Heartburn: An Embedded Pacemaker Control System

Group 3 - Heartburn

Fall 2025

Contents

1. Group Members

Table 1: Names and Student Numbers Table

Name	Student Number
Akash Ahilan	400521928
Elijah Cosby	400538072
Lorenzo Santos	400440807
Pranav Chandrakumar	400455551
Sam Wauchope	400530259
Sina Forouzanfar	400529762

2. Part 1

2.1. Introduction

This document outlines the components of both Phase 1 and Phase 2 of the PACEMAKER project, focusing on the design of a cardiac pulse generator and the development of its control application through a Device Controller-Monitor (DCM), with serial communication linking the two. The system's primary purpose is to continuously monitor and regulate a patient's heart rate by detecting bradycardia conditions and delivering corrective pulses. Given the critical nature of the device, the project emphasizes adherence to clear software engineering principles, with the first objective being to abstract all necessary requirements from the project specification documents and the second to design and implement the core logic using the defined abstracted variables. The scope of this work includes implementing control logic for the fundamental pacing modes (AOO, VOO, AAI, VVI, AOOR, VOOR, VVIR, and DDDR) on the NXP FRDM-K64F hardware platform, while concurrently developing the presentation layer (DCM) for user authentication and parameter input. This has required ensuring UART communication between the DCM and the Simulink model, and a critical design task common to both components has been the application of hardware abstraction, ensuring that the software logic remains fully independent of physical pin configurations

2.2. Requirements

2.2.1. Overall System Requirements

2.2.1.1. *System Purpose*

The overall PACEMAKER system, including the PG, shall monitor and regulate a patient's heart rate by detecting and providing therapy for bradycardia conditions.

2.2.1.2. *Pacing Pulse Characteristics*

The system shall generate pacing pulses with programmable amplitude and pulse width, independently adjustable for the atrial and ventricular chambers.

2.2.1.3. *Sensing Capability*

The PG shall be capable of sensing intrinsic atrial and ventricular cardiac signals using programmable thresholds.

2.2.1.4. *Hardware Hiding*

The system software design must abstract the physical hardware interactions ensuring the core pacing logic is part of the underlying software.

2.2.1.5. *Serial Communication*

The system shall support bi-directional serial communication to transmit and receive information between the DCM and the Pacemaker. The system shall be capable of transmitting real-time electrogram (EGM) data to the DCM. The system shall allow the DCM to set, store, and verify programmable parameters on the device.

2.2.1.6. *Activity Detection*

The system shall be able to detect instantaneous and prolonged physical movement of the wearer.

2.2.2. Mode-Specific Requirements

4 Modes are currently implemented within the system, following the NBG (NASPE/BPEG) pacemaker code system.

2.2.2.1. *AOO*

- The software shall deliver an atrial pacing pulse at the programmed lower rate limit.
- The software shall ensure that the atrial pulsing amplitude does not exceed the programmed atrial amplitude.
- The software shall ensure that the atrial pacing pulse duration does not exceed the programmed atrial pulse width in duration.

2.2.2.2. *VOO*

- The software shall deliver a ventricular pacing pulse at the programmed lower rate limit.
- The software shall ensure that the ventricular pulsing amplitude does not exceed the programmed ventricular amplitude.
- The software shall ensure that the ventricular pacing pulse duration does not exceed the programmed atrial pulse width in duration.

2.2.2.3. *AAI*

- The software shall deliver an atrial pacing pulse when the detected atrial rate is less than the programmed lower rate limit.
- When hysteresis pacing is enabled, the software shall deliver an atrial pacing pulse after the hysteresis interval expires with no atrial event detected, and pacing shall resume at the programmed lower rate limit.
- When rate smoothing is enabled, the software shall deliver an atrial pacing pulse at the atrial smoothing rate if the intrinsic atrial rate decreases faster than the smoothing rate.
- The software shall ensure that the atrial pulsing amplitude does not exceed the programmed atrial amplitude.
- The software shall ensure that the atrial pacing pulse duration does not exceed the programmed atrial pulse width in duration.

2.2.2.4. *VVI*

- The software shall deliver a ventricular pacing pulse when the detected ventricular rate is less than the programmed lower rate limit.
- When hysteresis pacing is enabled, the software shall deliver a ventricular pacing pulse after the hysteresis interval expires with no ventricular event detected, and pacing shall resume at the programmed lower rate limit.
- When rate smoothing is enabled, the software shall deliver a ventricular pacing pulse at the ventricular smoothing rate if the intrinsic ventricular rate decreases faster than the smoothing rate.
- The software shall ensure that the ventricular pulsing amplitude does not exceed the programmed ventricular amplitude.
- The software shall ensure that the ventricular pacing pulse duration does not exceed the programmed ventricular pulse width in duration.

2.2.2.5. *AOOR*

- The software shall pace the atrium at a Sensor Indicated Rate (SIR) determined by the accelerometer data, ranging between the programmed Lower Rate Limit (LRL) and Maximum Sensor Rate (MSR).

- In the absence of activity, the software shall pace at the LRL.

2.2.2.6. *VOOR*

- The software shall pace the ventricle at a Sensor Indicated Rate (SIR) determined by the accelerometer data, ranging between the programmed LRL and MSR.
- In the absence of activity, the software shall pace at the LRL.

2.2.2.7. *AAIR*

- The software shall pace the atrium when the intrinsic atrial rate is slower than the Sensor Indicated Rate (SIR).
- The software shall inhibit pacing if a valid intrinsic atrial event is sensed during the alert period.
- The pacing interval shall adjust dynamically based on activity levels, not exceeding the MSR.

2.2.2.8. *VVIR*

- The software shall pace the ventricle when the intrinsic ventricular rate is slower than the Sensor Indicated Rate (SIR).
- The software shall inhibit pacing if a valid intrinsic ventricular event is sensed during the alert period.
- The pacing interval shall adjust dynamically based on activity levels, not exceeding the MSR.

2.2.2.9. *DDDR*

- The software shall deliver an atrial pacing pulse when the detected atrial rate is less than the programmed lower rate limit.
- The software shall deliver a ventricular pacing pulse after a pulsed/sensed atrial event, if a ventricular pulse is not detected before the programmed atrioventricular delay.
- The software shall inhibit ventricular pacing while the push button is pressed.

- When hysteresis pacing is enabled, the software shall deliver an atrial pacing pulse after the hysteresis interval expires with no atrial event detected, and pacing shall resume at the programmed lower rate limit.
- When rate smoothing is enabled, the software shall deliver an atrial pacing pulse at the atrial smoothing rate if the intrinsic atrial rate decreases faster than the smoothing rate.
- The software shall dynamically increase the atrial pacing rate in response to detected movement by the wearer.
- The software shall dynamically decrease the atrial pacing rate in response to lack of movement by the wearer.
- The software shall ensure that the atrial pulsing amplitude does not exceed the programmed atrial amplitude.
- The software shall ensure that the atrial pacing pulse duration does not exceed the programmed atrial pulse width in duration.
- The software shall ensure that the ventricular pulsing amplitude does not exceed the programmed ventricular amplitude.
- The software shall ensure that the ventricular pacing pulse duration does not exceed the programmed ventricular pulse width in duration.

2.2.3. DCM-Specific Requirements

2.2.3.1. Login Page

- The user interface must accept a valid username and password if an account has been created.
- The user interface must display an error message if a username and password does not correspond to a created account.
- The user interface must deny the user access if their username and password are invalid.
- The user interface must give the user the option to create a new account.
- The user interface must redirect users to the login page is they are not logged in.

2.2.3.2. Create New User Page

- The user interface must allow new users to create a new account by inputting a name and password.
- The user interface should ask the user to confirm their password by re-entering it before account creation is completed.

- The user interface must display an error message if there are already 10 existing accounts, any attempt to register a new account must be denied.
- The user interface must prevent the creation of accounts with duplicate usernames.
- The user interface must redirect users to the login page if the creation of an account is successful.

2.2.3.3. Landing Page

- The user interface shall display a dashboard of available DCM pages:
    - Device Connection Page: relevant information regarding the user and device connection status, including the serial number and device name from the pacemaker board.
    - Parameters Page: access to pacing parameters.
    - Electrogram Page: viewing of ECG data.
    - Reports Page: generate various reports.
    - Utilities Page: access DCM information and configuration tools.
- The user interface should display the user's name and account creating date.
- The user interface should display the device model name, serial number, battery status, and last interrogation date.
- Upon logging into the DCM application successfully, the user interface should be initially set to the device connection page.
- The user interface must allow the user to switch between the available DCM pages
- The user interface must allow the user to log out and redirect them to the login page.

2.2.3.4. Parameter Table Page

- The system must display all programmable parameters available to the pacemaker, including the following:
    - Pacing mode
    - Upper and lower rate limit
    - Maximum sensor rate
    - Fixed and dynamic AV delay
    - Minimum dynamic AV delay
    - Sensed AV delay offset
    - Regulated A and V pulse amplitude
    - Unregulated A and V pulse amplitude
    - A and V pulse width
    - A and V Sensitivity
    - Ventricular and atrial refractory period
    - PVARP and PVARP extension

    - Hysteresis rate limit
    - Rate smoothening
    - ATP mode, duration, and fallback time
    - Ventricular blanking
    - Reaction and recovery time
    - Response factor
- The user interface must allow users to modify parameter values within the allowable range.
- The user interface must provide the option to reset values to the nominal values.
- The user interface must allow uses to save modified parameters.
- The user interface must validate all modified values against the allowable ranges before saving changes.
- The user interface should display the current save status of the parameters (saved, unsaved).
- The user interface must communicate with the python backend to send parameters to the pacemaker board.
- The user interface should verify the parameters were correctly sent to the pacemaker board.
- The user interface must prevent the sending of parameters to the pacemaker board if the parameters are in an invalid state (DCM not connected, invalid or conflicting parameter values, unsaved parameters).

2.2.3.5. Reports Page

- The user interface must allow users to generate, export, and print the following reports on demand:
    - Bradycardia parameters report
    - Temporary parameters report
    - Implant data report
    - Threshold test results report
    - Measured data report
    - Rate histogram and trending report
    - Final session report
- Each report must include a header that include the institution name, device name/serial number, DCM version, and date.
- The user interface must automatically update the bradycardia and temporary parameters page when any parameters are changed or saved.

2.2.3.6. System Utilities Page

- The user interface should include the following utilities:
    - Clock: adjust the DCM system clock
    - About: display the application version, serial number, model, and institution information.
    - New patient: start a new telemetry session without restarting the program, the user interface must redirect users to the login page.
    - Quit session: safely terminate the telemetry session and close the application.

2.3. Design

2.3.1. System Architecture

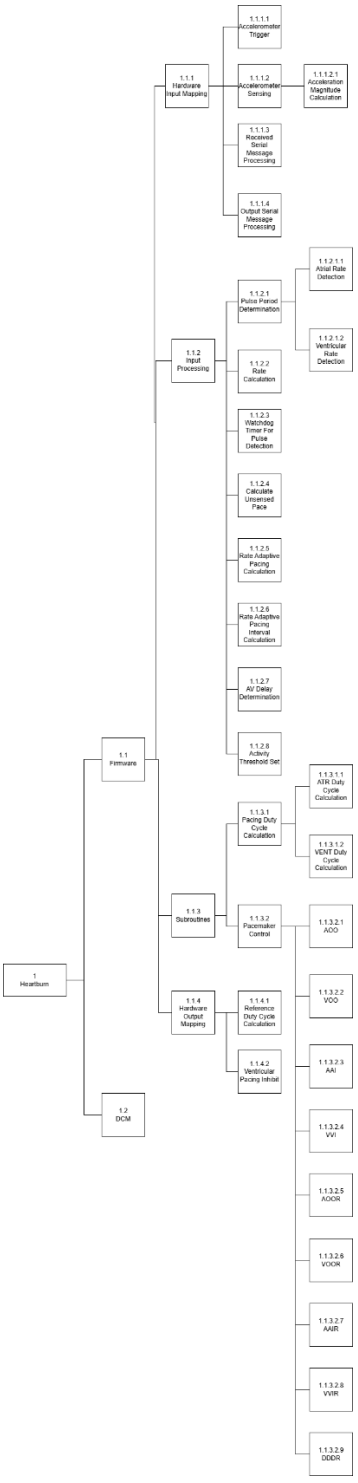


Figure 1: Module Decomposition Tree

Table 2: Module Guide for System Architecture

ID	Name	Responsibility	Secret Details
1	Heartburn	Main system. Contains all code, modules, and interfaces, encapsulated within one software package to deliver a pacemaker system.	Highest level module, contains all secrets.
1.1	Firmware	All hardware interfacing and software routines. Delivers pace control using programmed parameters.	All software implementation, hardware abstraction, and runtime logic.
1.1.1	Hardware Input Mapping	Interfaces with physical input pins Provides sensor status and programmed parameter values to the system.	Specific FRDM-K64F pin assignments, configuration of hardware interface blocks. Provides a method for representing fixed parameters.
1.1.1.1	Accelerometer Trigger	Acts as an external trigger to poll the accelerometer at discrete intervals.	Specific time period to toggle the accelerometer from on to off.
1.1.1.2	Accelerometer Sensing	Directly polls accelerometer pin and outputs current best estimation of acceleration.	Moving average implementation that reduces sensor noise.
1.1.1.2.1	Acceleration Magnitude Calculation	Computes the acceleration	Numerical algorithm to

		magnitude of the vector output of the accelerometer.	compute vector magnitude.
1.1.1.3	Received Serial Message Processing	Parses incoming byte stream from UART RX pin and executes associated command.	Byte headers to be parsed, and corresponding payload data.
1.1.1.4	Output Serial Message Processing	Sends outgoing byte streams to UART TX pin to be received by DCM.	Byte headers to be parsed, and corresponding payload data.
1.1.2	Input Processing	Process raw sensor data, detect cardiac events, calculate heart rates and determine appropriate pacing time intervals based on current mode and parameters.	Algorithms for heart rate calculation Stateflow logic for pulse detection, formulas for dynamic timing interval calculation, refractory period handling logic.
1.1.2.1	Pulse Period Determination	Delivers the period calculated between two pulses of the chamber of interest. Indicates when a first rate is calculated.	Timing details, logic for detecting a heart chamber event, detection subroutines based on mode active.
1.1.2.1.1	Atrial Rate Detection	Calculates the detected pulse period for the atrium.	Timing details, and logic for determining atrial pulse events.
1.1.2.1.2	Ventricular Rate Detection	Calculates the detected pulse period for the ventricle.	Timing details, and logic for determining ventricle pulse events.

1.1.2.2	Rate Calculation	Instantaneous heart rate and rolling average heart rate.	Conversion from pulse detection period to rate, average calculation.
1.1.2.3	Watchdog Timer for Pulse Detection	Provides a watchdog timer for timeout on failure to detect heart pulse.	Conversion from rate limits and smoothing rates to timing period.
1.1.2.4	Calculate Unsensed Pace	Provides timing period for lower rate limit to be used specifically for AOO and VOO.	Algorithm to convert from LRL to timing period.
1.1.2.5	Rate Adaptive Pacing Calculation	Computes the heart rate (in ppm) to pulse at, based on current user activity.	Algorithm to dynamically compute pacing rate.
1.1.2.6	Rate Adaptive Pacing Interval Calculation	Computes the timing period to send to pacemaker control logic for rate adaptive modes.	Algorithm to convert ppm to ms/pulse.
1.1.2.7	AV Delay Determination	Computes the AV Delay to be used for dual mode pacing, based off of programmed parameters.	Specific logic to toggle from dynamic to fixed AV delay.
1.1.2.8	Activity Threshold Set	Computes the threshold to be used to trip when physical movement is detected.	Algorithm to compute activity threshold.
1.1.3	Subroutines	Implement the core control logic for each pacing mode (AOO, VOO,	Specific Stateflow implementation (Pacemaker Control chart)

		AAI, VVI). Select the active mode and start the sequence of charging, pacing, and sensing inhibition.	including state definitions, transition conditions for controlling outputs, definition of mode constants.
1.1.3.1	Pacing Duty Cycle Calculation	Converts programmable parameters for pulse amplitude to duty cycles.	Algorithm to convert duty cycle and capacitor peak voltage.
1.1.3.1.1	ATR Duty Cycle Calculation	Computes the duty cycle to be applied to capacitors to discharge and pace the atrium.	Algorithm and capacitor peak voltage needed to compute duty cycle.
1.1.3.1.2	VENT Duty Cycle Calculation	Computes the duty cycle to be applied to capacitors to discharge and pace the ventricle.	Algorithm and capacitor peak voltage needed to compute duty cycle.
1.1.3.2	Pacemaker Control	Uses programmable parameters to set control system routine as active.	All control systems (ex. AOO, VVI), and internal variables used for system execution.
1.1.3.2.1	AOO	Consult requirements 2.2.2.1.	Internal variables required for pacing logic and timing.
1.1.3.2.2	VOO	Consult requirements 2.2.2.2.	Internal variables required for pacing logic and timing.
1.1.3.2.3	AAI	Consult requirements 2.2.2.3.	Internal variables required for pacing logic and timing.
1.1.3.2.4	VVI	Consult requirements 2.2.2.4.	Internal variables required for pacing logic and timing.

1.1.3.2.5	AOOR	Consult requirements 2.2.2.5.	Internal variables required for pacing logic and timing.
1.1.3.2.6	VOOR	Consult requirements 2.2.2.6.	Internal variables required for pacing logic and timing.
1.1.3.2.7	AAIR	Consult requirements 2.2.2.7.	Internal variables required for pacing logic and timing.
1.1.3.2.8	VVIR	Consult requirements 2.2.2.8.	Internal variables required for pacing logic and timing.
1.1.3.2.9	DDDR	Consult requirements 2.2.2.9.	Internal variables required for pacing logic and timing.
1.1.4	Hardware Output Pin Mapping	Translate logical commands into physical actions on the FRDM-K64F output pins.	Specific FRDM-K64F pin assignments, configuration of hardware interface blocks, method/formula for converting desired voltage/sensitivity amplitude into PWM duty cycle.
1.1.4.1	Reference Duty Cycle Calculation	Computes the duty cycle for reference pacing capacitors.	Algorithm to take reference voltages and convert to duty cycles.
1.1.4.2	Ventricular Pacing Inhibit	Inhibits pacing of ventricle while push button is pressed.	Logic to toggle pacing duty cycle to 0 while button is pressed.
2	DCM (Device Controller Monitor)	Contains all routines, functions and classes used by the DCM software.	All secrets contained within DCM submodules and subsystem components.

2.3.2. Programmable Parameters

The programmable parameters required for the AOO, VOO, AAI, and VVI modes are implemented within the HardwareInputMapping subsystem of the Simulink model. This approach implements the information hiding principle by encapsulating the source of these configuration values. The core logic modules (InputProcessing and Subroutines) receive these parameters as inputs downstream from the HardwareInputMapping, ensuring they depend only on the parameter *values*, not how or where they are defined.

The following parameters are defined as constant blocks within HardwareInputMapping:

Table 3: Hardware Input Mapping Programmable Parameters

Parameter Name	**Default Value in Model**	**Unit**	**Purpose**
Mode	AOO	-	Identifies the active pacing control mode
Lower Rate Limit	50	ppm	Minimum pacing rate (with no intrinsic activity).
Upper Rate Limit	175	ppm	Maximum rate for tracking sensed atrial events.
Atrial Amplitude	3.75	V	Target voltage for atrial pacing pulse. Used to calculate PWM duty cycle.
Atrial Pulse Width	4	ms	Duration of the atrial pacing pulse.
Ventricular Amplitude	3.75	V	Target voltage for ventricular pacing pulse. Determines PWM duty cycle.
Ventricular Pulse Width	4	ms	Duration of the ventricular pacing pulse.
Atrial Sensitivity	2.75	mV	Threshold voltage for detecting intrinsic atrial events. Determines ATR_CMP_REF_PWM duty cycle.
Ventricular Sensitivity	2.5	mV	Threshold voltage for detecting intrinsic ventricular events. Determines VENT_CMP_REF_PWM duty cycle.
Atrial Refractory Period	250	ms	Period after an atrial event during which atrial sensing is ignored.
Ventricular Refractory Period	250	ms	Period after a ventricular event during which ventricular sensing is ignored.

Maximum Sensor Rate	120	ppm	Maximum rate at which rate adaptive pacing can pace at.
Fixed AV Delay	150	ms	Static time delay between an atrial pulse and ventricular pulse.
Dynamic AV Delay	Off	-	Flag to enable dynamic adjustment of the AV delay used for dual mode pacing.
Minimum Dynamic AV Delay	50	ms	Minimum AV delay that can be applied when dynamic adjustment of AV delay is enabled.
Sensed AV Delay Offset	0	ms	Time adjustment applied for sensed atrial events in dual mode pacing to account for dynamic AV delay.
Atrial Amplitude Regulated	3.5	V	Target voltage for atrial pacing pulse. Used to calculate PWM duty cycle in rate adaptive modes.
Ventricular Amplitude Regulated	3.75	V	Target voltage for ventricular pacing pulse. Used to calculate PWM duty cycle in rate adaptive modes.
PVARP	250	ms	Time after a ventricular pulse that an atrial pulse cannot occur.

PVARP Extension	0	ms	Dynamic adjustment to PVARP when dynamic AV adjustment is enabled.
Hysteresis Rate Limit	0	ppm	Nominal rate to pace at when hysteresis pacing is enabled.
Rate Smoothing	0	-	Proportion of lower rate limit that is used to as threshold for when rate smoothing pacing should be applied.
ATR Mode	Off	-	Used to enable atrial tachycardia detection.
ATR Fallback Time	1	min	Time period allowed for tachycardia pacing inhibition before regular pacing is enabled.
Ventricular Blanking	40	ms	Time period to inhibit ventricular sensing after atrial event detection to prevent accidental pacing.
Activity Threshold	Medium	-	Arbitrary threshold identifier used to identify when physical activity is detected by the wearer.
Reaction Time	30	sec	Amount of time that elapses prior to pacing in rate adaptive modes after physical activity is detected.

Response Factor	8	-	Scaling applied to pacing in rate adaptive modes, controlling rate of pacing rate increase.
Recovery Time	5	min	Time delay to elapse after physical activity is no longer detected for rate adaptive pacing to return to nominal pacing.

2.3.3. Hardware Inputs and Outputs

To ensure the core pacemaker logic remains independent of specific hardware details, all interactions with the physical microcontroller pins are managed by two hardware hiding modules: HardwareInputMapping (for inputs) and Hardware Output Pin Mapping (for outputs). These modules abstract the hardware by mapping pin signals to variables used in the Simulink model, and vice-versa. In this design, input pins directly correspond to sensed signals, bringing information into the microcontroller about the heart's activity. Conversely, output pins correspond to controlled signals, sending commands from the microcontroller to execute specific actions. The following table details the pin mappings implemented in the main model:

Table 4: Hardware I/O Table

Hardware Pin	**Input /Output**	**Internal Variable Representation**	**Data Type**	**Functionality**
PTC16 (D0)	Input	ATR_CMP_DETECT	boolean	Digital signal indicating detection of an intrinsic atrial event (via comparator).
PTC17 (D1)	Input	VENT_CMP_DETECT	boolean	Digital signal indicating detection of an intrinsic ventricular event (via comparator).
PTB2 (A0)	Input	Analog Input (ATR_SIGNAL)	single	Raw analog electrogram signal from the atrium.

PTB11 (A3)	Input	Analog Input1 (ATR_RECT_SIGNAL)	single	Rectified analog signal from the atrium (used by shield comparator circuitry).
PTD1 (D13)	Output	Digital Write1 (FRONTEND_CTRL)	boolean	Enables/disables the shield's sensing circuitry (Currently set HIGH).
PTA0 (D8)	Output	Digital Write1 (ATR_PACE_CTRL)	boolean	Controls the switch to deliver a pacing pulse to the atrium.
PTC4 (D9)	Output	Digital Write6 (VENT_PACE_CTRL)	boolean	Controls the switch to deliver a pacing pulse to the ventricle.
PTB9 (D2)	Output	Digital Write2 (PACE_CHARGE_CTRL)	boolean	Controls the connection of the PWM signal to charge the main pacing capacitor (C22).
PTD0 (D10)	Output	Digital Write3 (PACE_GND_CTRL)	boolean	Connects the tip electrodes (return path) to ground during pacing pulse delivery.
PTD2 (D11)	Output	Digital Write (ATR_GND_CTRL)	boolean	Grounds the atrial ring electrode, typically used to discharge the blocking capacitor (C21).
PTD3 (D12)	Output	Digital Write5 (VENT_GND_CTRL)	boolean	Grounds the ventricular ring electrode, typically used to discharge the

				blocking capacitor (C21).
PTB23 (D4)	Output	Digital Write7 (Z_ATR_CTRL)	boolean	Connects the impedance measurement circuit to the atrial electrode.
PTC3 (D7)	Output	Digital Write8 (Z_VENT_CTRL)	boolean	Connects the impedance measurement circuit to the ventricular electrode.
PTA2 (D5)	Output	PWM Output (PACING_REF_PWM)	single	PWM signal whose duty cycle sets the amplitude of the pacing pulse.
PTC2 (D6)	Output	PWM Output1 (ATR_CMP_REF_PWM)	double	PWM signal whose duty cycle sets the comparator threshold voltage for atrial sensing.
FXOS8700 6-Axes Sensor	Input	activity_magnitude	single	Sensed magnitude of acceleration vector produced by onboard accelerometer.
Push Button (SW3)	Input	VentPaceInhibit	boolean	Detects state of push button in order to inhibit ventricular pacing.
UART0 RX	Input	rx_data, status	uint8, boolean	Receives byte stream by DCM for serial communication.
UART0 TX	Output	-	uint8	Outgoing byte stream for serial communication.

2.3.4. State Machine Design

The core pacing logic is implemented using Stateflow state machines, primarily located within the Subroutines/Pacemaker Control chart and helper charts in the Input Processing subsystem. This provides a clear structure for managing the different modes. The main Pacemaker Control chart selects the active mode based on the Mode input parameter.

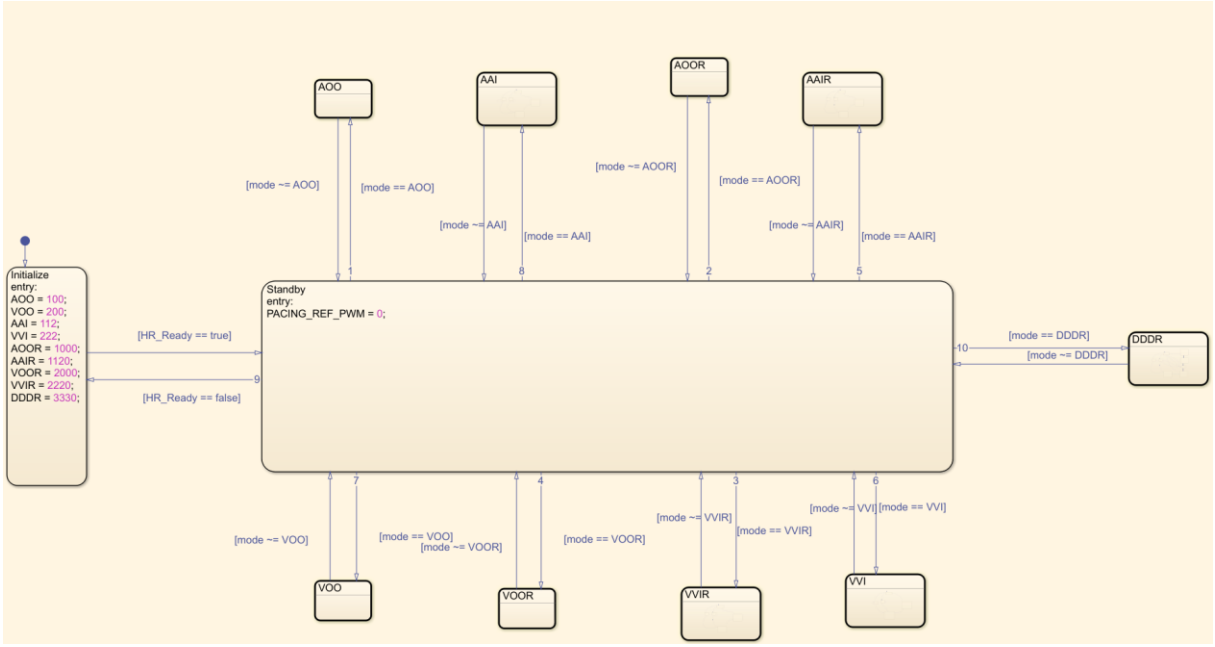


Figure 2: Pacemaker Mode Control

2.3.4.1. *AOO State Machine Design*

Table 5: Tabular State Diagram for AOO

Current State	**Condition / Event**	**Next State**	**Actions During Transition / Entry to Next State**
Initial	Entry to AOO mode	CHARGE	ATR_PACE_CTRL=LOW; VENT_PACE_CTRL=LOW; PACING_REF_PWM=ATR_DUTY_CYCLE; PACE_CHARGE_CTRL=HIGH; PACE_GND_CTRL=HIGH; Z_ATR_CTRL=LOW; Z_VENT_CTRL=LOW; ATR_GND_CTRL=HIGH; VENT_GND_CTRL=LOW;
CHARGE	after(f_aointerval-atrium_pulse_width, msec)	DISCHARGE	PACE_CHARGE_CTRL=LOW; PACE_GND_CTRL=HIGH; ATR_PACE_CTRL=HIGH;

			ATR_GND_CTRL=LOW; Z_ATR_CTRL=LOW; Z_VENT_CTRL=LOW; VENT_GND_CTRL=LOW; VENT_PACE_CTRL=LOW; PACING_REF_PWM=ATR_DUTY_CYCLE;
DISCHARGE	after(atrium_pulse_width, msec)	CHARGE	ATR_PACE_CTRL=LOW; VENT_PACE_CTRL=LOW; PACING_REF_PWM=ATR_DUTY_CYCLE; PACE_CHARGE_CTRL=HIGH; PACE_GND_CTRL=HIGH; Z_ATR_CTRL=LOW; Z_VENT_CTRL=LOW; ATR_GND_CTRL=HIGH; VENT_GND_CTRL=LOW;

2.3.4.2. *VOO State Machine Design*

Table 6: Tabular State Diagram for VOO

Current State	**Condition / Event**	**Next State**	**Actions During Transition / Entry to Next State**
Initial	Entry to VOO mode	CHARGE	ATR_PACE_CTRL=LOW; VENT_PACE_CTRL=LOW; PACING_REF_PWM=VENT_DUTY_CYCLE; PACE_CHARGE_CTRL=HIGH; PACE_GND_CTRL=HIGH; Z_ATR_CTRL=LOW; Z_VENT_CTRL=LOW; ATR_GND_CTRL=LOW; VENT_GND_CTRL=HIGH
CHARGE	after(f_voointerval-ventricle_pulse_width, msec)	DISCHARGE	PACE_CHARGE_CTRL=LOW; PACE_GND_CTRL=HIGH; ATR_PACE_CTRL=LOW; ATR_GND_CTRL=LOW; Z_ATR_CTRL=LOW; Z_VENT_CTRL=LOW; VENT_GND_CTRL=LOW; VENT_PACE_CTRL=HIGH; PACING_REF_PWM=VENT_DUTY_CYCLE;

DISCHARGE	after(ventricle_pulse_width, msec)	CHARGE	ATR_PACE_CTRL=LOW; VENT_PACE_CTRL=LOW; PACING_REF_PWM=VENT_DUTY_CYCLE; PACE_CHARGE_CTRL=HIGH; PACE_GND_CTRL=HIGH; Z_ATR_CTRL=LOW; Z_VENT_CTRL=LOW; ATR_GND_CTRL=LOW; VENT_GND_CTRL=HIGH;

2.3.4.3. *AAI State Machine Design*

Table 7: Tabular State Diagram for AAI

Current State	**Condition / Event**	**Next State**	**Actions During Transition / Entry to Next State**
Initial	Entry to AAI mode	Initializations	ATR_CMP_REF_PWM = ATR_SENS; count=0;
Initializations	(Hystersis_flag == 1) OR (Hystersis_flag == 0 && after(wait_time_LRL, msec))	Hystersis_mode_ON OR Hystersis_mode_OFF	count = count + 1;
Hystersis_mode_ON	(count<=WaitTime)&&(~(dig_atr_outputD0 == LOW && inst_heart_rate<lower_rate_limit)) OR (dig_atr_outputD0 == LOW && inst_heart_rate<h_r_l)||((count>WaitTime)&& (dig_atr_outputD0 == LOW))	Hystersis_mode_ON OR CHARGE	count = count + 1; OR PACE_CHARGE_CTRL=LOW; PACE_GND_CTRL=HIGH; ATR_PACE_CTRL=HIGH; ATR_GND_CTRL=LOW; Z_ATR_CTRL=LOW; Z_VENT_CTRL=LOW; VENT_GND_CTRL=LOW; VENT_PACE_CTRL=LOW; PACING_REF_PWM=ATR_DUTY_CYCLE;
Hystersis_mode_OFF	(count<=WaitTime)&&(~(dig_atr_outputD0 == LOW && inst_heart_rate<lower_rate_limit)) OR (dig_atr_outputD0 == LOW &&	Hystersis_mode_OFF OR CHARGE	count = count + 1; OR PACE_CHARGE_CTRL=LOW; PACE_GND_CTRL=HIGH;

	inst_heart_rate<lower_rate_limit)||((count>WaitTime) && (dig_atr_outputD0 == LOW))]		ATR_PACE_CTRL=HIGH; ATR_GND_CTRL=LOW; Z_ATR_CTRL=LOW; Z_VENT_CTRL=LOW; VENT_GND_CTRL=LOW; VENT_PACE_CTRL=LOW; PACING_REF_PWM=ATR_DUTY_CYCLE;
CHARGE	after(atrium_pulse_width,msec)	Pulse	ATR_PACE_CTRL=LOW; VENT_PACE_CTRL=LOW; PACING_REF_PWM=ATR_DUTY_CYCLE; PACE_CHARGE_CTRL=HIGH; PACE_GND_CTRL=HIGH; Z_ATR_CTRL=LOW; Z_VENT_CTRL=LOW; ATR_GND_CTRL=HIGH; VENT_GND_CTRL=LOW;
Pulse	after(a_r_p,msec)]	Initializations	ATR_CMP_REF_PWM = ATR_SENS; count=0;

2.3.4.4. *VVI State Machine Design*

Table 8: Tabular State Diagram for VVI

Current State	**Condition / Event**	**Next State**	**Actions During Transition / Entry to Next State**
Initial	Entry to VVI mode	Initializations	VENT_CMP_REF_PWM = VENT_SENS; count=0;
Initializations	(Hystersis_flag == 1) OR (Hystersis_flag == 0 && after(wait_time_LRL, msec))	Hystersis_mode_ON OR Hystersis_mode_OFF	count = count + 1;
Hystersis_mode_ON	(count<=WaitTime)&&(~(dig_vent_outputD1 == LOW && inst_heart_rate<lower_rate_limi	Hystersis_mode_ON OR CHARGE	count = count + 1; OR PACE_CHARGE_CTRL=LOW;

	t)) OR (dig_vent_outputD1 == LOW && inst_heart_rate<h_r_l)||((count>WaitTime)&& (dig_vent_outputD1 == LOW))		PACE_GND_CTRL=HIGH; ATR_PACE_CTRL=LOW; ATR_GND_CTRL=LOW; Z_ATR_CTRL=LOW; Z_VENT_CTRL=LOW; VENT_GND_CTRL=LOW; VENT_PACE_CTRL=HIGH; PACING_REF_PWM=VENT_DUTY_CYCLE;
Hystersis_mode_OFF	(count<=WaitTime)&&(~(dig_vent_outputD1 == LOW && inst_heart_rate<lower_rate_limit)) OR (dig_vent_outputD1 == LOW && inst_heart_rate<lower_rate_limit)||((count>WaitTime) && (dig_vent_outputD1 == LOW))]	Hystersis_mode_OFF OR CHARGE	count = count + 1; OR PACE_CHARGE_CTRL=LOW; PACE_GND_CTRL=HIGH; ATR_PACE_CTRL=LOW; ATR_GND_CTRL=LOW; Z_ATR_CTRL=LOW; Z_VENT_CTRL=LOW; VENT_GND_CTRL=LOW; VENT_PACE_CTRL=HIGH; PACING_REF_PWM=VENT_DUTY_CYCLE;
CHARGE	after(ventricle_pulse_width,msec)	Pulse	ATR_PACE_CTRL=LOW; VENT_PACE_CTRL=LOW; PACING_REF_PWM=VENT_DUTY_CYCLE; PACE_CHARGE_CTRL=HIGH; PACE_GND_CTRL=HIGH; Z_ATR_CTRL=LOW; Z_VENT_CTRL=LOW; ATR_GND_CTRL=LOW; VENT_GND_CTRL=HIGH;
Pulse	after(v_r_p,msec)]	Initializations	VENT_CMP_REF_PWM = VENT_SENS; count=0;

2.3.4.5. *AOOR State Machine Design*

Table 9: Tabular State Diagram for AOOR

Current State	**Condition / Event**	**Next State**	**Actions During Transition / Entry to Next State**
Initial	Entry to AOOR mode	CHARGE	ATR_PACE_CTRL=LOW; VENT_PACE_CTRL=LOW; PACING_REF_PWM=ATR_DUTY_CYCLE; PACE_CHARGE_CTRL=HIGH; PACE_GND_CTRL=HIGH; Z_ATR_CTRL=LOW; Z_VENT_CTRL=LOW; ATR_GND_CTRL=HIGH; VENT_GND_CTRL=LOW;
CHARGE	after(f_xoorinterval-atrium_pulse_width, msec)	DISCHARGE	PACE_CHARGE_CTRL=LOW; PACE_GND_CTRL=HIGH; ATR_PACE_CTRL=HIGH; ATR_GND_CTRL=LOW; Z_ATR_CTRL=LOW; Z_VENT_CTRL=LOW; VENT_GND_CTRL=LOW; VENT_PACE_CTRL=LOW; PACING_REF_PWM=ATR_DUTY_CYCLE;
DISCHARGE	after(atrium_pulse_width, msec)	CHARGE	ATR_PACE_CTRL=LOW; VENT_PACE_CTRL=LOW; PACING_REF_PWM=ATR_DUTY_CYCLE; PACE_CHARGE_CTRL=HIGH; PACE_GND_CTRL=HIGH; Z_ATR_CTRL=LOW; Z_VENT_CTRL=LOW; ATR_GND_CTRL=HIGH; VENT_GND_CTRL=LOW;

