Developer's Guide To Pacemaker Development

Tutorial 1.4: Getting Started II

SFWRENG/MECHTRON 3K04
McMaster University

September 2, 2025

Getting Started With The Pacemaker Development Platform

The purpose of this tutorial is to provide a high level description of the platform you will be using to develop and verify the pacemaker system. By now, perhaps you are curious to learn more about the items in your project kit. By the end of this tutorial, you will gain an understanding of the functionality, components and interfaces of the kit you have received. You will also understand how to operate the HeartView test suite to verify the pacemaker system.

Topics Covered

- Pacemaker development platform overview
- Comparison of the pacemaker development platform and implantable pacemakers
- Deploying and debugging a model on the FRDM-K64F
- Introduction to the Testing Controller and HeartView

Prerequisites Completed **Tutorial 1.3**

1 Background

1.1 Housekeeping — The SFWRENG/MECHTRON 3K04 Project Kit

The following figure is an annotated image of the pacemaker project kit. Before moving on to the next section of this tutorial, take a moment to make sure you have all the components shown in Figure 1:

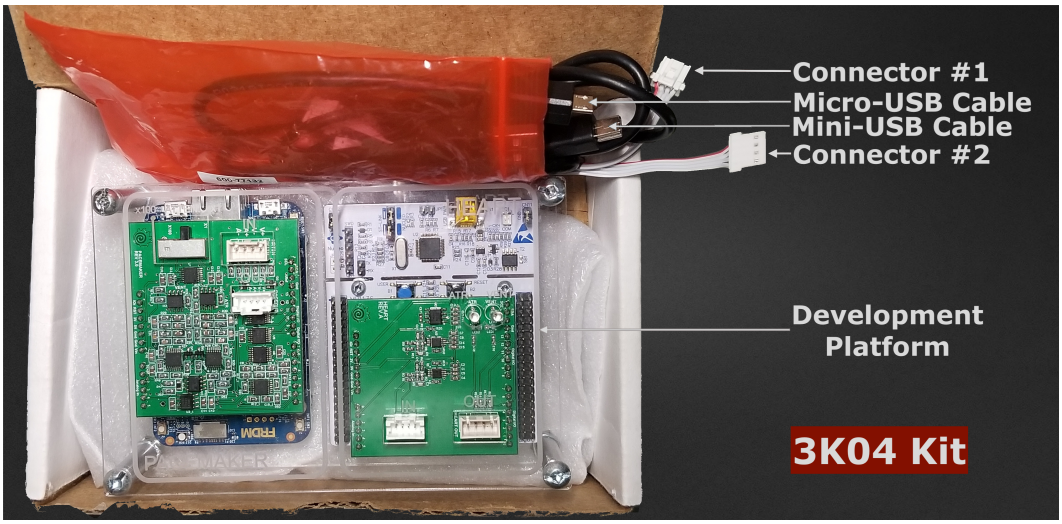


Figure 1: SFWRENG/MECHTRON 3K04 Kit

1. Flat ribbon cable with 1.27mm Molex connectors
2. Flat ribbon cable with 2mm Molex connectors
3. Micro-USB Cable
4. Mini-USB Cable
5. Development platform assembly

If you are missing any or all of the components in the above figure, notify your TA immediately. Do not leave the lab with an unreported uncompleted kit, otherwise, it will be your responsibility.

1.2 What is the Pacemaker Development Platform?

The Pacemaker Development Platform is an integrated bench top development and testing environment for implementing and verifying embedded pacemaker

software. The platform is made up of two components, as shown in Figure 2.

