Lab 7: Timers

ECE 3720

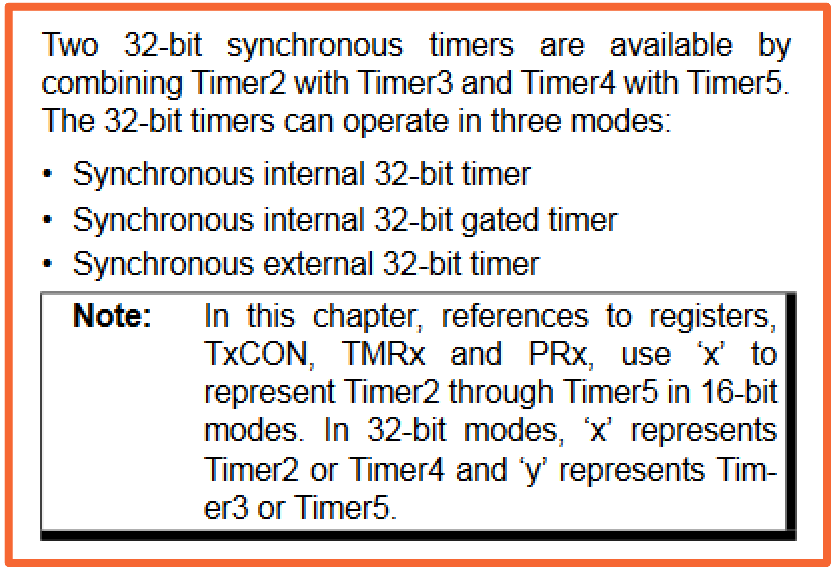
Preview

One of the microcontroller’s timers will be set up and used to trigger an interrupt at frequencies corresponding to musical notes. The interrupt will toggle the digital output to a piezo buzzer, causing it to produce the desired notes, and play a song.

Topic	Slide
Timers	3
Piezo Buzzer	6
Code	7
Lab Goals	8
Notes	9

Timers

- The PIC32's timers operate by counting up on every cycle of the clock. When a certain value is reached, an interrupt can be triggered, and the counting resets.
- Timer 1 (type A)
  - Datasheet pg. 151
  - 16-bit (so max count value is 0xFFFF)
  - Includes real-time clock functionality (not used in this lab)
- Timers 2-5 (type B)
  - Datasheet pg. 155
  - Each individual timer is 16-bit
  - Timers 2-3 and timers 4-5 can be combined to form 32-bit timers
    - Even-numbered timer supplies the control logic
    - Odd-numbered timer supplies the interrupt



Two 32-bit synchronous timers are available by combining Timer2 with Timer3 and Timer4 with Timer5. The 32-bit timers can operate in three modes:

- Synchronous internal 32-bit timer
- Synchronous internal 32-bit gated timer
- Synchronous external 32-bit timer

Note: In this chapter, references to registers, TxCON, TMRx and PRx, use 'x' to represent Timer2 through Timer5 in 16-bit modes. In 32-bit modes, 'x' represents Timer2 or Timer4 and 'y' represents Timer3 or Timer5.