2.3.4.6. *VOOR State Machine Design*

Table 10: Tabular State Diagram for VOOR

Current State	**Condition / Event**	**Next State**	**Actions During Transition / Entry to Next State**
Initial	Entry to VOOR mode	CHARGE	ATR_PACE_CTRL=LOW; VENT_PACE_CTRL=LOW; PACING_REF_PWM=VENT_DUTY_CYCLE; PACE_CHARGE_CTRL=HIGH; PACE_GND_CTRL=HIGH; Z_ATR_CTRL=LOW; Z_VENT_CTRL=LOW; ATR_GND_CTRL=LOW; VENT_GND_CTRL=HIGH
CHARGE	after(f_xoorinterval-ventricle_pulse_width, msec)	DISCHARGE	PACE_CHARGE_CTRL=LOW; PACE_GND_CTRL=HIGH; ATR_PACE_CTRL=LOW; ATR_GND_CTRL=LOW; Z_ATR_CTRL=LOW; Z_VENT_CTRL=LOW; VENT_GND_CTRL=LOW; VENT_PACE_CTRL=HIGH; PACING_REF_PWM=VENT_DUTY_CYCLE;
DISCHARGE	after(ventricle_pulse_width, msec)	CHARGE	ATR_PACE_CTRL=LOW; VENT_PACE_CTRL=LOW; PACING_REF_PWM=VENT_DUTY_CYCLE; PACE_CHARGE_CTRL=HIGH; PACE_GND_CTRL=HIGH; Z_ATR_CTRL=LOW; Z_VENT_CTRL=LOW; ATR_GND_CTRL=LOW; VENT_GND_CTRL=HIGH;

2.3.4.7. *AAIR State Machine Design*

Table 11: Tabular State Diagram for AAIR

Current State	**Condition / Event**	**Next State**	**Actions During Transition / Entry to Next State**
Initial	Entry to AAIR mode	Initializations	ATR_CMP_REF_PWM = ATR_SENS;
Initializations	((Hystersis_flag == 1) OR (Hystersis_flag == 0))&& after(wait_time_XXIR, msec))	Hystersis_mode_ON OR Hystersis_mode_OFF	No change
Hystersis_mode_ON	(dig_atr_outputD0 == LOW && inst_heart_rate<(CurrentRate-Hdiff))	CHARGE	PACE_CHARGE_CTRL=LOW; PACE_GND_CTRL=HIGH; ATR_PACE_CTRL=HIGH; ATR_GND_CTRL=LOW; Z_ATR_CTRL=LOW; Z_VENT_CTRL=LOW; VENT_GND_CTRL=LOW; VENT_PACE_CTRL=LOW; PACING_REF_PWM=ATR_DUTY_CYCLE;
Hystersis_mode_OFF	(dig_atr_outputD0 == LOW && inst_heart_rate<(CurrentRate))	CHARGE	PACE_CHARGE_CTRL=LOW; PACE_GND_CTRL=HIGH; ATR_PACE_CTRL=HIGH; ATR_GND_CTRL=LOW; Z_ATR_CTRL=LOW; Z_VENT_CTRL=LOW; VENT_GND_CTRL=LOW; VENT_PACE_CTRL=LOW; PACING_REF_PWM=ATR_DUTY_CYCLE;
CHARGE	after(ventricle_pulse_width,msec)	Pulse	ATR_PACE_CTRL=LOW; VENT_PACE_CTRL=LOW; PACING_REF_PWM=VENT_DUTY_CYCLE; PACE_CHARGE_CTRL=HIGH; PACE_GND_CTRL=HIGH; Z_ATR_CTRL=LOW; Z_VENT_CTRL=LOW; ATR_GND_CTRL=LOW;

			VENT_GND_CTRL=HIGH;
Pulse	after(v_r_p,msec)]	Initializations	ATR_CMP_REF_PWM = ATR_SENS;

2.3.4.8. *VVIR State Machine Design*

Table 12: Tabular State Diagram for VVIR

Current State	**Condition / Event**	**Next State**	**Actions During Transition / Entry to Next State**
Initial	Entry to AAIR mode	Initializations	VENT_CMP_REF_PWM = VENT_SENS;
Initializations	((Hystersis_flag == 1) OR (Hystersis_flag == 0))&& after(wait_time_XXIR, msec))	Hystersis_mode_ON OR Hystersis_mode_OFF	No change
Hystersis_mode_ON	(dig_atr_outputD0 == LOW && inst_heart_rate<(CurrentRate-Hdiff))	CHARGE	PACE_CHARGE_CTRL=LOW; PACE_GND_CTRL=HIGH; ATR_PACE_CTRL=LOW; ATR_GND_CTRL=LOW; Z_ATR_CTRL=LOW; Z_VENT_CTRL=LOW; VENT_GND_CTRL=LOW; VENT_PACE_CTRL=HIGH; PACING_REF_PWM=VENT_DUTY_CYCLE;
Hystersis_mode_OFF	(dig_atr_outputD0 == LOW && inst_heart_rate<(CurrentRate))	CHARGE	PACE_CHARGE_CTRL=LOW; PACE_GND_CTRL=HIGH; ATR_PACE_CTRL=LOW; ATR_GND_CTRL=LOW; Z_ATR_CTRL=LOW; Z_VENT_CTRL=LOW; VENT_GND_CTRL=LOW; VENT_PACE_CTRL=HIGH; PACING_REF_PWM=VENT_DUTY_CYCLE;
CHARGE	after(atrium_pulse_width,msec)	Pulse	ATR_PACE_CTRL=LOW; VENT_PACE_CTRL=LOW; PACING_REF_PWM=ATR_DUTY_CYCLE; PACE_CHARGE_CTRL=HIGH; PACE_GND_CTRL=HIGH;

			Z_ATR_CTRL=LOW; Z_VENT_CTRL=LOW; ATR_GND_CTRL=HIGH; VENT_GND_CTRL=LOW;
Pulse	after(a_r_p,msec)]	Initializations	VENT_CMP_REF_PWM = VENT_SENS;

2.3.4.9. *DDDR State Machine Design*

Table 13: Tabular State Diagram for DDDR

Current State	**Condition / Event**	**Next State**	**Actions During Transition / Entry to Next State**
Initial	Entry to DDDR mode	Initializations	ATR_CMP_REF_PWM = ATR_SENS; VENT_CMP_REF_PWM = VENT_SENS;
Initializations	((Hystersis_flag == 1) OR (Hystersis_flag == 0))&& after(wait_time_XXIR, msec)) OR dig_vent_outputD1 == HIGH	Hystersis_mode_ON OR Hystersis_mode_OFF OR Sensing_Ventricle	No change
Hystersis_mode_ON	(dig_atr_outputD0 == LOW && inst_heart_rate<(CurrentRate-Hdiff)) OR (dig_atr_outputD0 == HIGH && inst_heart_rate<(CurrentRate-Hdiff))	ATR_CHARGE OR Sensing_Ventricle	PACE_CHARGE_CTRL=LOW; PACE_GND_CTRL=HIGH; ATR_PACE_CTRL=HIGH; ATR_GND_CTRL=LOW; Z_ATR_CTRL=LOW; Z_VENT_CTRL=LOW; VENT_GND_CTRL=LOW; VENT_PACE_CTRL=LOW; PACING_REF_PWM=ATR_DUTY_CYCLE OR No change
Hystersis_mode_OFF	(dig_atr_outputD0 == LOW && inst_heart_rate<(CurrentRate)) OR (dig_atr_outputD0 == HIGH && inst_heart_rate<(CurrentRate))	ATR_CHARGE OR Sensing_Ventricle	PACE_CHARGE_CTRL=LOW; PACE_GND_CTRL=HIGH; ATR_PACE_CTRL=HIGH; ATR_GND_CTRL=LOW; Z_ATR_CTRL=LOW; Z_VENT_CTRL=LOW; VENT_GND_CTRL=LOW; VENT_PACE_CTRL=LOW;

			PACING_REF_PWM=ATR_DUTY_CYCLE; OR No change
ATR_CHARGE	after(atrium_pulse_width,msec)	ATR_Pulse	ATR_PACE_CTRL=LOW; VENT_PACE_CTRL=LOW; PACING_REF_PWM=ATR_DUTY_CYCLE; PACE_CHARGE_CTRL=HIGH; PACE_GND_CTRL=HIGH; Z_ATR_CTRL=LOW; Z_VENT_CTRL=LOW; ATR_GND_CTRL=HIGH; VENT_GND_CTRL=LOW;
ATR_Pulse	After(1,msec)	Sensing_Ventricle	No Change
Sensing_Ventricle	(dig_vent_outputD1 == HIGH) OR (after(AV_delay-1,msec))	Initializations OR VENT_CHARGE	ATR_CMP_REF_PWM = ATR_SENS; VENT_CMP_REF_PWM = VENT_SENS; OR PACE_CHARGE_CTRL=LOW; PACE_GND_CTRL=HIGH; ATR_PACE_CTRL=LOW; ATR_GND_CTRL=LOW; Z_ATR_CTRL=LOW; Z_VENT_CTRL=LOW; VENT_GND_CTRL=LOW; VENT_PACE_CTRL=HIGH; PACING_REF_PWM=VENT_DUTY_CYCLE;
VENT_CHARGE	after(ventricle_pulse_width,msec)	VENT_Pulse	ATR_PACE_CTRL=LOW; VENT_PACE_CTRL=LOW; PACING_REF_PWM=VENT_DUTY_CYCLE; PACE_CHARGE_CTRL=HIGH; PACE_GND_CTRL=HIGH; Z_ATR_CTRL=LOW; Z_VENT_CTRL=LOW; ATR_GND_CTRL=LOW; VENT_GND_CTRL=HIGH;

VENT_Pulse	after(p_v_a_r_p,msec)	Initializations	ATR_CMP_REF_PWM = ATR_SENS; VENT_CMP_REF_PWM = VENT_SENS;

2.3.5. Simulink Diagram

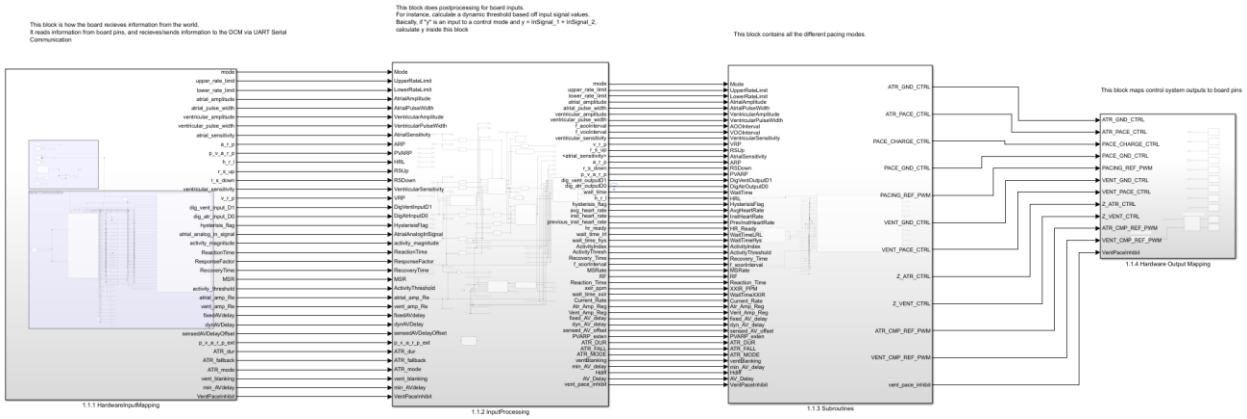


Figure 3: Simulink Main Model

2.3.6. DCM Software Structure

The DCM is a single-page application built on React 18, the Vite dev/build tool, and a Node.js/npm framework. The languages used include TypeScript (TS/TSX), JSX, HTML, and CSS. The single-page DCM portal does the following: (a) logs in a user (demo mode), (b) loads a single patient model, (c) edits & persists pacemaker parameters, (d) simulates telemetry/connectivity, (f) prints/exports reports, (g) provides help/utilities.

The primary entry point is the *app.tsx* file, which is the single page that is loaded for the site. It starts with the login page being rendered. Then the other components, such as the device connection, parameters, reports, and help & support tabs, are rendered based on the selection.

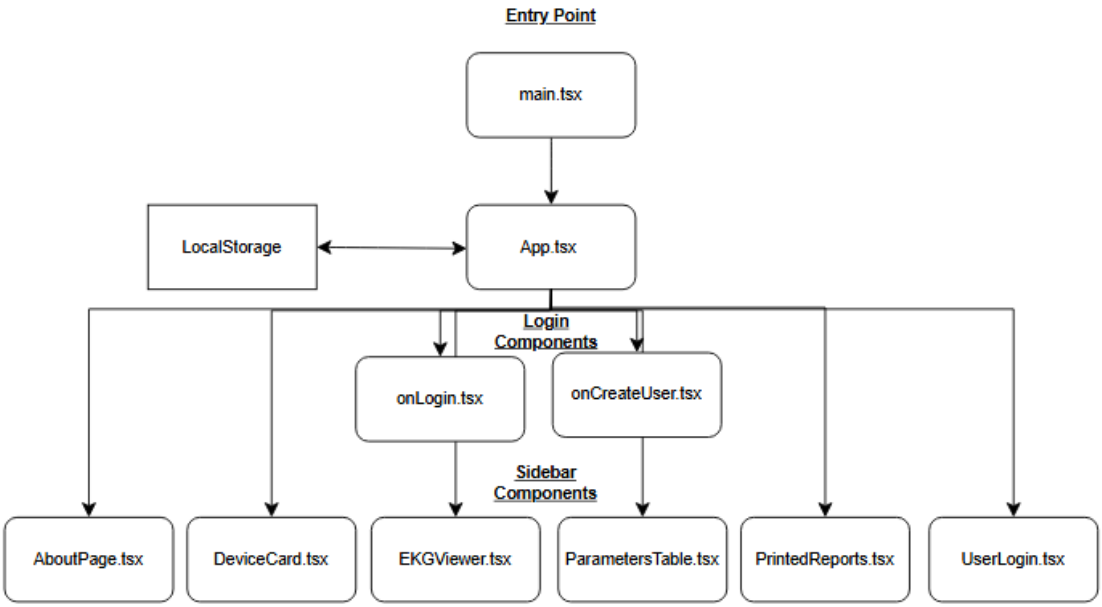


Figure 4: DCM Software Architecture Main Model

2.3.6.1. User Login

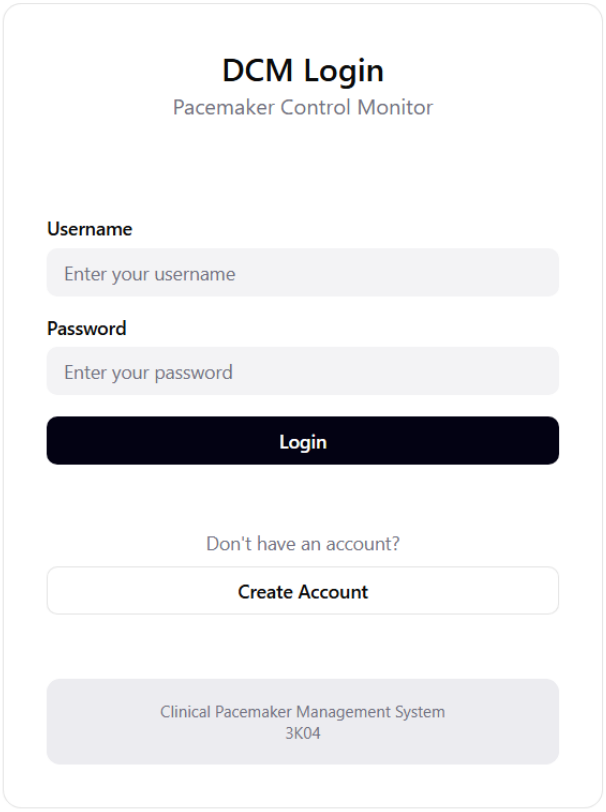


Figure 5: DCM Login Page Component

Component Props

Prop	Description
onLogin	Callback function that handles login validation. Returns boolean value if credentials are valid (true) or invalid (false)
onNavigateToCreateUser	Callback function that is triggered when user clicks the “Create Account” button

State Variables

Variables	Type	Initial	Description
username	string	“”	Stores the current value of the username input field, initially set blank

password	string	“”	Stores the current value of the password input field, initially set blank
error	string	“”	Stores error message to display to the user, initially set as blank indicating no alert being shown

State and Component Functions

Functions	**Arguments**	**Return**	**Description**
setUsername	new username: string	void	Updates the username state variable. Called from <Input onChange = {...}>
setPassword	new password: string	void	Updates the password state variable. Called from <Input onChange = {...}>
setError	error message: string	void	Updates the error state variable.
handleSubmit	e: React.FormEvent	void	Validates the user input, prevents empty submissions, and sets error message upon failure.

2.3.6.2. Create User Login Page

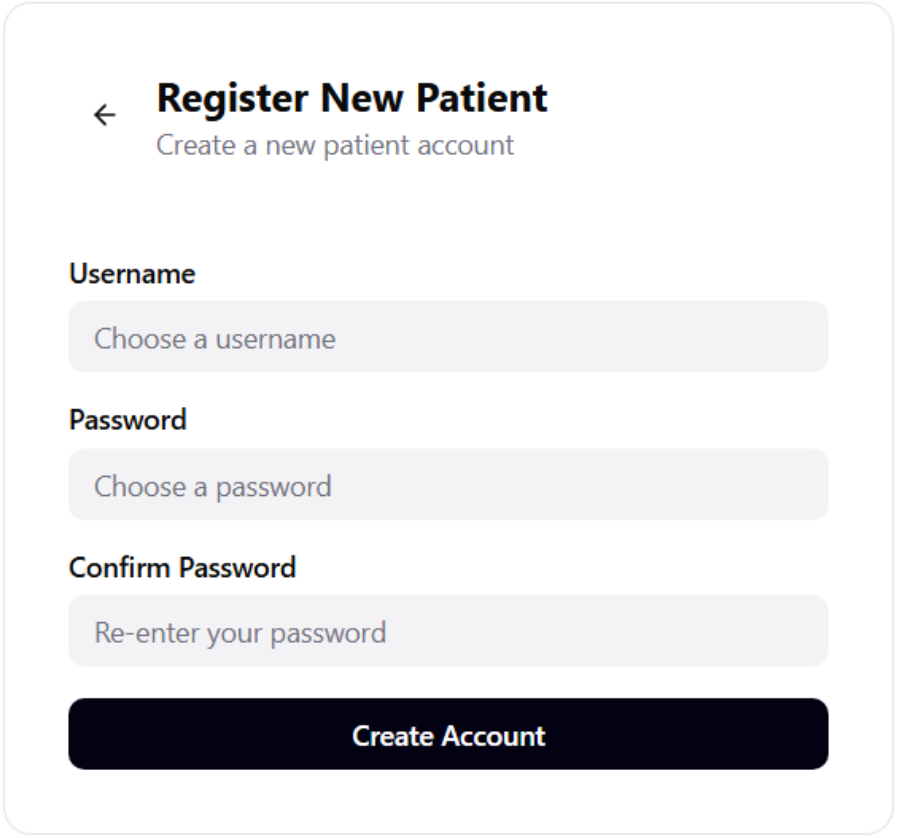


Figure 6: DCM Create User Page Component

Component Props

Prop	Description
onCreateUser	Callback function that attempts to create a new user. Returns a boolean value true if successful, of false if username exits or user limit is exceeded
onBackToLogin	Callback function to direct users to the login screen

State Variables

Variables	Type	Initial	Description
username	string	“”	Stores the username input value
password	string	“”	Stored the password input value

confirmPassword	string	“”	Stores the confirmed password input value (input when user is required to re-enter their original password)
error	string	“”	Stores error messages.
success	boolean	false	Stores boolean information is account creation is successful

State and Component Functions

Functions	**Arguments**	**Return**	**Description**
setUsername	new username: string	void	Updates the username state variable
setPassword	new password: string	void	Updates the password state variable
setConfirmPassword	confirm password: string	void	Updates the confirm password state variable
setError	error message: string	void	Updates the error message state variable
setSuccess	isSuccess: boolean	void	Updates the success status state variable.
handleSubmit	e:React.FormEvent	void	Validates that the passwords match, checks for empty fields, checks for duplicate usernames, determines if < 10 accounts saves, and calls the onCreateUser callback function

2.3.6.3. Parameters Page

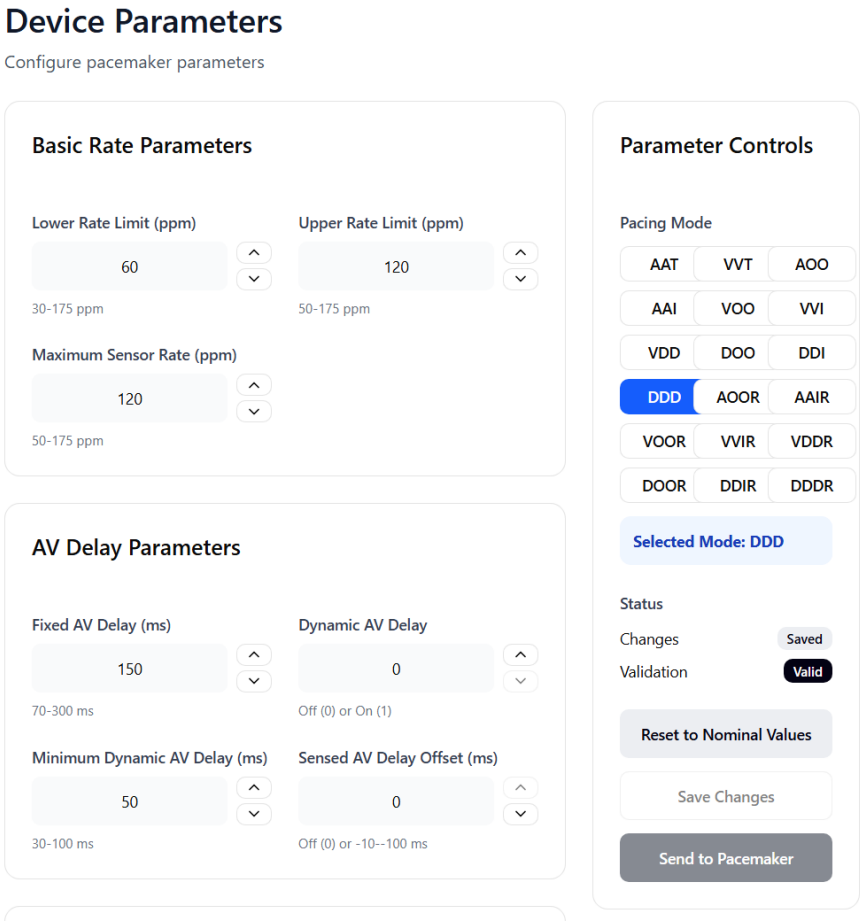


Figure 7: DCM First Part of Device Parameters Page

Component Props

Prop	Description
selectPatient	Current patient containing the ID and the saved parameters of the patient.
onParameterSaved	Callback function that tracks whether parameters are saved successfully.
onSendToPacemaker	Callback function to send parameters to pacemaker via a websocket.
onLoadBoardParameters	Callback function to request the current parameters from the pacemaker board.
verificationStatus	Current verification status after sending the parameters from the board (verified is sent values equal echoed parameters)

isConnected	Boolean indicating if the pacemaker device is connected to the DCM

State Variables

Variables	**Type**	**Initial**	**Description**
selectedPatient	string	'DDD'	Current pacing mode selected from available modes.
hasChanges	boolean	false	Tracks if any parameters have been modified since the last save.
Paremeters	Paramete[ ]	getInitial Paramet ers()	Array containing all the pacemaker parameters with the current validation state of all the parameters.
hasUnsavedCh anges	boolean	false	Tracks if the parameters have been modified but not completely saved to local storage (used in reports section).
loacalParamet ers	<string, number>	{}	Local copy of parameter values
hasChanges	boolean	n/a	Determined is parameters have been modified (derived from the Parameters variable)

State and Component Functions

Functions	**Arguments**	**Return**	**Description**
setSelectedMo de	mode: string	void	Updates the selected pacing mode.
setHasChange s	changed: boolean	void	Updates the unsaved changes flag.
setParameters	Parameter[] or update function	void	Updates the parameter array.
getInitialParam eters	None	Parameter[]	Initializes the parameter array from the patient data, also converts string values to int.
updateParamet er	id: string, value: number	void	Updates a single parameter value, validates against min and max options. Sets the isDirty and isValid flags, also triggers hasChanges.
incrementPara meter	param: Parameter	void	Increases parameter value by one step
decrementPara meter	param: Parameter	void	Decreases parameter value by one step

handleResetToNominal	None	void	Resets all parameters to nominal/default values
handleSavedChanges	None	void	Validates all parameters and saves to local storage via saveToLocalStorage function.
saveToLocalStorage	None	void	Saves parameters to local storage.
getStepValue	param: Parameter	number	Function for step values.
canIncrement	param: Parameter	boolean	Checks if the parameter can be incremented without exceeding the maximum value.
canDecrement	param: Parameter	boolean	Checks if the parameter can be incremented without exceeding the minimum value.
getLowerRateLimitStep	value: Number	number	Helper function that returns the step value for a piecewise step function.
handleSendToPacemaker	None	void	Converts parameters to object, sends via onSendToPacemaker callback
handleLoadFromBoard	None	void	Calls onLoadBoardParameters with a callback that updates all fields when board responds with parameter values

2.3.6.4. Reports Page

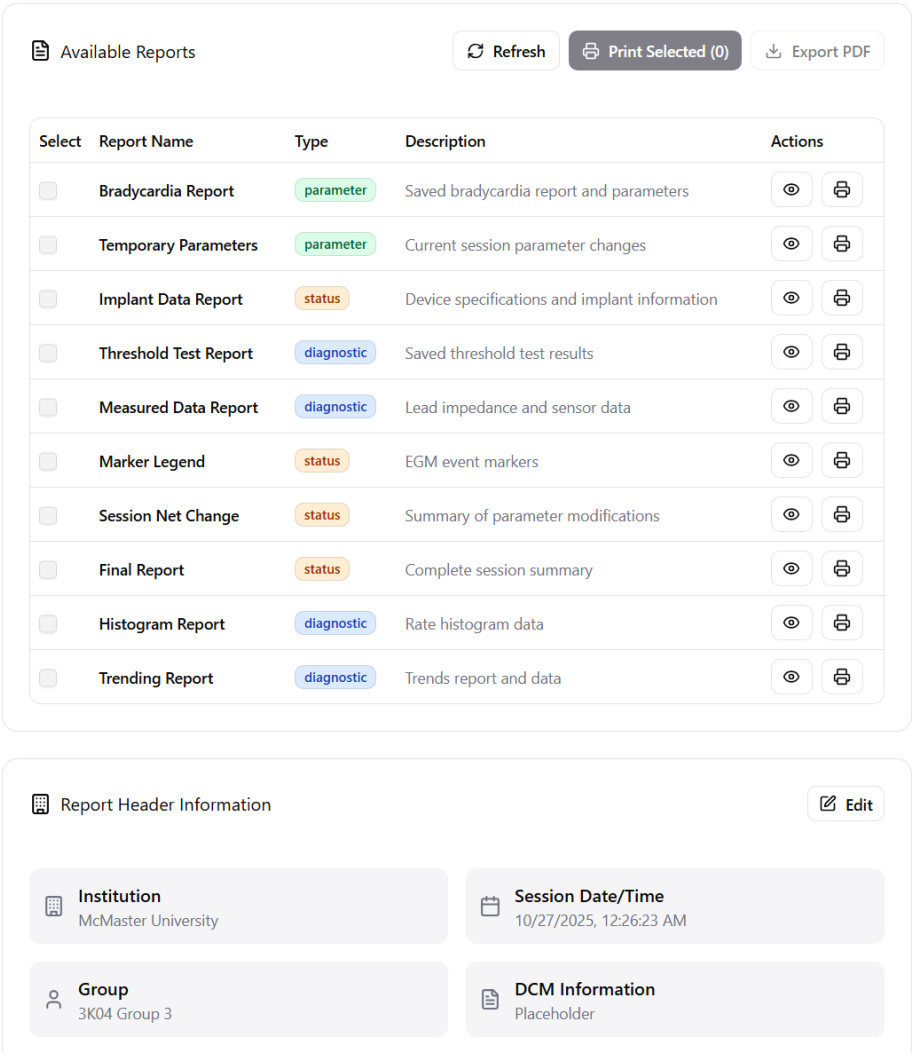


Figure 8: DCM Reports Page

Component Props

Prop	Description
selectedPatient	Current patient object containing the ID for loading the parameter history.
deviceInfo	Pacemaker device information including the serial number and device name.
ekgData	Real-time EKG datapoints from atrial and ventricular channels for measuring data points.

State Variables

Variables	Type	Initial	Description
selectedReports	string[]	[]	Array of the selected reports.
previewReport	report or null	null	Currently previewed report object.
isEditingHeader	boolean	false	Flag that indicates is header information is being edited.
headerinfo	object	{instution, date, deviceInfo, dcmInfo}	Report header information.
patientHistory	any[]	[]	An array of historical parameter entries for the current patient.
capturedEkgDate	any[]	[]	First 30 EKG data points captured for threshold/measured data reports
temporaryParams	any | null	null	Unsaved parameters from sessionStorage (updated but not saved parameters, used in the changes parameters and net session changes report).

State and Component Functions

Functions	Arguments	Return	Description
setSelectedReports	reports: string[]	void	Updates the array of selected report IDs.
setPreviewReports	report: Report or null	void	Set the report to preview the null to close preview.
setIsEditingHeader	editing: boolean	void	toggles the header editing mode.
setHeaderInfo	Info: object	void	Updates reporter header information.
setPatientHistory	history: any[]	void	Updates the patient parameter history array.
toggleReportSelection	reportId: string	void	Adds or removes a report ID from selected reports array.
loadPatientHistory	id: string	void	Loads parameters from local storage, used for parameters report.
handlePreview	report: Preview	void	Set the report to preview in dialog.

downloadLatestForCurrentPatient	None	void	Downloads the most recent parameter entry as a JSON file for current patient.
generateReportContent	report: Report or null headerInfo: any	string	Generates the test for specified report time, includes header info and parameters where applicable.

2.3.6.5. EKG Stream

Component Props

Prop	**Description**
isDeviceConnected	Boolean value used to determine if the board is connected
channelData	EKG datapoints from pacemaker board.
onStartEKG	Callback function to begin stream.
onEndEKG	Callback unction to end stream.

State Variables

Variables	**Type**	**Initial**	**Description**
isStreaming	boolean	false	controls whether animation loop is rendering and scrolling.
selectedChannels	string[]	'Atrial' or 'Ventricular'	Array of channel names currently displayed (user can toggle).
gain	number[]	[1]	Zoom levels for all channels.
timescale	number[]	[1]	Time factor for slowing down or speeding up the stream.
gainInputs	atrial, ventricular	atrial: 1 ventricular: 1	Individual gain multipliers for each channel.

State and Component Functions

Functions	**Arguments**	**Return**	**Description**
handleStartStop	None	void	Toggles streaming on and off
handleReset	None	void	resets scroll offset to 0 (beginning of the waveform)
toggle channel	channel: string	void	adds/removes channel from selectedChannel array

2.3.6.6. About Page

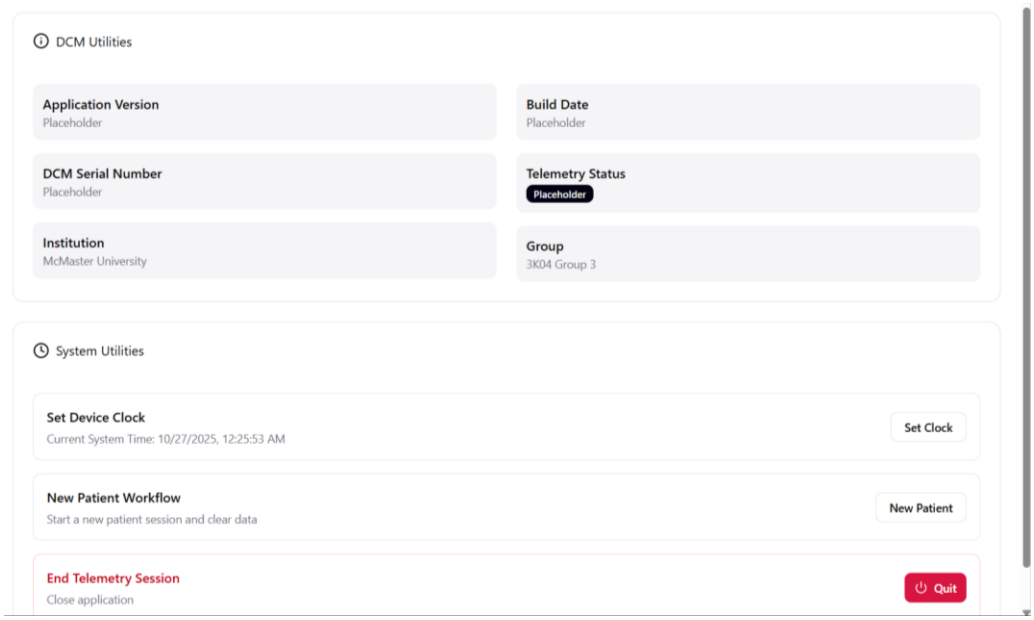


Figure 9: DCM About Page

Component Props

Prop	Description
savedUsers	Array of all registered users in the system (used for dev tools)
onDeleteUsers	Callback function to delete a user by username

State Variables

Variables	Type	Initial	Description
currentTime	Date	new Date()	Stores the information for the current time

State and Component Functions

Functions	Arguments	Return	Description
setCurrentTime	time: Date	None	Updates current time state variable
handleSetClock	None	void	Sets currentTime to Date()
handleQuitApplication	None	void	Logs application quit message

handleNewPatientFlow	None	void	Logs new patient workflow message

2.3.6.7. App (Main File)

The App loads all the components "*app.tsx*" file. This is the overall page, which is rendered. It includes the page header and all subtabs. When a tab is clicked, the corresponding component is rendered on the page in the dynamic flex box section labelled in Red.

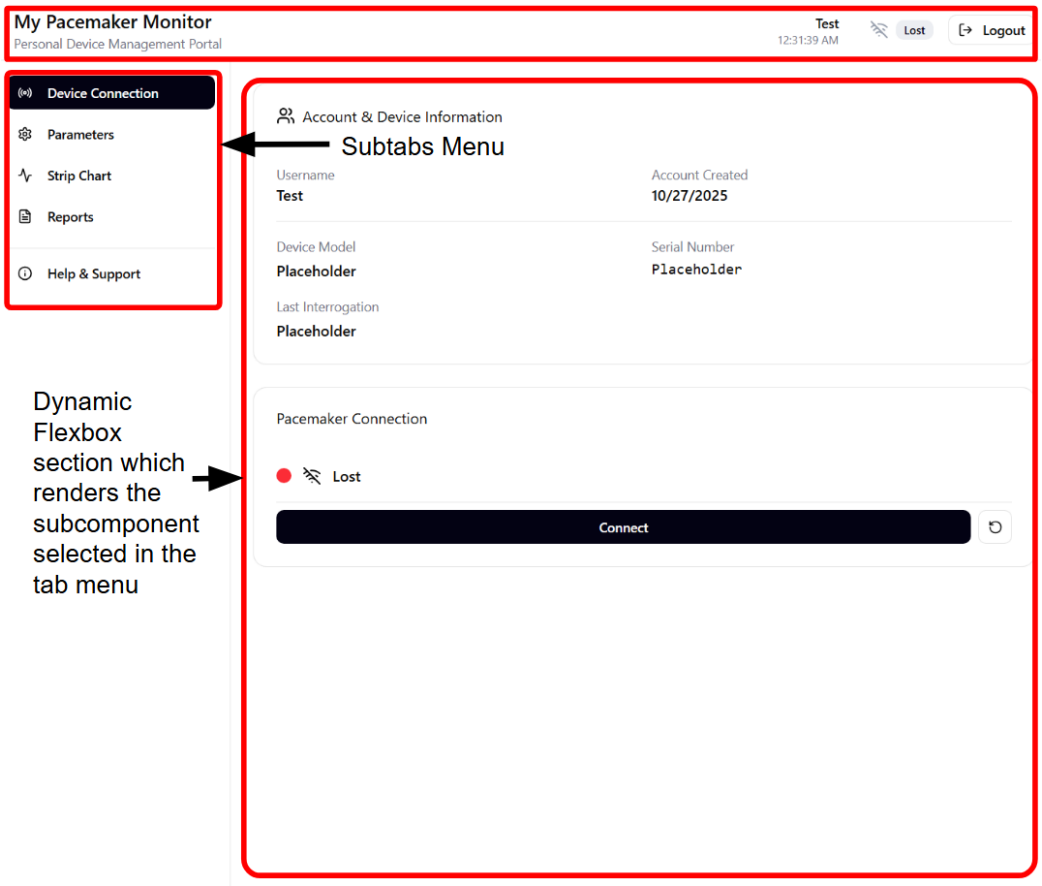


Figure 10: DCM App.tsx entry point annotated (Red lines are not in webpage)

State And Component Functions

Variable	Type	Initial	Description
isLoggedIn	boolean	false	Auth flag for showing login vs. app shell.

currentUser	string	""	Username of the signed-in user.
showCreateUser	boolean	false	Toggles the Create User screen.
selectedPatient	Patient | null	null	Active patient object tied to currentUser.
activeTab	"connection" | "parameters" | "egm" | "reports" | "about"	"connection"	Which main area is visible.
telemetryState	{ connectionState: "Connected" | "Lost" | "Out of Range" | "Noise"; isConnecting: boolean }	{ connectionState: "Lost", isConnecting: false }	Simulated device connection status + spinner state.
currentTime	Date	new Date()	Live clock shown in the header (updates each second).
savedUsers	User[]	From localStorage["dcm_users"] or a seeded demo user	All demo users with patientData and default parameters.
connectionState	ConnectionState	“”	Current pacemaker connection statue (either connected, noise, lost).
isConnecting	boolean	dales	Whether a connection attempt is in progress.
verificationStatus	VerificationStatus	{status: null, message: “”}	Status of last sent parameter verification (either pending, verified, failed).
ekgData	EKGData | null	null	Real-time EKG data points from atrial and ventricular channels.

