Exercise sheet 4 – Hardware, Processor architecture

Goals:

- Interrupt handling

Exercise 4.1: Processor architecture: Interrupt handling (theoretical)

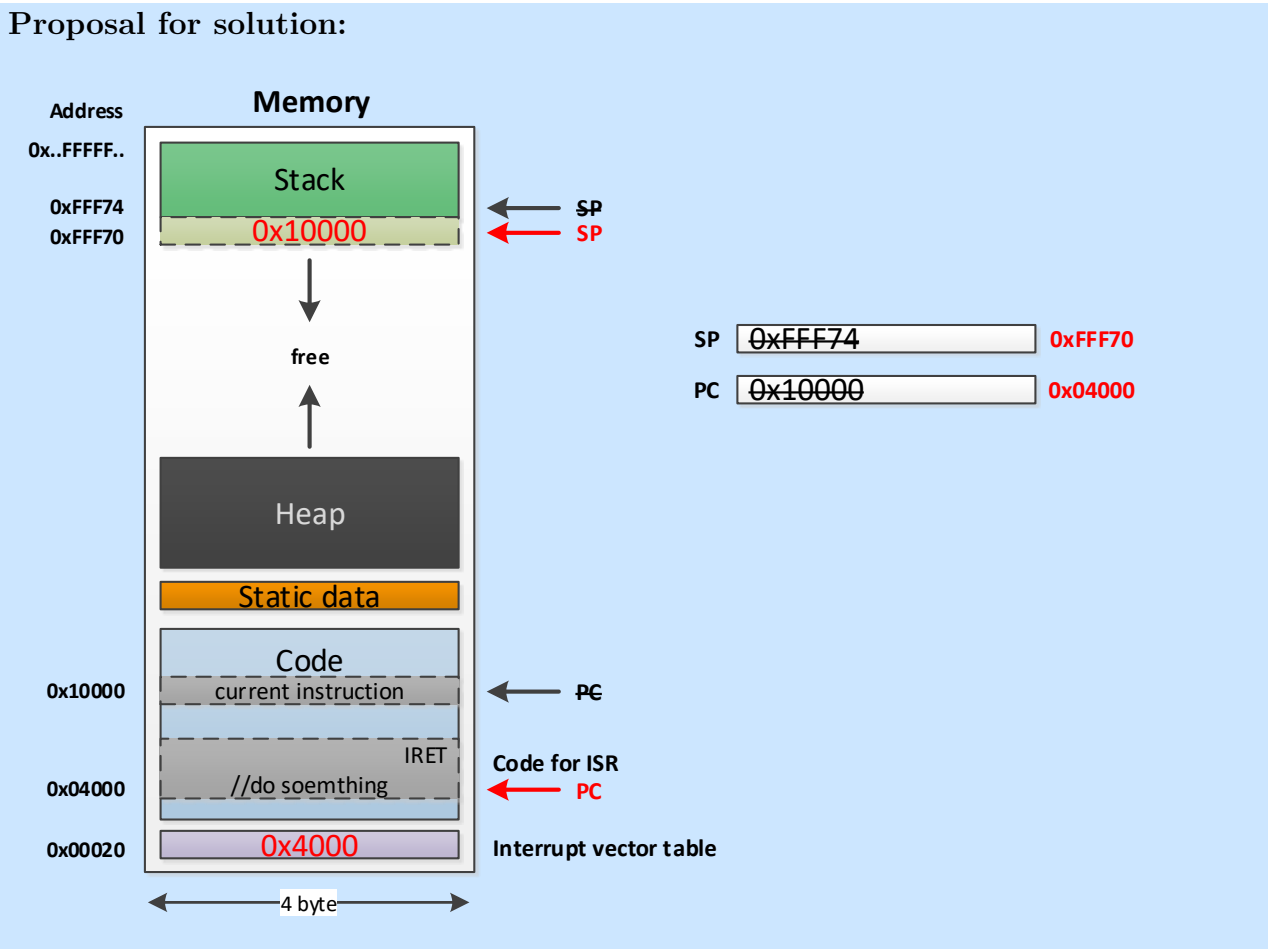
Given information:

- Interrupt vector address is 0x20
- Position of interrupt service routine (ISR) starts at 0x4000
- Stack pointer (SP) contains 0xFFF74
- Program counter (PC) respectively instruction pointer (IP) contains 0x10000
- Consider a micro controller without an operating system

(a) Recapitulate the sequence of an interrupt.

(b) Draw a sketch and show the changes according to the processing of an interrupt in different colours. The drawing should contain at least a memory view including addresses (32 bit: 4 byte with) and the PC and SP registers.

