CALYAN TECHNOLOGIES
Software Verification & Validation

SOFTWARE VERIFICATION TEST SUMMARY REPORT

Calyan P-300 VVIR Single Chamber Pacemaker

Pacing Engine | Main Processor | Bluetooth

Document ID:	RAW_DATA_15th_JAN	**Revision:**	1.0
Document Type:	Software Verification Test Summary	**Status:**	FINAL
Report Date:	2026-01-15	**Classification:**	Confidential

Software Versions Under Test

Component	Version	Description
Pacing Engine (PE)	1.05	Cardiac pacing control firmware
Main Processor (MP)	1.29	System control and communication firmware
Bluetooth (BT)	1.03	Wireless communication firmware

Test Execution Summary

Total Tests	Passed	Failed	Skipped	Pass Rate	Overall Result
74	**74**	**0**	**0**	**100.0%**	**PASS**

Revision History

Rev.	Date	Description of Change	Author	Approved
1.0	2026-01-15	Initial release - Automated validation test summary report	Test Automation	

Executive Summary

Purpose: This document provides a comprehensive summary of software verification testing performed on the Calyan P-300 VVIR Single Chamber Pacemaker device, covering three critical firmware subsystems: Pacing Engine (PE v1.05), Main Processor (MP v1.29), and Bluetooth (BT v1.03).

Scope: The verification testing validates that the software meets its specified requirements and functions correctly under defined operating conditions. Testing was performed using automated test scripts with comprehensive evidence capture and Allure reporting.

Results Summary:
• **Total Test Cases:** 74
• **Passed:** 74 (100.0%)
• **Failed:** 0
• **Skipped:** 0
• **Total Execution Time:** 3493.2 seconds

Conclusion: All 74 verification test cases have been executed successfully with a **100% pass rate**. The software under test has demonstrated full compliance with the specified requirements and acceptance criteria. Based on these results, the software is recommended for release.

Table of Contents

1. Purpose & Objectives

1.1 Purpose

This document provides a comprehensive summary of software verification testing performed on the Calyan P-300 VVIR Single Chamber Pacemaker device. The purpose is to document the execution and results of all verification tests, demonstrating compliance with specified requirements and acceptance criteria as defined in the applicable software requirements specifications.

1.2 Objectives

The objectives of this verification testing are to:
• Verify that software functions correctly according to specified requirements
• Demonstrate traceability between test cases and requirements
• Document test execution evidence for regulatory submission
• Identify and document any deviations or observations
• Provide formal evidence of verification completion for quality assurance

2. Device Under Test

Attribute	Value
Device Name	Calyan P-300 VVIR Single Chamber Pacemaker
Device Classification	Class III Active Implantable Medical Device
Intended Use	Long-term ventricular pacing therapy
Hardware Platform	Calyan P-300 Implantable Pulse Generator
PE Firmware Version	1.05
MP Firmware Version	1.29
BT Firmware Version	1.03
Test Execution Date	2026-01-15
Test Environment	Hardware-in-the-Loop (HIL) Automated Test System

3. Reference Documents

Document ID	Title	Description

REF-001	PE Allure Test Report	PE_Raw_Data_Report.html - Detailed PE test logs and evidence
REF-002	MP Allure Test Report	MP_Raw_data_Report.html - Detailed MP test logs and evidence
REF-003	BT Allure Test Report	BT_Raw_Data_Report.html - Detailed BT test logs and evidence
REF-004	Software Requirements Specification	System and subsystem software requirements
REF-005	Test Protocol Document	Verification test procedures and acceptance criteria
REF-006	Traceability Matrix	Requirements to test case traceability

4. Acronyms & Terminology

Term	Definition
V&V	Verification & Validation
MP	Main Processor / Microprocessor
PE	Pacing Engine
BT	Bluetooth
BLE	Bluetooth Low Energy
DIO	Digital Input/Output
POST	Power-On Self-Test
ACK	Acknowledgment
NACK	Negative Acknowledgment
CRC	Cyclic Redundancy Check
VVIR	Ventricular Rate Responsive Pacing Mode
AD2	Analog Discovery 2 (Digilent hardware interface)
HIL	Hardware-In-the-Loop
TP	Test Point
ECG	Electrocardiogram
UART	Universal Asynchronous Receiver-Transmitter
SDK	Software Development Kit
COM	Component Object Model
MAC	Media Access Control
JSON	JavaScript Object Notation
PNG	Portable Network Graphics
PDF	Portable Document Format
HTML	HyperText Markup Language

5. Test Environment & Configuration

5.1 Test Environment Overview

Testing was performed using an automated Hardware-in-the-Loop (HIL) test system. This system enables comprehensive software verification by interfacing directly with the device under test through standardized hardware interfaces and communication protocols. All test execution is automated using Python-based test scripts with Allure reporting for evidence capture.

5.2 Test Equipment and Software Tools

Equipment/Component	Model/Version	Details
Test Framework	pytest	Python testing framework
Test Reporting	Allure	Test result reporting and visualization
BLE Communication	Bleak Library	Cross-platform BLE communication
Hardware Interface	Digilent AD2	Analog Discovery 2 for signal capture
Python Version	3.x	Python runtime environment
Operating System	Windows	Test execution platform

5.3 Test Configuration Summary

MP Version Under Test:	1.29
PE Version Under Test:	1.05
BT Version Under Test:	1.03
Report Generation Date:	2026-01-15 07:21:02
Total Test Cases Executed:	74
Test Framework:	pytest with Allure Reporting
Report Type:	Formal Validation Summary
Execution Method:	Automated Test Scripts

6. Test Protocol & Acceptance Criteria

6.1 Test Methodology

Testing was performed using automated test scripts executed via the pytest framework. Each test case follows a structured approach:

- **Setup:** Initialize test environment and establish device communication
- **Execution:** Perform test operations and capture device responses
- **Verification:** Compare actual results against expected values
- **Evidence Capture:** Record logs, screenshots, and waveform data
- **Cleanup:** Reset device state and release resources

6.2 Acceptance Criteria

General Acceptance Criteria:

- All test cases must execute without errors or exceptions
- All assertions within test cases must pass
- Device responses must match expected values and formats
- CRC validation must pass for all BLE command responses
- Hardware signals (DIO toggles, pacing pulses) must match expected behavior
- Test evidence (plots, command/response logs) must be captured and attached

Test Resolution Criteria:

- **PASS:** All acceptance criteria are met
- **FAIL:** One or more acceptance criteria are not satisfied
- **SKIPPED:** Test prerequisites are not met
- **BROKEN:** Test execution is interrupted due to environment or setup issues

7. Test Execution Summary

Metric	Count	Percentage
Total Test Cases	74	100.0%
Passed	74	100.0%
Failed	0	0.0%
Skipped	0	0.0%
Broken	0	0.0%
Total Execution Time	3493.2 seconds	
Average Test Duration	47.20 seconds	

7.1 Subsystem Breakdown

Subsystem	Total	Passed	Failed	Skipped	Broken
Pacing Engine (PE)	11	11	0	0	0
Main Processor (MP)	51	51	0	0	0
Bluetooth (BT)	12	12	0	0	0

8. Detailed Test Results

This section presents the detailed test execution results for each subsystem. Each test case is documented with its identifier, description, status, and execution time. Test evidence including logs, screenshots, and waveform captures are available in the referenced Allure HTML reports.

8.1 Pacing Engine (PE) Test Results

Test Execution Summary Statistics

Metric	Count	Percentage
Total Test Cases	11	100.0%
Passed	11	100.0%
Failed	0	0.0%
Skipped	0	0.0%
Broken	0	0.0%
Total Execution Time	385.5 seconds	
Average Test Duration	35.04 seconds	

Test Suite Breakdown

Test Suite	Total	Passed	Failed	Skipped	Broken
test_731	1	1	0	0	0
test_733	2	2	0	0	0

test_734	5	5	0	0	0
test_735	2	2	0	0	0
test_736	1	1	0	0	0

Detailed Test Results

The following table provides detailed test information including test descriptions, test methods (steps), acceptance criteria, and test results. For complete raw data, evidence attachments, plots, and step-by-step execution traces, please refer to the accompanying Allure HTML report (PE_Raw_Data_Report.html).

Step #	Test Description	Step Description	Acceptance Criteria	Test Resolution Pass/Fail	HTML Reference
7.3.3 Step 10	Verify that the Pacing Engine transitions to Active State when all POST tests pass successfully and End of Service (EOS) Mode is not active. Test validates requirement PESRS-160: When in Boot State with all POST tests passing and EOS not active, the PE Application shall transition to Active State.	1. Step 1: Power-up and bring BT up via magnet toggle 2. Step 2: Enter Programmer Mode (mode 4) 3. Step 3: Set Active Mode to 1 4. Step 4: Set Parameters to VOO mode (mode 1) 5. Step 5: Wait for POST testing to complete (3 seconds) 6. Step 6: Get Version (target=2) and verify EOS flag not set 7. Step 7: Capture pacing pulses to verify Active State (10 seconds)	When the boot process completed, and POST passed successfully, the system transitioned to Active State, as shown by the Pacemaker pacing. The Get Version command did not have the End of Service Mode flag set (status bits did not include 0x0100)	Pass	See HTML: 7.3.3 - Step# 10: PE Transitions to Active State after POST
7.3.3 Step 20	Verify that after a power cycle, the device can successfully re-enter Programmer Mode and respond to Get Parameters command. This validates that the device properly boots up and can be reconfigured after a complete power cycle.	1. Step 1: Initial power-up and BLE connection 2. Step 2: Enter Programmer Mode (mode 4) 3. Step 3: Get Parameters (baseline before power cycle) 4. Step 4: Set Parameters to VOO mode (mode 1) 5. Step 4.5: Get Parameters BEFORE power cycle for comparison 6. Step 5: Disconnect BLE and power cycle device 7. Step 6: Power device back up 8. Step 7: Re-establish BLE connection after power cycle 9. Step 8: Re-enter Programmer Mode (mode 4) after power cycle 10. Step 9: Get Parameters AFTER power cycle and compare	The parameter values that were set were maintained over a power cycle (Non-Volatile Memory)	Pass	See HTML: 7.3.3 - Step# 20: Power Cycle and Re-enter Programmer Mode