Effects (Lifecycle)

Effect	Triggers/Dependencies	Purpose
Clock tick	runs once; interval every 1s	Updates currentTime. Cleans up timer on unmount.

Persist users	[savedUsers]	Writes savedUsers to localStorage["dcm_users"].
Select patient for currentUser	Should be [isLoggedIn, currentUser, savedUsers]	When logged in & currentUser changes, sets selectedPatient from savedUsers. *(Add deps to avoid rerun every render.)*

Websocket Message Handlers (Messages received from backend)

Message Type	Handler Logic	Description
CONNECT_RESPONSE	Updates connectionState and deviceInfo.	Processes connection status and device information from board
DISCONNECT_RESPONSE	Resets connectionState and deviceInfo.	Handles device disconnection to the pacemaker
PARAMETER VERIFICATION	Updates verificationStatus based on success or failure.	Shows verification results after sending parameters to the board
EKG_DATA	Appends data points to ekgData, maintains a 10-second rolling window.	Streams real-time cardiac signals for graph displays
LOAD_PARAMETER_RESPONSE	Executes stored callback, updated patient data.	Loads parameters from the board and passes it to parameter table to update user values.

Websocket Message Handlers (Messages sent to backend)

Message Type	Payload	Description
CONNECT_REQUEST	{type: “CONNECT_REQUEST”}	Initiate handshake.
DISCONNECT_REQUEST	{type: “DISCONNECT_REQUEST”}	Disconnect from board.
LOAD_PARAMETER_REQUEST	{type: “LOAD_PARAMETER_REQUEST”}	Request current board parameters.
EKG_START_REQUEST	{type: “EKG_START_REQUEST”}	Start EKG streaming.
EKG_STOP_REQUEST	{type: “EKG_STOP_REQUEST”}	Stop EKG streaming.

Local Storage Keys & Data

Key	Type	Description
dcm_users	string (JSON-encoded User[])	Persisted demo users and their patientData (including parameters). Seeded with a default if missing.
pacemakerParameters	timestamp: string, values	Patient parameter history, keyed by the patient ID
temporaryParameters	any	Parameters sent to pacemaker but not yet saved to patient record

2.3.7. Serial Communication in the DCM

2.3.7.1. Overview

Serial communication in the DCM is managed through a multilayer architecture that bridges the frontend with a python backend.

When the application starts, it establishes a websocket connection to the backend through the useEffect hook in App.tsx, the backend then initiates the serial connection to the pacemaker on COM4 at 115200 baud using the PySerial library. Communication flows in either direction through binary packets with specific header bytes.

2.3.7.2. Device connection and Identification

When the DCM connects to the pacemaker board, the DCM sends a handshake request by constructing a 125-byte packet with the function code [0x45]. It then asynchronously awaits a response packet [0x29] containing the device serial number and model name. Once, the response is received, the backend unpacks the binary data and converts it into a JSON object, it is then passed through the websocket to update the React frontend's state.

2.3.7.3. Parameter Sending

When the DCM needs to send parameters to the pacemaker, the frontend transmits a JSON message through the websocket. Upon detection of this message, the python backend process it through the mapData method. This extracts parameter values in a specific order, then constructs a 125-byte binary packet in little-endian format, with the function code [0x55].

2.3.7.4. Parameter Verification

The parameter verification operates through an echo mechanism where the pacemaker returns the received parameters in a binary packet with the function code [0x39]. The python backend continuously monitors incoming serial data through the awaitEchoedParameters coroutine, which reads the echoed parameters, unpacks the values then compares them against the original sent values. Once verified, the verify

Parameters method sends a status message to the frontend, allowing the UI to update the parameter verification state.

2.3.7.5. EKG Streaming

When the user initiates EKG streaming, the frontend sends an EKG start request, which triggers to start an EKG request loop. This request loop sends a 125-byte packet with the function code [0x36] every 0.1 ms. Upon each request, the pacemaker responds with a corresponding data packet containing the atrial signal, ventricular signal, and timestamp. The backend then unpacks the data and converts it to JSON format with x-y coordinate pairs for both atrial and ventricular signals. The system then accumulates the acquired data points in a react state to update the EKG stream.

2.3.7.6. Load Board Parameters

Loading the current pacemaker board parameters functions similarly to the device connection handshake. Where a request packet is sent with the function code [0x60] and awaits a 125-byte packet with the code [0x44]. Upon receiving board parameter packet, the backend unpacks the values to be passed to the frontend.

2.4. Assurance Cases

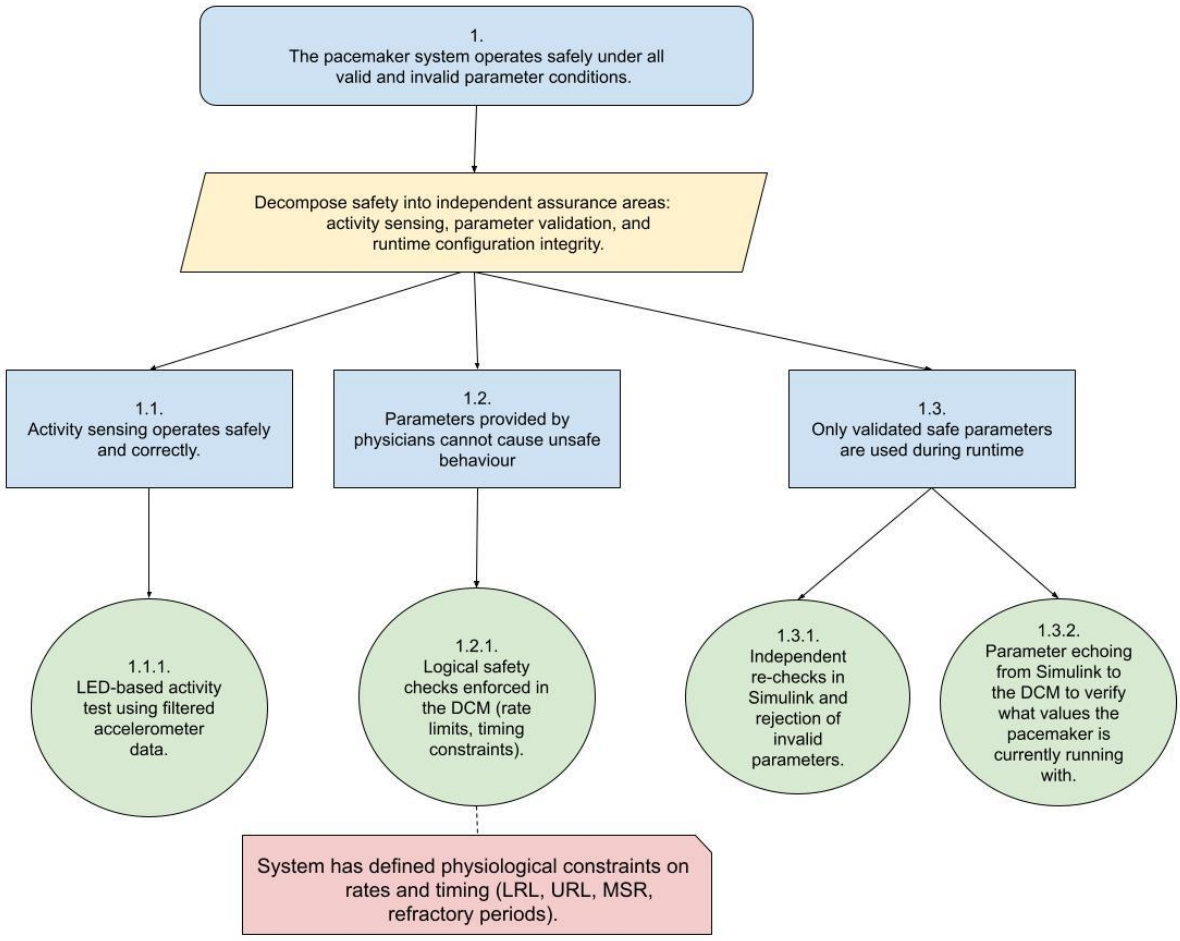


Figure 11: GSN Assurance Case Structure for the Pacemaker System

2.4.1. Accelerometer Verification

As shown in Figure 11, our assurance case is divided into three major components that together demonstrate safe pacemaker operation. The first part focuses on validating the accelerometer subsystem, which is directly responsible for activity detection. This links to the system requirement Activity Detection: “The system shall be able to detect instantaneous and prolonged physical movement of the wearer.” To ensure this requirement is being met, we implemented a simple verification mechanism using a green LED. The accelerometer measures raw acceleration along three axes, and our system computes the magnitude, applies a moving average for stability, and compares it to a

doctor-defined activity threshold. When this averaged value exceeds the threshold, the LED lights up. Because noise alone cannot cause a threshold crossing, the LED provides a reliable and intuitive confirmation that the accelerometer is functioning properly and satisfying the activity-detection requirement.

2.4.2. DCM Safety verification

The second part of the assurance case focuses on safe configuration of programmable parameters through the DCM. This relates directly to the Serial Communication requirement, which states that the device must "allow the DCM to set, store, and verify programmable parameters on the device," as well as the Pacing Pulse Characteristics and Mode-Specific Requirements, which depend on valid rate and timing inputs. To ensure these requirements are upheld, the DCM enforces a set of logical and physiological safety checks before any values are transmitted. Examples include preventing the lower rate limit from exceeding the upper rate limit, ensuring the maximum sensor rate lies between them, and verifying that timing intervals such as refractory periods and pulse widths fit within the heartbeat interval implied by the selected lower rate limit. These checks ensure that the system can only operate using medically realistic and internally consistent values, preventing unsafe pacing behaviours before they ever reach the pacemaker.

2.4.3. Simulink Parameter Verification

After parameters pass the DCM checks, the Simulink pacemaker controller performs a second independent verification step, also shown in Figure 11. This redundancy supports the System Purpose requirement by ensuring the pacemaker always operates using safe values when delivering therapy for bradycardia. The Simulink model repeats all necessary limit checks, validates timing relationships, and rejects any parameter set that violates physiological or logical constraints. If invalid values are detected, the pacemaker continues using the previously accepted configuration or defaults to safe initial parameters. This mechanism aligns with the Serial Communication requirement, which states that the system must "verify programmable parameters on the device." After the verification step, Simulink echoes the accepted parameters back to the DCM to confirm that both sides are synchronized. This prevents silent failures, supports consistent runtime behaviour, and ensures safe pacing delivery across all implemented modes (AOO, VOO, AAI, VVI, AOOR, VOOR, AAIR, VVIR, DDDR).

2.4.4. Assurance Cases Summary

Overall, the structure shown in Figure 11 combines accelerometer verification, DCM safety checks, redundant Simulink validation, and two-way confirmation between the modules. Together, these form a safety argument that directly supports critical requirements such as

Activity Detection, Pacing Pulse Characteristics, Serial Communication, and the general System Purpose. These assurance layers ensure the pacemaker always operates with safe, physiologically appropriate, and internally consistent settings, providing confidence that the system will behave reliably during normal use and whenever parameters are updated.

3. Part 2

3.1. Requirements Potential Changes

3.1.1. Simulink Requirements Potential Changes

3.1.1.1. *Addition of Rate-Adaptive Pacing Modes*

Requirement 2.2.1.2 will be extended to include rate-adaptive modes: AOOR, VOOR, AAIR, and VVIR. This requires modifying the mode-specific requirements (2.2.2.1-2.2.2.4) to incorporate sensor-driven rate adjustments:

- Asynchronous Rate-Adaptive Modes (AOOR/VOOR): Requirements 2.2.2.1 (AOO) and 2.2.2.2 (VOO) will change. In AOOR/VOOR, the software shall deliver pacing pulses at a rate determined by sensor input, adjusting between the LRL and the Maximum Sensor Rate, without sensing.
- Inhibited Rate-Adaptive Modes (AAIR/VVIR): Requirements 2.2.2.3 (AAI) and 2.2.2.4 (VVI will change. In AAIR/VVIR, the software shall deliver a pacing pulse when the intrinsic rate is less than the sensor-indicated rate while still inhibiting based on sensed events outside their refractory period.

These changes introduce the need for new programmable parameters:

Table 9: New Parameters for Rate-Adaptive Pacing Modes

New Parameter	**Description**	**Nominal Value (Example)**	**Unit**
Maximum Sensor Rate	The maximum pacing rate allowed because of sensor control.	120	ppm
Activity Threshold	The level of sensor output required to affect the pacing rate.	Med	N/A

Response Factor	Determines the rate of change in pacing rate for a given activity level.	8	N/A
Reaction Time	Time required for the rate to increase from LRL to MSR at maximum activity.	30	seconds
Recovery Time	Time required for the rate to decrease from MSR to LRL when activity ceases.	5	minutes

3.1.1.2. *Evolution of Pacing Pulse Characteristics*

While requirement 2.2.1.2, specifying programmable amplitude and width, remains the same, the context changes. In these new modes, the frequency of these pulses becomes dynamic unlike the fixed-rate logic of Deliverable 1 modes.

3.1.1.3. *Evolution of Sensing Capability*

Requirement 2.2.1.3 (Sensing Capability) for intrinsic cardiac signals remains for inhibited modes (AAIR, VVIR). However, it must be expanded to include:

- A new requirement to sense physical activity using an appropriate sensor.
- A requirement to process this activity sensor data to derive a relevant metric for rate adaptation.

3.1.1.4. *Evolution of Hardware Hiding*

Requirement 2.2.1.4 (Hardware Hiding) must be extended. In addition to abstracting the pacing and cardiac sensing I/O, the design must now also abstract the interface to the physical activity sensor. The rate-adaptive logic should depend on a logical activity level signal rather than the raw sensor output.

3.1.1.5. *New Communication Requirements*

Deliverable 2 requires a communication interface through UART within the Simulink model, introducing requirements not present in Deliverable 1:

- The software shall be able to receive and apply new parameter values including those for rate adaptation transmitted from the DCM via UART.
- The software shall be able to transmit data, such as sampled electrogram signals and potentially status information, to the DCM upon request or periodically.

3.1.2. DCM Requirements Potential Changes

3.1.2.1. *Support for Rate-Adaptive Modes (AOOR, VOOR, AAIR, VVIR)*

Change: Extend the DCM to expose, validate, store, and transmit parameters needed for rate-adaptive pacing.

Why: Deliverable 2 requires adding AOOR/VOOR/AAIR/VVIR, with sensor-driven rate adaptation and serial comms to/from the pacemaker. The DCM must therefore manage the full parameter set and communicate it reliably.

New/updated DCM requirements:

- DCM-REQ-RA-1: The DCM shall present UI controls for Maximum Sensor Rate (MSR), Activity Threshold, Response Factor, Reaction Time, and Recovery Time, in addition to existing LRL/URL and amplitude/width fields. (UI + persistence)
- DCM-REQ-RA-2: The DCM shall validate each RA parameter against allowed domains before enabling “Save/Transmit”. (Validation)
- DCM-REQ-RA-3: The DCM shall transmit RA parameters to the device and verify storage (read-back) after write. (Comms + verification)

3.1.2.2. *Programmable Parameter Domains & Tolerances*

Change: Implement the Deliverable-2 programmable ranges, increments, and nominal tolerances in the DCM validator so that only enforceable values are sent.

Why: D2 specifies concrete domains for amplitude, pulse width, and sensitivity; the DCM must guard entry and reflect step sizes.

3.1.2.3. *Serial Communication (Write/Read-back + Egrams)Programmable Parameter Domains & Tolerances*

Change: Implement UART comms for parameter write/read and streaming egrams; expose connect/disconnect, retry, and status in the UI.

Why: D2 Part 3 mandates TX/RX of parameters and egram display from the device.

3.1.2.4. *Egram Data Structures*

Change: Define a typed in-memory and persisted structure for sampled egram points, sampling rate, channel metadata, and ring buffer size.

Why: D1 asked you to “develop and document appropriate data structures for egram data”; D2 requires serial display.

- DCM-REQ-EGR-1: { t0: ms, fs: Hz, channel: 'A'|'V'|'AV', samples: Float32Array } with max duration & backpressure strategy (drop oldest).

3.1.2.5. *Communication Interface Changes*

Transport: UART per course docs; implement SetParameters, GetParameters, GetEgram(A/V/both), GetStatus.

Atomicity: Write parameters in a single frame where possible to avoid partial updates; fall back to chunking with sequence IDs.

Verification: Mandatory read-back after every write; DCM blocks UI until verified or timed-out.

Error Handling: Timeouts, checksum fail, NAK → display actionable messages and allow retry, keeping last good config.

3.1.2.6. *Information Hiding (DCM side)*

Serial Abstraction: Hide UART specifics behind a CommsService so React components receive logical events (connected, parametersRead, egramFrame). This mirrors the Simulink-side hardware hiding principle in D1.

Mode Policies: Encapsulate per-mode field enablement/visibility & validation rules in a ModePolicy map, not in the UI layer.

3.2. Design Decision Potential Changes

3.2.1. Simulink Design Decision Potential Changes

Table 10: Deliverable 2 Anticipated Design Decision Changes for Simulink Logic

Decision ID	Current Implementation	Anticipated Change	MIS/MID
1.	Static Parameter Storage: Programmable parameters are fixed as Constant blocks within the Hardware Input Mapping module (1.1).	Dynamic Parameter Update: Constants must be replaced with access methods linked to a new serial communication	This is a change to the MID (implementation) of Hardware Input Mapping (1.1). The module's external MIS remains the same.

		module to receive runtime updates from the DCM.	
2.	Digital Sensing Method: Cardiac sensing uses a calculated PWM reference voltage feeding a comparator circuit on the shield to produce a simple digital event flag.	Activity Sensing: The system must incorporate logic to read and process raw or filtered data from the physical activity sensor.	This affects the MID of Input Processing (1.2) and adds a new activity sensor input to Hardware Input Mapping (1.1).
3.	Asynchronous Rate Calculation: The interval calculation for AOO/VOO relies on averaging LRL and URL.	LRL-Only Simplification: The calculation logic should be simplified to use only the LRL to calculate the pacing interval.	This happens entirely within the MID of the Input Processing (1.2). The module's output remains unchanged.
4.	Electrogram/Status Data Handling: The current design outputs are only control signals; there is no logic for collecting or outputting diagnostic data.	Electrogram Buffering and Packaging: New logic must be added to serialize electrogram data and status information for transmission to the DCM.	This introduces new functions in the MID of a new communication module and requires new ports on the MIS of subroutines (1.3) and hardware output pin mapping (1.4).

3.2.2. DCM Design Decision Potential Changes

Decision ID	Current Implementation	Anticipated Change	Rationale
1.	User persistence local: localStorage (`dcm_users`) with plaintext demo passwords	User Persistence web Backend auth + DB (tokens), or WebAuthn for clinic devices	Security, multi-device access, and audit.
2.	Session State: In-memory only	Refresh tokens or secure cookies	Durable sessions rather than having a system that can be easily erased and corrupted.

3.	localStorage Conversion: Direct JavaScript to JSON conversion.	localStorage handling: Wrap read/writes in try/catch; validate shapes before using	Prevent crashes on malformed JSON or quota errors
4.	Parameter change values Currently, using an up-and-down button to change values by pressing it each time.	Accessible Parameter Changing Implementing the button to change if it is held dynamically	It can be difficult and tiring for people with limited hand mobility to repeatedly press the button.
5.	Parameter Listing: Currently listing all the parameters that can be changed regardless of the mode.	Dynamic Parameter Values: Dynamic implementation of the parameters so only the ones required to change will appear for the modes	TO easily determine which values to change and to send the data

3.3. Module Description

3.3.1. (1.1.1) Hardware Input Mapping

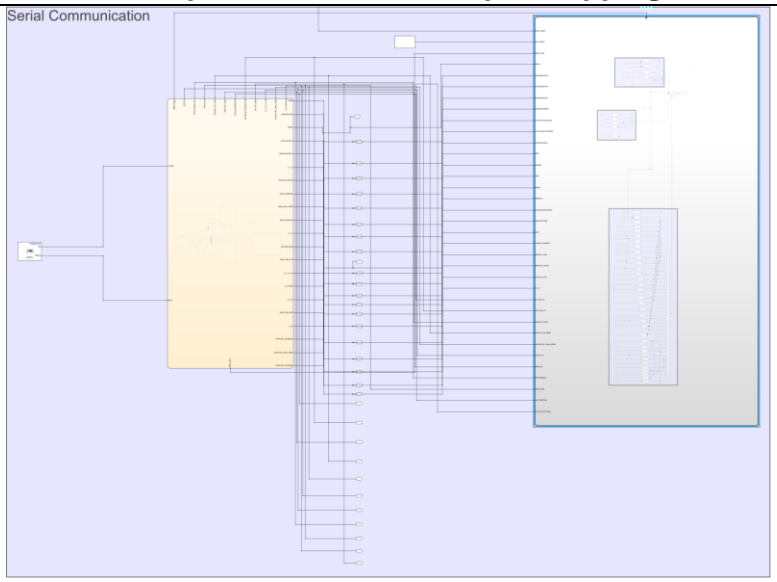
<table>
<tr><th colspan="4">Component: Hardware Input Mapping</th></tr>
<tr><td colspan="4"> 

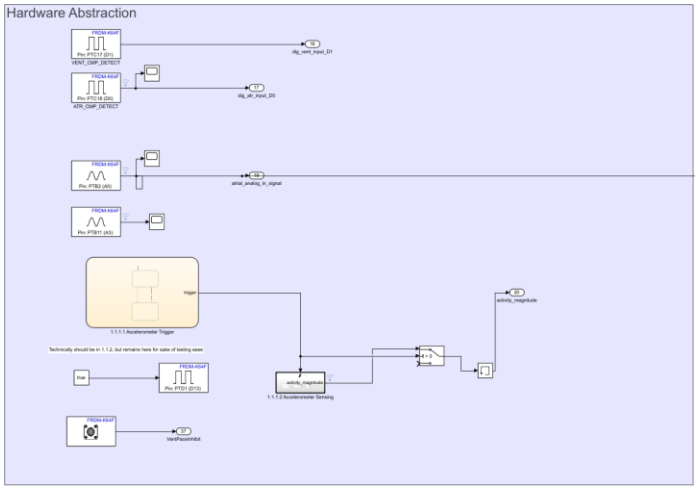
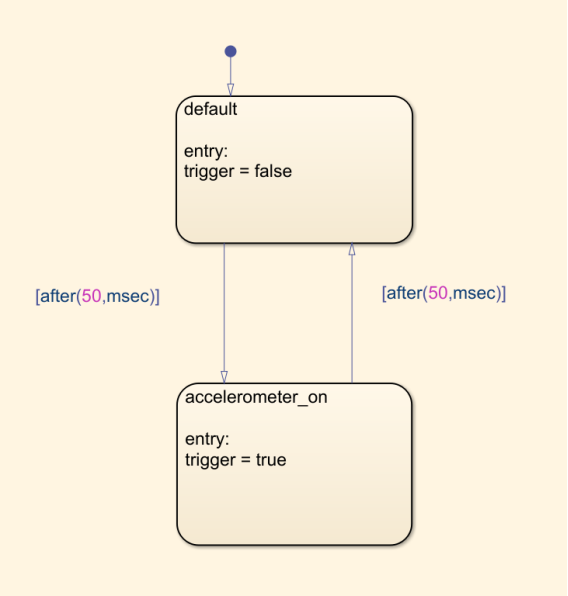
Figure 114: Serial Communication Input Component
 

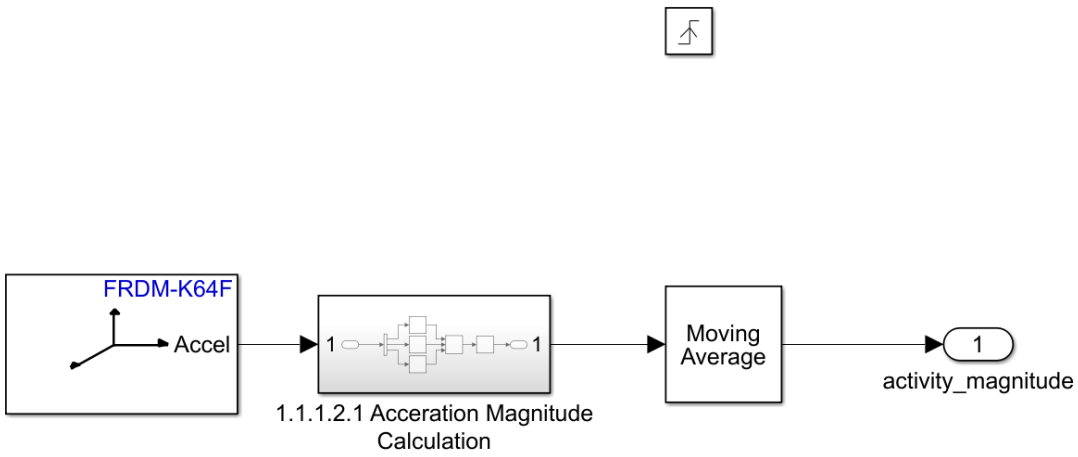
Figure 12 5: Hardware Abstraction Component</td></tr>
<tr><th>Purpose</th><th>Key Functions (Public/Internal)</th><th>Global/State Variables</th><th>Interactions with other Components</th></tr>
<tr><td>Abstracts physical input pins.</td><td>Public:
Provides Simulink signals for mode, rate limits, amplitudes, pulse widths, sensitivities, etc.</td><td>N/A</td><td>Used in Input Processing (1.1.2) and propagates to signals to Subroutines (1.1.3).</td></tr>
</table>

	Internal: Uses FRDM-K64F library blocks configured for specific pins and uses blocks for constants.		

3.3.1.1. (1.1.1.1) Accelerometer Trigger

Component: Accelerometer Trigger			
			
Purpose	**Key Functions (Public/Internal)**	**Global/State Variables**	**Interactions with other Components**
Acts as an explicit periodic trigger of subsystem 1.1.1.2 in order correctly poll the accelerometer.	**Public:** Inputs: Outputs: trigger **Internal:** States: default, accelerometer_on	Variables are associated with active states (default, or accelerometer_on). **State Variables:** trigger **Local Variables:**	Externally triggers subsystem 1.1.1.2 whenever the trigger is set to true.

3.3.1.2. (1.1.1.2) Accelerometer Sensing

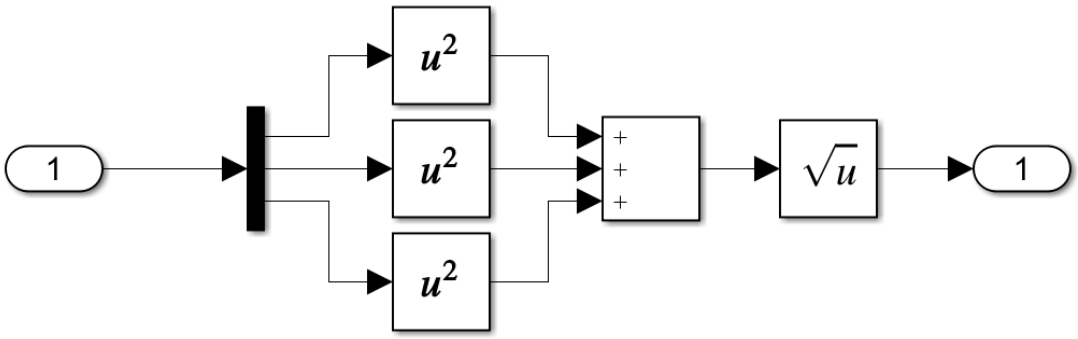
<table>
<tr><th colspan="4">Component: Accelerometer Sensing</th></tr>
<tr><td colspan="4"> 
</td></tr>
<tr><th>Purpose</th><th>Key Functions (Public/Internal)</th><th>Global/State Variables</th><th>Interactions with other Components</th></tr>
<tr><td>Implements a moving average block to sample the accelerometer correctly and attenuate sensor noise.</td><td>Public:
<u>Inputs:</u>
FXOS8700 6-Axes Sensor

<u>Outputs:</u>
activity_magnitude

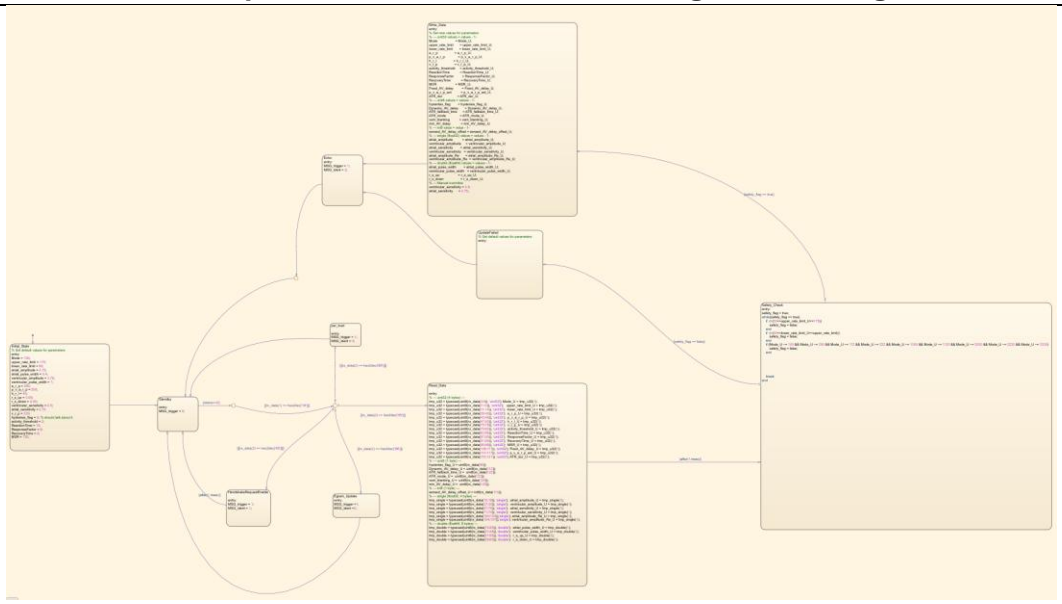
Internal:
Moving average DSP toolbox block.</td><td>Not a Stateflow diagram, so and no internally contained Stateflows. No subsystem-specific internal variables.
State Variables:

Local Variables:</td><td>Output activity_magnitude used for activity detection to inform rate adaptive pacing modes.</td></tr>
</table>

3.3.1.2.1. (1.1.2.1.1) Acceleration Magnitude Calculator

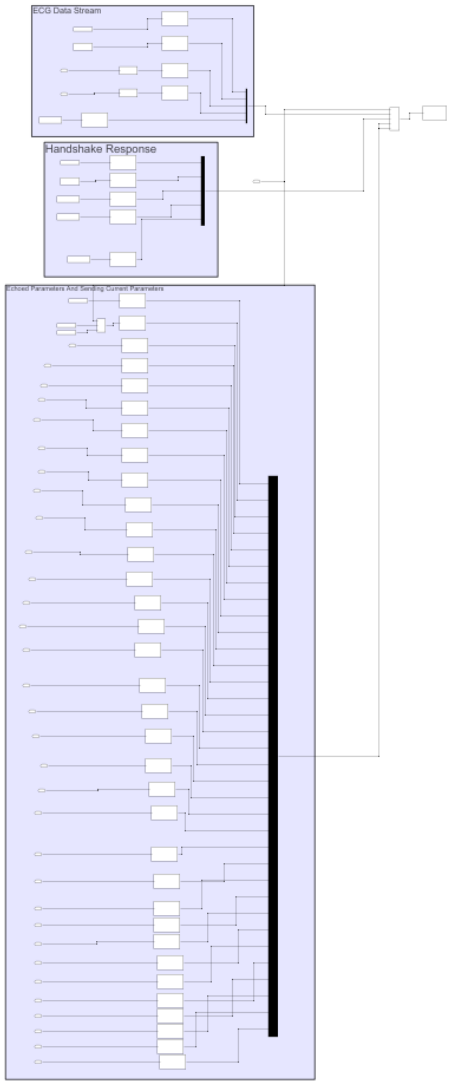
Component: Accelerometer Sensing			
			
Purpose	**Key Functions (Public/Internal)**	**Global/State Variables**	**Interactions with other Components**
Calculates the magnitude of the acceleration vector output by the accelerometer in subsystem 1.1.1.2.	**Public:** Inputs: FXOS8700 6-Axes Sensor data. Outputs: acceleration_magnitude **Internal:**	Not a Stateflow diagram, so and no internally contained Stateflows. No subsystem-specific internal variables. **State Variables:** **Local Variables:**	Computes the acceleration magnitude for all samples output by the accelerometer in subsystem 1.1.1.2. Used by a downstream moving average block to attenuate transient sensor noise.

3.3.1.3. (1.1.1.3) Received Serial Message Processing

Component: Received Serial Message Processing			
			
Purpose	**Key Functions (Public/Internal)**	**Global/State Variables**	**Interactions with other Components**
Process incoming UART RX data from DCM, and assign corresponding model data. Verifies byte stream length, and maps data to internal firmware variables.	**Public:** <u>Inputs:</u> rx_data status <u>Outputs:</u> MSG_trigger ATR_dur ATR_fallback_time ATR_mode Dynamic_AV_delay Fixed_AV_delay atrial_amplitude_Re ventricular_amplitude_Re min_AV_delay p_v_a_r_p_ext sensed_AV_delay_offset vent_blanking MSR ReactionTime Mode	**State Variables:** Active state within the Stateflow chart **Local Variables:** ATR_dur_U ATR_fallback_time_U ATR_mode_U Dynamic_AV_delay_U Fixed_AV_delay_U atrial_amplitude_Re_U ventricular_amplitude_Re_U min_AV_delay_U p_v_a_r_p_ext_U sensed_AV_delay_offset_U vent_blanking_U MSR_U ReactionTime_U Mode_U RecoveryTime_U	Writes all Programmable Parameters (2.3.2) to the firmware, and processes all requests for data by the DCM (2.3.7).

	RecoveryTime ResponseFactor a_r_p activity_threshold atrial_amplitude atrial_pulse_width atrial_sensitivity h_r_l hysteresis_flag lower_rate_limit p_v_a_r_p r_s_down r_s_up upper_rate_limit v_r_p ventricular_amplitude ventricular_pulse_width ventricular_sensitivity MSG_ident **Internal:** Stateflow chart with states (Initial_State, Standby, par_load, Egram_update, HandshakeRequestEnable, Read_Data, Safety_Check, UpdateFailed, Write_Data, Echo)	ResponseFactor_U a_r_p_U activity_threshold_U atrial_amplitude_U atrial_pulse_width_U atrial_sensitivity_U h_r_l_U hysteresis_flag_U lower_rate_limit_U p_v_a_r_p_U r_s_down_U r_s_up_U upper_rate_limit_U v_r_p_U ventricular_amplitude_U ventricular_pulse_width_U ventricular_sensitivity_U	

3.3.1.4. (1.1.1.4) Output Serial Message Processing

Component: Output Serial Message Processing			
			
Purpose	**Key Functions (Public/Internal)**	**Global/State Variables**	**Interactions with other Components**
Processes the message identifier received by 1.1.1.3, and sends the corresponding payload to the outgoing TX byte stream.	**Public:** Inputs: Atrial_signal Vent_signal MSG_ident Mode UpperRateLimit LowerRateLimit	Not a Stateflow diagram, and no internally contained Stateflows. No subsystem-specific internal variables.	Uses current internal data and message identifiers from 1.1.1.3 to inform the outgoing byte stream payload to

	AtrialAmplitude AtrialPulseWidth VentricularAmplitude AtrialSensitivity ARP PVARP HRL RSUp RSDown VentricularSensitivity HysteresisFlag VRP Activity_threshold Reaction_time Response_factor Recovery_time m_s_r atr_temp_re vent_amp_re fixed_AV_delay dynamic_AV_delay sensed_AV_delay_offset pvarp_ext ATR_dur ATR_fallback ATR_mode Vent_blanking min_dyn_AV_delay <u>Outputs:</u> **Internal:**	**State Variables:** **Local Variables:**	be received by the DCM UART receive.

