closed. Prevents it from short circuiting.

Rate Adaptive Pacing

Table #16: Rate Adaptive Pacing Monitored Variable and Controlled Quantities

Name	Function	Initialization Values	Limitations
Monitored Variables			
Adapted_Rate	Rate affected by Accelerometer Outputted rate in Rate Adaptive modes (instead of LRL)	Adapted_Rate = LRL	Between LRL and MSR
LRL	Lower Rate Limit Number of generated pulses delivered/min	60ppm (nominal value)	Programmable Values: 30-50 ppm 50-90 ppm 90-175 ppm
Time	Keeps track of progress throughout reaction time, it is the amount of time that the current state has been running	0	Between 0 to Reaction_Time
Rate_Diff	Assists in calculating slope when decreasing from Adapted_Rate to LRL	Adapted_Rate - LRL	
Recovery_Time	Amount of time to decrease pace from Adapted_Rate to LRL	5 min (nominal value)	Programmable Values: 10-50 sec
Activity_Threshold	Quantity Activity has to surpass in order to	0.2	

	increase rate due to accelerometer		
Response_Factor	Determines the pacing rate at various steady states	8 (nominal value)	Programmable Values: 1-16
Activity_Rate	Calculated desired rate according to Activity	(MSR-LRL_ *Activity	
Reaction_Time	Time required to change the rate from LRL to MSR	30 sec (nominal value)	Programmable Values: 10 - 50 sec
Max_Sensor_Rate	Max pacing rated allowed	120 ppm (nominal value)	Programmable Values: 50 - 175 ppm
Activity	Measures acceleration from accelerometer		

Merged_Simulink_Modes

The merged_simulink_modes collects all of the various heart modes into one file. Therefore all of the monitored variables and controlled quantities that exist in each of the modes also exist in the merged simulink. Below are the only variables/quantities that exist in Merged_Simulink_Modes that do not exist in any of the modes.

Table #17: Merged_Simulink_Modes Monitored Variable and Controlled Quantities

Name	Function	Initialization Values	Limitations
Monitored Variables			
Mode_Select	Allows the input to indicate which heart mode should be	Default Value = 11111	When Mode_Select has not been assigned a value it will be assigned a default value of 11111. This allows the Simulink to remain in Start_State

	used		waiting, waiting for Mode_Select to change.

Serial Communication

Table #18: Data Packet for Transmission from Pacemaker to DCM

Item	Number of Bytes	Start Bit Number	Default Value
MODE	1	1	0
LRL	1	2	60ppm
URL	1	3	120ppm
Maximum_Sensor_Delay	1	4	120ppm
AV_Delay	1	5	150ms
A_Pulse_Amplitude	8	6	5.0V
V_Pulse_Amplitude	8	14	5.0V
A_Pulse_Width	1	22	1ms
V_Pulse_Width	1	23	1ms
A_Sensitivity	8	24	-
V_Sensitivity	8	32	-
V_Refractory_Period	2	40	320ms
A_Refractory_Period	2	42	250ms
PVARP	1	44	250ms
Rate_Smoothing	8	45	Off
Activity_Threshold	1	53	-
Reaction_Time	1	54	30 sec
Response_Factor	1	55	8
Recovery_Time	1	56	5 min
AtrSignal	8	57	-

VentSignal	8	65	-
Function_Call	1	73	0

Table #19: Data Packet for Transmission from DCM to Pacemaker

Parameter Name	Number of Bytes	Start Bit Number	Default Value
MODE	1	1	0
LRL	1	2	60ppm
URL	1	3	120ppm
Maximum_Sensor_Delay	1	4	120ppm
AV_Delay	1	5	150ms
A_Pulse_Amplitude	8	6	5.0V
V_Pulse_Amplitude	8	14	5.0V
A_Pulse_Width	1	22	1ms
V_Pulse_Width	1	23	1ms
A_Sensitivity	8	24	-
V_Sensitivity	8	32	-
V_Refractory_Period	2	40	320ms
A_Refractory_Period	2	42	250ms
PVARP	1	44	250ms
Rate_Smoothing	8	45	Off
Activity_Threshold	1	53	-
Reaction_Time	1	54	30 sec
Response_Factor	1	55	8
Recovery_Time	1	56	5 min
Function_Call	1	57	0

Decision Tables

Pacemaker

AOO & VOO

Table #20: AOO Decision Table

	Pacing	ChargingAndSensing
after(PacingPeriod-Pulse Width): {0,1}	1	*
after(PulseWidth): {0,1}	*	1

AAI

Table #21: AAI Decision Table

	Pacing	ChargingAnd Sensing	ChargingAnd Sensing	ChargingAnd Sensing
ATR_CMP_DETECT: {0,1}	0	1	*	*
PushButton: {0,1}	0	*	1	*
after(PacingPeriod-Pulse Width): {0,1}	1	*	*	*
after(PulseWidth): {0,1}	*	*	*	1

VVI

Table #22: VVI Decision Table

	Pacing	ChargingAnd Sensing	ChargingAnd Sensing	ChargingAnd Sensing
VENT_CMP_DETECT: {0,1}	0	1	*	*
PushButton: {0,1}	0	*	1	*
after(PacingPeriod-Pulse Width): {0,1}	1	*	*	*
after(PulseWidth): {0,1}	*	*	*	1

DOO

Table #23: DOO Decision Table

	AOO Charging	AOO Pacing	VOO Charging	VOO Pacing
after(PacingPeriod-Pulse_Width-AV_Delay): {0,1}	*	1	*	*
after(Pulse_Width); {0,1}	*	*	1	*
after(AV_Delay-Pulse_Width); {0,1}	*	*	*	1
after(Pace_Period-Pulse_Width); {0,1}	1	*	*	*

Merged_Simulink_Modes

Table #24: Merged_Simulink_Modes Decision Table

Mode Select Value	Pacing Mode	Return to Start_State Value
1	AOO	~1
2	VOO	~2
3	AAI	~3
4	VVI	~4
5	DOO	~5
6	AOOR	~6
7	VOOR	~7
8	AAIR	~8
9	VVIR	~9
10	DOOR	~10
0	Start_State (Default)	0

DCM

Table #25: User Login / Account Creation

Conditions:					
Login Attempt	1	1	0	0	0
Account Creation	0	0	1	1	1
Matching Credentials In Database	1	0	1	0	0
Max Users In Database	*	*	*	1	0
Actions:					
Login Granted	1	0	0	0	1
New Account Created	0	0	0	0	1

Table #26: Attempted Change Of Pacemaker Parameters

Conditions:								
VOO Mode	1	1	0	0	0	0	0	0
VVI Mode	0	0	1	1	0	0	0	0
AOO Mode	0	0	0	0	1	1	0	0
AAI Mode	0	0	0	0	0	0	1	1
VOO Parameter Changed	1	0	1	0	1	0	1	0
VVI Parameter Changed	*	*	*	*	*	*	*	*
AOO Parameter Changed	*	*	*	*	*	*	*	*
AAI Parameter Changed	*	*	*	*	*	*	*	*
Action:								
Change Allowed	1	0	1	0	1	0	1	0

Serial Communication

Table #27: Serial Communication Decision Table

Conditions:				
Status_bit	0	0	32	
Function_Call	0x00	0x01	*	1
Actions:				
Change Parameter Values	1	0	0	1
Send Parameters/Egram Data	0	1	1	1