7.3.6 Step 10	Verify that the Pacing Engine sets defined parameters and returns an acknowledgement. This test sends a SetParameters command with specific values (2000ms rate, 1400mV amplitude, 500µs pulse width, Blank Post Vp: 50ms) and verifies: 1. The SetParameters command receives a response indicating acknowledgement 2. The GetParameters command shows the Pacemaker was set to the values in SetParameters 3. Pacing pulses are captured and observed on oscilloscope	1. Step 1: Power-up and bring BT up via magnet toggle 2. Step 2: Enter Programmer Mode (mode 4) 3. Step 3: Set Active Mode to 1 4. Step 4: Send SetParameters command with specified values 5. Step 5: Observe the scope output - Capture pacing pulse on oscilloscope (TP38) 6. Step 6: Send GetParameters command to verify values were set	The Set Parameters command received a response from the Pacemaker indicating that the Set Parameters command was received. The Get Parameters command showed that the Pacemaker was set to the values in the Set Parameters command	Pass	See HTML: 7.3.6 - Step# 10: Set Parameters and Verify Acknowledgement
7.3.4 Step 20	Verify that the Pacing Engine controls the pacing circuitry to deliver pacing at different programmed rates. This test captures pacing pulses at two different rate settings (2000ms and 429ms) and verifies that the measured intervals correspond to the programmed values.	1. Step 1: Power-up and bring BT up via magnet toggle 2. Step 2: Enter Programmer Mode (mode 4) 3. Step 3: Set Active Mode to 1 4. Step 4: Set Parameters to VVI mode with 2000ms pacing rate 5. Step 6: Capture pacing pulses at 2000ms rate (10 seconds) 6. Step 7: Set Parameters to VVI mode with 429ms pacing rate 7. Step 8: Capture pacing pulses at 429ms rate (10 seconds) 8. Step 9: Compare pacing rate measurements	Oscilloscope output showed that the Pacing Pulses were delivered at the set Pacing rate of 2000ms and 429mS	Pass	See HTML: 7.3.4 - Step# 20: Pacing Rate Control at 2000ms and 429ms
7.3.4 Step 10	Verify that the Pacing Engine controls the pacing circuitry to deliver pacing energy at the programmed amplitude. This test captures pacing pulses at two different amplitude settings (3V and 12V) and verifies that the measured amplitudes correspond to the programmed values. Test validates requirement PESRS-200: When in Active State, the PE shall control the pacing circuitry to deliver pacing energy at the programmed amplitude.	1. Step 1: Power-up and bring BT up via magnet toggle 2. Step 2: Enter Programmer Mode (mode 4) 3. Step 3: Set Active Mode to 1 4. Step 4: Set Parameters to VVI mode with 3V amplitude 5. Step 5: Get Parameters to verify 3V amplitude (dpot1=21, dpot2=5) 6. Step 5: Capture pacing pulses at 3V amplitude (10 seconds) 7. Step 6: Set Parameters to VVI mode with 12V amplitude 8. Step 7: Get Parameters to verify 12V amplitude (dpot1=0, dpot2=250) 9. Step 8: Capture pacing pulses at 12V amplitude (10 seconds) 10. Step 9: Compare amplitude measurements and verify	Oscilloscope output showed that the Pacing Pulses were delivered at the set Pacing Amplitude of 3V and 12V	Pass	See HTML: 7.3.4 - Step# 10: Pacing Amplitude Control at 3V and 12V

7.3.4 Step 70	Verify that the Pacing Engine enables EGM blanking prior to delivering pacing energy and leaves EGM blanking enabled for the programmed blanking duration plus charge equalization (50ms). This test verifies EGM blanking duration at two different blanking periods: 150ms (expected 200ms total) and 190ms (expected 240ms total).	1. Step 1: Power-up and bring BT up via magnet toggle 2. Step 2: Enter Programmer Mode (mode 4) 3. Step 3: Set Active Mode to 1 4. Step 4: Set Parameters with Blanking set to 150ms 5. Step 4: Capture pacing pulses and EGM blanking line simultaneously 6. Step 4: Analyze EGM blanking durations 7. Step 4: Verify EGM blanking duration 8. Step 5: Create plot for 150ms blanking period 9. Step 6: Set Parameters with Blanking set to 190ms 10. Step 6: Capture pacing pulses and EGM blanking line simultaneously 11. Step 6: Analyze EGM blanking durations 12. Step 6: Verify EGM blanking duration 13. Step 7: Create plot for 190ms blanking period 14. Step 8: Summary and comparison	Prior to pacing pulse delivery, EGM blanking was enabled. EGM blanking continued throughout pulse delivery for Blanking time (Programmed Pulse Width + Charge equalization 50mS + Programmed Blanking). All EGM blanking was correct (when set to 150mS and when set to 190mS)	Pass	See HTML: 7.3.4 - Step# 70: EGM Blanking Duration Verification at 150ms and 190ms
7.3.1 Step 10	Verify that the Pacing Engine Watchdog line toggles when pacing pulses are delivered or when inhibit signals are received in VVI mode (mode 3).	1. Step 1: Power-up and bring BT up via magnet toggle 2. Step 2: Enter Programmer Mode 3. Step 3: Set Active Mode to 1 4. Step 4: Set Device Parameters to VVI mode (mode 3) 5. Step 5: Monitor Pacing Engine Watchdog baseline (3 seconds) 6. Step 6: Send intrinsic signal and continue watchdog monitoring (2 seconds) 7. Step 7: Analyze watchdog transitions 8. Step 8: Generate watchdog plot and save evidence 9. Step 9: Stop intrinsic signal and cleanup	When the Pacing Engine delivered a pacing pulse or received an Inhibit signal, it toggled the Pacing Engine watchdog line	Pass	See HTML: 7.3.1 - Step# 10: Watchdog Toggle in VVI Mode

7.3.4 Step 40	Verify that when a new set of valid parameters is programmed in the Active State, the new parameters will take effect no later than immediately after the next pacing pulse is delivered or immediately after an inhibit notification is received (whichever occurs first). This test sets initial parameters (VVI, 429ms, 2V), captures pulses, then changes to new parameters (VVIR, 500ms, 3V) and verifies the transition occurs after the next pulse.	1. Step 1: Power-up and bring BT up via magnet toggle 2. Step 2: Enter Programmer Mode (mode 4) 3. Step 3: Set Active Mode to 1 4. Step 4: Set Parameters - VVI mode, 429ms rate, 2V amplitude 5. Step 5: Capture initial pulses (3 seconds at 429ms rate) 6. Step 6: Send new parameters - VVIR mode, 500ms rate, 3V amplitude 7. Step 7: Capture pulses after parameter change (10 seconds) 8. Step 8: Verify parameter changes took effect	When in the Active State and a new set of valid parameters was programmed, the pacing pulse delivery showed that parameter changes occurred after the next pulse delivery that occurred after the command was sent. For the Set Parameter command listed in the Test Procedures column, the first pacing pulse after setting the parameters matched the initial parameters and the next pacing pulses matched the new parameters. First pulse after 2nd Set Parameters command sent: Rate (from start of pulse to start of next pulse):500msec, Amplitude:3V	Pass	See HTML: 7.3.4 - Step# 40: Parameter Changes Take Effect After Next Pulse
7.3.5 Step 20	Verify that the Pacing Engine controls the pacing circuitry to deliver emergency pacing energy with defined parameters upon entry of the Emergency State through the programmer application regardless of whether MRI mode is enabled or disabled. This test verifies emergency mode entry with MRI mode enabled, normal pacing with MRI disabled, and emergency mode entry again with MRI disabled.	1. Step 1: Power-up and bring BT up via magnet toggle 2. Step 2: Enter Programmer Mode (mode 4) 3. Step 3: Enable MRI mode and send Emergency command 4. Step 3.5: Get Parameters to verify emergency amplitude with MRI (dpot1=255, dpot2=85) 5. Step 4: Verify emergency state using GetDeviceStatus (with MRI mode) 6. Step 5: Capture pacing pulses to verify emergency parameters (with MRI mode) 7. Step 6: Disable MRI mode and set normal parameters 8. Step 7: Verify emergency state is cleared using GetDeviceStatus 9. Step 8: Verify normal pacing parameters with GetParameters 10. Step 9: Capture pacing pulses to verify normal pacing parameters 11. Step 10: Send Emergency command again (with MRI mode disabled) 12. Step 10.5: Get Parameters to verify emergency amplitude without MRI (dpot1=255, dpot2=85) 13. Step 11: Verify emergency state using GetDeviceStatus (without MRI mode) 14. Step 12: Capture pacing pulses to verify emergency parameters (without MRI mode)	The oscilloscope output matched the values that were programmed as an emergency parameters regardless of whether MRI mode is enabled or disabled. Pacing Rate: 857 ms, Amplitude: 5000, Pulse Width: 1 ms, Rate Hysteresis: N/A, Blank Post Vp: 150 ms, Polarity: 0. The Set Parameters command received a response from the Pacemaker indicating that the Set Parameters command was received. The Get Parameters command showed that the Pacemaker was set to the values in the Set Parameters command	Pass	See HTML: 7.3.5 - Step# 20: Emergency Pacing Energy with Defined Parameters (MRI Mode)