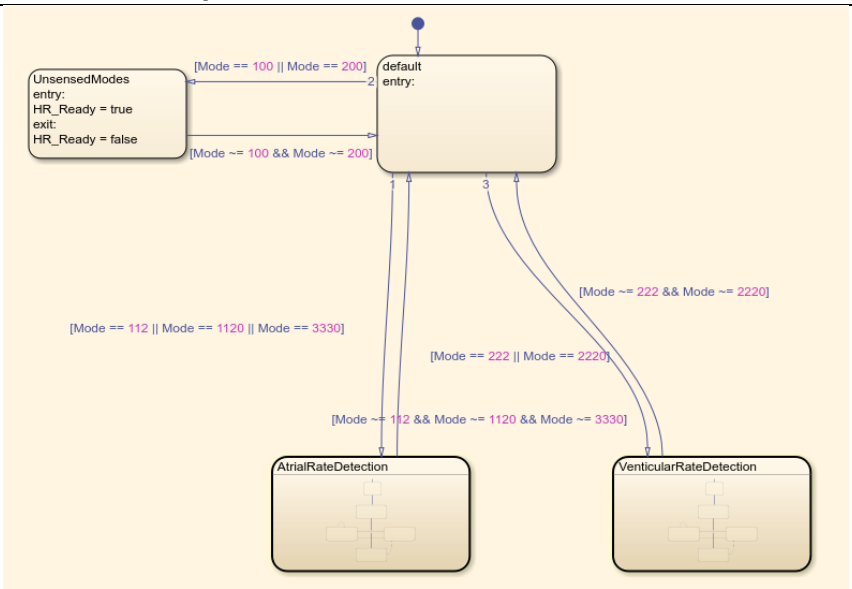
3.3.2. (1.1.2) Input Processing

<table>
<tr><th colspan="4">Component: Input Processing</th></tr>
<tr><td colspan="4"> 

Figure 136: Input Processing Subsystem</td></tr>
<tr><th>Purpose</th><th>Key Functions (Public/Internal)</th><th>Global/State Variables</th><th>Interactions with other Components</th></tr>
<tr><td>Contains all subsystems and Stateflow diagrams used to compute inputs needed for pacemaker control logic in downstream components.</td><td>Public:
<u>Inputs:</u>
Mode, UpperRateLimit, Lower Rate Limit, AtrialAmplitude, AtgrialPulseWidth, VentricularAmplitude, VentricularPulseWidth, AtrialSensitivity, ARP, PVARP, Hysteresis Rate Limit, Rate Smoothing limit, Ventricular Sensitivity, VRP, Digital Ventricle Chamber, Digital Atrial Chamber, Hysteresis Flag, AtrialAnalogInSignal, activity_magnitude, ReactionTime, ResponseFactor, RecoveryTime, MSR, ActivityThreshold, atrial_amp_Re,</td><td>Not a Stateflow, just a subsystem containing other Stateflows and control systems. No global variables accessible by the rest of the model, or internal state variables</td><td>Outputs signals to be used by Subroutines (1.1.3) and Hardware Output Mapping (1.1.4). Passes through signals needed for duty cycle calculation, and pacemaker control logic/timing.</td></tr>
</table>

	vent_amp_Re, fixedAVdelay, dynAVdelay, sensedAVDelayOffset, p_v_a_r_p_ext, ATR_dur, ATR_fallback, ATR_mode, vent_blanking, min_Avdelay, VentPaceInhibit <u>Outputs</u> mode, upper_rate_limit, lower_rate_limit, atrial_amplitude, atrial_pulse_width, ventricular_amplitude, ventricular_pulse_width, f_aoolnterval, f_voolnterval, ventricular_sensitivity, v_r_p, r_s_up, atrial_sensitivity, a_r_p, r_s_down, P_v_a_r_p, dig_vent_outputD1, dig_atr_outputD0, wait_time, h_r_l, hysterisis_flag, avg_heart_rate, inst_heart_rate, previous_inst_heart_rate, hr_ready, wait_time_lrl, wait_time_hys, ActivityIndex, ActivityThresh, Recovery_Time, f_voornterval, MSRate, RF, Reaction_Time, xxirp, wait_time_xxir, Current_Rate, Atr_Amp_Reg, Vent_Amp_Reg, fixed_AV_delay, dyn_AV_delay, sensed_AV_offset, PVARP_exten, ATR_DUR, ATR_FALL, ATR_MODE, ventBlanking, min_AV_delay, Hdoff, AV_Delay, vent_pace_inhibit **Internal:** Contains subroutines and modules for calculating time-varying signals and properties from heart.	(other than those from contained subsystems).	

3.3.2.1. (1.1.2.1) Pulse Period Determination

<table>
<tr><th colspan="4">Component: Pulse Period Determination</th></tr>
<tr><td colspan="4"> 
</td></tr>
<tr><td colspan="4">Figure 147: Pulse Period Determination Component</td></tr>
<tr><td>Purpose</td><td>Key Functions (Public/Internal)</td><td>Global/State Variables</td><td>Interactions with other Components</td></tr>
<tr><td>Monitors heart chamber voltage, and maintains an active counter to compute the time delta between two pulses.</td><td>Public:
<u>Inputs:</u>
Mode, AtrialPulseDetected, VentricularPulseDetected, VRP, ARP
<u>Outputs:</u>
time_between_pulses, pulse_finished, HR_ready
Internal:
Uses Mode to identify which chamber to sense, and computes pulsing period using refractory period and PulseDetected parameters.</td><td>State Variables:
Active state within the Stateflow chart.

Local Variables:
HR_ready</td><td>Outputs a time in milliseconds that is used by Rate Calculation (1.1.2.2) for its internal processes. Also outputs a flag, HR_ready, that informs the Pacemaker Control (1.1.3.2) when to begin control logic.</td></tr>
</table>

4.3.2.1.1. (1.1.2.1.1) Atrial Rate Detection

Component: Atrial Rate Detection

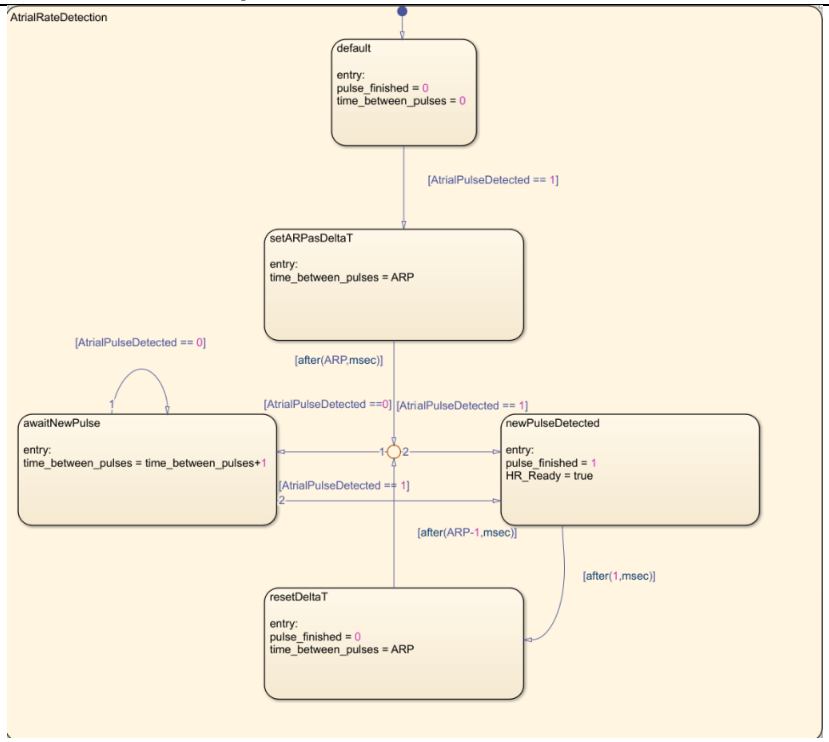
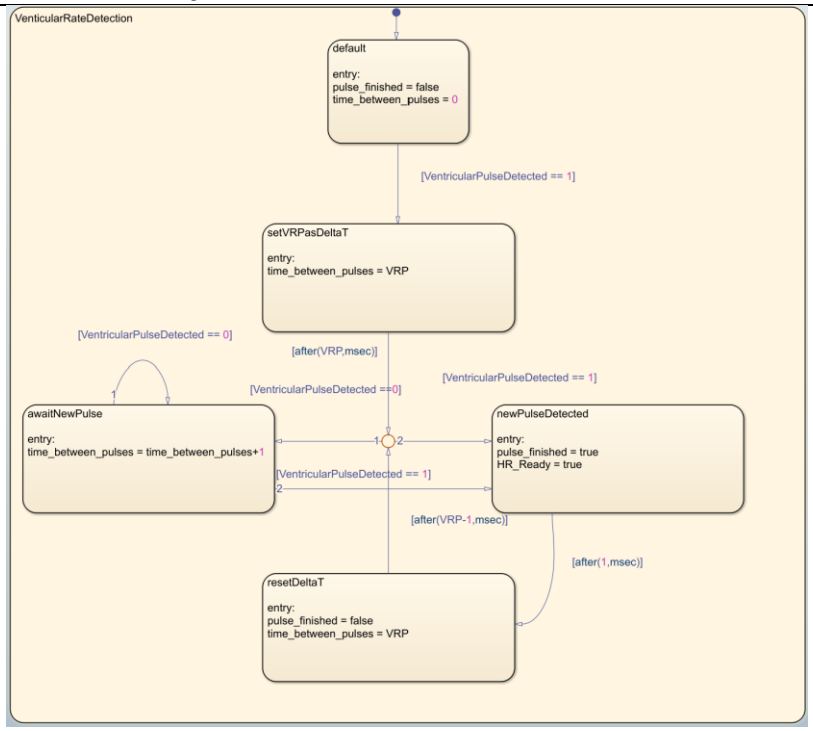


Figure 158: Atrial Rate Detector Component

Purpose	**Key Functions (Public/Internal)**	**Global/State Variables**	**Interactions with other Components**
Detects valid atrial pulses and refractory period, measures time between pulses, and signals for HR calculation through a flag. Encapsulates the state logic for pulse validation and timing	**Public:** Inputs: Mode, AtrialPulseDetected, ARP Outputs: time_between_pulses, pulse_finished, HR_Ready **Internal:** Stateflow chart with states (default, setARPasDeltaT, awaitNewPulse, newPulseDetected, resetDeltaT)	**State Variables:** Active state within the Stateflow chart **Local Variables:** time_between_pulses, pulse_finished	Is used by Rate Calculation system (1.1.2.2) to calculate the atrial rate used for calculating both the instantaneous and average heart rate.

4.3.2.1.2. (1.1.2.1.2) Ventricular Rate Detection

Component: Ventricular Rate Detection			
 *Figure 169: Ventricular Rate Detection Component*			
Purpose	**Key Functions (Public/Internal)**	**Global/State Variables**	**Interactions with other Components**
Detects valid ventricular pulses and refractory period, measures time between pulses, and signals for HR calculation through a flag. Encapsulates the state logic for pulse validation and timing	**Public:** Inputs: Mode, AtrialPulseDetected, VRP Outputs: time_between_pulses, pulse_finished, HR_Ready **Internal:** Stateflow chart with states (default, setVRPasDeltaT, awaitNewPulse, newPulseDetected, resetDeltaT)	**State Variables:** Active state within the Stateflow chart **Local Variables:** time_between_pulses, pulse_finished	Is used by Rate Calculation system (1.1.2.2) to calculate the ventricular rate used for calculating both the instantaneous and average heart rate.

3.3.2.2. (1.1.2.2) Rate Calculation

Component: Rate Calculator			

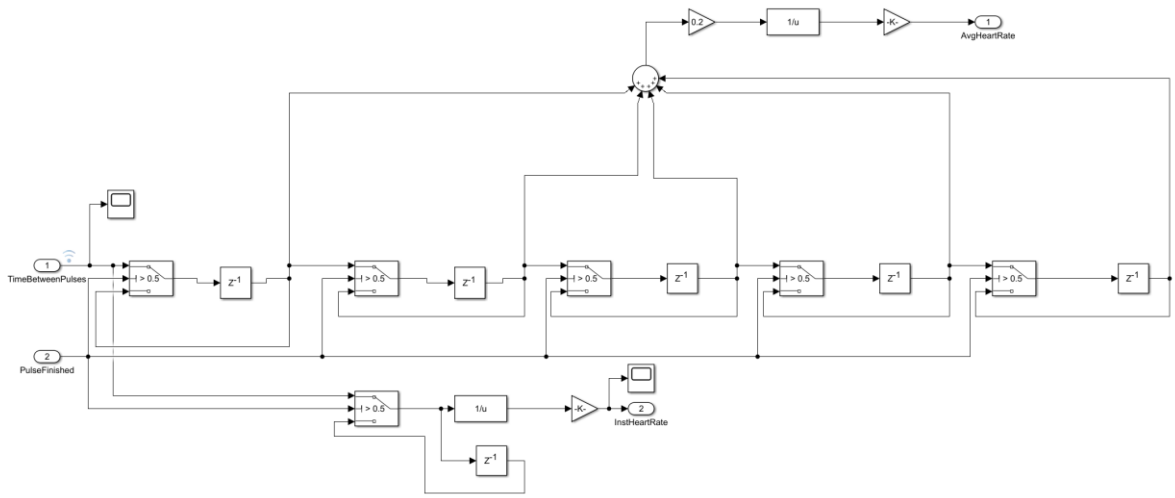
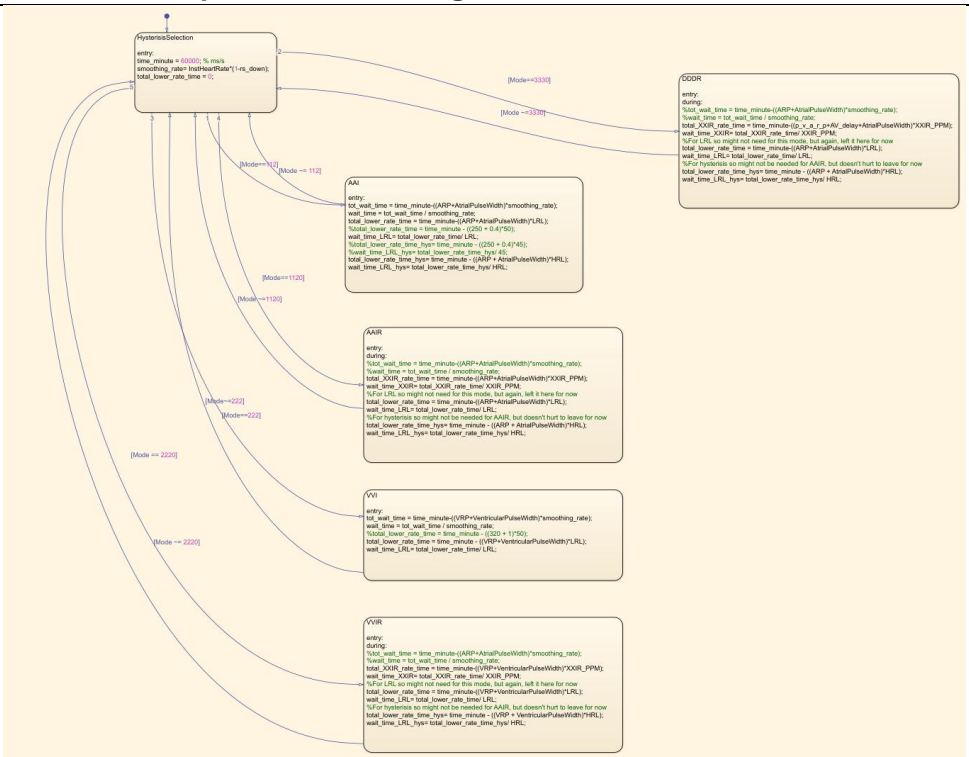



Figure 1710: Rate Calculator Component			
Purpose	**Key Functions (Public/Internal)**	**Global/State Variables**	**Interactions with other Components**
Calculates instantaneous heart rate and average heart rate for comparisons to the LRL.	**Public:** Inputs: TimeBetweenPulses PulseFinished flag Outputs: AvgHeartRate, InstHeartRate, PrevInstHeartRate **Internal:** Uses a series of Delay blocks and Switch blocks to implement a moving average filter based on the last 5 valid pulse intervals.	State is stored within the five Delay blocks holding previous interval values.	Is Used By: Subroutines (1.1.3) to make comparisons to LRL.

3.3.2.3. (1.1.2.3) Watchdog Timer For Pulse Detection

Component: Watchdog Timer for Pulse Detection			
 *Figure 1811: Watchdog Timer for Pulse Detection Component*			
Purpose	**Key Functions (Public/Internal)**	**Global/State Variables**	**Interactions with other Components**
Calculates the appropriate waiting interval before the next pace, considering mode (AAI/VVI), heart rate, refractory periods, and pulse widths.	**Public:** Inputs: Mode, LRL, HRL, HFlag, VRP, XXIR_PPM, ARP, p_v_a_r_p, AV_delay, AtrialPulseWidth, VentricularPulseWidth, rs_down, InstHeartRate Outputs: wait_time, wait_time_LRL, wait_time_LRL_hys, wait_time_XXIR **Internal:** Uses entry actions in AAI, VVI, AAIR, VVIR, and DDDR to calculate wait_time and wait_time_LRL based on input parameters.	**State Variables:** Active state within the Stateflow chart. **Local Variables:** tot_wait_time, time_minute, smoothing_rate, total_XXIR_rate_time, total_lower_rate_time	Receives direct instantaneous heart rate from rate calculation subroutine. Outputs a watchdog timer to be used by AAI (1.1.3.2.3), VVI (1.1.3.2.4), AAIR (1.1.3.2.7), VVIR (1.1.3.2.8), and DDDR (1.1.3.2.9). It is used to determine if heartrate drop has exceeded threshold for Rate Smoothing/Lower Rate Limit/Hysteresis Limit

3.3.2.4. (1.1.2.4) Calculate Unsensed Pace

Component: Unsensed Pace Calculator

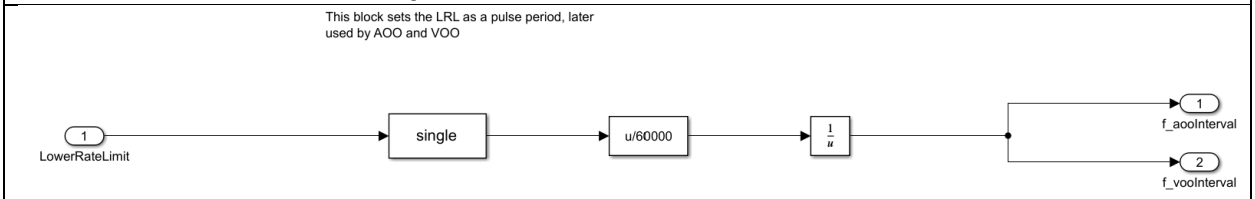
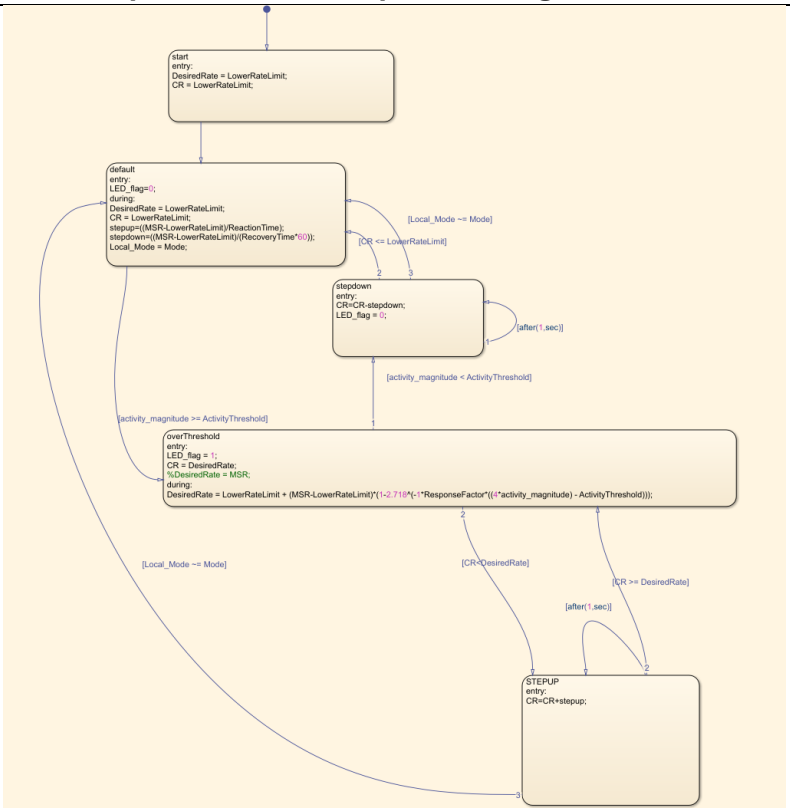


Figure 1912: Unsensed Pace Calculator Component

Purpose	**Key Functions (Public/Internal)**	**Global/State Variables**	**Interactions with other Components**
Converts the lower rate limit (programmable parameter) to a time representation in milliseconds/beat.	**Public:** Inputs: LowerRateLimit **Internal:** Uses mathematical operations to convert a heart rate in bpm to ms/beat.	None. Stateless subsystem (not a Stateflow), and contains no global variables, only outputs a signal.	Used by AOO (1.1.3.2.1), VOO (1.1.3.2.2), AOOR (1.1.3.2.5), and VOOR (1.1.3.2.6) to define base pacing period.

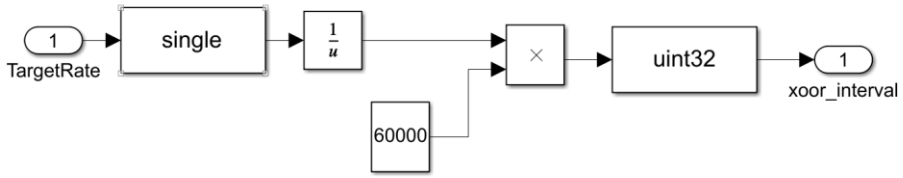
3.3.2.5. (1.1.2.5) Rate Adaptive Pacing Calculation

<table>
<tr><th colspan="4">Component: Rate Adaptive Pacing Calculation</th></tr>
<tr><td colspan="4"> 
</td></tr>
<tr><th>Purpose</th><th>Key Functions (Public/Internal)</th><th>Global/State Variables</th><th>Interactions with other Components</th></tr>
<tr><td>Dynamically computes the desired rate based off of current activity level.</td><td>Public:
<u>Inputs:</u>
activity_magnitude, MSR, LowerRateLimit, ResponseFactor, ActivityThreshold, Mode, RecoveryTime, ReactionTime
<u>Outputs:</u>
LED_flag, Desired Rate, Current Rate

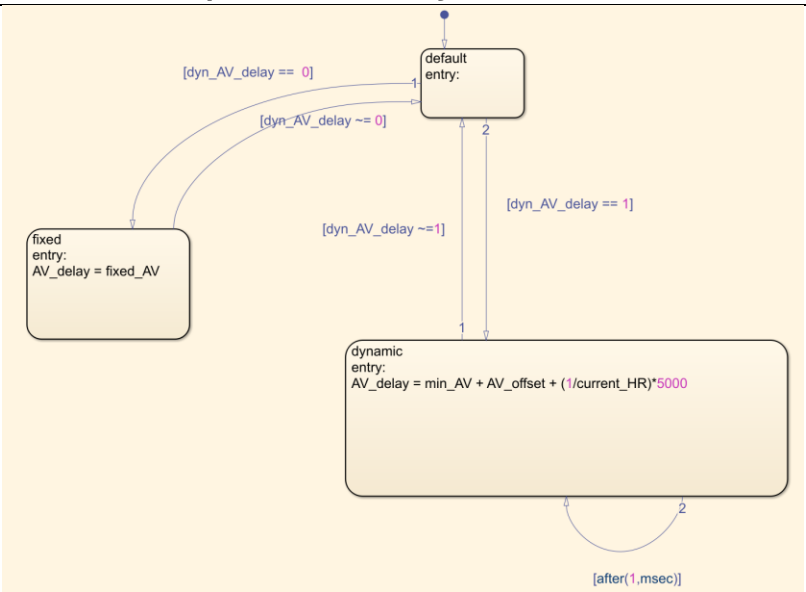
Internal:
Stateflow chart with states (start, default, stepdown, overThreshold,STEPUP)</td><td>State Variables:
Active state within the Stateflow chart.

Local Variables:
stepdown, stepup, Local_Mode</td><td>Dynamically determines required rate, and desire rate, given the current activity of the wearer. Outputs a Boolean LED_flag to turn the green board LED on when the activity threshold is surpassed.</td></tr>
</table>

3.3.2.6. (1.1.2.6) Rate Adaptive Pacing Interval Calculation

Component: Rate Adaptive Pacing Interval Calculation			
			
Purpose	**Key Functions (Public/Internal)**	**Global/State Variables**	**Interactions with other Components**
Converts the desired rate dynamically output by the Rate Adaptive Pacing Calculator (1.1.2.6) to a time interval in ms/pulse.	**Public:** Inputs: Target Rate Outputs: xoor_interval **Internal:**	None. Stateless subsystem (not a Stateflow), and contains no global variables, only outputs a signal **State Variables:** **Local Variables:**	Output xoor_interval is directly fed into Subroutines (1.1.3) and used by rate adaptive modes AOOR (1.1.3.2.5) and VOOR (1.1.3.2.6) for pacing logic.

3.3.2.7. (1.1.2.7) AV Delay Determination

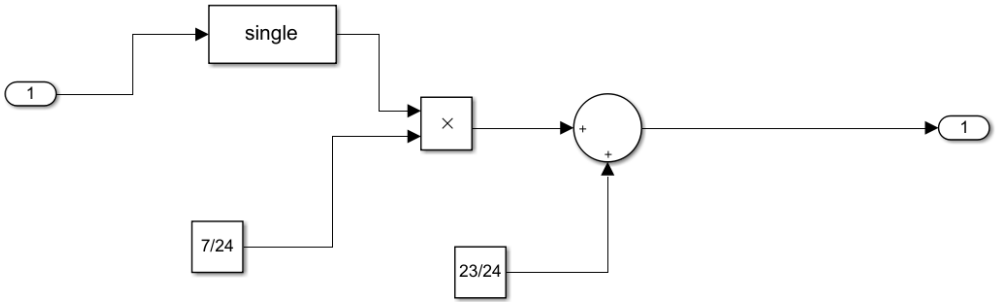
<table>
<tr><th colspan="4">Component: AV Delay Determination</th></tr>
<tr><td colspan="4"> 
</td></tr>
<tr><th>Purpose</th><th>Key Functions (Public/Internal)</th><th>Global/State Variables</th><th>Interactions with other Components</th></tr>
<tr><td>Uses current programmable parameters pertaining to AV delay to set the AV delay used for dual mode pacing logic.</td><td>Public:
<u>Inputs:</u>
fixed_AV, min_AV, AV_offset, current_HR, dyn_AV_delay

<u>Outputs:</u>
AV_delay

Internal:
Stateflow chart with states (default, fixed, dynamic)</td><td>State Variables:
Active state within the Stateflow chart.

Local Variables:</td><td>AV delay is used by dual mode pacing DDDR (1.1.3.2.9) in order to dictate atrium/ventricular pacing timing.</td></tr>
</table>

3.3.2.8. (1.1.2.8) Activity Threshold Set

Component: Activity Threshold Set			
			
Purpose	**Key Functions (Public/Internal)**	**Global/State Variables**	**Interactions with other Components**
Sets the activity threshold to be used to determine when activity has exceeded programmed limit.	**Public:** Inputs: ActivityThreshold Outputs: ActivityThresh **Internal:**	None. Stateless subsystem (not a Stateflow), and contains no global variables, only outputs a signal **State Variables:** **Local Variables:**	Adjusts the programmed activity threshold, and feeds it into the Rate Adaptive Pacing Calculation (1.1.2.5).

3.3.3. (1.1.3) Subroutines

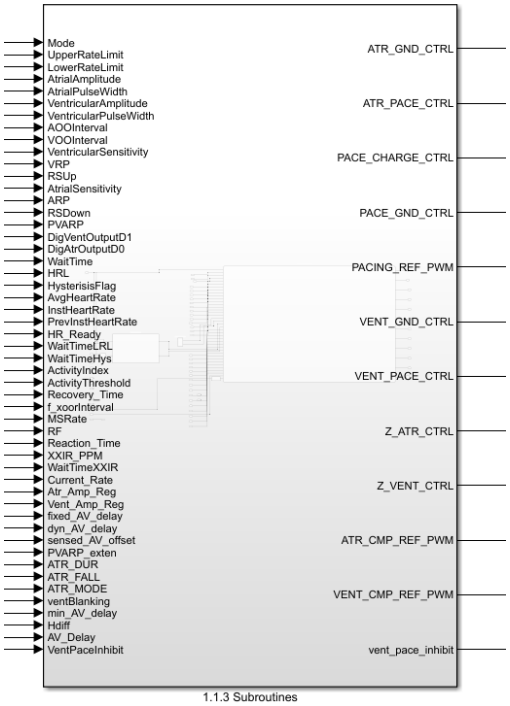
<table>
<tr><th colspan="4">Component: Subroutines</th></tr>
<tr><td colspan="4"> 

Figure 2013: Subroutines Subsystem</td></tr>
<tr><th>Purpose</th><th>Key Functions (Public/Internal)</th><th>Global/State Variables</th><th>Interactions with other Components</th></tr>
<tr><td>Contains all the pacing modes in their Stateflow representations, and contains blocks to compute PWM output to be used in control logic. Main purpose of this module is to encapsulate pacing control.</td><td>Public:
<u>Inputs:</u>
Mode, UpperRateLimit, LowerRateLimit, AtrialAmplitude, AtrialPulseWidth, VentricularAmplitude, VentricularPulseWidth, AOOInterval, VOOInterval, VentricularSensitivity, VRP, RSUp, AtrialSensitivity, ARP, RSDown, PVARP, DigVentOutputD1, DigAtrOutputD0, WaitTime, HRL, HysterisisFlag, AvgHeartRate, InstHeartRate, PrevInstHeartRate, HR_Ready, WaitTimeHRL, WaitTimeHys,</td><td>Not a Stateflow, just a subsystem containing other Stateflows and control systems. No global variables accessible by the rest of the model, or internal state variables</td><td>Sends outputs from Pacemaker Control (1.1.3.2) to Hardware Output Mapping (1.1.4). Informs voltage cycling and board pin output control.</td></tr>
</table>

	ActivityIndex, ActivityThreshold, Recovery_Time, xoorInterval, MSRate, RF, Reaction_Time, XXIR_PPM, WaitTimeXXIR, Current_Rate, Atr_Amp_Reg, Vent_Amp_Reg, fixed_AV_delay, dyn_AV_delay, sensad_AV_offset, PVARP_exten, ATR_DUR, ATR_FALL, ATR_MODE, ventBlanking, min_AV_delay, Hdiff, AV_Delay, VentPaceInhibit <u>Outputs</u> ATR_GND_CTRL, ATR_PACE_CTRL, PACE_CHARGE_CTRL, PACE_GND_CTRL, PACING_REF_PWM, VENT_GND_CTRL, VENT_PACE_CTRL, Z_ATR_CTRL, Z_VENT_CTRL, ATR_CMP_REF_PWM, VENT_CMP_REF_PWM, vent_pace_inhibit **Internal:** Subsystems to compute PWM output (4.3.3.1), and Pacing Control logic (1.1.3.2)	(other than those from contained subsystems).	

3.3.3.1. (1.1.3.1) Pacing Duty Cycle Calculation

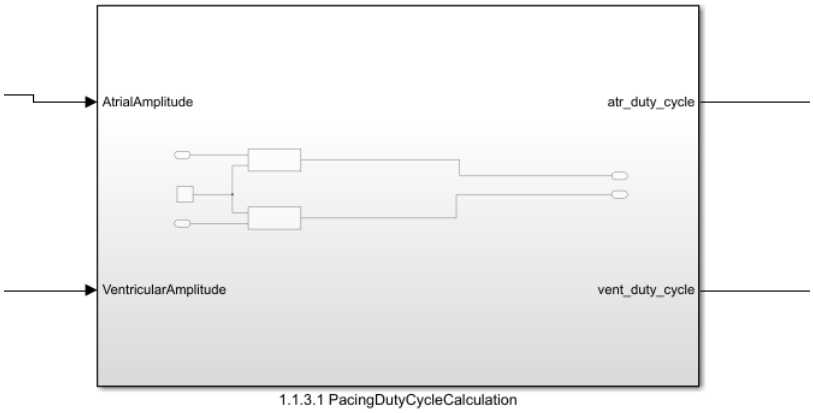
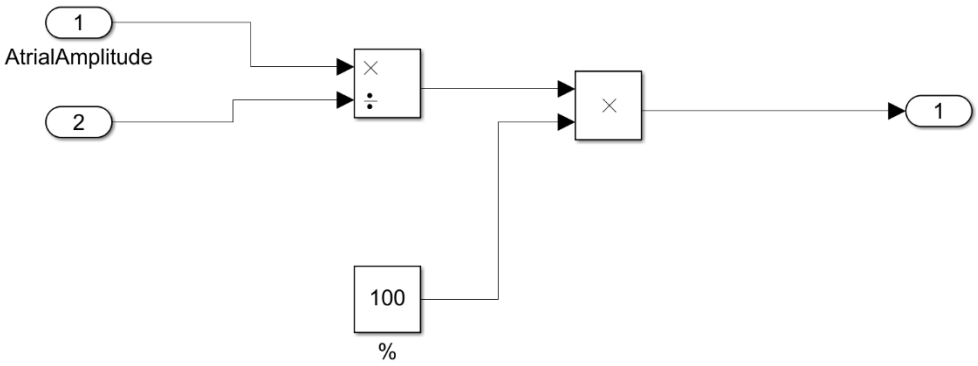
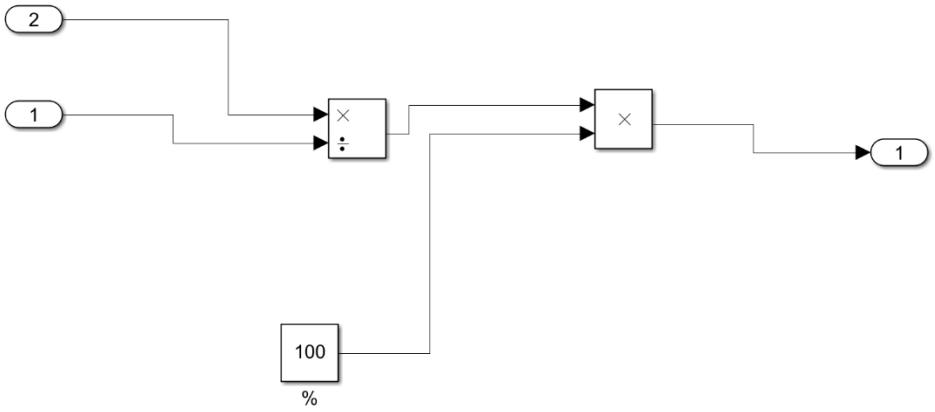
Component: Pacing Duty Cycle Calculation			
			
Purpose	**Key Functions (Public/Internal)**	**Global/State Variables**	**Interactions with other Components**
Encapsulates subsystems 1.1.3.1.1 and 1.1.3.1.2, which compute and output the PWM duty cycle for pacing capacitors.	**Public:** Inputs: AtrialAmplitude, Ventricular Amplitude **Internal:**	None. Stateless subsystem (not a Stateflow), and contains no global variables, only outputs a signal.	Outputs used in Pacemaker Control (1.1.3.2) to define discharge amplitude, and implicitly used in Hardware Output Mapping (1.1.4)

Figure 21: Pacing Duty Cycle Subsystem

4.3.3.1.1. (1.1.3.1.1) ATR Duty Cycle Calculation

Component: Atrial Duty Cycle Calculator			
 *Figure 2214: Atrial Duty Cycle Calculator Component*			
Purpose	**Key Functions (Public/Internal)**	**Global/State Variables**	**Interactions with other Components**
Calculates the PWM duty cycle percentage for PACING_REF_PWM based on the desired Atrial Amplitude. Encapsulates the conversion formula.	**Public:** Inputs: AtrialAmplitude, MaxVoltage Outputs: ATR_Duty_Cycle **Internal:** Uses Divide and Product blocks to perform the calculation.	None. Stateless subsystem (not a Stateflow), and contains no global variables, only outputs a signal.	Is used by Pacemaker Control (1.1.3.2) in AOO (1.1.3.2.1), AAI (1.1.3.2.3), AOOR (1.1.3.2.5), AAIR (1.1.3.2.7), and DDDR (1.1.3.2.9) to charge a capacitor to a desired voltage using PWM.

4.3.3.1.2. (1.1.3.1.2) VENT Duty Cycle Calculation

Component: Ventricle Duty Cycle Calculator			
 *Figure 2315: Ventricle Duty Cycle Calculator Component*			
Purpose	**Key Functions (Public/Internal)**	**Global/State Variables**	**Interactions with other Components**
Calculates the PWM duty cycle percentage for PACING_REF_PWM based on the desired Ventricular Amplitude. Encapsulates the conversion formula.	**Public:** Inputs: VentricleAmplitude, MaxVoltage Outputs: VENT_Duty_Cycle **Internal:** Uses Divide and Product blocks to perform the calculation.	None. Stateless subsystem (not a Stateflow), and contains no global variables, only outputs a signal.	Is used by Pacemaker Control (1.1.3.2) in VOO (1.1.3.2.2), VVI (1.1.3.2.4), VOOR (1.1.3.2.6), VVIR (1.1.3.2.8), and DDDR (1.1.3.2.9) to charge a capacitor to a desired voltage using PWM.

3.3.3.2. (1.1.3.2) Pacemaker Control

State machine design is detailed in Section 3.3.4.