PIC32 datasheet, pg. 156

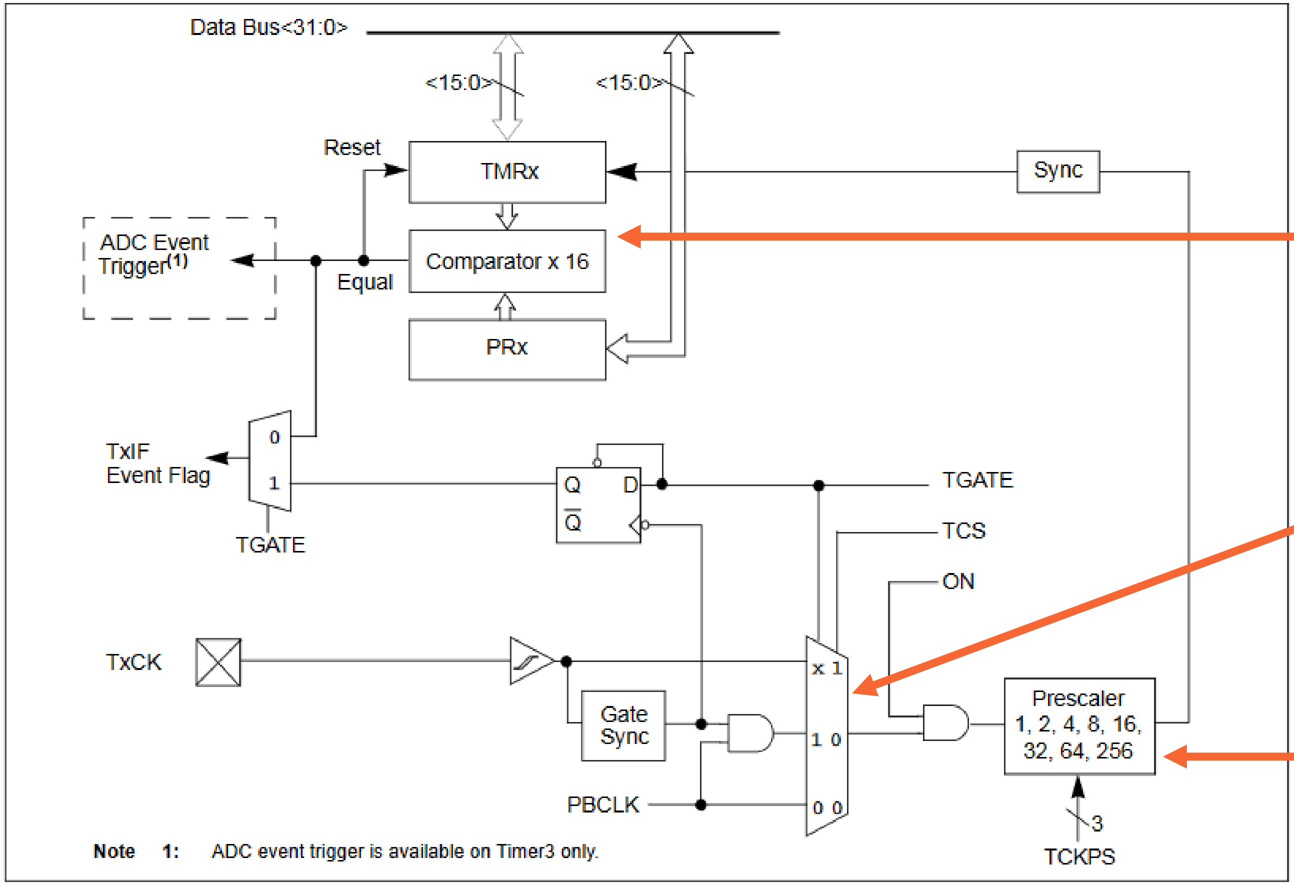
Timer Registers

The following are the registers of interest for this lab. The diagram on the next slide shows how they are used.

- **TxCON** (datasheet pg. 152/157)
  - Timer control register
  - Enable/disable timer
  - Select clock source
  - Select prescaler value
- **TMRx**
  - Holds the timer's running count value
  - Increments on each clock cycle (or every few cycles, depending on prescaler)
  - **You do not need to write to this register.**
- **PRx**
  - Holds value the timer should count to before resetting
  - **You will set this value to control how frequently the timer interrupt occurs.**
  - This value should only be changed in the ISR, or when timer is disabled.

Timer Diagram

FIGURE 13-1: TIMER2-TIMER5 BLOCK DIAGRAM (16-BIT)



Note 1: ADC event trigger is available on Timer3 only.

PIC32 datasheet, pg. 155

Diagram for timers 2-5 is shown

- Timer 1’s diagram is on pg. 151.
- The 32-bit timer diagram is on pg. 156.