7.3.4 Step 60	Verify that the Pacing Engine controls the pacing circuitry to deliver pacing pulses at the programmed pulse width. This test captures pacing pulses at two different pulse width settings (100µs and 1500µs) and verifies that the measured pulse widths correspond to the programmed values.	1. Step 1: Power-up and bring BT up via magnet toggle 2. Step 2: Enter Programmer Mode (mode 4) 3. Step 3: Set Active Mode to 1 4. Step 4: Set Parameters with 100µs pulse width 5. Step 5: Capture pacing pulse with 100µs width (10 seconds) 6. Step 6: Set Parameters with 1500µs pulse width 7. Step 7: Capture pacing pulse with 1500µs width (10 seconds) 8. Step 8: Compare pulse width measurements	The Pacing energy delivery is done at the correct Pulse Width (matching what was set). The scope output displayed the pacing pulse width of 100uS and 1500uS respectively	Pass	See HTML: 7.3.4 - Step# 60: Pacing Pulse Width Control at 100µs and 1500µs
7.3.5 Step 10	Verify that the Pacing Engine controls the pacing circuitry to deliver emergency pacing energy with defined parameters. This test verifies emergency mode entry/exit via SetParameters, emergency pacing parameters (70 bpm, 5V, 1ms width), and magnet-triggered emergency mode.	1. Step 1: Power-up and bring BT up via magnet toggle 2. Step 2: Enter Programmer Mode (mode 4) 3. Step 3: Send SetParameters command with specified values 4. Step 4: Get Parameters to verify values were set 5. Step 5: Enter Emergency mode by setting emergency parameters 6. Step 5.5: Get Parameters to verify emergency amplitude (dpot1=255, dpot2=85) 7. Step 6: Verify emergency state using GetDeviceStatus 8. Step 7: Capture pacing pulses to verify emergency pacing parameters 9. Step 8: Exit emergency by setting initial (non-emergency) parameters 10. Step 9: Verify emergency state is cleared using GetDeviceStatus 11. Step 10: Place magnet to trigger reed switch (set_di0 True) 12. Step 11: Verify emergency state with magnet using GetDeviceStatus 13. Step 12: Verify emergency parameters with magnet active	The oscilloscope output matched the values that are listed below: Pacing Rate: 857 ms, Amplitude: 5000, Pulse Width: 1 ms, Rate Hysteresis: N/A, Blank Post Vp: 150 ms, Polarity: 0	Pass	See HTML: 7.3.5 - Step# 10: Emergency Pacing Energy with Defined Parameters

Note: Complete raw data, test descriptions, test methods (steps), acceptance criteria, evidence attachments, plots, and step-by-step execution traces for all tests are available in the accompanying Allure HTML report:
./allure-final-report/PE_Raw_Data_Report.html

8.2 Main Processor (MP) Test Results

Test Execution Summary Statistics

Metric	Count	Percentage
Total Test Cases	51	100.0%
Passed	51	100.0%
Failed	0	0.0%
Skipped	0	0.0%
Broken	0	0.0%
Total Execution Time	2896.6 seconds	
Average Test Duration	56.80 seconds	

Test Suite Breakdown

Test Suite	Total	Passed	Failed	Skipped	Broken
test_741	6	6	0	0	0
test_742	7	7	0	0	0
test_743	1	1	0	0	0
test_744	4	4	0	0	0
test_745	2	2	0	0	0
test_746	11	11	0	0	0
test_747	3	3	0	0	0
test_748	2	2	0	0	0
test_749	12	12	0	0	0
test_74_10	1	1	0	0	0
test_74_12	2	2	0	0	0

Detailed Test Results

The following table provides detailed test information including test descriptions, test methods (steps), acceptance criteria, and test results. For complete raw data, evidence attachments, plots, and step-by-step execution traces, please refer to the accompanying Allure HTML report (MP_Raw_data_Report.html).

Step #	Test Description	Step Description	Acceptance Criteria	Test Resolution Pass/Fail	HTML Reference
7.4.1 Step 10	Induce a 10s timeout while magnet is present and verify the device enters Emergency Mode via status word.	1. Step 1 – Power up and activate BLE via magnet toggle 2. Step 2 – Connect over BLE 3. Step 3 – Enter Programmer Mode 4. Step 4 – Clear errors 5. Step 5 – Get baseline status 6. Step 6 – Hold magnet ON for 10s to trigger emergency 7. Step 7 – Verify Emergency Mode bit set	The Get Device Status command response showed the pacemaker to be in Emergency State (0x0001)	Pass	See HTML: 7.4.1 - Step #10: Emergency Mode Entry via Timeout
7.4.1 Step 20	Enter Storage mode, observe BT line transitions to LOW after inactivity; verify no pacing, then recover to Active with parameters.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode and request Storage Mode 3. Step 3 – Remove magnet and poll BT line for LOW transition 4. Step 4 – Verify no pacing in Standby mode 5. Step 5 – Cycle magnet and verify no pacing in Storage mode 6. Recovery – Restore device to normal operation	When in Standby state (as shown by BT power high and PE power low and no pacing pulses) and 30 seconds passed from when Bluetooth was enabled (magnet removed, BT Power high) the pacemaker dropped back to Storage State as shown by PE power low and BT power Low	Pass	See HTML: 7.4.1 - Step #20: Storage Mode Entry and No Pacing Verification
7.4.1 Step 30	Enter Programmer Mode, wait for BT to idle LOW, retrigger via magnet, capture ~60 bpm pacing for 10s.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Wait for BT line to go LOW (30s monitoring) 4. Step 4 – Re-trigger magnet and verify BT goes HIGH 5. Step 5 – Capture pacing pulses to illustrate Active Mode	When the BT power was enabled and no Bluetooth communication was received from the Programmer Application for 30 seconds, the Pacemaker remained in Active State (as shown by Pacing Pulses continuing at the same rate)	Pass	See HTML: 7.4.1 - Step #30: BT Idle Behavior and Pacing Rate Verification
7.4.1 Step 40	Program Emergency Mode (~70 bpm), wait for BT to idle LOW, retrigger via magnet, confirm Emergency persists and pacing rate is maintained.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Set Emergency Mode parameters 4. Step 4 – Verify Emergency Mode bit set 5. Step 5 – Wait for BT to go LOW and reconnect 6. Step 6 – Verify Emergency Mode persists and capture pacing 7. Recovery – Restore device to normal operation	When the BT power was enabled and no Bluetooth communication was received from the Programmer Application for 30 seconds, the Pacemaker remained in Emergency State (as shown by Get Device Status). The Pacemaker stayed in Emergency State when no communication was received from the Programmer Application for 30 seconds of enabling Bluetooth power	Pass	See HTML: 7.4.1 - Step #40: Emergency Mode Pacing and Persistence

7.4.1 Step 50	Corrupt the CRC of a valid frame and verify device responds with NACK (opcode 0x01).	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Corrupt CRC and verify NACK response	MP returned a nack for the Get Parameters command	Pass	See HTML: 7.4.1 - Step #50: NACK Response on Invalid CRC
7.4.1 Step 70	Send swTestServices (target=1, test=1, data=0) three times; verify BT line goes LOW (device disconnects).	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Record initial BT state before commands 4. Step 4 – Send swTestServices commands 5. Step 5 – Verify BT line goes LOW after commands	When the Pacemaker Application received three consecutive bad command packets it shut down the Bluetooth power.	Pass	See HTML: 7.4.1 - Step #70: BT Disable via swTestServices
7.4.2 Step 10	Powering up the device, connecting via BLE, entering Storage State and verifying BT line behaviour	1. Step 1 – Power up and activate BLE via magnet 2. Step 2 – Connect over BLE 3. Step 3 – Enter Programmer Mode 4. Step 4 – Enter Storage Mode 5. Step 5 – Poll BT line for 30s 6. Step 6 – Verify BT is HIGH after magnet is restored 7. Step 7 – Plot BT behavior as proof 8. Recovery – Restore device to normal operation	Activating the Reed Switch when in Storage caused BT power to be enabled	Pass	See HTML: 7.4.2 - Step #10: Storage Mode Verification
7.4.2 Step 20	Powering up the device, connecting via BLE, polling on BT to go low and then triggering Reed switch to verify BT line behaviour	1. Step 1 – Power up and connect via BLE 2. Step 2 – Poll BT line for 30s (expect BT to go low) 3. Step 3 – Toggle magnet to bring BT back HIGH 4. Step 4 – Plot BT behavior as proof	When the Reed Switch was activated in Standby State, BT power was enabled. When no Reed Switch activation occurred, and no command was received for 30 seconds BT power was disabled	Pass	See HTML: 7.4.2 - Step #20: Standby Mode Verification
7.4.2 Step 30	Powering up the device, connecting via BLE, polling on BT to go low, trigger the Reed switch and send a Get Parameters command	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Poll BT line for 30s (expect BT to go low) 4. Step 4 – Toggle magnet to bring BT back HIGH 5. Step 5 – Send GetParameters command 6. Step 6 – Plot BT behavior as proof	When the Reed Switch was activated in Active State, BT power was enabled. When no Reed Switch activation occurred, and no command was received for 30 seconds BT power was disabled. Time to disable after command was received: 28sec	Pass	See HTML: 7.4.2 - Step #30: Active Mode / GetParameters Verification

