

Software Development - 3K04

Assignment 1 - Ventricular VOO Pacemaker and DCM Interface

Instructor: Dr. Alan Wassyng
October 24, 2017

Group 7 - “Group 4”

Adam Cool	400032857	coola
Andrew Rehkopf	001412499	rehkopaz
Evan Gagich	400027794	gagiche
Ibrahim Tannir	001204850	tanniri
Michael Henry	400011728	henrym1
Michael Soosaipillai	400034820	soosaipm
Moshiur Howlader	001316948	howlam

Please note that this single document contains all details of the documentation of parts one through three. This document outlines the requirements, design decisions and testing of all parts.

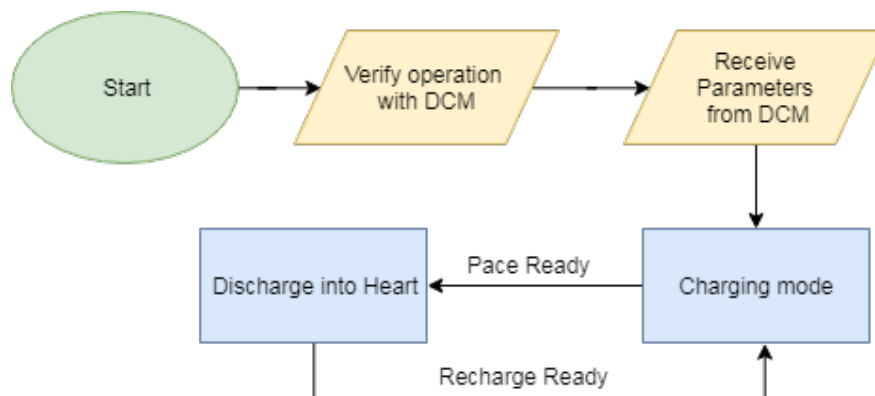
1 VOO Pacing: Part 1 and Part 3

The first iteration of pacemaker control is implemented in this part of the assignment. This simple version of the pacemaker control includes VOO operation meaning that the ventricle chamber is paced and both the ventricle and atrium are not monitored by the pacemaker. Using the FRDM-K64F microcontroller and MATLAB 2017b Simulink, the pacemaker shield provided as part of the 3K04 course is controlled to operate in the VOO mode. In the following sections, the design process followed during the assignment and every element that went into this operation are documented and justified.

1.1 Design Process

As part of the design process that went into the control of the pacemaker, a variety of concerns had to be addressed. The first concern is of course the safety critical nature of the operation where incorrect design choices could lead to injury or even death. To minimize these risks, essential design principles were incorporated into the design process. These principles include, separation of concerns, information hiding and high cohesion between modules. The next concern would be the correctness of the design, which can only be verified through vigorous testing against the requirements and dialogue with those who this device is for.

The overarching design of this section can be summarized in the following flowchart.



Pacemaker Flowchart

Note that the first two I/O operations that would communicate with the DCM have not been included in this iteration of the design due to complications with Simulink. For this reason, the parameters which dictate the operation of the design are constant. Also not included in the above flowchart is the additional operation of a button to inhibiting the pacing which is included in the design as part of the bonus.

1.2 Module overview

Please refer to the appendix for documentation of the entire Simulink model.

1.2.1 Input Block - USER INPUT

This module in this early iteration of the design, sets the constants that dictate the operation of the pacemaker. Within this block, the pulse duration, beat rate in beats per minute, amplitude of the pulse and PWM period are taken in. This block is contained to comply with information hiding where the means of getting said parameters are hidden from the rest of the design. At the moment, these values are set as constants but in upcoming iterations of the design these values will be for the most part, passed through serial communication from the DCM. Every value is taken as type uint16 except the amplitude which is implemented as a double to preserve decimal accuracy.

The requirements of this block are likely to change in future assignments where the required values to be passed may change their form or their units. In addition, security values such as a pass code could be passed into this block to enable safe communication with the DCM. Perhaps it may prove to be easier to transmit the double value amplitude as an integer through serial communication and then convert it to a double.

