Electrocardiogram Device and Application Design Document

Version 1.5

# Document Revision History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Change summary** | **Reviewed By** |
| 1.0 | 17/11/2019 | Andrew Belcher | Device serial handler, device button handler, and application serial handler FSM/pseudocode completed (my 1/4th) of design group work. | Jamie Bateman |
| 1.1 | 21/11/2019 | Jamie Bateman | Add display handlers, heart rate reading module, and high-level diagram. Describe messaging protocol and a list of events that can be broadcast. Add descriptions to each module. | Najma Abdirahman |
| 1.2 | 21/11/2019 | Andrew Belcher | Add ACP diagram | Andrew Belcher |
| 1.3 | 21/11/2019 | Andrew Belcher | Fixed formatting for fsm/pseudocode |  |
| 1.4 | 21/11/2019 | Najma Abdirahman | Report generator, FRam, LED0 and LED1 handlers FSM/pseudocode |  |
| 1.5 | 22/11/2019 | Megan Jones | File Handler and Slider Handler added |  |

# Summary

This document denotes the design of the electrocardiogram device and companion application. The design specifies a breakdown of the system, the messaging system that shall be utilized to communicate between components, and the states of each individual component. This document shall be conformed to and edited wherever necessary throughout development to ensure that below correctly describes the functionality of the system.

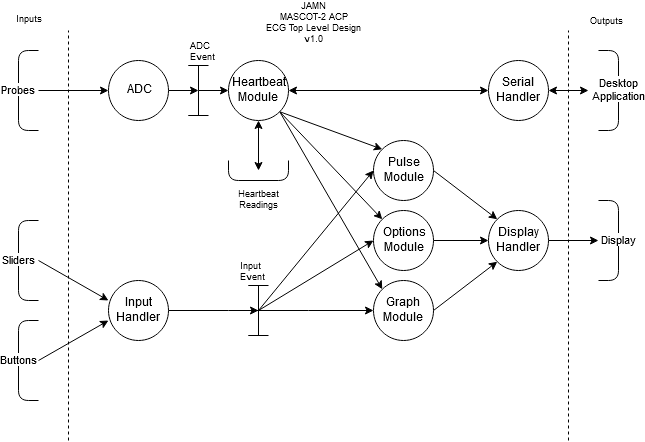
# ECG Device

## Definition

The ECG device is the handheld component of the system. This device will collect data from a patient, present that information, and be able to send the information to the companion application via a serial connection.

## High level design

**ACP Top Level Design**



The diagram above denotes several key modules for data flow, along with what data they shall receive and from where they shall source said data.

## Component messaging protocol

Each component will have access to a broadcast event function, and each function will have its own process event function. A global event queue is populated whenever a component wishes to broadcast a message. Messages are comprised of an event ID, sender ID and a data field. Some events require no data, such as a “start recording” event. Others, such as “heartrate reading” events require a value for parsing.

The following is a list of events that can be broadcast across the system.

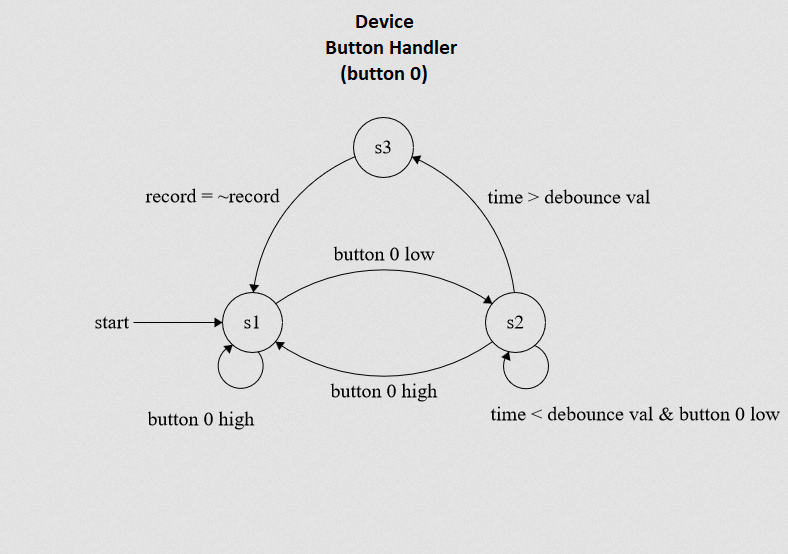
|  |  |  |  |
| --- | --- | --- | --- |
| Event ID | Event Description | Data | Sent by |
| 0 | Button0 pressed | n/a | Input module |
| 1 | Button0 held | n/a | Input module |
| 2 | Button1 pressed | n/a | Input module |
| 3 | Button1 held | n/a | Input module |
| 4 | Start heartrate recording | n/a | Display handler |
| 5 | Stop heartrate recording | n/a | Display handler |
| 6 | Heartrate data | Probe value readings | Heartrate reading module |
| 7 | Change display event | n/a | Display handler |
| 8 | Open options event | n/a | Display handler |
| 9 | Close options event | n/a | Display handler |

