CSE 433 – Embedded Systems STM HW1

Gökçe Nur Erer - 171044079

1) At this part it was required to use the count down property and prescalar settings of the timer to change the overflow period to 500ms and blink a led on the card with 500ms intervals. For this first prescalar value is calculated. The clock for this project was setted as 100 mHz which means it has a 10 ns period. Also since our counter is 16-bits we can count up to 65535. So without the prescalar setting the overflow period would be 655.35 microseconds. Which is not enough for the request. So the prescalar value is calculated like this:

   If the prescalar value being 1 results in 655.35 microseconds of an overflow period, which prescalar value will result the overflow period higher or equal but closest to 500ms:

$$\frac{500ms}{655{,}35\mu s} = 762.951095 \ \approx 763$$

   The new period of the clock would be 7630 ns from 10ns * 763. Now it is required to find the maximum count value which will result in 500ms in total. For that the counter period can be calculated as:

$$\frac{500ms}{7630ns} = 65530.7995 \ \approx 65531$$

   After these values are setted in CubeMX perspective in the TIM_2 timer of the STM32 card, it is required to code what will happen when the counter period is exceeded. So HAL_TIM_PeriodElapsedCallback function is called when this counter period exceeds, which means 500ms has passed. (In this case since both counter period and prescalar values are rounded up to higher number the overflow period is not exactly 500ms but a little bit higher like 500.001ms). In this function we toggle the led pin and with that the led blinks with 500ms intervals on the card.

2) At this part it was required to generate a square wave with 1ms period using the output compare property of a timer. For this a channel is selected on the TIM_2 timer which was used in part 1 to be used as an output compare channel. To determine the signal creation on the channel we first calculate how much counts it is needed to pass for 1ms period:

$$\frac{1ms}{7630ns} = 131.061599 \ \approx 132 \ counts$$

   Output compare has the property of toggling a signal with given pulse value, so if we want a 132 count of a wave we need to toggle the signal at 132 / 2 = 66th count. After finding this value, this value is setted in the CubeMX perspective in the TIM_2 timer's channel 2. When the output compare register changes an interrupt happens and the HAL library's HAL_TIM_ OC_DelayElapsedCallback function handles what has to be done when this register changes. So we get the value from the register and increase it with 66 which is the count we found at the previous calculation.

3) At this part it was required to detect the generated signal at part 2's frequency with the input capture feature. For this first a channel is selected on the TIM_2 timer which was used in part 1 and 2 to be used as an input capture channel. To determine when we will capture the input we set the channel to capture the input at rising edge. So we can calculate the difference between the two rising edges and we can calculate the frequency. When input compare register changes and interrupt happens and the HAL library's HAL_TIM_ IC_CaptureCallback function handles what has to be done when this register changes. So we check if we got the first rising edge count if we didn't we read the register and set it to be the first rising edge. If it is not we again read the register and set it to the second rising edge value. Then when both values are setted we calculate the count difference between these values which results 132 in my case. Then we find the required frequency with the following calculation:

$$\frac{1}{difference * prescalared\ period} = \frac{1}{132 * 0{,}00000763s} = 992.890901\ Hz \approx 1000\ Hz$$

Expression	Type	Value
(x)= freq	double	0.99289090114778178

Since at all the parts some calculations had been rounded up to the higher value the frequency resulted a little bit less that how it should be.

**PS: All the parts are done in the same main.c file within different functions and same configurations.

** Screenshots of the settings and the code explained above :

Code:

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef * htim){
     if (htim->Instance==TIM2){
         HAL_GPIO_TogglePin(LD2_GPIO_Port, LD2_Pin);
     }
}

void HAL_TIM_OC_DelayElapsedCallback(TIM_HandleTypeDef *htim){
    uint32_t capture_content;
    if(htim->Channel == HAL_TIM_ACTIVE_CHANNEL_2){
        capture_content = HAL_TIM_ReadCapturedValue(htim,TIM_CHANNEL_2);
        __HAL_TIM_SET_COMPARE(htim, TIM_CHANNEL_2, capture_content+66);
    }
}

void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim){
    if(htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1){
        if(is_first_captured == 0){
            captured_value_1 = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_1);
            is_first_captured = 1;
        }
        else if(is_first_captured == 1){
            captured_value_2 = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_1);
            if(captured_value_2 > captured_value_1){
                diff = captured_value_2 - captured_value_1;
            }
            else{
                diff = captured_value_1 - captured_value_2;
            }
            is_first_captured = 0;

            freq = 1.0 / (diff * 7630);
            freq *= 1000000 ; // converting to khz
        }
    }
}
```

Timer Configuration: