Национальный исследовательский университет ИТМО

Факультет Программной Инженерии и Компьютерной Техники

Вариант №3

Лабораторная работа №1

«Интерфейсы ввода-вывода общего назначения (GPIO)»

По дисциплине:

«Проектирование вычислительных систем»

Работу выполнили:

Никонова Наталья Игоревна

Сенина Мария Михайловна

Группа: P34102

Преподаватель:

Пинкевич Василий Юрьевич

Санкт-Петербург

2023

Цель работы

1. Получить базовые знания о принципах устройства программировании микроконтроллеров.
2. Изучить устройство интерфейсов ввода-вывода общего назначения (GPIO) в микроконтроллерах и приемы использования данных интерфейсов.

Задание

Разработать и реализовать драйверы управления светодиодными индикаторами и чтения состояния кнопки стенда SDK-1.1М (расположены на боковой панели стенда). Контакты подключения кнопки и светодиодов должны быть настроены в режиме GPIO. Функции и другие компоненты драйверов должны быть универсальными, т.е. пригодными для использования в любом из вариантов задания и не должны содержать прикладной логики программы. Функции драйверов должны быть неблокирующими, то есть не должны содержать ожиданий события (например, нажатия кнопки). Также, в драйверах не должно быть пауз с активным ожиданием функция HAL\_Delay() и собственные варианты аналогичной реализации. Обработка нажатия кнопки в программе должна включать программную защиту от дребезга.

Написать программу с использованием разработанных драйверов в соответствии с вариантом задания.

Реализовать «передатчик» азбуки Морзе. Последовательность из нажатий кнопки (короткое – точка, длинное – тире) запоминается и после сигнала окончания ввода (долгая пауза) начинает «отправляться» при помощи зелѐного светодиода, последовательностью быстрых (точка) и долгих (тире) мерцаний. Во время ввода последовательности после каждого нажатия двухцветный светодиод должен коротким мерцанием индицировать, какой сигнал был введѐн (мигание жѐлтым – точка, мигание красным – тире).

Используемые контакты

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

* PC15 – перехватывает нажатия кнопки (обозначен как BUT, настроен на GPIO\_INPUT)
* PD13 – управляет зеленым светодиодом (обозначен как LEDG, настроен на GPIO\_OUTPUT)
* PD14 - управляет желтым светодиодом (обозначен как LEDY, настроен на GPIO\_OUTPUT)
* PD15 - управляет красным светодиодом (обозначен как LEDR, настроен на GPIO\_OUTPUT)
* PA10 – получает прерывания от таймера (обозначен как TIM1\_CH3)
* PB3, PB4, PA13, PA14, PA15 – J-Tag для отладки

Описание алгоритма

Блок-схема обработки прерывания таймера

Блок-схема основной логики

Исходный код

Драйвер вывода (светодиоды)

**void** **turn\_on\_green\_led**() {

HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_13, *GPIO\_PIN\_SET*);

}

**void** **turn\_off\_green\_led**() {

HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_13, *GPIO\_PIN\_RESET*);

}

**void** **turn\_on\_yellow\_led**() {

HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_14, *GPIO\_PIN\_SET*);

}

**void** **turn\_off\_yellow\_led**() {

HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_14, *GPIO\_PIN\_RESET*);

}

**void** **turn\_on\_red\_led**() {

HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_15, *GPIO\_PIN\_SET*);

}

**void** **turn\_off\_red\_led**() {

HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_15, *GPIO\_PIN\_RESET*);

}

Драйвер ввода (кнопка, реализовано через опрос по прерыванию от таймера)

**void** **HAL\_TIM\_PeriodElapsedCallback**(TIM\_HandleTypeDef \*htim)

{

**if**(htim->Instance==TIM1)

{

**int** but = HAL\_GPIO\_ReadPin(BUT\_GPIO\_Port, BUT\_Pin);

but = !but;

**if**(is\_reading == 1){

**if**(but == 1 && noisy == 0){

noisy = 1;

} **else** **if**(but == 1 && noisy == 1) {

is\_pressed = 1;

count\_tick++;

} **else** **if**(noisy == 1 && but == 0){

noisy = 0;

} **else** **if**(but == 0 && noisy == 0){

is\_pressed = 0;

count\_tick++;

}

}

}

}

Основная логика

**#define** LONG\_PERIOD\_CT 125

**#define** BUFFER\_SIZE 2048

**#define** DEFAULT\_DELAY 200

**#define** ERROR\_DELAY 600

**#define** ERROR\_COUNT 2

**#define** COMMA\_DELAY DEFAULT\_DELAY

**#define** DASH\_DELAY 1000

**int** is\_pressed = 0; // нажата ли кнопка, устанавливается в таймере

**int** is\_wait\_unpressed = 0; // ожидаем ли отпускание кнопки

**int** is\_reading = 1; // есть ли что в памяти чтобы читать (можно pointer сделать глобальным и проверять на ноль

**int** count\_tick = 0; // кол-во тиков таймера в текущем состоянии is\_pressed, устанавливается в таймере

**int** noisy = 0; // доп проверка для дребезга

**void** **send\_message**(**int** buffer[], **int** pointer) {

**for** (**int** i = 0; i < pointer; i++) {

turn\_on\_green\_led();

**if** (1 == buffer[i]) {

HAL\_Delay(DASH\_DELAY);

} **else** {

HAL\_Delay(COMMA\_DELAY);

}

turn\_off\_green\_led();

HAL\_Delay(DEFAULT\_DELAY);

}

}

/\* USER CODE BEGIN WHILE \*/

**int** buffer[BUFFER\_SIZE] = {0};

**int** pointer = 0;

HAL\_TIM\_Base\_Start\_IT((TIM\_HandleTypeDef \*)&htim1);

**while** (1)

{

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

**printf**("%i %i", is\_pressed, is\_wait\_unpressed);

**if** (0 == is\_pressed && 0 == is\_wait\_unpressed) {

**if** (count\_tick > LONG\_PERIOD\_CT && 0 != pointer){

is\_reading = 0;

send\_message(buffer, pointer);

pointer = 0;

is\_reading = 1;

}

} **else** **if** (0 == is\_pressed && 1 == is\_wait\_unpressed) {

is\_wait\_unpressed = 0;

**if** (count\_tick > LONG\_PERIOD\_CT) {

turn\_on\_red\_led();

HAL\_Delay(DEFAULT\_DELAY);

turn\_off\_red\_led();

buffer[pointer] = 1;

} **else** {

turn\_on\_yellow\_led();

HAL\_Delay(DEFAULT\_DELAY);

turn\_off\_yellow\_led();

buffer[pointer] = 0;

}

count\_tick = 0;

pointer++;

**if** (BUFFER\_SIZE == pointer) {

**for** (**int** i = 0; i < ERROR\_COUNT; i++) {

turn\_on\_red\_led();

turn\_on\_green\_led();

HAL\_Delay(ERROR\_DELAY);

turn\_off\_red\_led();

turn\_off\_green\_led();

HAL\_Delay(ERROR\_DELAY);

}

send\_message(buffer, pointer);

pointer = 0;

}

} **else** **if** (1 == is\_pressed && 0 == is\_wait\_unpressed) {

is\_wait\_unpressed = 1;

count\_tick = 0;

}

}

/\* USER CODE END 3 \*/

Выводы

В ходе выполнения лабораторной работы мы вспомнили, как писать программы под микроконтроллеры семейства STM32, научились их дебажить и попробовали настроить таймер и реализовать логику неблокирующего чтения через опросы по прерыванию от него.