Национальный исследовательский университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №3
по дисциплине «Встроенные системы»

Вариант 2

Выполнил: Ларочкин Г.
Шуст И.
Тарасенко Д.
Группа: P3400
Преподаватель: Ключев А.О.

Санкт-Петербург
2020 г.

Описание задачи

Реализовать обработчик прерываний, а также основную программу, которая будет прерываться по таймеру. Основная программа – вывод на светодиоды анимации из ЛР№2. В обработчике прерывания последовательно включаются красный, желтый и зеленый светодиоды. Периодичность прерывания – 2 секунды.

Выполнение

Таймер

Для реализации прерываний был выбран таймер TIM7. На рисунке 1 представлена схема работы таймера. Это 16-ти битный таймер, с возможностью установки значения *Prescaler* и *Auto-reload Register (ARR)*, которые позволяют установить необходимое время для таймера.

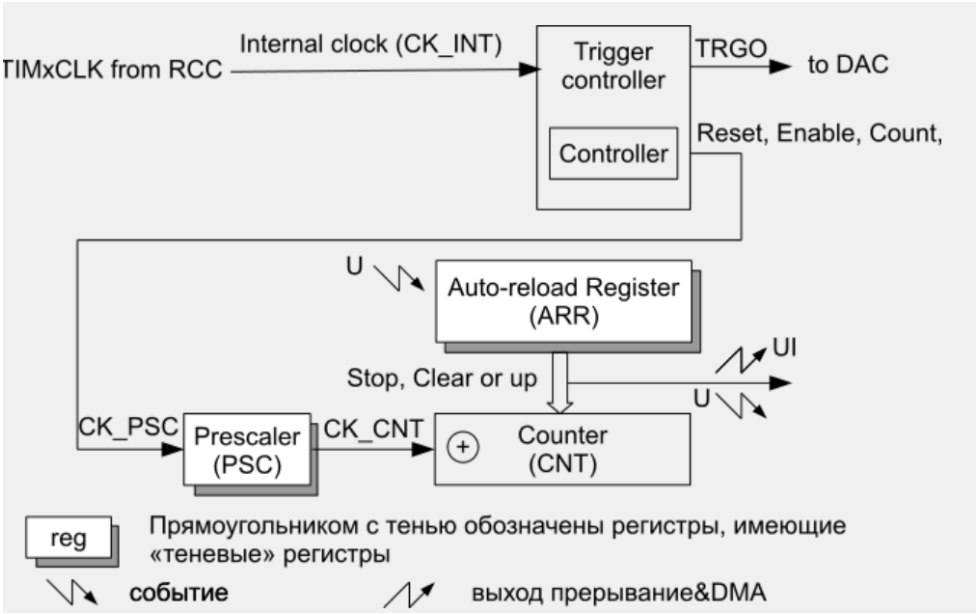

Рис. 1 – схема таймера TIM7

Частота такого таймера равна частоте процессора, деленное на значение Prescaler + 1. Если счётчик таймера достигает значения ARR, то он сбрасывается в 0, в этот момент и происходит прерывание.

Таким образом, при инициализации таймера выставляем Prescaler для получения частоты 1MHz, а ARR в 20000. Получаем время таймера, равное 2 секундам:

```
/* TIM7 init function */
void MX_TIM7_Init(void)
{
  TIM_MasterConfigTypeDef sMasterConfig = {0};

  htim7.Instance = TIM7;
  htim7.Init.Prescaler = 8390; // 1 MHz
  htim7.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim7.Init.Period = 20000; // 2000 ms
```

```
  htim7.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim7) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim7, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
}
```

Прерывание

Прерывание вызывается по истечению таймера TIM7. В самом обработчике последовательно включаются светодиоды: зеленый -> желтый -> красный. В переменной irq_led хранится номер следующего светодиода (1 – зеленый, 2 – желтый, 3 - красный) Далее представлена функция обработчика прерываний:

```
/**
  * @brief This function handles TIM7 global interrupt.
  */
void TIM7_IRQHandler(void)
{
  uint8_t prev_led = get_led();

  SDK_TRACE_Timestamp(PRINT, 1);
  SDK_TRACE_Print("Interrupt: change from led%d to led%d", prev_led, irq_led);
  SDK_TRACE_Timestamp(PRINT, 0);

  SDK_TRACE_Timestamp(prev_led, 0);
  SDK_TRACE_Timestamp(irq_led, 1);
  set_led(irq_led);

  if (LED1 == irq_led)
  {
         led_green();
         irq_led = LED2;
  }
  else if (LED2 == irq_led)
  {
         led_yellow();
         irq_led = LED3;
  }
  else // if (LED3 == irq_led)
  {
         led_red();
         irq_led = LED1;
  }
  HAL_TIM_IRQHandler(&htim7);
}
```

