SFWRENG 3K04: Software Development

Assignment 2 – Part 1 - Simulink Design

Dr. Alan Wassyng

December 4, 2019

Group 5: More Life Pacemaker

Adam Bujak – bujaka – 400113347

Eric Hillebrand – hillebre – 400143468

Harneet Singh – singhh76 – 400110275

Karan Gill – gillk20 – 400130973

Stephen Scott – scotts24 – 400139933

Table of Contents

Table of Figures

List of Tables

Likely Requirement Changes

- Incorporate rate adaptive modes (AOOR, VOOR, AAIR, VVIR, DOOR)
- For long term usage of the pacemaker, impedance of the muscles must be taken into consideration
- Improve pacing efficiency to conserve battery life
- Reduce the overall dimensional specifications of the electrical circuitry, thereby minimizing the space consumption
- Implement dual-pacing, dual-sensing functionality (DDDR)
- Implement accountability for hysteresis

Likely Design Changes

- Modularize the design to make the components more flexible and to allow design scalability
- To account for variation in muscle impedance, circuitry on the microcontroller can be used to measure muscle-resistance between different points in the chambers, which can then be used to adjust the programmable parameters
- Increase the rate at which electrogram (egram) data is sampled in order to attain a continuous plot
- Include a stateflow inside the SYS_IN subsystem to control the parameters for rate modulation
- Add various programmable parameters to be used in rate modulation
  (*SEE Table 7: Parameters introduced for the implementation of rate modulation*)
    - p_reactionDelay
    - p_recoveryDelay
    - p_sameBPMDelay
    - p_activityThreshold
    - k_currentActivity
    - k_BPM
- Adjust serial protocol for the incorporation of the new programmable parameters

Overview of Pacemaker Model Design

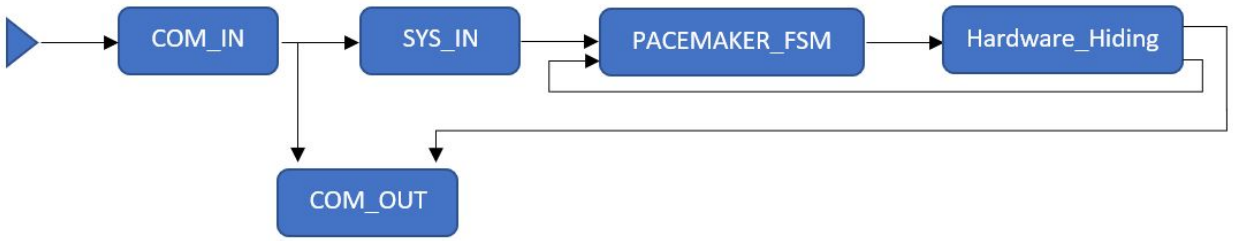

Figure 1: An overview of the stateflow objects

Denotes a 'receiving port' which allows and maintains a communication channel from the DCM to this model (collects programmable parameters set from the DCM interface)

COM_IN: Initializes the programmable parameters with default values, and when data is received from the DCM, this subsystem re-assigns the programmed variables after parsing the array of received data. This module also enables the DCM to display the programmable and electrogram (egram) values by reverting them through the COM_OUT global function

SYS_IN: Performs necessary functions to convert raw programmable variables to appropriate types and units of the Finite State Machine (FSM). This technique of making the values compatible with the FSM state helps eliminate unnecessary modification of the FSM's charts

PACEMAKER_FSM: Includes state flow charts to achieve the functioning of different modes such as AOO, VOO, AAI and etc. The hardware controlling variables are activated/deactivated from various states of this sub-system

HARDWARE_HIDING: Contains all the hardware components to control the pins on the microcontroller and eventually, the pacemaker shield. By toggling the pins, the switches in the circuitry can be turned ON/OFF and also, the charge on the capacitor can be altered to obtain a certain outcome. Having a separate sub-system for the hardware allows the capability to replace the hardware (in case of a fault) without making drastic changes to the whole system

COM_OUT: Represented as a global function, known as 'send_data()', transmits data back to the DCM where it can be displayed on the interface. After byte packing, data of length 21 bytes is sent through UART serial transmit port

Description of Subsystems

NOTE: For more details on inputs and outputs of the subsystems, please read the appendix section at the end of this document.

Serial Receive

- This receiving block reads data from the UART port on the board.

Parameters:

UART:	0	Specifies which port to use from the microcontrollers board
Data type:	unit8	Block receives data of type uint8
Data length(N):	23	Represents the length of data to be received 23 (bytes)
Output status:	Enabled	Status port is available to display the status of the read operation
Sample time:	-1	Determines the time according to the designed model

Outputs:

Rx:	Continuously delivers the data to the model after each sample time
Status:	Shows the result of receiving data operation. Outputs a zero when completed successfully, otherwise 32, which tells no new data received.

COM_IN

This subsystem declares default programmable values and receives data from DCM

Inputs: rxdata, status
Outputs: p_pacingMode, p_lowerRateLimit, p_upperRateLimit, p_atrPulseAmplitude, p_ventPulseAmplitude, p_atrPulseWidth, p_ventPulseWidth, p_atrThreshold, p_ventThreshold, p_arpDelay, p_vrpDelay, p_fixedAVDelay, p_rateModulation, p_modulationSenstivity, k_echoData
Locals: k_sampleRate
NOTE: This module is executed at each initialization step.

