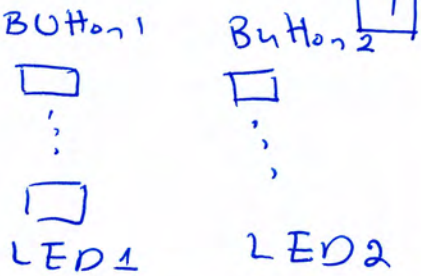
EEL 4742C - 04/10/2024

Masks

Port 1 { BUT1, BUT2 / LED1, LED2



Types of unfairness

(I) Button 1 pushed first, LED2 is lit.

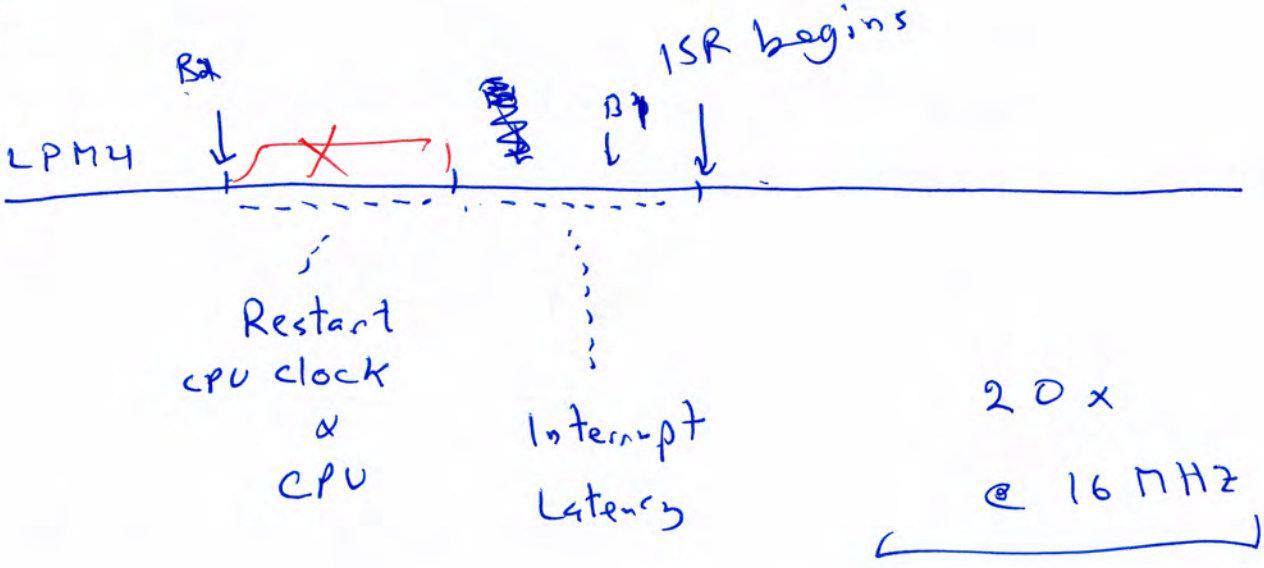
(II) Button 1 pushed first, both LEDs are lit.

Main

```
P1DIR |= LED1 | LED2;
P1OUT &= ~(LED1 | LED2); // OFF
// Config button interrupts
...
__low_power_mode_4();
__enable_interrupts();
while(1) {}
```

Main ends

```
void Port1_ISR() {
unsigned char temp = P1IFG;
// Button 1
if ((P1IFG & BUT1) != 0) {
    P1OUT |= LED1;
    // Disable buttons' interrupt
    P1IE &= ~(BUT1 | BUT2);
}
// Button 2
if ((P1IFG & BUT2) != 0) {
    P1OUT |= LED2;
    P1IE &= ~(BUT1 | BUT2);
}
}
```



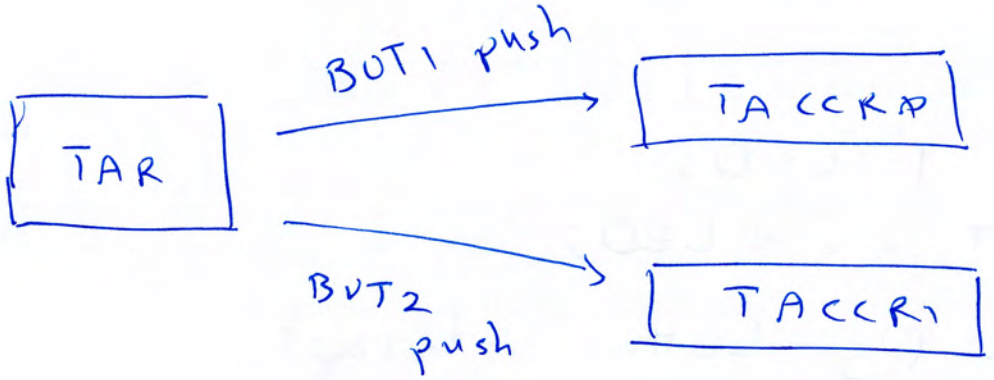
w) Polling

```
while (1) {
    if ( B1 ↓  &  B1 ↑ )
        ...
    if ( B2 ↓  &  B1 ↑ )
        ...
}
```