1.2.2 PWM Duty Cycle Calculation - PWM_CALC

This block is passed an enable signal and desired amplitude and through the following calculation, the duty cycle is calculated.

$$\text{Duty Cycle} = \frac{\text{Amplitude}}{5V} \times 100\%$$

This calculation results in an uint16 duty cycle percentage that is to be used for the PACING_REF_PWM PWM output pin. If the enable is set to true this value is given that which results from the calculation, if false, the duty cycle is set to zero. This block like that above is designed such that its internal workings have low impact and dependence on outside blocks.

The design of this block is unlikely to change in future designs as it can already account for any properly inputted amplitude so long as these values comply with the expected input of the block. This block may at sometime become unnecessary if perhaps the duty cycle is directly inputted through DCM serial communication. The requirements of this block will not require expansion in future iterations.

1.2.3 Beat Period Calculation - BEAT_PER_CALC

This block functions like that of the PWM duty cycle calculation block. It is designed to encapsulate the calculation of the heartbeat period. This calculation is documented below.

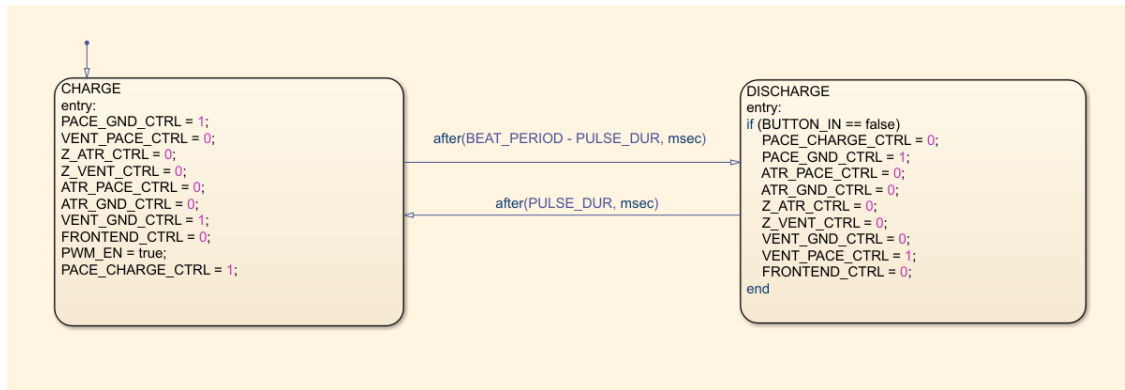
$$\text{Beat Period (ms)} = \frac{60}{\text{Beats Per Minute}} \times 1000$$

The block receives the desired beats per minute in integer form and outputs the period of a single heart beat in an integer representing the milliseconds per period. Future implementations of this block may require greater accuracy on the output which at the moment is an uint16 subject to small rounding errors of one millisecond.

There is not much in terms of design decisions that are subject to modification unless drastic change is made to the form of input (perhaps beats per second instead of bpm). Or take beat period thus making nullifying this block.

1.2.4 Open State Flow - OPEN_STATEFLOW

This is by far the most complex block in this design and thus it warrants the longest explanation. This open stateflow chart performs the pacing of the ventricle. The block takes three inputs and outputs to ten lines. The chart receives the period of the heartbeat from the BEAT_PER_CALC, a push button output and the pulse duration of the pace from the USER_INPUT block. It takes these values and then implements the aforementioned flowchart functionality of charging and discharging. This state flow is documented below.



Open State Flow

The chart begins in the charging state and then after the period of a heartbeat minus the pulse duration, it switches to the discharging state in which it remains for the duration of the pulse. After the duration of the pulse, it returns to the charging state. This flow is interrupted by the push button input which when pressed, will prevent the charge to discharge state change.

Note that for many pins, there is no state change however as these pins must be written to in later designs and are still expecting a value, they are given a value which minimizes their impact on the design. For example ATR_GND_CTRL experiences no change because there is no pacing to the atrium however this pin cannot be left floating as it may have undesired impact on the design which can be potentially hazardous thus it is set low. In addition, the order in which the pins are written to has also been carefully coordinated to prevent any undesired behaviour and to protect the patient. This order precisely follows that documented in the Pacemaker Explained document on Avenue to learn.