When TMRx == PRx,

- TMRx gets reset
- Corresponding timer interrupt is triggered

Select clock source

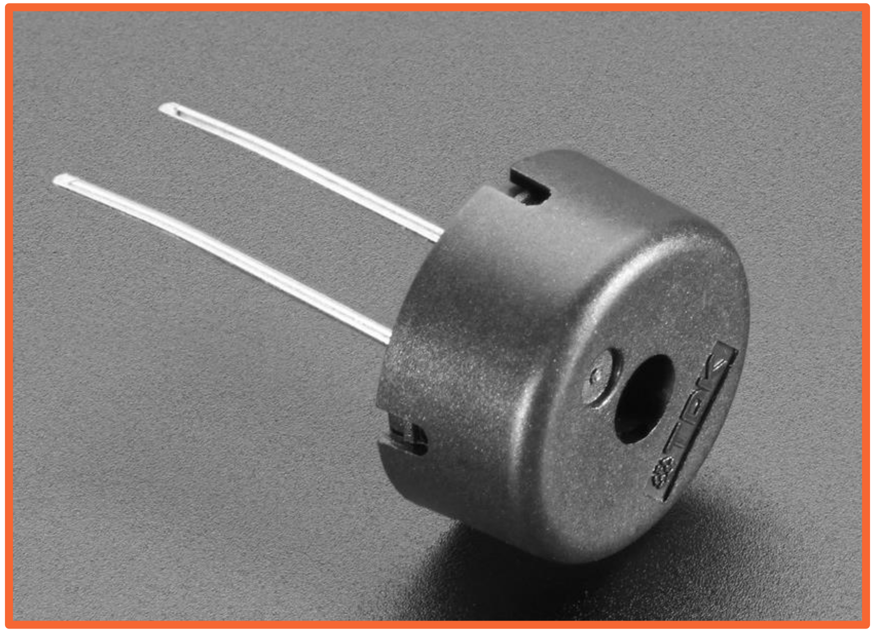
- **We’ll use the Peripheral Bus Clock (PBCLK),** which is derived from the built-in system clock.
- The alternative would be connecting an external source (TxCK).

Prescaler divides the clock frequency

- e.g., a prescaler value of 4 would cause the timer to count 4 times slower than the clock.

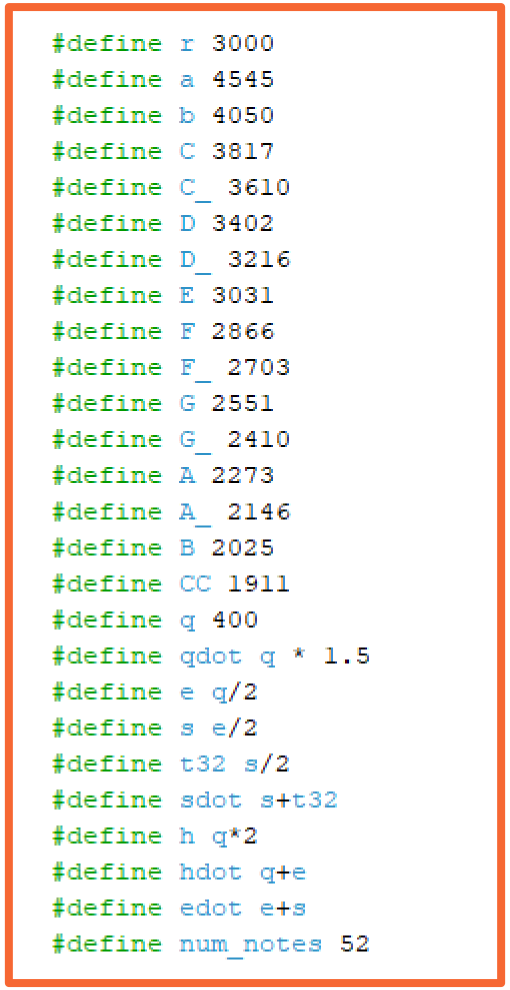
Piezo Buzzer

- A piezoelectric material changes shape when exposed to an electric field.
- Changing the voltage across the buzzer at a high frequency causes the material inside to vibrate at the same frequency, producing a note.
- **We will use timers to toggle an output voltage on and off at specific frequencies in order to play a song.**



Code

- For this lab you are provided a skeleton code (main_skeleton.c) which you will complete.
- The definitions at the top of the program (*a* through *CC*) represent the periods of notes, in terms of PIC32 clock cycles.
    - Recall that period is the inverse of frequency.
    - See the "notes" slide for info on how these are calculated.
- Definitions *q* through *edot* represent note lengths. *q* is a quarter note, *e* is an eighth note, etc.
- Notice you are given two arrays
    - *delay* is full of note lengths
    - *music_notes* contains notes in the order they are to be played
- **You do not need to change anything inside the *while(1)* loop**
    - The *if* statement steps through the notes of the song, remaining on each note until *j* reaches the value of *delay[i].*
    - Notice that *j* is not changed anywhere in the given code. You must increment *j* somewhere.