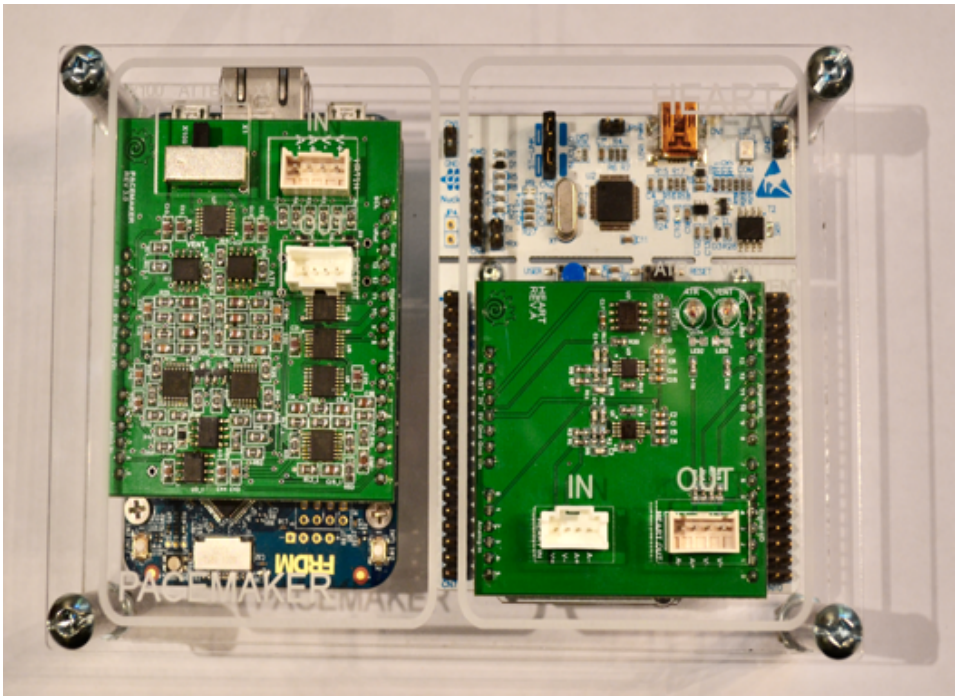


Figure 2: Pacemaker Hardware Reference Platform (left) and Heartview Testing Controller (right).

Pacemaker Hardware Reference Platform
The component on the left hand side is a reference hardware for the pacemaker pulse generator. It is made up of the FRDM-K64F microcontroller and a driver circuit called the **pacemaker shield**. The pacemaker shield is mounted on the K64F and connected to its pins.

Pacemaker Testing Controller
The component on the right hand side is a testing controller that will allow you to simulate the electrical activity of the heart and verify the pacemaker system. It is made up of the Nucleo-L476RG microcontroller and its own driver circuit.

1.3 Pacemaker System Functional Description

To understand the interactions between the two components of the development platform, it may be helpful to visualize each component in the context of the pacemaker system. Figure 3 shows the system level functional description of the pacemaker system.

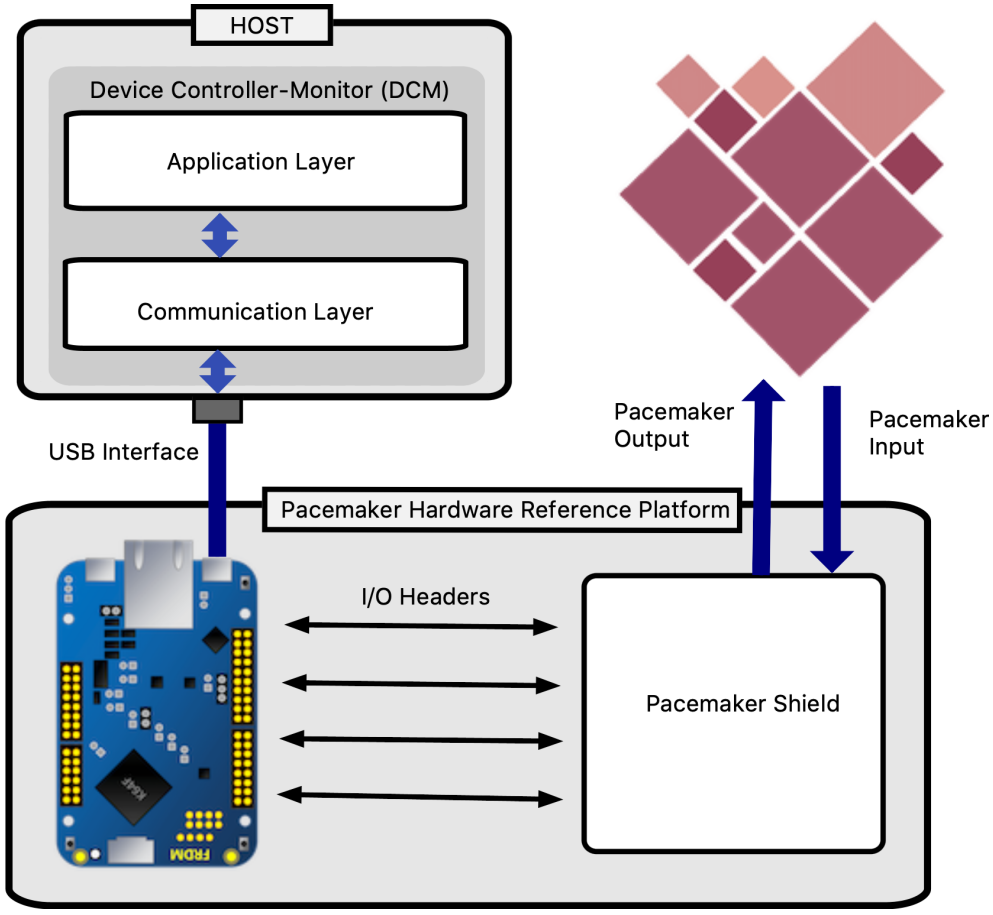


Figure 3: Pacemaker System Functional Description.

Pacemaker Hardware Reference Platform: represents the pacemaker pulse generator embedded computer. The K64F executes the embedded pacemaker software that controls the GPIO pins to drive the pacemaker shield. The embedded pacemaker software also has a communication layer that implements a communication protocol for communicating with the DCM over a virtual serial port as shown in Figure 4 below.

