Лабораторная работа №2. «Программирование дискретных портов ввода/вывода (работа со светодиодами и кнопками)»

Задание

Разработать и реализовать драйверы GPIO (управление светодиодными индикаторами и обработка нажатий кнопки контроллера SDK-1.1M). Написать программу с использованием разработанных драйверов по алгоритму, соответствующему варианту задания.

Общие сведения

Интерфейс ввода/вывода общего назначения (general-purpose input/output, GPIO) – базовый интерфейс взаимодействия компьютерной системы с внешним миром. С его помощью чаще всего подключаются такие внешние элементы как светодиоды, кнопки, переключатели, осуществляется управление периферийными устройствами и т.д.

GPIO контакты группируются в порты, обычно по 8, 16 или 32 линий. Одни и те же контакты GPIO могут выступать в роли входа или выхода. Режим входа обычно включен по умолчанию, для работы в качестве выхода необходимо настроить соответствующую ножку на выход при помощи управляющего регистра. Каждый контакт настраивается индивидуально – на одном порте могут чередоваться входы и выходы в любых комбинациях.

Управление выходом может быть прямым, из программы (записать 0 или 1), может осуществляться при помощи внутренних устройств (например, реализация широтно-импульсной модуляции на базе таймеров).

В учебном стенде SDK-1.1M с процессорным модулем на базе STM32 есть одна кнопка, подключённая к PC15, и два управляемых светодиода: зеленый, подключенный к PD13, и двухцветный красный/желтый, подключенный к контактам PD14, PD15 (Рисунок 1).

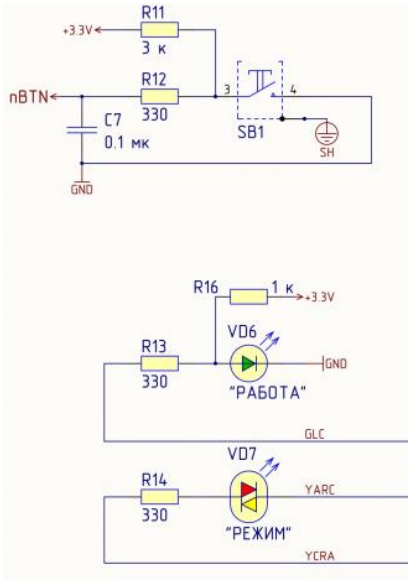

Рис. 1. Кнопка и управляемые светодиоды на принципиальной схеме стенда

Работа с GPIO

Сперва, необходимо произвести настройку Pinout в STM32CubeMX: включить контакты PD13-PD15 в режим GPIO_Output (рисунок 2).

Добавить Debug JTAG (5 pins): Вкладка Pinouts & Configuration -> System Core -> SYS-> Debug -> JTAG (5 pins).

Включить тактовый генератор RCC: Вкладка Pinouts & Configuration -> System Core -> RCC-> High Speed Clock -> Crystal/Ceramic Resonator.

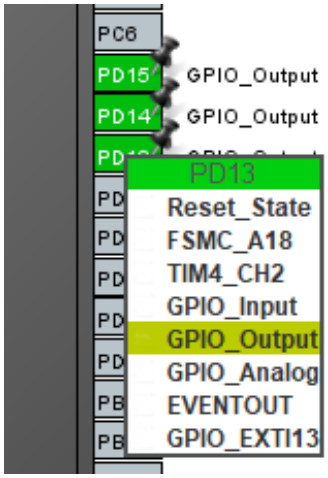

Рис. 2. Окно настройки pinout

Для работы с GPIO на STM32 используется библиотека HAL (Hardware Abstract Layer). STM32CubeMX при генерации проекта добавляет в код функцию инициализации GPIO `MX_GPIO_Init`.

```
static void MX_GPIO_Init(void)
{
GPIO_InitTypeDef GPIO_InitStruct = {0};
__HAL_RCC_GPIOH_CLK_ENABLE();
__HAL_RCC_GPIOD_CLK_ENABLE();
__HAL_RCC_GPIOA_CLK_ENABLE();
__HAL_RCC_GPIOB_CLK_ENABLE();
HAL_GPIO_WritePin(GPIOD,GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15,
GPIO_PIN_RESET);
GPIO_InitStruct.Pin = 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_FREQ_LOW;
HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);
}
```

Для переключения состояния светодиода можно использовать стандартную функцию из библиотеки HAL: `HAL_GPIO_TogglePin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);`

- `GPIOx` – буква x предназначена для выбора периферийного устройства, может быть (A-K). Для управления светодиодом нужно использовать букву D;
- `GPIO_Pin` – номер управляемого контакта. Например: `GPIO_Pin_13`.