Component: Pacemaker Control

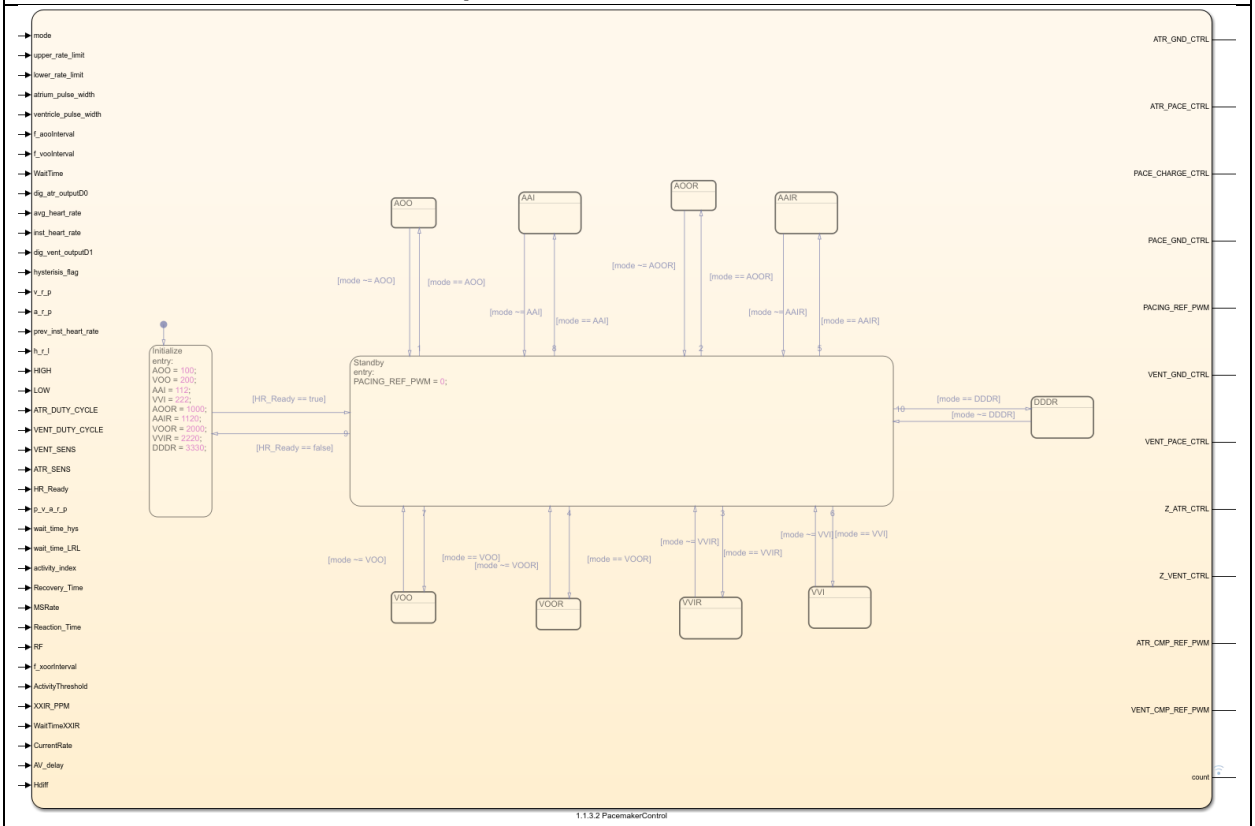
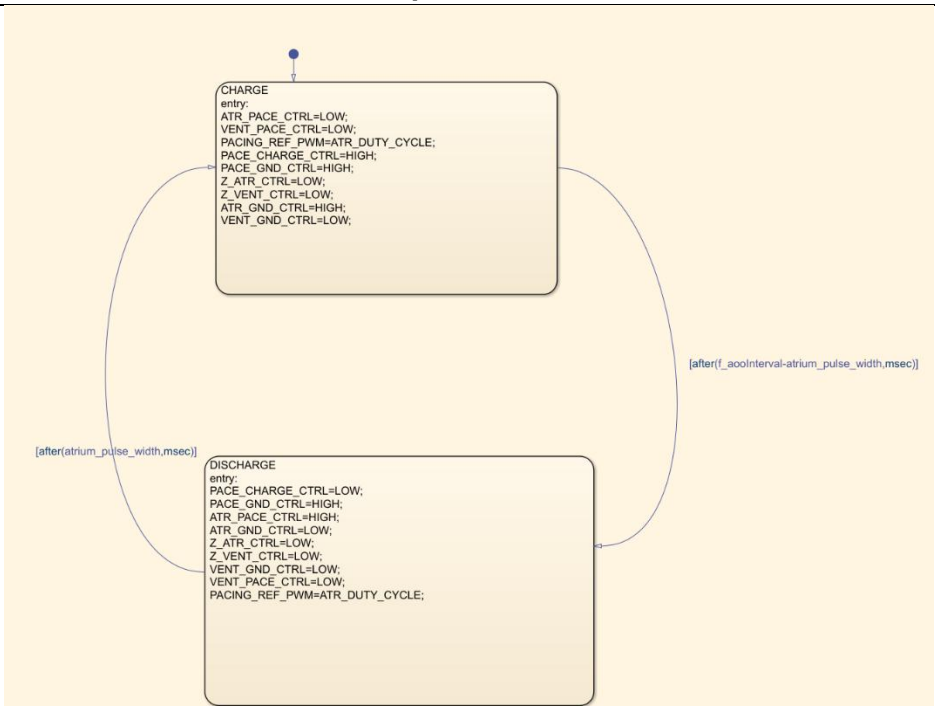


Figure 24: Pacemaker Control Stateflow Top Layer

Purpose	**Key Functions (Public/Internal)**	**Global/State Variables**	**Interactions with other Components**
Contains all the logic for pacing control, and the different control systems.	**Public:** Inputs: (Available to all control systems) mode, upper_rate_limit, lower_rate_limit, atrium_pulse_width, ventricle_pulse_width, f_aooInterval, f_vooInterval, WaitTime, dig_atr_outputD0, avg_heart_rate, inst_heart_rate, dig_vent_outputD1, hysterisis_flag, v_r_p, a_r_p, prev_inst_heart_rate, h_r_l, HIGH, LOW, ATR_DUTY_CYCLE,	Active state Initialize or Standby, or: AOO (1.1.3.2.1) VOO (1.1.3.2.2) AAI (1.1.3.2.3) VVI (1.1.3.2.4)	Only activates a control system corresponding to the mode, if the correct programmable parameters have been set by Received Serial Message Processing (1.1.1.3). Sends outputs from pacemaker control systems to

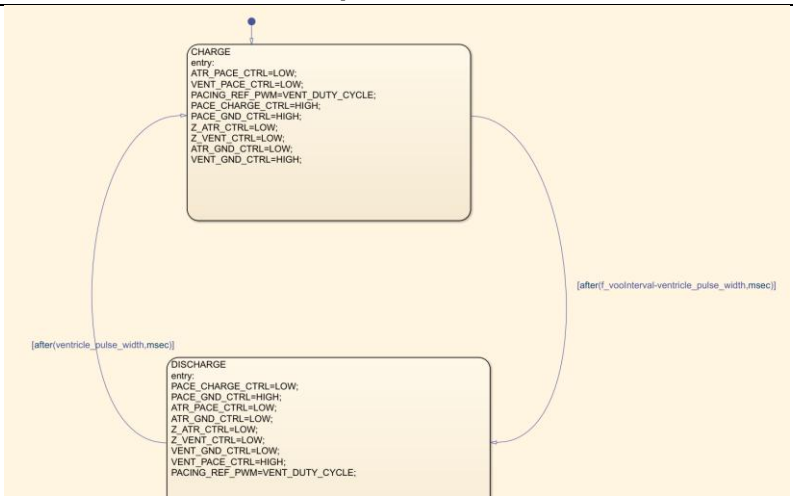
	VENT_DUTY_CYCLE, VENT_SENS, ATR_SENS, HR_Ready, P_v_a_r_p, wait_time_hys, wait_time_LRL, activity_index, Recovery_Time, MSRate, Reaction_Time, RF, f_xoorInterval, ActivityThreshold, XXIR_PPM, WaitTimeXXIR, CurrentRate, AV_delay, Hdiff **Internal:** Contains initialize and standby states, as well as all pacing control systems.	AOOR (1.1.3.2.5) VOOR (1.1.3.2.6) AAIR (1.1.3.2.7) VVIR (1.1.3.2.8) DDDR (1.1.3.2.9)	downstream subsystem Hardware Output Mapping (1.1.4) to interface with board pins.

4.3.3.2.1. (1.1.3.2.1) AOO

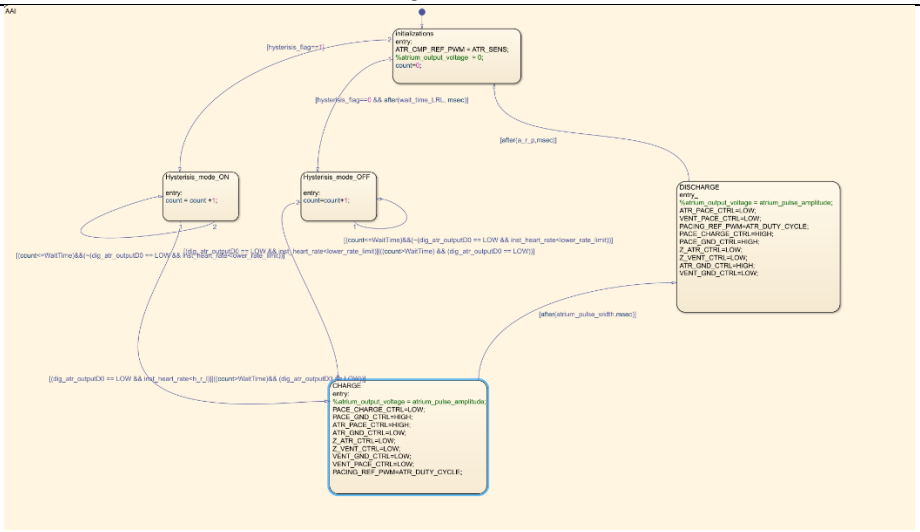
Component: AOO
 *Figure 2516: AOO Component*

Purpose	Key Functions (Public/Internal)	Global/State Variables	Interactions with other Components
Implements the logic for AOO mode by providing a way to do asynchronous atrial pacing.	**Public:** Inherits inputs/outputs from parent Pacemaker Control chart relevant to AOO. **Internal:** Contains states CHARGE and DISCHARGE.	Explicit FSM design is tabulated in 2.4.1.1.	Subcomponent system of the parent Pacemaker Control (1.1.3.2) routine. Control system outputs are directly sent to Hardware Output Mapping (1.1.4) to interface with board bins.

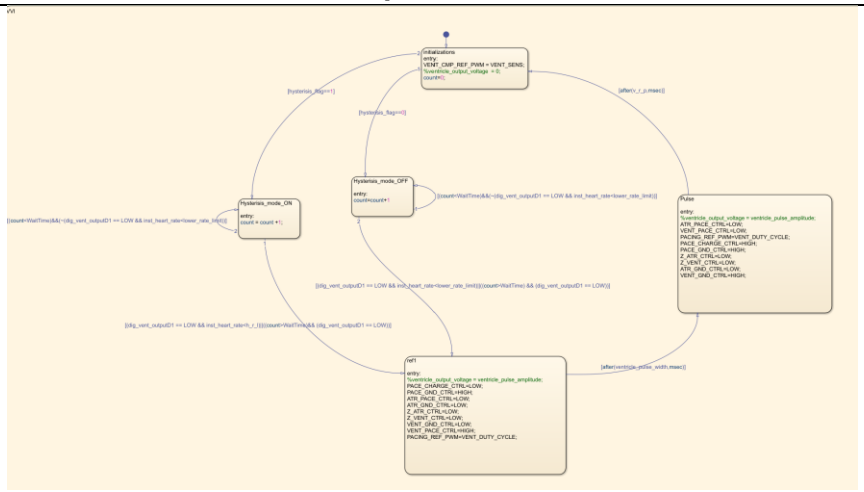
4.3.3.2.2. (1.1.3.2.2) VOO

Component: VOO			
 *Figure 2617: Ventricle Duty Cycle Calculator Component*			
Purpose	**Key Functions (Public/Internal)**	**Global/State Variables**	**Interactions with other Components**
Implements the logic for VOO mode by providing a way to do asynchronous atrial pacing.	**Public:** Inherits inputs/outputs from parent Pacemaker Control chart relevant to VOO. **Internal:** Contains states CHARGE and DISCHARGE.	Explicit FSM design is tabulated in 2.4.1.2.	Subcomponent system of the parent Pacemaker Control (1.1.3.2) routine. Control system outputs are directly sent to Hardware Output Mapping (1.1.4) to interface with board bins.

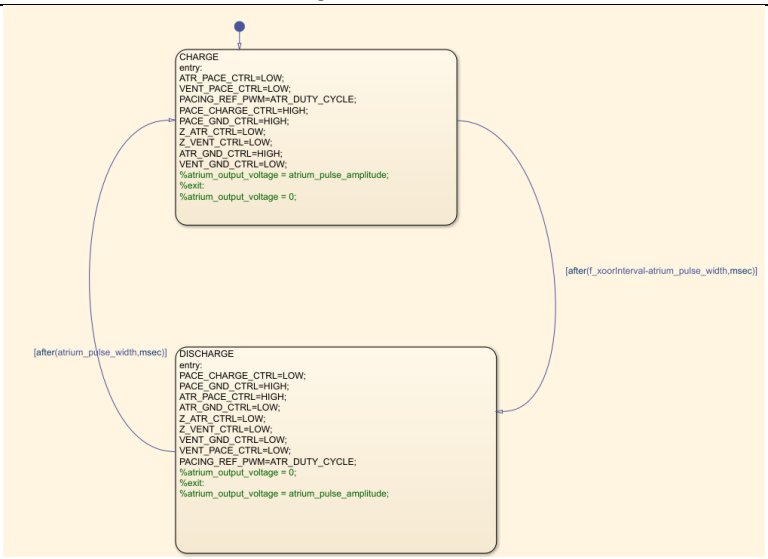
4.3.3.2.3. (1.1.3.2.3) AAI

Component: AAI			
 *Figure 2718: AAI Component*			
Purpose	**Key Functions (Public/Internal)**	**Global/State Variables**	**Interactions with other Components**
Implements the logic for AAI mode by providing a way to do atrial pacing, inhibited by atrial sensing. Also includes hysteresis logic.	**Public:** Inherits inputs/outputs from parent Pacemaker Control chart relevant to AAI. **Internal:** Contains states: initializations, Hysteresis_mode_ON/OFF, CHARGE and Pulse	Explicit FSM design is tabulated in 2.4.1.3.	Subcomponent system of the parent Pacemaker Control (1.1.3.2) routine. Control system outputs are directly sent to Hardware Output Mapping (1.1.4) to interface with board bins.

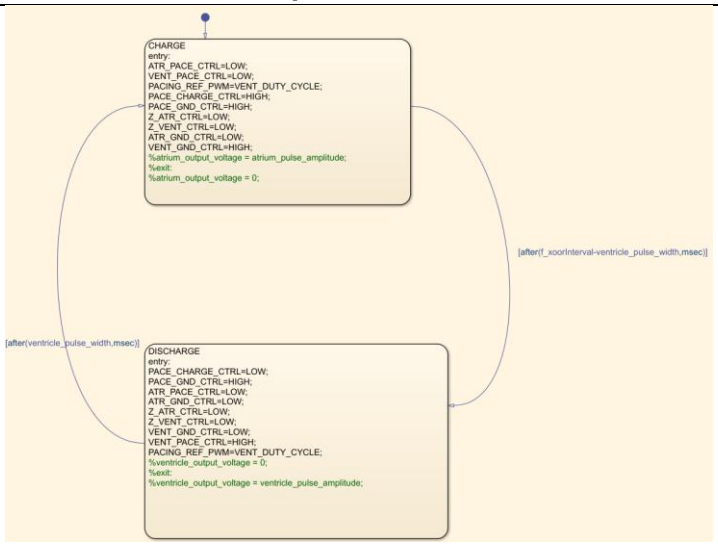
4.3.3.2.4. (1.1.3.2.4) VVI

Component: VVI			
 *Figure 2819: VVI Component*			
Purpose	**Key Functions (Public/Internal)**	**Global/State Variables**	**Interactions with other Components**
Implements the logic for VVI mode by providing a way to do ventricular pacing, inhibited by ventricular sensing. Also includes hysteresis logic.	**Public:** Inherits inputs/outputs from parent Pacemaker Control chart relevant to VVI. **Internal:** Contains states: initializations, Hysteresis_mode_ON/OFF, CHARGE and Pulse	Explicit FSM design is tabulated in 2.4.1.4.	Subcomponent system of the parent Pacemaker Control (1.1.3.2) routine. Control system outputs are directly sent to Hardware Output Mapping (1.1.4) to interface with board bins.

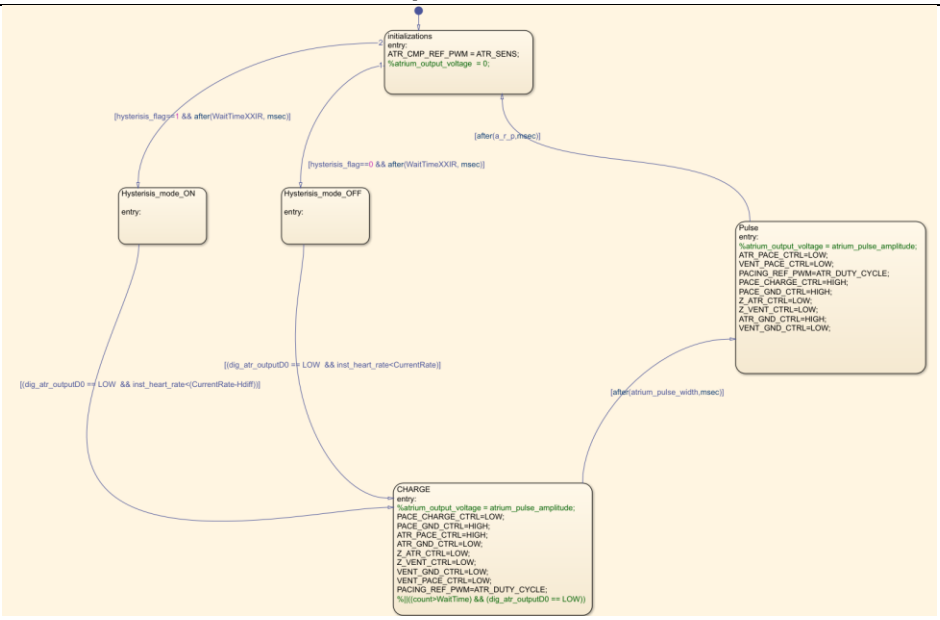
4.3.3.2.5. (1.1.3.2.5) AOOR

Component: AOOR			
			
Purpose	**Key Functions (Public/Internal)**	**Global/State Variables**	**Interactions with other Components**
Implements the logic for AOOR by performing standard chamber unsensed atrial pacing, and responds dynamically to detected physical activity by the wearer.	**Public:** Inherits inputs/outputs from parent Pacemaker Control chart relevant to AOOR. **Internal:** Contains states: CHARGE, DISCHARGE	Explicit FSM design is tabulated in 2.4.1.5.	Subcomponent system of the parent Pacemaker Control (1.1.3.2) routine. Control system outputs are directly sent to Hardware Output Mapping (1.1.4) to interface with board bins.

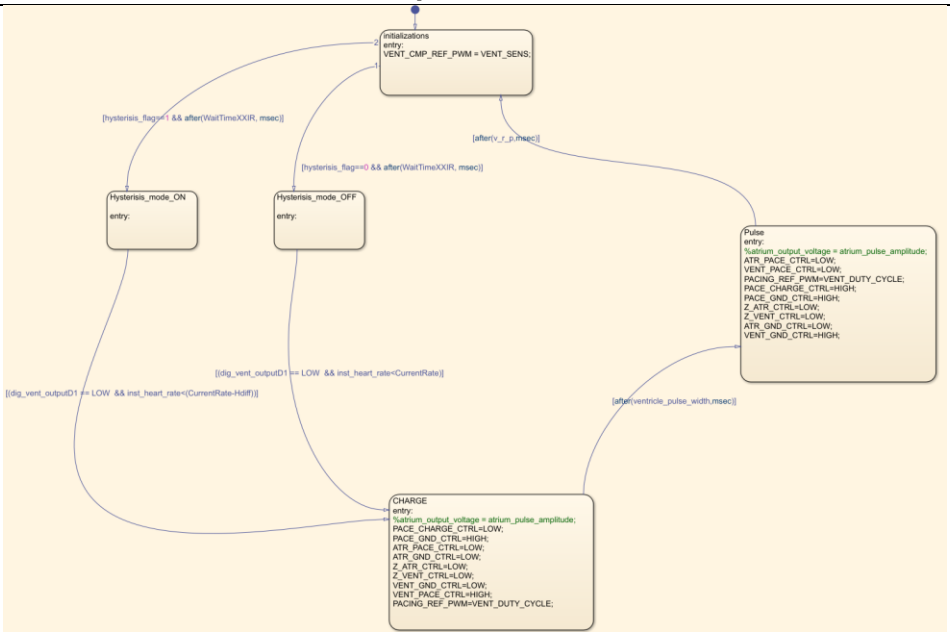
4.3.3.2.6. (1.1.3.2.6) VOOR

Component: VOOR			
			
Purpose	**Key Functions (Public/Internal)**	**Global/State Variables**	**Interactions with other Components**
Implements the logic for VOOR by performing standard chamber unsensed ventricular pacing, and responds dynamically to detected physical activity by the wearer.	**Public:** Inherits inputs/outputs from parent Pacemaker Control chart relevant to VOOR. **Internal:** Contains states: CHARGE, DISCHARGE	Explicit FSM design is tabulated in 2.4.1.6.	Subcomponent system of the parent Pacemaker Control (1.1.3.2) routine. Control system outputs are directly sent to Hardware Output Mapping (1.1.4) to interface with board bins.

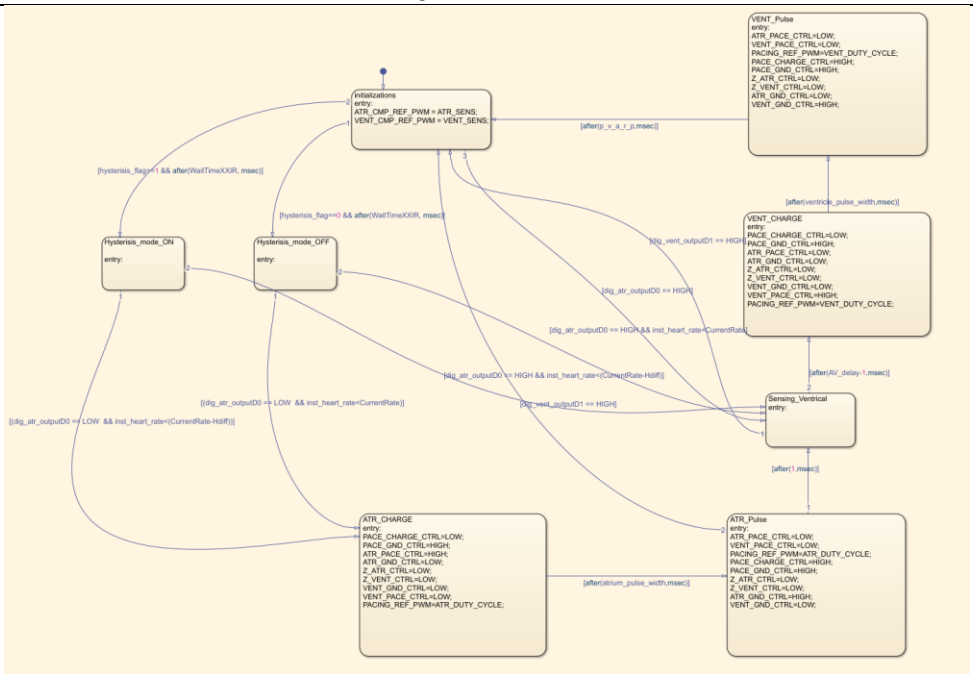
4.3.3.2.7. (1.1.3.2.7) AAIR

Component: AAIR			
			
Purpose	**Key Functions (Public/Internal)**	**Global/State Variables**	**Interactions with other Components**
Implements the logic for AAIR by performing active atrial chamber sensing, and activity sensing. Responds dynamically to detected physical activity by the wearer, and atrial bradycardia.	**Public:** Inherits inputs/outputs from parent Pacemaker Control chart relevant to AAIR. **Internal:** Contains states: initializations, Hysteresis_mode_ON/OFF, CHARGE and Pulse	Explicit FSM design is tabulated in 2.4.1.7.	Subcomponent system of the parent Pacemaker Control (1.1.3.2) routine. Control system outputs are directly sent to Hardware Output Mapping (1.1.4) to interface with board bins.

4.3.3.2.8. (1.1.3.2.8) VVIR

<table>
<tr><th colspan="4">Component: VVIR</th></tr>
<tr><td colspan="4"> 
</td></tr>
<tr><th>Purpose</th><th>Key Functions (Public/Internal)</th><th>Global/State Variables</th><th>Interactions with other Components</th></tr>
<tr><td>Implements the logic for VVIR by performing active ventricular chamber sensing, and activity sensing. Responds dynamically to detected physical activity by the wearer, and ventricular bradycardia.</td><td>Public:
Inherits inputs/outputs from parent Pacemaker Control chart relevant to VVIR.
Internal:
Contains states: initializations, Hysteresis_mode_ON/OFF, CHARGE and Pulse</td><td>Explicit FSM design is tabulated in 2.4.1.8.</td><td>Subcomponent system of the parent Pacemaker Control (1.1.3.2) routine. Control system outputs are directly sent to Hardware Output Mapping (1.1.4) to interface with board bins.</td></tr>
</table>

4.3.3.2.9. (1.1.3.2.9) DDDR

Component: DDDR			
			
Purpose	**Key Functions (Public/Internal)**	**Global/State Variables**	**Interactions with other Components**
Implements the logic for DDDR by performing active dual chamber, and activity sensing. Responds dynamically to detected physical activity by the wearer, cardiac arrhythmia..	**Public:** Inherits inputs/outputs from parent Pacemaker Control chart relevant to VVIR. **Internal:** Contains states: initializations, Hysteresis_mode_ON/OFF, CHARGE and Pulse.	Explicit FSM design is tabulated in 2.4.1.9.	Subcomponent system of the parent Pacemaker Control (1.1.3.2) routine. Control system outputs are directly sent to Hardware Output Mapping (1.1.4) to interface with board bins.

3.3.4. (1.1.4) Hardware Output Mapping

Component: Hardware Output Pin Mapping

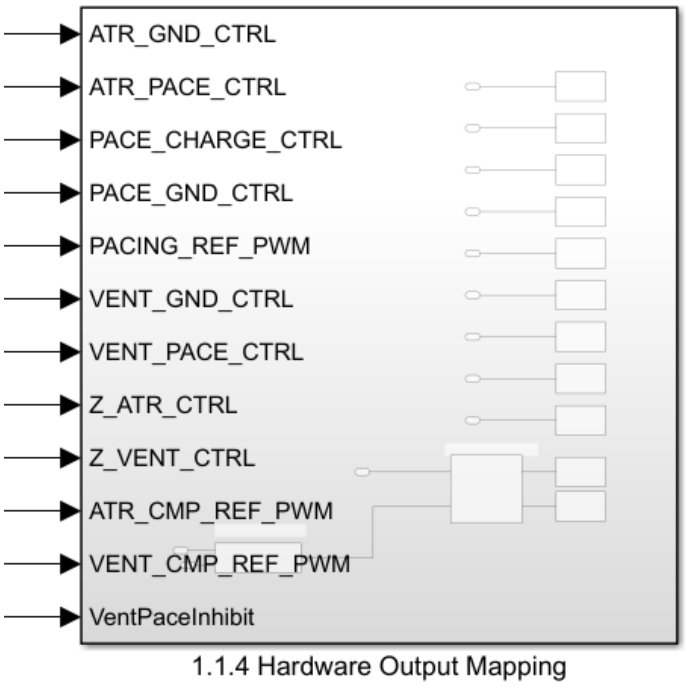


Figure 2920: Hardware Output Pin Mapping Component

Purpose	**Key Functions (Public/Internal)**	**Global/State Variables**	**Interactions with other Components**
Translates hardware control commands received from subroutines (1.3) into physical pin actions through setting digital HIGH/LOW or PWM duty cycles on the FRDM-K64F output pins. Hides the output hardware interface.	**Public:** Inputs: ATR_GND_CTRL, ATR_PACE_CTRL, PACE_CHARGE_CTRL, PACE_GND_CTRL, PACING_REF_PWM, VENT_GND_CTRL, VENT_PACE_CTRL, Z_ATR_CTRL, Z_VENT_CTRL, ATR_CMP_REF_PWM, VENT_CMP_REF_PWM, VentPaceInhibit **Internal:** Uses FRDM-K64F blocks: Digital Write and PWM Output	None. Stateless subsystem (not a Stateflow), and contains no global variables, only outputs a signal.	Is used by none as it directly interacts with the hardware. Uses Subroutines (1.1.3) to receive all control command inputs.

3.3.4.1. (1.1.4.1) Reference Duty Cycle Calculation

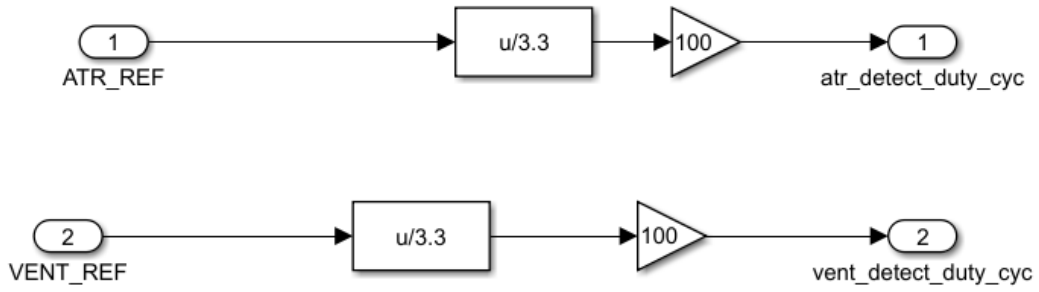
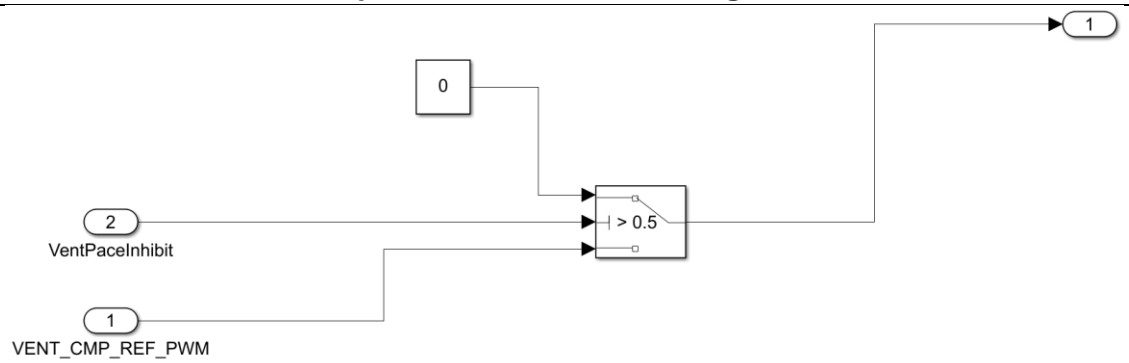
Component: Reference Duty Cycle Calculation


Figure 30: Reference Duty Cycle Calculation Subsystem

Purpose	**Key Functions (Public/Internal)**	**Global/State Variables**	**Interactions with other Components**
Calculates the PWM duty cycle percentage for the ATR_CMP_REF_PWM and VENT_CMP_REF_PWM by using threshold voltages.	**Public:** Inputs: ATR_REF, VENT_REF Outputs: atr_detect_duty_cyc vent_detect_duty_cyc **Internal:** Uses Divide and Product blocks to perform the calculation.	None. Stateless subsystem (not a Stateflow), and contains no global variables, only outputs a signal.	Is used by Hardware Output Pin Mapping (1.1.4) to provide PWM inputs to the FDRM-K64F pins associated with threshold detection of pulses.

3.3.4.2. (1.1.4.2) Ventricular Pacing Inhibit

Component: Ventricular Pacing Inhibit			
			
Purpose	**Key Functions (Public/Internal)**	**Global/State Variables**	**Interactions with other Components**
Detects push button state though hardware encapsulated variable, and inhibits ventricular pacing while the button is pressed.	**Public:** Inputs: VENT_CMP_REF_PWM VentPaceInhibit Outputs: VENT_REF	None. Stateless subsystem (not a Stateflow), and contains no global variables, only outputs a signal.	VENT_REF directly sets the duty cycle for pacing stimulus to be applied to the ventricular chamber on the board in Hardware Output Mapping (1.1.4).

3.4. Testing

3.4.1. AOO Test

Test Objective:
Verify that in AOO mode the device delivers atrial pacing at the programmed Lower Rate Limit with the programmed pulse conditions, independent of sensed activity.

Test Environment Setup:

- HeartView
    - Natural Atrium ON; Natural Ventricle OFF.
    - Natural Heart Rate: 50 BPM (LRL)
- Relevant Input Variables
    - Device mode: AOO
    - Lower_Rate_Limit: 50 ppm
    - Amplitude: nominal
    - Atrial Pulse Width: nominal

Pass/Fail Logic:

- PASS if: (i) blue A-A interval = 1200 ± 8ms, (ii) pulse width matches programmed value within measurement resolution, and (iii) no ventricular activity is produced.
- FAIL otherwise.

Figure AOO-1: With Natural HR = 50 BPM, blue atrial paces occur at the same periodicity as the red natural atrial pulses (top plot); ventricular plot remains = 0 as expected for AOO.

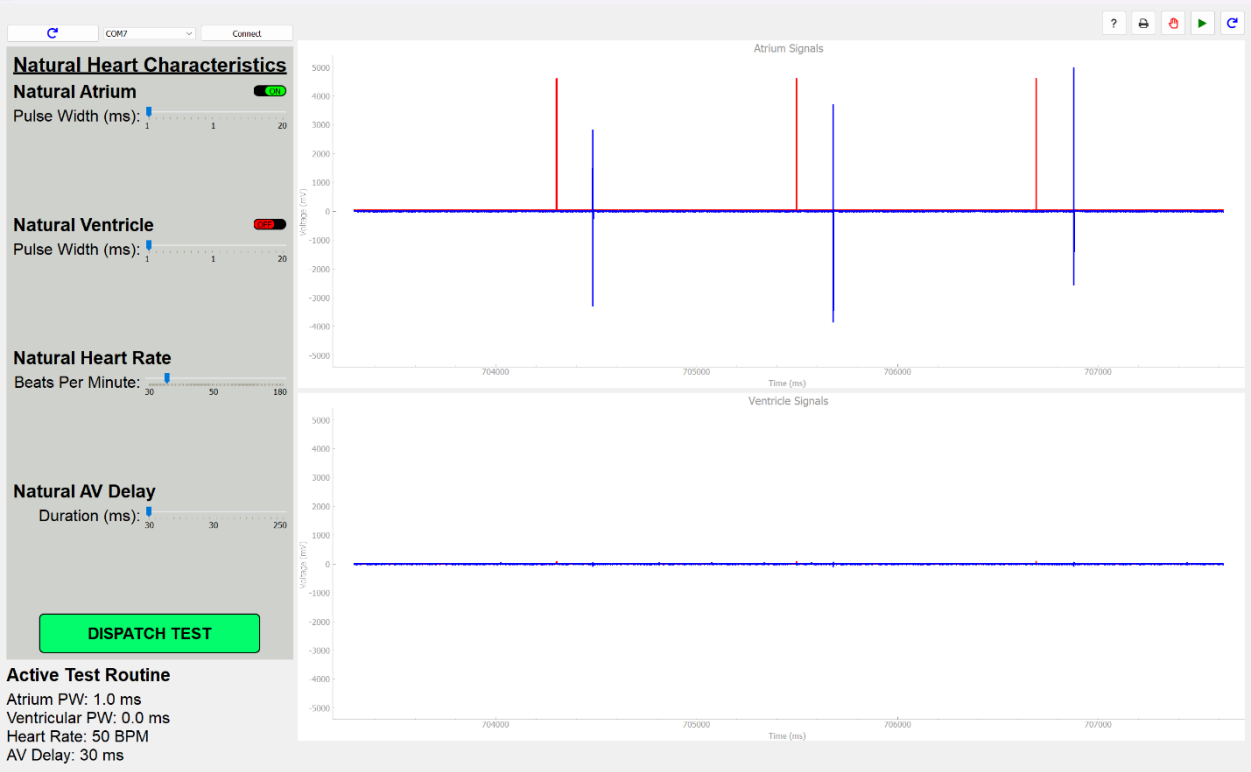


Figure AOO-2 : Consecutive blue paces are separated by a constant interval matching consecutive red pulses showing rate equivalence when the natural heart is at the LRL.

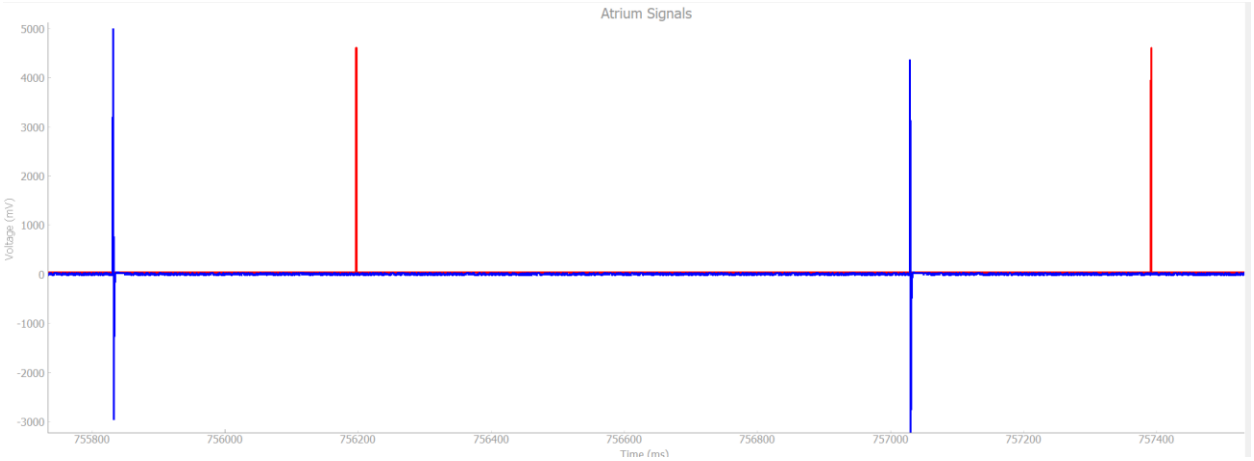
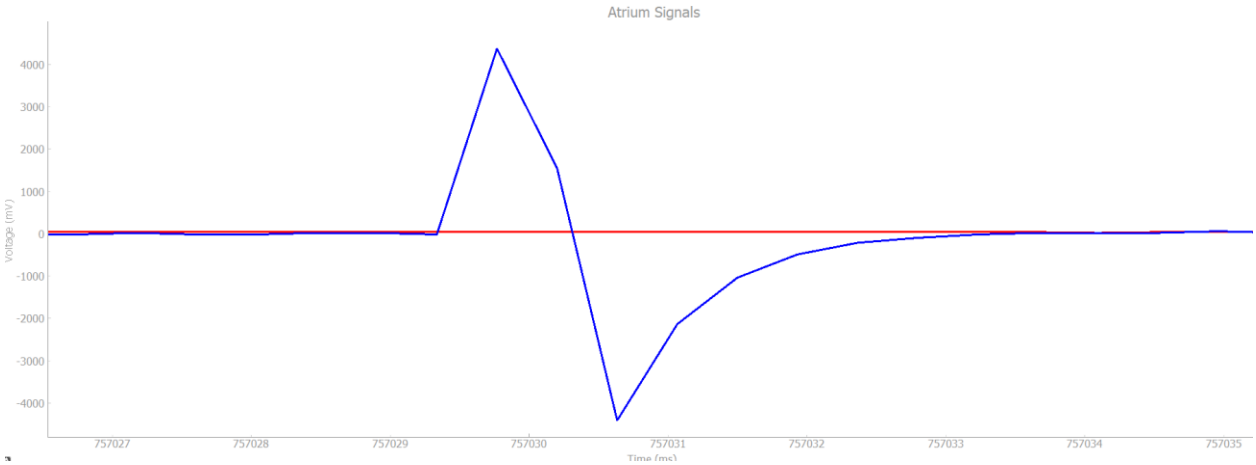


Figure AOO-3: A single blue pace output shows the expected rise, then fall and return to baseline during the refractory period which is consistent with the shield's charge/discharge pacing sequence and capacitor returning to net-zero.



