Timer: A peripheral for tracking time

The purpose of the prescaler is to allow the timer to be clocked at the rate a user desires.

Frequency depends on the attached oscillator device

Figure adapted from Prabal Dutta’s EE373 slides
Timers, why do we need them?

In the first project, what do we need timers for?

• Determining when to change LEDs
  – 20 Hz means change LED bits every 50 milliseconds
  – How to measure 50 ms?
  – Option 1: Use the timer hardware to let you know when 50 ms has passed.
  – Option 2: Count how many processor cycles it would take to equal 50 ms.
How does the number in the counter register correspond to wall clock time?

Frequency (Hz) = Cycles / Second

1 / Frequency (Hz) = Seconds / Cycle

The counter is incremented once per cycle.

You read 100 from the counter register which is clocked by a 1 MHz oscillator. How much time has passed since the counter was reset?
How should we choose the OSC frequency?

For timers, there will often be a tradeoff between resolution (high resolution requires a high clock rate) and range (high clock rates cause the timer to overflow more quickly).

1MHz OSC: resolution = 1 / 1e6 second = 1us
10MHz OSC: resolution = 1/10e6 second = 0.1us

16-bits timer:
1MHz OSC: max range = 1 / 1e6 * 2^16 = 65.536ms
10MHz OSC: max range = 1/10e6 * 2^16 = 6.5536ms
How does a firmware developer use the capture register?

1. Stop the timer
2. Setup the timer to capture when a particular event occurs (e.g., change of GPIO pin)
3. Reset the counter
4. Start the timer
5. Wait for the counter to reach a capture event (via interrupt or check status reg)
A timer for keeping track of wall-clock time
Real Time Clock (RTC)

Note: RTCs have their own oscillator.

Why is it 32,768 kHz?

The reason the 32,768 Hz resonator has become so common is due to a compromise between the large physical size of low frequency crystals and the large current drain of high frequency crystals.

Figure adapted from Prabal Dutta’s EE373 slides
Interrupts

How peripherals notify the CPU that an event has occurred.

Example: GPIO detected that a button was just pressed.
Interrupts

Definition

– An event external to the currently executing process that causes a change in the normal flow of software execution; usually generated by hardware peripherals, but can also be generated by the CPU.

– Key point is that interrupts are asynchronous w.r.t. current software procedure

– Typically indicate that some device needs service immediately
Why interrupts?

- MCUs have many external peripherals
  - Keyboard, mouse, screen, disk drives, scanner, printer, sound card, camera, etc.
- These devices occasionally need the CPU to act
  - But we can’t predict when
- We want to keep the CPU busy (or idle for power) between these events
  - Need a way for CPU to find out when a particular peripheral needs attention
Possible Solution: Polling

• CPU periodically checks each device to see if it needs service
  – “Polling is like picking up your phone every few seconds to see if you have a call. ...”
Possible Solution: Polling

- CPU periodically checks each device to see if it needs service
  - “Polling is like picking up your phone every few seconds to see if you have a call. ...”
  - Cons: takes CPU time even when no requests pending
  - Pros: can be efficient if events arrive rapidly