Санкт-Петербургский Политехнический Университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Аппаратные платформы встраиваемых систем

Отчет по лабораторной работе №1

На тему **«Работа с линиями ввода-вывода общего назначения»**

Работу выполнили:

Студенты гр. 13541/1 Онищенко Д.И.

Шаменов А.А.

Преподаватель: Васильев А.Е.

Санкт – Петербург

2019

Содержание

Цель работы:

Получение навыков работы с портами ввода-вывода общего назначения МК STM32, светодиодами и механическими кнопками.

Подготовка к работе:

1. Подготовить проект в IARWE, согласно документу IAR Project for IAR SK Board
2. Ознакомиться со схемой платы IAR SK (STM32F407ZG-board-schematic.pdf)
3. Ознакомиться с документацией МК STM32F407 (STM32F4xx_RM.pdf)
4. Используя полученные теоретические навыки, спроектировать систему, которая будет поочередно мигать четырьмя светодиодами при нажатой кнопке (USER_BUTTON)

Теоретическая информация:

Многие выводы МК STM32F407 представляют собой цифровые линии ввода или вывода. Каждую такую линию можно программным путем сконфигурировать как цифровой вход, либо цифровой выход, и использовать для взаимодействия с внешними схемами. Для удобства использования линии ввода-вывода объединены в порты по 16 линий. Такие порты называют портами ввода-вывода общего назначения.

К линиям, сконфигурированным как цифровые входы, подключают механические кнопки, выключатели, контакты реле и т.д. При наличии или отсутствии сигнала на таком входе, микроконтроллер сможет, например, «узнать» состояние кнопки (нажата / не отжата).

В целом, при помощи этих линий, микроконтроллер получает информацию от подключенных к нему устройств. Линии,

сконфигурированные как цифровые выходы, позволяют выдавать управляющие сигналы для подключенных к микроконтроллеру устройств. Таким сигналом через соответствующую схему можно запустить электродвигатель, включить электромагнитное реле или подсветить диод.

На самом низком уровне работа с портами ввода-вывода (да и со всеми другими периферийными устройствами) осуществляется с помощью специальных регистров микроконтроллера. Эти регистры доступны как ячейки памяти, расположенные по определенным адресам. Используя данные адреса, можно записывать в регистры определенные значения, задавая требуемую конфигурацию. Через другие регистры можно получить данные от периферийных устройств.

Таблица 1 - Порты микроконтроллера STM32F407 и их адреса в памяти:

0x4002 2800	0x4002 2BFF	GPIOK
0x4002 2400	0x4002 27FF	GPIOJ
0x4002 2000	0x4002 23FF	GPIOI
0x4002 1C00	0x4002 1FFF	GPIOH
0x4002 1800	0x4002 1BFF	GPIOG
0x4002 1400	0x4002 17FF	GPIOF
0x4002 1000	0x4002 13FF	GPIOE
0x4002 0C00	0x4002 0FFF	GPIOD
0x4002 0800	0x4002 0BFF	GPIOC
0x4002 0400	0x4002 07FF	GPIOB
0x4002 0000	0x4002 03FF	GPIOA

Для облегчения труда программиста многие разработчиками микроконтроллеров предоставляют специальные библиотеки функций для работы с периферийными устройствами. В лабораторных работах будет использоваться Standard Peripheral Library, разработанная специально для МК семейства STM32F4. В этой библиотеке есть функции для работы со всеми

устройствами в составе микроконтроллера. Не нужно напрямую обращаться к регистрам, достаточно вызвать подходящую функцию, что гораздо удобней. Исходники каждой функции доступны, что позволяет при необходимости посмотреть, как осуществляется работа с периферией.

В данной лабораторной работе будем использовать следующую структуру типа GPIO_InitTypeDef. Вот пример ее описания:

```
1.    GPIO_InitTypeDef GPIO_InitStructure;
```

Тип GPIO_InitTypeDef описан в заголовке stm32f4xx_gpio.h и выглядит следующим образом:

```
1. typedef struct
2. {
3. uint32_t GPIO_Pin;
4. GPIOMode_TypeDef GPIO_Mode
5. GPIOSpeed_TypeDef GPIO_Speed;
6. GPIOOType_TypeDef GPIO_OType;
7. GPIOPuPd_TypeDef GPIO_PuPd;
8. }
9. GPIO_InitTypeDef;
```

Перед использованием структуры для конфигурации очередного набора линий ввода-вывода желательно инициализировать ее следующим образом, дабы избежать ошибок при конфигурировании

```
1. GPIO_Init (&GPIO_InitStructure);
```

GPIO_Mode_IN & GPIO_Mode_OUT - конфигурирование линии на вход или выход соотвественно

GPIO_Mode_AF - конфигурирование линии для работы в режиме альтеранативной функции

GPIO_Mode_AN - режим аналоговой линии выбирается, если требуется работать с АЦП, ЦАП, аналоговым компаратором или внешним низкочастотным кварцевым резонатором.