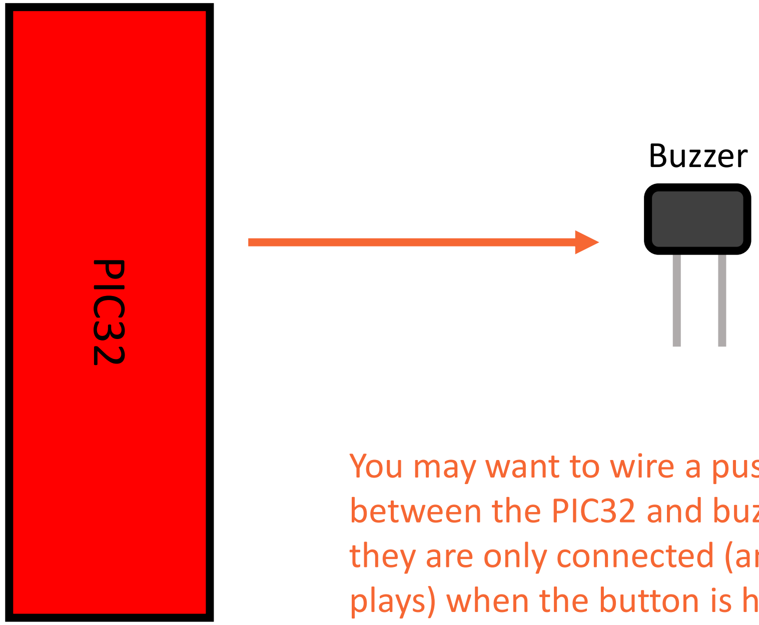


```
#define r 3000
#define a 4545
#define b 4050
#define C 3817
#define C_ 3610
#define D 3402
#define D_ 3216
#define E 3031
#define F 2866
#define F_ 2703
#define G 2551
#define G_ 2410
#define A 2273
#define A_ 2146
#define B 2025
#define CC 1911
#define q 400
#define qdot q * 1.5
#define e q/2
#define s e/2
#define t32 s/2
#define sdot s+t32
#define h q*2
#define hdot q+e
#define edot e+s
#define num_notes 52
```

Lab Goals

- Connect the buzzer to an output of the MC.
- Set up a timer using TxCON.
- Set up an interrupt triggered by your chosen timer.
- Use the interrupt to toggle the output. This will produce a note dependent on how frequently the interrupt occurs.
- The interrupt should also update the note being played.

Simple Diagram

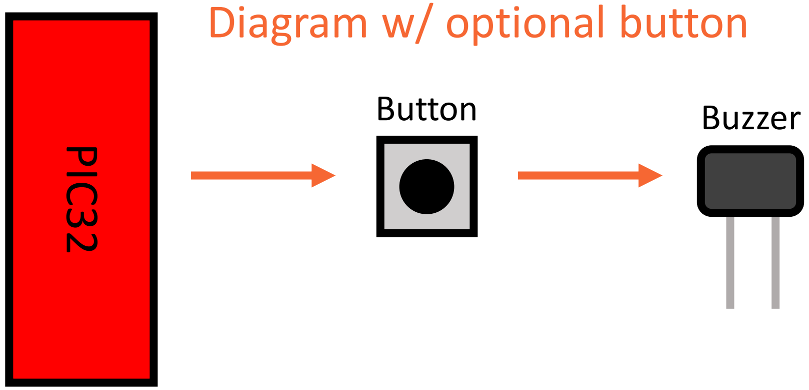


You may want to wire a push button between the PIC32 and buzzer such that they are only connected (and sound only plays) when the button is held down.

Notes

- Middle C example:
  - Middle C has a frequency of 261.6 Hz
  - MC clock runs at ~2 MHz
  - 2M cycles/s * 1s/261.6 = 7645 cycles per middle C period
  - Divide by 2 to get cycles per inversion → *#define C 3817*
- *r*, which appears in *music_notes*, represents a rest. No sound should be played when this is the current note.
- Consider the possible values to be loaded into PRx. Will a 32-bit mode timer be necessary?

Diagram w/ optional button


PIC32
Button
Buzzer