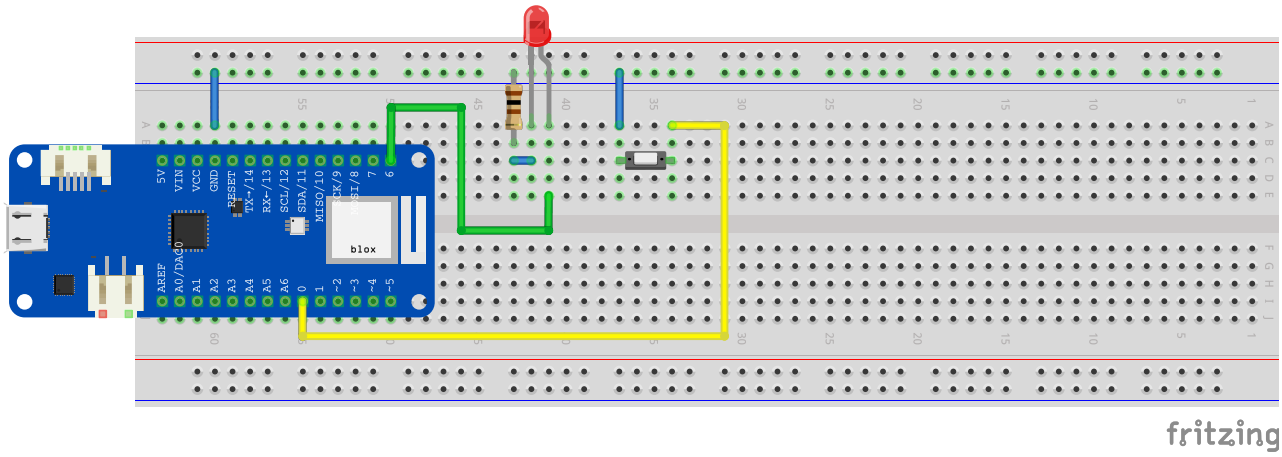
Proposal for solution:



Exercise 4.2: Processor architecture: Interrupt handling (coding)

We want to write an Arduino sketch which toggles the built-in LED when a button is pressed. If the button is pressed, an interrupt occurs which calls an ISR.

(a) Prepare the wiring with the Arduino MKR WiFi 1010 as follows:



(b) Please double check your wiring with the lecturer, before you connect the Arduino MKR WiFi 1010.

(c) Make sure that you have installed the Arduino IDE (`https://www.arduino.cc/en/software`).

(d) Make sure you have installed the board SDK:

- `Tools -> Board: -> Boards Manager...`
- Install (latest version): `Arduino SAMD Boards (32-bits ARM Cortex-M0+)`

(e) Open the skeleton file from
`CA_exercises/sheet_04_interrupts_hw/io_interrupt/io_interrupt.ino`
with the Arduino IDE.

(f) Follow the TODOs in the code. Some configuration depends on your wiring of the I/O pins. *Hint: The Arduino reference contains descriptions of the used functions:* ***https://www.arduino.cc/reference/en.***

Proposal for solution:

```
1  //PIN configuration
2  const int BUTTON_PIN = 0;
3  const int LED_PIN    = LED_BUILTIN; //Internal built-in LED
4
5  //Global variables
6  volatile bool led_state = false;
7  volatile unsigned long time_prev_rising_edge = 0;
8
9  /*!
10  * setup() is called once on startup/reset of the Arduino
11  */
12 void setup(){
13   //Configure serial connection
14   Serial.begin(9600);
15
16   //configure PINS
17   pinMode(BUTTON_PIN, INPUT_PULLUP);
18   pinMode(LED_PIN,    OUTPUT);
```

```
19
20    //Switch LED initally off
21    digitalWrite(LED_PIN, false);
22
23    //Configure interrupt
24    attachInterrupt(digitalPinToInterrupt(BUTTON_PIN), isr_button_pressed, RISING);
25  }
26
27  /*!
28   * loop() is called as fast as possible.
29   *
30   * As you can see, there is no call to a function
31   * changing the LED state in the main-loop
32  */
33  void loop(){
34    Serial.print("LED state is: ");
35    if(led_state) {
36      Serial.println("ON");
37    } else {
38      Serial.println("OFF");
39    }
40    delay(1000); //wait for 1000 ms -> 1 sec
41  }
42
43  /*!
44   * isr_button_pressed() = Interrupt service routine
45   * Change here the state of the LED
46   */
47  void isr_button_pressed(){ //interrupt service routine
48    if (millis() - time_prev_rising_edge > 250) { //only react on rising edges every 250
49      led_state = !led_state;
50      digitalWrite(LED_PIN, led_state);
51
52      if(led_state) {
53        Serial.println("LED STATE UPDATED: ON");
54      } else {
55        Serial.println("LED STATE UPDATED: OFF");
56      }
57
58      time_prev_rising_edge = millis();
59    }
60  }
```

(g) Configure the board within the Arduino IDE: `Tools -> Board: -> Arduino MKR WiFi 1010`.