States

Pacemaker

Stateflow Chart

AOO

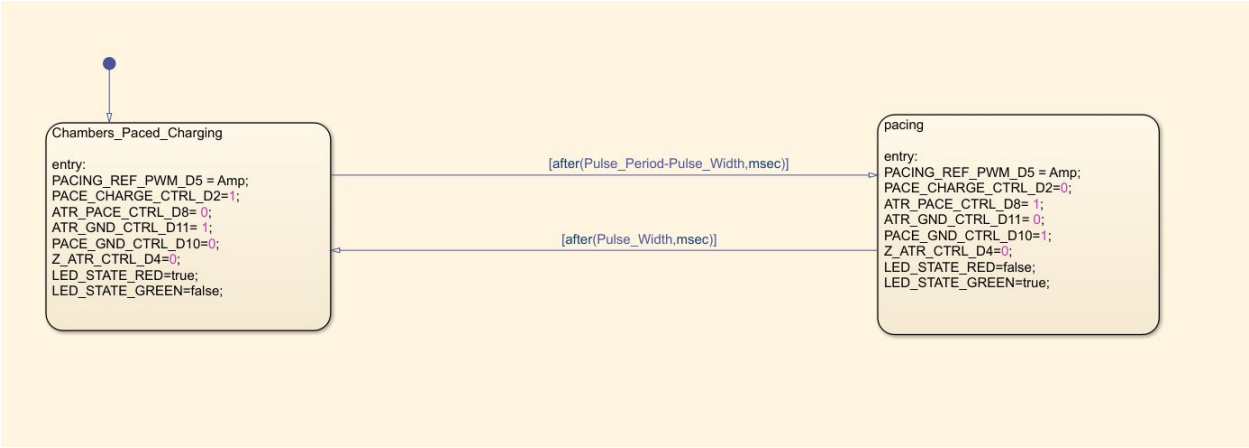


Figure #2: AOO Stateflow Chart

VOO

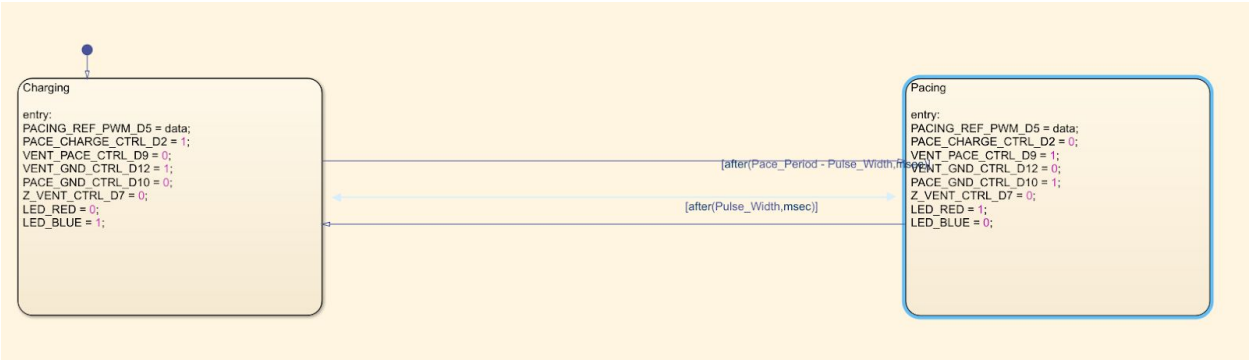


Figure #3: VOO Stateflow Chart

AAI

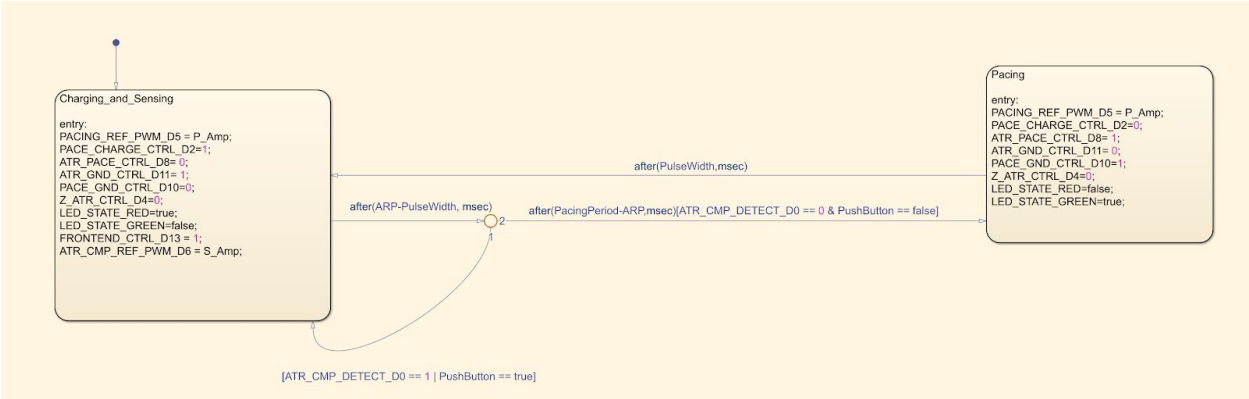


Figure #4: AAI Stateflow Chart

VVI

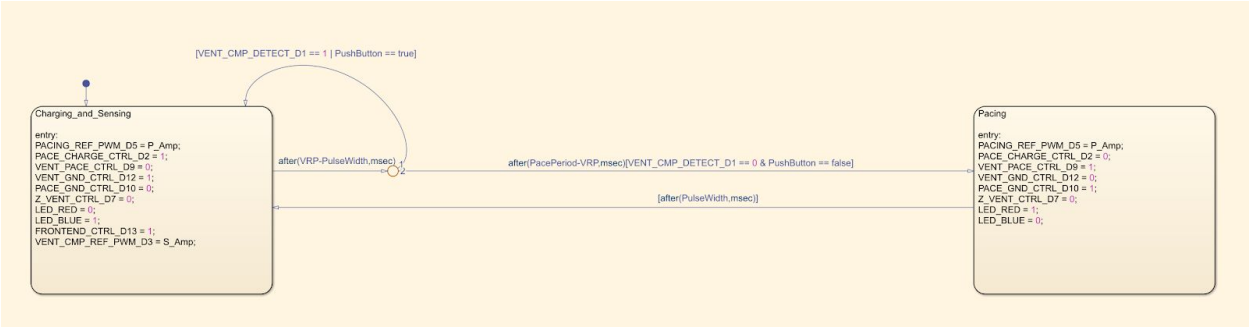


Figure #5: VVI Stateflow Chart

DOO

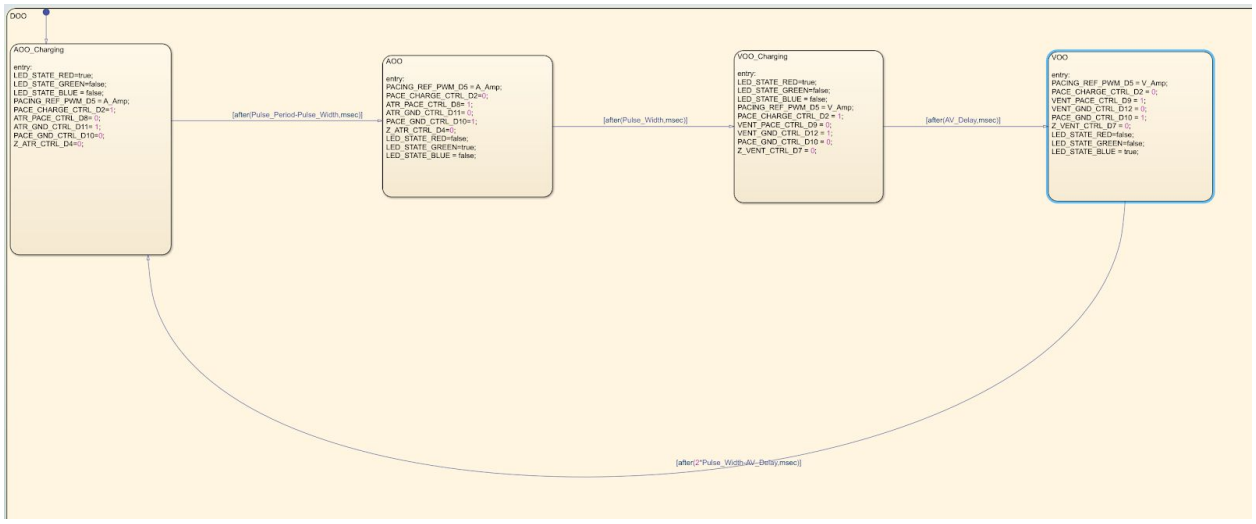


Figure #6: DOO Stateflow Chart

Rate Adaptive Pacing

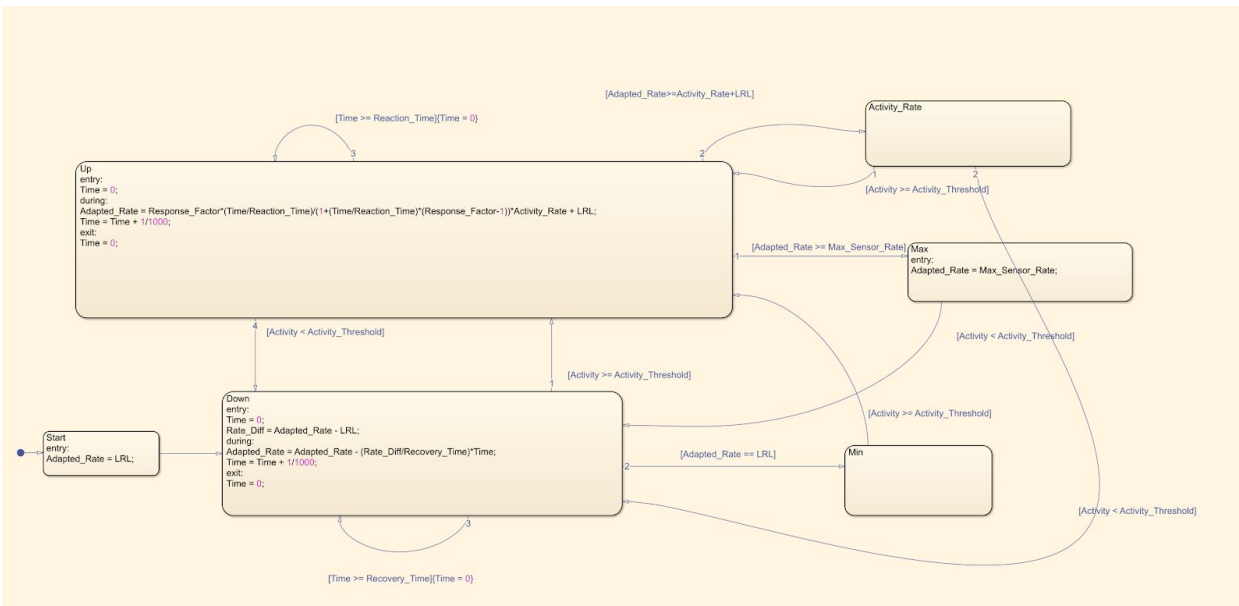


Figure #7: Rate Adaptive Pacing Stateflow Chart

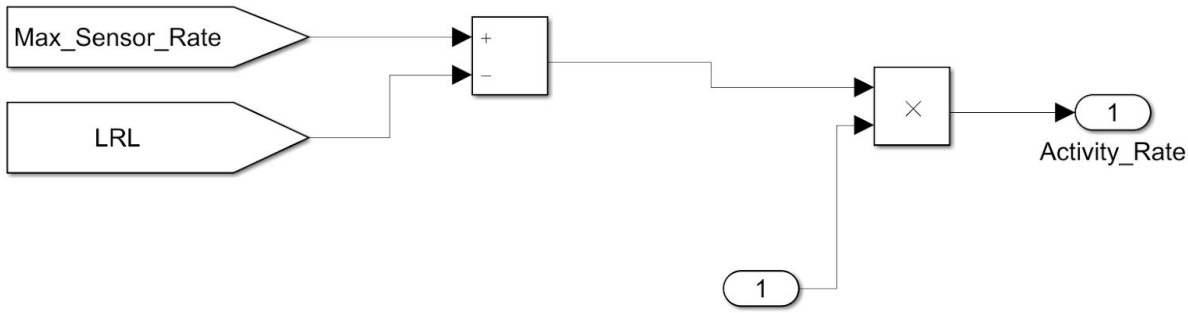


Figure #8: Rate Adaptive Pacing Activity Rate Calculation (Activity input, Activity_Rate output subsystem)