Host: represents your computer. The host machine communicates with the

K64F using serial communication over the **USB interface**. **Note:** more information on serial communication will be provided in **Tutorial 3: Serial Communication**.

USB Interface: represents the micro-USB cable connection between your computer and the K64F. The USB interface is also used to supply power to the K64F.

Device Controller-Monitor (DCM): represents a **user application** you will run on your computer to interact with the pacemaker model that is running on the FRDM-K64F microcontroller (as embedded software). Your DCM will be used to change the behaviour of the embedded software by modifying programmable parameters or to send instructions to interrogate the pulse generator. The **Application Layer** comprises the front-end graphical user interface (GUI) that allows the user to interact with the embedded software using interactive visual components such as graphical icons and/or audio indicators. The **Communication Layer** implements a communication protocol that is shared between the embedded software and the DCM to translate user application information to serial data (and vice versa).

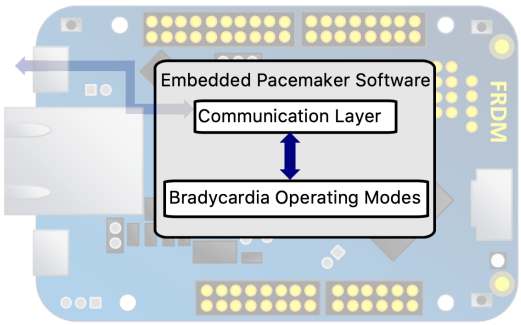


Figure 4: Simplified Functional Description of the Embedded Software.

Heart: in theory, the image in the top right corner of Figure 3 would represent a heart. In the context of the project, sadly the pulse generator will not be connected to a real heart. For our purposes, the image in the top right corner of Figure 3 represents the pacemaker testing controller. The pacemaker testing controller is described in more detail in The Testing Controller and HeartView.

Pacemaker Output: represents the pacing functionality of pacemaker leads that allow it to deliver electrical signals to the heart. For our purposes, one set of ribbon cables is used to connect the pacemaker output to the heart input. The ribbon cable has four wires representing the **pacing output** to:

the ventricle ring electrode, ventricle tip electrode, atrium ring electrode and atrium tip electrode, respectively.

Pacemaker Input: represents the sensing functionality of the pacemaker leads that allow it to carry electrical activity from the heart. A separate set of ribbon cables is used to connect the heart output to the pacemaker input. The ribbon cable has four wires representing the **sensing input** from: the ventricle ring electrode, ventricle tip electrode, atrium ring electrode and atrium tip electrode, respectively.

1.4 The Testing Controller and HeartView

The station you will use for the project is also a platform that provides tools for the measurement of pacemaker signals and the manipulation of natural heart signals to verify the correctness of the pacemaker system. The **testing platform** is comprised of the Pacemaker Testing Controller and the HeartView user application. For more information on the testing platform, please refer to the document **Intro To HeartView**.

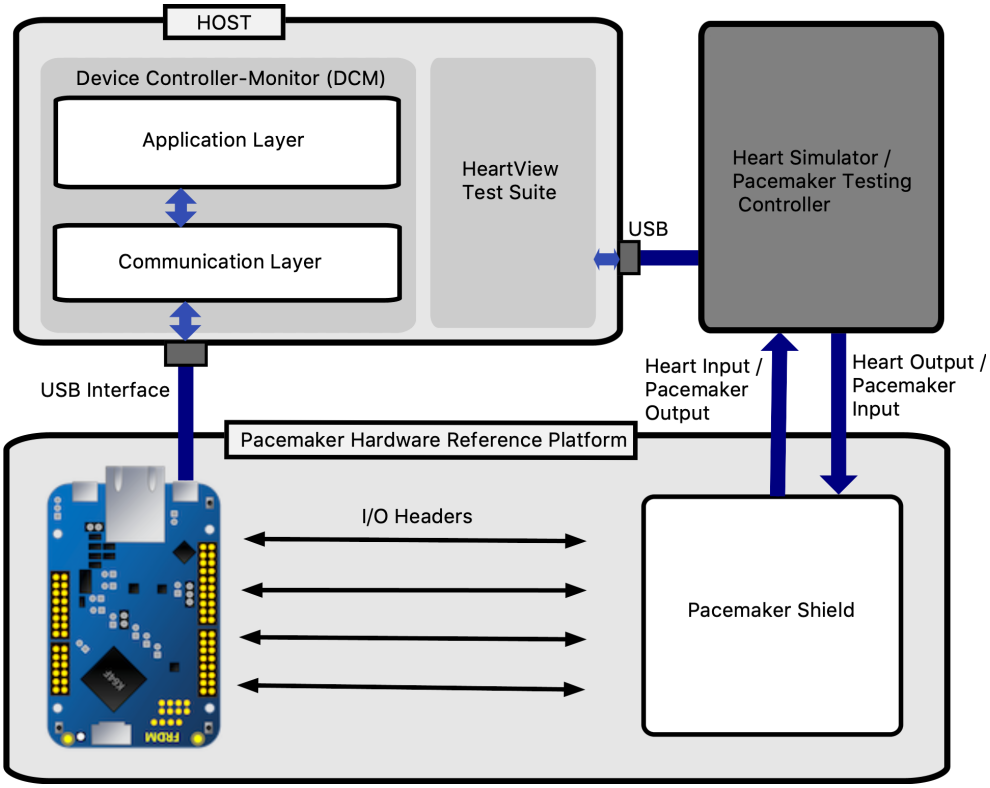
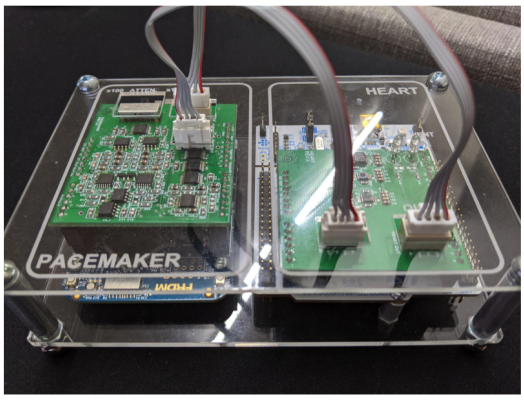