- The INIT state sets default values for the programmable parameters
- The model waits for the communication packet in the STANDBY state
- If a packet has been received and the first byte of the received data is 0x16 (a value chosen to indicate that the correct device is communicating with the simulink model), then there are three options to implement, namely SET_PARAM, ECHO_PARAM or ECHO_EGRAM

Table 1: State Transition Table of COM_IN subsystem

Current State	**Event**		**Next State**
INIT	---		STANDBY
STANDBY	status == 0 and rxdata(1) == 0x16	rxdata(2) == 0x55	SET_PARAM
		rxdata(2) == 0x22	ECHO_PARAM
		rxdata(2) == 0x47	ECHO_EGRAM
		NOT(rxdata(2) == 0x55 or 0x22 or 0x47)	---
	NOT(status == 0 and rxdata(1) == 0x16)		---
SET_PARAM	---		STANDBY
ECHO_PARAM	---		STANDBY
ECHO_EGRAM	status == 0 and rxdata(1) == 0x16 and rxdata(2) == 0x62		STANDBY
	NOT(status == 0 and rxdata(1) == 0x16 and rxdata(2) == 0x62)	After(k_sampleRate)	ECHO_EGRAM
		NOT(After(k_sampleRate))	---

State Configurations

INIT

p_pacingMode = 0
p_lowerRateLimit = 60
p_upperRateLimit = 120
p_atrPulseAmplitude = 3500
p_ventPulseAmplitude = 3500
p_atrPulseWidth = 10
p_ventPulseWidth = 10
p_atrThreshold = 2640
p_ventThreshold = 2640
p_arpDelay = 250
p_vrpDelay = 320
p_fixedAVDelay = 150
p_rateModulation = 0
p_modulationSenstivity = 8
k_sampleRate = 100

STANDBY

%Waiting for communication packet

SET_PARAM

p_pacingMode = rxdata(3)
p_lowerRateLimit = rxdata(4)
p_upperRateLimit = rxdata(5)
p_atrPulseAmplitude = typecast(rxdata(6:7), ‘uint16’)
p_ventPulseAmplitude = typecast(rxdata(8:9), ‘uint16’)
p_atrPulseWidth = rxdata(10)
p_ventPulseWidth = rxdata(11)
p_atrThreshold = typecast(rxdata(12:13), ‘uint16’)
p_ventThreshold = typecast(rxdata(14:15), ‘uint16’)
p_arpDelay = typecast(rxdata(16:17), ‘uint16’)
p_vrpDelay = typecast(rxdata(18:19), ‘uint16’)
p_fixedAVDelay = typecast(rxdata(20:21), ‘uint16’)
p_rateModulation = rxdata(22)
p_modulationSenstivity = rxdata(23)

ECHO_PARAM

k_echoData = rxdata(2)
send_data()

ECHO_EGRAM

k_echoData = rxdata(2)
send_data()

- k_echoData informs the send_data() function whether to send params or egram data back to the DCM
- ECHO_EGRAM block is periodically entered at a rate given by k_sampleRate
- ECHO_EGRAM continues to sample until a communication packet is received of which the second byte is 0x62

COM_OUT

- This subsystem represents the global function send_data()

Inputs: p_pacingMode, p_lowerRateLimit, p_upperRateLimit, p_atrPulseAmplitude, p_ventPulseAmplitude, p_atrPulseWidth, p_ventPulseWidth, p_atrThreshold, p_ventThreshold, p_arpDelay, p_vrpDelay, p_fixedAVDelay, p_rateModulation, p_modulationSenstivity, m_atrSignal, m_ventSignal, k_echoData

Outputs: k_transmitData

NOTE: Executes every time the chart is initialized, and outputs are initialized every time the chart wakes up.

- f() function enables this subsystem to act as a global function and triggers this block each time send_data() is called
- Two multiplexers (mux) are needed to combine the data into a packet of 21 bytes

First multiplexer (mux): Packs programmable variables, received from COM_IN subsystem

Second multiplexer (mux): Packs egram data, received from the Hardware_Hiding subsystem. Addition to the m_atrSignal and m_ventSignal (each of type double - 8 bytes each) variables, five zeros (uint8) are used to fill the entire packet since the sending packets length must be 21 bytes

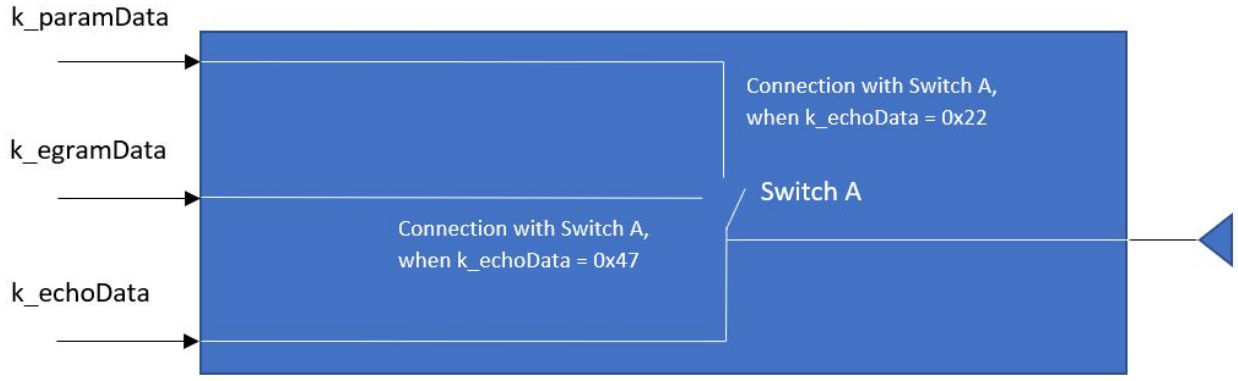

Figure 2: TRANSMIT_DATA block portraying switch-case mechanism

denotes serial transmit port which sends data to the DCM

- The k_transmitData matrix [uint8] delivers the 21 bytes to the serial transmit port
- k_egramData is sent to the DCM every k_sampleRate [ms]; however, k_paramData is only sent once per DCM-request