**Note that the import numbered 1 is the Activity quantity (inputted into the subsystem)

Merged Modes

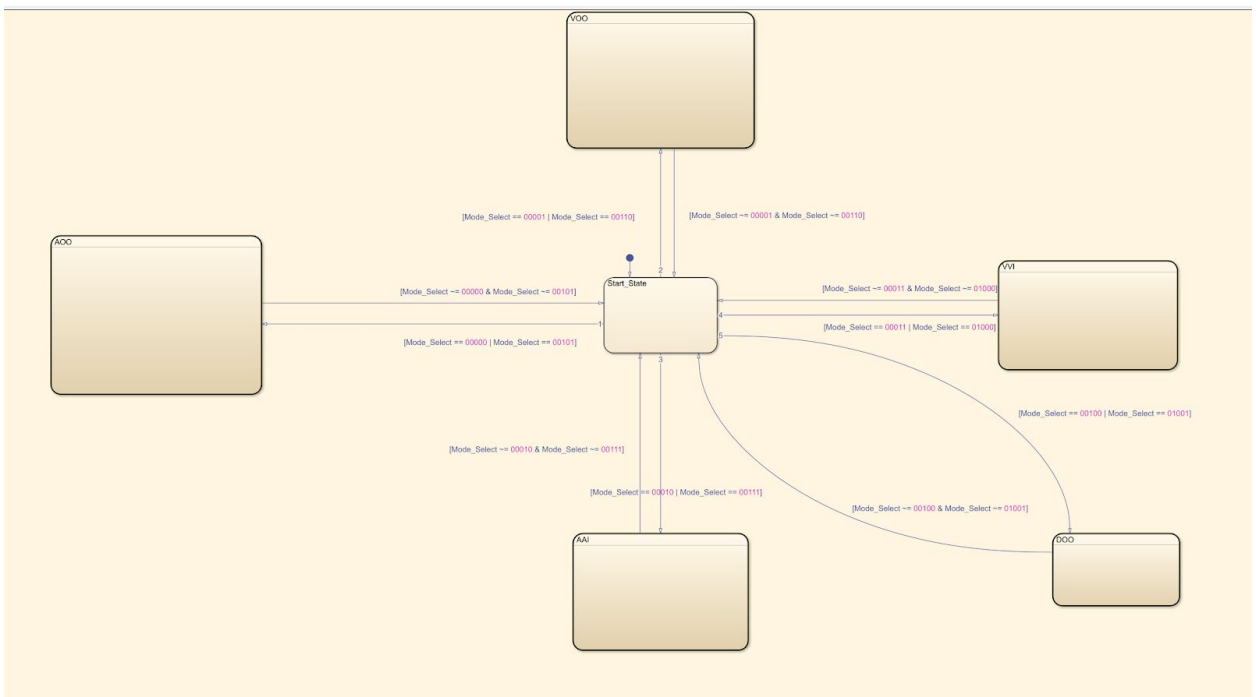


Figure #9: Merged Modes Stateflow Chart

State Diagram

AOO

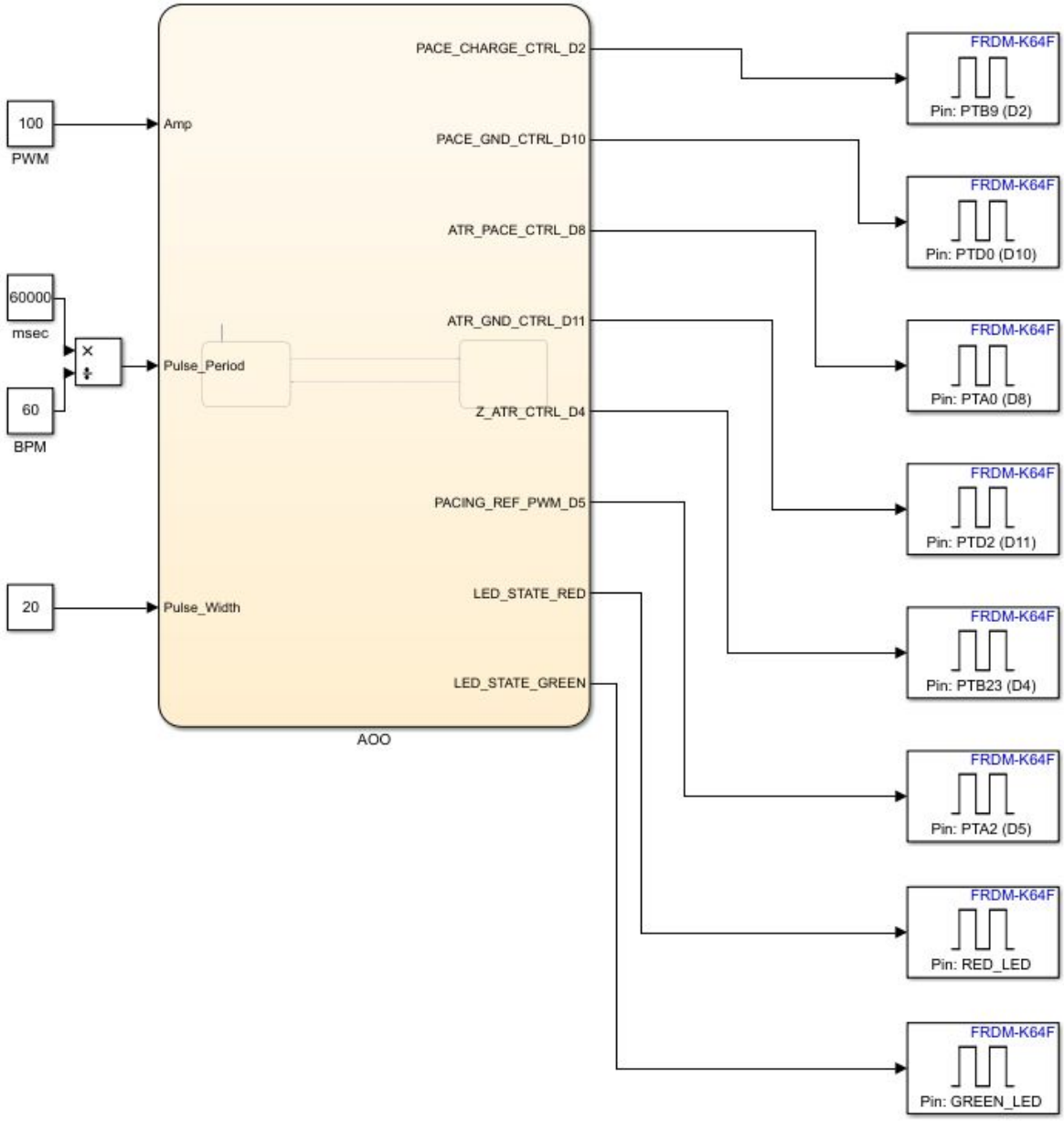


Figure #10: AOO State Diagram

VOO

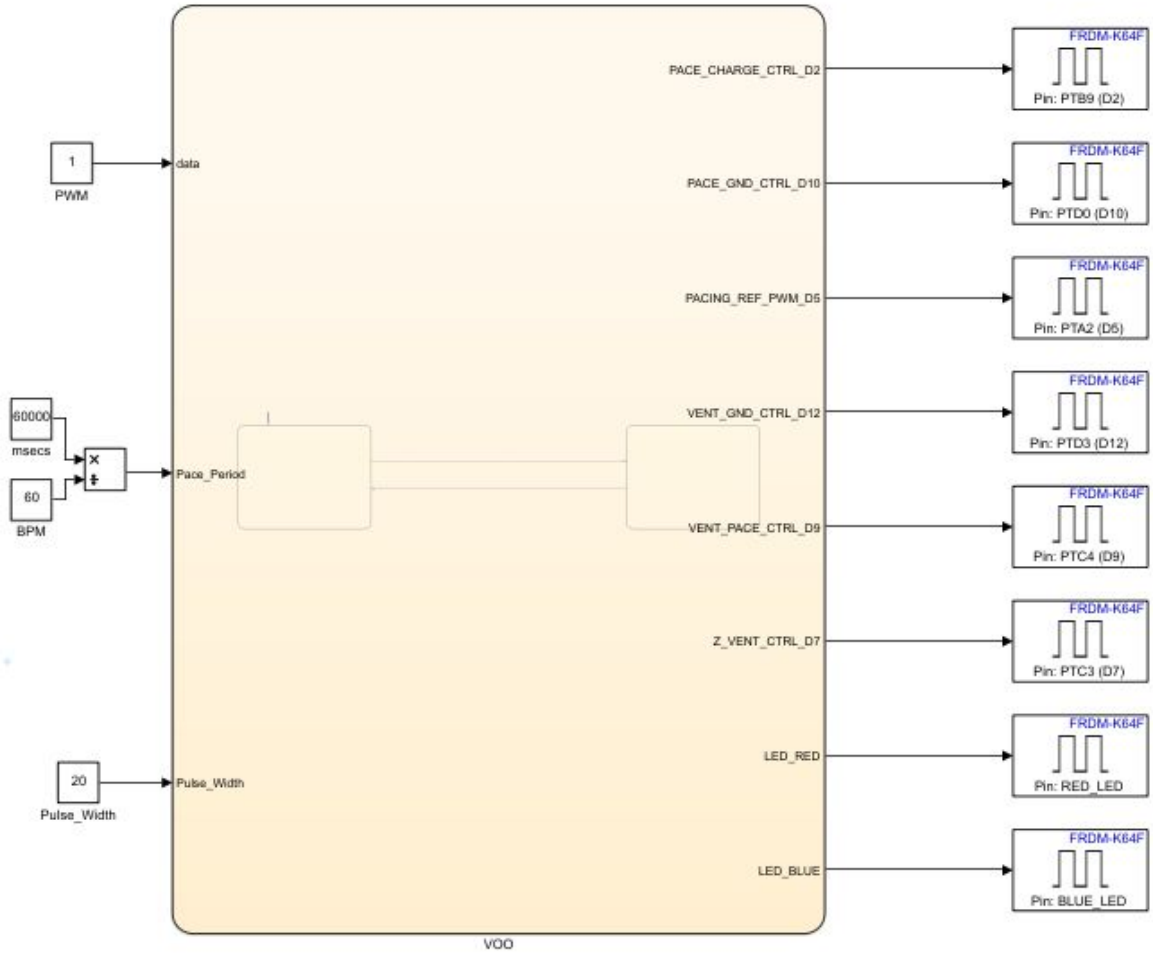


Figure #11: VOO State Diagram

AAI

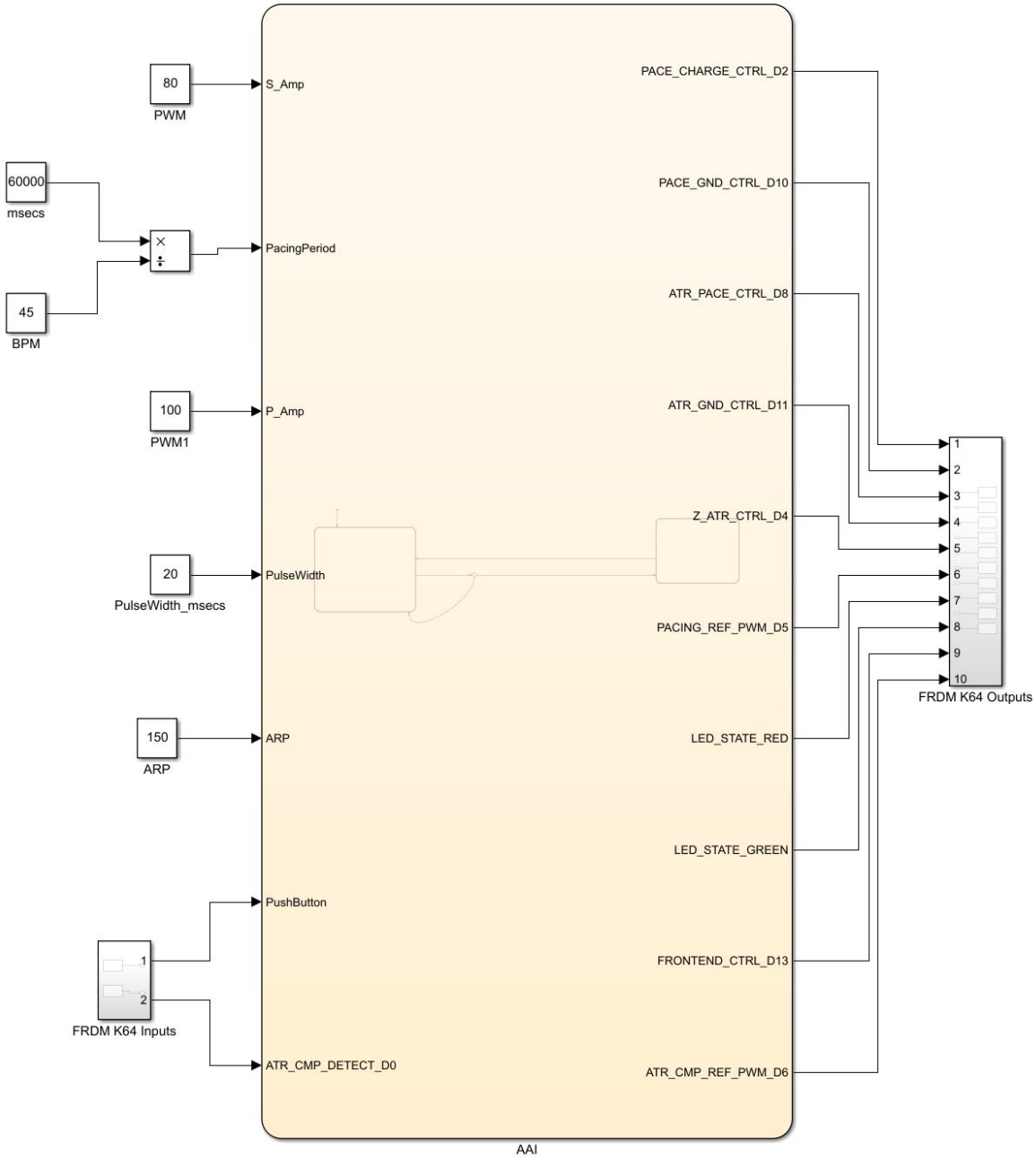


Figure #12: AAI State Diagram

VVI

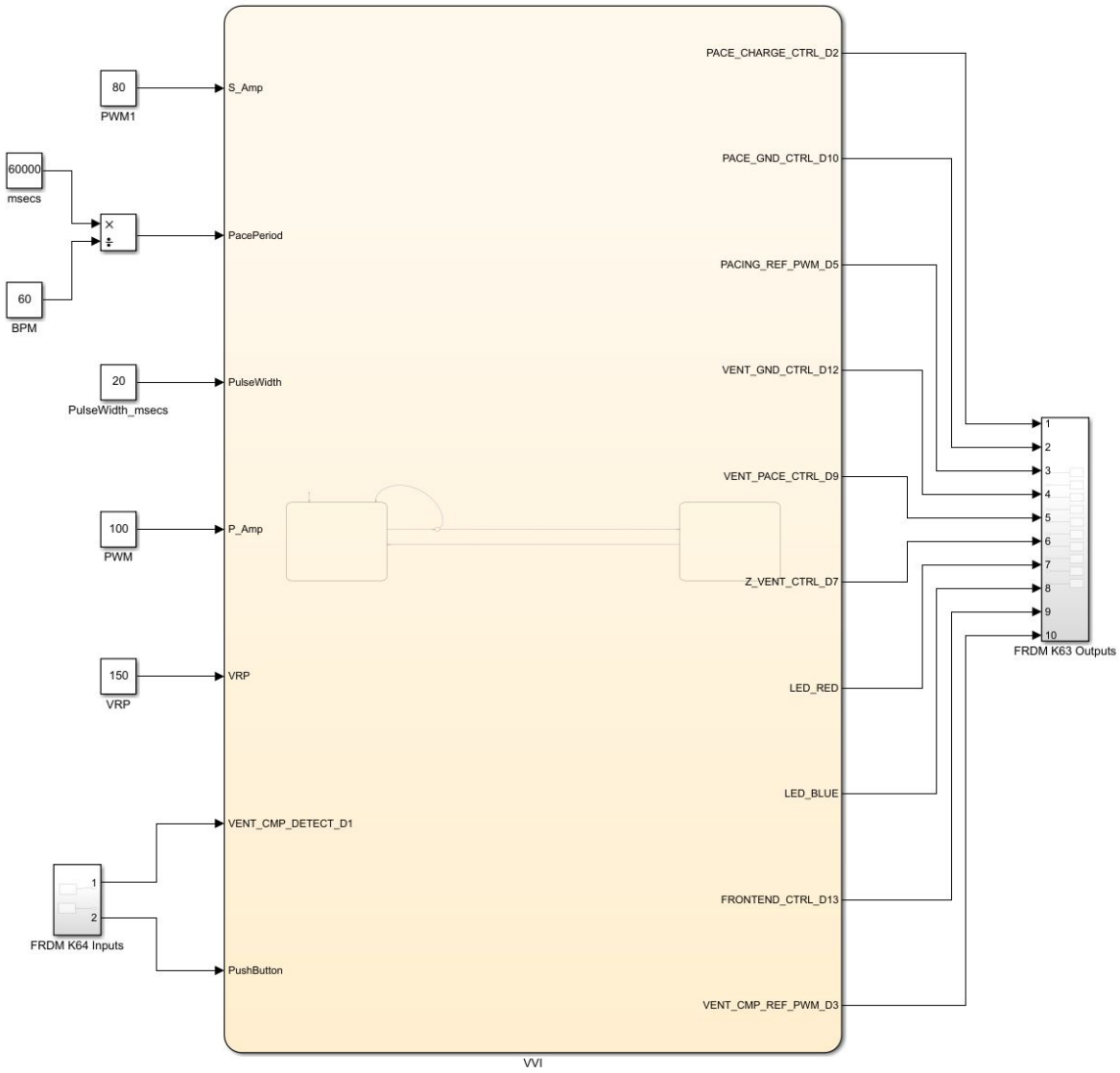


Figure #13: VVI State Diagram

DOO

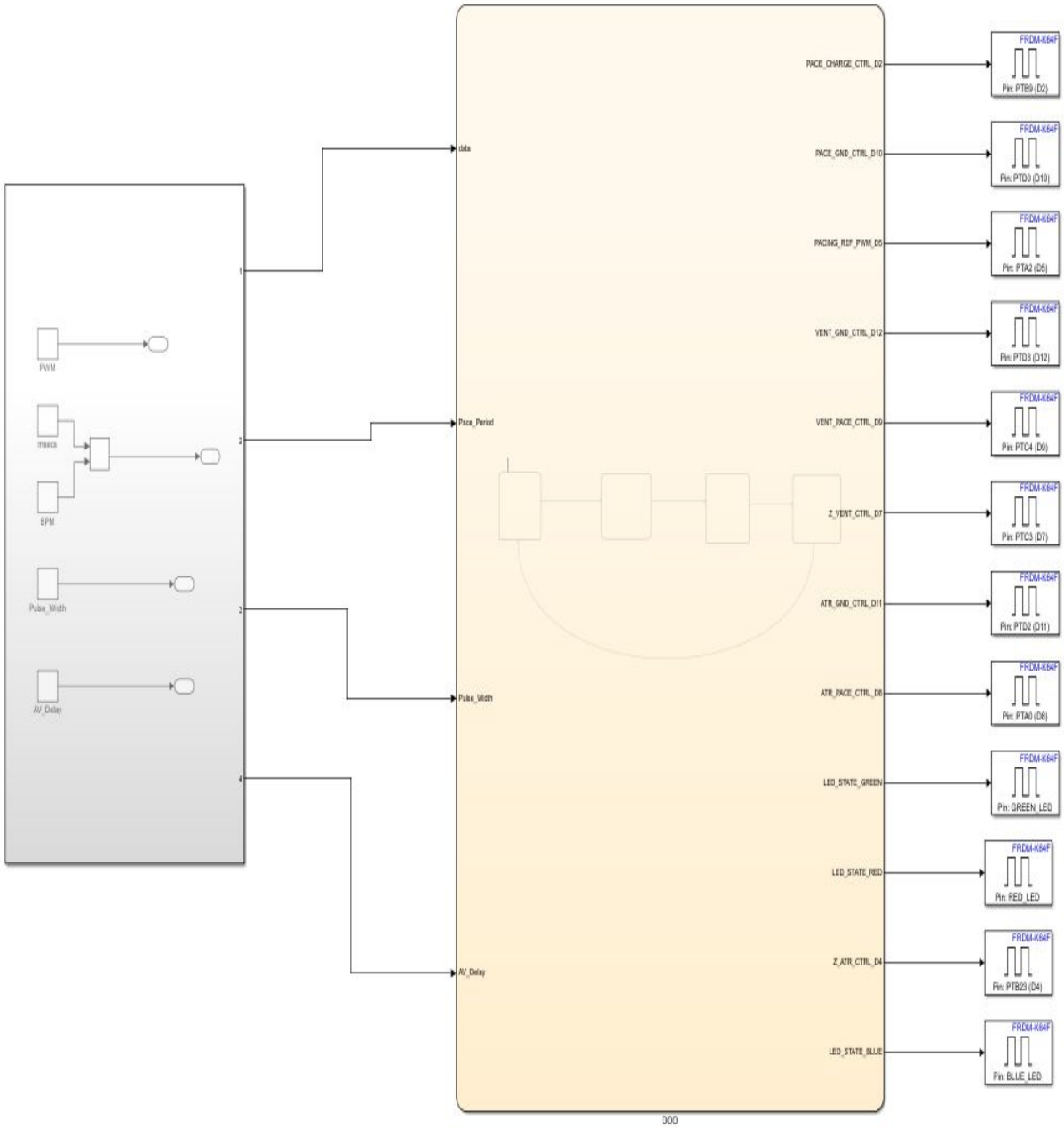


Figure #14: DOO State Diagram

Rate Adaptive Pacing

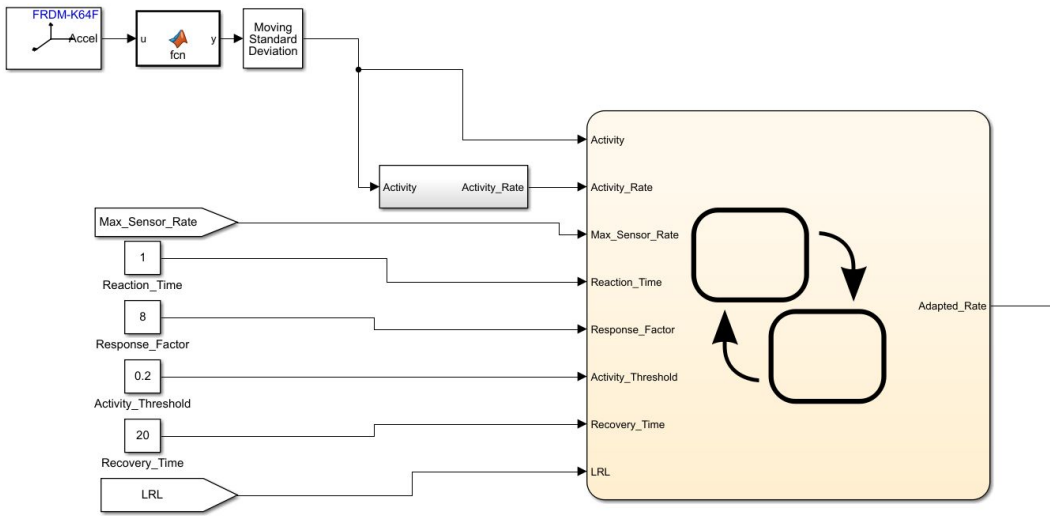


Figure #15: Rate Adaptive Pacing State Diagram

Merged Modes

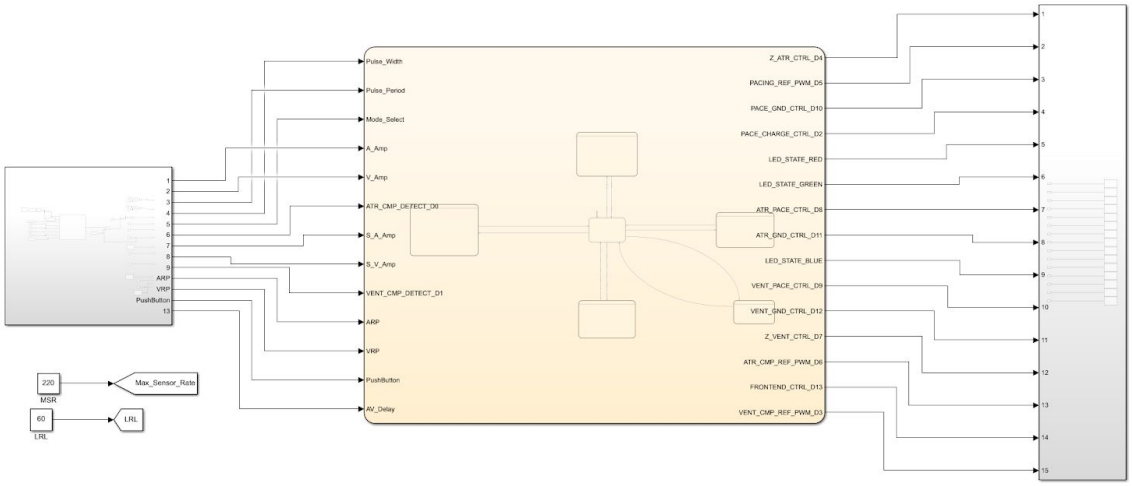


Figure 16: Merged Modes State Diagram

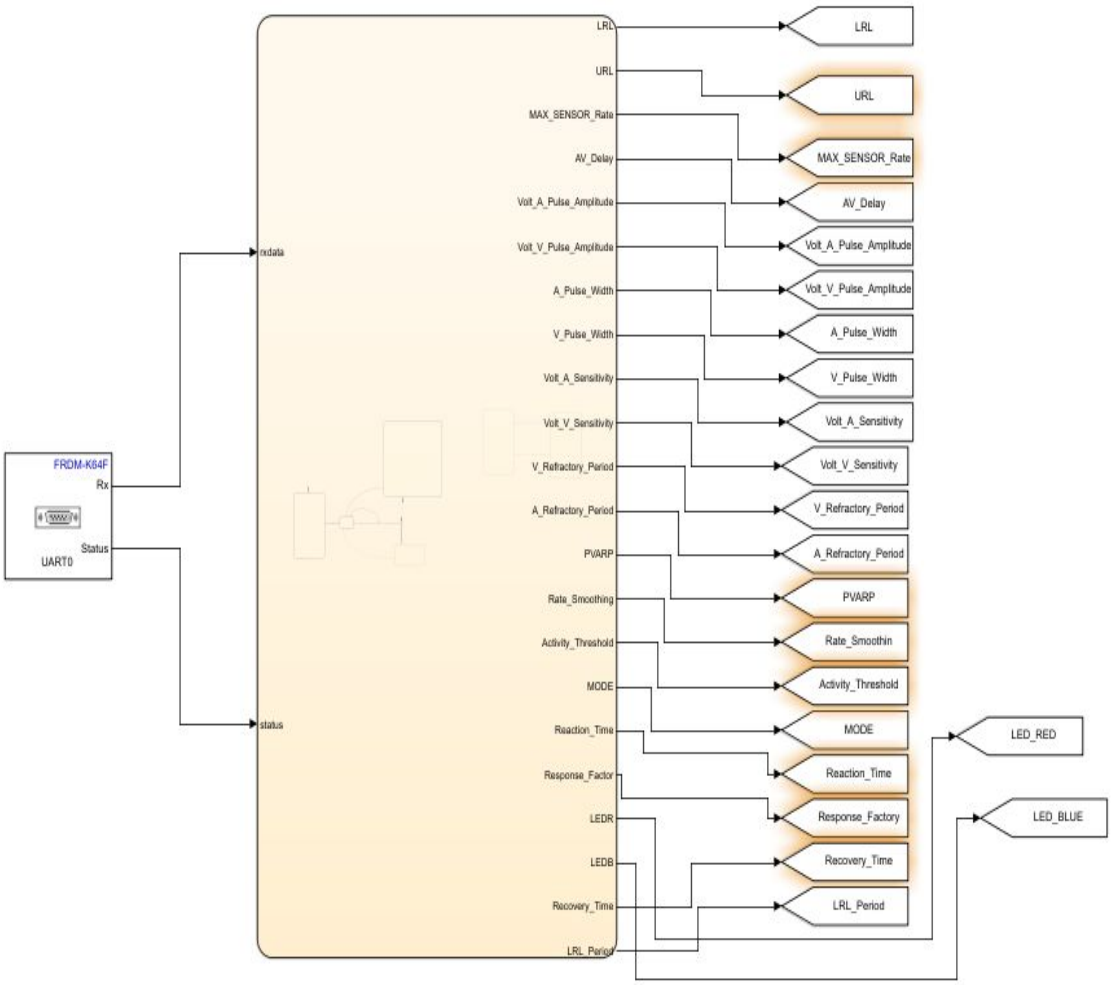


Figure #17: DCM to Pacemaker Serial Communication State Diagram

State Transition Table

AOO

Table #28: AOO State Transition Table

<table>
<tr><th>Current State</th><th>Variables to get to Next State</th><th>Output</th></tr>
<tr><td>Charging_and_Sensing (Default state)</td><td><table><tr><th>Variable</th><th>Value for Current State</th><th>Value for Next State</th></tr><tr><td>PACE_CHARGE_CTRL_D2</td><td>1</td><td>0</td></tr><tr><td>ATR_PACE_CTRL_D8</td><td>0</td><td>1</td></tr><tr><td>ATR_GND_CTRL_D11</td><td>1</td><td>0</td></tr><tr><td>PACE_GND_CTRL_D10</td><td>0</td><td>1</td></tr><tr><td>Z_ATR_CTRL_D4</td><td>0</td><td>0</td></tr><tr><td>LED_STATE_RED</td><td>true</td><td>false</td></tr><tr><td>LED_STATE_GREEN</td><td>false</td><td>true</td></tr></table></td><td>Capacitor is charged (next state charge from capacitor is released to emit pulse)</td></tr>
<tr><td>Pacing</td><td><table><tr><th>Variable</th><th>Value for Current State</th><th>Value for Next State</th></tr><tr><td>PACE_CHARGE_CTRL_D2</td><td>0</td><td>1</td></tr><tr><td>ATR_PACE_CTRL_D8</td><td>1</td><td>0</td></tr><tr><td>ATR_GND_CTRL_D11</td><td>0</td><td>1</td></tr><tr><td>PACE_GND_CTRL_D10</td><td>1</td><td>0</td></tr><tr><td>Z_ATR_CTRL_D4</td><td>0</td><td>0</td></tr><tr><td>LED_STATE_RED</td><td>false</td><td>true</td></tr></table></td><td>Pulse is emitted (once the charge is released, it returns back to charging state)</td></tr>