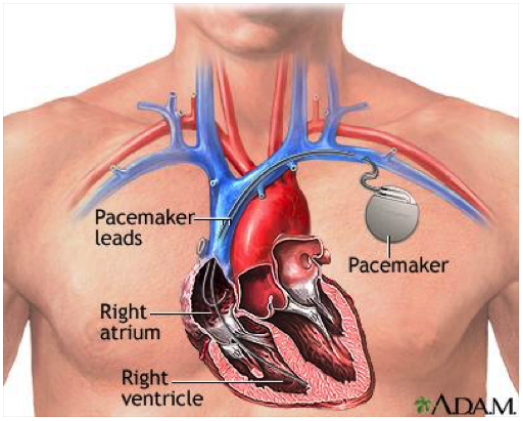
Figure 5: Pacemaker System Functional Description.

HeartView: a test suite for the pacemaker is used to simulate the electrical activity in the atria and ventricles, and visualize the pacing signals produced by the pulse generator. HeartView is a user application that you can run on your computer.

Pacemaker Testing Controller: generates natural heart signals to verify the pacemaker. The testing controller also receives pacemaker output and transmits them to your computer. The testing controller uses the USB interface to communicate with the HeartView program and receive power from your computer.

1.5 The Pacemaker Development Platform Compared To Implantable Pacemakers





There are some notable differences between our hardware reference platform and an implantable pacemaker.

1. **Size:** Implantable pacemakers are typically smaller than our reference platform. The smallest implantable pacemakers today are as small as a medical pill!

2. **Telemetry:** As indicated in Figure 5, our platform uses a wired connection medium for communication between the DCM and the pacemaker. Implantable pacemakers typically use a wireless communication medium (i.e. transmitting of digitally coded radiofrequency waves using a programming wand) to alter programmable functions or retrieve parameters noninvasively.

3. **Leads:** Implantable pacemaker leads perform sensing and pacing through the same electrical connection for each chamber. On the contrary, our platform has one distinct connection (ribbon cable) for delivering pacing stimuli and another distinct connection for sensing the heart signal. This is because the pacing circuitry in the pacemaker shield is separate from its sensing circuitry. (Essentially this means your pacemaker will only be able to sense the electrical activity generated by the testing controller; **it will not be able to sense its own paces**).

   The following figure compares the configuration of the pacemaker system connectors with implantable pacemaker leads. The blue arrow line represents the direction of positive current flow relative to the pulse generator when delivering pacing stimuli. The red arrow line represents input sig-

nal direction relative to the pulse generator when sensing natural heart activity.