Пример кода:

```
while (1)
  {
       HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_13);
       HAL_Delay(500);
       HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_13);
       HAL_Delay(500);
       HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_14);
       HAL_Delay(500);
       HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_14);
       HAL_Delay(500);
       HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_15);
       HAL_Delay(500);
       HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_15);
       HAL_Delay(500);
  }
```

Этот код поочередно включает и выключает светодиоды с задержкой в 500 мс.

Варианты заданий

Вариант 1.

Сымитировать работу светофора пешеходного перехода. В режиме по умолчанию светофор переключает цвета в следующем режиме: зелёный-мигающий зелёный-жёлтый-карсный-зелёный…, при этом период красного значительно больше. При нажатии кнопки происходит переключение с красного на зелёный, но два включения «зелёных» не могут идти сразу друг за другом – между ними должен быть период, больше или равный ¼ периода красного.

Вариант 2.

Реализовать простой имитатор гирлянды с переключением режимов. Должно быть реализовано не менее 4х последовательностей переключения диодов, обязательно с разной частотой мигания. Нажатие кнопки реализует переключение на другой режим. Если режим последний в списке, нажатие кнопки должно переключать на первый режим.

Вариант 3.

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

Вариант 4.

Реализовать двоичный двухразрядный счётчик на светодиодах, с возможностью вычитания. Быстрое нажатие кнопки должно прибавлять 1 к отображаемому на диодах двоичному числу, по переполнению счётчика должна отображаться простая анимация: мигание обоими светодиодами, затем количество миганий зелёным светодиодом, равное количеству переполнений. Долгое нажатие кнопки должно вычитать 1 от отображаемого на диодах двоичного числа, если происходит вычитание из нуля – уменьшается на 1 количество переполнений, и отображается анимация, аналогичная анимации при переполнении.

Вариант 5.

Реализовать «кодовый замок». После ввода единственно верной последовательности из не менее чем восьми коротких и длинных нажатий должен загореться зелёный светодиод, индицирующий «открытие» замка. Светодиод горит некоторое время, потом гаснет и система вновь переходит в «режим ввода». Каждый неправильно введённый элемент последовательности должен сопровождаться миганием красного светодиода и сбросом в «начало», каждый правильный – миганием жёлтого. После трёх неправильных вводов начинает мигать красный диод, и через некоторое время возвращается в «режим ввода».

Требования к выполнению работы

Проект должен быть в виде отдельных файлов, каждый из которых содержит отдельный драйвер или цельную, логически законченную часть программы.

Код должен быть структурированным, хорошо читаемым. Названия переменных и функций должны быть осмысленными и поясняющими назначение данных переменных и функций. На «верхнем уровне» программы (функция main) должна быть только прикладная часть, вся логика работы с аппаратурой скрывается от пользователя «верхнего уровня» и реализуется на уровне драйверов.

Не должно быть одинаковых или сильно похожих блоков кода, желательно реализовывать их с помощью отдельных функций. Крайне желательно минимизировать количество вложенных циклов и свести их к необходимому минимуму.

Требования к отчёту

Отчёт должен содержать:

1. Титульный лист с номером лабораторной работы, её названием, ФИО и группой исполнителей.
2. Номер варианта, задание.
3. Блок-схема прикладного алгоритма.
4. Подробное описание инструментария, который использовался в работе.
5. Исходные коды прикладной программы и использованных библиотек.
6. Выводы, в которых описываются проблемы, которые возникли в ходе работы, и способы их решения.

Защита лабораторной работы

На защите задаются вопросы по:

1. Теории.
2. Инструментальным средствам, которые использовались для работы.
3. Принципиальной схеме стенда SDK.
4. Исходным кодам, включая библиотеки.