⇓

```
Loop:
bit test ... bit ~~~
        ...
        bit ....
        ...
        JMP Loop .... 3x
```

w/ Timer Input Capture



1x ⟶ @ 16 MHz

Inline Interrupt Processing

Main

Infinite Loop: {

→ wait for interrupt 1 (LPMx) ⟶ ISR

Process interrupt 1

Return in Active Mode

→ wait for interrupt 2 (LPMy) ⟶ ISR 2

Process interrupt 2

Return in Active Mode

}

⊛ Button push turns the LED ON for 3 seconds.

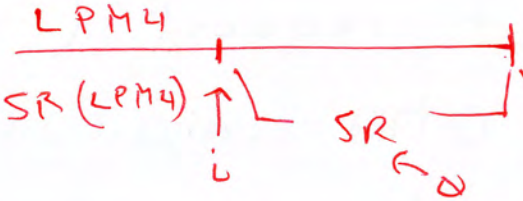
Main

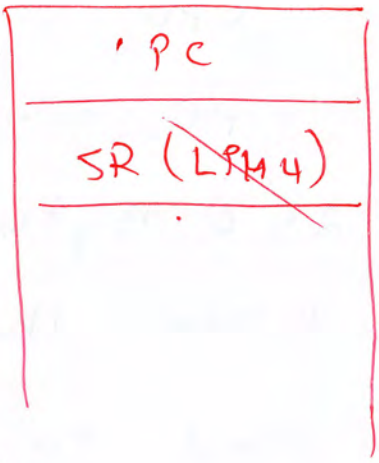
```
P1DIR |= LED;
P1OUT &= ~LED;
// Config button interrupt

while (1) {
    // wait for button push
    __low_power_mode_4();
    P1OUT |= LED;
    // Timer, Up, 3s, ch0 int.
    TACCR0 = (32768/2 * 3) - 1;
    TACCTL0 &= ~CCIFG;
    TACCTL0 |= CCIE;
    TACTL = TASSEL_1 | ID_1 | MC_1 | TACLR;
    // wait for timer
    __low_power_mode_3();
    P1OUT &= ~LED;
    TACTL = 0;
    TACCTL0 &= ~CCIE;
}
```

↳ 16 kHz * 3s

/2

```
void Port1_ISR () {
    - delay - cycles (15000); // Debounce
    P1IFG &= ~BUT;
}
```





CCS

```
-BIC_SR_REGISTER_ON_EXIT (0xFFFF);
```

reti

Modifying the SR copy on the stack

```
-low_power_mode_off_on_exit();
```

IAR

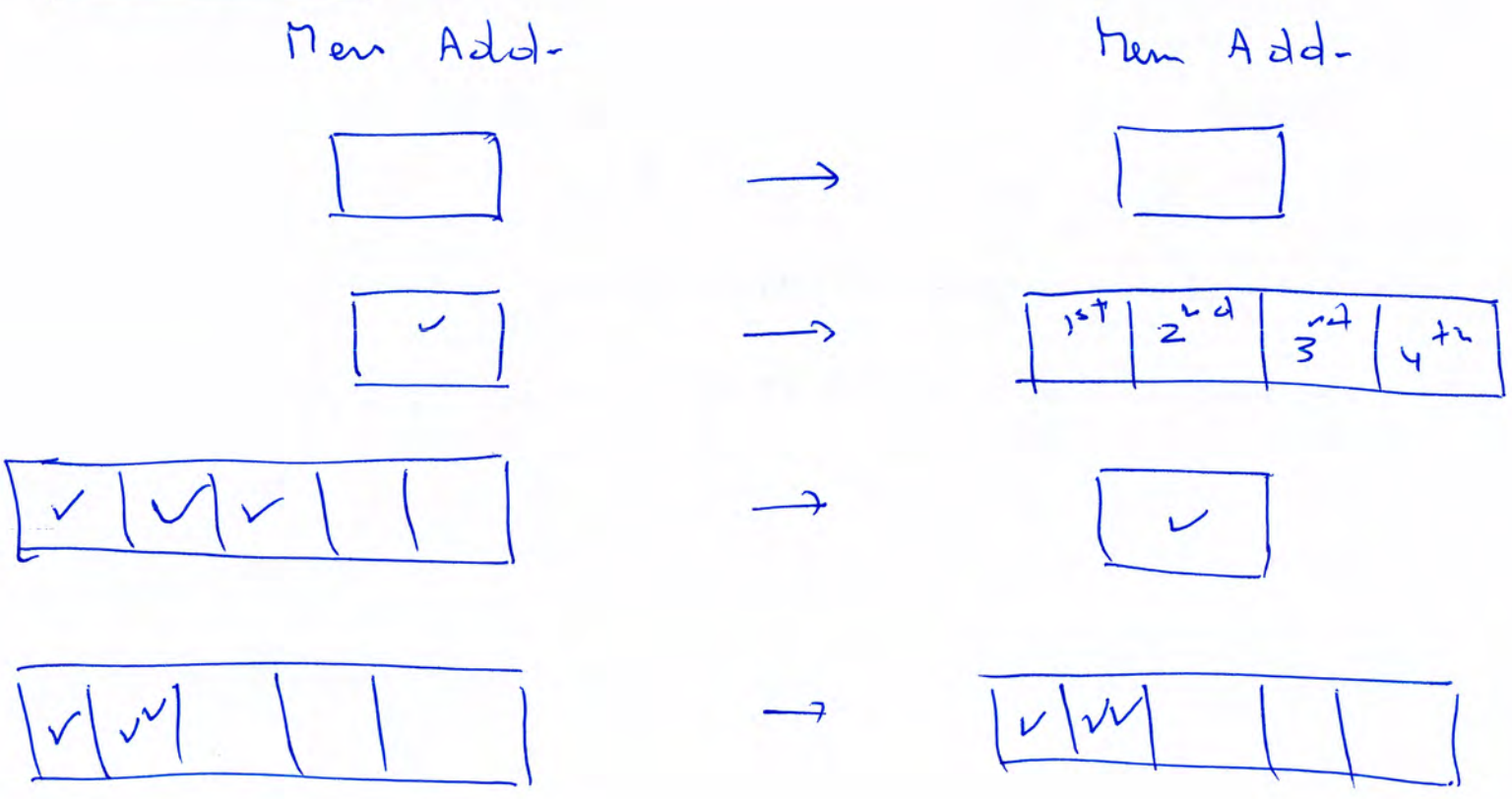
intrinsics.h

```
void TA0_A0_ISR () {
    // HW clears the flag
    -BIC_SR_REGISTER_ON_EXIT (0xFFFF);
}
```

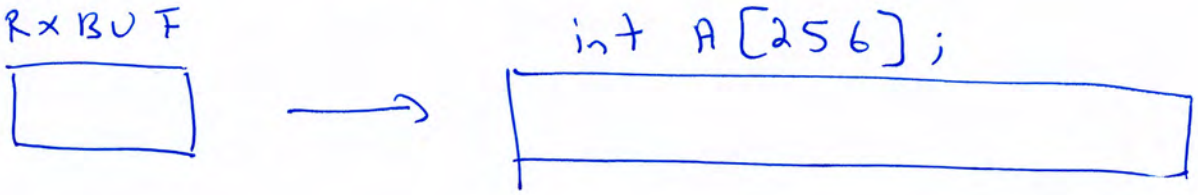
Direct Memory Access (DMA) Module

- Copies from Memory to Memory w/o CPU interference.
- CPU makes a request (e.g copy 256 Bytes), DMA raises an interrupt when the whole operation is done.
- Used to process I/O data.
- Triggers: UART data arrival, UART unit is ready, ADC conversion complete.

Setup:



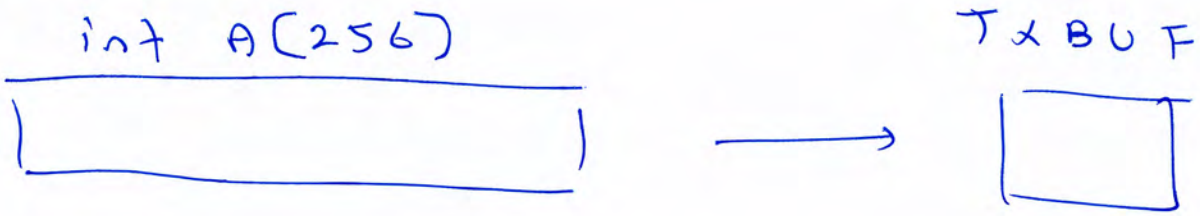
⊛ Data arriving on UART is copied to an array / buffer.



Address: &RxBUF A { &A ; &A[0]

Trigger: RxIFG

⊛ Transmit a block of Memory (array) on UART.



Address: A &TxBUF

Trigger: TxIFG ..... { 0: In progress ; 1: Unit is Ready