7.4.2 Step 40	Power up the device, connect via BLE, set Emergency Mode, Poll on BT to go low, trigger the Reed switch and verify Emergency mode is still active	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Set Emergency Mode parameters 4. Step 4 – Verify Emergency Mode via GetDeviceStatus 5. Step 5 – Poll BT line for 30s (expect BT to go low) 6. Step 6 – Trigger magnet to bring BT back and reconnect 7. Step 7 – Enter Programmer Mode again 8. Step 8 – Verify Emergency Mode still active 9. Step 9 – Capture pacing pulses in Emergency Mode 10. Recovery – Restore device to normal operation	When the Reed Switch was activated in Emergency State, BT power was enabled. When no Reed Switch activation occurred, and no command was received for 30 seconds BT power was disabled	Pass	See HTML: 7.4.2 - Step #40: Emergency Mode Verification
7.4.2 Step 50	Power up the device, connect via BLE, verify that activating the Reed switch while in Active State forces device into emergency pacing. Test is repeated across all modes.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3.1 – Emergency Mode Validation (VOO) 4. Step 3.2 – Emergency Mode Validation (OVO) 5. Step 3.3 – Emergency Mode Validation (VVI) 6. Step 3.4 – Emergency Mode Validation (VVIR)	When the reed switch is activated, the emergency settings for Pacing Mode and Blank Post VP are used (See 8.5). Expected values: Pacing mode: VOO. The responses from the Get Stored Data with target 3 showed only the pacing count was increasing, the sensing (inhibits) count was not increasing which shows VOO and not OVO	Pass	See HTML: 7.4.2 - Step #50: Emergency Mode Verification Across All Modes
7.4.2 Step 60	Ensure that when the Calyan device is in Active state and the reed switch is activated, the pacing circuitry delivers emergency pacing with defined parameters. The test is repeated for modes VVI, VOO, OVO, and VVIR.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3.1 – Emergency Status Verification (VVI) 4. Step 3.2 – Emergency Status Verification (VOO) 5. Step 3.3 – Emergency Status Verification (OVO) 6. Step 3.4 – Emergency Status Verification (VVIR) 7. Recovery – Restore device to normal operation	When the reed switch was deactivated the Pacing Mode, Blank Post Vp and Refractory were set to the values that were programmed in the Set Parameters command. The Get Device Status command showed that the Pacing Engine was in Emergency State (includes 0x0001)	Pass	See HTML: 7.4.2 - Step #60: Emergency Mode Entry via Reed Switch
7.4.2 Step 70	Power up the device, connect via BLE, put the device into Storage mode, trigger a MP reset and verify the mode is retained. Following that, reset the MP and place and leave the magnet to trigger Emergency mode.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Enter Storage Mode 4. Step 4 – Reset MP and verify Storage Mode retained 5. Step 5 – Reconnect and restore to Active Mode 6. Step 6 – Reset MP with magnet ON 7. Step 7 – Verify Emergency Mode after magnet-on reset	After each power cycle the Pacemaker transitioned back into the state it was in except when a magnet was induced it transitioned to an Emergency State	Pass	See HTML: 7.4.2 - Step #70: State Retention After MP Reset

7.4.3 Step 20	Ensure that the Calyan device maintains pacing parameters after a watchdog reset. The test triggers a watchdog reset using SwTestServices(2, 0, 0), waits for the device to recover, and verifies that the MainProcessorResetReason shows Watchdog (5) and that pacing parameters are retained.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Configure pacing parameters 4. Step 4 – Read and save parameters before reset 5. Step 5 – Get baseline device status 6. Step 6 – Trigger Watchdog Reset 7. Step 7 – Wait for watchdog reset to complete 8. Step 8 – Reconnect after watchdog reset 9. Step 9 – Verify watchdog reset in device status 10. Step 10 – Verify pacing parameters retained 11. Cleanup: Disconnect and power off	After the Watchdog Reset, the pacing parameters were maintained at the setting in the Set Parameters command.	Pass	See HTML: 7.4.3 - Step #20: Watchdog Reset Parameter Retention Verification
7.4.4 Step 10	Ensure the device stops pacing and disables Bluetooth when entering Storage Mode, and confirm normal operation can be restored afterward.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Enter Storage Mode 4. Step 4 – Verify pacing engine power (DIO14) transitions LOW 5. Step 5 – Verify pacing is disabled in Storage Mode 6. Step 6 – Verify Bluetooth line (TP19) transitions LOW 7. Recovery – Restore device to normal operation	When the Storage Mode command was pressed, the Pacemaker transitioned to Storage State (PE Low, BT Low). The BT Power line went low (disabled) when the Pacemaker transitioned to Storage State	Pass	See HTML: 7.4.4 - Step #10: Verify BT and pacing disabled in Storage Mode
7.4.4 Step 20	Ensure that when the device enters Storage Mode, the pacing engine and accelerometer lines go low, indicating deactivation, and verify no pacing pulses are output.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Set parameters for VVIR mode 4. Step 4 – Confirm VVIR entry via DIO2 5. Step 5 – Capture accelerometer and pacing engine power transition 6. Step 6 – Verify pacing engine deactivated 7. Recovery – Restore device to normal operation	When the Pacemaker Application was in Storage State, the accelerometer was deactivated (accelerometer line was low). When the Pacemaker Application was in Storage State, the Pacing Engine was deactivated (Pacing Engine power line low)	Pass	See HTML: 7.4.4 - Step #20: Verify accelerometer and pacing engine disable in Storage Mode

7.4.4 Step 30	Ensure that when the device enters Storage Mode with the magnet applied, the pacing engine and BT power deactivate, and that toggling the magnet restores BT power without resuming pacing.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Verify pacing engine power (DIO14) is HIGH 4. Step 4 – Enter Storage Mode with magnet ON 5. Step 5 – Capture pacing waveform (expect none) 6. Step 6 – Poll BT Power line for 30s (expect low) 7. Step 7 – Toggle magnet to exit standby 8. Step 8 – Verify pacing remains disabled 9. Step 9 – Verify pacing engine power (DIO14) is LOW 10. Recovery – Restore device to normal operation	When the reed switch was activated, the Pacemaker Application transitioned into Standby State as shown by: BT Power high, Pacing Engine Power low, No pacing pulse delivery	Pass	See HTML: 7.4.4 - Step #30: Verify pacemaker enters Standby when magnet is ON in Storage Mode
7.4.4 Step 40	Ensure that when the device is in Storage Mode there are no pacing pulses, and that sending Set Active Mode and Set Parameters restores pacing output.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Enter Storage Mode and verify no pacing 4. Step 4 – Poll BT and reconnect BLE 5. Step 5 – Recovery sequence to enable pacing 6. Step 5.1 – Start TP38 monitoring before SetActiveMode 7. Step 5.2 – Send SetActiveMode command 8. Step 5.3 – Verify single pulse after SetActiveMode 9. Step 6 – Set Parameters to restore continuous pacing 10. Cleanup – Disconnect and power off	When the Pacemaker Application was in Storage State, there were no pacing pulses. When the Pacemaker application was sent a Set Active Mode, the device paced once. When a Set Parameters was sent, the Pacemaker Application reflected these changes	Pass	See HTML: 7.4.4 - Step #40: Verify pacemaker transitions from Storage to Active Mode correctly
7.4.5 Step 10	Ensure that when the Calyan device is in Standby (Storage) state and the reed switch is activated, the Bluetooth power line (TP61) transitions from LOW to HIGH, indicating activation.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Enter Storage Mode with magnet ON 4. Step 4 – Poll BT line until it goes LOW 5. Step 5 – Record BT line before and after magnet toggle 6. Recovery – Restore device to normal operation	When the Pacemaker transitioned into Standby State, Bluetooth power was activated	Pass	See HTML: 7.4.5 - Step #10: Verify BT Power activates when reed switch is triggered in Standby

7.4.5 Step 20	Ensure that when the device is in Standby and Bluetooth power is disabled, sending the Disconnect command transitions the pacemaker into Storage Mode (no pacing, BT low, PE low).	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Enter Storage Mode with magnet ON 4. Step 4 – Poll BT line until LOW 5. Step 5 – Toggle magnet to exit Standby and reconnect 6. Step 6 – Enter Programmer Mode again 7. Step 7 – Send Disconnect command 8. Step 8 – Verify Storage Mode: Plot DIO14 (Pacing Engine Power) 9. Step 9 – Verify no pacing output 10. Recovery – Restore device to normal operation	Sending the Disconnect command put the Pacemaker into Storage State (no pulsing, BT low, PE low)	Pass	See HTML: 7.4.5 - Step #20: Verify Disconnect command places device into Storage Mode
7.4.6 Step 30	Ensure that the Calyan device correctly programs and acknowledges pacing parameters for both VOO and VVIR modes, and retains these values after power cycle.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Send SetParameters (VOO Mode) 4. Step 4 – Verify parameters via GetParameters (VOO Mode) 5. Step 5 – Send SetParameters (VVIR Mode) 6. Step 6 – Verify VVIR parameters via GetParameters 7. Step 7 – Power cycle device and reconnect 8. Step 8 – Verify GetParameters after power cycle 9. Cleanup: Disconnect and power off	A Set Parameters command was received from the Programmer Application and an acknowledgement (response packet) from the Pacemaker Application. The Get Parameters command showed that the values were the same as the values in the Set parameters command sent to the pacemaker. After the power cycle, the parameters retained their values set before the power cycle (non-volatile memory)	Pass	See HTML: 7.4.6 - Step #30: Parameter Programming and Retention
7.4.6 Step 40	Ensure that SetParameters NACKs invalid pacing rate values (below/above limit) and ACKs valid boundary values. If ACKed, pacing waveform is captured for 20 s to prove acceptance.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Send SetParameters rate=332 ms (Below Min Limit) 4. Step 3 – Send SetParameters rate=3001 ms (Above Max Limit) 5. Step 3 – Send SetParameters rate=333 ms (Valid Lower Boundary) 6. Step 3 – Send SetParameters rate=3000 ms (Valid Upper Boundary) 7. Cleanup: Disconnect and power off	When sending the Pacing Rate 3000 and 333 commands, a Nack was received. The Pacing rate 3000 and 333 commands functioned properly and set the Pacing Rate as observed on the oscilloscope output	Pass	See HTML: 7.4.6 - Step #40: Parameter Boundary Handling

