УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Проектирование вычислительных систем

Лабораторная работа №1

Вариант 2: Интерфейсы ввода/вывода общего назначения (GPIO)

Преподаватель: Пинкевич Василий Юрьевич

Выполнили: Тарасов Александр Станиславович, Кульбако Артемий Юрьевич
P34112

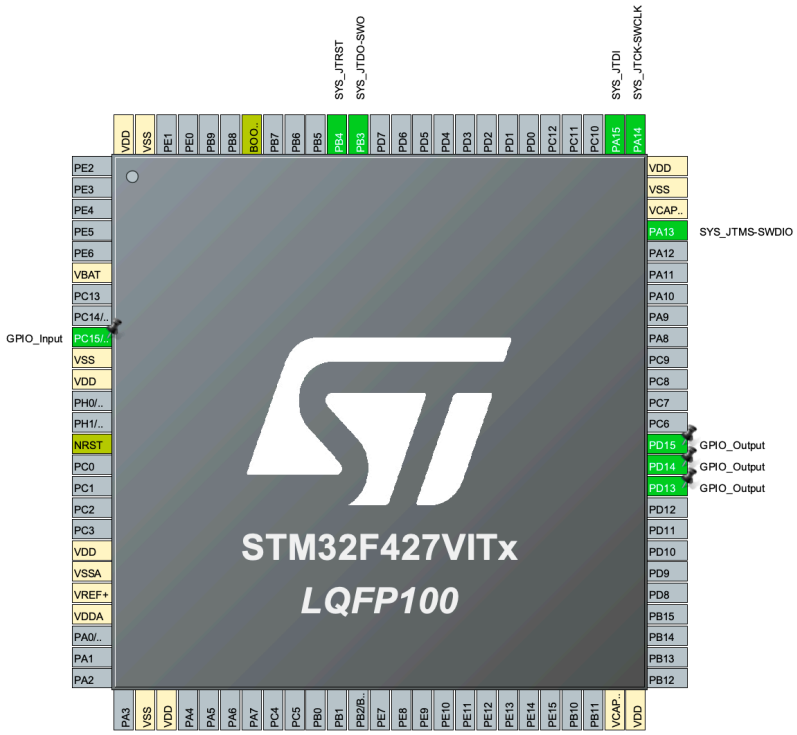
Санкт-Петербург
2021

Задание

Разработать и реализовать драйверы управления светодиодными индикаторами и обработки нажатий кнопки стенда SDK-1.1M (индикаторы и кнопка расположены на боковой панели стенда). Написать программу с использованием разработанных драйверов в соответствии с вариантом задания.

Реализовать простой имитатор гирлянды с переключением режимов. Должно быть реализовано не менее четырех последовательностей переключения светодиодов, обязательно с разной частотой мигания. По нажатию кнопки происходит переключение на следующий режим. Если режим последний в списке, нажатие кнопки должно переключать на первый режим. При повторном выборе режима анимация на светодиодах должна запускаться с того места, на котором была прервана переключением на следующий режим.