The design of this block will experience significant change in future iterations of the pacemaker. For instance, this chart only implements VOO functionality while the pacing of the atrium and sensing of both chambers are to be implemented in following iterations. These additional requirements will greatly change the flow of the state diagram by adding more states and intricate state transitions. In addition, the transitions between states will have to be redesigned to rework how the device is occupied as it waits for the transition as this time will be needed for sensing.

Requirements for this block are subject to change as well as the values taken as input currently do not account for any sensing pins which will be necessary for future functionality. In addition, the simple push button to disable transitions will prove to be pointless in following implementations as this button currently simulates receiving a natural pace which will be accurately detected later. There will likely be additional subsystems that will take input from the pacemaker and decode it to be used as an input for this chart. These subsystems will have specific outputs and types all of which will need to be accounted for in following design decisions.

1.2.5 Output Block (Hardware Hiding) - PIN_CONFIG

The purpose of this block is to perform hardware hiding for the interface between the pacemaker and the FRDM-K64F. The requirements are to translate control variables from the microcontroller to the pacemaker. Every input pin to the pacemaker is written to in this block to protect against floating voltage on pins. The requirements of this block are likely not to change as the block simply maps variables to their respective pins. At the moment, any unused PWM inputs are set to have zero duty cycle and every PWM pin is given a frequency of 1000 Hz.

1.3 Testing

The testing process for part 1 included individual tests for each of the functional blocks using the scope block in Simulink. In addition, an onboard LED is turned on during the discharge phase to confirm that the discharge occurred. This testing is outlined for each section below.

1.3.1 PWM Duty Cycle Calculation - PWM_CALC

Provided a variety of amplitudes of PWM as input to the block and confirmed that the appropriate duty cycle is returned.

1.3.2 Beat Period Calculation - BEAT_PER_CALC

Multiple values for the beats per minute are inputted into the block and the output is scoped to confirm that the correct period is computed.

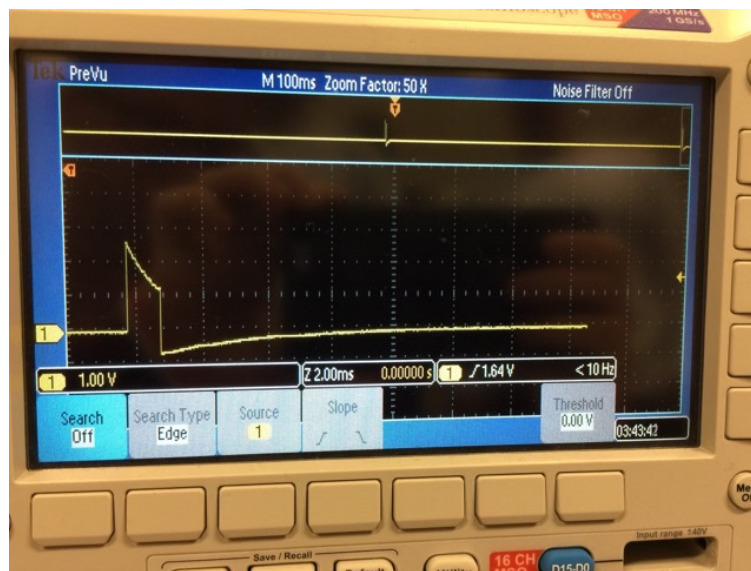
1.3.3 Open State Flow - OPEN_STATEFLOW

Using different values for beat period and pulse duration, VENT_PACE_CTRL is probed to check that indeed it is set high for the pulse duration then set low for the remainder of the beat period (beat period - pulse duration milliseconds).