7.4.6 Step 50	Send SetParameters commands with out-of-range and valid pulse widths. Expect NACK for 99 µs and 3001 µs, ACK for 100 µs and 3000 µs. During the entire test, TP65 (Pace Rx) and TP66 (Pace Tx) are monitored continuously.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Send SetParameters: Pulse Width = 99 µs (Below Min Limit) 4. Send SetParameters: Pulse Width = 3001 µs (Above Max Limit) 5. Send SetParameters: Pulse Width = 100 µs (Valid Lower Boundary) 6. Send SetParameters: Pulse Width = 3000 µs (Valid Upper Boundary) 7. Cleanup – Disconnect and Power Off	The Pacing Width 99 and 3001 commands did not actually set any parameters. The Pacing Width 3001 and 99 commands were responded to with a Nack. The Pacing Width 3000 and 100 commands functioned properly and set the Pacing Width	Pass	See HTML: 7.4.6 - Step #50: Pulse Width Boundary Verification
7.4.6 Step 60	Send SetParameters commands with out-of-range and valid Sensitivity values. Expect NACK for 20001 µV and ACK for 0 µV and 20000 µV. During the test, TP65 (Pace Rx) and TP66 (Pace Tx) are continuously monitored.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Send SetParameters: Sensitivity = 20001 µV (Above Max Limit) 4. Send SetParameters: Sensitivity = 0 µV (Valid Lower Boundary) 5. Send SetParameters: Sensitivity = 20000 µV (Valid Upper Boundary) 6. Cleanup – Disconnect and Power Off	The Sensitivity 20001command did not actually set any parameters. The Sensitivity 20001command was responded to with a Nack. The Sensitivity 20000and 0 commands functioned properly and set the Sensitivity	Pass	See HTML: 7.4.6 - Step #60: Sensitivity Boundary Verification
7.4.6 Step 70	Send SetParameters commands with out-of-range and valid Refractory Period values. Expect NACK for 501 ms and ACK for 50 ms and 500 ms. During the test, TP65 (Pace Rx) and TP66 (Pace Tx) are monitored continuously.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Send SetParameters: Refractory = 501 ms (Above Max Limit) 4. Send SetParameters: Refractory = 50 ms (Valid Lower Boundary) 5. Send SetParameters: Refractory = 500 ms (Valid Upper Boundary) 6. Cleanup – Disconnect and Power Off	The Refractory Period 501 command did not actually set any parameters. The Refractory Period 501 command was responded to with a Nack. The Refractory Period 500 and 0 commands functioned properly and set the Refractory Period	Pass	See HTML: 7.4.6 - Step #70: Refractory Period Boundary Verification
7.4.6 Step 80	Send SetParameters commands with out-of-range and valid Rate Hysteresis values. Expect NACK for 1099 ms and ACK for 1100 ms and 1101 ms. During the test, TP65 (Pace Rx) and TP66 (Pace Tx) are continuously monitored to capture transitions.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Send SetParameters: Rate Hysteresis = 1099 ms (Below Min Limit) 4. Send SetParameters: Rate Hysteresis = 1100 ms (Valid Lower Boundary) 5. Send SetParameters: Rate Hysteresis = 1101 ms (Valid Upper Boundary) 6. Cleanup – Disconnect and Power Off	The Hysteresis 1099 command did not actually set any parameters. The Hysteresis 1099 command was responded to with a Nack. The Hysteresis 1100 and 1101 commands functioned properly and set the Rate Hysteresis	Pass	See HTML: 7.4.6 - Step #80: Rate Hysteresis Boundary Verification

7.4.6 Step 90	Set refractory period to 49ms, then test Blank VP values. Expect NACK for blank_vp=50ms (> Refractory) and ACK for blank_vp=48ms (≤ Refractory). Monitor TP65/TP66 (DIO 3/4) to verify toggles occur only when command is accepted.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Set Refractory Period to 49ms 4. Step 4 – Test Blank VP = 50ms (should NACK, no toggle expected) 5. Step 5 – Test Blank VP = 48ms (should ACK, toggle expected) 6. Step 6 – Verify parameters via GetParameters 7. Cleanup – Disconnect and Power Off	The Blank Post Vp 49 command did not actually set any parameters. The Blank Post Vp 49 command was responded to with a Nack. The Blank Post Vp 50 command functioned properly and set the Pacing Blank Post Vp	Pass	See HTML: 7.4.6 - Step #90: Blank Post Vp Boundary Verification
7.4.6 Step 100	Send SetParameters commands with out-of-range and valid Polarity values. Expect NACK for Polarity=2 (invalid) and ACK for Polarity=0 (unipolar) and Polarity=1 (bipolar). When ACKed, capture TP65 (Pace Rx) and TP66 (Pace Tx) transitions for confirmation.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Send SetParameters: Invalid Polarity – Expect NACK 4. Send SetParameters: Valid Polarity (Unipolar) – Expect ACK 5. Send SetParameters: Valid Polarity (Bipolar) – Expect ACK 6. Step 3 – Verify final polarity via GetParameters 7. Cleanup – Disconnect and Power Off	The Polarity 2 command did not actually set any parameters. The Polarity 2 command was responded to with a Nack. The Polarity 0 and 1 command functioned properly and set the Pacing Polarity	Pass	See HTML: 7.4.6 - Step #100: Polarity Boundary Verification
7.4.6 Step 110	Send SetParameters commands with out-of-range and valid pacing mode values. Expect NACK for Mode=6 (invalid), and ACK for Mode=0 and Mode=5 (valid). During valid commands, capture TP65 (Pace Rx) and TP66 (Pace Tx) transitions.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Send SetParameters: Invalid Mode – Expect NACK 4. Send SetParameters: Valid Mode – V00 (Expect ACK) 5. Send SetParameters: Valid Mode – VVIR (Expect ACK) 6. Step 3 – Verify active parameters via GetParameters 7. Cleanup – Disconnect and Power Off	The Pacing Mode 6 command did not actually set any parameters. The Pacing Mode 6 command was responded to with a Nack. The Pacing Mode 0 and 5 commands functioned properly and set the Pacing Mode	Pass	See HTML: 7.4.6 - Step #110: Pacing Mode Boundary Verification
7.4.6 Step 120	Place magnet ON and enter Storage Mode. Verify ActiveMode is disabled. Then remove magnet, send SetActiveMode, confirm GetParameters shows OOO (Mode=0), and finally confirm ActiveMode flag is enabled. Proofs include BLE responses and parsed logs.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Keep Magnet ON and enter Storage Mode 4. Step 4 – GetActiveMode (expect Disabled) 5. Step 5 – Send SetActiveMode command 6. Step 6 – GetParameters (expect Mode = 0 / OOO) 7. Step 7 – GetActiveMode again (expect Enabled) 8. Recovery – Restore device to normal operation	The Set Active Mode command sets the Active Mode flag when in Standby State as shown by the response in the 1st Get Active Mode command. Get Parameters command responded with OOO mode. The Get Active Mode commands returned the state of the Active Mode Flag	Pass	See HTML: 7.4.6 - Step #120: Standby to Active State Transition

7.4.6 Step 160	Verify that a watchdog reset timer is used in Active State. The test triggers a watchdog reset using SwTestServices_WDT(), waits for device recovery, and verifies that the Get Device Status bits indicate there was a Watchdog Reset in the MP.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Get baseline device status 4. Step 4 – Trigger Watchdog Reset 5. Step 5 – Wait for watchdog reset to complete 6. Step 6 – Reconnect after watchdog reset 7. Step 7 – Verify watchdog reset in device status 8. Cleanup: Disconnect and power off	A watchdog reset timer shall be used in Active State (shown by the Watchdog Reset Flag being set). The Get Device Status bits before the watchdog shall not indicate a Watchdog Reset for the MP. After the Watchdog Reset, the Get Device Status bits shall indicate there was a Watchdog Reset in the MP	Pass	See HTML: 7.4.6 - Step #160: Watchdog Reset Fault Flag Verification
7.4.7 Step 30	Simulate intrinsic heartbeat via analog waveform output and confirm pacing inhibition. 1. VVIR mode: pacing should be inhibited during intrinsic signal (DIO6 toggles). 2. VVI mode: same verification of inhibition. Capture DIO6 (Inhibit Line) and TP38 (Pacing Output) as proof.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Set VVIR mode parameters 4. Step 4 – Send Intrinsic Signal (Simulate heartbeat) 5. Step 5 – Capture Inhibit Line (DIO6) in VVIR mode 6. Step 6 – Capture TP38 (Pacing Output) – VVIR mode 7. Step 7 – Change parameters to VVI mode 8. Step 8 – Capture Inhibit Line (DIO6) in VVI mode 9. Step 9 – Capture TP38 (Pacing Output) – VVI mode 10. Step 10 – Stop intrinsic waveform output 11. Cleanup – Disconnect and Power Off	While in Active State and VVIR Pacing Mode and a heartbeat was observed, the Inhibit line toggled. When the intrinsic signal was being sent, the pacemaker did not deliver pacing pulses	Pass	See HTML: 7.4.7 - Step #30: Verify inhibition behavior via DIO6 and TP38
7.4.8 Step 40	This test verifies the device's pacing behavior when in Emergency Mode. Applying the magnet should force the device into Emergency Mode, where pacing pulses are expected on TP38 at approximately 70 bpm, 1 ms width, and amplitude between 2.0–5.5 V.	1. Step 1 – Power up and connect via BLE 2. Step 1 – Enter Programmer Mode 3. Step 2 – Apply magnet to enter Emergency Mode 4. Step 3 – Capture TP38 waveform for emergency pacing verification 5. Step 4 – Analyze pacing parameters from raw signal 6. Step 5 – Attach summary and proof plot 7. Cleanup – Disconnect and Power Off	The captured pacing data is within the accuracy restrictions listed in the table in the Req. column: Pacing Amplitude +/- 5%, Pulse Duration +/- 5% or 20 uS whichever is greater, Pulse Rate +/- 5%, Sensitivity +/- 50%, Refractory Period +/- 10mS, Hysteresis Period +/- 5 %	Pass	See HTML: 7.4.8 - Step #40: Emergency Pacing Parameters Verification