Используемые контакты



`PC15` - перехватывает нажатие кнопки

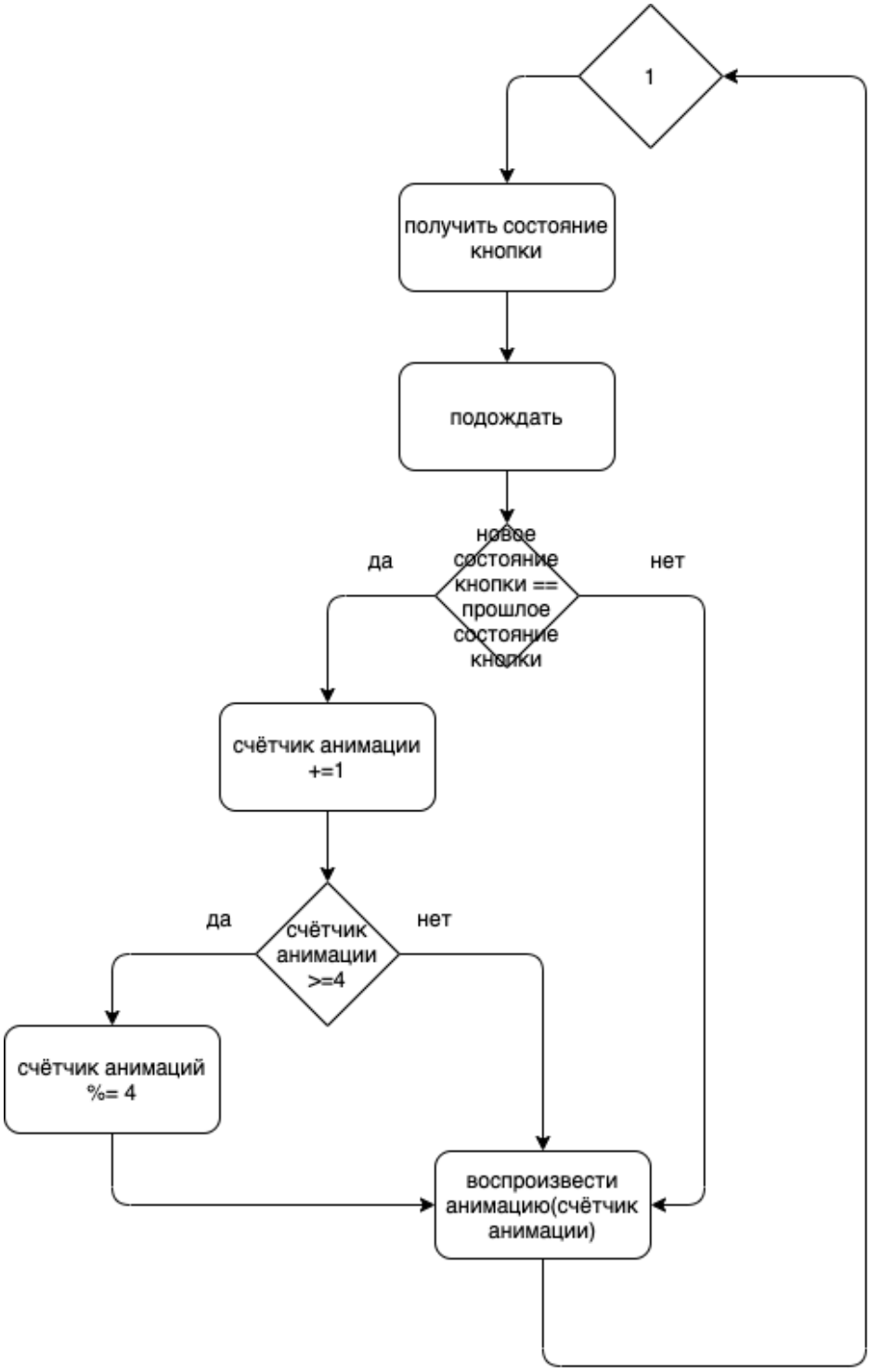
`PD13` - управляет зелёным светодиодом

`PD14` - управляет красным светодиодом

`PD15` - управляет жёлтым светодиодом

`PB3, PB4, PA13, PA14, PA15` - J-Tag для отладки

Блок-схема


1
получить состояние кнопки
подождать
новое состояние кнопки == прошлое состояние кнопки
да
нет
счётчик анимации +=1
счётчик анимации >=4
да
нет
счётчик анимаций %= 4
воспроизвести анимацию(счётчик анимации)

Драйвер

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "gpio.h"
#include <stdbool.h>
#include "LEDMode.c"
#include "utils.h"
#define MODES_AMOUNT 4

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/

/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

struct LEDMode MODES[] = {
                  { 6, { -1,GPIO_PIN_13,-1,GPIO_PIN_13,-1,GPIO_PIN_13 }, 1000,0 },
                  { 6, { -1,GPIO_PIN_14,-1,GPIO_PIN_14,-1,GPIO_PIN_14 }, 200,0 },
                  { 6, { -1,GPIO_PIN_15,-1,GPIO_PIN_15,-1,GPIO_PIN_15 }, 1000,0 },
                  { 6, { -1,GPIO_PIN_15,-1,GPIO_PIN_13,-1,GPIO_PIN_14 }, 500,0 }
};

bool activate_LEDMode(struct LEDMode* current_mode) {
        bool clicked_while_animation = false;
        uint32_t led = current_mode->code[current_mode->current_code_index];
        if (led == -1) {
                int start_time = HAL_GetTick();
                while (HAL_GetTick() < start_time + current_mode->delay_time)
                        if (is_btn_pressed() == true) clicked_while_animation = true;
        } else HAL_GPIO_TogglePin(GPIOD, led);
```

```
        current_mode->current_code_index++;
        return clicked_while_animation;
}

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void) {

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */

  int cur_mode_index = 0;
  bool flag;
  while (1) {
          flag = activate_LEDMode(&MODES[cur_mode_index ]);
          if (MODES[cur_mode_index ].current_code_index > MODES[cur_mode_index ].lenght)
MODES[cur_mode_index ].current_code_index = 0;
          if (flag || is_btn_pressed()) {
                  reset_LEDs();
                  cur_mode_index += 1;
                  if (cur_mode_index >= MODES_AMOUNT) cur_mode_index = 0;
          }
  }

    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = 15;
  RCC_OscInitStruct.PLL.PLLN = 216;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
```

```
  RCC_OscInitStruct.PLL.PLLQ = 4;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Activate the Over-Drive mode
  */
  if (HAL_PWREx_EnableOverDrive() != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5) != HAL_OK)
  {
    Error_Handler();
  }
}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

```
typedef struct LEDMode {
        int lenght;
        int code[6];
        int delay_time;
        int current_code_index;
};
```

```
#include <stdbool.h>
#include "gpio.h"
```

```
bool is_btn_pressed() {
        if (HAL_GPIO_ReadPin(GPIOC,GPIO_PIN_15) == 0)
                while (1) {
                        HAL_Delay(10);
                        if (HAL_GPIO_ReadPin(GPIOC,GPIO_PIN_15) == 1)
                                return true;
                }
        return false;
 }


void reset_LEDs() {
       HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, 0);
       HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, 0);
       HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, 0);
}
```

Вывод

В ходе выполнения лабораторной работы мы научились работать с светодиодами и кнопкой стенда SDK-1.1M на базе ARM-процессора STM32F427VI, разработали драйвер для управления им на языке C с библиотекой HAL.

При реализации возникло несколько проблем:

1. Обеспечить работу двух операций - управление светодиодами и считывание кнопок одновременно, без использования поток выполнения. Решение - не блокировать главный поток командой HAL_Delay(), а считывать в текущие время анимации функцией HAL_GetTick().
2. Вторая проблема - правильно обрабатывать нажатие кнопки, а именно менять анимации мигания при отпускании кнопки, а не нажатии во избежании двойного нажатия.
3. Третья проблема - сохранение состояния анимации при переключении режимов, решается выделением отдельного переменной для состояний.