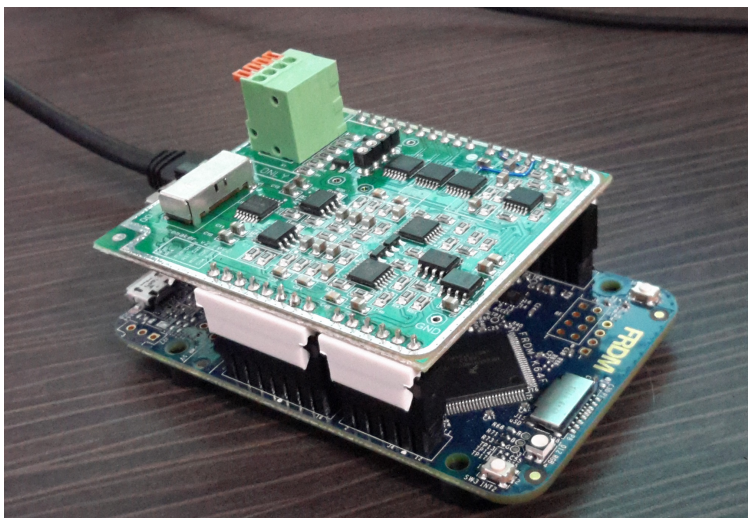
Finally the button input is checked in Simulink by setting the button input low to confirm that the pacing still occurs. When the button is pressed and set high, the pacing ceases.

1.3.4 System Testing

To confirm that the deployment to the board worked correctly, the LED tied to the discharge state is visually confirmed to pulse at a rate corresponding to the beat period. In addition, the ventricular ring output is probed with an oscilloscope to confirm that the amplitude is correct alongside the beat period and pulse duration. This result is documented below alongside a picture of the pacemaker in operation.



VOO Operation



Pacemaker in Operation

2 Device Controller-Monitor: Part 2

At this time in the development of the Device Controller-Monitor (DCM) the only changes that are anticipated are based on the requirements of future assignments. The most important functionality that will be added is serial communication. Currently the DCM cannot communicate with the pacemaker board rendering its functionally useless. Adding serial communication will allow all the components of the pacemaker to communicate and achieve full functionality. We will also add a range check for the user input of the programmable parameter. We will use a drop down menu, and a slider to help manage all the programmable parameter shown to the user in the DCM. Some more aesthetic changes will be needed.

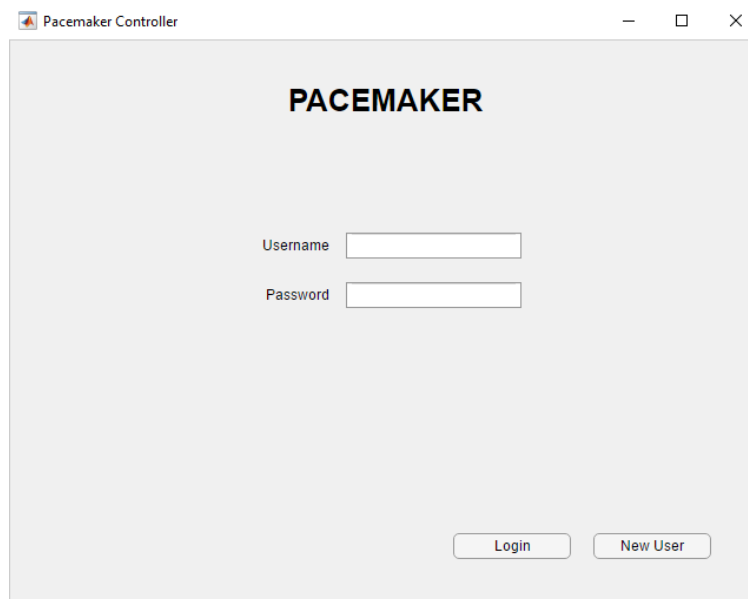
The design decisions that have been made are also unlikely to change. We selected MATLAB as the platform with which we designed and implemented the DCM. This decision was made because the other parts of the project are being designed and implemented using MATLAB. Having all project components be extremely compatible and easily integrated speeds up development and avoids challenging conflicts between differing standards. Due to these reasons, the design decisions made are unlikely to change.

The purpose of this component of the Pacemaker design is to allow for user interface and control of the pacemaker. This includes allowing for certain parameters in the pacemaker to be modified without direct contact with the physical Pacemaker. To allow for specific users to control these modifiable parameters, a graphical user interface was created using Matlab and contains several modules and functions.

2.1 Module Overview

2.1.1 Login Screen

The login screen is the first screen displayed upon running the program. This window contains two editable text fields for a user to enter the corresponding username and password. In addition, it contains a label for the title, a push button for if a user is new to the pacemaker program, and a push-button to log in. Upon pressing the 'New User' button, a new screen will be displayed where an individual can create an account. This window is documented below.



The DCM Login Screen

2.1.1.1 Login button Callback Function

Pressing the login button initiates a callback function which determines the GUI's next action. In this case, the program will check whether any users have been created by searching for a file where usernames and passwords are stored. If this file does not exist, it means no users are currently registered and the individual will not be able to

proceed to the Programmable Parameters window. If, however, the file does exist then the program will check whether the strings entered in the fields for “username” and “password” match any of the previously registered users which are stored in the file. A matching username and password will allow the user to proceed to the Programmable Parameters window, whereas a username and password that does not match will result in the user remaining on the Login window.

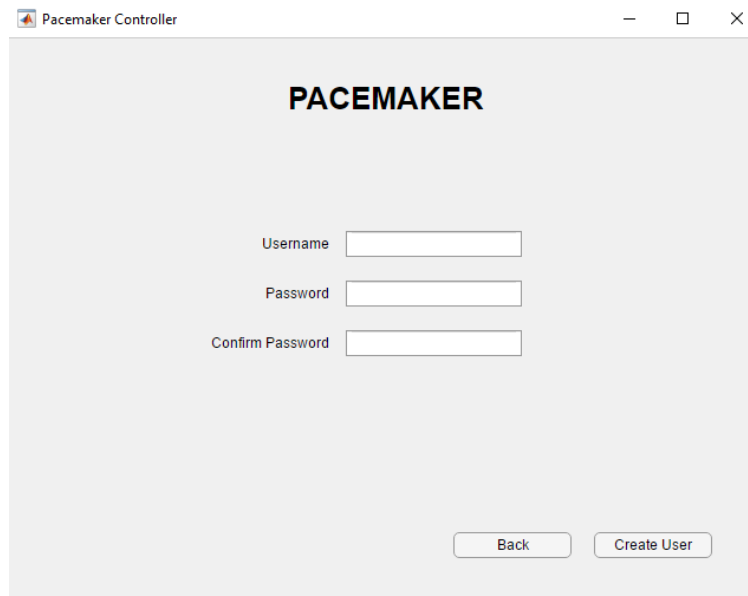
Perhaps the secret of this module is how we compare the entered username and password with each in the file one at a time by comparing the username first with the already saved usernames, and then checking if the password is correct for that username.

2.1.1.2 New User Callback Function

Pressing the “New User” button causes the Login Window to become invisible and the “Create User” window is displayed instead. This function also clears all editable textbox fields in the login window by replacing them with empty strings.

2.1.2 Create a New User Screen

If an individual presses the “New User” button from the login page, this window will be shown. Here, the person will have the opportunity to create an account by typing their desired username and password into the editable textbox and then pressing “New User”. This screen is displayed below.



The screenshot shows a window titled "Pacemaker Controller" with a light gray background. At the top center, the word "PACEMAKER" is displayed in bold black capital letters. Below this, there are three text input fields arranged vertically. The first field is labeled "Username", the second "Password", and the third "Confirm Password". At the bottom right of the window, there are two buttons: "Back" and "Create User".

The DCM New User Screen

2.1.2.1 New User Callback Function

When an individual is creating a new account, they must enter their desired password into two editable textboxes. This is to help avoid accidental errors in typing and ensure that the password linked with their username is the one they want. If these two passwords do not match and the “Create User” button is pressed, the user will be instructed to try again and the account will not be created. In contrast, if the passwords match and the “Create User” button is pressed, the program will then check to see if the file exists. If the file does not exist, it will be created as an .MAT file with one array for the usernames and one for passwords. When a user creates an account, the string will be stored in these array elements.

On the other hand, if the file does exist then previously entered usernames are and the username entered by the new user will be compared to usernames in each array index to ensure the username does not already exist. To avoid overwriting these usernames and passwords, when the number of accounts is about to exceed one, the new user’s information will be entered into the index corresponding to the length of the array plus one. In addition, the maximum

number of accounts allowed in this program is 10. In the event that an eleventh user is attempting to create an account, he/she will be notified that the program is at full capacity. If the program is not at capacity, and an account has been successfully created, the user will be returned to the Login Window.

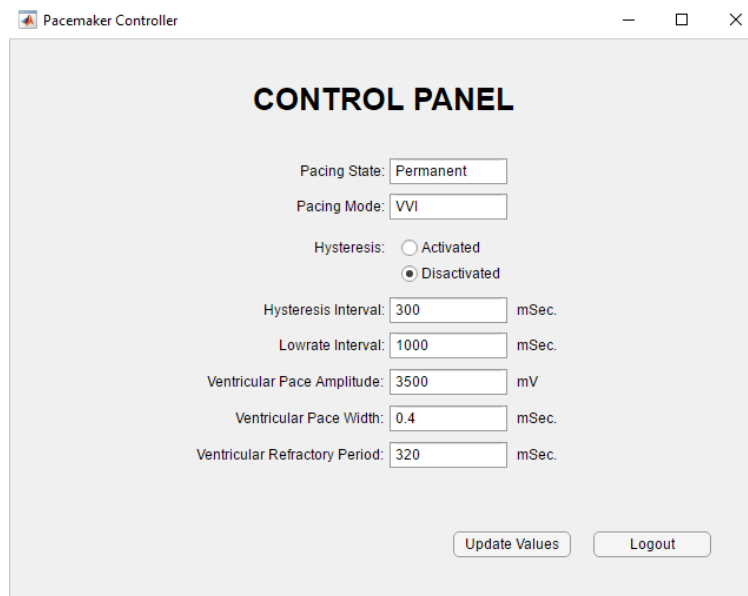
The secret of this module is perhaps how we compare the new user's username with those in the file one at a time for uniqueness. This is done by cycling through the user file and comparing each username with the username that wants to be created, if any of them match then this username is not unique.

2.1.2.2 Back Callback Function

Pressing the "Back" button causes the to become invisible and the "Create User" window is displayed instead. This function also clears all editable textbox fields in the New User Window by replacing them with empty strings.

2.1.3 Programmable Parameters Screen

After a successful login, this window will be shown. This window will allow the user to change any settings in the Pacemaker that can be changed, and see what current settings are implemented. These setting include: The pacing state; pacing mode; hysteresis toggle; hysteresis interval; lowrate interval; amplitude of the pace; duration of the pace; ventricular refractory period. Each setting is stored along with the user's name and password, so each user had their own unique settings, as well as being able to retain the settings from last time. This screen is documented below.



The screenshot shows a window titled "Pacemaker Controller" with a standard Windows title bar (minimize, maximize, close buttons). The main content area is titled "CONTROL PANEL" in bold. Below the title, there are several settings:

- Pacing State: A dropdown menu showing "Permanent".
- Pacing Mode: A dropdown menu showing "VVI".
- Hysteresis: Two radio buttons, "Activated" and "Disactivated". The "Disactivated" button is selected.
- Hysteresis Interval: A text input field showing "300" followed by "mSec".
- Lowrate Interval: A text input field showing "1000" followed by "mSec".
- Ventricular Pace Amplitude: A text input field showing "3500" followed by "mV".
- Ventricular Pace Width: A text input field showing "0.4" followed by "mSec".
- Ventricular Refractory Period: A text input field showing "320" followed by "mSec".

At the bottom right of the panel, there are two buttons: "Update Values" and "Logout".

The DCM Programmable Parameters Screen

2.1.3.1 Update Values Callback Function

Any modifications to the programmable parameters are saved to a .MAT file. If the file does not exist, then it will be created and the values corresponding to each parameter are stored.

2.1.3.2 Logout Callback Function

Pressing the "Logout" button causes the Programmable Parameters Window to become invisible and the Login Window is displayed instead.

2.2 Testing

In order to test the functionality of the DCM two different types of testing were employed, unit testing and integration testing.