7.4.8 Step 50	This test verifies that the device disables Bluetooth power after a Disconnect command. TP61 (BT Power line) should transition from HIGH to LOW once the Disconnect command is issued. The captured BT trace will be plotted to confirm the falling edge transition.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Begin monitoring BT Power (TP61) 4. Step 4 – Send Disconnect command and monitor transition 5. Step 5 – Generate and attach BT Power transition plot 6. Cleanup – Disconnect and Power Off	When the disconnect command is sent, the Bluetooth Power Line goes low	Pass	See HTML: 7.4.8 - Step #50: Verify BT Power (TP61) goes LOW after Disconnect
7.4.8 Step 60	This test verifies that the device correctly enters Emergency Mode when a magnet is placed over the reed switch, and exits Emergency Mode when the magnet is removed. Status bits are verified before and after magnet placement using the GetDeviceStatus command.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Set Active Mode and verify 4. Step 4 – Place a magnet to trigger Emergency Mode 5. Step 5 – Verify Emergency Mode via GetDeviceStatus 6. Step 6 – Remove magnet 7. Step 7 – Verify device exits Emergency Mode and stayed in Active Mode 8. Cleanup – Disconnect and Power Off	The system indicated that it was in emergency state (0x0001). The Pacing Engine transitioned to Active state (not include 0x0001)	Pass	See HTML: 7.4.8 - Step #60: Emergency Mode Activation and Deactivation
7.4.8 Step 60	This test ensures the pacemaker disables both the pacing engine and Bluetooth power when entering Storage Mode. The Active Mode flag should clear (Active=0), and pacing output on TP38 should cease once the magnet is removed.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Get Active Mode (Expect Active = 1) 4. Step 4 – Simulate placing magnet (Reed Switch ON) 5. Step 5 – Get Device Status (Expect Emergency = 1) 6. Step 5.1 – Capture Emergency Pacing Plot 7. Step 6 – Send EnterStorageMode command 8. Step 7 – Remove magnet (Reed Switch OFF) and wait 5s 9. Step 7.1 – Capture Pacing Engine Power (DIO14) after Storage Mode 10. Step 8 – Get Active Mode again (Expect Disabled = 0) 11. Step 9 – Capture pacing output (TP38 – Expect No Pulses) 12. Recovery – Restore device to normal operation	When in Active State, the GetActive Mode command returned 0001 in the data. Programmer application displayed emergency mode is active which indicates that the magnet is present. Programmer displayed that the test unit is in OOO mode. When in Standby State, the Get-Active Mode command returned 0000 in the data. The pacemaker transitioned to Standby State and disabled power to the Pacing Engine	Pass	See HTML: 7.4.8 - Step #60: Verify Pacing and BT Power disable when entering Storage Mode

7.4.9 Step 10	This test verifies that the EGRAM Streaming Packets Sent parameter (ID 513) is updated after a Start/Stop Streaming cycle and that its value is readable and writable via GetDeviceParameter and SetDeviceParameter commands.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Start and Stop Streaming (target 0) 4. Step 4 – Get EGRAM Streaming Packets (Param 513) 5. Step 5 – Repeat Streaming Cycle (target 0) 6. Step 6 – Get EGRAM Streaming Packets After Second Cycle (Param 513) 7. Step 7 – Write Device Parameter 513 (Value = 1) 8. Step 8 – Read EGRAM Streaming Packets After Write 9. Cleanup – Disconnect and Power Off	The EGRAM Streaming Packets Sent count changed (Initial read: 12, Final read: 24, Stream Count diff: 12). The EGRAM Streaming Packets Set value was readable and writeable(set it to 1)	Pass	See HTML: 7.4.9 - Step #10: Verify EGRAM Streaming Packet Count Read/Write Functionality
7.4.9 Step 20	This test verifies that the Sensing Filter Calculations Run Count (Param ID 512) increases while intrinsic signals are being sent, and that the parameter is both readable and writable through the programmer interface.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Set device parameters to VVIR pacing mode 4. Step 4 – Start sending intrinsic signal to simulate heartbeats 5. Step 5 – Read initial Sensing Filter Calculations Run Count (Param 512) 6. Step 6 – Wait and read updated Sensing Filter Calculations Run Count 7. Step 7 – Write new value to Param 512 (Set to 1) 8. Step 8 – Read back Param 512 to verify write 9. Cleanup – Disconnect and Power Off	There was a change in Sensing Filter Calculations Run as expected (Count increased from 406 -> 817). The Sensing Filter Calculations count was writeable (set it to 1 and it started counting again)	Pass	See HTML: 7.4.9 - Step #20: Verify Sensing Filter Calculations Run Count Functionality
7.4.9 Step 30	This test verifies that the Bluetooth Messages Sent counter (Param ID 516) increments correctly when BLE communication commands are exchanged and that the parameter is both readable and writable.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Get initial Bluetooth Messages Sent (Param 516) 4. Step 4 – Send GetVersion command (target = 1) 5. Step 5 – Send SetParameters command (mode changed to 1) 6. Step 6 – Send GetParameters command 7. Step 7 – Send GetTime command 8. Step 8 – Read updated Bluetooth Messages Sent (Param 516) 9. Step 9 – Write new value to Device Parameter 516 (set to 1) 10. Step 10 – Read back Param 516 to confirm write 11. Cleanup – Disconnect and Power Off	The change in Bluetooth Messages Sent was 1 after sending the Get FW Version command (Initial read: 142, Final read: 143). The Bluetooth Messages sent was readable and writeable (Final Bluetooth Message Count After Write = 2 is the reset command)	Pass	See HTML: 7.4.9 - Step #30: Verify Bluetooth Message Counter Functionality

7.4.9 Step 40	This test verifies that the Pacing Engine Messages Sent parameter (Param ID 517) increments when SetParameters commands are sent and that the parameter is both readable and writable.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Get initial Pacing Engine Messages Sent (Param 517) 4. Step 4 – Send SetParameters command (change pacing rate) 5. Step 5 – Get updated Pacing Engine Messages Sent (Param 517) 6. Step 6 – Write Device Parameter 517 = 0 7. Step 7 – Confirm write by reading back Param 517 8. Cleanup – Disconnect and Power Off	The change in Pacing Engine Messages Sent increased after sending Set Parameters commands (Counters increased by 1 (from 22 -> 23)). The Pacing Engine Messages sent was readable and writeable (Final Message Count (after write) = 0)	Pass	See HTML: 7.4.9 - Step #40: Verify Pacing Engine Message Counter Functionality
7.4.9 Step 50	This test verifies that the Pacing Pulses Delivered counter (Param ID 518) correctly increments while the device is pacing, and that the counter is also writable/resettable. The test reads the counter, waits for natural pacing activity to occur, verifies the count increased, resets the counter to 0, and finally confirms that the counter is near zero (<5 pulses).	1. Step 1 – Power up device and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Read initial Pacing Pulses Delivered (Param 518) 4. Step 4 – Wait 10 seconds for pacing pulses to accumulate 5. Step 5 – Read updated Pacing Pulses Delivered (Param 518) 6. Step 6 – Reset the Pacing Pulses Delivered Count (Set Param 518 = 0) 7. Step 7 – Read Param 518 after reset 8. Cleanup – Disconnect and Power Off	The change in Pacing Pulses Delivered Count increased (Before=1051, After=1692). The Pacing Pulses Delivered count sent was readable and writeable (reset to 0, started counting again)• After setting the counter to 0, reading back Param 518 should return a value <5.	Pass	See HTML: 7.4.9 - Step #50: Verify Pacing Pulses Delivered Counter
7.4.9 Step 60	This test validates that the INHIBITS DELIVERED TO THE PACING ENGINE counter (Param ID 520) increments while pacing/inhibition occurs, and that the counter is writable by setting it to zero. The device must be in pacing mode 3 or 4, and intrinsic heartbeats must be delivered.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Start intrinsic signal (simulate heartbeats) 4. Step 4 – Configure device to VVI mode (Mode 3) 5. Step 5 – Read initial INHIBITS Delivered count (Param 520) 6. Step 6 – Wait 10 seconds for inhibits to accumulate 7. Step 7 – Read updated INHIBITS Delivered count (Param 520) 8. Step 8 – Reset INHIBITS Delivered count (Set Param 520 = 0) 9. Step 9 – Verify reset by reading Param 520 again 10. Cleanup – Disconnect and Power Off	The change in INHIBITS DELIVERED TO THE PACING ENGINE increased (Initial read: 160, Final read: 171). The INHIBITS DELIVERED TO THE PACING ENGINE count sent was readable and writeable (reset to 0, started counting again)	Pass	See HTML: 7.4.9 - Step #60: Verify INHIBITS Delivered to the Pacing Engine

