**Лабораторная работа №2: Порты ввода/вывода(General-purposeinput/output). Библиотека SPL.**

**Цель работы:** Изучить основные принципы работы GPIO. Ознакомиться с библиотекой StandartPeriphlibrary(SPL) и ее использованием.

**Теоретический материал**

Чтобы взаимодействовать с внешним миром – получать и передавать информацию – у микроконтроллера имеются порты ввода/вывода. У микроконтроллера STM32F051R8, используемого в лабораторных работах, 5 портов, содержащих в сумме 55 выводов.

**Режимы работы выводов**

Пользовательские выводы могут быть сконфигурированы в один из следующих режимов работы:

Input floating - это высокоимпедансный вход (Hi-Z), он же плавающий, так как не имеет подтяжки к питанию или земле, поэтому его логическое состояние (1 или 0) всегда определяется напряжением на нём.

Input pull-up - это вход с подтяжкой к питанию, т.е. между входом и питанием включен подтягивающий резистор (номинала порядка кОм). Подтягивающий резистор позволяет выходу находиться либо в высоком (“1”, подтянут к питанию), либо в низком (“0”, подтянут к земле) состоянии, когда к выходу не приложено внешнее напряжение. Это позволяет избежать спонтанных появлений 0 и 1 на входе.

Input-pull-down - по аналогии с предыдущим, этот вход — с подтяжкой к земле. При отсутствии напряжения имеет низкое логическое состояние (“0”).

Analog – это аналоговый вход или выход. Это касается отдельных периферийных блоков(АЦП, ЦАП).

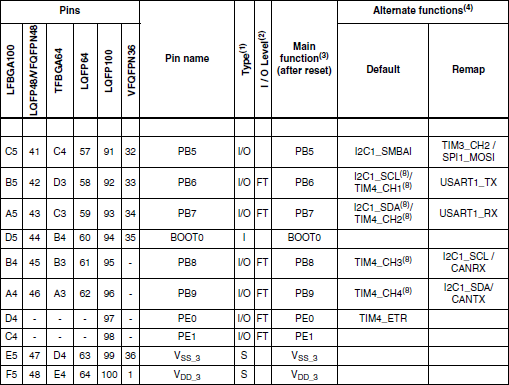
Outputopen-drain - "выход с открытым стоком".

Outputpush-pull – “двухтактный выход”. Самый часто используемый тип выхода: подали 0 — выход подключился к земле, 1 — подключился к питанию.

Alternate function push-pull - уже описанный ранее двухтактный выход, только для альтернативной функции – периферии (SPI, USART…)

Alternate function open-drain -соответственно выход с открытым стоком, так же для альтернативной функции.

Альтернативные функции, соответствующие выводам, описаны в Datasheet на микроконтроллер (рисунок 4.1).

Рисунок 4.1 Альтернативные функции

Перед использованием вывода он должен быть сконфигурирован в один из вышеперечисленных режимов работы. Это можно сделать несколькими способами:

-Непосредственно через регистры микроконтроллера

-Используя библиотеку StandartPeripheralLibrary (SPL)

В данном лабораторном курсе для настройки и работы с периферией будет использоваться библиотека SPL.

StandartPeripheralLibrary – библиотека, созданная компанией STMicroelectronics. Она содержит функции и структуры для настройки и работы с периферией. Эти функции и структуры берут на себя работу с регистрами, значительно упрощая написание прошивки.

**Структура программы**

В качестве примера использования GPIO напишем программу, в которой с помощью микроконтроллера будет загораться светодиод после нажатия кнопки.

Все функции, классы и константы отвечающие за GPIO представлены в файлах *stm32f0xx\_gpio.h*и *stm32f0xx\_gpio.c*.

-Инициализация кнопки и светодиода

Нужная кнопка подключена к 4-ому выводу порта C, а светодиод – к 8-ому выводу порта A. (В файле «Распиновка» описано к каким выводам подключены кнопки, светодиоды и прочие детали макета)

За конфигурацию портов отвечает структура *GPIO\_InitTypeDef*, поэтому объявим переменную такого типа:

GPIO\_InitTypeDefport;

Как и для любой другой периферии сначала необходимо включить тактирование используемых портов (в данном случае Aи C):

RCC\_AHBPeriphClockCmd(RCC\_AHBPeriph\_GPIOA, ENABLE);

RCC\_AHBPeriphClockCmd(RCC\_AHBPeriph\_GPIOC, ENABLE);

Функции для работы с тактированием представлены в файлах *stm32f0xx\_rcc.h*и *stm32f0xx\_rcc.c*.

Структура *GPIO\_InitTypeDef*содержит переменные:

*GPIO\_Pin*- отвечает за номер вывода порта, которую мы хотим настроить, *GPIO\_Speed*–отвечает за скорость работы порта,

*GPIO\_Mode -*отвечает за режим работы,

*GPIO\_PuPd*– отвечает за подтяжку к земле или питанию.

**Ход работы**

**Задание на лабораторную работу**

1. Ознакомиться с теоретическими сведениями.

2.Написать, отладить и запустить программу.

3. Выполнить индивидуальное задание.

**Текст программы**

voidinitAll(void); voidinitAll()

{

GPIO\_InitTypeDef port;

void GPIO\_Init(GPIO\_TypeDef\* GPIOx, GPIO\_InitTypeDef\* GPIO\_InitStruct); // RCC\_AHBPeriphClockCmd(RCC\_AHBPeriph\_GPIOA, ENABLE); RCC\_AHBPeriphClockCmd(RCC\_AHBPeriph\_GPIOC, ENABLE); GPIO\_StructInit(&port);

port.GPIO\_PuPd = GPIO\_PuPd\_DOWN; port.GPIO\_Mode = GPIO\_Mode\_IN; port.GPIO\_Pin = GPIO\_Pin\_4; port.GPIO\_Speed = GPIO\_Speed\_2MHz; GPIO\_Init(GPIOC, &port); port.GPIO\_Mode = GPIO\_Mode\_OUT; port.GPIO\_OType = GPIO\_OType\_PP; port.GPIO\_Pin = GPIO\_Pin\_8; port.GPIO\_Speed = GPIO\_Speed\_2MHz; GPIO\_Init(GPIOA, &port);

}

int main()

{

uint8\_t buttonState = 0;

initAll(); //инициализациявыводов while(1)

{

buttonState = GPIO\_ReadInputDataBit(GPIOA, GPIO\_Pin\_0); if (buttonState == 1) //проверкасостояниякнопки

{

GPIO\_SetBits(GPIOC, GPIO\_Pin\_8);//включениесветодиода

}

else

{

GPIO\_ResetBits(GPIOC, GPIO\_Pin\_8);//отключениесветодиода

}

}

}

**Индивидуальные задания**

1.Реализовать сумматор двоичных чисел

В ходе работы микроконтроллер должен считывать с переключателей два 4-х значных двоичных числа и выводить их сумму на светодиоды.

2.Реализовать декодер

Входе работы микроконтроллер должен считывать с переключателей двоичное число

–номер светодиода, который должен гореть.