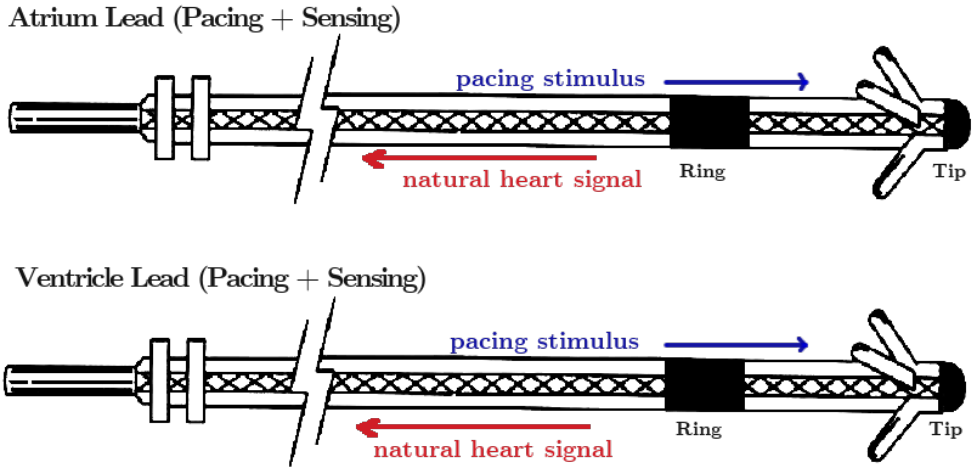


Figure 6: Implantable Pacemaker Leads

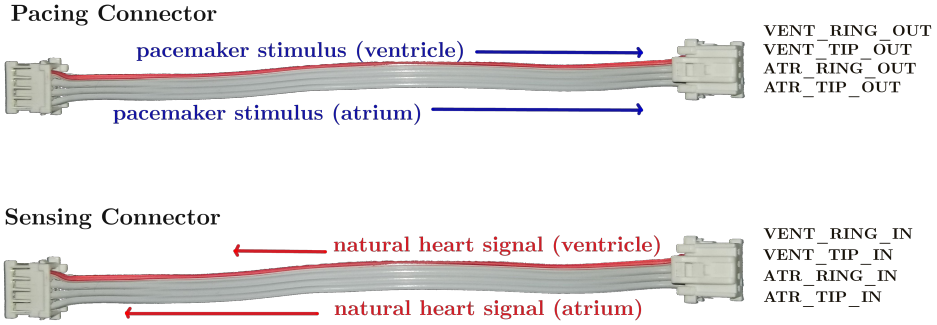


Figure 7: Connectors for the Pacemaker System

Deploying & Debugging Embedded Software

We approach the end of our discussion on the Pacemaker Development Platform with a demonstration of how to deploy a pacemaker model from Simulink® to the FRDM-K64F microcontroller and a description of steps you can take to debug your model during development.

Building and Deploying Embedded Software

The following diagram illustrates the general process that is followed when you run the command to deploy code to the FRDM-K64F in Simulink®.

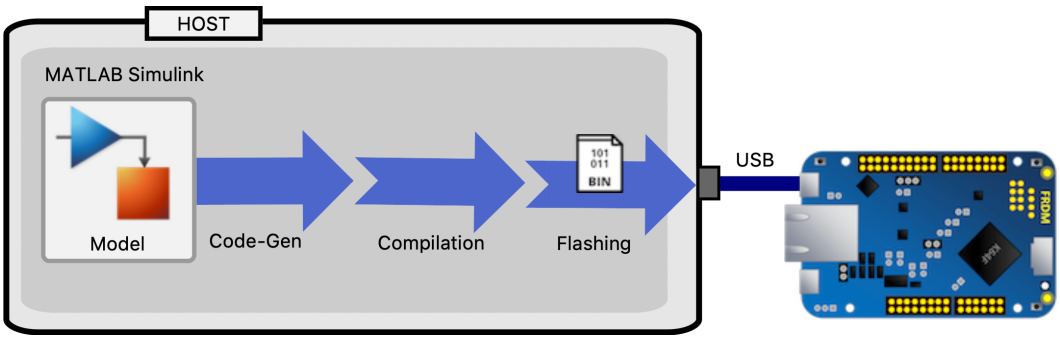


Figure 8: Build and Deploy Code to the K64F

You may flash the blinking LED model you created in **Tutorial 1.3**. To compile your model and flash the board:

1. Press CTRL + B (for Windows) or ⌘ + B (for Mac), *or*
2. Click the "Build, Deploy and Start" button on the Hardware tab in the top toolstrip

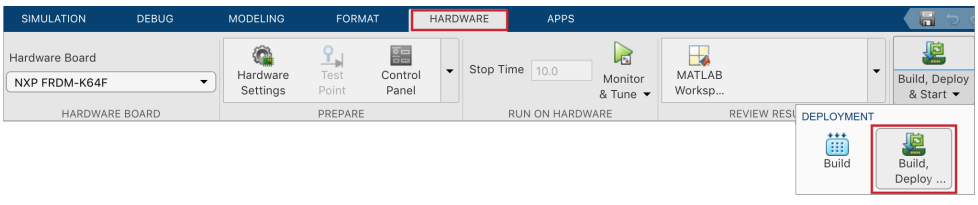
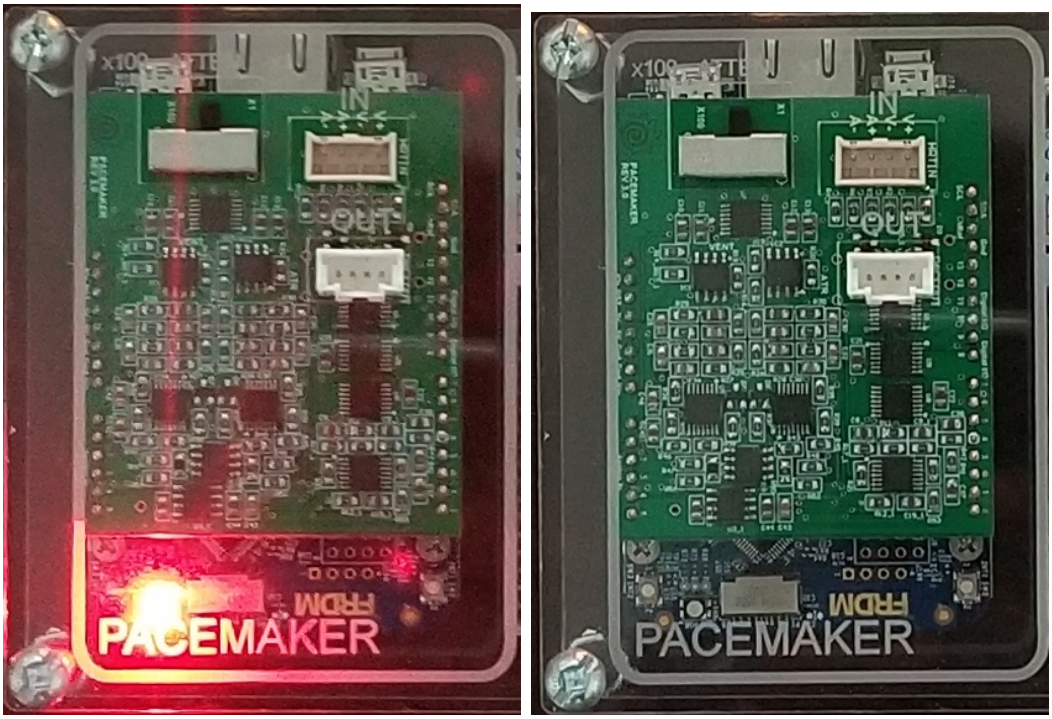


Figure 9: Build, Deploy and Start menu

Hardware

Tutorial 1.2 Part C.

Note that deploying can take some time. Once the flashing is complete you should see the LED blinking on your microcontroller.



(a) State **LED_ON** (b) State **LED_OFF**

Figure 10: Blinking LED

Build Only

The **Build** icon in Figure 9 allows you to compile the model without deploying it to the hardware. A binary (*.bin* file) will be generated and saved in your current working folder (refer to the "Current Folder" pane in MATLAB® or type `pwd` on the MATLAB command line). The file name of the binary will be the same as the name of your model.

To flash the binary at a later point in time, click and drag the file to the FRDM-K64FJ drive on your computer. The Build option may be useful in situations where you do not have your kit nearby but want to make sure your model compiles successfully. You can also send the binary to a team member

to flash to their board.

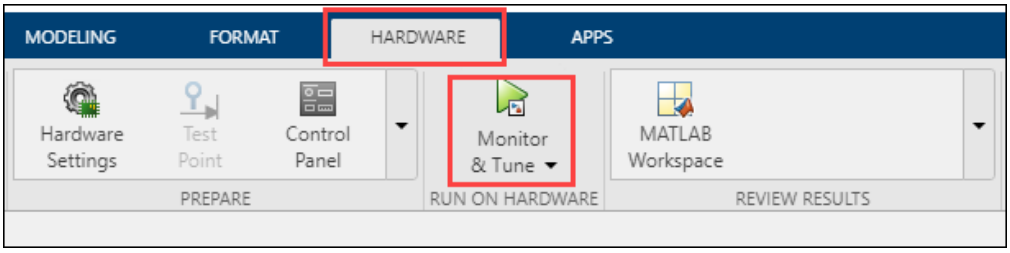
Debugging Embedded Software

LED Indicator: When it comes to debugging embedded software, we are not able to conveniently utilize temporary `printf` statements to track the flow of execution of the program as we can with application software. However, a close analogue to this approach is to use the onboard LEDs as a visual indicator to determine when a model steps through certain states.

Tip: you can use the onboard multicolor LED to encode the states in your model for debugging purposes.

Testing Controller LEDs: for modes involving sensing, the ATR/VENT LEDs on the testing controller can give you a visual indication of when the pacemaker is receiving natural heart activity.

Monitor and Tune: Alternatively, the OpenSDA J-Link firmware provides a debugging functionality that is supported in Simulink® with the Monitor and Tune feature. **Monitor and Tune** is a debugging tool that allows you to debug your model in real time as the Simulink Engine steps through the model. Simulink® uses serial communication over the USB interface to exchange data with the FRDM-K64F.



The Monitor and Tune button can be accessed in the Hardware tab in the top toolstrip. To use Monitor and Tune, you must include at least one "Sink" block (i.e. a Scope or Display block) in your model. A Scope block can be used to visualize the inputs you receive from the microcontroller pins or the output variables you write to the pins. You can also insert breakpoints to temporarily stop the flow of execution of the model and analyze the variables in your model workspace. Note that Monitor and Tune can be unreliable from time to time especially as your model becomes larger and more complex.