7.4.9 Step 70	Ensures the pacemaker provides accurate, readable, and writable Accelerometer Reads count (Param ID 521). The count must increase over time while in pacing mode 4, and must reset when writing 0 to the parameter.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Configure VVIR Mode (Mode 4) 4. Step 4 – Read initial Accelerometer Reads Count (Param 521) 5. Step 5 – Wait 60 seconds (BT will shut down) 6. Step 6 – Reconnect BLE after BT shutdown 7. Step 7 – Read updated Accelerometer Reads Count 8. Step 8 – Reset Accelerometer Reads Count (Set Param 521 = 0) 9. Step 9 – Read Accelerometer Reads Count after reset 10. Cleanup – Disconnect and Power Off	The change in ACCELEROMETER READS increased (Initial read: 2016, Final read: 2556). The ACCELEROMETER READS count sent was readable and writeable (reset to 0, started counting again)	Pass	See HTML: 7.4.9 - Step #70: Accelerometer Reads Counter Verification
7.4.9 Step 100	Ensures the pacemaker provides accurate, readable, and writable MAIN PROCESSOR RESET count (Param ID 524). The counter must increase by 1 after a power cycle and must reset correctly when writing 0 to the parameter.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Read initial MAIN PROCESSOR RESET count 4. Step 4 – Power cycle device 5. Step 5 – Reconnect BLE and re-enter Programmer Mode 6. Step 6 – Read updated MAIN PROCESSOR RESET count 7. Step 7 – Reset MAIN PROCESSOR RESET count to 0 8. Step 8 – Confirm MAIN PROCESSOR RESET counter is now 0 9. Cleanup – Power off and close device	The change in MAIN PROCESSOR RESET increased (Initial read:45, Final read: 46). The MAIN PROCESSOR RESET count sent was readable and writeable (reset to 0)	Pass	See HTML: 7.4.9 - Step #100: Main Processor Reset Counter Verification
7.4.9 Step 110	Ensures the pacemaker provides a readable and writable count for magnet pickups (Param ID 525). The counter should increment when the reed switch is triggered by a magnet and should reset when writing 0 to the parameter.	1. Step 1 – Power up device and prepare BLE connection 2. Step 2 – Enter Programmer Mode 3. Step 3 – Get initial MAGNET PICKUPS count (Param 525) 4. Step 4 – Trigger reed switch using simulated magnet (set_di0 ON → OFF) 5. Step 5 – Get updated MAGNET PICKUPS count 6. Step 6 – Reset MAGNET PICKUPS counter (Set Param 525 = 0) 7. Step 7 – Read MAGNET PICKUPS count after reset 8. Cleanup – Disconnect and Power Off	The change in MAGNET PICKUPS increased (Initial read: 44, Final : 45). The MAGNET PICKUPS count sent was readable and writeable (reset to 0)	Pass	See HTML: 7.4.9 - Step #110: Magnet Pickups Counter Verification

7.4.9 Step 120	Ensures the pacemaker maintains an accurate, readable, and writable counter for SECONDS OF BLUETOOTH POWER (Param ID 527). The value must increase while BLE is active and must reset when writing zero to the parameter.	1. Step 1 – Power up and prepare BLE connection 2. Step 2 – Enter Programmer Mode 3. Step 3 – Read initial SECONDS OF BLUETOOTH POWER (Param 527) 4. Step 4 – Wait 5 seconds 5. Step 5 – Read updated SECONDS OF BLUETOOTH POWER (Param 527) 6. Step 6 – Reset BT Power Seconds (Write 0 to Param 527) 7. Step 7 – Read Param 527 to confirm reset 8. Cleanup – Disconnect and Power Off	The change in SECONDS OF BLUETOOTH POWER increased (Initial read: 1591, Final read: 1597). The SECONDS OF BLUETOOTH POWER count sent was readable and writeable (reset to 0)	Pass	See HTML: 7.4.9 - Step #120: Seconds of Bluetooth Power Counter Verification
7.4.9 Step 160	Ensures the pacemaker provides readable and writable fields for tracking manufacturing time and battery capacity. The test reads initial values, writes new values, verifies they are updated, and restores the original values.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Read DEVICE MANUFACTURING TIME (Param 256) 4. Step 4 – Read BATTERY CAPACITY (Param 257) 5. Step 5 – Write 0 to DEVICE MANUFACTURING TIME 6. Step 6 – Read Param 256 after writing 0 7. Step 7 – Write new value to BATTERY CAPACITY 8. Step 8 – Confirm updated BATTERY CAPACITY 9. Step 9 – Restore original values to Param 256 and 257 10. Cleanup – Disconnect and Power Off	The DEVICE MANUFACTURING TIME was read out (Initial read: E8-03-00-00). The DEVICE MANUFACTURING TIME sent was readable and writeable (reset to 0)	Pass	See HTML: 7.4.9 - Step #160: Manufacturing Time & Battery Capacity Fields
7.4.9 Step 170	Ensures the pacemaker provides programmer-readable and writable non-volatile memory for manufacturing information (Model, Serial, Initials, DOB, Implant Date). The Manufacturing Data Buffer must be readable, must accept new data, must persist across a power cycle, and the original contents must be restorable.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Get Manufacturing Data Buffer (baseline) 4. Step 4 – Write new data to Manufacturing Data Buffer 5. Step 5 – Power cycle device and reconnect BLE 6. Step 6 – Verify Manufacturing Data Buffer persisted 7. Step 7 – GetPatientInfo and log fields 8. Step 8 – Restore original Manufacturing Data Buffer 9. Cleanup – Disconnect and Power Off	After performing a power cycle, the test unit retrieve the data for patient info and manufacturing buffer as they were set in the programming	Pass	See HTML: 7.4.9 - Step #170: Manufacturing Data & Patient Info Memory Verification

7.4.10 Step 10	Verifies correct behavior of ClearLog, TriggerLogEvent, and ReadLogData commands. Ensures logs are cleared, new entries are generated, and ReadLogData returns valid 0x3C frames with expected byteCount and log IDs.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Clear logs with IDs 0, 1, and 2 4. Step 4 – Start Streaming for 10 seconds 5. Step 5 – Stop Streaming 6. Step 6 – Send Version command 7. Step 7 – Trigger first log event 8. Step 8 – Read LogData for ID = 0 9. Step 9 – Get Time 10. Step 10 – Trigger second log event 11. Step 11 – Read LogData for ID = 1 12. Step 12 – Get Parameters 13. Step 13 – Trigger third log event 14. Step 14 – Re-read LogData for ID = 1 15. Step 15 – Get Patient Info 16. Step 16 – Trigger fourth log event 17. Step 17 – Read LogData for ID = 2 18. Cleanup – Disconnect and Power Off	First log file display Get Version command. Second log file display Get Time command. Third log file identical to second log file. Fourth log file display Get Patient Info	Pass	See HTML: 7.4.10 - Step #10: Event Log Mechanism Verification
7.4.12 Step 10	Ensures that the device allows configuring the rate response algorithm parameters via SetRateResponseParameters and that the updated values are correctly reported back by GetRateResponseParameters while pacing mode is set to VVIR (Mode 4).	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – GetRateResponseParameters (baseline) 4. Step 4 – Set pacing parameters to VVIR mode 5. Step 5 – Read parameters to confirm VVIR mode 6. Step 6 – Set Rate Response parameters 7. Step 7 – Verify Rate Response parameters 8. Cleanup – Disconnect and Power Off	The Rate Response Algorithm Parameters were read, written, and read again and showed that they were changed from original value	Pass	See HTML: 7.4.12 - Step #10: Rate Response Algorithm Parameters

7.4.12 Step 40	This test validates that after placing the pacemaker in OOO (no pacing), performing a full power cycle, and toggling the magnet, the implant produces *exactly one* pacing pulse during boot. This pulse is captured from TP38 via the Digilent Analog Discovery.	1. Step 1 – Power up and connect via BLE 2. Step 2 – Enter Programmer Mode 3. Step 3 – Send SetParameters (Mode 0 - OOO) 4. Step 4 – Confirm mode is now 0 (OOO) 5. Step 5 – Start TP38 capture (Max Sampling Rate) 6. Step 6 – Power cycle implant during capture 7. Step 7 – Toggle magnet input after reset 8. Step 8 – Reconnect BLE after reboot 9. Step 9 – Verify pulses after TP38 capture completion 10. Cleanup – Disconnect and Power Off	After the power cycle there was a single pulse	Pass	See HTML: 7.4.12 - Step #40: Post-Reset Single-Pulse Verification

Note: Complete raw data, test descriptions, test methods (steps), acceptance criteria, evidence attachments, plots, and step-by-step execution traces for all tests are available in the accompanying Allure HTML report:
./allure-final-report/MP_Raw_data_Report.html

8.3 Bluetooth (BT) Test Results

Test Execution Summary Statistics

Metric	Count	Percentage
Total Test Cases	12	100.0%
Passed	12	100.0%
Failed	0	0.0%
Skipped	0	0.0%
Broken	0	0.0%
Total Execution Time	211.0 seconds	
Average Test Duration	17.59 seconds	

Test Suite Breakdown

Test Suite	Total	Passed	Failed	Skipped	Broken
test_761	1	1	0	0	0
test_762	1	1	0	0	0
test_763	9	9	0	0	0
test_764	1	1	0	0	0

Detailed Test Results

The following table provides detailed test information including test descriptions, test methods (steps), acceptance criteria, and test results. For complete raw data, evidence attachments, plots, and step-by-step execution traces, please refer to the accompanying Allure HTML report (BT_Raw_Data_Report.html).