NOTE: Byte Pack block is used to convert a uint16 data type to two uint8 data

Table 2: State Transition Table of COM_OUT subsystem

Current State	Event	Next State
INIT	k_echoData == 0x22	TRANSMIT_PARAMS
	k_echoData == 0x47	TRANSMIT_EGRAM
	NOT(k_echoData == 0x22 or 0x47)	---
TRANSMIT_PARAMS	---	INIT
TRANSMIT_EGRAM	---	INIT

State Configurations

INIT

%Check which data to echo to DCM

TRANSMIT_PARAMS

%Set the transmit data to the programmable parameters
k_transmitData = k_paramData

TRANSMIT_EGRAM

%Set the transmit data to the electrogram data
k_transmitData = k_egramData

SYS_IN

- This subsystem manipulates the input variables for the next subsystem (PACEMAKER_FSM)

Inputs: p_pacingMode, p_lowerRateLimit, p_upperRateLimit, p_atrPulseAmplitude, p_ventPulseAmplitude, p_atrPulseWidth, p_ventPulseWidth, p_atrThreshold, p_ventThreshold, p_arpDelay, p_vrpDelay, p_fixedAVDelay, p_rateModulation, p_modulationSenstivity

Outputs: k_pacingMode, k_atrPulseAmplitude, k_ventPulseAmplitude, k_atrPaceDelay, k_atrPulseWidth, k_ventPaceDelay, k_ventPulseWidth, k_atrThreshold, k_ventThreshold, k_arpDelay, k_vrpDelay, k_pacingAVDelay, k_atrialEscapeInterval

The following table indicates how the SYS_IN subsystems transforms the programmable parameters into useful parameters for the PACEMAKER_FSM:

Table 3: Shows manipulation of SYS_IN inputs

Input	**Modification**	**Output**
p_pacingMode	-	k_pacingMode
p_atrPulseAmplitude [mV]	$p\_atrPulseAmplitude \div 50$ Divided by 50 to get the Atrium duty cycle	k_atrPulseAmplitude
p_ventPulseAmpitude [mV]	p_ventPulseAmpitude ÷ 50 (divided by 50 to get the ventricle duty cycle)	k_ventPulseAmplitude
p_lowerRateLimit [BPM]	[{ (1 ÷ p_lowerRateLimit) X 60000 } - p_atrPulseWidth] Finding the amount of msec/beat - by inverting the BPM to get min/beat, and multiplying by msec/min. Finally, subtracting p_atrPulseWidth to get k_atrPaceDelay	k_atrPaceDelay This delay is equal to the total period between beats minus atrium pulse width
p_atrPulseWidth	-	k_atrPulseWidth

Input	Modification	Output
p_lowerRateLimit [BPM]	[{ (1 ÷ p_lowerRateLimit) X 60000 } - p_ventPulseWidth] Finding the amount of msec/beat - by inverting the BPM to get min/beat, and multiplying by msec/min. Finally, subtracting p_ventPulseWidth to get k_ventPaceDelay	k_ventPaceDelay This delay is equal to the total period between beats minus the ventricle pulse width
p_ventPulseWidth	-	k_ventPulseWidth
p_atrThreshold [mV]	[p_atrThreshold ÷ 33] Divided by 33 (~3.3V) to get the atrium duty cycle	k_atrThreshold
p_ventThreshold [mV]	[p_ventThreshold ÷ 33] Divided by 33 (~3.3V) to get the ventricle duty cycle	k_ventThreshold
p_arpDelay	-	k_arpDelay
p_vrpDelay	-	k_vrpDelay
p_fixedAVDelay	[p_fixedAVDelay - p_atrPulseWidth]	k_pacingAVDelay
p_lowerRateLimit	[{ (1 ÷ p_lowerRateLimit) X 60000 } - k_pacingAVDelay - k_ventPulseWidth]	k_atrialEspaceInterval This delay is equal to the total period between beats minus the pacingAVDelay and minus the ventricle-pulse-width

PACEMAKER_FSM

- This subsystem includes implemented pulsating modes and controls the hardware circuitry

Inputs: k_pacingMode, k_atrPulseAmplitude, k_ventPulseAmplitude, k_atrPaceDelay, k_atrPulseWidth, k_ventPaceDelay, k_ventPulseWidth, k_atrThreshold, k_ventThreshold, k_arpDelay, k_vrpDelay, k_pacingAVDelay, k_atrialEscapeInterval, m_pushButton, m_atrCMPDetect, m_ventCMPDetect

Outputs: c_atrPaceCtrl, c_ventPaceCtrl, c_pacingRefPWM, c_paceChargeCtrl, c_paceGNDCtrl, c_atrGNDCtrl, c_ventGNDCtrl, c_zAtrCTRL, c_zVentCtrl, c_frontEndCtrl, c_atrCMPRefPWM, c_ventCMPRefPWM, c_blueLED, c_redLED

Table 4: State Transition Table of PACEMAKER_FSM subsystem

Current State	**Event**			**Next State**
INIT	k_pacingMode == 0 or 2 or 4			6. A_CHARGING
	k_pacingMode == 1 or 3			2. V_CHARGING
	NOT(k_pacingMode == 0 or 1 or 2 or 3 or 4 or 5)			---
A_CHARGING	m_pushButton == true			9. A_CHARGING
	m_pushButton == false			---
	k_pacingMode == 0	After(k_atrPaceDelay)		7. A_PACING
		NOT(After(k_atrPaceDelay))		---
	k_pacingMode == 1			13. V_CHARGING
	k_pacingMode == 2	m_atrCMPDetect == true	After(k_arpDelay)	9. A_CHARGING
			NOT(After(k_arpDelay))	---
		m_atrCMPDetect == false		---
	k_pacingMode == 3			13. V_CHARGING
	k_pacingMode == 4		After(k_pacingAVDelay)	11. V_PACING
			NOT(After(k_pacingAVDelay))	---

Current State	Event			Next State
A_PACING	After(k_atrPulseWidth)			8. A_CHARGING
	NOT(After(k_atrPulseWidth))			---
V_CHARGING	m_pushButton == true			5. V_CHARGING
	m_pushButton == false			---
	k_pacingMode == 0			12. A_CHARGING
	k_pacingMode == 1	After(k_ventPaceDelay)		3. V_PACING
		NOT(After(k_ventPaceDelay))		---
	k_pacingMode == 2			12. A_CHARGING
	k_pacingMode == 3	m_ventCMPDetect == true	After(k_vrpDelay)	5. V_CHARGING
			NOT(After(k_vrpDelay))	---
		m_ventCMPDetect == false		---
	k_pacingMode == 4		After(k_atrialEscapeInterval)	10. A_PACING
			NOT(After(k_atrialEscapeInterval))	---
V_PACING	After(k_ventPulseWidth)			4. V_CHARGING
	NOT(After(k_ventPulseWidth))			---

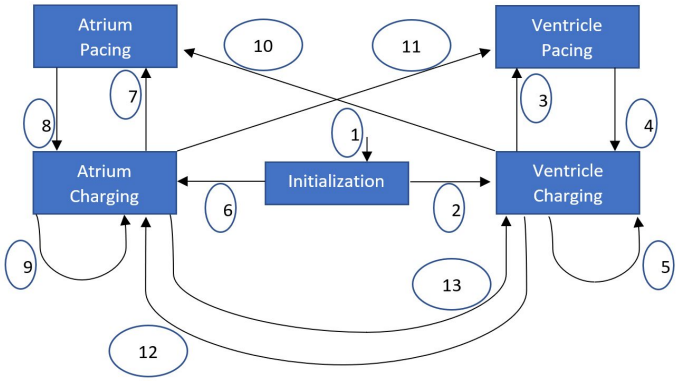

Figure 3: Figure of Pacemaker FSM charts and their transition from one state to another

State Configurations

Initialization/INIT

- All switches are open and there is no current flow in the system
- Subsequently, the pacemaker goes to the appropriate atrium or ventricle charging states depending on which mode is being implemented

Ventricle Charging/V_CHARGING

- Charging the C22 capacitor and reversing the current flow through the ventricle in order to create net zero charge
- This is done to ensure that the patient is not directly connected to the voltage supply by setting ventPaceCtrl to FALSE, which opens the switch to prevent current flow from C22 to C21 capacitor

Ventricular Pacing/V_PACING

- Allowing current to flow from C22 capacitor across the impedance of the ventricle to C21 capacitor to generate an artificial pace in the ventricle (turns **red** LED on)

Atrium Charging/A_CHARGING

- Charging the C22 capacitor and reversing the current flow through the atrium in order to create net zero charge
- This is done to ensure that the patient is not directly connected to the voltage supply by setting atrPaceCtrl to FALSE, which opens the switch to prevent current flow from C22 to C21 capacitor

Atrium Pacing/A_PACING

- Allowing current to flow from C22 capacitor across the impedance of the atrium to C21 capacitor to generate an artificial pace in the ventricle (turns **blue** LED on)

NOTE: For elaborated details on what is included in the charts and how the charts control the hardware circuitry, please read the appendix at the end of this document

Stateflow of DOO Mode

A_PACING □ A_CHARGING□ V_PACING □ V_CHARGING□
A_PACING □ A_CHARGING□ V_PACING □ …

NOTE: For all the other mode implementation, please read the documentation of Assignment #1

HARDWARE_HIDING

- This subsystem contains blocks for reading and writing from the pins of the FRDM-K64F micro-controller/shield
- This subsystem enables simple modification of the hardware in case of a hardware malfunction

Inputs: c_atrPaceCtrl, c_ventPaceCtrl, c_pacingRefPWM, c_paceChargeCtrl, c_paceGNDCtrl, c_atrGNDCtrl, c_ventGNDCtrl, c_zAtrCTRL, c_zVentCtrl, c_frontEndCtrl, c_atrCMPRefPWM, c_ventCMPRefPWM, c_blueLED, c_redLED

Outputs: m_atrCMPDetect, m_ventCMPDetect, m_pushButton, m_atrSignal, m_ventSignal

The following table illustrates how the pins on the microcontroller are controlled by the input variables:

Table 5: Behavior of microcontroller pins

Controlled Variables	**Write/Read**	**Pin**
c_atrPaceCtrl	Digital Write	D8
c_ventPaceCtrl	Digital Write	D9
c_pacingRefPWM	PWM Output	D5
c_paceChargeCtrl	Digital Write	D2
c_paceGNDCtrl	Digital Write	D10
c_atrGNDCtrl	Digital Write	D11
c_ventGNDCtrl	Digital Write	D12
c_zAtrCtrl	Digital Write	D4
c_zVentCtrl	Digital Write	D7
c_frontEndCtrk	Digital Write	D13
c_atrCMPRefPWM	PWM Output	D6
c_ventCMPRefPWM	PWM Output	D3
c_blueLED	Digital Write	BLUE_LED
c_redLED	Digital Write	RED_LED
m_atrCMPDetect	Digital Read	D0

m_ventCMPDetect	Digital Read	D1
Controlled Variables	**Write/Read**	**Pin**
m_pushButton	Digital Read	Push Button (Physical Button)
m_atrSignal	Analog Input	A0 Before the value is registered in m_atrSignal, it is first multiplied by 3.3 to map the result within the range of (0 - 3.3V)
m_ventSignal	Analog Input	A1 Before the value is registered in m_ventSignal, it is first multiplied by 3.3 to map the result within the range of (0 - 3.3V)

Rate Modulation

NOTE: Functioning of accelerometer was tested in a different Simulink model.

- The accelerometer block senses forces from -4g to +4g
- The FRDM-K64F accelerometer experiences approx. 1g at rest, as depicted below

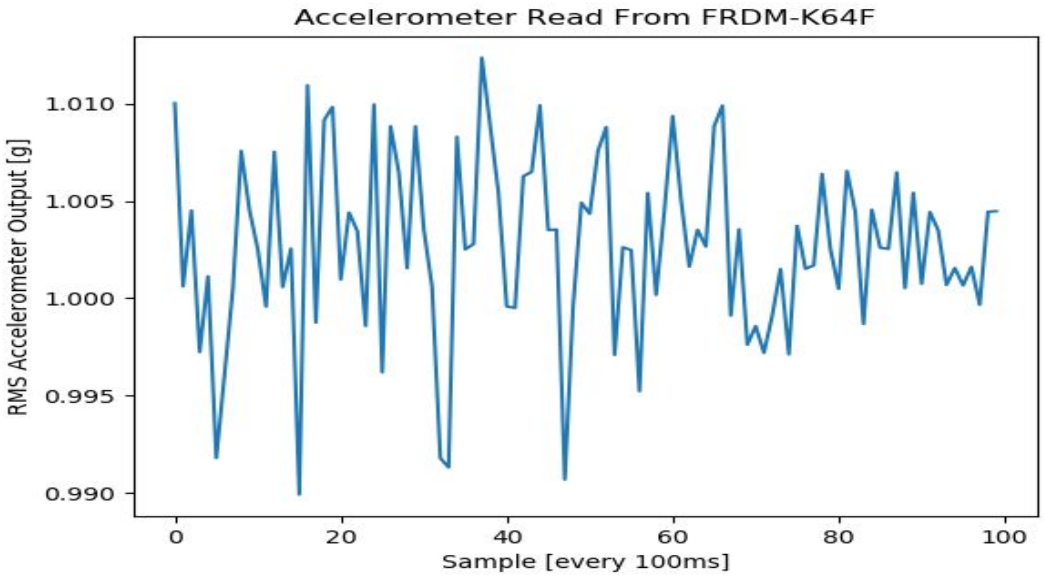

Figure 4: Accelerometer readings in idle state

- Since the accelerometer is always experiencing a force of approx. 1g, the threshold for rate modulation, k_activityThreshold is chosen to be 1.25g

The following figure shows an accelerometer vector reading being decomposed into three components:

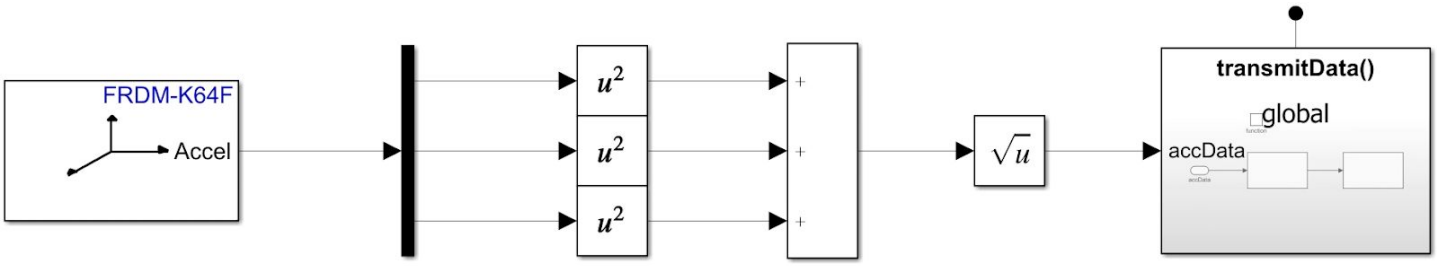

Figure 5: Data manipulation of an accelerometer

- After the data is passed through a demux, the useful acceleration value is estimated using RMS (root-mean-square) function on all three vectors
- The RMS value is used to measure the current activity of the patient, namely k_currentActivity

The following table provides state transition expressions for the rate modulation functionality:

Table 6: Function table for Rate Modulation

<table>
<tr><th>Current State</th><th colspan="4">Event</th><th>Next State</th></tr>
<tr><td rowspan="4">MIN_BPM</td><td colspan="4">p_rateModulation == e_off</td><td>---</td></tr>
<tr><td rowspan="3">p_rate Modu lation == e_on</td><td colspan="3">k_currentActivity < p_activityThreshold</td><td>---</td></tr>
<tr><td rowspan="2">k_currentActivity ≥ p_activityThresho l d</td><td colspan="2">After(p_reactionDelay)</td><td>UP_BPM</td></tr>
<tr><td colspan="2">NOT(After(p_reactionDelay))</td><td>---</td></tr>
<tr><td rowspan="6">UP_BPM</td><td colspan="4">p_rateModulation == e_off</td><td>DOWN_BPM</td></tr>
<tr><td rowspan="5">p_rate Modu lation == e_on</td><td rowspan="2">k_current activity < p_activityThresho l d</td><td colspan="2">After(p_reactionDelay)</td><td>SAME_BPM</td></tr>
<tr><td colspan="2">NOT(After(p_reactionDelay))</td><td>---</td></tr>
<tr><td rowspan="3">k_currentActivity ≥ p_activityThresho l d</td><td rowspan="2">After(p_reac tionDelay)</td><td>k_BPM + p_modulationSensitivit y ≤ p_upperRateLimit</td><td>UP_BPM</td></tr>
<tr><td>k_BPM + p_modulationSensitivt y > p_upperRateLimit</td><td>MAX_BPM</td></tr>
<tr><td colspan="2">NOT(After(p_reactionDelay))</td><td>---</td></tr>
<tr><td rowspan="5">SAME_BPM</td><td colspan="4">p_rateModulation == e_off</td><td>DOWN_BPM</td></tr>
<tr><td colspan="2" rowspan="4">p_rateModulation == e_on</td><td rowspan="2">k_currentA ctivity < p_activityT hreshold</td><td>After(p_sameBPMDela y)</td><td>DOWN_BPM</td></tr>
<tr><td>NOT(After(p_sameBP MDelay))</td><td>---</td></tr>
<tr><td rowspan="2">k_currentA ctivity ≥ p_activityT hreshold</td><td>After(p_sameBPMDela y)</td><td>UP_BPM</td></tr>
<tr><td>NOT(After(p_sameBP MDelay))</td><td>---</td></tr>
</table>

<table>
<tr><th>Current State</th><th colspan="4">Event</th><th>Next State</th></tr>
<tr><td rowspan="8">DOWN_BPM</td><td rowspan="5">p_rateModulation == e_on</td><td rowspan="2">k_currentActivity ≥ p_activityThreshold</td><td colspan="2">After(p_recoveryDelay)</td><td>SAME_BPM</td></tr>
<tr><td colspan="2">NOT(After(p_recoveryDelay))</td><td>---</td></tr>
<tr><td rowspan="3">k_current activity < p_activityThreshold</td><td rowspan="2">After(p_recoveryDelay)</td><td>k_BPM - p_modulationSensitivity ≥ p_lowerRateLimit</td><td>DOWN_BPM</td></tr>
<tr><td>k_BPM - p_modulationSensitivity < p_lowerRateLimit</td><td>MIN_BPM</td></tr>
<tr><td colspan="2">NOT(After(p_recoveryDelay))</td><td>---</td></tr>
<tr><td rowspan="3">p_rateModulation == e_off</td><td rowspan="2">After(p_recoveryDelay)</td><td colspan="2">k_BPM - p_modulationSensitivity ≥ p_lowerRateLimit</td><td>DOWN_BPM</td></tr>
<tr><td colspan="2">k_BPM - p_modulationSensitivity < p_lowerRateLimit</td><td>MIN_BPM</td></tr>
<tr><td colspan="3">NOT(After(p_recoveryDelay))</td><td>---</td></tr>
<tr><td rowspan="3">MAX_BPM</td><td colspan="3">p_rateModulation == e_off</td><td>DOWN_BPM</td></tr>
<tr><td rowspan="2">p_rateModulation == e_on</td><td rowspan="2">k_currentActivity < p_activityThreshold</td><td>After(p_recoveryDelay)</td><td>DOWN_BPM</td></tr>
<tr><td>NOT(After(p_recoveryDelay)</td><td>---</td></tr>
<tr><td colspan="3">k_currentActivity ≥ p_activityThreshold</td><td>---</td></tr>
</table>

- If at any point, rate modulation is disabled, k_BPM will decrease by p_modulationSensitivty every p_recoveryDelay until k_BPM is equal to p_lowerRateLimit
- If rate modulation is enabled and the current activity is greater than the threshold, k_BPM will increase by p_modulationSensitivity every p_reactionDelay until it reaches p_upperRateLimit
- If rate modulation is enabled and the current activity is less than the threshold, the BPM will remain at a constant rate for p_sameBPMDelay
    - After p_sameBPMDelay, if the threshold is exceeded, k_BPM will increase by p_modulationSensitivity every p_reactionDelay until it reaches p_upperRateLimit
    - After p_sameBPMDelay, if the current activity remains less than the threshold, k_BPM will decrease by p_modulationSensitivty every p_recoveryDelay until k_BPM is equal to p_lowerRateLimit

Table 7: Parameters introduced for the implementation of rate modulation

Parameter	Data Type	Description
p_rateModulation	uint8	Indicates whether rate modulation is enabled or disabled
p_modulationSensitivity	uint8	Amount by which the BPM is increased or decreased each period
p_reactionDelay	uint16	Time to wait before comparing the patient's current activity to the threshold when increasing the BPM
p_recoveryDelay	uint16	Time to wait before comparing the patient's current activity to the threshold when decreasing the BPM
p_sameBPMDelay	uint16	Time to wait before comparing the patient's current activity to the threshold when at a constant BPM
p_activityThreshold	double	The amount of activity which must be exceeded in order to initiate rate modulation, 1.25g
k_currentActivity	double	The measure of the patient's current activity, which comes from the on-board accelerometer
k_BPM	uint8	Represents the current BPM rate

State Configurations

MIN_BPM

k_BPM = p_lowerRateLimit

UP_BPM

%New BPM is current BPM plus modulation amount
k_BPM = k_BPM + p_modulationSensitivty

