Interfacing to Matrix Keypad and Beeper

For this lab, you will add a matrix keypad and beeper to your Discovery board. You will sound a tone on the beeper while a key is pressed using the PWM output function of the timer module. Each key will sound a different tone.

Portions of this lab document come from Zhu, Lab_03_Keypad_4x4

As always, when building circuits and when working with active circuits, wear safety glasses.

Due: ***End of Semester, W2020. Friday 1 May***
Submit your code listing to the eConestoga dropbox by 11:59 pm

Submit lab report to dropbox including:
- Your commented c source code and header files

Demo when ready via zoom

The basic steps of the lab are:

1. Connect a piezo sounder (beeper) to a STM32 timer module output pin and write software to generate tones using the module

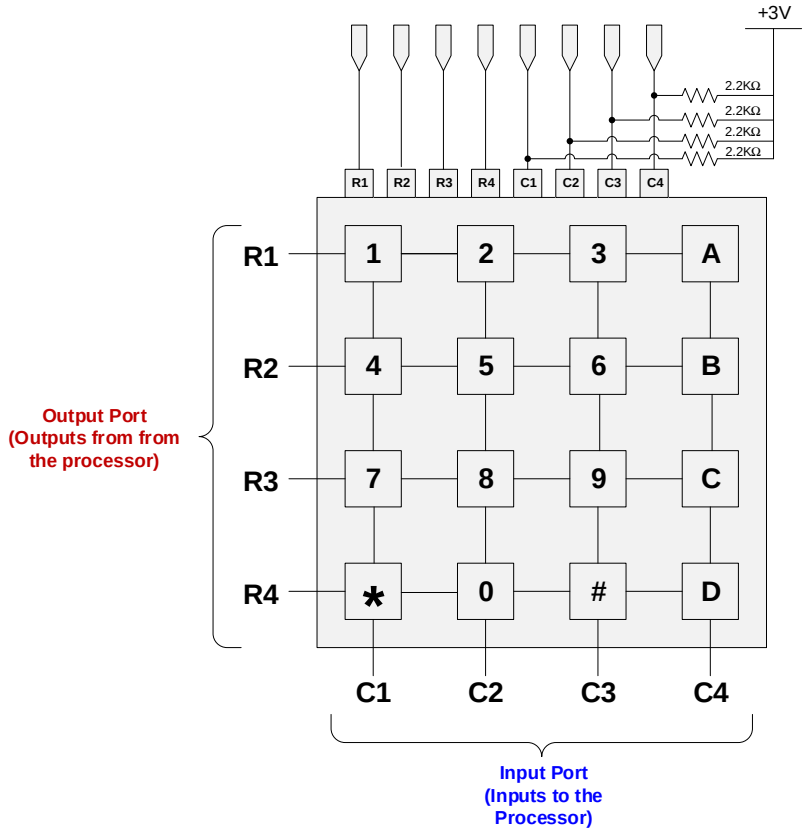
2. Connect a 4x4 matrix keypad to GPIO pins of the STM32 and write software to scan the keypad and return keypress status

3. Combine the code with a main that will initialize all the hardware and then loop continuously reading the keypad and sounding a tone while a key is pressed.

Keyboard Interface

Place the keypad used in this lab on a hard surface so that the keypad keys work consistently. The 4x4 keypad used in this lab requires 8 pins (four row pins and four column pins). For this lab, connect the keypad to the discovery board as follows:

Row	R1 → PE 15	R2 → PE 14	R3 → PE 13	R4 → PE 12
Column	C1 → PA 1	C2 → PA 2	C3 → PA 3	C4 → ***PA 5***