(h) Compile (verify) your sketch within the Arduino IDE. If it compiles then upload it your sketch.

(i) Open the 'Serial Monitor' to see the printed strings and to do some debugging with the text based logging.

(j) Press the button to test your sketch. Does it work as expected?

Exercise 4.3: Processor architecture: Interrupt handling (coding) – additional coding exercise

We want to write an Arduino sketch which toggles the built-in LED when a timer is triggered. If the timer is triggered, an interrupt occurs which calls an ISR. Click here to get information about the timer module for the CPU.

(a) Make sure you have removed any wiring from the Arduino MKR WiFi 1010.

(b) Open the skeleton file from

`CA_exercises/sheet_04_interrupts_hw/timer_interrupt/timer_interrupt.ino` with the Arduino IDE.

(c) Make sure you have installed the `Adafruit_ZeroTimer` library:
`Sketch -> Include Library -> Manage Libraries... -> Adafruit_ZeroTimer`.

(d) Follow the TODOs in `timer_interrupt.ino`.
Hint 1: The Arduino reference contains descriptions of the used functions: ***https://www.arduino.cc/reference/en****.*
Hint 2: The Adafruit_ZeroTimer github repository examples: ***https://github.com/adafruit/Adafruit_ZeroTimer****.*

Proposal for solution:

```
#include <Arduino.h>
#include <Adafruit_ZeroTimer.h>

//PIN configuration
const int LED_PIN    = LED_BUILTIN; //Internal built-in LED

//Global variables
volatile bool led_state = false;

//Defines a timer on timer 3 (16 bit timer)
Adafruit_ZeroTimer timer = Adafruit_ZeroTimer(3);

/*!
 * TC3_Handler() is the ISR for timer 3.
 * We have to forward the interrupt to the Adafruit_ZeroTimer library.
 */
void TC3_Handler() {
  Adafruit_ZeroTimer::timerHandler(3);
}

/*!
 * setup() is called once on startup/reset of the Arduino
 */
void setup() {
  //Configure serial connection
  Serial.begin(9600);

  //configure PINS
  pinMode(LED_PIN,    OUTPUT);

  //Switch LED initally off
  digitalWrite(LED_PIN, false);

  timer.configure(TC_CLOCK_PRESCALER_DIV1024,    // prescaler
                  TC_COUNTER_SIZE_16BIT,         // bit width of timer/counter
                  TC_WAVE_GENERATION_NORMAL_FREQ // frequency or PWM mode
                );
  timer.setCompare(0, 0xFFFF); //if the internal counter=0xFFFF
                               //then it calls the timer3_callback
  timer.setCallback(true, TC_CALLBACK_CC_CHANNEL0, timer3_callback);
  timer.enable(true);
}

/*!
 * loop() is called as fast as possbile.
```

```
 */
void loop() {
  Serial.print("LED state is: ");
  if(led_state) {
    Serial.println("ON");
  } else {
    Serial.println("OFF");
  }
  delay(1000); //wait for 1000 ms -> 1 sec
}

/*!
 * This is the callback from timer3. It is called from the
 * Adafruit_ZeroTimer timer library.
 */
void timer3_callback()
{
  led_state = !led_state;
  digitalWrite(LED_PIN, led_state);

  Serial.println("Timer3 is triggered");
}
```

(e) To get an interrupt frequency between 1 and 2 seconds, we will use a prescaler of 1024. Search inside the Adafruit_ZeroTimer driver header to find the right enum value to set.

Proposal for solution: If we have 48 MHz (wich means a tick every 0.0208 μs), our 16-bit timer would count with this frequency too, which would lead to an overflow (which triggers the interrupt in the end) every 1365 μs (1.365 ms). To increase the time until the next overflow, we have to slow down the count of the timer by dividing the base clock rate by our prescaler value. `TC_CLOCK_PRESCALER_DIV1024` will divide the clock frequency (48 MHz) by 1024, which causes an overflow every 1.39 s. So we can summarize: The prescaler divides the clock frequency of the TC module to make the counter count slower.

(f) Configure the board in Arduino IDE: `Tools -> Board: -> Arduino MKR WiFi 1010`.

(g) Compile (verify) your sketch within the Arduino IDE. If it compiles, upload your sketch.
Hint: Sometimes it will help to enter the bootloader mode (double press the built-in button) to upload sketches to the MKR Wifi 1010.

(h) Open the „Serial Monitor“ to see the printed strings. Does it work as expected?