SAME_BPM

%No change in BPM

DOWN_BPM

%New BPM is current BPM minus modulation amount
k_BPM = k_BPM - p_modulationSensitivity

MAX_BPM

k_bpm = p_upperRateLimit

APPENDIX

Serial Inputs

Table A-1

Variables	Type	Description
rxdata	uint8	Variable which stores the received data (through the serial communication)
status	uint8	Variable used to detect successful transmission of serial data

Programmable Parameters

Table A-2

Parameters	Type	Description
p_pacingMode	uint8	Specifies the mode based on the assigned value 0 - AOO, 1 - VOO, 2 - AAI, 3 - VVI, 4 - DOO
p_lowerRateLimit	uint8	Specifies the minimum attainable value of BPM
p_upperRateLimit	uint8	Specifies the maximum attainable value of BPM
p_atrPulseAmplitude	uint16	Amplitude of the atrium pulse calculated for a certain frequency and duty cycle. Calculates the amplitude for the atrium chamber pulse by controlling the duty cycle. e.g. the divide block is used to calculate duty cycle: (3000mV/5000mV)*100% = 60%
p_ventPulseAmplitude	uint16	Amplitude of the ventricle pulse calculated for a certain frequency and duty cycle.Calculates the amplitude for the ventricle chamber pulse by controlling the duty cycle. e.g. the divide block is used to calculate duty cycle: (3000mV/5000mV)*100% = 60%
p_atrPulseWidth	uint8	Time during which the atrium chamber is paced
p_ventPulseWidth	uint8	Time during which the ventricle chamber is paced
p_atrThreshold	uint16	Used to sense activity in the atrium chamber. Value which sufficiently provides evidence of a natural pulse. Currently, set at 80% duty cycle

Parameters	Type	Description
p_ventThreshold	uint16	Used to sense activity in the ventricle chamber. Value which sufficiently provides evidence of a natural pulse. Currently, set at 80% duty cycle
p_arpDelay	uint16	Defines the refractory period in the atrium chamber
p_vrpDelay	uint16	Defines the refractory period in the ventricle chamber
p_fixedAVDelay	uint16	Specifies the duration between an atrium and ventricle pace
p_rateModulation	uint8	Specifies whether rate modulation is enabled or disabled
p_modulationSenstivity	uint8	Specifies the amount by which the rate changes per event

Local Variables

Table A-3

Parameters	Type	Description
k_sampleRate	uint8	COM_IN subsystem - Specifies the period by which the egram data is sampled
k_echoData	uint8	COM_IN/COM_OUT subsystems - second byte of the received serial data. Represents which data string needs to be sent back to the DCM.
k_paramData	21 bytes	COM_OUT subsystem - collection of all the programmable parameters which is collated to be sent back to the DCM.
k_egramData	21 bytes	COM_OUT subsystem - contains voltage readings of the m_atrSigna and m_ventSignal.
k_transmitData	21 bytes	COM_OUT subsystem - holds a string of data which is either k_paramData or k_egramData.

Monitored Variables

Table A-4

Parameters	Type	Description
m_atrSignal	Double / 8-bytes	Outputs the analog signal of the atrium chamber - represents whats is happening in the heart in real-time. Used to collect information, in mV, about atrial electrogram.
m_ventSignal	Double / 8-bytes	Outputs the analog signal of the ventricle chamber - represents whats is happening in the heart in real-time. Used to collect information, in mV, about ventricular electrogram.
m_pushButton	SW2	A physical push button used to delay an artificial beat
m_atrCMPDetect	Input (∵, pin's value is read from the microcontroller to observe natural pacing)	Used in the sensing circuitry. When a signal higher than threshold voltage is detected in the atrium chamber, pin D0 is set to HIGH, and OFF otherwise (includes 5mV hysteresis).
m_ventCMPDetect	Input (∵, pin's value is read from the microcontroller to observe natural pacing)	Used in the sensing circuitry. When a signal higher than threshold voltage is detected in the ventricular chamber, pin D1 is set to HIGH, and OFF otherwise (includes 5mV hysteresis).

Controlled Variables

Table A-5

Parameters	Type	Description
c_atrPaceCtrl	Boolean	When this controllable switch is closed, the current in the primary capacitor can drain and potentially, induce a contraction in atrium. Remember to turn OFF c_paceChargeCtrl to avoid connecting the atrium directly with the PWM signal.
c_ventPaceCtrl	Boolean	When this controllable switch is closed, the current in the primary capacitor can drain and potentially, induce a contraction in ventricle. Remember to turn OFF c_paceChargeCtrl to avoid connecting the atrium directly with the PWM signal.
c_pacingRefPWM	uint8	Controls the charge of the primary capacitor with duty cycle controlling the capacitor's voltage
c_paceChargeCtrl	Boolean	Required to charge the primary capacitor for pacing capabilities. To achieve this, set pin D2 to HIGH. Make sure to turn OFF pin D8 or D9, before setting pin D2 to HIGH.
c_paceGNDCtrl	Boolean	This is the final link in the chain of inducing a heartbeat. If this variable is left ON, along with either of the c_...PaceCtrl, then a pulse is created in the respective chamber.
c_atrGNDCtrl	Boolean	Used to discharge excess charge buildup at the tip of the electrode in the atrium.
c_ventGNDCtrl	Boolean	Used to discharge excess charge buildup at the tip of the electrode in the ventricle.