## Component Designs

### Button Handler

This module broadcasts button press/held messages to the event system, which are then read by display modules.

**FSM diagram**



**Pseudo Code**

Init button 0 on port

Port 0 isr

    if button 0 is low

        while denounce value < denounce max

        if button 0 is low

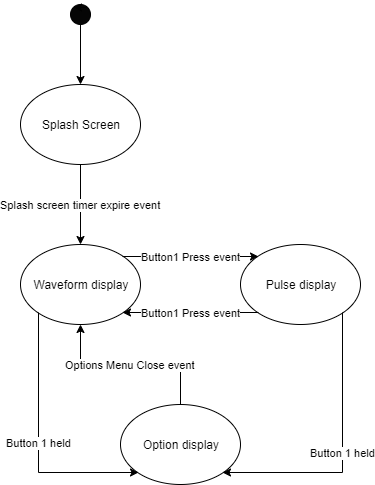
            while button 0 is low

            Record is = to not record

### Display handler

This module handles selecting which display module to enable. The selected display module will be able to output its own data to the display. Some display modules are inactive if they are not selected, i.e. the pulse display, as there is no need for it to listen to data constantly. The waveform display, though, must be constantly receiving heartbeat data, and so still operates even when not selected by the display handler.

**FSM diagram**



**Pseudo Code**

state = splashscreen

switch(state)

    case splashscreen:

        if first time

            clear display

            load splashscreen

        if timer expires

            state = waveform

    case waveform:

        if first time

            clear display

            load waveform table

        update waveform table

        if button1 pressed

            state = pulse

        if button1 held

            state = options

    case pulse:

        if first time

            clear display

            load pulse display

        update pulse value

        if button1 pressed

            state = waveform

        if button1 held

            state = options

    case options:

        if first time

            clear display

            load option buffer

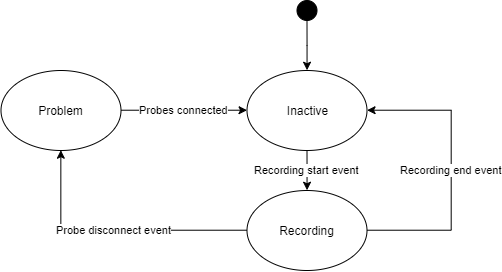
        if options exit event

            state = waveform

### Heartrate Reading Module

The heartrate reading module polls data from the probes when put into recording state. This data is broadcast in the form of an event on the event queue.

**FSM Diagram**



**Pseudo Code**

state = inactive

switch(state)

    case inactive:

        if first time

            broadcast stopped recording event

        if recording event

            state = active

    case active:

        if first time

            broadcast start recording event

        check probe state

        if probe state issue

            state = problem

        else

            broadcast new probe readings

    case problem:

        if first time

            broadcast problem event

        check probe state

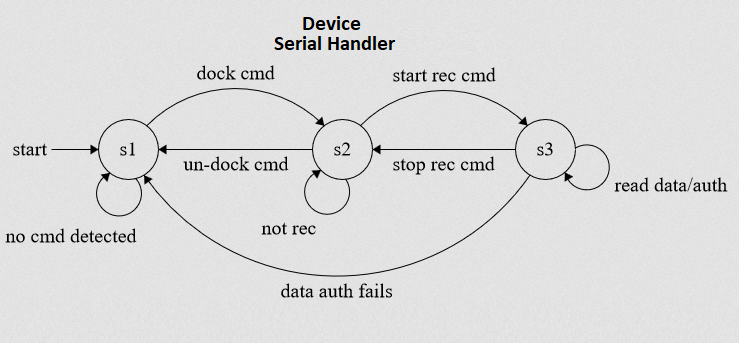
        if probe state is OK

            state = inactive

### Serial Module

The serial module controls receiving commands from the desktop app, along with sending data to the desktop app whilst heartbeat data events are being broadcasted.

