Training Project Laboratory :
ARM microcontrollerprogramming in C language

Name: Das Anjan Kumar (D42DQA)

Instructor name: Kovacs Viktor

Task 1.1:

In this laboratory experiment we have worked with ARM microcontroller. For he first task we worked to get LED lights to work in a specific implimentation. First we have written the function implementation inside the various functions in the sbsp_led.c source file. There were various functions such as `LED_Clock_On()`, `LED_Clock_off()`, `LED_SetValue()`.

bsp_led.c file description:

```
#include "bsp.h"

/*
 *   Initialize pins given by ledPos for LED usage
 */
void LED_Init(uint16_t ledPos)
{

     /* Check if the input parameter is valid */
     assert_param(ledPos & LED_ALL);
     ledPos &= LED_ALL;
     /* GPIOE Pin8..15 are connected to LD1..LD8 LEDs */
```

```
    /* Enable periphery clock */
    __HAL_RCC_GPIOE_CLK_ENABLE();
    /* Setup pin parameters (output, push-pull,
pullup/down disabled, low speed */

    GPIO_InitTypeDef GPIO_InitStruct;
    GPIO_InitStruct.Pin =
GPIO_PIN_8|GPIO_PIN_9|GPIO_PIN_10|GPIO_PIN_11|GPIO_PIN_12|
GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_LOW;


    HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);


    /* LED_CLK is connected to GPIOA Pin15*/
    /* Enable periphery clock */
    __HAL_RCC_GPIOA_CLK_ENABLE();
    /* Setup pin parameters (output, push-pull,
pullup/down disabled, low speed */

    GPIO_InitStruct.Pin = GPIO_PIN_15;


    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

    /* #LEDEN is connected to GPIOC Pin11*/
    /* Enable periphery clock */
    __HAL_RCC_GPIOC_CLK_ENABLE();
    /* Setup pin parameters (output, push-pull,
pullup/down disabled, low speed */



    GPIO_InitStruct.Pin = GPIO_PIN_11;


    HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
```

```
}

/*
  * Set the value to the LEDs given by ledPos
 */
void LED_SetValue(uint16_t ledPos, uint16_t value)
{
     /* Check if the input parameter is valid */
     assert_param(ledPos & LED_ALL);
     ledPos &= LED_ALL;

     /* Mask out the bits from value which are not given by
ledPos */
     /* Write the value to the output data register (ODR)
      * Note that change only the values that are given in
ledPos! */

     GPIOE->ODR = value;

     /* Give an impulse to LED_CLK pin to write the pin
states to the output buffer*/

     LED_Clock_Pulse();



}

/*
 * Set LED_CLK pin to high (LED_CLK=1)
 */
void LED_Clock_On()
{
     /* Set LED_CLK pin to high */
     HAL_GPIO_WritePin(GPIOA, GPIO_PIN_15, SET);
}
/*
 * Set LED_CLK pin to low (LED_CLK=0)
 */
void LED_Clock_Off()
{
     /* Set LED_CLK pin to low*/
```

```
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_15, RESET);

}
/*
 * Send an impulse to LED_CLK (LED_CLK=1;LED_CLK=0)
 */
void LED_Clock_Pulse()
{
    LED_Clock_On();
    LED_Clock_Off();
    DisplaySampleLEDs();
}
/*
 * Enabled LEDs using pin #LEDEN (#LEDEN=0)
 */
void LED_Enable()
{
    /* Set #LEDEN to logical 0 */
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_11, RESET);

}
/*
 * Disable LEDs by pin #LEDEN (#LEDEN=1)
 */
void LED_Disable()
{
    /* Set #LEDEN to logical 1 */
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_11, SET);

}
```

The logic used to make the SIDE BY SIDE led lights:

initializing the ledpos and the value of a:

```
  uint16_t ledPos,a;
  ledPos = 0xff00;
  a = 0x0100;
```

Logic:

```
if(a!=0)  {
     LED_SetValue(0xff00, a);
     a = a<<1;
}
else {
 LED_SetValue(0xff00, a);
 a = 0x0100;
}
```

In this task we have we will keep turning on lights from one side and and one by onethe light will turn on and one by one the light will turn off.

In the functions we have set the GPIO ports waccording to its parameters like PIN, MODE, PULL and SPEED. We have also set the GPIO ports as E A and C and we have also implemented the function of the clock pulse and the LEDSETVALUE tor un the onboard LEDs of the ARM microcontroller.

Task 1.2:

```
#include "bsp.h"
```

```
TIM_HandleTypeDef Tim4Handle;
TIM_HandleTypeDef Tim6Handle;
/*
 * TIM4 timer initialization
 */
void Timer4_Init()
{
    /* Enable periphery clock */
    __HAL_RCC_GPIOC_CLK_ENABLE();


    /* Timer initialization */
    HAL_TIM_Base_Init();



    Tim4Handle.Instance = TIM4;
    Tim4Handle.Init.CounterMode =
TIM_COUNTERMODE_UP;
    Tim4Handle.Init.Period = 20999;
    Tim4Handle.Init.Prescaler = 999;
    Tim4Handle.Init.ClockDivision =
TIM_CLOCKDIVISION_DIV1;




    /* TIM4 interrupt priority setup*/
    HAL_NVIC_SetPriority(TIM4,1,1);


    /* enable TIM4 interrupt */
    HAL_NVIC_EnableIRQ(TIM4);
}

/*
 * Start TIM4 timer
```

```
 */
void Timer4_Start()
{


    HAL_TIME_BASE_START_IT(&Tim4Handle);


}

/*
 * Stop TIM4
 */
void Timer4_Stop()
{
    HAL_TIME_BASE_STOP_IT(&Tim4Handle);

}

/* Interrupt handler for TIM4 */
void TIM4_IRQHandler(void)
{
    HAL_TIM_IRQHandler(&Tim4Handle);
}

/* Interrupt handler for TIM6  */
void TIM6_DAC_IRQHandler(void)
{
    HAL_TIM_IRQHandler(&Tim6Handle);
}

/* Interrupt handler callback for a Timer */
void
HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *
htim)
{
```

```
    volatile static int a = 0x100;
    if(htim->Instance==TIM4)
    {
        if(a!=0x0000)
        {
            LED_SetValue(0xff00, a);
                          a = a<<1;

        }
        else
        {
               LED_SetValue(0xff00, a);
               a = 0x0100;
        }

    }

}
```

In this task we set the intial parameters of the timer. After that we have set the on off function of the timer accordingly. We have also created another function named elasped function in the timer 4 task which implemens as the same task we have done previously in 1.1.