Step #	Test Description	Step Description	Acceptance Criteria	Test Resolution Pass/Fail	HTML Reference
7.6.1 Step 10	Test 7.6.1 Step 10: Monitor BT Power (TP61) on DIO1 and verify Bluetooth starts advertising after magnet toggle.	1. Step 1: Open Digilent device 2. Step 2: Check if device is advertising (should not be) 3. Step 3: Power on device 4. Step 4: Connect to device via BLE (magnet toggle occurs) 5. Step 5: Monitor DIO1 (BT Power) after magnet toggle 6. Step 6: Create plot of DIO1 trace	When Bluetooth power goes high due to the presence of the magnet (starts Bluetooth Application), the Bluetooth Application shall start advertising	Pass	See HTML: BT Test 7.6.1 Step 10 - BT Power Monitoring and Advertising Verification

7.6.2 Step 10	Test 7.6.2 Step 10: Monitor BT Power (TP19) on DIO1 and verify Bluetooth starts advertising after magnet toggle.	1. Step 1: Open Digilent device 2. Step 2: Check if device is advertising (should not be) 3. Step 3: Power on device 4. Step 4: Connect to device via BLE (magnet toggle occurs) 5. Step 5: Monitor DIO1 (BT Power) after magnet toggle 6. Step 6: Create plot of DIO1 trace	When put into advertising state (the magnet put next to the pacemaker) the Bluetooth Application begins advertising with the name Calyan X.X, where X.X is the software model number under test	Pass	See HTML: BT Test 7.6.2 Step 10 - BT Power Monitoring and Advertising Verification
7.6.3 Step 10	Test 7.6.3 Step 10: To ensure the Calyan device responds with a NACK when it receives an invalid command.	1. Step 1: Power-up and bring BT up via magnet toggle 2. Step 2: Enter Programmer Mode (mode 4) 3. Step 3: Set Active Mode 4. Step 4: Send Get Parameters command to verify queue is not full 5. Step 5: Send Set Parameters command with invalid Pacing Pulse Width (3001) 6. Step 6: Verify NACK response	When in connected state and with the queue not full, the Bluetooth Application responded to the invalid command with a NACK	Pass	See HTML: BT Test 7.6.3 Step 10 - NACK Response to Invalid Command
7.6.3 Step 20	Test 7.6.3 Step 20: To ensure the Calyan device responds with a NACK when the current programming mode does not support a command.	1. Step 1: Power-up and bring BT up via magnet toggle 2. Step 2: Get device into Active State 3. Step 3: Set Mode to 2 4. Step 4: Attempt Set Active Mode command in Mode 2 5. Step 5: Verify NACK response	When a command is received and is not supported by the current programmer mode, and the response queue is not full, the Bluetooth Application returns a NACK. Response queue shown to be not full by the fact that Set Active Mode command when the Programmer Application was in Factory Mode was received and showed the valid response	Pass	See HTML: BT Test 7.6.3 Step 20 - NACK Response to Unsupported Command in Programming Mode
7.6.3 Step 40	Test 7.6.3 Step 40: To ensure the Calyan device uses a 128-bit service when the wireless interface is active to receive data.	1. Step 1: Power-up and bring BT up via magnet toggle 2. Step 2: Get 128-bit UUID for receiving data (PairedWrite Characteristic UUID)	The Bluetooth Software used a 128-bit UUID for receiving data (as shown when connected to the Calyan Test Application)	Pass	See HTML: BT Test 7.6.3 Step 40 - 128-bit UUID for Receiving Data
7.6.3 Step 50	Test 7.6.3 Step 50: To ensure the Calyan device uses a 128-bit service when the wireless interface is active for BLE notifications.	1. Step 1: Power-up and bring BT up via magnet toggle 2. Step 2: Get 128-bit UUID for BLE notifications (Notify Characteristic UUID)	The Bluetooth Software used a 128-bit UUID for BLE notifications (as shown when connected to the Calyan Test Application)	Pass	See HTML: BT Test 7.6.3 Step 50 - 128-bit UUID for BLE Notifications

7.6.3 Step 60	Test 7.6.3 Step 60: To ensure the Calyan device command queue is not full and the current programmer mode allows a command, software activates Request Service line and store the command to the command queue.	1. Step 1: Power-up and bring BT up via magnet toggle 2. Step 2: Enter Programmer Mode (Factory Mode 4) 3. Step 3: Monitor DIO10 and send first Set Parameters command 4. Step 4: Wait 5 seconds before second command 5. Step 5: Monitor DIO10 and send second Set Parameters command	When the new command is received while in connected state and the command queue was not full (as shown by the response to the first Set Parameters command), the Request Service line was set activated. The new command was queued as shown by their response to the second command	Pass	See HTML: BT Test 7.6.3 Step 60 - Request Service Line Activation
7.6.3 Step 70	Test 7.6.3 Step 70: To ensure the Calyan device initiates a data read and there is data in the command queue, software transmits queued command data in the order it was received. Monitor DIO8 and DIO9 (I2C lines) to capture command transmission to Pacemaker.	1. Step 1: Power-up and bring BT up via magnet toggle 2. Step 2: Enter Programmer Mode (Factory Mode 4) 3. Step 3: Monitor DIO8 and DIO9 and send Get Parameters command 4. Step 4: Wait before sending Set Parameters command 5. Step 5: Monitor DIO8 and DIO9 and send Set Parameters command	When the following commands: Get Parameters, Set Parameters (valid values) Were sent, the I2C line showed that they were received by the Pacemaker (i.e., transmitted by Bluetooth)	Pass	See HTML: BT Test 7.6.3 Step 70 - I2C Line Activity During Command Transmission
7.6.3 Step 80	Test 7.6.3 Step 80: To ensure the Calyan device returns with a Bluetooth software version when a Bluetooth software version command is received.	1. Step 1: Power-up and bring BT up via magnet toggle 2. Step 2: Enter Programmer Mode (Factory Mode 4) 3. Step 3: Send Get Version command with target 1	The response to the Get Version command returned the correct current version of the Bluetooth Firmware	Pass	See HTML: BT Test 7.6.3 Step 80 - Bluetooth Software Version
7.6.3 Step 90	Test 7.6.3 Step 90: To ensure that commands are allowed or disallowed according to specified programming modes. For each Set Mode of 4, 3, 2: - First command sent should be successful (non-NACK response) - Second command should receive a NACK response	1. Step 1: Power-up and bring BT up via magnet toggle 2. Step 2: Set Mode 4 and test Get Rate Response Parameters command 3. Step 3: Set Mode 3 and test Get Active Mode and Init DFU commands 4. Step 4: Set Mode 2 and test Get All Device Parameters and Get Address commands	For each Set Mode of 4,3,2, the first command sent was successful and got a non-NACK response and the second command set received a NACK response	Pass	See HTML: BT Test 7.6.3 Step 90 - Command Allowance by Programming Mode
7.6.3 Step 100	Test 7.6.3 Step 100: To ensure the Calyan device can retrieve Bluetooth stats.	1. Step 1: Power-up and bring BT up via magnet toggle 2. Step 2: Send Set Mode command with value 4 3. Step 3: Send BT Stats command	CTA shall retrieve Bluetooth stats when BT Stats command was sent	Pass	See HTML: BT Test 7.6.3 Step 100 - Bluetooth Stats

7.6.4 Step 40	To ensure Calyan device supports queueing a minimum of two commands.	1. Step 1: Power-up and bring BT up via magnet toggle 2. Step 2: Set to Factory Mode (Mode 4) 3. Step 3: Send Get Parameters 4. Step 4: Send Set Device Parameter with param_id=4, value=0 (BLUETOOTH_MESSAGES_TRIM) 5. Step 5: Send Get Device Parameter with param_id=4 to verify value=0 6. Step 6: Send Two Commands (Set Programmer Mode + Set Active Mode) to test queueing 7. Step 7: Send Get Parameters to verify values 8. Step 8: Send Get Device Parameter with param_id=4 to verify value remains 0	Get parameter and Get Device Parameter with 4 (BLUETOOTH_MESSAGES_TRIM) shall display the programmed values that were programmed using Send Two Commands button	Pass	See HTML: BT Test 7.6.4 Step 40 - Command Queueing Support

Note: Complete raw data, test descriptions, test methods (steps), acceptance criteria, evidence attachments, plots, and step-by-step execution traces for all tests are available in the accompanying Allure HTML report:
./allure-final-report/BT_Raw_Data_Report.html

Report Generated: 2026-01-15 07:21:02 | Source: Allure Test Results | Total Test Cases: 12 | Pass Rate: 100.0%

9. Deviations & Observations

No Deviations: All test cases executed as planned. No failures, broken tests, or unexpected observations were recorded during test execution. All acceptance criteria were met.

General Observations:

- All automated test scripts executed as designed
- Device communication remained stable throughout testing
- Test evidence (logs, screenshots, waveforms) was captured successfully
- No environmental issues impacted test execution

10. Conclusions & Recommendations

Conclusion:
All acceptance criteria were met for combined validation testing across Main Processor (MP version 1.29), Pacing Engine (PE version 1.05), and Bluetooth (BT version 1.03); therefore, it can be concluded that all subsystems met their user needs and intended use. The validation testing demonstrates full compliance with the specified requirements, with 74 out of 74 test cases passing successfully (100.0% pass rate).

10.2 Recommendations

Based on the test execution results, the following recommendations are provided:

1. **Documentation Review:** Review the detailed Allure HTML reports for comprehensive test logs, step-by-step execution traces, and evidence attachments
2. **Issue Resolution:** Investigate and resolve all failed and broken test cases identified
3. **Regression Testing:** Re-run failed tests after fixes to verify resolution
4. **Record Retention:** Maintain this summary PDF alongside the detailed reports for regulatory submission
5. **Change Control:** Document any known issues, deviations, or limitations in the appropriate systems

10.3 Declaration

I hereby declare that the software verification testing documented in this report has been executed in accordance with the approved test protocols. All test cases have been completed successfully, and the software under test has demonstrated compliance with the specified requirements.

Verification Engineer: _____ **Date:** _____

— END OF DOCUMENT —

Report Generated: 2026-01-15 07:21:02 | Document ID: RAW_DATA_15th_JAN | Total Test Cases: 74 | Pass Rate: 100.0%