В поле GPIO_Speed типа GPIOSpeed_TypeDef указывают примерную скорость работы линии, т.е. какой частоты сигнал может через нее проходить.

Ход работы

Используя полученные теоретические навыки, спроектируем систему, работающую с кнопкой USER_BUTTON и диодами STAT1-4 платы IAR SK. Для этого обратимся дополнительно к схеме платы, и, найдя, к каким именно портами и пинам этих портов подключены те или иные периферийные устройства, составим программу.

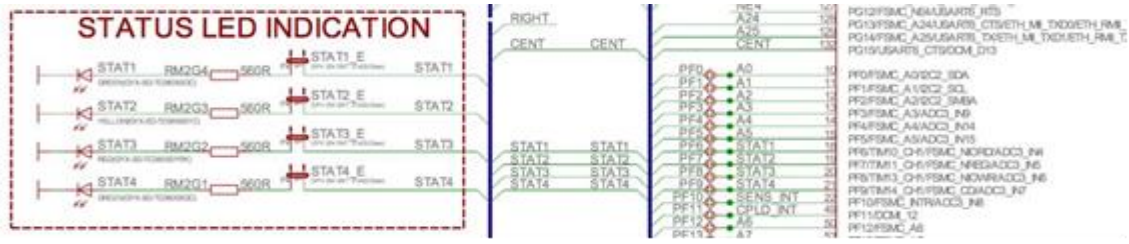

Рисунок 1 - Фрагмент схемы платы IAR SK, с информацией о диодах

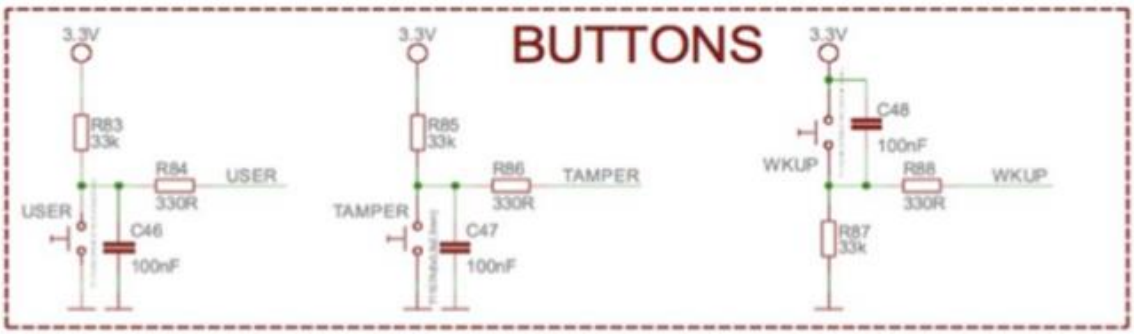

Рисунок 2 – Фрагменты схемы платы IAR SK, с информацией о кнопках

В подключенном файле board_config.h определены константы с номерами портов, которые используются далее в нашей программе.

Благодаря подключенным конфигурациям можно настроить порт и пин, определить первый светодиод, на режим вывода, тактируя его частотой в 100 МГц.

```
1. #include "stm32f4xx.h"
2. #include "stm32f4xx_conf.h"
3. #define _STM_IAR_BOARD_
4. #include "board_config.h"
5. #define DELAY 200000
6. static void Delay ( uint32_t delay);
7. int main()
8. {
9.     GPIO_InitTypeDef  GPIO_InitStruct;
10.    RCC_AHB1PeriphClockCmd(F_ROC, ENABLE);
11.    GPIO_InitStruct.GPIO_Pin = LED_PIN1;
12.    GPIO_InitStruct.GPIO_Mode = GPIO_Mode_OUT;
13.    GPIO_InitStruct.GPIO_OType = GPIO_OType_PP;
14.    GPIO_InitStruct.GPIO_Speed = GPIO_Speed_100MHz;
15.    GPIO_InitStruct.GPIO_PuPd = GPIO_PuPd_NOPULL;
16.    GPIO_Init(F_PORT, &GPIO_InitStruct);
17.    while(1){
18.        GPIO_ToggleBits(LED_PORT, LED_PIN);
19.    }
20.}
```