Results.

Based on Figures AOO-1 through AOO-3, the device **passes** the AOO functional test. The blue atrial pace events occur at a constant interval of approximately 1200 ms (50 ppm), satisfying the criterion (A–A = 1200 ± 8 ms). Close-zoom shows a single-pulse waveform with rise/fall and return to baseline consistent with the shield's charge–discharge sequence and blocking-capacitor return to net-zero; no ventricular output is observed. Therefore criteria (i)–(iii) are met, and AOO fixed-rate pacing at the programmed LRL is verified. This demonstrates conformance to the LRL timing requirement and intended pulse delivery behavior for AOO.

3.4.2. VOO Test

Test Objective

Verify that in VOO mode the device delivers ventricular pacing at the programmed Lower Rate Limit with the programmed pulse conditions, independent of sensed activity.

Test Environment Setup

- HeartView
    - Natural Atrium OFF; Natural Ventricle ON.
    - Natural Heart Rate: 50 BPM (LRL)
- **Relevant Input Variables**
    - Device mode: VOO
    - Lower_Rate_Limit: 50 bpm
    - Amplitude: nominal
    - Ventricular Pulse Width: nominal

Pass/Fail Logic

- PASS if: (i) blue V–V interval = 1200 ± 10 ms, (ii) pulse width matches programmed value within measurement resolution, and (iii) no atrial activity is produced.
- FAIL otherwise.

Figures

Figure VOO-1: With Natural HR = 50 BPM, blue ventricular paces occur at the same periodicity as the red natural ventricular pulses (bottom plot); atrium plot remains ≈ 0 as expected for VOO.

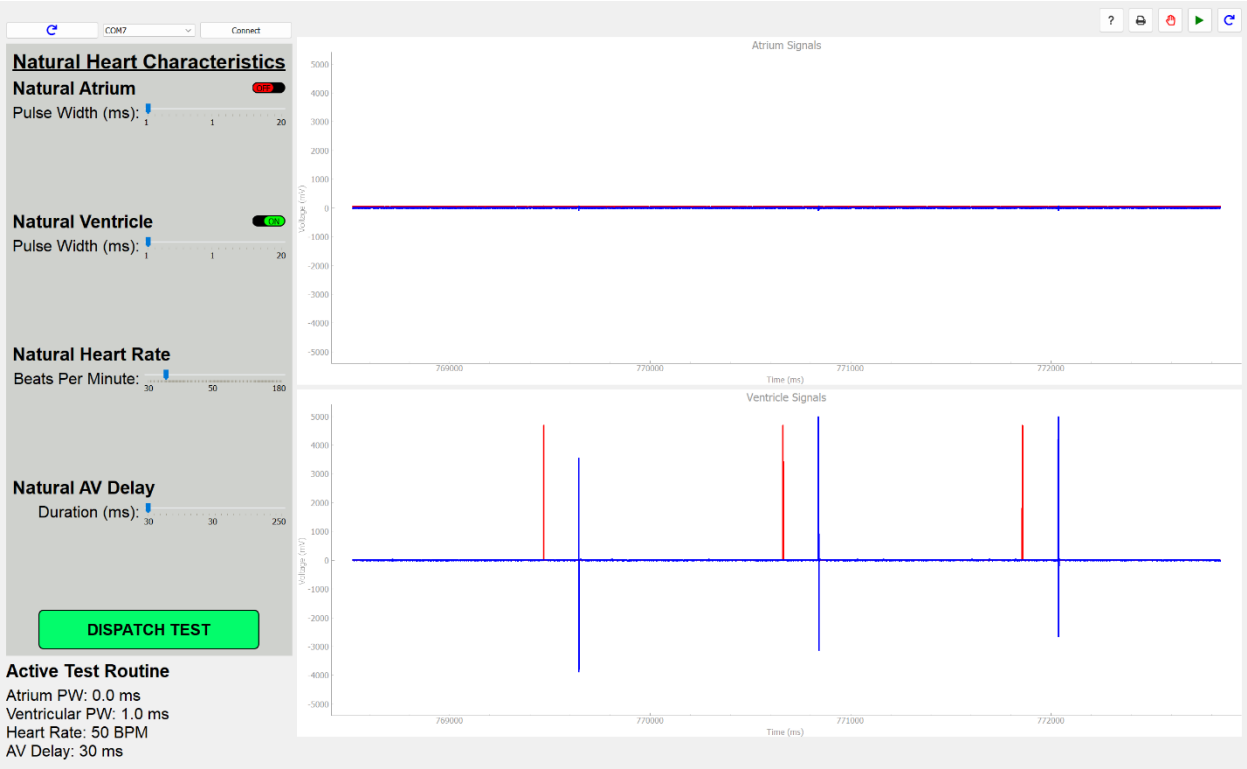


Figure VOO-2: Consecutive blue ventricular paces are separated by a constant interval matching consecutive red pulses, showing rate equivalence when the natural heart is at the LRL.

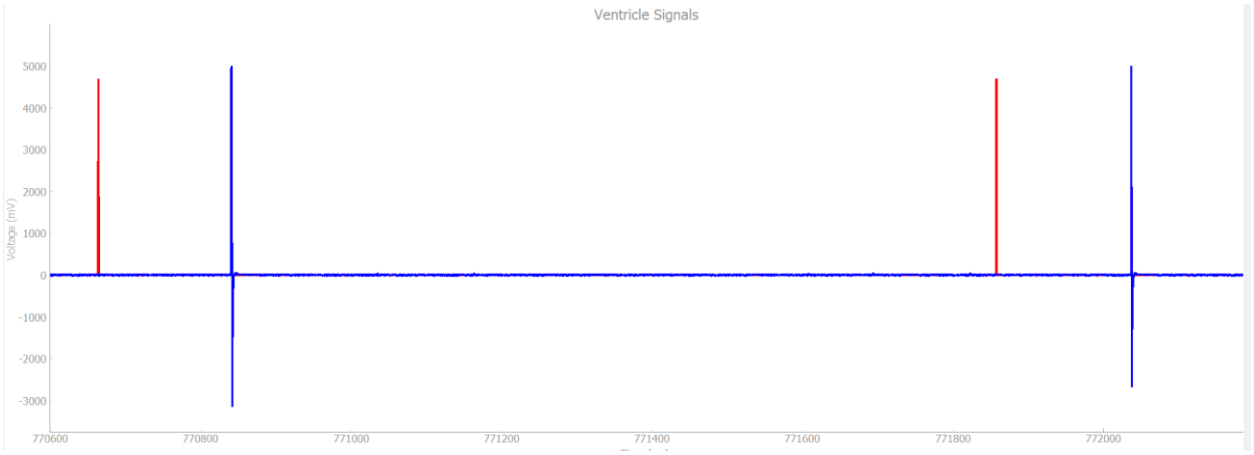
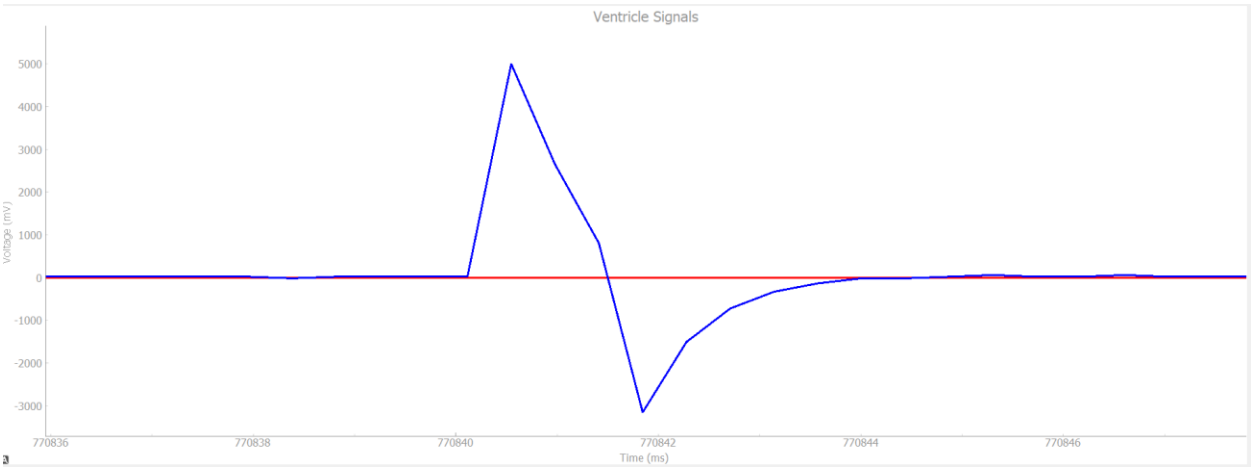


Figure VOO-3: A single blue ventricular pace output shows the expected rise, fall, and return to baseline—consistent with the shield's charge/discharge pacing sequence and capacitor returning to net-zero.



Results

Pass. From Figures VOO-1 to VOO-3, the blue ventricular pace events occur at a constant interval of ~1200 ms (50 ppm), meeting criterion (i). The close-zoom waveform shows a single-pulse morphology with rise/fall and baseline recovery (criterion ii), and no atrial output is observed (criterion iii). Therefore, the device passes the VOO functional test, confirming fixed-rate ventricular pacing at the programmed LRL and correct pulse delivery behavior for VOO.

3.4.3. AAI Tests

3.4.3.1. AAI-1 (Hysteresis OFF)

Test Objective
Verify that in AAI mode the device paces only when the heart's rate drops below the LRL and confirm the pulse morphology.

Test Environment Setup

- **HeartView**
  - Natural Atrium ON; Natural Ventricle OFF.
  - Test points (BPM): 100, 50, 49, 30.
- **Relevant Input Variables**
  - Device mode: AAI
  - Lower_Rate_Limit: 50 ppm
  - Hysteresis: Disabled
  - Amplitude: nominal
  - Atrial Pulse Width: nominal

Pass/Fail Logic

- PASS if:
  (i) No blue atrial paces at 100 BPM and 50 BPM (above LRL),
  (ii) Blue paces appear once HR < 50 BPM and the A–A interval = 1200 ± 8 ms,
  (iii) Zoomed pulse width matches programmed value within measurement resolution,
  (iv) No ventricular output is produced.
- FAIL otherwise.

Figures (Hysteresis OFF)

Figure AAI-1.1 (100 BPM): Red atrial beats at 100 BPM; no blue pacing (AAI pulsing is inhibited).

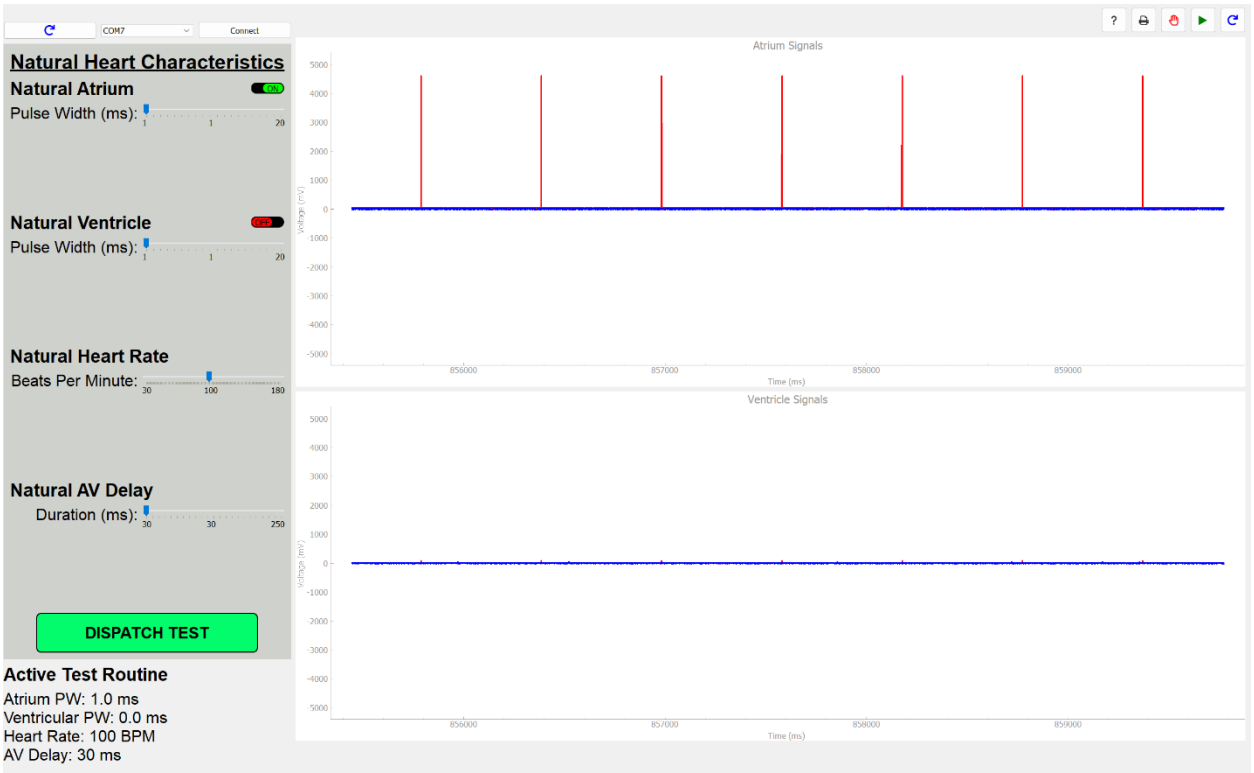


Figure AAI-1.2 (50 BPM = LRL): Red beats at 50 BPM; no blue pacing (AAI pulsing is still inhibited at the threshold).

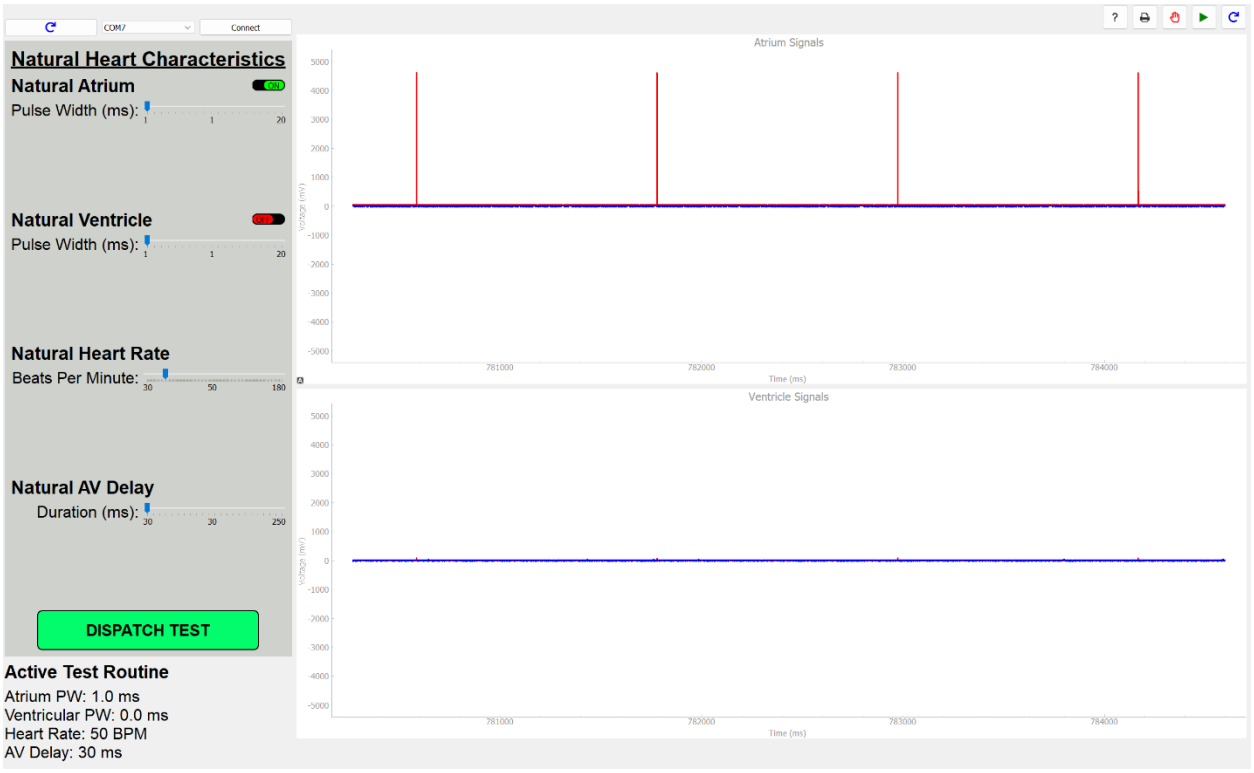


Figure AAI-1.3 (49 BPM < LRL): Pacing starts; blue paces present.

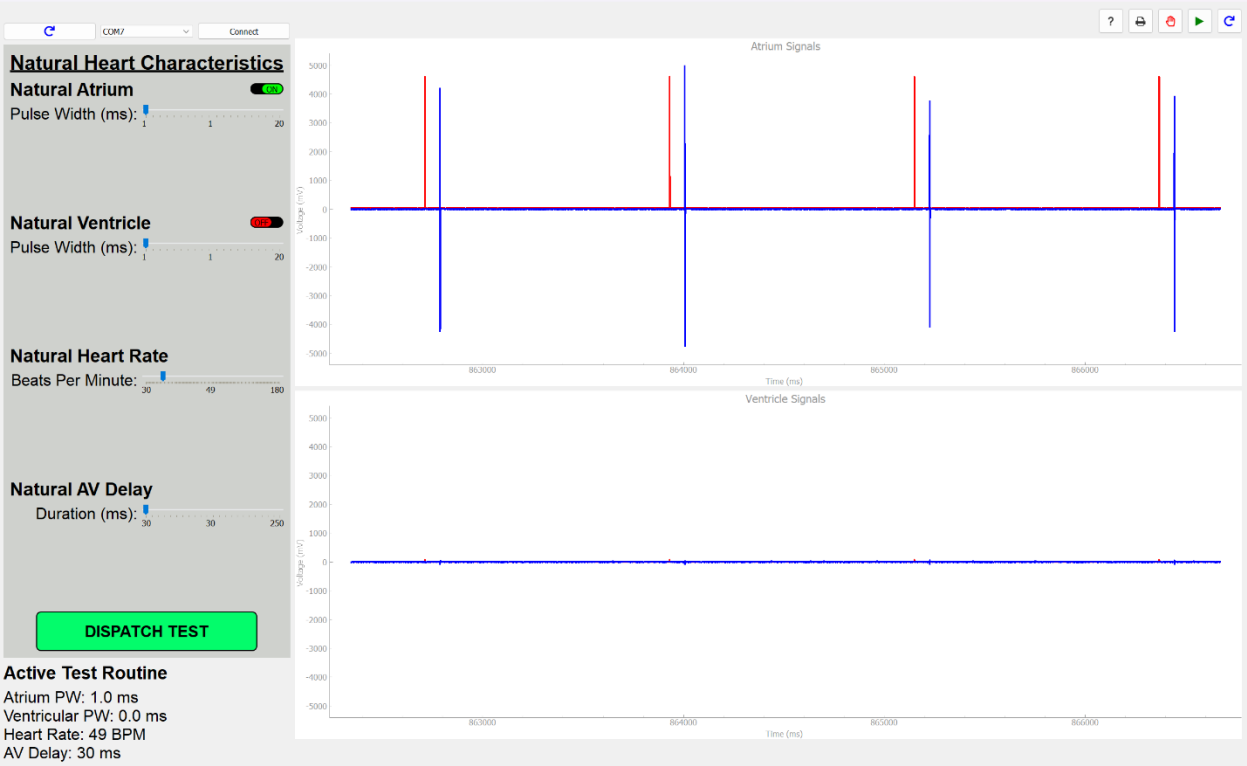


Figure AAI-1.4 (30 BPM): Continued pacing at LRL (blue paces periodic).

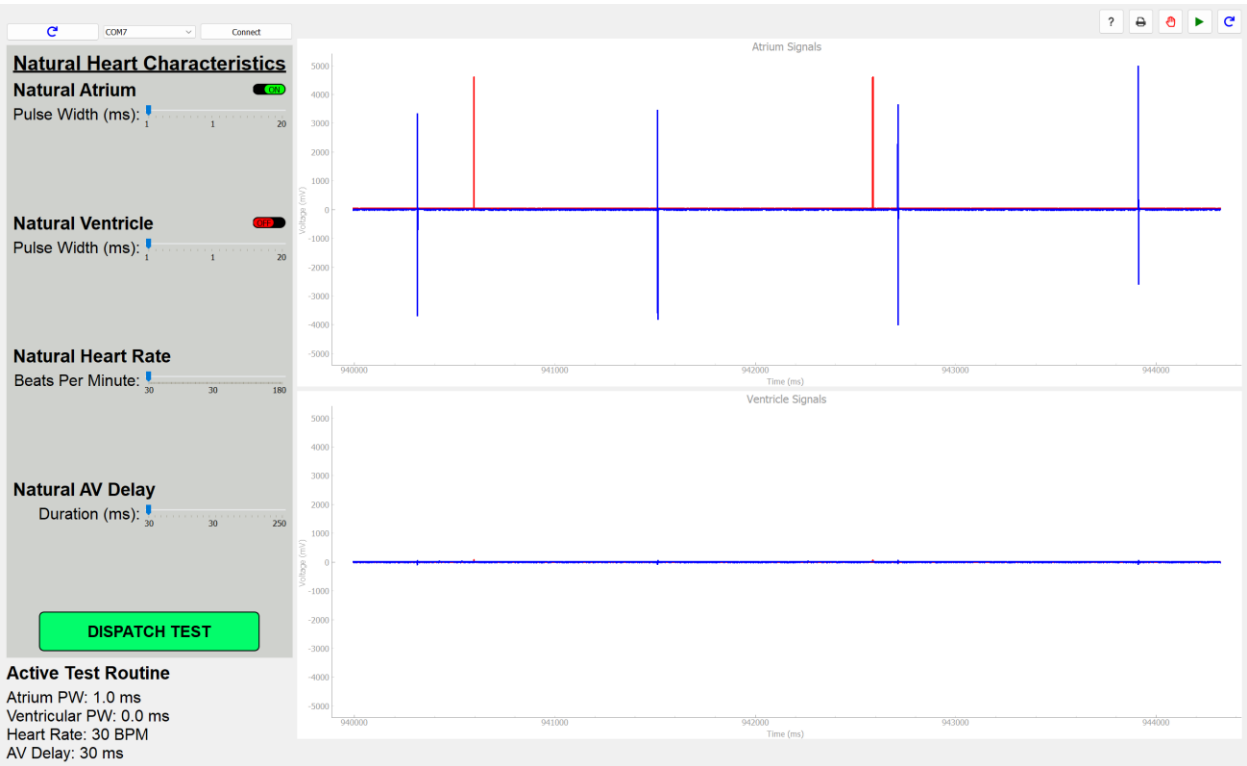
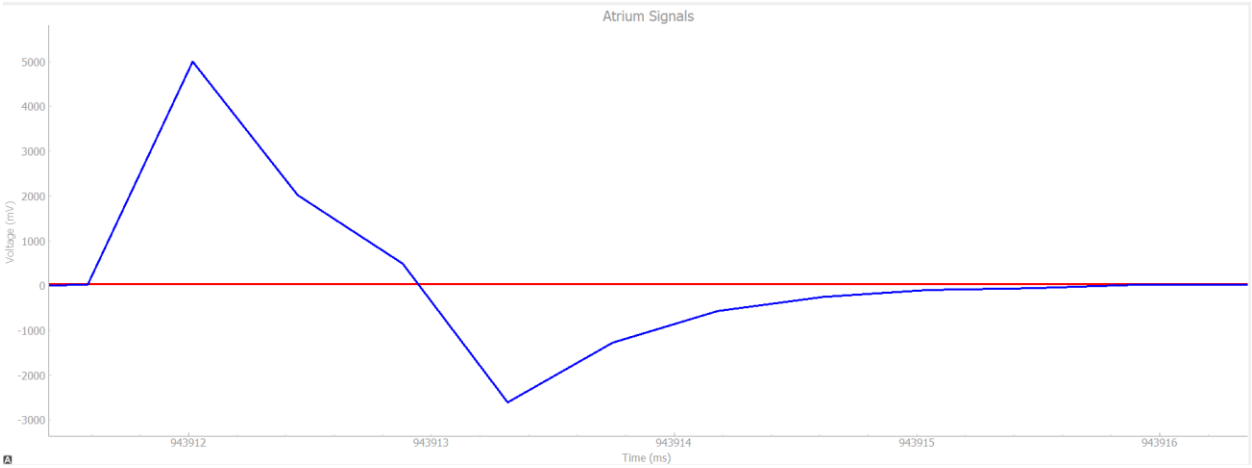


Figure AAI-1.5 (zoom): Single atrial pace shows rise–fall–return to baseline.



Results (Hysteresis OFF)

Pass. At 100 BPM and 50 BPM no blue atrial paces are observed (criteria i). Dropping to 49 BPM triggers pacing and the measured A–A is approximately 1200 ms (criteria ii). Close-zoom confirms a single-pulse morphology with the expected return to baseline and no ventricular output (criteria iii–iv). Therefore, AAI inhibited behavior and LRL pacing are verified with hysteresis disabled.

3.4.3.2. AAI-2 (Hysteresis ON)

Test Objective

Verify that when Hysteresis is enabled, pacing does not begin until the sensed heart's rate falls below the Hysteresis Rate Limit (HRL), and when it does, pacing resumes at the LRL.

Test Environment Setup

- HeartView
  - Natural Atrium ON; Natural Ventricle OFF.
  - Test points (BPM): 45 BPM (=HRL), then 44 BPM (< HRL).
- Relevant Input Variables
  - Device mode: AAI
  - Lower_Rate_Limit: 50 bpm
  - Hysteresis: Enabled
  - Hysteresis_Rate_Limit (HRL): = 45 bpm
  - Amplitude & Pulse Width: nominal

Pass/Fail Logic

- PASS if:
  (i) At HRL (= 45 BPM) there are no blue atrial paces (still inhibited),
  (ii) When intrinsic HR < HRL (e.g., 44 BPM), pacing starts,
  (iii) Once pacing resumes, A–A interval = 1200 ± 8 ms (i.e., resumes at LRL, not HRL),
  (iv) Pulse width matches the programmed value; no ventricular output.
- FAIL otherwise.

Figures (Hysteresis ON)

Figure AAI-2.1 (≈ HRL): Intrinsic = 45 BPM; no blue paces (inhibited above the hysteresis threshold).

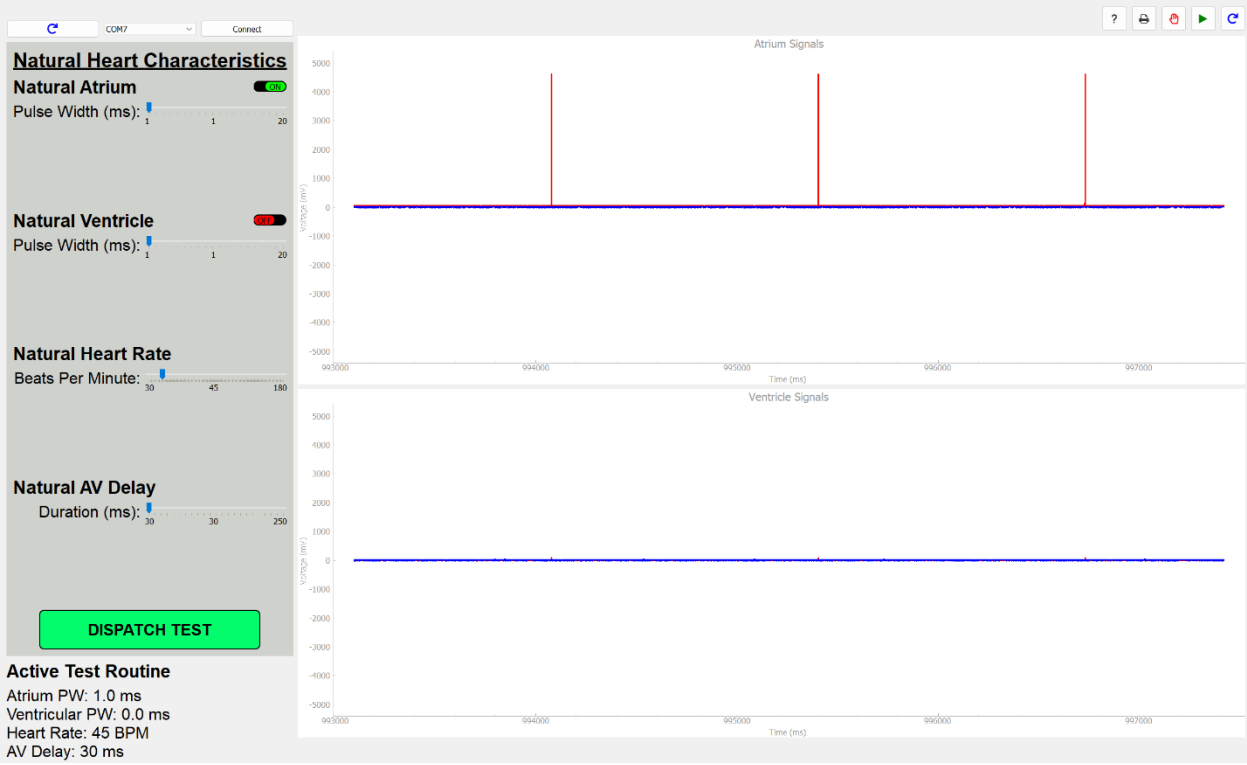
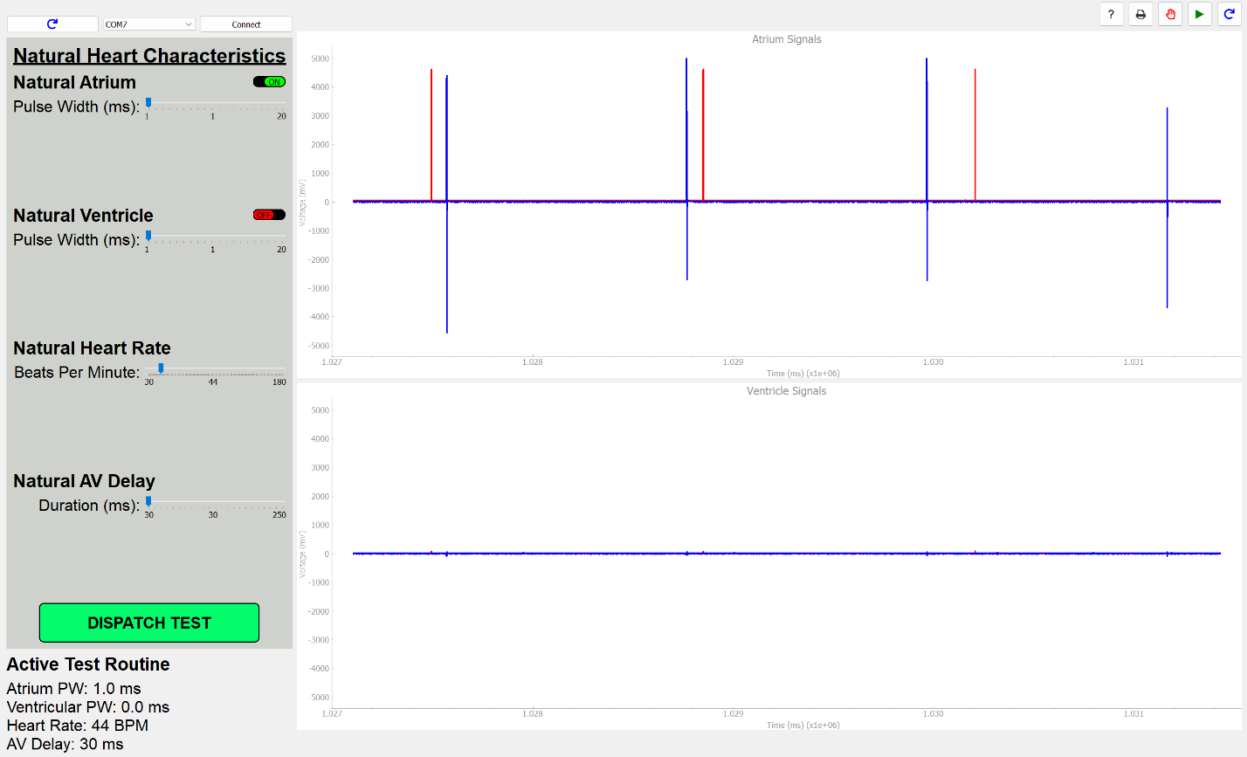


Figure AAI-2.2 (< HRL): Intrinsic 44 BPM; pacing begins; blue paces present.



Results (Hysteresis ON)

Pass. With hysteresis enabled, no pacing occurs at = 45 BPM (above HRL). Dropping to 44 BPM initiates pacing and the measured A–A returns to ~1200 ms, confirming that pacing resumes at the LRL once hysteresis is triggered. Pulse morphology remains correct and no ventricular output appears. Criteria (i)–(iv) are met.

AAI Tests Results Table

Test	Condition	Expected	Observed	Result
AAI-1	100 BPM, 50 BPM	Inhibit	No blue paces	Pass
AAI-1	49BPM, 30 BPM	Pace @ LRL & A–A = 1200 ± 8 ms	~1200 ms	Pass
AAI-2	=45 BPM (HRL)	Inhibit	No blue paces	Pass
AAI-2	<HRL (44 BPM)	Start pacing @ LRL	Blue paces, ~1200 ms	Pass

3.4.4. VVI Tests

3.4.4.1. *VVI-1 (Hysteresis OFF)*

Test Objective
Verify that in VVI mode the device paces only when the heart's rate drops below the LRL and confirm the pulse morphology.

Test Environment Setup

- HeartView
  - Natural Atrium OFF; Natural Ventricle ON.
  - Test points (BPM): 100, 50, 49, 30.
- Relevant Input Variables
  - Device mode: VVI
  - Lower_Rate_Limit: 50 bpm
  - Hysteresis: Disabled
  - Ventricular Amplitude / Pulse Width: nominal

Pass/Fail Logic

- PASS if:
  (i) No blue ventricular paces at 100 BPM and 50 BPM (inhibited at/above LRL),
  (ii) Blue paces appear once HR < 50 BPM and the V–V interval = 1200 ± 8 ms,
  (iii) Zoomed pulse width matches the programmed value within measurement resolution,
  (iv) No atrial output is produced.
- FAIL otherwise.

Figures (Hysteresis OFF)

Figure VVI-1.1 (100 BPM): Red ventricular beats at 100 BPM; no blue pacing (inhibited).

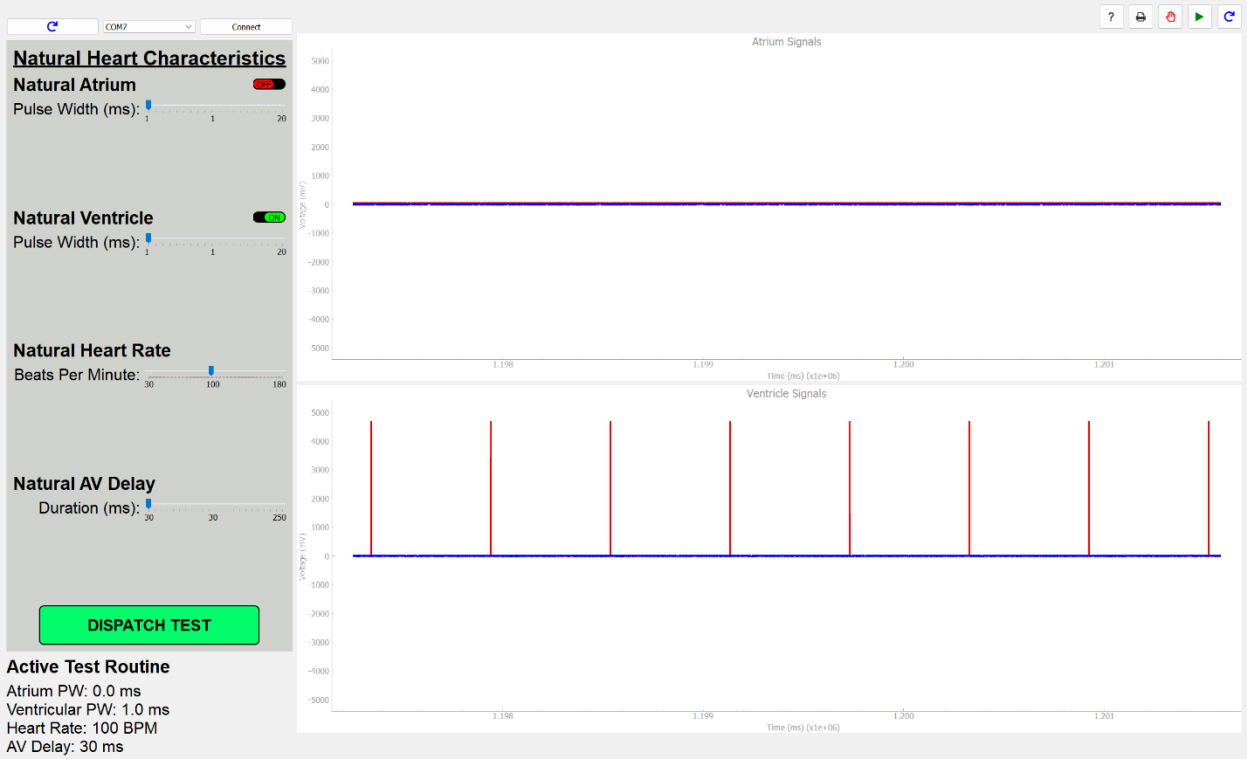


Figure VVI-1.2 (50 BPM = LRL): Red beats at 50 BPM; no blue pacing (still inhibited at the threshold).

