ASSIGNMENT 1

Simulinkers
Group 8
October 25, 2017
SFWR ENG 3K04

Table of Contents

Introduction

This assignment is the initial step in the process of the pacemaker project. This assignment is made up of two major sections. One part involved creating real time software on Simulink on the hardware platform (FRDM-K64F). This was done by creating an open-state flow diagram. The other part involves creating a preliminary version of the Device Controller Monitor (DCM) which is the user interface involved with the pace maker that allows for input from the user.

Part 1

1: Requirements Likely to Change

Because this is the initial step in the pacemaker project, this version of the pacemaker is not as complex as the expected final version. For example, this version of the pacemaker was only required to have one pacing mode (VOO). This pacing mode only involves the ventricle chamber to be paced with no chambers being sensed and no response to sensing. The final expected version would include more pacing modes which would involve pacing the atrium along with pacing the ventricle or independently of the ventricle. As mentioned previously, this current version does not sense a heartbeat. In the future, the pacemaker should be able to sense the heartbeat and then react as opposed to sending pulses at a set frequency. The future version of this pace maker should also be able to do rate modulation. It would be useful if the pace maker senses the activity level of the patient in whom it is implanted in and then ensure a sufficient blood flow necessary for the blood flow.

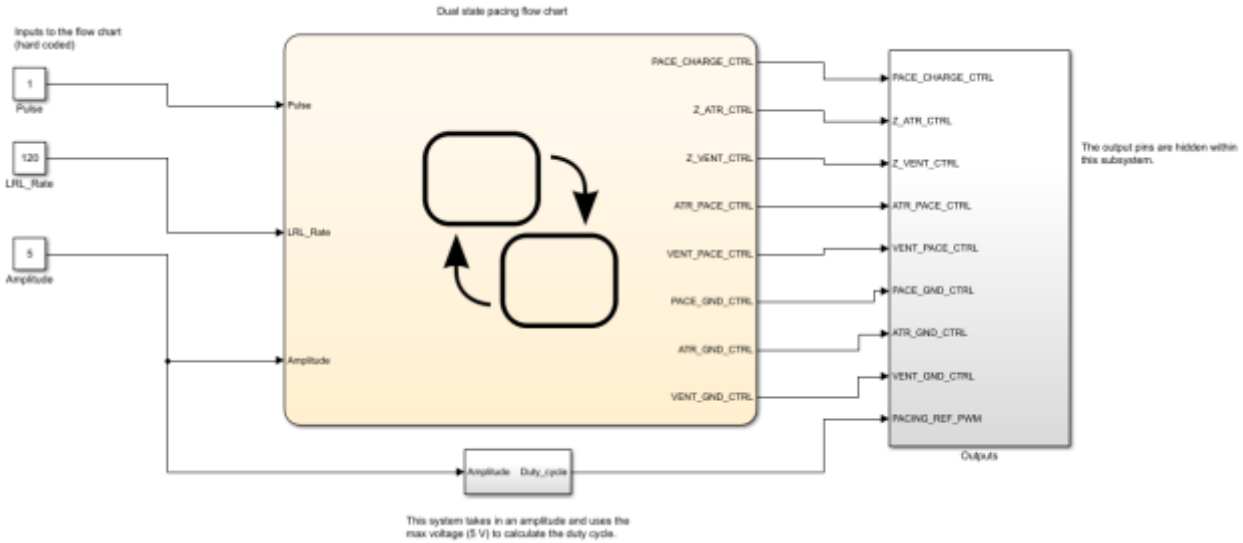
2: Design Decisions Likely to Change

Throughout this part of the assignment we went through numerous design iterations that played an integral part in the success of implementing VOO mode. To begin with, the initial states of the state flow diagram are discharging and charging. Next design decision was to create a hardware hiding subsystem for the output that would hide the specific pins used on the board. Also a subsystem was created to output a duty cycle given the inputs of pulse width and period.

Later on in this project an additional state called the waiting period would be added. This would be responsible for activating the pacemaker based on specific conditions instead of looping through the charging state right after the discharge state. For the future assignments, additional pacing modes need to be created to implement VOOR mode.

3: Simulink Diagram

The inputs to the state flow chart were all hard-coded values. These values were pulse, lower rate limit and amplitude. The amplitude input value was passed to a subsystem that calculated the duty cycle by dividing the given amplitude by the highest voltage value (5 V) and multiplying that by 100. For example, given an amplitude of 2.5 V, the duty cycle would be 50%.



We attached oscilloscopes in Simulink to the output pins and verified the output wave form was a square wave.

Part 2

1: Requirements Likely to Change

The initial version of the DCM currently has limited features and functionality. However, updates will be made in the future to develop a fully functional DCM. Here are some of the requirement changes that are likely to occur:

- Serial communication between the pacemaker and Simulink.
- A working feature that displays whether an active connection has been made between the DCM and the pacemaker.
- Capability of storing and altering all the programmable parameters for the pacemaker
- Store more than 10 patients in the database.

2: Design Decisions Likely to Change

- A Boolean variable should be implemented with checkboxes instead of textboxes.
- A confirm prompt before reprogramming the pacemaker parameters.
- Safety check to ensure pacemaker parameters that have been inputted are a reasonable value.
- Instead of a text file, use SQL as a safer system to store patient information and login details.
- Display patient's name on top of the pacemaker parameter page for Doctor's reference.
- Program should check whether entry is a string or integer.

3: Module Descriptions

Module: Login Page

a) Reference login details with a database to ensure correct login credentials. If the correct login credentials are entered, the pacemaker parameters page is called, otherwise a message box notification is displayed.

b) The user data is stored in a hash table which is saved within a text file. The program uses the input username as a hash 'key', and retrieves the saved password of the registered patient.

c) Public Function loadHash()

d) This function takes no parameters and loads the hash table from a known text file.

e) Global Variables:

User: This is a class which contains patient details such as name, username and password

Database: This a hash table which stores patient's information 'keyed' to a username.

f)

1. loginfetch(sender As Object, e As EventArgs) Handles MyBase.Load

2. newUser(sender As Object, e As LinkLabelLinkClickedEventArgs)

3. loginUser(sender As Object, e As EventArgs) Handles loginButton.Click

g)

Public Functions:

1. This function loads the hash table from the text file stored in a known directory. The function then de-serializes it, returning an object instance of a hash table.

Private Functions:

1. This function loads the hash table containing the user data of all registered user from a text file using the public function loadHash() as described above. It is an event handler.

2. This function checks the database for the number of patients registered. If there are more than 10 patients registered, a message box appears. Otherwise, the New Patient module is opened.

3. This function references the hash table from the database to determine whether the correct login credentials have been entered. If the username is not a key within the hash table, then the program displays a message box notification. If the username is a key within the hash table, then the program compares the value of the key with the entered password to check for a match. If there is a match then the program navigates to the parameters page, otherwise it displays a message box notification.

Module: New Patient

a) This module registers a new patient into the database. The database is stored as a hash table which contains the username as the 'key', and a class User as the 'value'. The class User will contain the following details: name[string], username[string], password[string], paceParameters[hash table]

b) There are no secrets to this module.

c) 1. Public Function createHash()

d) 1. This function takes no input parameters and creates a hash table and stores it within a text file.**e)** The global variables in this function are strings containing: name, username and password. There is also a hashtable containing the pacemaker parameters.

e) The global variables in this function are strings containing: name, username and password. There is also a hashtable containing the pacemaker parameters.

f)

Private Functions:

1. NewUserSignUp(sender As Object, e As EventArgs) Handles ConfirmNewUser.Click

g) Internal Behaviour:

Public Functions:

1. This function creates a directory called ‘K04’ and saves within it a text file named ‘database.txt’. If the text file already exists, the function deletes the text file and rewrites it. The text file stores the ‘Database’ hash table

Private Functions:

1. If any of the textboxes on the registration page are left blank then a message box notification is prompted. Otherwise, the function then creates a new instance of the class User, and stores the textbox information from the Registration Page within the class. This class is then stored within the database hash table and ‘keyed’ to the username. This database is then updated and saved.

Module: Pacemaker Parameters

a) The purpose of this module is to view, save and alter pacemaker parameters.

b) This module has no secrets.

c) This module uses only private functions.

d) Not applicable.

e) This module does not use any global variables.

f) Private Functions:

1. saveParameters()

2. load_textbox_vals(user As User)

3. initializeVals(user As User)

4. DefaultButton_click(sender As Object, e As EventArgs) Handles DefaultButton.Click

5. setValue_click(sender As Object, e As EventArgs) Handles setValue.Click

6. Display_Click(sender As Object, e As EventArgs) Handles Display.Click

7. ListBox2_SelectedIndexChanged(sender As Object, e As EventArgs) Handles ListBox2.SelectedIndexChanged

8. ListBox1_SelectedIndexChanged(sender As Object, e As EventArgs) Handles ListBox1.SelectedIndexChanged

g) Internal Behaviour:

1. This function iterates through each textbox parameter and updates the database value for the parameter. The update occurs if the value in the textbox is different than that stored in the database.

2. This function iterate through each textbox and sets its value to the saved value in the database. The value in the database can easily be accessed by using the parameter ‘key’ and the hash table.

3. This function sets the pacemaker parameters to their default values and calls initializeVals(user As User) to update the values.

4. If the ‘default’ button is clicked, the function initializeVals(user As User) is called.

5. If the ‘set value’ button is clicked, the function saveParameters() is called.

6. If the ‘display’ button is clicked, the function load_textbox_vals(user As User) is called. The textboxes in the DCM will display the saved value of each parameter.

7. This function will save the ‘state’ value within the database. The selected item in the list box will be converted to a string and saved within the parameter’s hash table.

8. This function will save the ‘mode’ value within the database. The selected item in the list box will be converted to a string and saved within the parameter’s hash table.

DCM Test Cases

Module: Login Page

Test Case: 1
Input: unregistered username
Expected Output: “Invalid Username”
Actual Output: “Invalid Username”
Pass/Fail: Pass

Test Case: 2
Input: Registered username and password
Expected Output: Parameters Page
Actual Output: Parameters Page
Pass/Fail: Pass

Test Case: 3
Input: Registered username and invalid password
Expected Output: Invalid Password
Actual Output: Invalid Password
Pass/Fail: Pass

Module: New Patient

Test Case: 1
Input: Username: ##$!# , Password: ##@@
Expected Output: “New Patient Saved in Database”
Actual Output: New Patient Saved in Database”
Pass/Fail: Pass

Test Case: 2
Input: Blank textboxes
Expected Output: “Please input the required information”
Actual Output: “Please input the required information”
Pass/Fail: Pass

Test Case: 3
Input: Existing username in to the database
Expected Output: “New Patient Saved in Database”
Actual Output: Error: System.ArgumentException
Pass/Fail: Fail

Module: Pacemaker Parameters

Test Case: 1
Input: String input for integer value.
Expected Output: “Invalid Entry”
Actual Output: No error or Prompt
Pass/Fail: Fail
Note: DCM should check for proper input type.

Test Case: 2
Input: Blank textboxes
Expected Output: No error
Actual Output: No error
Pass/Fail: Pass

Test Case: 3
Input: Integer values
Expected Output: No error, values saved and displayed upon login.
Actual Output: No error, values saved and displayed upon login.
Pass/Fail: Pass