</table>

	LED_STATE_GREEN	true	false

VOO

Table #29: VOO State Transition Table

Current State	Variables to get to Next State			Output
Charging_and_Sensing (Default state)	Variable	Value for Current State	Value for Next State	Capacitor is charged (next state charge from capacitor is released to emit pulse)
	PACE_CHARGE_CTRL_D2	1	0	
	VENT_PACE_CTRL_D9	0	1	
	VENT_GND_CTRL_D12	1	0	
	PACE_GND_CTRL_D10	0	1	
	Z_VENT_CTRL_D7	0	0	
	LED_RED	false	true	
	LED_BLUE	true	false	
Pacing	Variable	Value for Current State	Value for Next State	Pulse is emitted (once the charge is released, it returns back to charging state)
	PACE_CHARGE_CTRL_D2	0	1	
	VENT_PACE_CTRL_D9	1	0	
	VENT_GND_CTRL_D12	0	1	
	PACE_GND_CTRL_D10	1	0	

Z_VENT_CTRL_D7	0	0	
LED_RED	false	true	
LED_BLUE	true	false	

AAI

Table #30: AAI State Transition Table

Current State	Variables to get to Next State			Output
Charging_and_Sensing (Default state)	Variable	Value for Current State	Value for Next State	Charging the Capacitor
	PACE_CHARGE_CTRL_D2	1	0	
	ATR_PACE_CTRL_D8	0	1	
	ATR_GND_CTRL_D11	1	0	
	PACE_GND_CTRL_D10	0	1	
	Z_ATR_CTRL_D4	0	0	
	ATR_CMP_DETECT_D0	Will only move to the next state (pacing) if it remains 0 after PacingPeriod-Pulse Width amount of time (in msec)		
	PushButton	Will only move to the next state (pacing) if is still false after PacingPeriod-Pulse Width amount of time (in msec)		

<table>
<tr><td></td><td></td><td></td></tr>
<tr><td>Pacing</td><td>
<table>
<tr><th>Variable</th><th>Value for Current State</th><th>Value for Next State</th></tr>
<tr><td>PACE_CHARGE_CTRL_D2</td><td>0</td><td>1</td></tr>
<tr><td>ATR_PACE_CTRL_D8</td><td>1</td><td>0</td></tr>
<tr><td>ATR_GND_CTRL_D11</td><td>0</td><td>1</td></tr>
<tr><td>PACE_GND_CTRL_D10</td><td>1</td><td>0</td></tr>
<tr><td>Z_ATR_CTRL_D4</td><td>0</td><td>0</td></tr>
</table>
</td><td>Setting the Pace

(next state is charging_and_sensing)</td></tr>
</table>

These are the only two states and they will move back and forth between then if the conditions are met.