Светодиоды

Подсветку светодиодов было решено вынести в отдельные исходные файлы (led.h, led.c) для читабельности программы. Установка каждого цвета вынесена в отдельные функции *led_green, led_yellow, led_red*. Также были переработаны функции подсветки *led_dot, led_dash, led_delay*. Далее представлен исходный код led.h и led.c:

led.h

```
#ifndef INC_LED_H_
#define INC_LED_H_
#include <stdint.h>

uint8_t get_led();
void set_led(uint8_t val);

void led_green();
void led_yellow();
void led_red();
void led_dash();
void led_dot();
void led_delay();

#endif /* INC_LED_H_ */
```

led.c

```
#include "led.h"
#include "trace.h"
#define T 300
uint8_t led = LED1;

uint8_t get_led()
{
	return led;
}

void set_led(uint8_t led_val)
{
	led = led_val;
}

void led_green()
{
	led = LED1;
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, GPIO_PIN_RESET);
}

void led_yellow()
{
	led = LED2;
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, GPIO_PIN_RESET);
}

void led_red()
{
	led = LED3;
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, GPIO_PIN_SET);
}

void led_dash()
```

```
{
        led_green();

        SDK_TRACE_Timestamp(led, 1);
    HAL_Delay(2*T);
        SDK_TRACE_Timestamp(led, 0);
}

void led_dot()
{
        led_yellow();

        SDK_TRACE_Timestamp(led, 1);
    HAL_Delay(T);
        SDK_TRACE_Timestamp(led, 0);
}

void led_delay()
{
        led_red();

        SDK_TRACE_Timestamp(led, 1);
    HAL_Delay(T);
        SDK_TRACE_Timestamp(led, 0);
}
```

Результат

На рисунке 2 можно видеть результат выполнения программы в виртуальной лаборатории cLab. Как можно видеть, диаграмма событий отличается от аналогичной диаграммы в лабораторной работе №2. На диаграмме можно заметить синие события, после которых подсветка светодиодов меняется (при первом событии/прерывании сменяется цвет на зеленый, при втором на желтый, при третьем на красный).

Также в окне трассировочного буфера можно увидеть сообщения для логгирования прерываний.

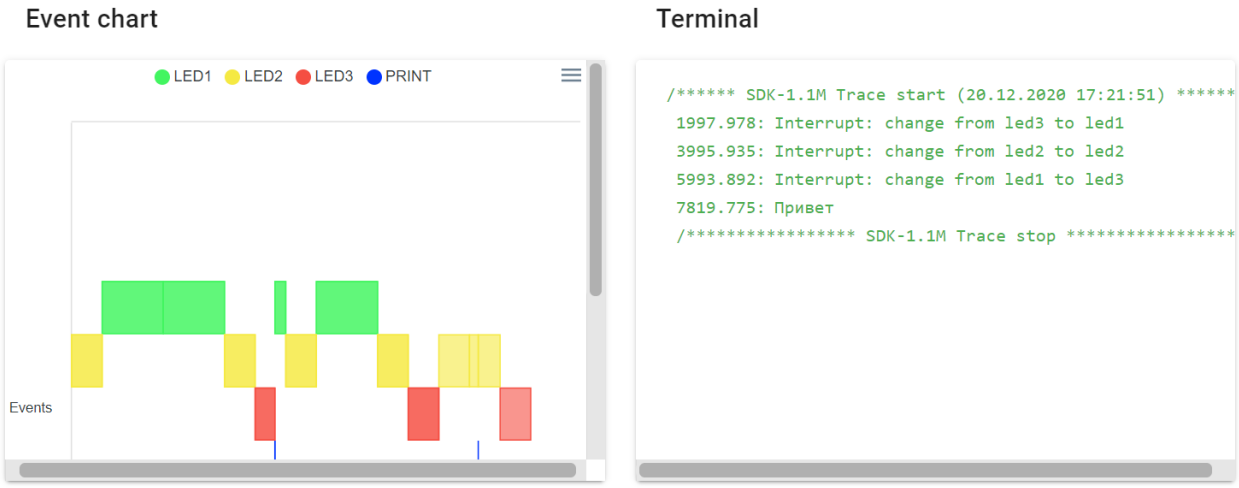

Рис. 2 – результат выполнения программы в виртуальной лаборатории cLab

Вывод

В ходе данной лабораторной работы был изучен механизм прерываний. Также была изучена работа таймеров, один из которых был использован для вызова прерываний.