2.2.1 Unit Testing of New User Window

The New User window was the first to be completed, so it was the first to be tested. First, all buttons were tested to make sure they redirected to the correct screen or ran the correct callback function. Once the buttons were verified to function correctly, the storing of usernames and passwords in .MAT files was tested and verified. All possible types of input were tested in the various fields. Through this testing it was verified that all invalid input was rejected by the DCM and an appropriate error message was shown to the user. These invalid input scenarios included the following: password field and verify password field did not match, one or more fields were left blank, username submitted was already registered, and 10 users were already registered. Once all test cases, that included these scenarios, were passed by the DCM it was declared fully operational.

2.2.2 Unit Testing of Login Window

The Login window was the second to be completed and tested. In the same manner as the previous unit testing, the buttons were tested to see if they directed to the correct screen or called the correct callback function. Then the login fields were tested to only allow access to the control panel screen if a valid username or password was entered. Erroneous inputs were rejected by the DCM and a corresponding error was shown to the user. These scenarios included entering a username and/or password that didn't match the stored values, and leaving one or more fields blank.

2.2.3 Unit Testing of Programmable Parameters Window

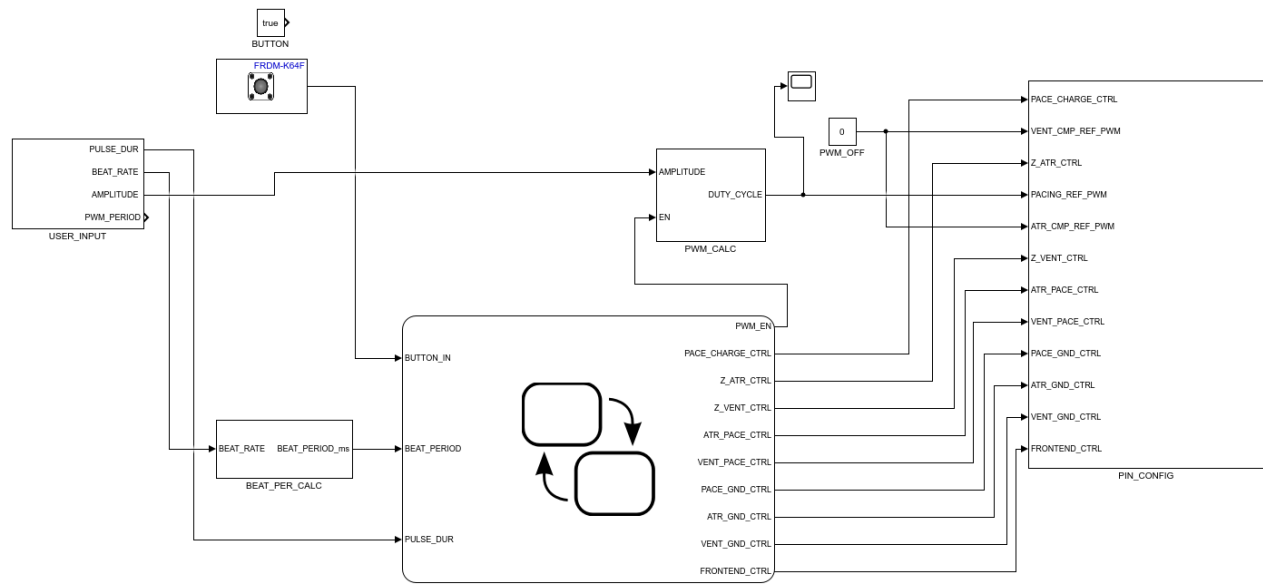
The Programmable Parameters window was the last to be tested. All buttons were verified to navigate to the correct screen or run the appropriate callback function. Then the storing of parameters was tested and it was verified that all entered values were stored and preserved across logins.

2.2.4 Integration Testing

Once all windows were verified to function independently they were combined and the entire DCM system was tested as a whole. Compound tests were used to see if users could cause any unexpected behaviour. The first part of this testing was to make sure a user could register and then log on once the registration process was complete. This revealed a problem where fields were not cleared of their contents when transitioning from window to window. This meant if someone logged out of the control panel their login credentials would still be visible on the login screen. The code was amended and then tested and verified that all input fields were cleared when transition between screens except for the programmable parameters. No other unexpected behaviour was revealed during the integration testing.

Once all tests were complete the DCM was declared fully functional and ready to be delivered.

3 Appendix



Full System