Parameters	Type	Description
c_zAtrCtrl	Boolean	Used to monitor the impedance of the atrium chamber. Impedance can be measured at Z_signal pin A2.
c_zVentCtrl	Boolean	Used to monitor the impedance of the ventricle chamber. Impedance can be measured at Z_signal pin A2.
c_frontEndCtrl	Boolean	Deployed in activating the sensing modes. This variable stays OFF unless sensing of the heartbeat is required.
c_atrCMPRefPWM	uint8	Used to set a threshold voltage for sensing in the atrium chamber.
c_VentCMPRefPWM	uint8	Used to set a threshold voltage for sensing in the ventricle chamber.
c_blueLED	Boolean	Used to flash blue LED when atrium is paced artificially.
c_redLED	Boolean	Used to flash red LED when ventricle is paced artificially.

PACEMAKER_FSM State Configurations

INIT

Table A-6

c_frontEndCtrl = true;	Enables the sensing circuitry. In this assignment, the sensing mode is activated, however this parameter can be set to false as well.
c_pacingRefPWM = 0;	Turn OFF voltage supply
c_paceChargeCtrl = false;	OPEN connection between the voltage supply and C22 capacitor. Stops charging of C22 capacitor.
c_atrPaceCtrl = false;	OPEN connection between C22 capacitor and the atrium chamber. Halts any pacing in the atrium chamber.
c_ventPaceCtrl = false;	OPEN connection between C22 capacitor and the ventricle chamber. Halts any pacing in the ventricle chamber.
c_atrGNDCtrl = false;	OPEN connection between the atrium and GND
c_ventGNDCtrl = false;	OPEN connection between the ventricle and GND
c_paceGNDCtrl = false;	No discharging of C21 capacitor
c_blueLED = false;	Built-in LED is off (used to indicate an event of artificial pacing in the atrium chamber)
c_redLED = false;	Built-in LED is off (used to indicate an event of artificial pacing in the ventricle chamber)

V_CHARGING

Table A-7

c_ventCMPRefPWM = p_ventThreshold;	Not needed in this mode (useful during sensing)
c_pacingRefPWM = p_ventPulseAmplitude;	Assigning the PWM reference value to a programmable variable to be able to control it with GUI
c_paceGNDCtrl = true;	Discharge Step #1 – to drain the C21 capacitor (blocking capacitor)
c_ventPaceCtrl = false;	Discharge Step #2 – blocks current to go back to C22 capacitor
c_zAtrCtrl = false;	No measurement of impedance in the leads or atrium muscles
c_zVentCtrl = false;	No measurement of impedance in the leads or ventricular muscles
c_atrPaceCtrl = false;	Turns off atrium response in this mode
c_atrGNDCtrl = false;	Turns OFF atrium GND connections
c_ventGNDCtrl = true;	Discharge Step #3 – current from C21 flows to the GND
c_paceChargeCtrl = true;	Charge Step #1 – since C22 is only connected to the voltage supply, it can independently charge the C22 capacitor
c_redLED = false;	Built-in LED is off (used to indicate an event of artificial pacing in the ventricle chamber)

V_PACING

Table A-8

c_paceChargeCtrl = false;	Now, we cut off the power supply to the C22 capacitor
c_paceGNDCtrl = true;	Allows a direct connection between capacitor C22 and C21
c_atrPaceCtrl = false;	No involvement of atrium modes
c_atrGNDCtrl = false;	Limits the functionality available only for ventricle chamber
c_zAtrCtrl = false;	Not involving impedance as a factor in this case
c_zVentCtrl = false;	Not involving impedance from any chambers
c_ventGNDCtrl = false;	Makes sure that the current does not flow back to the GND terminal
c_ventPaceCtrl = true;	Finally, switch activation to let the current flow from C22 to C21 capacitor
c_redLED = true;	Built-in LED is ON (used to indicate an event of artificial pacing in the ventricle chamber)

A_CHARGING

Table A-9

c_atrCMPRefPWM = p_atrThreshold;	Not needed in this mode (useful during sensing)
c_pacingRefPWM = p_atrPulseAmplitude;	Assigning the PWM reference value to a programmable variable to be able to control it with GUI
c_paceGNDCtrl = true;	Discharge Step #1 – to drain the C21 capacitor (blocking capacitor)
c_atrPaceCtrl = false;	Discharge Step #2 – blocks current to go back to C22 capacitor
c_zAtrCtrl = false;	No measurement of impedance in the leads or atrium muscles
c_zVentCtrl = false;	No measurement of impedance in the leads or ventricular muscles
c_ventPaceCtrl = false;	Turns off ventricular response in this mode
c_ventGNDCtrl = false;	Turns OFF ventricular GND connections
c_atrGNDCtrl = true;	Discharge Step #3 – current from C21 flows to the GND
c_paceChargeCtrl = true;	Charge Step #1 – since C22 is only connected to the voltage supply, it can independently charge the C22 capacitor
c_blueLED = false;	Built-in LED is off (used to indicate an event of artificial pacing in the atrium chamber)

A_PACING

Table A-10

c_paceChargeCtrl = false;	Now, we cut off the power supply to the C22 capacitor
c_paceGNDCtrl = true;	Allows a direct connection between capacitor C22 and C21
c_ventPaceCtrl = false;	No involvement of ventricular modes
c_ventGNDCtrl = false;	Limits the functionality available only for atrium chamber
c_zAtrCtrl = false;	Not involving impedance as a factor in this case
c_zVentCtrl = false;	Not involving impedance from any chambers
c_atrGNDCtrl = false;	Makes sure that the current does not flow back to the GND terminal
c_atrPaceCtrl = true;	Finally, switch activation to let the current flow from C22 to C21 capacitor
c_blueLED = true;	Built-in LED is ON (used to indicate an event of artificial pacing in the atrium chamber)