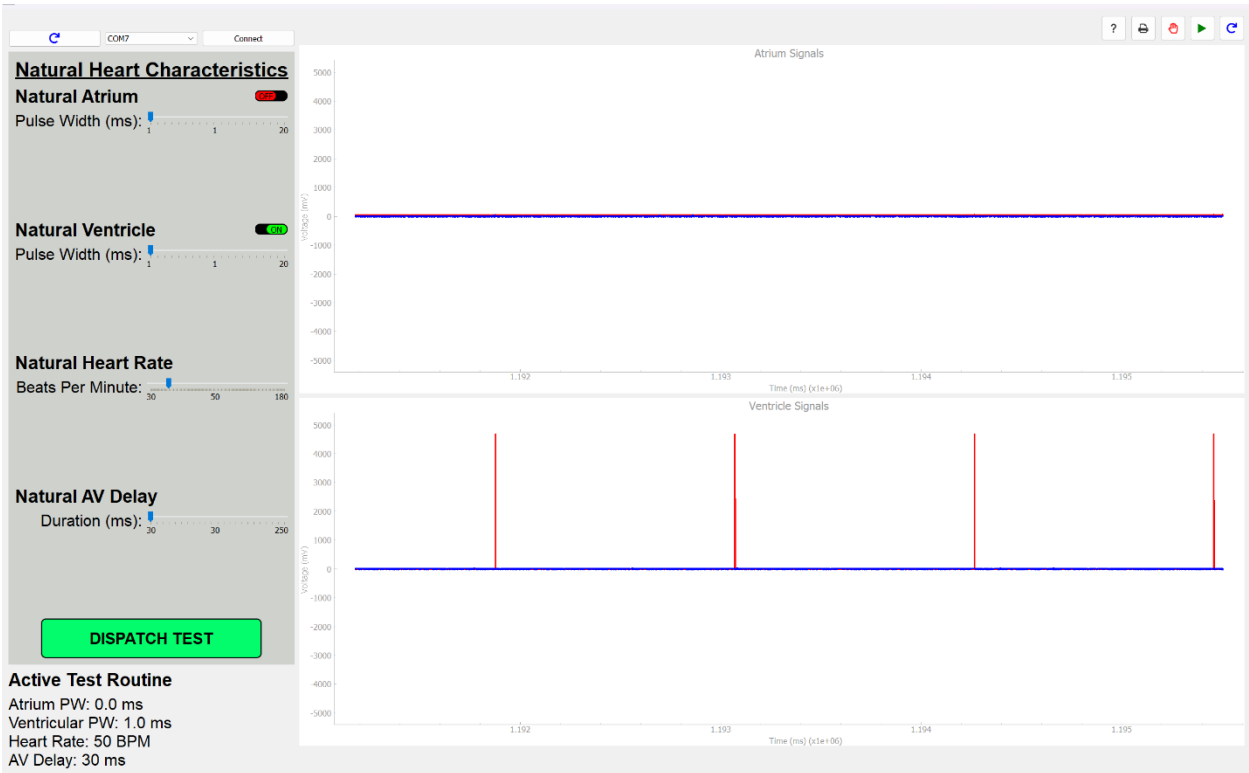


Figure VVI-1.3 (49 BPM < LRL): Pacing starts; blue ventricular paces present.

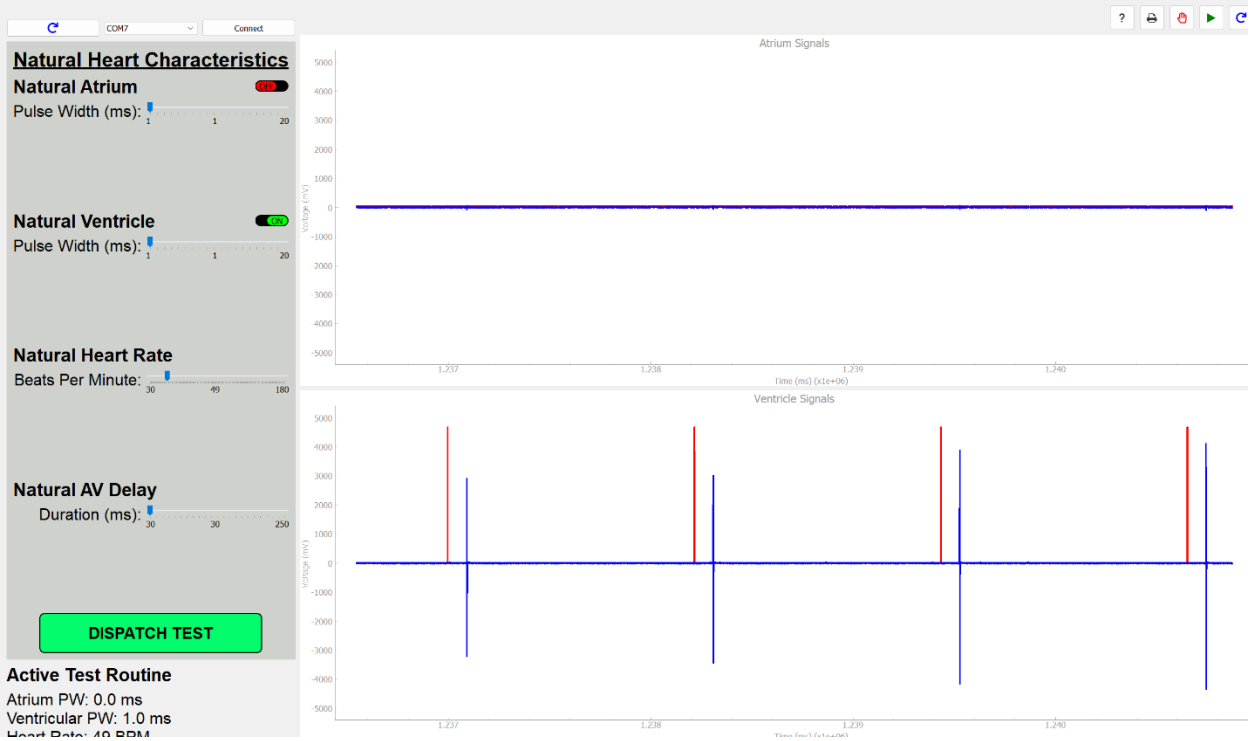


Figure VVI-1.4 (30 BPM): Continued pacing at LRL (blue paces periodic).

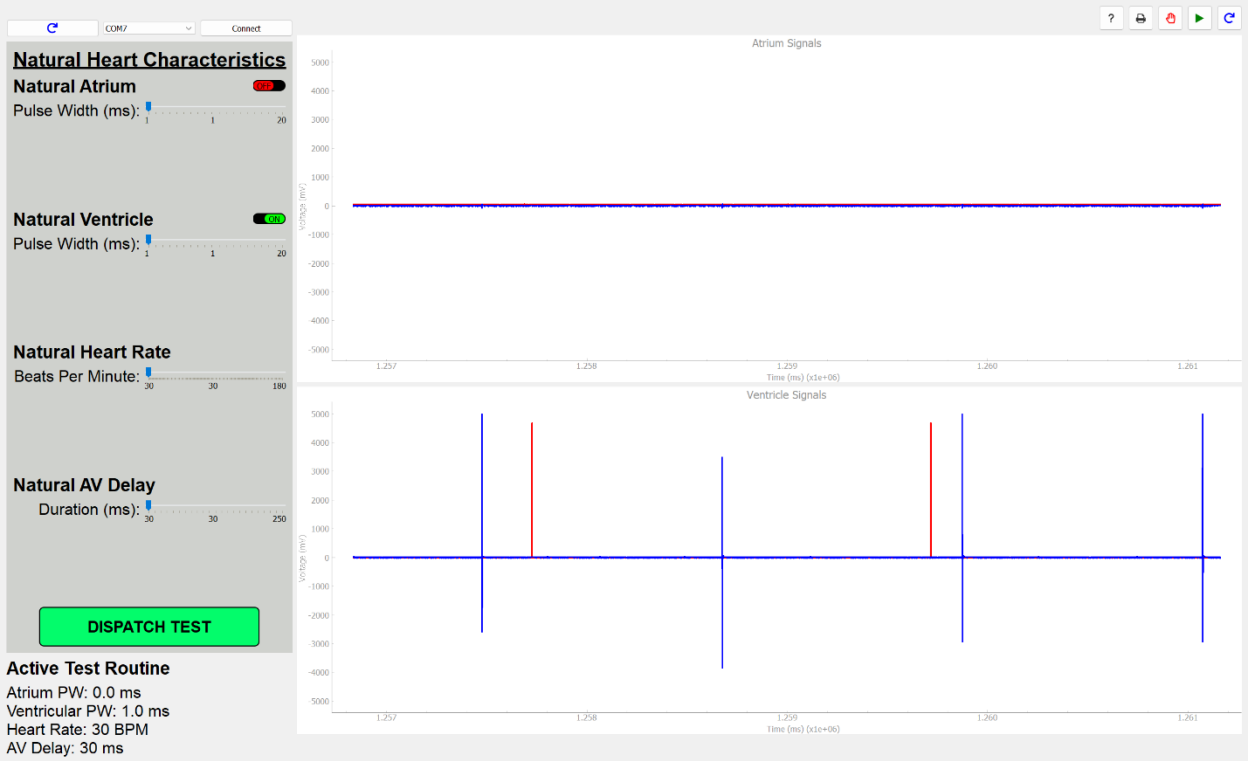
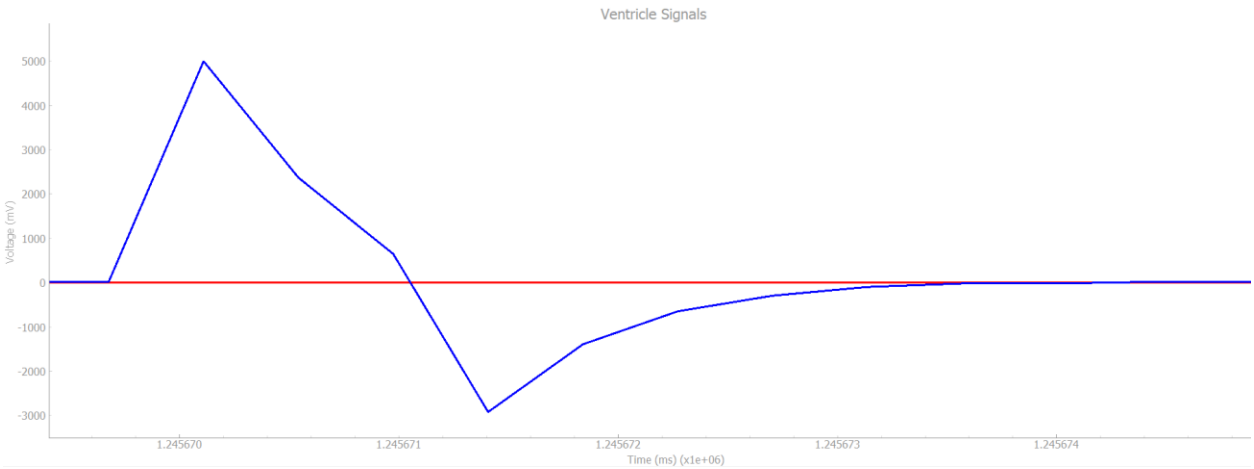


Figure VVI-1.5 (zoom): Single ventricular pace shows rise–fall–return to baseline (charge/discharge profile).



Results (Hysteresis OFF)

Pass. At 100 BPM and 50 BPM no blue ventricular paces are observed (criteria i). Dropping to 49 BPM triggers pacing and the measured V–V ≈ 1200 ms (criterion ii). The close-zoom pulse has the expected morphology and there is no atrial output (criteria iii–iv). Therefore, VVI inhibited behavior and LRL pacing are verified with hysteresis disabled.

3.4.4.2. VVI-2 (Hysteresis ON)

Test Objective

Verify that with Hysteresis enabled, pacing does not begin until the sensed heart's ventricular rate falls below the Hysteresis Rate Limit (HRL), and when it does, pacing resumes at the LRL.

Test Environment Setup

- HeartView
    - Natural Atrium OFF; Natural Ventricle ON.
    - Test points (BPM): 45 BPM (= HRL), then 44 BPM (< HRL).
- Relevant Input Variables
    - Device mode: VVI
    - Lower_Rate_Limit: 50 bpm
    - Hysteresis: Enabled
    - Hysteresis_Rate_Limit (HRL): = 45 bpm (per figures)
    - Ventricular Amplitude / Pulse Width: nominal

Pass/Fail Logic

- PASS if:
  (i) At HRL (≈ 45 BPM) there are no blue ventricular paces (still inhibited),
  (ii) When intrinsic HR < HRL (e.g., 44 BPM), pacing starts,
  (iii) Once pacing resumes, V–V interval = 1200 ± 8 ms (pacing resumes at LRL, not HRL),
  (iv) Pulse width matches the programmed value; no atrial output.
- FAIL otherwise.

Figures (Hysteresis ON)

Figure VVI-2.1 (= HRL): Intrinsic = 45 BPM; no blue ventricular paces (inhibited above the hysteresis threshold).

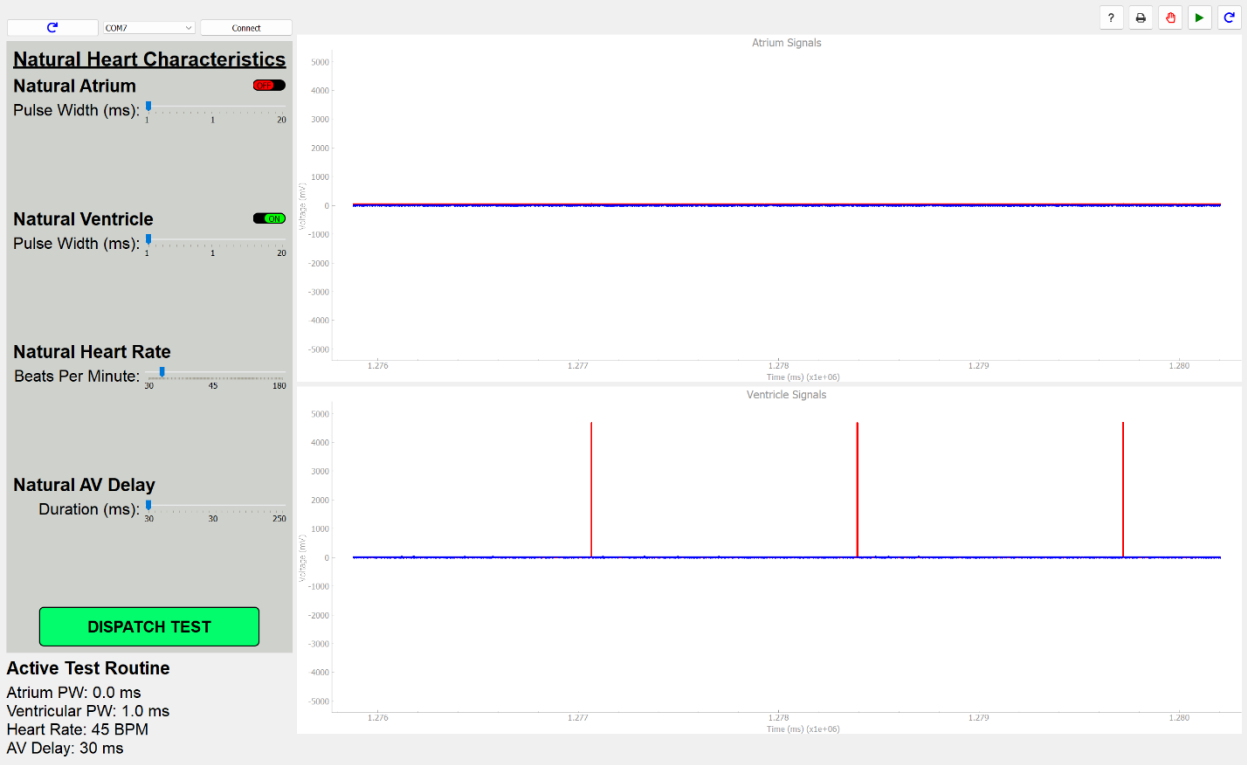
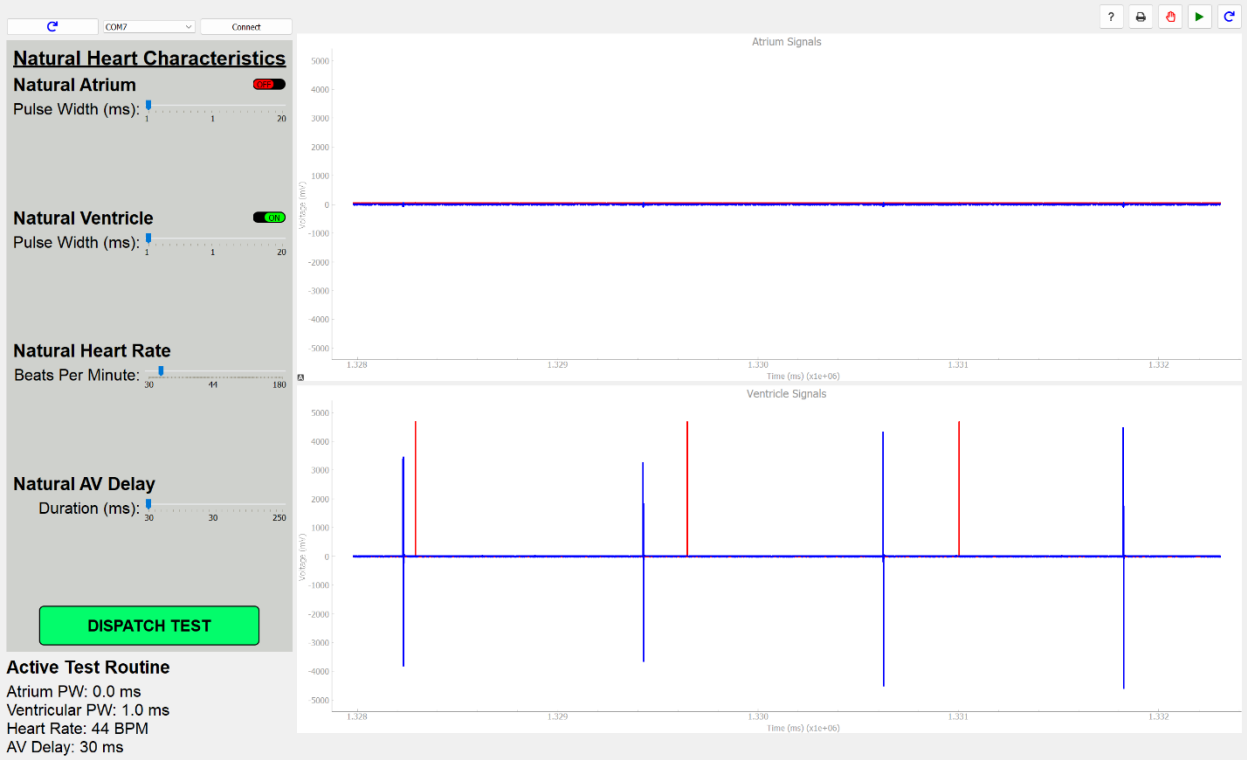


Figure VVI-2.2 (< HRL): Intrinsic 44 BPM; pacing begins; blue ventricular paces present.



Results (Hysteresis ON)

Pass. With hysteresis enabled, no pacing occurs at = 45 BPM (above HRL). Dropping to 44 BPM initiates pacing and V–V ≈ 1200 ms, confirming pacing resumes at the LRL once hysteresis is triggered. Pulse morphology remains correct and there is no atrial output. Criteria (i)–(iv) are met.

VVI Tests Results Table

Test	Condition	Expected	Observed	Result
VVI-1	100 BPM, 50 BPM	Inhibit	No blue ventricular paces	Pass
VVI-1	49 BPM, 30 BPM	Pace @ LRL & V–V = 1200 ± 8 ms	~1200 ms	Pass
VVI-2	=45 BPM (HRL)	Inhibit	No blue ventricular paces	Pass
VVI-2	<HRL (44 BPM)	Start pacing @ LRL	Blue paces, ~1200 ms	Pass

3.4.5. AOOR Tests

Test Objective:
Verify that in AOOR mode, the device delivers atrial pacing at the programmed Lower Rate Limit (LRL) when the device is at rest. Verify that the pacing rate increases towards the Maximum Sensor Rate (MSR) when the activity magnitude exceeds the set activity threshold, and recovers to the LRL when activity ceases.

Test Environment Setup:

- HeartView
    - Natural Atrium ON; Natural Ventricle OFF.
    - Natural Heart Rate: 60 BPM (LRL)
- Relevant Input Variables
    - Device mode: AOOR
    - Lower_Rate_Limit: 60 ppm (1000 ms)
    - Maximum_Sensor_Rate: 120 ppm (500 ms)
    - Activity Threshold: Med
    - Response Factor: 8
    - Amplitude: nominal
    - Atrial Pulse Width: nominal
    - Action: The FRDM board is shaken rhythmically to simulate patient activity.

Pass/Fail Logic:

- PASS if: (i) Rest Phase: Blue A-A interval ≈ 1000 ±10ms (60 ppm) when the board is stationary. (ii) Activity Phase: Blue A-A interval decreases (rate increases) significantly below 1000 ms when the board is shaken, approaching the MSR interval (500 ms). (iii) Recovery Phase: Blue A-A interval gradually returns to approximately 1000 ms after shaking stops. (iv) Pulse width matches programmed value within measurement resolution. (v) No ventricular activity is produced.
- FAIL otherwise.

Figure AOOR-1 (Rest): With the board stationary, blue atrial paces occur at the programmed LRL (60 ppm), showing a consistent interval of ~1000 ms.

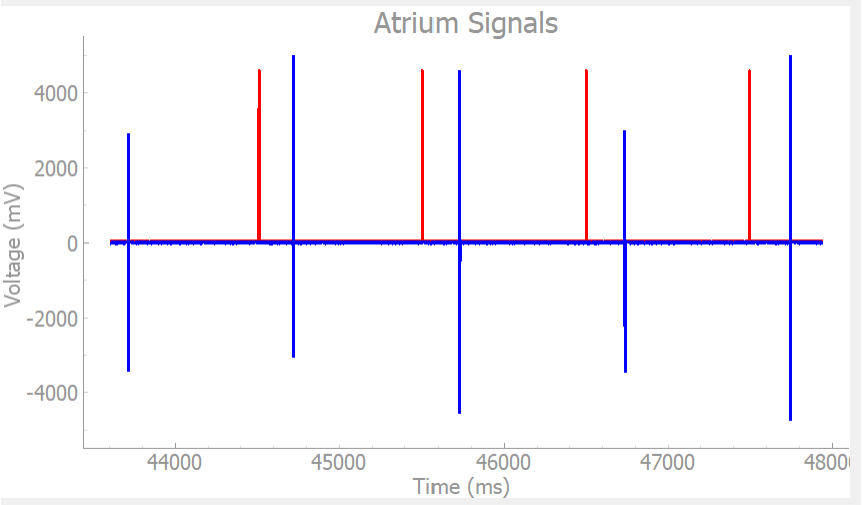


Figure AOOR-2 (Activity): As the board is shaken, the interval between consecutive blue paces shrinks (rate increases). The pacing rate rises above the LRL, demonstrating sensor-driven rate adaptation.

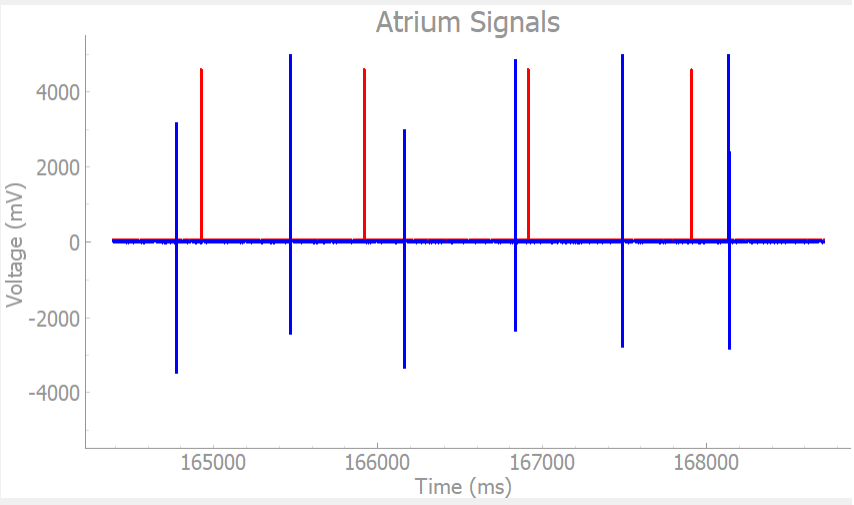
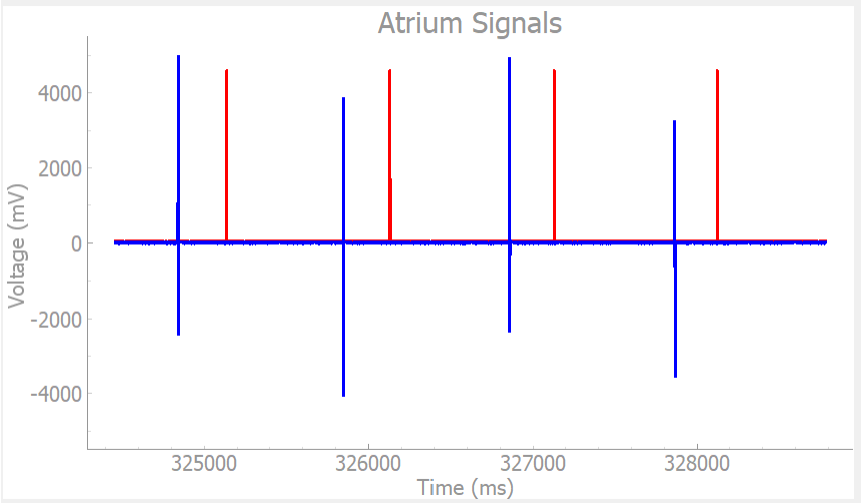


Figure AOOR-3 (Recovery/Morphology): After activity ceases, the pacing rate gradually slows back to the LRL. The output shows the expected rise, fall, and return to baseline, consistent with the shield's charge/discharge sequence.



Results.
Based on Figures AOOR-1 through AOOR-3, the device passes the AOOR functional test. At rest, the blue atrial pace events occur at a constant interval of approximately 1000 ms (60 ppm). Upon applying physical activity (shaking), the pacing interval shortened to approximately 500-600 ms, confirming the rate increased towards the programmed Maximum Sensor Rate (120 ppm). Once activity stopped, the rate smoothly returned to the LRL.

Therefore, criteria (i)–(v) are met, and AOOR rate-adaptive pacing is verified. This demonstrates conformance to the requirement that the pulse rate increases once sufficient activity is detected.

3.4.6. VOOR Tests

Test Objective:

Verify that in VOOR mode, the device delivers ventricular pacing at the programmed Lower Rate Limit (LRL) when the device is at rest. Verify that the pacing rate increases towards the Maximum Sensor Rate (MSR) when the activity magnitude exceeds the set activity threshold and recovers to the LRL when activity ceases.

Test Environment Setup:

- HeartView
    - Natural Atrium OFF; Natural Ventricle OFF.
    - Natural Heart Rate: 60 BPM (LRL)
- Relevant Input Variables

- Device mode: VOOR
- Lower_Rate_Limit: 60 ppm (1000 ms)
- Maximum_Sensor_Rate: 120 ppm (500 ms)
- Activity Threshold: Med
- Response Factor: 8
- Amplitude: nominal
- Ventricular Pulse Width: nominal

Pass/Fail Logic:

- PASS if: (i) Rest Phase: Blue V-V interval ≈ 1000 ±10ms (60 ppm) when the board is stationary. (ii) Activity Phase: Blue V-V interval decreases (rate increases) significantly below 1000 ms when the board is shaken, approaching the MSR interval (500 ms). (iii) Recovery Phase: Blue V-V interval gradually returns to approximately 1000 ms after shaking stops. (iv) Pulse width matches programmed value within measurement resolution. (v) No atrial activity is produced.
- FAIL otherwise.

Figure VOOR-1 (Rest): With the board stationary, blue ventricular paces occur at the programmed LRL (60 ppm), showing a consistent interval of ~1000 ms.

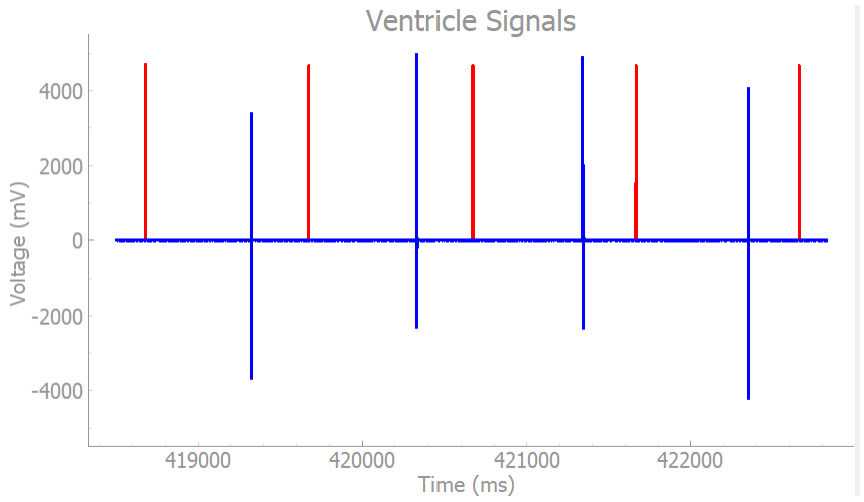


Figure VOOR-2 (Activity): As the board is shaken, the interval between consecutive blue paces shrinks (rate increases). The pacing rate rises above the LRL, demonstrating sensor-driven rate adaptation.

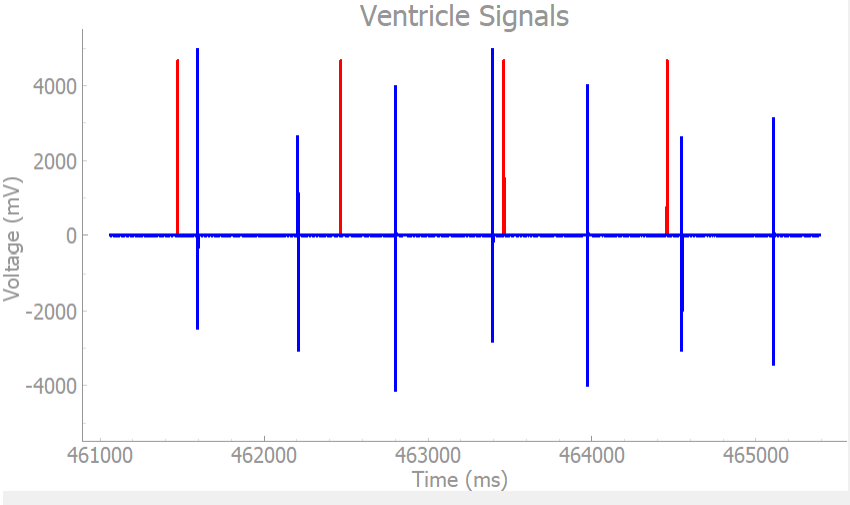
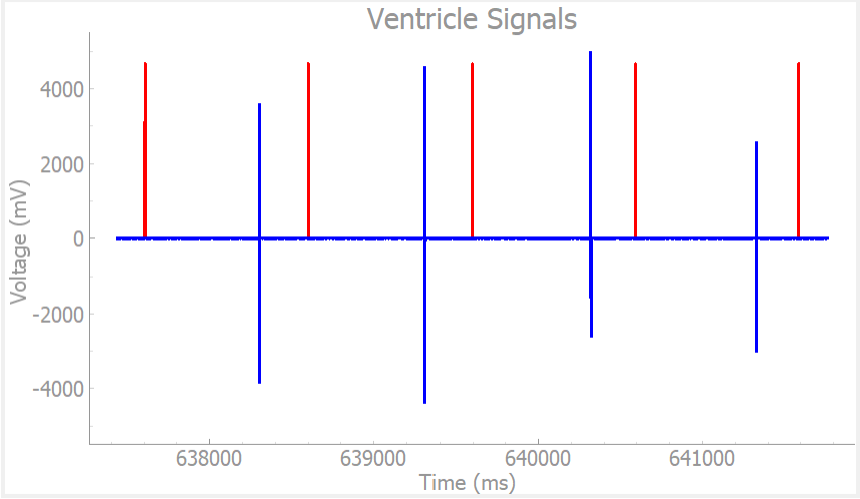


Figure VOOR-3 (Recovery): After activity ceases, the pacing rate gradually slows back to the LRL. The output shows the expected rise, fall, and return to baseline, consistent with the shield's charge/discharge sequence.



Results:

Based on Figures VOOR-1 through VOOR-3, the device passes the VOOR functional test. At rest, the blue ventricular pace events occur at a constant interval of approximately 1000 ms (60 ppm). Upon applying physical activity (shaking), the pacing interval shortened to approximately 500-600 ms, confirming the rate increased towards the programmed Maximum Sensor Rate (120 ppm). Once activity stopped, the rate smoothly returned to the LRL.

Therefore, criteria (i)–(v) are met, and VOOR rate-adaptive pacing is verified. This demonstrates conformance to the requirement that the pulse rate increases once sufficient activity is detected.

3.4.7. AAIR Tests

Test Objective:

Verify that in AAIR mode, the device delivers atrial pacing at the programmed Lower Rate Limit (LRL) when the intrinsic atrial rate is below the LRL and the device is at rest. Verify that the pacing rate increases towards the Maximum Sensor Rate (MSR) when activity exceeds the threshold and recovers to the LRL when activity ceases.

Test Environment Setup:

- HeartView
    - Natural Atrium ON; Natural Ventricle OFF.
    - Natural Heart Rate: 50 BPM (Below LRL to ensure pacing occurs).
- Relevant Input Variables
    - Device mode: AAIR
    - Lower_Rate_Limit: 60 ppm (1000 ms)
    - Maximum_Sensor_Rate: 120 ppm (500 ms)
    - Activity Threshold: Med
    - Response Factor: 8
    - Atrial Amplitude: nominal
    - Atrial Pulse Width: nominal

Pass/Fail Logic:

- PASS if: (i) Rest Phase: Blue atrial paces occur at the LRL (≈1000ms interval) because the intrinsic rate (50 BPM) is slower than the LRL (60 BPM) and no pacing occurs when natural heart rate > LRL (ii) Activity Phase: As the board is shaken, the blue A-A interval decreases (rate increases) significantly below 1000 ms, overriding the LRL and approaching the MSR (500 ms). (iii) Recovery Phase: After shaking stops, the blue A-A interval gradually widens back to ≈1000 ms. (iv) Pulse width

matches programmed value within measurement resolution. (v)No ventricular activity is produced.

- FAIL otherwise.

Figures:

Figure AAIR-1 (Above LRL): With the board stationary and natural heart rate at 61 BPM, no blue atrial paces occur.

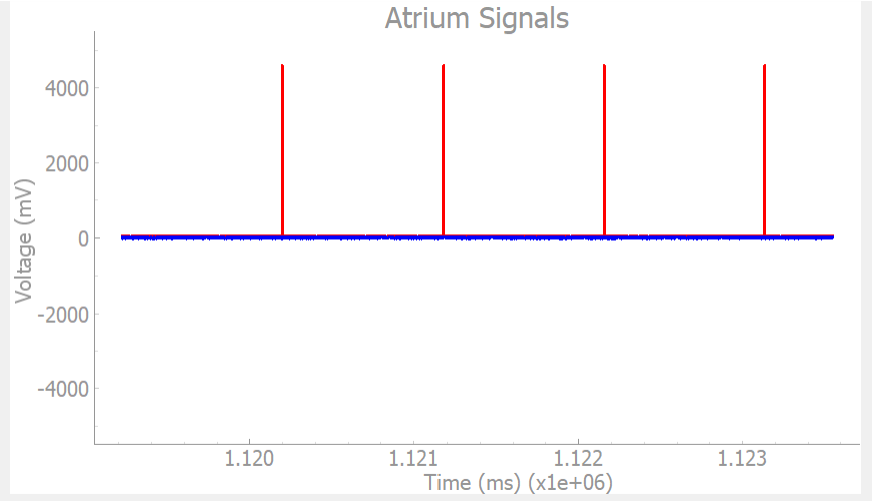


Figure AAIR-2 (Rest): With the board stationary and natural heart rate at 50 BPM, blue atrial paces occur at the programmed LRL (60 ppm / 1000 ms).