**FSM Diagram**



**STATE KEY**

S1: undocked

S2: docked & not recording

S3: docked & recording

**Pseudo Code**

init serial

if command detected

    if command is corrupt

        if in docked state

            exit docked state

    if command is dock command

        enter dock state

    if command is record command

        enter record state

    else

        process alt command

if in docked state

    if in recording state

        if data received

            if data is valid

                store data

            else

                exit docked state

# Companion Application

## Definition

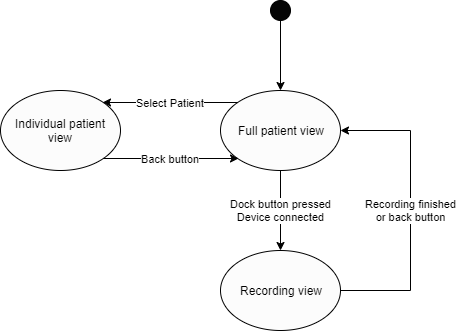
The companion application receives data over serial from the device. This data can be attributed to patients defined on the desktop app and can be saved into PDF format reports for later viewing. Large collections of saved data can also be analyzed to find trends over time, which can be represented in other PDF reports.

## Component Designs

### Display Module

The display module is responsible for switching between the different available views based on user input. User inputs cannot be handled in the same fashion as on the device; the available inputs change depending on the current display. This means that input handling isn’t necessary; buttons on the UI will automatically broadcast a specified event ID to represent a request to cause an action to happen, such as starting the recording on the ECG device or going to a specific view.

**FSM Diagram**



**Pseudo Code**

state = fullPatientView

targetPatient = undefined

switch(state)

    case fullPatientView:

        if first time

            load patient list

            display patients

        if device is docked

            show dock button

        if dock button pressed

            state = recordingView

        if patient selected

            targetPatient = selected patient

            state = individualPatientView

    case recordingView:

        if first time

            load recording view

        if back button pressed

            state = fullPatientView

        if recording finished event

            state = fullPatientView

    case individualPatientView:

        if first time:

            load targetPatient data

            show targetPatient data

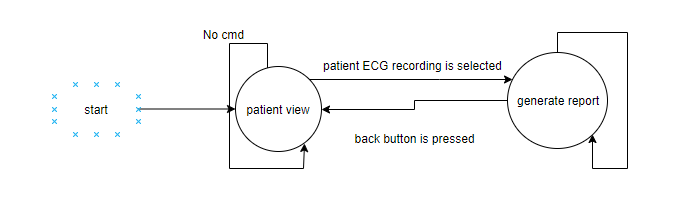
        if back button pressed:

            state = fullPatientView

## Report generator

This generates a PDF format report from a patient’s saved ECG recordings using the desktop app.

**FSM diagram**



**Pseudo Code**

state = patientview

switch(state)

case patient\_view:

if patient ECG is selected

generate report

if back button pressed

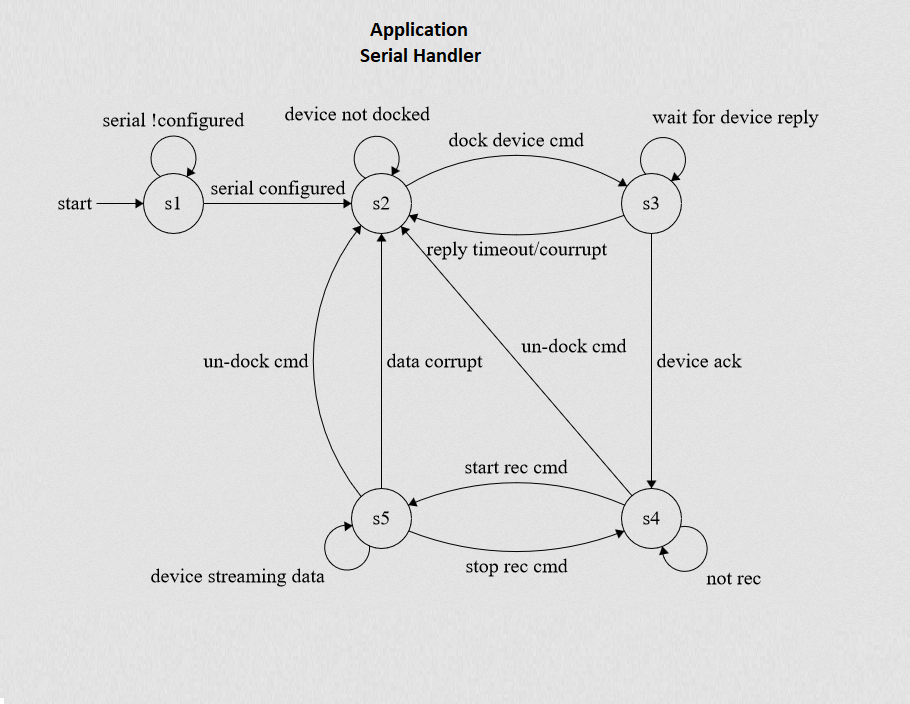
state = PatientView

end case

### Serial Handler

The serial handler is responsible for initiating docked mode with the device, sending commands to the device to change onboard configuration such as the time, and for receiving heartrate data for interpretation by other modules.

**FSM Diagram**



**STATE KEY**

S1: undocked

S2: docked & not recording

S3: docked & recording

**Pseudo Code**

if serial configured

    Init serial

if docking device

    Send dock command

    while device hasn’t replied

        Timeout, return

    if device reply

        if device reply is corrupt

            Return

        if device reply ack

            Device docked

if device is docked

    if not recording

        if undock cmd

            Device is not docked

    if start recording and not recording

        Send record cmd

    if stop recording & recording

        Send stop record cmd

    while device recording

        if data is ok

            Store data

        else

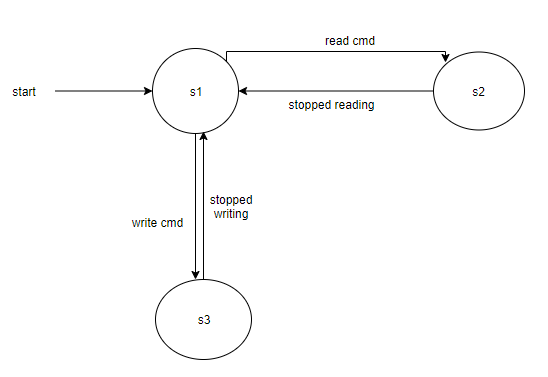
            Device is not docked

                return

## FRam

This reads data from FRam and writs data to FRam.

**FSM diagram**



**Pseudo Code**

#define FRAM\_START\_ADDR

#ifdef FRAM\_END\_ADDR

if command is read command

read data from FRam

return data

else if command is write command

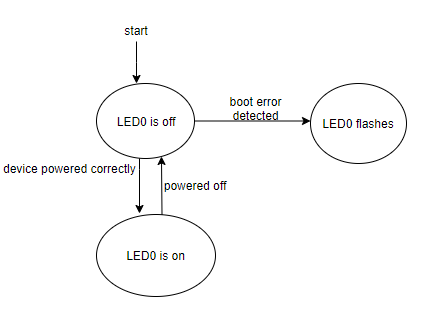
write to fRam

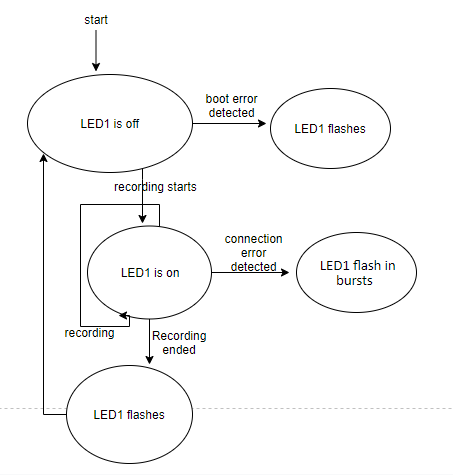
end if

## LED0 and LED1

This initiates the device start up, also allows the user to identify errors and shows whether device is in recording mode or not.

