Embedded Software
CS 145/145L

Caio Batista de Melo
● Project 3 is short!
  ○ [https://canvas.eee.uci.edu/courses/45047/assignments/929272](https://canvas.eee.uci.edu/courses/45047/assignments/929272)
  ○ We’ll talk more about it on Thursday
  ○ It’s due next week (2022-05-06)!
  ○ Instead of early submission extra credit, you can get points for submitting a mid-quarter evaluation.
    ■ [https://evaluations.eee.uci.edu/](https://evaluations.eee.uci.edu/)
    ■ The evaluation should open as we talk about it in class
    ■ It’s open until Saturday (2022-04-30)
    ■ Completely anonymous, please provide your honest feedback :)

---

Announcements (2022-04-26)
Recap

State machines are useful in a lot of situations and are easy to understand!

http://howtomakeanrpg.com/a/state-machines.htm
Time-Ordered Behavior

Outputs

B0

Time

0 1s 2s 3s 4s

500 ms 500 ms
Project 1 State Machine

Init

Idle

ON

OFF

SET_BIT(PORTB, 0);
avr_wait(500);

CLR_BIT(PORTB, 0);
avr_wait(500);

GET_BIT(PINB, 1)

GET_BIT(PINB, 1)
Synchronous SM for Project 1

Period: 500ms

Init

Idle

ON

OFF

GET_BIT(PINB, 1)

!GET_BIT(PINB, 1)

SET_BIT(PORTB, 0);

CLR_BIT(PORTB, 0);

DDRB = 1;
Period and Frequency Relation

$$T = \frac{1}{f}$$

Frequency (hertz) $$\rightarrow$$ Frequency (hertz)

Period (seconds) $$\rightarrow$$ Period (seconds)

# Units of Time

<table>
<thead>
<tr>
<th>base unit</th>
<th>10^0</th>
<th>10^{-1}</th>
<th>10^{-2}</th>
<th>10^{-3}</th>
<th>10^{-6}</th>
<th>10^{-9}</th>
<th>10^{-10}</th>
<th>10^{-12}</th>
</tr>
</thead>
<tbody>
<tr>
<td>deci (d)</td>
<td></td>
<td>1/10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>centi (c)</td>
<td></td>
<td></td>
<td>1/100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>milli (m)</td>
<td></td>
<td></td>
<td></td>
<td>1/1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>micro (μ)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1/1000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nano (n)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1/100000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ångström (Å)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1/10000000000</td>
<td></td>
</tr>
<tr>
<td>pico (p)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1/100000000000</td>
</tr>
</tbody>
</table>

[https://heathermicrobiologyjackson.files.wordpress.com/2013/06/metric_prefix.jpg](https://heathermicrobiologyjackson.files.wordpress.com/2013/06/metric_prefix.jpg)
Synchronous SM for Project 1

Period: 500ms

Frequency: 2 Hz

Any possible problems? Might miss a button push…

<table>
<thead>
<tr>
<th>State</th>
<th>Transition</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Init</td>
<td>GET_BIT(PINB, 1)</td>
<td>DDRB = 1;</td>
</tr>
<tr>
<td>Idle</td>
<td>ON</td>
<td>SET_BIT(PORTB, 0);</td>
</tr>
<tr>
<td>Idle</td>
<td>OFF</td>
<td>CLR_BIT(PORTB, 0);</td>
</tr>
<tr>
<td>Idle</td>
<td>Idle</td>
<td>!GET_BIT(PINB, 1)</td>
</tr>
</tbody>
</table>

Frequency: 2 Hz

Period: 500ms

Any possible problems? Might miss a button push…
Increase frequency of the SM so we have faster checks; Add “loops” on states that need to execute longer.
volatile unsigned char TimerFlag = 0; // ISR raises, main() lowers

void TimerISR() {
    TimerFlag = 1;
}

*/ Unchanged SM code. */

void main() {
    B = 0; // Init outputs
    TimerSet(2000); // 2s period
    TimerOn();
    BL_State = BL_SMStart; // Indicates initial call to tick-fct
    while (1) {
        TickFct_Blink(); // Execute one synchSM tick
        while (!TimerFlag) {} // Wait for period
        TimerFlag = 0; // Lower flag
    }
}
Interrupts vs Wait functions

- Wait loops are usually known as busy waiting;
  - `while (!condition) {}`

- Interrupts allow your code to keep doing something else while not the time;

- Two possible benefits of interrupts:
  - multitasking;
  - sleep to save energy.
State Machine Timings

- Tick function should never be longer than period
- Tick function’s duration varies (It can be at different states.)
  - No control on x.
- Elastic code
- We can systematically make the wait smarter (e.g., sleep mode)
  - The wait function is not part of the SM specification and design.
What if the SM designer does not comply with the rules?
Missed deadline/Tasks

Timer
Flag

Main Loop

Timer
Flag

(2 - x)

Missing a full period

0 2 4 6 8

Time

Tick Function
Wait Function

Flag = 1
Flag = 0
while (1) {
    TickFct_Blink();
    while (!TimerFlag) {}  
    TimerFlag = 0;
}

Our code can detect these adverse situations!
The while loop won’t execute since the flag is already set!
We expect the flag to be 0 when I complete the Tick Function. We don’t only rely on the SM developer, we check and enforce it.

```c
while (1) {
    TickFct_Blink();
    if (TimerFlag) {
        assert(0);
    }
    while (!TimerFlag) {}
    TimerFlag = 0;
}
```

Can this code tell jitters and missed deadlines apart?

What can we change to allow that?
Interrupts on ATmega32

// global variable to count the number of overflows
volatile unsigned char timer_0_overflow;

// TIMER0 overflow interrupt service routine
// called whenever TCNT0 overflows
ISR(TIMER0_OVF_vect) {
    // Keep a track of number of overflows
    timer_0_overflow++;
}

https://maxembedded.com/2011/06/avr-timers-timer0/

Other types of interrupts also available: https://exploreembedded.com/wiki/AVRExternalInterrupts
See you next time :)