Anneysha Sarkar
COMP2300: Computer Organisation and Program Execution (2023)

Patient Monitor Display on Micro:bit Using ARM Assembly

Overview

My project makes an attempt to emulate some of the audio and visual features of a Patient Monitor[1], coded using ARM Assembly on a micro:bit. Currently, it has the following features:

- shows heart when there is a heartbeat
- un-shows heart when there is no heartbeat

A Patient Monitor displays the changing state of a patient through its audio and visual peripherals. My project changes its audio and visual outputs based on button inputs.

Design

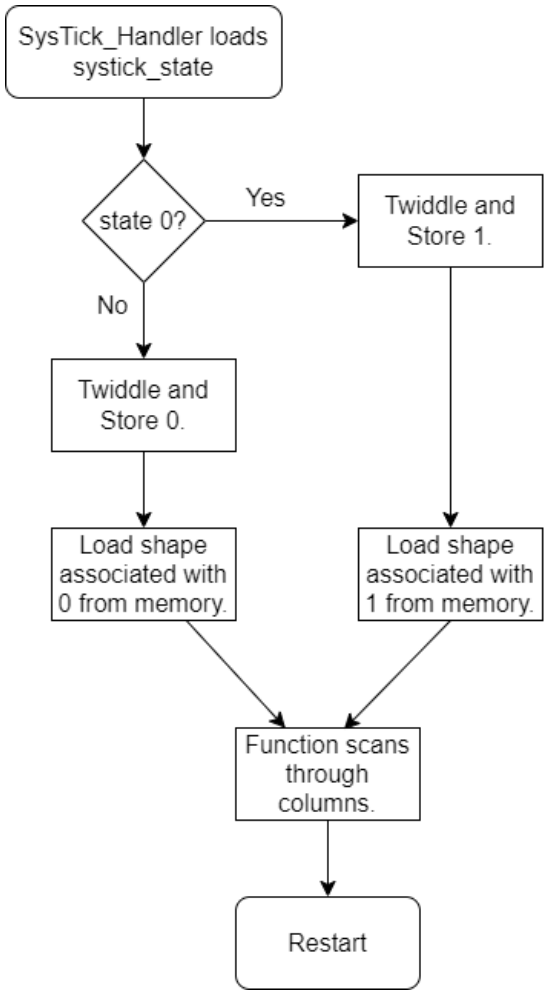
My project implements Pulse Width Modulation (PWM) and controls per-row/column LED screen brightness. Buttons have been configured to receive input and to further demonstrate Pulse-Width Modulation and change of sound. The reset button is used to turn on and off the system. Press Button B to start. Pressing Button A exhibits PWM. Pressing Button B 'kills' the heart, and leads to a dead pulse line and sound. Pressing it again 'revives' the heart and brings back the sound beat.

Implementation

Memory was extensively and optimally used in the code. Shapes were stored and loaded from memory, which was used in scanning. Furthermore, variable states like **`brightness_mode`**, **`systick_state`**, **`audio_mode`**, and **`delay_times`** were also stored in memory which proved very useful for working with interrupts and handlers as they would follow load-twiddle-store to create effects in the display. Load-twiddle-store was also used to go through every row. I imagine this method would be very useful if there is too much data or not enough memory, however, since we have the luxury of enough memory, it made sense to work as much using memory as possible, eliminating any hardcoding.

There have been attempts to incorporate animation, with and without **`SysTick_Handler`**. Unfortunately, I was unable to correctly set a higher priority to the button interrupt than systick handler, as a result of which the initial heart shape never showed up despite pressing the buttons. Interestingly, the audio did turn on and off using the button interrupts, indicating that this may not have been an interrupt issue, but rather an LED configuration issue. However, I did not have enough time to pursue the bug further.

The following flowchart illustrates my attempted implementation of animation using systick.



Parts of my implementation can be found commented in **`src/main.S`** and **`scratch-code/systic-animation.S`**.

PWM in my code was implemented in a similar way, by toggling brightness_mode in memory and changing brightness accordingly. Please refer to the table for more information.

Functions and Handlers:

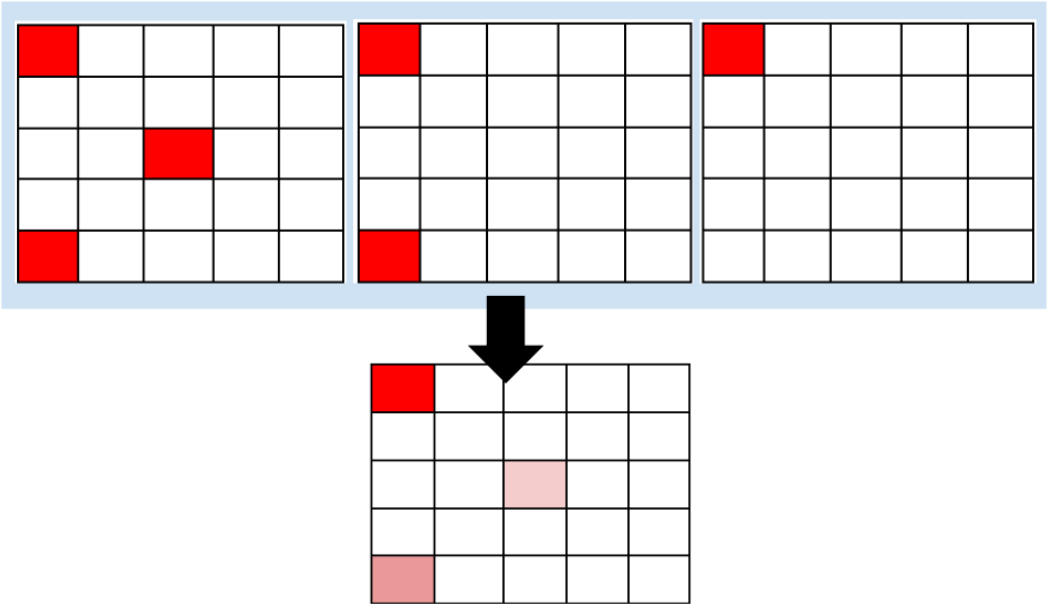
`delay_gate`	Toggles delay values by calling **`flip_delay`** if **`brightness_mode`** is 0
`flip_delay`	Toggles delay values in memory
`SysTick_Handler`	Modifies **`systick_state`** in memory such that animation images can run alternately
`GPIOTE_IRQHandler`	Checks which button was pressed by reading and comparing

	values in the **`GPIOTE_EVENTS`** registers.

Finally, the implementation of audio in the micro:bit. In order to change the amplitude of sound waves to make it seem closer to a heart beat, I increased the amplitude from 1 to 6. The idea behind creating different sounds was to model `((t >> 6) & 30) * t` to register memory. There was also some randomness in sounds involved. As soon as the sound seemed closer to a pulse or dead pulse, I let it be since highly-emphasised audio was not the focus for this project.

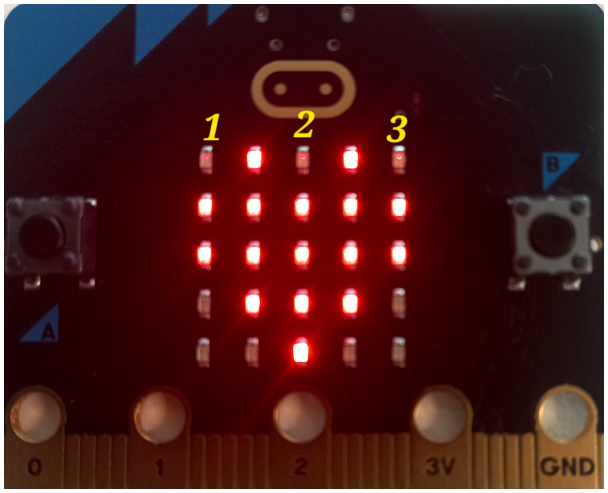
Analysis

I would have liked to implement per-led PWM, however, my approach of using toggled delay from memory does not seem very effective. This is probably because the **`delay_gate`** function works with rows or columns as a whole, rather than an LED at a time. While it might have been possible to implement per-LED brightness using this method, I think it might be more useful to explore a different approach. Another approach I wanted to try would be creating two frames: one frame has both LEDs on and the other frame has either LED on. The LED that is always on would appear brighter than the LED on half the time. This can also be extended to making more brightness levels. For example, if we wanted three brightness levels: 100%, 66%, and 33%, we have three frames combined into one as below.



Initially, one of my button interrupts was configured to turn the heart on and off. This feature has since been removed because the reset button on the micro:bit serves the same function and it doesn't add value to its existing features. Instead, the button is now used to toggle brightness along different rows to demonstrate the code can do a PWM across any row or column at a time, just by modifying memory.

In an earlier version of the code (May 18, got interrupt + shape working commit), during scanning, some of the LED lights were on and very dim although they were coded to be off. The numbered LEDs below demonstrate the problem.



There is a very short interval when the row pins for the previous row were on and the next rows were being read until the program looped and the next column and row pins were set. As observed, only the first row has this problem, since the next rows have fewer LED lights on, as a result, the ghost images get hidden. An easy fix was to clear row pins a second time in our `check` code. This actually clears the row pins and sets the right row pins.

Conclusion and Next Steps

This project was an opportunity to explore low-level programming with the help of a practical component like the micro:bit. I have learnt how different pins are configured, and how handlers work at the much lower level. Controlling data structures to create logic gates (the state data structures) was good to learn.

I will continue working on my light show during the winter. The first step would be to fully implement the dead pulse line animation when the button is pressed.

References

[1] Mueller, J., JD, & Jacks, D., MD (n.d.). How to Read a Patient Monitor at the Hospital. WikiHow. Retrieved May 30, 2023, from https://www.wikihow.com/Read-a-Hospital-Monitor