Documentation

DCM:

For us to be able to store our AOO, VOO, AAI, VVI, AOOR, VOOR, AAIR, VVIR programmable parameters for each user, we decided to use a SQL database. We started off by using a basic text file to just store the 10 users, later we moved to the SQL database as each user had 8 parameters. Using the text file would have been very messy and not organized using the SQL database; everything was easy to manage. Since we were using Tkinter for our GUI we had to use stringVar and doubleVar for our variables as a normal string would not be able to take different values based on user inputs. After running tests within our database and serial communication we can conclude that we are able to send and store the same data. The data was carefully implemented to match the structure of the pacemaker when sending and receiving the UART data stream. We weren't fully able to implement serial communication on the simulink side, thus the pacemaker does not actually receive the data. We also decided to use double data types because of the accuracy specifications and the allowed ranges. This is confirmed in our test methods in SerialCom.py.

Simulink:

Simulink is intended to receive the data in bits, and because each parameter was sent as a data type of type double, simulink was expecting to receive 8 bits per parameter. There are a max of 9 parameters (including the mode) that can be programmed (LRL, URL, AMPL, PW, ATR_Sensitivity, VENT_Sensitivity, ATR_REF_P, VENT_REF_P, and Mode), plus the two starting bits, so simulink was expecting 74 bits. Despite there being an atrial and ventricular variable in simulink for amplitude and pulse width, these parameters were received as one value each, as the mode that the user selected would tell simulink to assign that value to either an atrial amplitude and pulse width or ventricular amplitude and pulse width. With that being said, the received data is then casted as type double and assigned to simulink's internal variables.

However, data could never be received when testing our model, as it is suspected that the UART port0 on the board is not working. To explain, there are two ports on the board, and when the board is connected via the one corresponding to UART0, simulink is unable to detect it. We tried changing the UART port block on simulink to UART port 1, 2, or 3, but all these GPIO pins are in use by other outputs from the model corresponding to the pacing action of the atrium and the ventricle.

Sending data was also not possible due to the port not working, but the logic for it has still been implemented. This is achieved through a subsystem that packs each parameter into bits representative of type double to be received by the DCM. Upon receiving the data, the DCM unpacks it using 'struct.unpack' and specifying array indices in intervals of 8's, as each interval would represent one parameter. If the usb port had worked, this functionality would allow the doctor to verify that the parameters they set on the DCM are the same ones being used to pace the patient on simulink.