Demonstration — Using HeartView

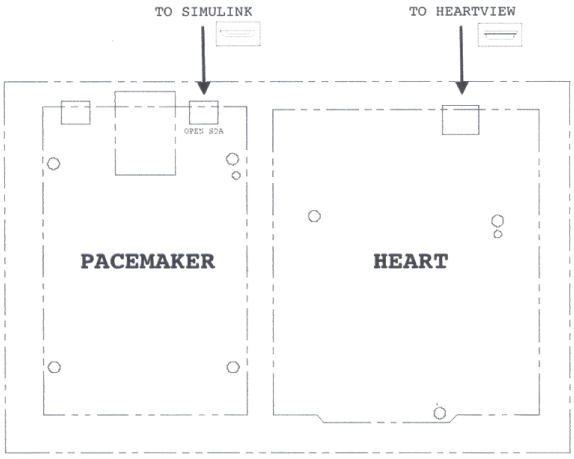
The HeartView application allows you to build test cases by generating natural heart signals, visualizing pacemaker pacing stimuli on a real time plot, and generating reports summarizing the results of your testing procedures to prove the correctness of your model.

Installing HeartView

The HeartView application is uploaded on Avenue To Learn. Unzip the files on the course website to install the application to your computer.

Connecting the Kit

Connecting the cables in the kit is fairly straight forward process. The connectors are designed to fit in one direction and the connection blocks on the kit are colour coded.



1. Connect one end of the 1.27mm ribbon cable to the **PACEMAKER OUT** connector and the other end to the **HEART IN** connection port.
2. Connect one end of the 2mm ribbon cable to the **PACEMAKER IN** connector and the other end to the **HEART OUT** connection port.
3. Connect one end of the micro-USB cable to the OpenSDA port of the FRDM-K64F board and the other end to your computer USB port.
4. Connect one end of the mini-USB cable to the testing microcontroller and the other end to another USB port on your computer.

Navigating the HeartView User Interface

The HeartView user interface is split into two main sections. The **left side** of the HeartView UI consists of a set of controls for the test suite. The **right side** consists of controls and displays for presenting the output of the testing procedure.

For more details on the use and operation of HeartView, review sections 2.3-2.5 of the document **Intro To HeartView**. Also refer to Page 13 of the document **Intro To HeartView** for a description of the general layout of the HeartView user interface.

Note: on establishing a serial connection with the testing controller in HeartView — a convenient way to determine out which COM port your testing controller is connected to is to:

1. Disconnect the testing controller from your computer
2. Click the refresh button on the **Serial Controls** panel of HeartView
3. Take note of the ports that are available
4. Reconnect the testing controller
5. Click the refresh button on the **Serial Controls** panel of HeartView
6. Select the new port that was added to the drop down menu

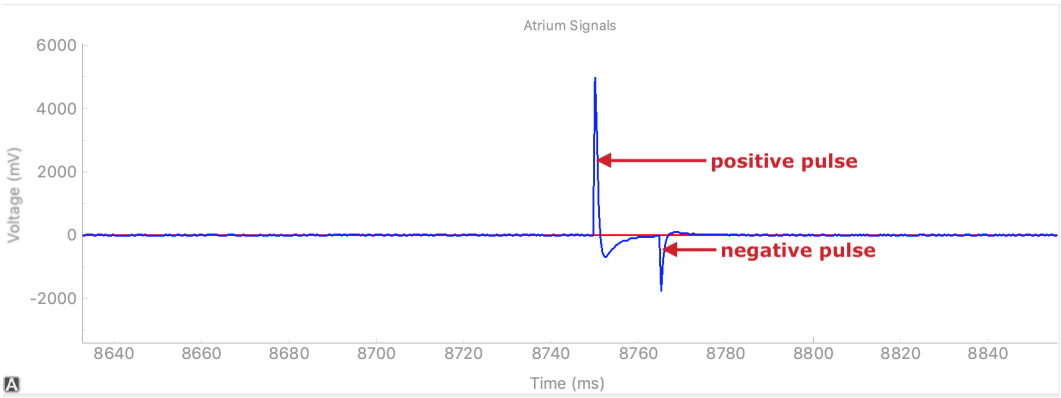
Notable Observations When Using HeartView

The following is a "sneak peek" of one of the modes you may be asked to implement at the end of the semester. DDD is a pacemaker mode that involves pacing the atria and ventricles, while responding to sensed activity in either atria or ventricles. An implementation of DDD was flashed to the K64F to obtain the resulting plots below. In the following plots, the blue waveform represents the pacemaker output waveform and the red waveform represents the natural heart signal.

Note that this demonstration is not an example of how to build good test cases. This subsection rather aims to address some observations you may have while using the HeartView user application. When using HeartView to verify your model, you are required to define a reasonable and sufficient set of test cases to prove the correctness of your model.

Interpolating the Plots: the above figure is a simulation of cardiac arrest or a situation where the heart is turned off. The pacemaker delivers paces to sustain the heart rate. Try to interpolate the plot in the figure above to determine the pacing period. Then determine the rate at which the pacemaker is delivering a pulse.