**FSM diagram**





**Pseudo Code**

if device is off

if device powered correctly

turn LED0 on

if boot error detected

flash LED0 & LED1

if command is record command

turn LED1 on

if connection error detected

flash LED1 in bursts

if recording ended

flash LED1

else if device powered off

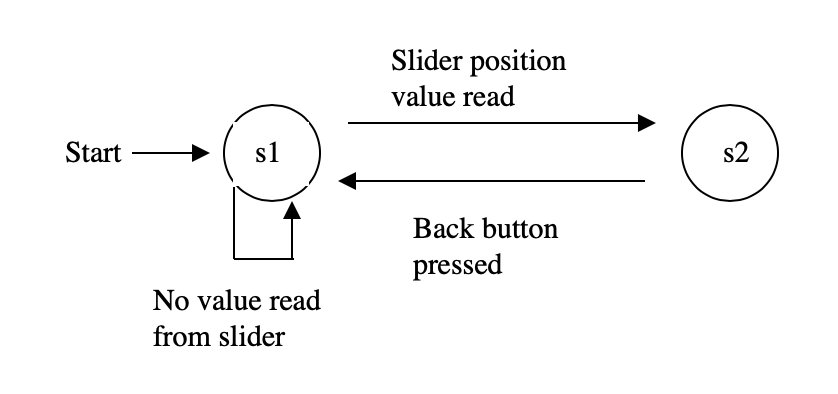
turn off both LEDs

end if

### Slider Handler

This module reads in a value from the device sliders and selects or adjusts a selected option appropriately.

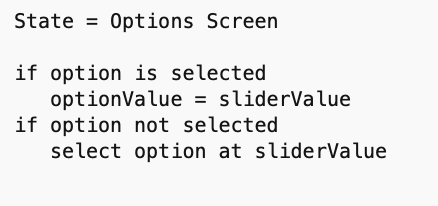
**FSM diagram**

****

s1 = Options Screen

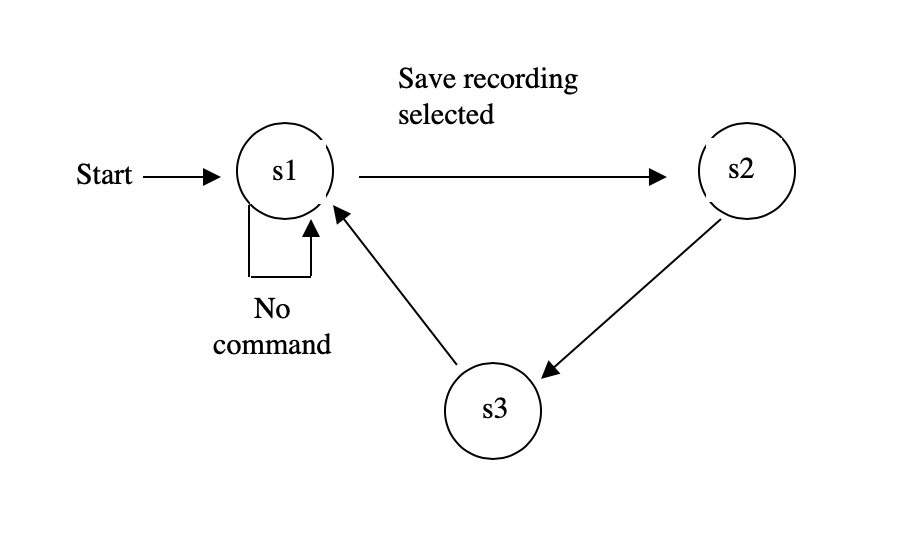
s2 = Option or option value selected

**Pseudo Code**



### File handler

**FSM diagram**



s1 = Patient View

s2 = record file generated from read in serial packets

s3 = record file saved

**Pseudo Code**

A screenshot of a cell phone

Description automatically generated