VVI

Table #31: VVI State Transition Table

<table>
<tr><th>Current State</th><th>Variables to get to Next State</th><th>Output</th></tr>
<tr><td>Charging_and_Sensing (Default state)</td><td>
<table>
<tr><th>Variable</th><th>Value for Current State</th><th>Value for Next State</th></tr>
<tr><td>PACE_CHARGE_CTRL_D2</td><td>1</td><td>0</td></tr>
<tr><td>VENT_PACE_CTRL_D9</td><td>0</td><td>1</td></tr>
<tr><td>VENT_GND_CTRL_D12</td><td>1</td><td>0</td></tr>
<tr><td>PACE_GND_CTRL_D10</td><td>0</td><td>1</td></tr>
<tr><td>Z_VENT_CTRL_D13</td><td>0</td><td>0</td></tr>
<tr><td>VENT_CMP_DETECT_D0</td><td colspan="2">Will only move to the next state if it is</td></tr>
</table>
</td><td>Charging the Capacitor</td></tr>
</table>

<table>
<tr><td></td><td><table><tr><td></td><td>0</td></tr><tr><td>PushButton</td><td>Will only move to the next state if is still false after period of time</td></tr></table></td><td></td></tr>
<tr><td>Pacing</td><td><table><tr><th>Variable</th><th>Value for Current State</th><th>Value for Next State</th></tr><tr><td>PACE_CHARGE_CTRL_D2</td><td>1</td><td>0</td></tr><tr><td>VENT_PACE_CTRL_D9</td><td>0</td><td>1</td></tr><tr><td>VENT_GND_CTRL_D12</td><td>1</td><td>0</td></tr><tr><td>PACE_GND_CTRL_D10</td><td>0</td><td>1</td></tr><tr><td>Z_VENT_CTRL_D13</td><td>0</td><td>0</td></tr></table></td><td>Sending the Pace

(next state is charging_and_sensing)</td></tr>
</table>

These are the only two states and they will move back and forth between then if the conditions are met.

DOO

Table #32: DOO State Transition Table

<table>
<tr><th>Current State</th><th>Variables to get to Next State</th><th>Output</th></tr>
<tr><td>AOO_Charging</td><td><table><tr><th>Variable</th><th>Value for Current State</th><th>Value for Next State</th></tr><tr><td>PACE_CHARGE_CTRL_D2</td><td>1</td><td>0</td></tr><tr><td>PACE_GND_CTRL_D10</td><td>1</td><td>0</td></tr><tr><td>ATR_PACE_CTRL_D8</td><td>0</td><td>1</td></tr></table></td><td>Capacitor is charged and a pulse will be emitted to the atrium in the next state.</td></tr>
</table>

<table>
<tr><td></td><td><table><tr><td>VENT_PACE_CTRL_D9</td><td>0</td><td>0</td></tr><tr><td>ATR_GND_CTRL_D11</td><td>1</td><td>0</td></tr><tr><td>VENT_GND_CTRL_D12</td><td>0</td><td>0</td></tr></table></td><td></td></tr>
<tr><td>AOO</td><td><table><tr><th>Variable</th><th>Value for Current State</th><th>Value for Next State</th></tr><tr><td>PACE_CHARGE_CTRL_D2</td><td>0</td><td>1</td></tr><tr><td>PACE_GND_CTRL_D10</td><td>0</td><td>1</td></tr><tr><td>ATR_PACE_CTRL_D8</td><td>1</td><td>0</td></tr><tr><td>VENT_PACE_CTRL_D9</td><td>0</td><td>0</td></tr><tr><td>ATR_GND_CTRL_D11</td><td>0</td><td>0</td></tr><tr><td>VENT_GND_CTRL_D12</td><td>0</td><td>1</td></tr></table></td><td>Capacitor is charged, and a pulse is emitted to the atrium.</td></tr>
<tr><td>VOO_Charging</td><td><table><tr><th>Variable</th><th>Value for Current State</th><th>Value for Next State</th></tr><tr><td>PACE_CHARGE_CTRL_D2</td><td>1</td><td>0</td></tr><tr><td>PACE_GND_CTRL_D10</td><td>1</td><td>1</td></tr><tr><td>ATR_PACE_CTRL_D8</td><td>0</td><td>0</td></tr><tr><td>VENT_PACE_CTRL_D9</td><td>0</td><td>1</td></tr><tr><td>ATR_GND_CTRL_D11</td><td>0</td><td>0</td></tr><tr><td>VENT_GND_CTRL_D12</td><td>1</td><td>0</td></tr></table></td><td>Capacitor is charged and a pulse will be emitted to the ventricle in the next state.</td></tr>
<tr><td>VOO</td><td></td><td>Capacitor is charged, and a</td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><th>Variable</th><th>Value for Current State</th><th>Value for Next State</th></tr>
<tr><td>PACE_CHARGE_CTRL_D2</td><td>0</td><td>1</td></tr>
<tr><td>PACE_GND_CTRL_D10</td><td>1</td><td>1</td></tr>
<tr><td>ATR_PACE_CTRL_D8</td><td>0</td><td>0</td></tr>
<tr><td>VENT_PACE_CTRL_D9</td><td>1</td><td>0</td></tr>
<tr><td>ATR_GND_CTRL_D11</td><td>0</td><td>1</td></tr>
<tr><td>VENT_GND_CTRL_D12</td><td>0</td><td>0</td></tr>
</table>
</td><td>pulse is emitted to the ventricle.</td></tr>
</table>

Rate Adaptive Pacing

Table #33: Rate Adaptive Pacing State Transition Table

<table>
<tr><th>Current State</th><th>Variables to get to Next State</th><th>Output State</th></tr>
<tr><td>Start</td><td>The Start state always transitions to the Down state</td><td>Down</td></tr>
<tr><td rowspan="3">Down</td><td>
<table>
<tr><th>Variable</th><th>Value for Current State</th><th>Value for Next State</th></tr>
<tr><td>Adapted_Rate</td><td>Adapted_Rate - (Rate_Diff / Recovery_ Time)*Time</td><td>Response_Factor * (Time / Reaction_Time) / (1 + abs(Time/Reaction_Time) * (Response_Factor-1)) * Activity_Rate + LRL</td></tr>
</table>
Transition when: Activity >= Activity_Threshold</td><td>Up</td></tr>
<tr><td>Transition when: Adapted_Rate == LRL</td><td>Min</td></tr>
<tr><td>No Change</td><td>Down</td></tr>
</table>

<table>
<tr><td></td><td>Occurs when: Time >= Recovery_Time | (~[Activity >= Activity_Threshold] & ~[Adapted_Rate == LRL])</td><td></td></tr>
<tr><td rowspan="4">Up</td><td><table><tr><th>Variable</th><th>Value for Current State</th><th>Value for Next State</th></tr><tr><td>Adapted_Rate</td><td>Response_Factor * (Time / Reaction_Time) / (1 + abs(Time/Reaction_Time) * (Response_Factor-1)) * Activity_Rate + LRL</td><td>Max_Sensor_Rate</td></tr></table>Transition when: Adapted_Rate >= Max_Sensor_Rate</td><td>Max</td></tr>
<tr><td>Transition when: Adapted_Rate >= Activity_Rate + LRL</td><td>Activity_Rate</td></tr>
<tr><td>No Change
Transition when: Time >= Reaction_Time | (~[Adapted_Rate >= Max_Sensor_Rate] & ~[Adapted_Rate >= Activity_Rate + LRL] & ~[Activity < Activity_Threshold])</td><td>Up</td></tr>
<tr><td><table><tr><th>Variable</th><th>Value for Current State</th><th>Value for Next State</th></tr><tr><td>Adapted_Rate</td><td>Response_Factor * (Time / Reaction_Time) / (1 + abs(Time/Reaction_Time) * (Response_Factor-1)) * Activity_Rate + LRL</td><td>Adapted_Rate - (Rate_Diff / Recovery_ Time)*Time</td></tr></table>Transition when: Activity < Activity_Threshold</td><td>Down</td></tr>
<tr><td rowspan="2">Activity_ Rate</td><td>Transition when: Activity >= Activity_Threshold</td><td>Up</td></tr>
<tr><td>Transition when: Activity < Activity_Threshold</td><td>Down</td></tr>
<tr><td>Max</td><td><table><tr><th>Variable</th><th>Value for Current State</th><th>Value for Next State</th></tr><tr><td>Adapted_Rate</td><td>Max_Sensor_Rate</td><td>Adapted_Rate - (Rate_Diff / Recovery_ Time)*Time</td></tr></table>Transition when: Activity < Activity_Threshold</td><td>Down</td></tr>
</table>

Min	Transition when: Activity >= Activity_Threshold	Up

Serial Communication

Table #34: State Transition for Serial Communication

Present State	Status_bit	Next State
STANDBY (default)	0	SET_PARAMS
STANDY (default)	32	ECHO_PARAMS
SET_PARAMS	*	STANDBY
ECHO_PARAMS	*	STANDBY

Simulink Stateflow Tabular Expression

AOO

Table #35: AOO Stateflow Tabular Expression

Source State	Event	Conditions	During Actions	Condition Actions	Exit Actions	Transition Actions	Destination State	Entry Actions
Chambers _Paced_ Charging	TRUE	after(Pulse_Period - Pulse_Width ,msec)	none	none	none	none	Pacing	PACING_REF_PWM_D5 = Amp; PACE_CHARGE_CTRL_D2=1; ATR_PACE_CTRL_D8= 0; ATR_GND_CTRL_D11= 1; PACE_GND_CTRL_D10=0; Z_ATR_CTRL_D4=0; LED_STATE_RED=true; LED_STATE_GREEN=false;
		~after(Pulse_ Period - Pulse_Width ,msec)	none	none	none	none	No Change- Chambers_ Paced_ Charging	Remain in present state until the condition is satisfied.
Pacing	TRUE	after(Pulse_ Width,msec)	none	none	none	none	Chamber_ Paced_ Charging	PACING_REF_PWM_D5 = Amp; PACE_CHARGE_CTRL_D2=0; ATR_PACE_CTRL_D8= 1; ATR_GND_CTRL_D11= 0; PACE_GND_CTRL_D10=1; Z_ATR_CTRL_D4=0; LED_STATE_RED=false; LED_STATE_GREEN=true;
		~after(Pulse_ Width,msec)	none	none	none	none	No Change- Pacing	Remain in present state until the condition is satisfied.

VOO

Table #36: VOO Stateflow Tabular Expression

Source State	Event	Conditions	During Actions	Condition Actions	Exit Actions	Transition Actions	Destination State	Entry Actions
Charging	TRUE	after(Pulse_Period - Pulse_Width ,msec)	none	none	none	none	Pacing	PACING_REF_PWM_D5 = data; PACE_CHARGE_CTRL_D2 = 1; VENT_PACE_CTRL_D9 = 0; VENT_GND_CTRL_D12 = 1; PACE_GND_CTRL_D10 = 0; Z_VENT_CTRL_D7 = 0; LED_RED = false; LED_BLUE = true;
		~after(Pulse_Period - Pulse_Width ,msec)	none	none	none	none	No Change-Charging	Remain in present state until the condition is satisfied.
Pacing	TRUE	after(Pulse_Width,msec)	none	none	none	none	Charging	PACING_REF_PWM_D5 = data; PACE_CHARGE_CTRL_D2 = 0; VENT_PACE_CTRL_D9 = 1; VENT_GND_CTRL_D12 = 0; PACE_GND_CTRL_D10 = 1; Z_VENT_CTRL_D7 = 0; LED_RED = true; LED_BLUE = false;
		~after(Pulse_Width,msec)	none	none	none	none	No Change-Pacing	Remain in present state until the condition is satisfied.

AAI

Table #37: AAI Stateflow Tabular Expression

Source State	Event	Condition		During Actions	Condition Actions	Exit Actions	Transition Actions	Destination State	Entry Actions
State == Charging_and_ Sensing	TRUE	after(ARP-PulseWidth,msec)	ATR_CMP_DETECT_ D0 == 1 OR PushButton == true	none	none	none	none	Chargin_and_ Sensing	PACING_REF_PWM_D5 = P_Amp; PACE_CHARGE_CTRL_D2=1; ATR_PACE_CTRL_D8= 0; ATR_GND_CTRL_D11= 1; PACE_GND_CTRL_D10=0; Z_ATR_CTRL_D4=0; LED_STATE_RED=true; LED_STATE_GREEN=false; FRONTEND_CTRL_D13 = 1; ATR_CMP_REF_PWM_D6 = S_Amp;
			after(PacingPeriod-ARP,msec) AND ATR_CMP_DETECT_ D0 == 0 AND PushButton == false]	none	none	none	none	Pacing	PACING_REF_PWM_D5 = P_Amp; PACE_CHARGE_CTRL_D2=0; ATR_PACE_CTRL_D8= 1; ATR_GND_CTRL_D11= 0; PACE_GND_CTRL_D10=1; Z_ATR_CTRL_D4=0; LED_STATE_RED=false; LED_STATE_GREEN=true;
		~after(ARP-PulseWidth,msec)		none	none	none	none	No change -- Charging_and _Sensing	[have already occurred]
State == Pacing	TRUE	after(PulseWidth, msec)		none	none	none	none	Charging_and _Sensing	PACING_REF_PWM_D5 = P_Amp; PACE_CHARGE_CTRL_D2=1; ATR_PACE_CTRL_D8= 0; ATR_GND_CTRL_D11= 1; PACE_GND_CTRL_D10=0; Z_ATR_CTRL_D4=0; LED_STATE_RED=true; LED_STATE_GREEN=false; FRONTEND_CTRL_D13 = 1; ATR_CMP_REF_PWM_D6 = S_Amp;
		~after(PulseWidth,msec)		none	none	none	none	No change -- Pacing	[have already occurred]

VVI

Table #38: VVI Stateflow Tabular Expression

Source State	Event	Condition		During Actions	Condition Actions	Exit Actions	Transition Actions	Destination State	Entry Actions
State == Charging_and _Sensing	TRUE	after(VRP-PulseWidth,msec)	VENR_CMP_DETECT_D1 == 1 OR PushButton == true	none	none	none	none	Chargin_and_Sensing	PACING_REF_PWM_D5 = P_Amp; PACE_CHARGE_CTRL_D2=1; VENT_PACE_CTRL_D9= 0; VENT_GND_CTRL_D12= 1; PACE_GND_CTRL_D10=0; Z_VENT_CTRL_D7=0; LED_STATE_RED=0; LED_STATE_GREEN=1; FRONTEND_CTRL_D13 = 1; ATR_CMP_REF_PWM_D6 = S_Amp;
			after(PacingPeriod-ARP,msec) AND ATR_CMP_DETECT_D0 == 0 AND PushButton == false]	none	none	none	none	Pacing	PACING_REF_PWM_D5 = P_Amp; PACE_CHARGE_CTRL_D2=0; VENT_PACE_CTRL_D9= 1; VENT_GND_CTRL_D12= 0; PACE_GND_CTRL_D10=1; Z_VENT_CTRL_D7=0; LED_STATE_RED=1; LED_STATE_GREEN=0;
		~after(VRP-PulseWidth,msec)		none	none	none	none	No change -- Charging_and _Sensing	[have already occurred]
State == Pacing	TRUE	after(PulseWidth, msec)		none	none	none	none	Charging_and _Sensing	PACING_REF_PWM_D5 = P_Amp; PACE_CHARGE_CTRL_D2=1; VENT_PACE_CTRL_D9= 0; VENT_GND_CTRL_D12= 1; PACE_GND_CTRL_D10=0; Z_VENT_CTRL_D7=0; LED_STATE_RED=0; LED_STATE_GREEN=1; FRONTEND_CTRL_D13 = 1; ATR_CMP_REF_PWM_D6 = S_Amp;
		~after(PulseWidth,msec)		none	none	none	none	No change -- Pacing	[have already occurred]

DOO

Table #39: DOO Stateflow Tabular Expression

Source State	Event	Condition	During Actions	Condition Actions	Exit Actions	Transition Actions	Destination State	Entry Actions
State == Chambers _Paced_ Charging1	TRUE	after(Pace_Period-Pulse_Width-AV_Delay,msec)	none	none	none	none	AOO	PACING_REF_PWM_D5 = data; PACE_CHARGE_CTRL_D2=0; ATR_PACE_CTRL_D8= 1; ATR_GND_CTRL_D11= 0; PACE_GND_CTRL_D10=1; Z_ATR_CTRL_D4=0; LED_STATE_RED=false; LED_STATE_GREEN=true; LED_STATE_BLUE=false;
		~after(Pace_Period-Pulse_Width-AV_Delay,msec)	none	none	none	none	No change -- Chambers_Paced _Charging1	[have already occurred]
State == AOO	TRUE	after(Pulse_Width, msec)	none	none	none	none	Chambers_Paced _Charging2	PACING_REF_PWM_D5 = data; PACE_CHARGE_CTRL_D2=1; VENT_PACE_CTRL_D9= 0; VENT_GND_CTRL_D12= 1; PACE_GND_CTRL_D10=0; Z_VENT_CTRL_D7=0; LED_STATE_RED=true; LED_STATE_GREEN=false; LED_STATE_BLUE=false;
		~after(Pulse_Width,msec)	none	none	none	none	No change -- AOO	[have already occurred]
State == Chambers _Paced_ Charging2	TRUE	after(AV_Delay, msec)	none	none	none	none	VOO	PACING_REF_PWM_D5 = data; PACE_CHARGE_CTRL_D2=0; VENT_PACE_CTRL_D9= 1; VENT_GND_CTRL_D12= 0; PACE_GND_CTRL_D10=1; Z_VENT_CTRL_D7=0; LED_STATE_RED=false; LED_STATE_GREEN=false; LED_STATE_BLUE=true;
		~after(AV_Delay, msec)	none	none	none	none	No change -- Chambers_Paced _Charging2	[have already occurred]
State == VOO	TRUE	after(Pace_Period-Pulse_Width, msec)	none	none	none	none	Chambers_Paced _Charging1	PACING_REF_PWM_D5 = data; PACE_CHARGE_CTRL_D2=1; ATR_PACE_CTRL_D8= 0; ATR_GND_CTRL_D11= 1; PACE_GND_CTRL_D10=0; Z_ATR_CTRL_D4=0; LED_STATE_RED=true; LED_STATE_GREEN=false; LED_STATE_BLUE=false;
		~after(Pace_Period-Pulse_Width, msec)	none	none	none	none	No change -- VOO	[have already occurred]

Rate Adaptive Pacing

Table #40: Rate Adaptive Pacing Stateflow Tabular Expression

Source State	Event	Condition	During Actions	Condition Actions	Exit Actions	Transition Actions	Destination State	Entry Actions
State == Start	TRUE		Adapted_Rate = Adapted_Rate - (Rate_Diff/Recovery_Time)*Time Time = Time + 1/1000	none	Time = 0;	none	Down	Time = 0; Rate_Diff = Adapted_Rate - LRL;
State == Down	TRUE	[Activity >= Activity_Threshold]	Adapted_Rate = Response_Factor*(Time/Reaction_Time)/(1+abs(Time/Reaction_Time)*(Response_Factor-1))*Activity_Rate+LRL;	none	Time = 0;	none	Up	Time = 0;
		[Adapted_Rate == LRL]	none	none	none	none	Min	none
		[Time >= Recovery_Time]	Adapted_Rate = Adapted_Rate - (Rate_Diff/Recovery_Time)*Time Time = Time + 1/1000	none	Time = 0;	none	Down	Time = 0; Rate_Diff = Adapted_Rate - LRL;
		~[Activity >= Activity_Threshold] & ~[Adapted_Rate == LRL] & ~[Time >= Recovery_Time]	none	none	none	none	No change -- Down	[have already occurred]
State == Up	TRUE	[Adapted_Rate >= Max_Sensor_Rate]	none	none	none	none	Max	Adapted_Rate = Max_Sensor_Rate;
		[Adapted_Rate >= Activity_Rate + LRL]	none	none	none	none	Activity_Rate	none
		[Time >= Reaction_Time]	Adapted_Rate = Response_Factor*(Time/Reaction_Time)/(1+abs(Time/Reaction_Time)*(Response_Factor-1))*Activity_Rate+LRL;	none	Time = 0;	none	Up	Time = 0;
		[Activity < Activity_Threshold]	Adapted_Rate = Adapted_Rate - (Rate_Diff/Recovery_Time)*Time Time = Time + 1/1000	none	Time = 0;	none	Down	Time = 0; Rate_Diff = Adapted_Rate - LRL;
		~[Adapted_Rate >= Max_Sensor_Rate] & ~[Adapted_Rate >= Activity_Rate + LRL] & ~[Time >= Reaction_Time] & ~[Activity < Activity_Threshold]	none	none	none	none	No change -- Up	[have already occurred]
State == Activity_Rate	TRUE	[Activity >= Activity_Threshold]	Adapted_Rate = Response_Factor*(Time/Reaction_Time)/(1+abs(Time/Reaction_Time)*(Response_Factor-1))*Activity_Rate+LRL;	none	Time = 0;	none	Up	Time = 0;
		[Activity < Activity_Threshold]	Adapted_Rate = Adapted_Rate - (Rate_Diff/Recovery_Time)*Time Time = Time + 1/1000	none	Time = 0;	none	Down	Time = 0; Rate_Diff = Adapted_Rate - LRL;
State == Max	TRUE	[Activity < Activity_Threshold]	Adapted_Rate = Adapted_Rate - (Rate_Diff/Recovery_Time)*Time Time = Time + 1/1000	none	Time = 0;	none	Down	Time = 0; Rate_Diff = Adapted_Rate - LRL;
		~[Activity < Activity_Threshold]	none	none	none	none	No change -- Max	[have already occurred]
State == Min	TRUE	[Activity >= Activity_Threshold]	Adapted_Rate = Response_Factor*(Time/Reaction_Time)/(1+abs(Time/Reaction_Time)*(Response_Factor-1))*Activity_Rate+LRL;	none	Time = 0;	none	Up	Time = 0;
		~[Activity >= Activity_Threshold]	none	none	none	none	No change -- Min	[have already occurred]

States Descriptions

Table #41: Simulink State Descriptions

State	Function
Charging_and_Sensing	Charges the capacitor and is continuously sensing when in this state.
Pacing	Discharges the capacitor, sets the pace.
AOO_Charging	Capacitor is charged and a pulse will be emitted to the atrium in the next state.
VOO_Charging	Capacitor is charged and a pulse will be emitted to the ventricle in the next state.
Start	Default, used to initialize the adapted rate
Down	Decreases from adapted rate to LRL
Up	Increases the adapted rate
Activity_Rate	Pauses activity at the Activity_Rate
Max	Stops the adapted rate at the MSR leads to Down to bring the adapted rate down
Min	Stops the adapted rate at the LRL leads to Up to bring the adapted rate up

Table #42: State Definitions

State Name	Description
STANDY (default)	No new data is available.
SET_PARAMS	New data has been received from the DCM and the parameters are being assigned new values.
ECHO_PARAMS	New data has been received from the pacemaker and is being sent to the DCM.

DCM

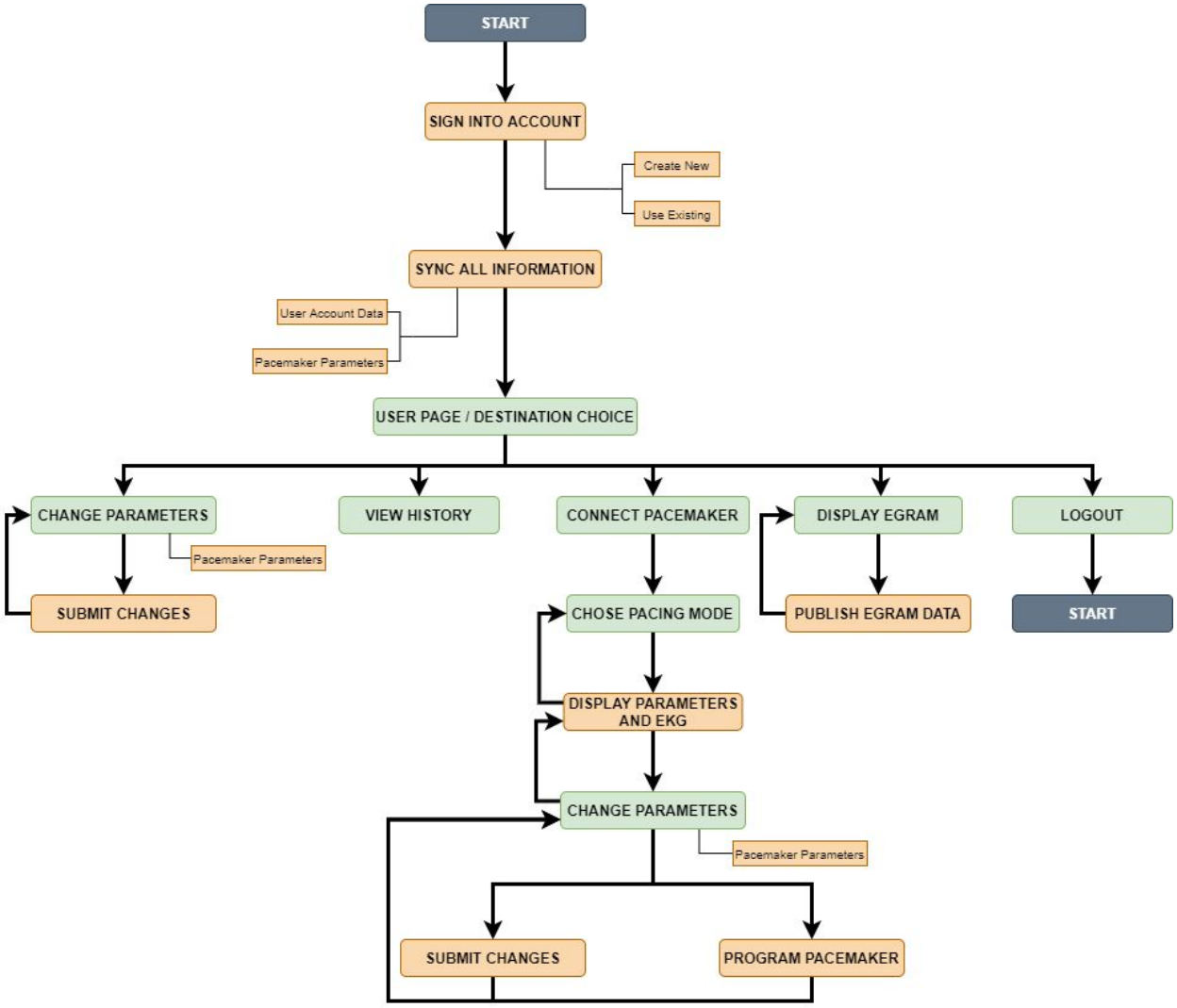


Figure #18: DCM State Transition Chart

Design Decisions

There was no change in information from the supplied information. Our design follows all laid out parameters. In the coming project alterations will be made in order to improve the design.