В бесконечном цикле применяется команда для переключения светодиода. В результате светодиод хоть и зажигается на плате, однако его

мигание незаметно, так как происходит слишком быстро. Для этого мы определили дополнительную функцию задержки:

```
1. static void Delay (uint32_t delay){
2.    while (--delay) {
3.    __NOP();
4.    }
5. }
```

Вызывая эту функцию в цикле после каждого переключения для больших чисел, можно будет заметить, что мигание светодиода замедлилось.

Теперь необходимо, чтобы мы могли управлять светодиодами с кнопок. Для этого необходимо инициализировать порт и пин для кнопки.

Определим необходимую конфигурацию для кнопки WKUP:

```
1. RCC_AHB1PeriphClockCmd(A_RCC, ENABLE);
2. GPIO_InitStruct.GPIO_Pin = WKUP_BTN;
3. GPIO_InitStruct.GPIO_Mode = GPIO_Mode_IN;
4. GPIO_Init(A_PORT, &GPIO_InitStruct);
```

И для кнопки USER

```
1. RCC_AHB1PeriphClockCmd(G_ROC, ENABLE);
2. GPIO_InitStruct.GPIO_Pin = USER_BTN;
3. GPIO_InitStruct.GPIO_Mode = GPIO_Mode_IN;
4. GPIO_Init(G_PORT, &GPIO_InitStruct);
```

Необходимо сделать переключение светодиодов при помощи кнопок. Добавим в инициализацию порта светодиодов, в поле пинов, номера всех пинов светодиодов через знак |, что позволит применить одинаковые настройки на все светодиоды:

```
1. GPIO_InitStruct.GPIO_Pin = LED_PIN1 | LED_PIN2 | LED_PIN3 |
2. LED_PIN4 | LED_PIN5;
```

Полный текст реализованной программы представлен в приложении А. Здесь убрана функция Delay, так как раз мы ждем, что кнопку опустят, то задержка не нужна.

Таким образом, при каждом переключении кнопки мы гасим по очереди светодиоды, зажигая соседние, при нажатии кнопки USER – слева направо, при нажатии кнопки WKUP – справа налево.

Вывод

В ходе работы были получены навыки по взаимодействию с портами ввода-вывода на примере платы МК STM32F407 и реализована программа согласно заданию. Также было произведено ознакомление с документацией на плату. Стоит отметить, что даже такая работа, как конфигурация портов ввода-вывода представляет собой достаточно нетривиальную задачу, так как для работы с периферией необходимо ее затактировать и только потом использовать ее в проекте.