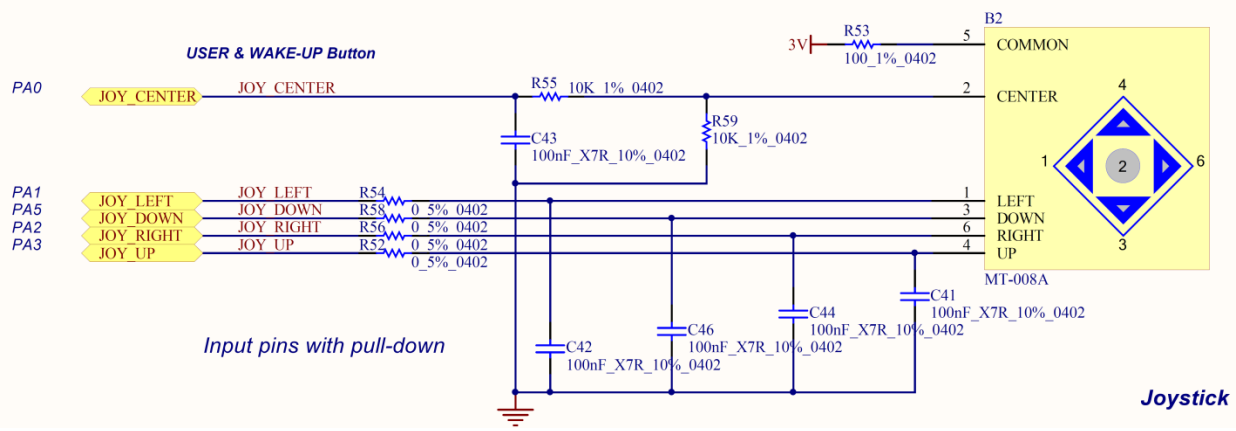


Notice that the keypad is being used in parallel with the joystick switches on the Discovery board. Within the processor, each GPIO pin can be pulled up via an internal resistor (between 20 and 55 KΩ, typically 40 KΩ). However, the internal pull-up capability is weak; the circuit will work better with external pull-up resistors. If available, add external 2.2KΩ pull-up resistors to 3V to all pins of the input port (C1, C2, C3, C4).

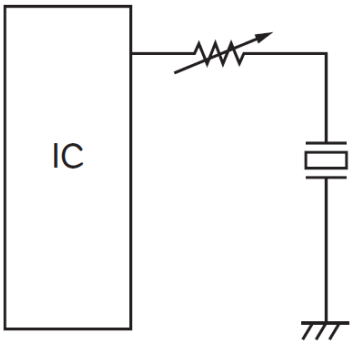
All GPIO pins connected to both the row outputs and column inputs will need the internal pull-up resistors turned on. The row output pins must be configured as open-drain to avoid damaging the GPIO pin drivers should 2 keys in the same column be pressed simultaneously.

Avoid pressing a key on the keypad and simultaneously pressing the joystick. This could result in the GPIO row pin being connected to 3V through a 100 ohm resistor which may momentarily exceed the 20 mA sink current threshold when keypad row associated with the pressed key is scanned.

On the STM32L4 board, all pins in the input port (PA1, PA2, PA3, PA5) are connected to ground via a 100nF capacitor, as shown in the figure below. You will need to account for the time constant formed by these capacitors and the pull-up resistors in your scanning algorithm. ***Add a delay of 15 ms between selecting a row to check and reading the column register to allow the capacitor on the previous row to recharge.***



Beeper Interface



Murata Electronics
PKM22EPPH4001-B0 Speaker

Your STM kit includes a MuRata piezo sounder (the beeper). Connect the beeper to PE8 – the same GPIO pin as the green LED – as shown above. If available, add an external 2.2KΩ resistor in series with the beeper. This will improve long-term reliability and reduce the risk of the processor pin being damaged if the board is dropped (piezo element in the beeper can generate a large voltage spike if driven mechanically by an external force).

The beeper should not be driven by a DC voltage signal; ensure that the timer output pin is low (green LED off) when not sounding a tone.

Tone Reference

https://en.wikipedia.org/wiki/Piano_key_frequencies

Each key on the keypad will be associated with a different tone. For this lab, use the 2 octaves of the C major scale centred on middle C. See the above link for a table of frequencies. The C major scale will be all the white keys of the associated piano keyboard image in the link.

You will need to write the following:

- A function to initialize the GPIO port pins connected to the keypad
- A function to initialize the pin connected to the beeper, select the timer alternate function for that pin, and configure the timer to generate a 50% duty-cycle square wave with a frequency that can be specified
- A GetKey() function to scan the keypad and return its state (i.e. which key is pressed, or no key is pressed). This function should use software debouncing. The Ganssle reference in the lab folder discusses debouncing; you may use this information to improve the basic debouncing algorithm covered in the textbook.
- A Beep() function that takes a specified tone as a parameter, and can be called in such as way as to turn the beeper off. Alternatively, you may use 2 different functions for this.

- A main() to do the following:

    - Call functions to initialize the GPIO ports and timer module
    - Loop forever doing the following:
        - Read the keypad and take the appropriate action depending on whether a key is pressed, or if no key is pressed
            - While any key is pressed, sound a tone associated with that key
            - When no key is pressed, turn the beeper off

To receive maximum credit, use good coding style, appropriately separate your code into modules, and comment as outlined for previous labs.