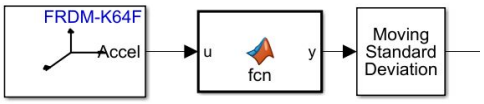
AOO, VOO, AAI & VVI

Table #43: AOO, VOO, AAI & VVI Design Decisions

Requirement being Changed	Resulting Change in Parameter + Assigned Values
PacePeriod	PacePeriod will be changed from a controlled quantity to a monitored variable, so that the input can be controlled from the user
BPM	Beats per minute have been restricted to be between 30 and 180. This is a restriction implemented due to heartview since these are the minimum and maximum BPM values.
PulseWidth	PulseWidth will be changed from a controlled quantity to a monitored variable, so that the input can be controlled from the user. The constant value is also restricted to be between 0ms and 20ms, so no quantity larger can be entered, otherwise design does not run.
P_Amp	P_Amp will be changed from a controlled quantity to a monitored variable, so that the amplitude of the pace can be controlled from the user
S_Amp	S_Amp will be changed from a controlled quantity to a monitored variable, so that the amplitude of the pace can be controlled from the user
Amp	PWM will be changed from a controlled quantity in AOO and VOO to manipulate the amplitude of the pace.
ARP	ARP will be changed from a controlled quantity to a monitored variable, so that the time interval after an atrial event can be controlled from the user
VRP	ARP will be changed from a controlled quantity to a monitored variable, so that the time interval after a ventricular event can be controlled from the user

The design decisions that were made were around the implementation of the rate adaptive modes. To implement this functionality, multiple formulas and parameters were designed, and the choice of these parameters and formulas will be explained here:

- Accelerometer Signal Filtering: Since the accelerometer outputs a 1x3 vector of the acceleration on each axis (x, y, z), it was connected to a Matlab function which takes the vector sum of each of these accelerations to produce a singular total acceleration. This is then connected to a Moving Standard Deviation with a window of 10 seconds so momentary jerky movements do not affect the accelerometer reading.



- Activity Rate Calculation: The activity rate is the desired additional heart rate the pacemaker should add to the Lower Rate Limit due to acceleration detected from the accelerometer and is calculated by: (MSR-LRL)*(AccelerometerActivity)
    - This was designed so that the increased rate due to acceleration would be proportional to the amount of activity itself and the difference between the MSR and the LRL (i.e., the greater the difference between MSR and LRL, the more wiggle room there is to shift, and so the heart rate increases accordingly)
- Increasing Adapted Rate Calculation: when accelerometer activity is greater than the threshold, the Lower Rate Limit must increase to the Activity Rate, with a slope inversely proportional to the reaction time and proportional to the response factor. The following is this calculation: Adapted_Rate = Response_Factor*(Time/Reaction_Time)/(1+abs(Time/Reaction_Time)*(Response_Factor-1))*Activity_Rate + LRL;
    - Where time is increased incrementally within the state from zero to the Reaction_Time
    - This was formulated so that the slope would increase with a greater Response Factor and lower Reaction Time, and so that once Time reaches Reaction Time, the formula simply becomes Activity Rate + LRL
- Decreasing Adapted Rate Calculation: when accelerometer activity is less than the threshold, the Adapted Rate must decrease back down to the LRL with a slope inversely proportional to the Recovery Time. The following is this calculation: Adapted_Rate = Adapted_Rate - (Rate_Diff/Recovery_Time)*Time;
    - Where Rate_Diff is the difference between the Adapted Rate and the LRL
    - This calculation is simply a linear function of a line from the Adapted Rate to the LRL, where the x-axis is time and the y-axis is heart rate. Since the Adapted Rate must decrease down to the LRL over the Recovery Time, the slope of the line becomes Rate_Diff/Recovery_Time

Merged_Simulink_Modes

- The accelerometer was integrated into the merged Simulink design
- The Simulink would change modes depending on the value of the *Mode_Select* variable which had a threshold of 00101
    - If the *Mode_Select* value is greater than or equal to 00101, then the accelerometer would be initiated and the values used for the respective pacing mode would be received from it
    - If the *Mode_Select* value is less than 00101, then the pacing modes will function as normal without the help of the accelerometer
    - Note that all modes enumerated greater than or equal to 00101 are rate adaptive pacing modes, all those below are not rate adaptive

Hardware Hiding

In order to implement hardware hiding, subsystems were made for the hardware inputs and outputs. These subsystems can be seen highlighted in the image below. These subsystems are essentially folders which contain all of the pins. They are named in consistently with the inputs and outputs from the board in order for the ease of understanding for the programmers, however the names could be altered as to further add to the hardware hiding.

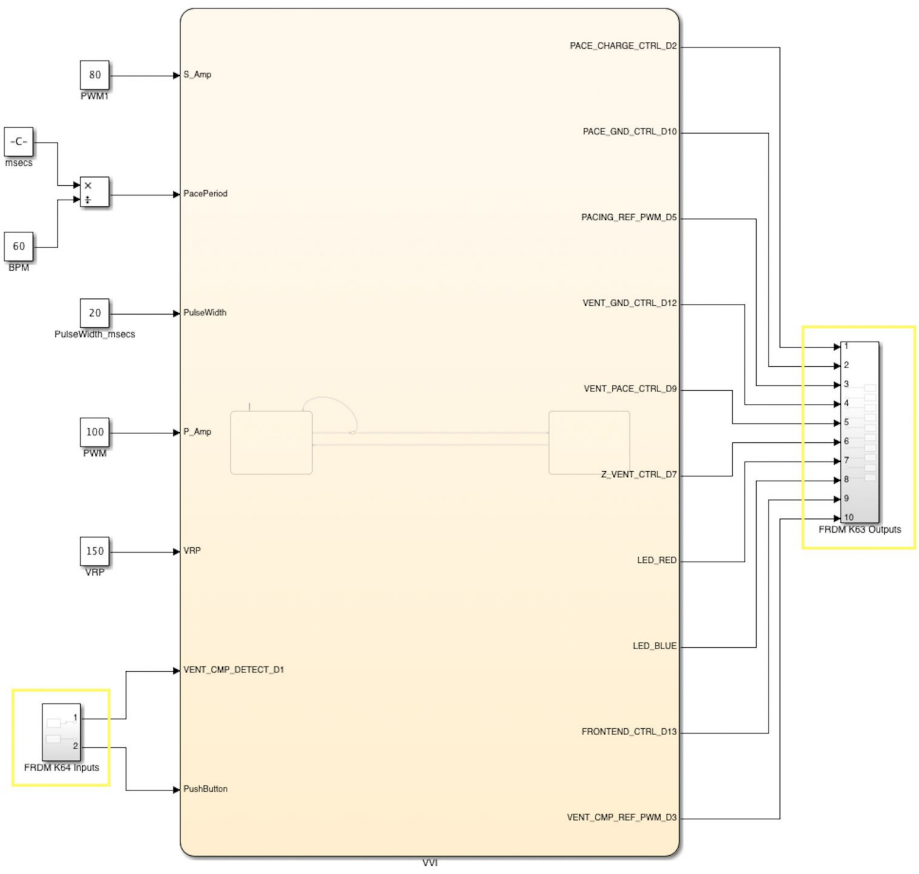


Figure #19: VVI State Diagram Highlighted for Hardware Hiding

Merged Modes

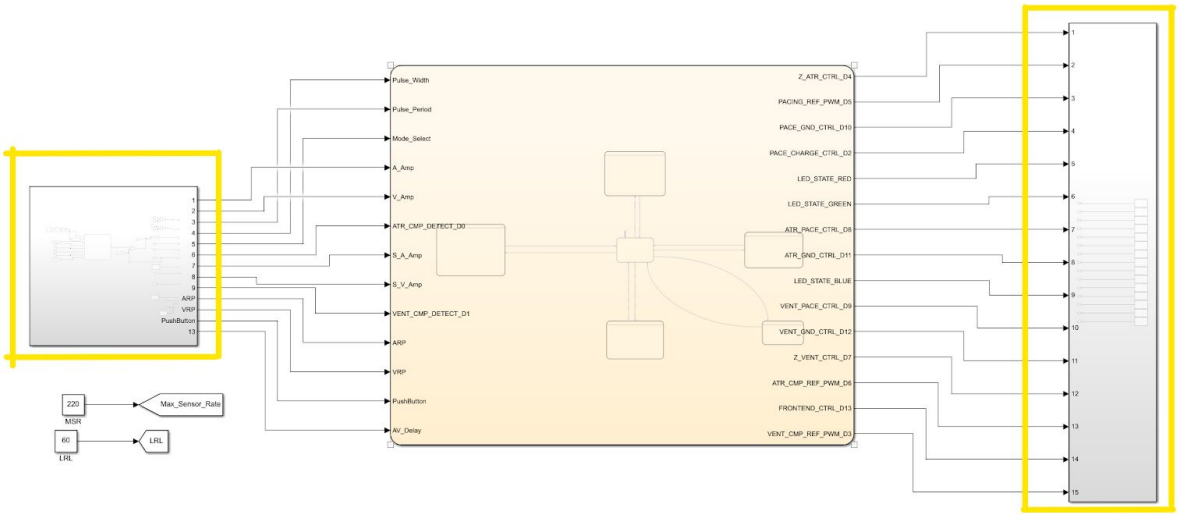
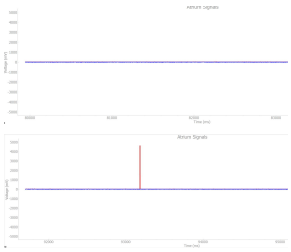
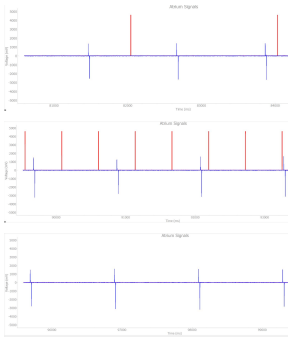
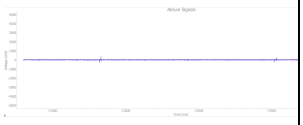
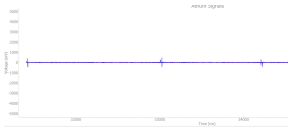
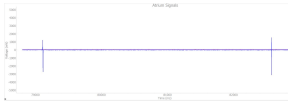


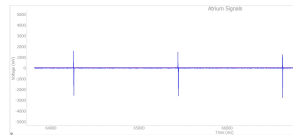
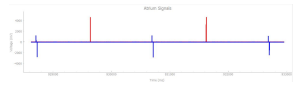
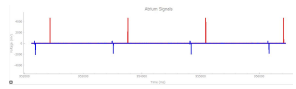
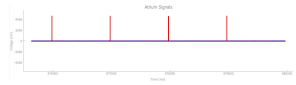
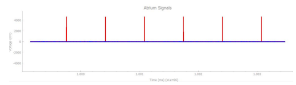
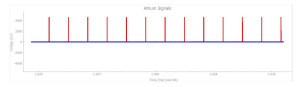
Figure 20: Merged Modes State Diagram Highlighted for Hardware Hiding

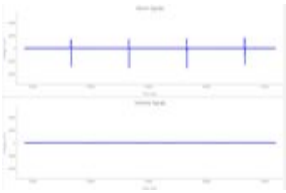
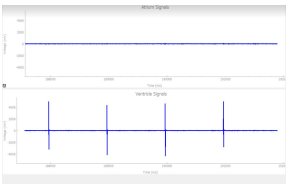
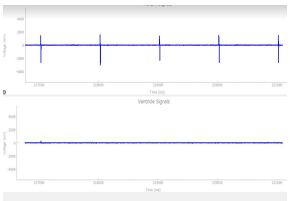
Testing

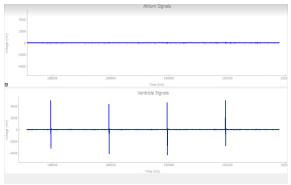
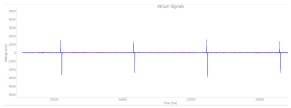
Simulink

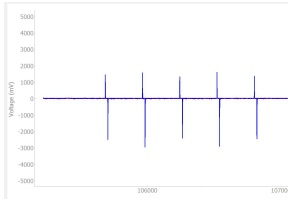
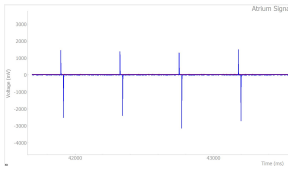
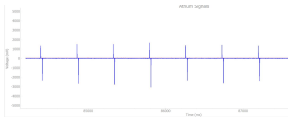
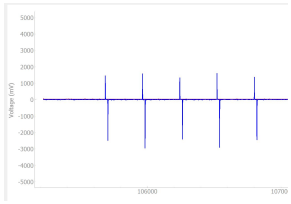
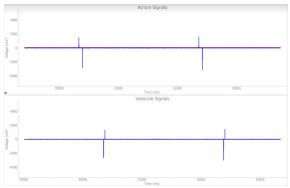
Table #44: Simulink Test Cases