We pause the plot by clicking the Stop button on the Plotter Controls to take a closer look at the waveform.



Observation — Pacemaker Output Voltage Waveform: Note that the waveform has a positive pulse component and a negative pulse component. The waveform is correct, however it is different than the waveform described in Figure 12 in

the document **pacemaker_shield_explained** for the reasons discussed in section 3.3.1 of **Intro To HeartView**. When verifying your model, try to obtain the waveform in the figure above.

See if you can determine the pulse width (a.k.a. pace width) of the pulse in the above figure.

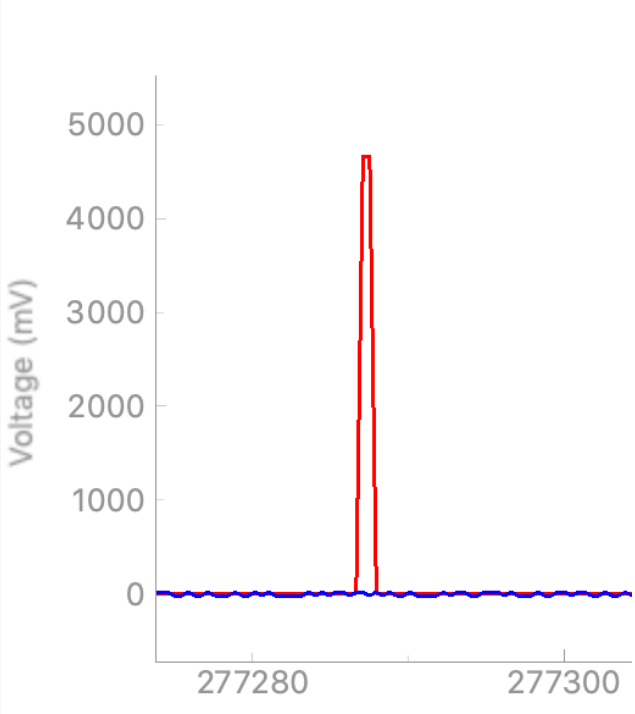
We dispatch another test routine and turn on the natural heart signals for both chambers. The red waveforms appear on the plots. When a test routine is dispatched, observe that the ATR/VENT LEDs on the testing controller are flashing based on the chamber that is receiving a natural heart stimulus.

Natural Heart Signals: It is important to note that the testing controller is not a closed loop heart simulator. The testing controller is intended to generate natural heart signals in order to verify the pacemaker. It does not generate action potentials in response to synthetic pulses generated by the pulse generator. When you are testing your software, it is possible for the testing controller to produce natural pulses immediately after a paced activity. You must verify that the pacemaker software responds appropriately. In the above figure, can you explain why the pacemaker does not inhibit any paces in response to the natural heart signals?

Pacemaker Inhibits Paces: after increasing the heart rate, the pacemaker begins to inhibit its paces after sensing natural heart stimuli. At a natural heart rate of 60 BPM, the pacemaker effectively delivers a single pace in between natural heart pulses to sustain its pacing rate. Note that the time period between the natural heart signal and the next pacemaker pace is equal to the pacing period.

Increase Natural Heart Rate: observe what happens when we increase the

natural heart rate to 160 BPM. The pacemaker inhibits all paces. Is this the correct functionality for the pacemaker in DDD mode given the pacing rate determined above?



Intracardiac Signals: a closer look at the atrium and ventricle signals shows that the heart simulator generates square wave waveforms with a strength of 5V. Natural intracardiac signals are not square waves, however the shape of the waveform will not affect your logic for the pacemaker model. On the other hand, natural intracardiac signals are typically within the millivolt range (as indicated in **Table 7: Programmable Parameters** in the Pacemaker System Requirements Specifications under **A or V Sensitivity**). Since the test suite generates signals in the voltage range, you can select appropriate values for the atrium sensitivity and ventricle sensitivity programmable parameters within the voltage range when verifying your model.

Ripples: Note that although the pacemaker output waveform is near zero, the blue waveform does not sustain a baseline voltage of zero volts perfectly. The small oscillations in the blue waveform are due to the effects of noise in the mV range and quantization from the testing controller's analog-to-digital converter. Do not worry if you are seeing these ripples.

2 Tutorial

1. Get a hardware kit from the TAs (two per team). Make sure you have all the items in the kit, if not, notify the TAs. The TAs will keep track of the kit numbers you are borrowing.
2. Explore the HeartView user application and begin to think about some test cases for Deliverable 1. Consider creating a test plan.
3. Review the document: Pacemaker Microcontrolled Shield.
4. Start organizing project tasks with your team.

3 Revision History

Version	Date	Modification	Modified by
1.0	Oct. 1, 2020	Initial Document Creation	Kehinde, Michael
2.0	Sep. 2, 2025	Update files and text	Zavaleta, Angela