Приложение А - Полный текст программы

```
1. #include "stm32f4xx.h"
2. #include "stm32f4xx_conf.h"
3. #define _STM_IAR_BOARD_
4. #include "board_config.h"
5. #define DELAY 200000
6. static void Delay ( uint32_t delay);
7. int main()
8. {
9.     GPIO_InitTypeDef  GPIO_InitStruct;
10.    uint8_t flag;
11.    uint8_t flag1;
12.    RCC_AHB1PeriphClockCmd(F_ROC, ENABLE);
13.    GPIO_InitStruct.GPIO_Pin = LED_PIN1 | LED_PIN2 | LED_PIN3
14.    | LED_PIN4 | LED_PIN5;
15.    GPIO_InitStruct.GPIO_Mode = GPIO_Mode_OUT;
16.    GPIO_InitStruct.GPIO_OType = GPIO_OType_PP;
17.    GPIO_InitStruct.GPIO_Speed = GPIO_Speed_100MHz;
18.    GPIO_InitStruct.GPIO_PuPd = GPIO_PuPd_NOPULL;
19.    GPIO_Init(F_PORT, &GPIO_InitStruct);
20.    RCC_AHB1PeriphClockCmd(G_ROC, ENABLE);
21.    GPIO_InitStruct.GPIO_Pin = USER_BTN;
22.    GPIO_InitStruct.GPIO_Mode = GPIO_Mode_IN;
23.    GPIO_InitStruct.GPIO_OType = GPIO_OType_PP;
24.    GPIO_InitStruct.GPIO_Speed = GPIO_Speed_100MHz;
25.    GPIO_InitStruct.GPIO_PuPd = GPIO_PuPd_NOPULL;
26.    GPIO_Init(G_PORT, &GPIO_InitStruct);
27.    RCC_AHB1PeriphClockCmd(A_RCC, ENABLE);
28.    GPIO_InitStruct.GPIO_Pin = WKUP_BTN;
29.    GPIO_InitStruct.GPIO_Mode = GPIO_Mode_IN;
30.    GPIO_InitStruct.GPIO_OType = GPIO_OType_PP;
31.    GPIO_InitStruct.GPIO_Speed = GPIO_Speed_100MHz;
32.    GPIO_InitStruct.GPIO_PuPd = GPIO_PuPd_NOPULL;
33.    GPIO_Init(A_PORT, &GPIO_InitStruct);
34.    uint32_t currentLed;
35.    uint32_t previusLed;
36.    currentLed = LED_PIN1;
37.    uint32_t nextLed;
38.    nextLed = LED_PIN2;
39.    while(1){
40.        flag1 = GPIO_ReadInputDataBit(GPIOA, WKUP_BTN);
41.        flag = GPIO_ReadInputDataBit(GPIOG, USER_BTN);
42.        if (!flag){
43.            if (nextLed == LED_PIN1) {
44.               currentLed = LED_PIN2;
45.               previusLed = LED_PIN3;
46.               GPIO_ResetBits(F_PORT, previusLed);
47.               GPIO_SetBits(F_PORT,currentLed);
48.               nextLed = LED_PIN4;
49.            } else if (nextLed == LED_PIN4) {
```

```
50.                currentLed = LED_PIN1;
51.                previusLed = LED_PIN2;
52.                GPIO_ResetBits(F_PORT, previusLed);
53.                GPIO_SetBits(F_PORT,currentLed);
54.                nextLed = LED_PIN3;
55.             } else if (nextLed == LED_PIN3) {
56.                currentLed = LED_PIN4;
57.                previusLed = LED_PIN1;
58.                GPIO_ResetBits(F_PORT, previusLed);
59.                GPIO_SetBits(F_PORT,currentLed);
60.                nextLed = LED_PIN2;
61.             }else if (nextLed == LED_PIN2) {
62.                currentLed = LED_PIN3;
63.                previusLed = LED_PIN4;
64.                GPIO_ResetBits(F_PORT, previusLed);
65.                GPIO_SetBits(F_PORT,currentLed);
66.                nextLed = LED_PIN1;
67.     }
68.       while (!flag) {
69.         flag = GPIO_ReadInputDataBit(GPIOG, USER_BTN);
70.       }
71.     }
72.     if (flag1){
73.        if (nextLed == LED_PIN1) {
74.           currentLed = LED_PIN4;
75.           previusLed = LED_PIN3;
76.           GPIO_ResetBits(F_PORT, previusLed);
77.           GPIO_SetBits(F_PORT,currentLed);
78.           nextLed = LED_PIN2;
79.        } else if (nextLed == LED_PIN2) {
80.           currentLed = LED_PIN1;
81.           previusLed = LED_PIN4;
82.           GPIO_ResetBits(F_PORT, previusLed);
83.           GPIO_SetBits(F_PORT,currentLed);
84.           nextLed = LED_PIN3;
85.        } else if (nextLed == LED_PIN3) {
86.           currentLed = LED_PIN2;
87.           previusLed = LED_PIN1;
88.           GPIO_ResetBits(F_PORT, previusLed);
89.           GPIO_SetBits(F_PORT,currentLed);
90.           nextLed = LED_PIN4;
91.        } else if (nextLed == LED_PIN4) {
92.           currentLed = LED_PIN3;
93.           previusLed = LED_PIN2;
94.           GPIO_ResetBits(F_PORT, previusLed);
95.           GPIO_SetBits(F_PORT,currentLed);
96.           nextLed = LED_PIN1;
97.     }
98.     while (flag1) {
99.         flag1 = GPIO_ReadInputDataBit(GPIOA, WKUP_BTN);
100.        //Delay(DELAY);
101.    }
102.}
```

```
103.  static void Delay (uint32_t delay){
104.      while (--delay) {
105.        __NOP();
106.      }
107.  }
```