Test Cases	Expected Outcome	Actual Outcome	Pass/Fail
Decrease S_Amp (S_Amp = 50)	Sensing threshold too low → pacemaker is too sensitive to signals from the heart, so will interpret even small signals as a pulse → will not send any paces		Pass
Increase S_Amp (S_Amp = 100)	Sensing threshold is too high → pacemaker will not register any electrical signal from the heart as a pulse → will continue to send paces no matter the heart's pulse		Pass
P_Amp = 10	Pace amplitude should be 90% (100-10) smaller than default		Pass
P_Amp = 50	Pace amplitude should be 50% (100-50) smaller than default		Pass
Push button many times quickly and consecutively	Inhibit pace until button is no longer being pressed & once pace period has passed (assuming no natural heartbeat)		Pass
Make PulseWidth longer than PacingPeriod	A pulse should not be emitted because it is not possible in reality. A restriction was put on	Pulse Width = 1500ms Matlab Diagnostics: "Inconsistent numeric values for parameter 'Value' in 'AOO/Constant': Quantized	Pass

	Pulse Width variable so that it can only be between 0 and 20ms.	parameter value (1500) is greater than maximum (20)”	
Shorten PulseWidth to 5 msec	Pulse width should become 5 ms (still within the boundary of 1-20 msec)		Pass
Vary BPM less than 30, greater than 180	Simulink should give an error since the BPM values not within 30 and 180 cannot be supported by heartview. Simulink prevents these settings from being registered to the pacemaker.	BPM = 20 Matlab Diagnostic: “Inconsistent numeric values for parameter 'Value' in 'AOO/BPM': Quantized parameter value (20) is less than minimum (30)” BPM = 190 “Inconsistent numeric values for parameter 'Value' in 'AOO/BPM': Quantized parameter value (190) is greater than maximum (180)”	Pass
BPM (less than, equal to and greater than heartbeat): - 30 - 45 - 60 - 90 - 180	Pacemaker is set to 60BPM. When the heart rate is less than 60BPM, pulses will be emitted by the pacemaker. When the heart rate is greater than 60BPM, pulses will be inhibited.	BPM = 30  BPM = 45  BPM = 60  BPM = 90  BPM = 180 	Pass
The following data packet is received from DCM: MODE: 1 MODE: 2 MODE: 3 MODE: 4	Depending on the value of MODE, different pacing modes are implemented. 1 - AOO 2 - VOO 3 - VVI	The following pacing modes are implemented: 1 - AOO 2 - VOO 3 - VVI 4 - AAI	PASS

MODE: 5 MODE: 6 MODE: 7 MODE: 8 MODE: 9 MODE 10 [other parameters]	4 - AAI 5 - DOO 6 - 7 - 8 - 9 - 10 -	5 - DOO 6 - 7 - 8 - 9 - 10 -	
The following data packet is received from the DCM: [… MODE: 1 Pulse_Amplitude: 3.5V Pulse_Width: 5ms LRL: 60bpm ...]	The difference in the pacing is 995ms. The width is 5ms. Signal amplitude is to be 3.5V	Width: 5ms Amplitude: 3.5V Difference in pacing: ~995ms 	PASS
The following data packet is received from the DCM: … MODE: 2 Pulse_Amplitude: 3.5V Pulse_Width: 5ms LRL: 50bpm ...]	The difference in the pacing is 1200ms. The width is 5ms. Signal amplitude is to be 2.5V	Width: 5ms Amplitude: 3.5V Difference in pacing: ~1200ms 	PASS
[… MODE: 1 Pulse_Amplitude: 3V Pulse_Width: 6ms LRL: 60bpm ...]	The difference in the pacing is 995ms. The width is 6ms. Signal amplitude is to be 3V	Width: 6ms Amplitude: 3V Difference in pacing: ~995ms 	PASS
The following data packet is received from the DCM: [… MODE: 2 Pulse_Amplitude: 3V	The difference in the pacing is 995ms. The width is 6ms. Signal amplitude is to be 3V	Width: 6ms Amplitude: 3V Difference in pacing: ~995ms	PASS

Pulse_Width: 6ms LRL: 60bpm ...]			
Rate Adaptive Pacing: Vary Activity_Threshold: (0, 0.2, 0.5, 0.8, 1)	Activity_Threshold: =0 -- accelerometer senses every movement and paces accordingly =1 -- accelerometer activity will most likely never surpass this threshold, therefore no rate adaptive pacing will occur 0 < Activity_Threshold < 1: surpassing activity threshold to observe rate adaptive pacing becomes progressively more challenging	Actual Outcome matches Expected Outcome -- Heartview screenshots cannot prove this	PASS
Vary Reaction_Time: (1 sec, 30 sec)	Time to reach Adapted_Rate should correspond with Reaction_Time	Actual Outcome matches Expected Outcome -- Heartview screenshots cannot prove this	PASS
Vary Recovery_Time: (1 sec, 30 sec, 1 min)	Time to reach LRL from Adapted_Rate should correspond with Recovery_Time	Actual Outcome matches Expected Outcome -- Heartview screenshots cannot prove this	PASS
Vary Response_Factor (1-16)	Time to reach Adapted_Rate should decrease with increasing Response_Factor	Actual Outcome matches Expected Outcome -- Heartview screenshots cannot prove this	PASS
Vary Maximum_Sensor_Rate (50, 100, 175, 220)	Adapted_Pace should be higher (given the same level of activity) with higher Maximum_Sensor_Rate	MSR = 50:  → MSR is less than LRL so adapted rate will stay at LRL MSR = 100:	PASS

		 MSR = 175: MSR = 220: 	
Vary Activity (slow shake, medium-paced shake, fast shake)	Adapted_Pace should increase with increasing activity	Slow Shake:  Medium Shake:  Fast Shake: 	PASS
DOO AV Delay, and Pulse Width fall within given parameters.	Pacemaker should pace both atrium and ventricle with a small AV delay.		PASS
AV Delay > 2*Pulse_Width	Error should be given since it is not physically possible so have a delay twice as large as the pulse width.	Matlab Diagnostic: "Inconsistent numeric values for parameter 'Value' in 'DOO/AV_Delay': Quantized parameter value (50) is greater than maximum(40)"	PASS
Pulse_Width >	Simulink should	Matlab Diagnostic:	PASS

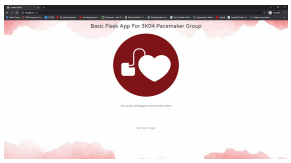
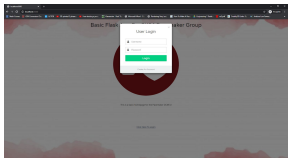
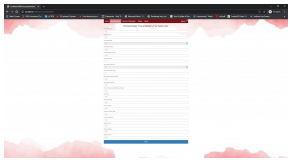
Pulse_Period	generate an error because it is not possible to have the pulse width larger than the pulse period.	"Inconsistent numeric values for parameter 'Value' in 'DOO/Pulse_Width': Quantized parameter value (1500) is greater than maximum(1000)"	

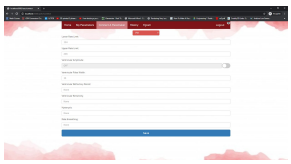
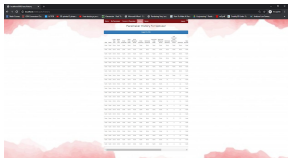
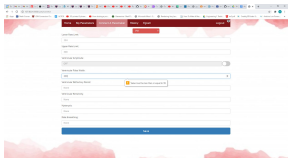
Python

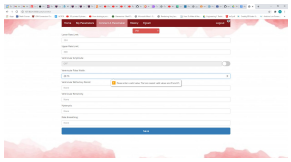
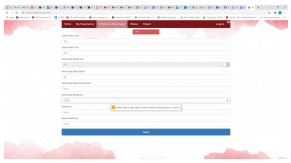
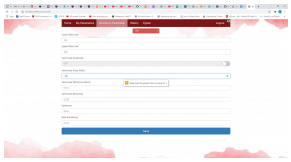
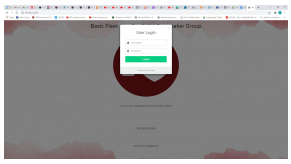
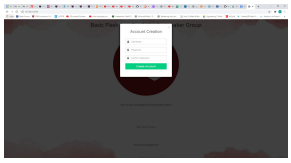
Unit testing for python was completed using the PyTest library. Test results were output to both html, the generally accepted viewing method for python documentation and testing, and pdf for those unable to open an html document. To view these test results please refer to the DCM Testing document [2].

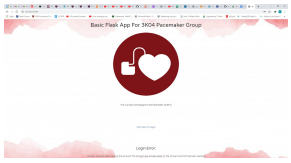
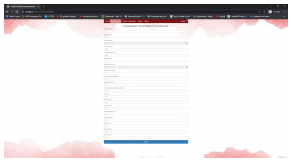
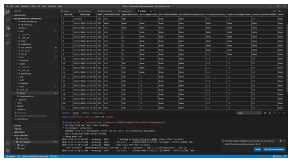
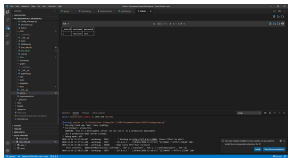
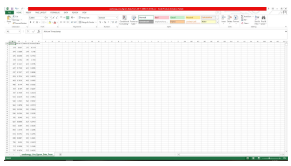
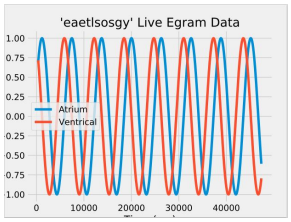
All black box testing was done using images generated by the unit tests, so the results can be visually compared to ensure the DCM acts as expected by a user with no knowledge of its implementation.

Table #45: Python Test Cases

Tested Requirement	Expected Outcome	Actual Outcome	Pass/Fail
DCM-APP-REQ-02	The DCM welcome screen should properly render itself to the screen when opened		Pass
DCM-APP-REQ-02	The DCM User login modal should properly render itself to the screen when the login button is clicked		Pass
DCM-APP-REQ-02	When a user attempts to log in with valid credentials the DCM should take them to their user welcome screen		Pass
DCM-APP-REQ-02	When a user is logged in, they should be able to access their pacemaker parameters by clicking on the 'My Parameters' page		Pass

DCM-APP-REQ-02	When a user is logged in, they should be able to access their mode specific pacemaker parameters and pacemaker programming options by clicking on the 'Connect A Pacemaker' page		Pass
DCM-APP-REQ-02	When a user is logged in, they should be able to access their parameter history by clicking on the 'History' page		Pass
DCM-APP-REQ-02	When a user is logged in, they should be able to access a live graph of their egram data by clicking the 'Egram' page		Pass
DCM-APP-REQ-02	When a user is logged in, they should be able to logout and be sent back to the login screen by clicking 'Logout'		Pass
DCM-APP-REQ-04	When a user is logged in, the icon in the top right of the screen should indicate when a pacemaker is not connected		Pass
DCM-APP-REQ-04	When a user is logged in, the icon in the top right of the screen should indicate when a pacemaker is connected		Pass
DCM-APP-REQ-06	The DCM should limit the values a user can enter to match the restrictions detailed in the configuration file		Pass

DCM-APP-REQ-06	The DCM should limit the values a user can enter to match the restrictions detailed in the configuration file		Pass
DCM-APP-REQ-06	The DCM should limit the values a user can enter to match the restrictions detailed in the configuration file		Pass
DCM-APP-REQ-06	The DCM should limit the values a user can enter to match the restrictions detailed in the configuration file		Pass
DCM-APP-REQ-07	When on the 'Egram' page, the user should be able to see a live graph with accurate atrium and ventricle values (Since a test pulse was used for this test to isolate the DCM functionality from its serial comms, the expected graph is a sine and cosine test wave)		Pass
DCM-APP-REQ-08	When first opened, the DCM should render a login screen, allowing the user to login by clicking a 'Login' button		Pass
DCM-APP-REQ-08	When the user clicks the 'Login' button on the login page, the DCM should allow the user to login using valid credentials		Pass
DCM-APP-REQ-08	When the user clicks the 'Create An Account' button on the login page, the DCM should allow the user to		Pass

	create a new account		
DCM-APP-REQ-09	When the user tries to create an account, if they use pre-existing credentials, the DCM should deny the account creation and notify the user		Pass
DCM-APP-REQ-10	When on the 'My Parameters' page, the user should be able to see an editable list of all their pacemaker parameters		Pass
DCM-DBS-REQ-01	When the DCM is opened, it should create a database which stores the users data		Pass
DCM-DBS-REQ-01	When the DCM is opened, it should create a database which stores the users data		Pass
DCM-GPH-REQ-01	When on the 'Egram' page, the user should be able to view a graph which updates in real time		Pass
DCM-GPH-REQ-03	If the user clicks the 'Export' button on the 'Egram' page, a .csv file containing all the graph data should be published to the 'downloads' folder		Pass
DCM-GPH-REQ-03	If the user clicks the 'Export' button on the 'Egram' page, a .pdf file containing the graph should be published to the 'downloads' folder		Pass