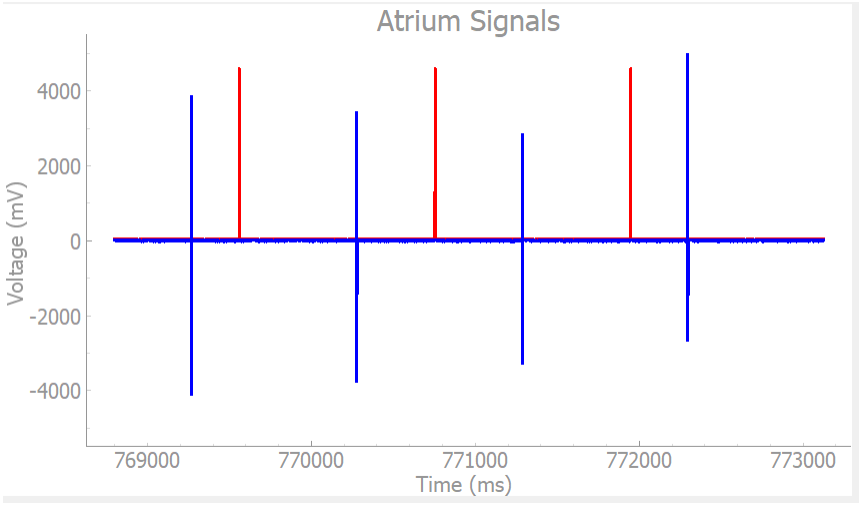


Figure AAIR-3 (Activity): As the board is shaken, the interval between blue paces shrinks (rate increases) despite the natural heart rate remaining constant. The pacing rate rises towards the MSR.

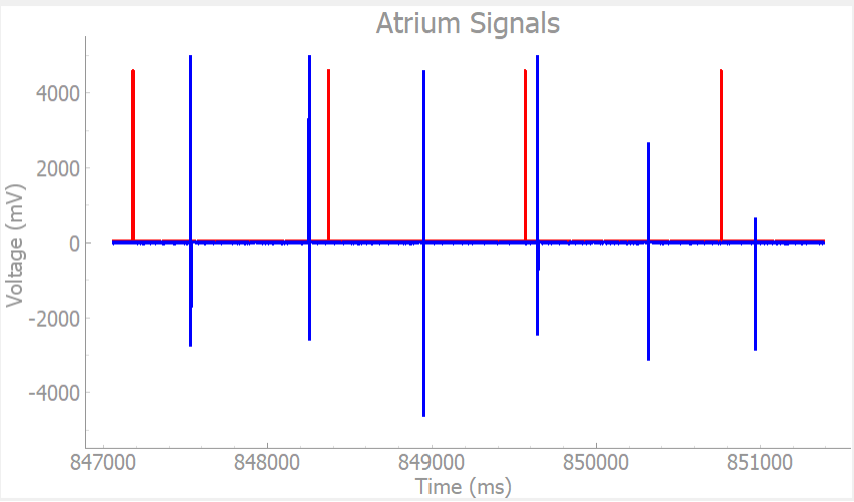
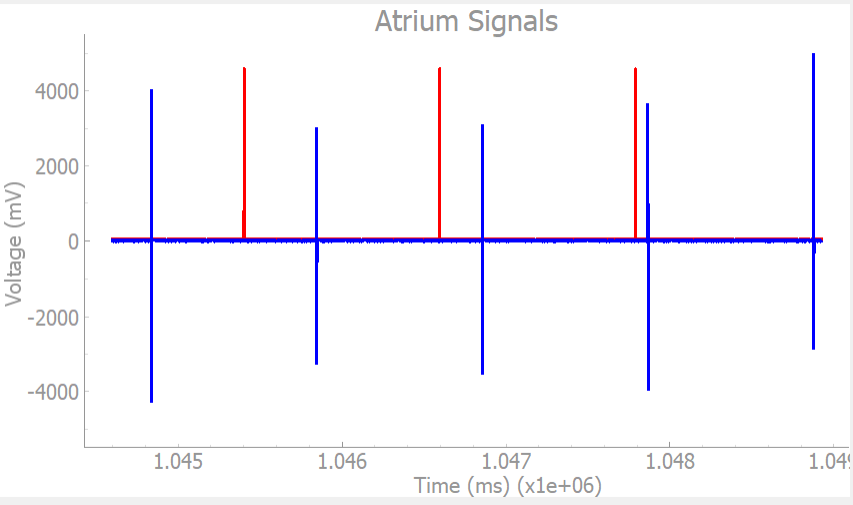


Figure AAIR-4 (Recovery): After activity ceases, the pacing rate gradually slows back to the LRL. The pulse waveform shows the expected rise, fall, and return to baseline.



Results:

Based on Figures AAIR-1 through AAIR-4, the device passes the AAIR functional test. At rest, the device paced the atrium at the LRL (60 ppm) because the intrinsic rate was too slow. Upon applying physical activity, the pacing rate increased significantly (interval shortened to ~500-600 ms), demonstrating that the Sensor Indicated Rate (SIR) took control over the LRL. Once activity stopped, the rate smoothly recovered to the LRL. No ventricular output is observed.

Therefore, criteria (i)–(v) are met, and AAIR rate-adaptive pacing is verified.

3.4.8. VVIR Tests

Test Objective:

Verify that in VVIR mode, the device delivers atrial pacing at the programmed Lower Rate Limit (LRL) when the intrinsic ventricular rate is below the LRL and the device is at rest. Verify that the pacing rate increases towards the Maximum Sensor Rate (MSR) when activity exceeds the threshold and recovers to the LRL when activity ceases.

Test Environment Setup:

- HeartView
    - Natural Atrium OFF; Natural Ventricle ON.
    - Natural Heart Rate: 50 BPM (Below LRL to ensure pacing occurs).
- Relevant Input Variables
    - Device mode: VVIR
    - Lower_Rate_Limit: 60 ppm (1000 ms)
    - Maximum_Sensor_Rate: 120 ppm (500 ms)
    - Activity Threshold: Med
    - Response Factor: 8
    - Ventricular Amplitude: nominal
    - Ventricular Pulse Width: nominal

Pass/Fail Logic:

- PASS if: (i) Rest Phase: Blue ventricular paces occur at the LRL (≈1000ms interval) because the intrinsic rate (50 BPM) is slower than the LRL (60 BPM) and no pacing occurs when natural heart rate > LRL (ii) Activity Phase: As the board is shaken, the blue V-V interval decreases (rate increases) significantly below 1000 ms, overriding the LRL and approaching the MSR (500 ms). (iii) Recovery Phase: After shaking stops, the blue V-V interval gradually widens back to ≈1000 ms. (iv) Pulse width matches programmed value within measurement resolution. (v)No ventricular activity is produced.
- FAIL otherwise.

Figure VVIR-1 (Above LRL): With the board stationary and natural heart rate at 61 BPM, no blue ventricular paces occur.

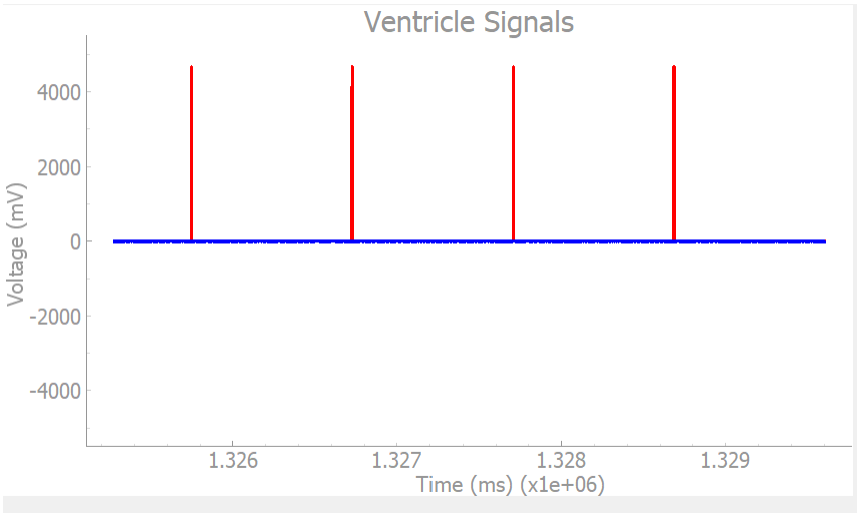


Figure VVIR-2 (Rest): With the board stationary and natural heart rate at 50 BPM, blue ventricular paces occur at the programmed LRL (60 ppm / 1000 ms).

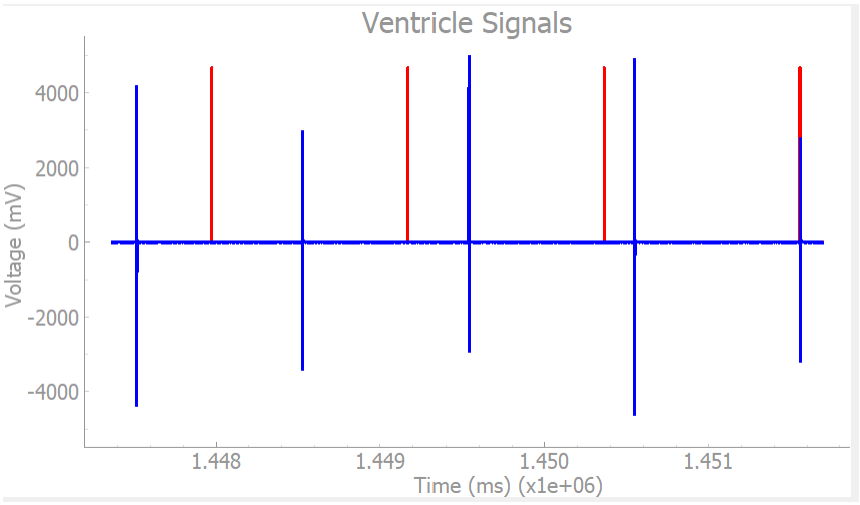


Figure VVIR-3 (Activity): As the board is shaken, the interval between blue paces shrinks (rate increases) despite the natural heart rate remaining constant. The pacing rate rises towards the MSR.

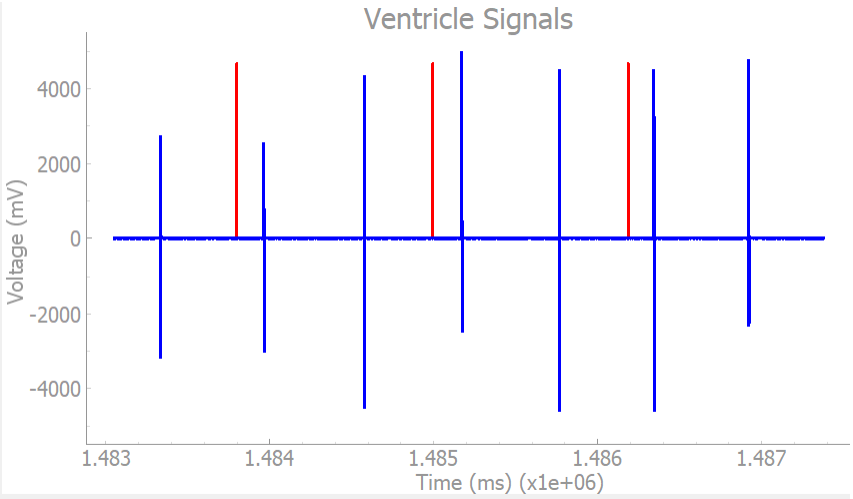
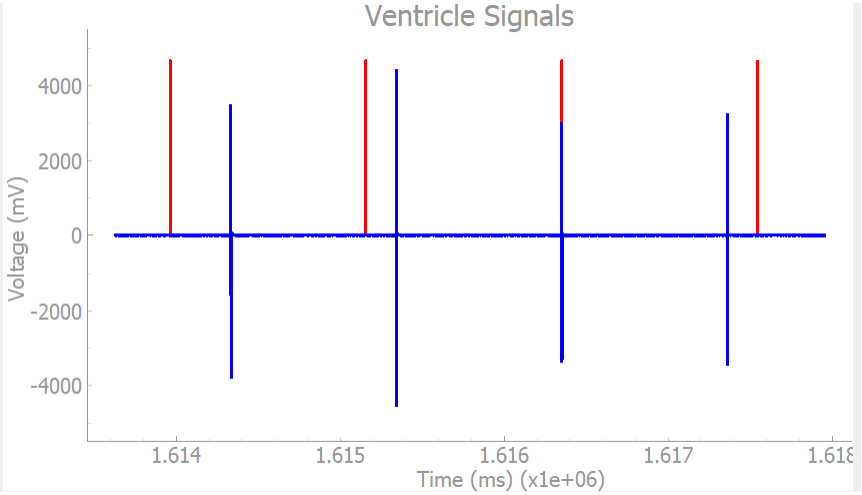


Figure VVIR-4 (Recovery): After activity ceases, the pacing rate gradually slows back to the LRL. The pulse waveform shows the expected rise, fall, and return to baseline.



Results:

Based on Figures VVIR-1 through VVIR-4, the device passes the VVIR functional test. At rest, the device paced the ventricle at the LRL (60 ppm) because the intrinsic rate was too slow. Upon applying physical activity, the pacing rate increased significantly (interval shortened to ~500-600 ms), demonstrating that the Sensor Indicated Rate (SIR) took control over the LRL. Once activity stopped, the rate smoothly recovered to the LRL. No atrial output is observed.

Therefore, criteria (i)–(v) are met, and VVIR rate-adaptive pacing is verified.

3.4.9. DDDR Tests

Test Objective:

Verify that in DDDR mode, the device delivers AV sequential pacing (Atrial Pace followed by Ventricular Pace) at the programmed Lower Rate Limit (LRL) when the intrinsic rate is below the LRL and the device is at rest. Verify that the pacing rate increases towards the Maximum Sensor Rate (MSR) when activity exceeds the threshold, while maintaining the programmed AV Delay.

Test Environment Setup:

- HeartView
    - Natural Atrium ON; Natural Ventricle ON.
    - Natural Heart Rate: 30 BPM (Below LRL to ensure the device paces both chambers).
- Relevant Input Variables
    - Device mode: DDDR
    - Lower_Rate_Limit: 80 ppm
    - Maximum_Sensor_Rate: 120 ppm (500 ms)
    - Fixed AV Delay: 150 ms
    - Activity Threshold: Med
    - Response Factor: 8
    - Atrial and Ventricular Amplitudes/Pulse Widths: nominal

Pass/Fail Logic:

- PASS if: (i) Rest Phase: Blue Atrial and Ventricular paces occur at the LRL (≈ 1000ms A-A interval) because the intrinsic rate (50 BPM) is slower than the LRL (60 BPM). (ii) AV Delay: A constant delay of ≈ 150ms is observed between the Atrial Pace and the Ventricular Pace. (iii) Activity Phase: As the board is shaken, the blue A-A interval decreases (rate increases) significantly below 1000 ms, overriding the LRL and approaching the MSR (500 ms). (iv) Recovery Phase: After shaking stops, the A-A interval gradually widens back to ≈ 1000ms. (v) Pulse widths match programmed values for both chambers.
- FAIL otherwise.

Figure DDDR-1 (Above LRL): With the board stationary and natural heart rate at 81 BPM, no blue atrial or ventricular paces occur.

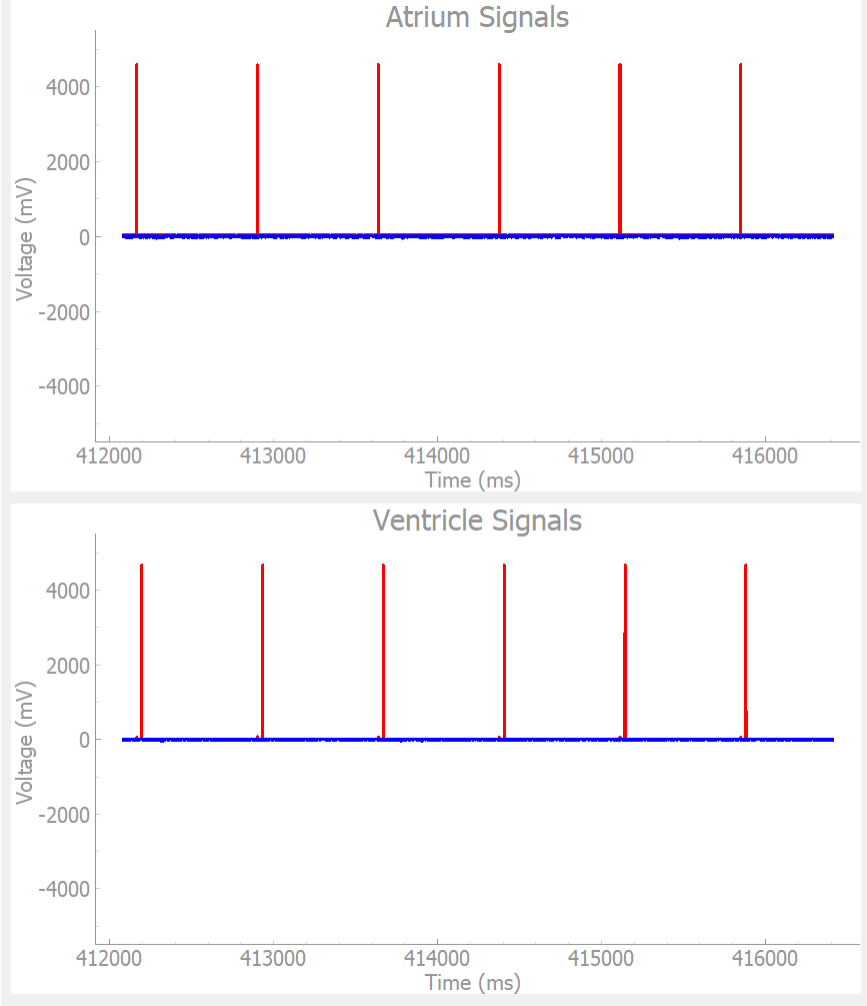


Figure DDDR-2 (Rest): With the board stationary and natural heart rate at 50 BPM, the device delivers AV sequential pacing at the LRL (80 ppm). The gap between the Atrial (top) and Ventricular (bottom) pace is consistent with the programmed AV Delay (150 ms).

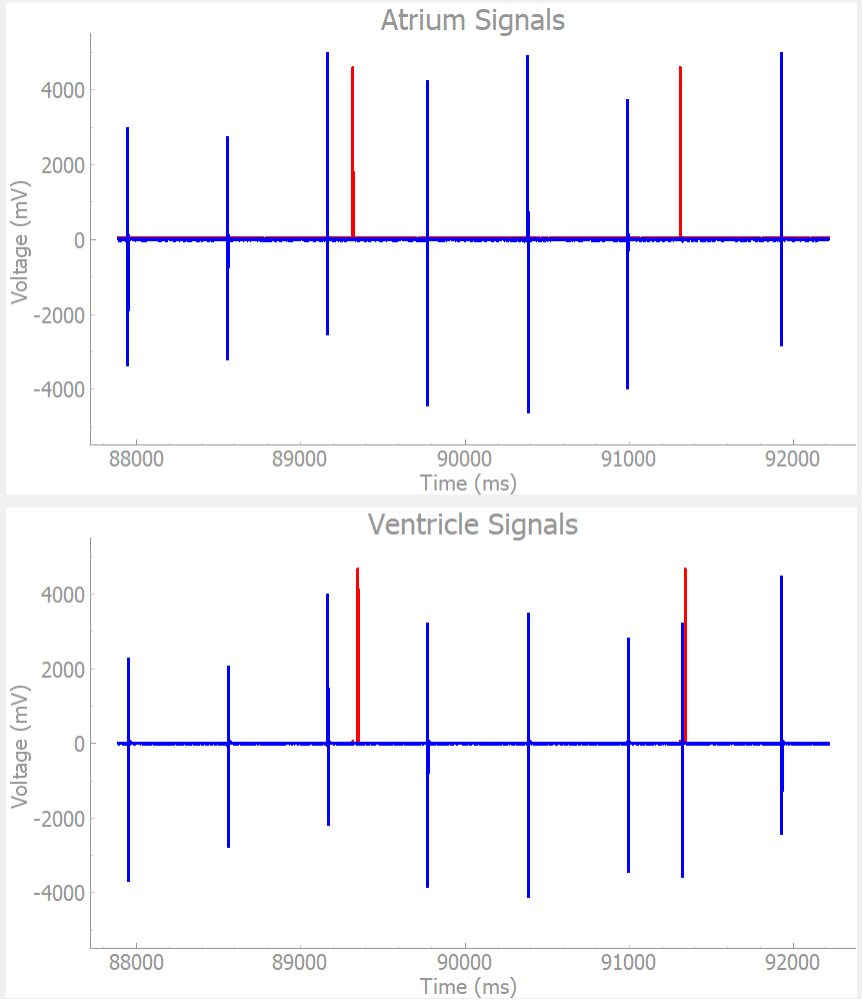


Figure DDDR-3 (Activity): As the board is shaken, the interval between the Atrial paces shrinks (rate increases) towards the MSR. Ventricular paces track the atrial paces, maintaining 1:1 synchrony.

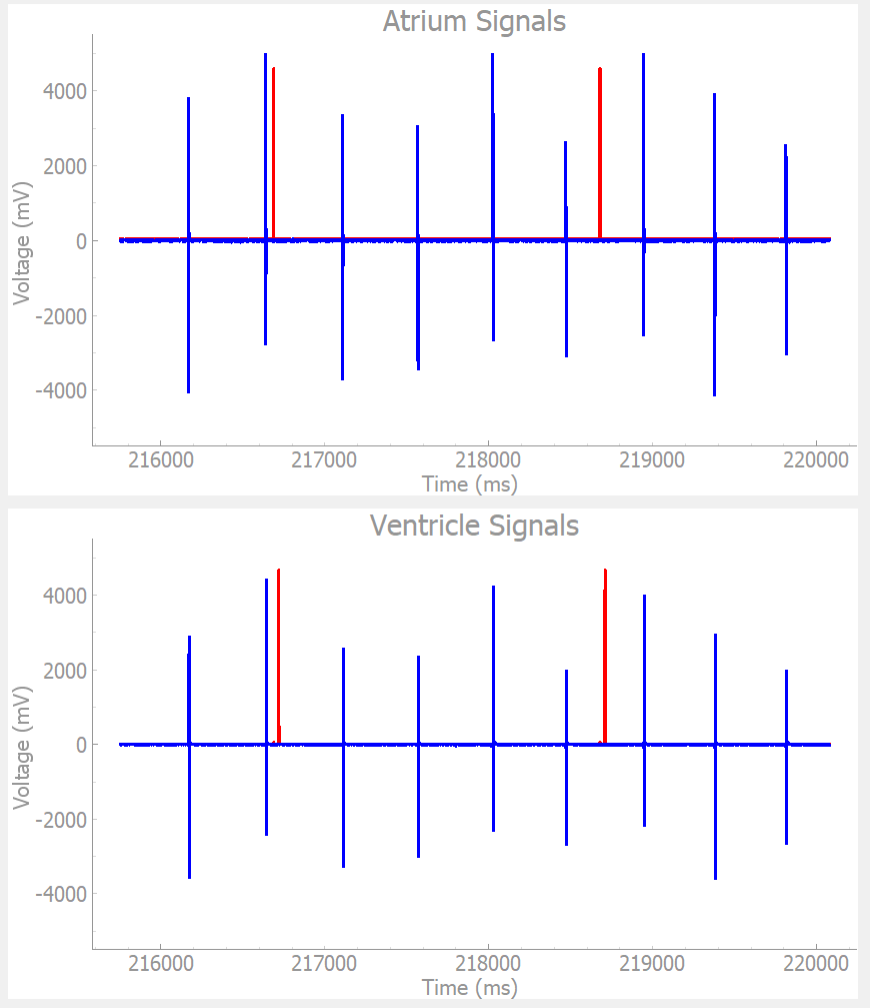


Figure DDDR-4 (Recovery/Morphology): After activity ceases, the pacing rate gradually slows back to the LRL. Both atrial and ventricular waveforms show the expected charge/discharge output.

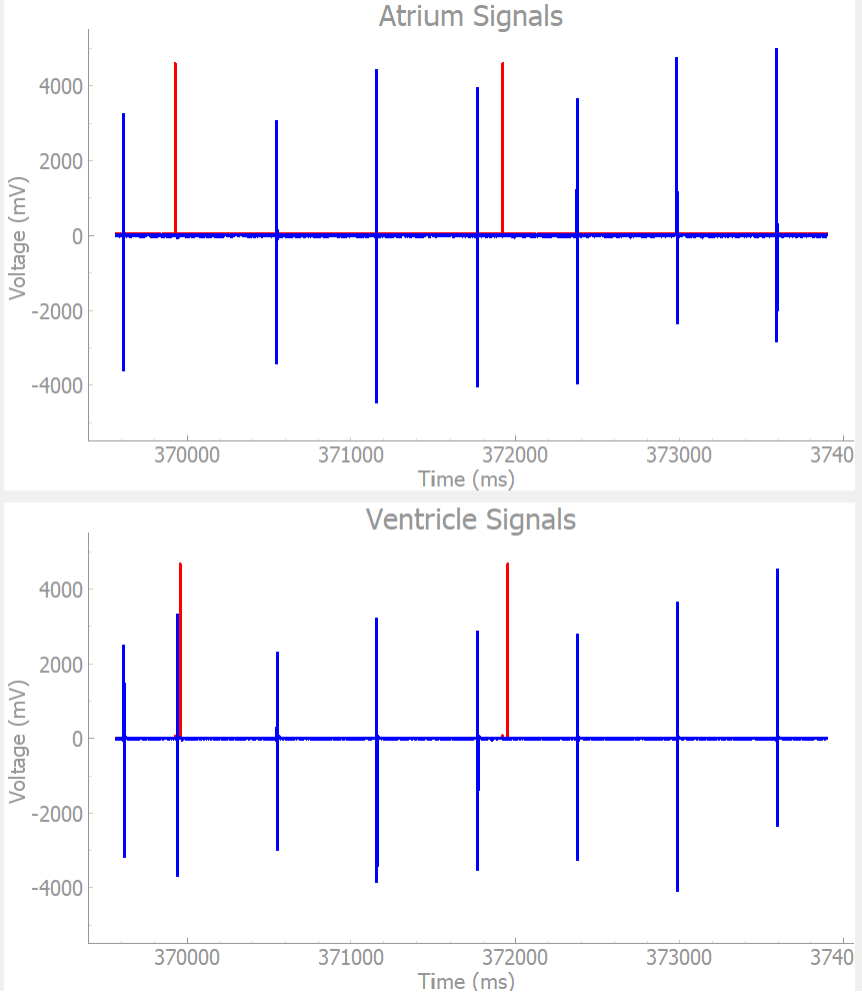
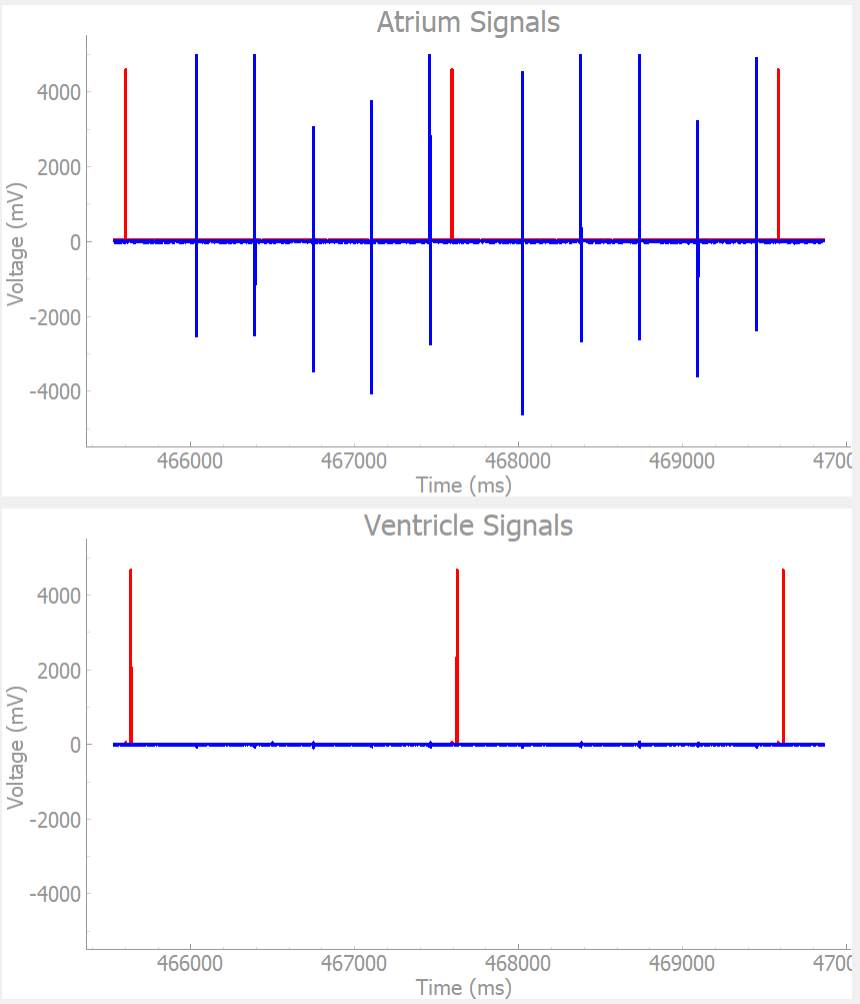


Figure DDDR-5 (Push-button): Once the button is pushed, ventricular pacing halts, leaving only pacing in the atrium.



Results:

Based on Figures DDDR-1 through DDDR-5, the device passes the DDDR functional test. At rest, the device paced both the atrium and ventricle at the LRL (80 ppm) with a consistent AV Delay of ~150 ms, satisfying the dual-chamber pacing requirement. Upon applying physical activity, the pacing rate increased significantly, demonstrating that the Sensor Indicated Rate (SIR) took control. The ventricle continued to pace after the atrium at the higher rate, maintaining AV synchrony. Once activity stopped, the rate smoothly recovered to the LRL.

Therefore, criteria (i)–(v) are met, and DDDR rate-adaptive pacing is verified.

3.4.10. DCM Testing

3.4.10.1. Login Page

Test Objective: Determine if the user can successfully log in when using the correct credentials that correspond to an existing account.

Pass/Fail Criteria:

- Pass If: The system redirects the user to the main page.
- Fail If: The system does not redirect the user to the main page.

Results: Pass, account was successfully created, and user was directed to the home page.

Test Objective: Determine if the user is denied access to the DCM when using incorrect credentials, or credentials that do not correspond to an existing account

Pass/Fail Criteria:

- Pass If: The system redirects the user to the main page.
- Fail If: The system does not redirect the user to the main page.

Results: Pass, when using incorrect credentials, the user was denied access to the system and was redirected to the login page.

3.4.10.2. Registration Page

Test Objective: Determine if the system will successfully create a new account (unique username, <10 users in database)

Pass/Fail Criteria:

- Pass If: The system does successfully create a new account.
- Fail If: The system does not successfully create a new account.

Results: Pass, after registering the account, the user was redirected to the login page.

Test Objective: Determine if the system can correctly handle account creation when there are 10 users in the database.

Pass/Fail Criteria:

- Pass If: The system notifies the user that the maximum number of accounts have been created and redirects the user to the login page.

- Fail If: The system does not notify the user or does not redirect the user to the login page.

Results: Pass, the user is notified and denied permission to create an account.

Test Objective: Determine if the system can correctly handle account creation when users register an account with a username that already exists.

Pass/Fail Criteria:

- Pass If: The system notifies the user that there is an account with the same username, and the user is redirected to the login page
- Fail If: The system does not notify the user or does not redirect the user to the login page.

Results: Pass, the user is notified and denied permission to create an account.

3.4.10.3. Input Validation

Test Objective: Determine if the system will allow users to edit the parameters using the up and down arrows.

Pass/Fail Criteria:

- Pass If: The system increments or decrements the parameter value by one step every time the up or down arrow is pressed.
- Fail If: The system does not increment or decrements the parameter value by one step every time the up or down arrow is pressed.

Results: Pass, the user can increment or decrement the parameter value.

Test Objective: Determine if the system will not increment or decrement the parameter value if it is at the maximum or minimum value.

Pass/Fail Criteria:

- Pass If: The system does not allow the user to increment or decrement the parameter.
- Fail If: The system allows the user to increment or decrement the parameter.

Results: Pass, the system does not allow the user to press the up or down arrows when the parameter is at the maximum or minimum value.

Test Objective: Determine if the system can save the users modified parameters.

Pass/Fail Criteria:

- Pass If: The system can locally save the users modified parameters and can be accessed after signing out.
- Fail If: The system can not locally save the users modified parameters.

Results: Pass, the system was able to successfully save the parameters access them after the user signed in.

Test Objective: Determine if the system can reset the parameter to its nominal value.

Pass/Fail Criteria:

- Pass If: The parameters were reset to the nominal value when the “Reset to Nominal Values” button is pressed.
- Fail If: The parameters were not reset to the nominal value when the “Reset to Nominal Values” button is pressed.

Results: Pass, the system was able to reset all the parameters to their nominal value.

Test Objective: Determine if the system can handle a lower rate limit that does not exceed the upper rate limit.

Pass/Fail Criteria:

- Pass If: The lower rate limit cannot be adjusted below the upper limit.
- Fail If: The lower rate limit can be above the upper rate limit.

Result: Pass, the system does not allow the lower rate limit to be below the upper rate limit.

Test Objective: Determine if the system can handle a lower rate limit that does not exceed the upper rate limit.

Pass/Fail Criteria:

- Pass If: The lower rate limit cannot be adjusted below the upper limit.
- Fail If: The lower rate limit can be above the upper rate limit.

Result: Pass, the system

3.4.10.4. Reports

Test Objective: Determine if the parameter reports display the correct parameters that are set in the parameter table page.

Pass/Fail Criteria:

- Pass If: The parameters in the reports align with the saved parameters in the parameters table.
- Fail If: The parameters in the reports do not align with the saved parameters in the parameters table.

Results: Pass, the system was able to generate parameter reports that align with the parameters table page.

3.4.10.5. System Utilities

Test Objective: Determine if the system will successfully log the user out of the system and redirect them to the create new patient page when the “New Patient” button is pressed.

Pass/Fail Criteria

- Pass If: the system directs the user to the create new user page when the “New Patient” button is pressed.
- Fail If: the system does not direct the user to the create new user page.

Results: Pass, the system was able to direct the user to the create new user page.

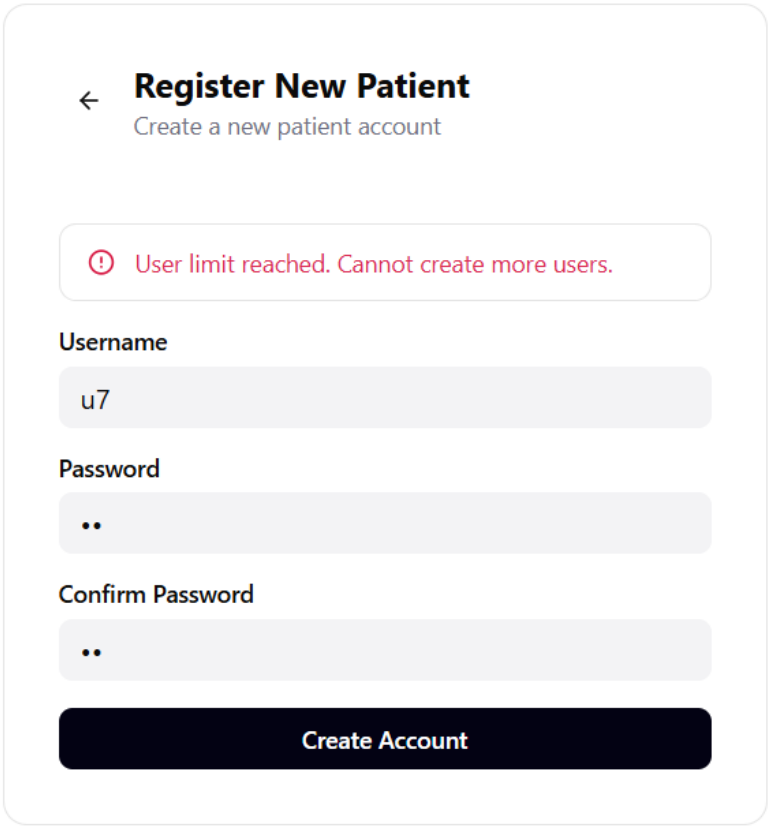


Figure 3121. Error message when >10 accounts created.

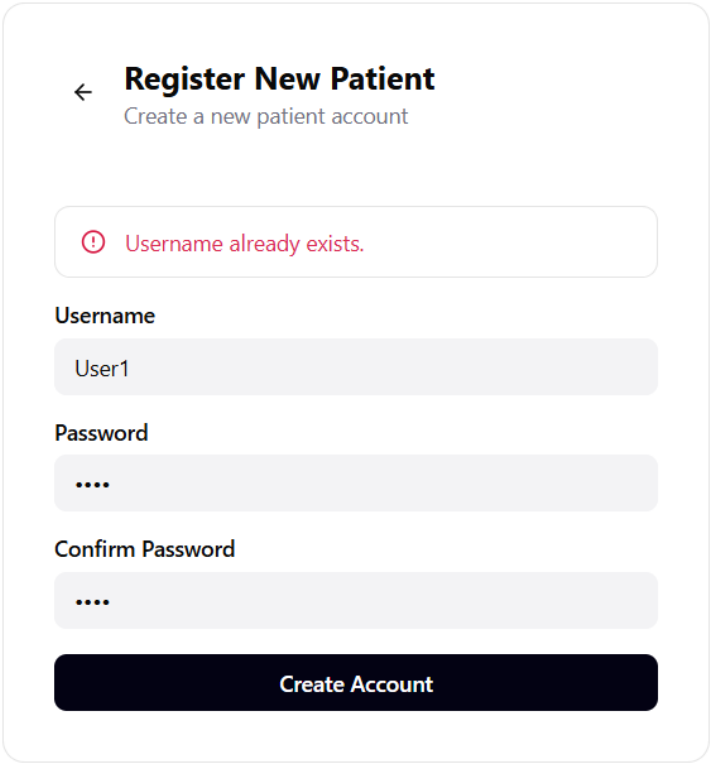


Figure 32 22. Error message when the username already exists.

3.5. GenAI Usage

- Used for some reformatting/rewording of some parts of documentation.
- Used to understand Simulink and State flow intricacies, in tandem with official documentation
- Used to help gain an understanding of natural heart characteristics and behaviour to help facilitate system development and safety concerns.
- Used to better understand and filter through the documentation for ease of finding relevant information. By uploading documents, we were able to use AI as a chatbot that pulled information from the material we provided.
- Used Figma to create the UI components and style components (button variants, typography scales), then converted the design into CSS files using Figma AI.
- Using AI to help with debugging errors in both DCM and Simulink.