自动控制实践 B

嵌入式系统实验报告

STM32-DMCK

实 验 名 称	： 实验一 综合性实验
姓　　名	： 朱方程
学　　号	： SZ170410221
班　　级	： SZ1703202
撰 写 日 期	： 2020.7.6

实验一 LED 流水灯实验

1. 实验目的

   掌握 STM32CubeMX 使用方法、程序外设的配置，和其他相关的操作。

   在熟练使用 MDK 的过程中、熟悉实验平台的仿真和下载方法。

   掌握 STM32 单片机的 GPIO 口的输出控制。

2. 实验设备

   PC 机一台

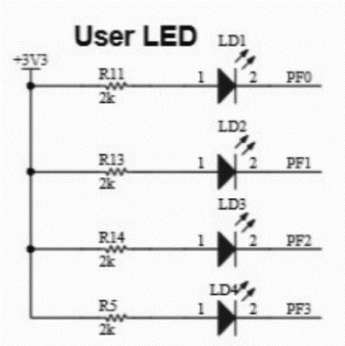
   实验平台一台

3. 实验内容

   实现实验平台底板上的指示灯 LD1-LD4（TFT LCD 正下方）四个指示灯流水点亮的功能。

4. 实验原理

   主控板上配有 4LED 指示灯。 PF0-PF3 为底板上 LD1-LD4 的接口。将 4 个接口配置为推挽输出模式（Out_pp）,对 PF0-PF3 延时配置高或者低，即可实现流水点亮的功能。通过软件延时语句实现最基本的闪烁功能，延时的长短决定闪烁的快慢。



5. 实验步骤

   （1）建立一个新的工程，通过 STM32CubeMX 完成对外设和时钟的配置。

   （2）生成工程文件并打开，在相应的位置添加程序。

   （3）打开实验平台，对工程文件进行仿真下载，查看指示灯的流水效果。

6. 实验代码

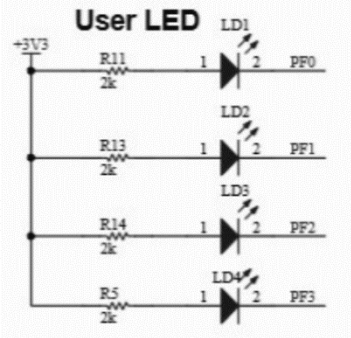
```
HAL_Delay(100);
HAL_GPIO_TogglePin(GPIOF, GPIO_PIN_0);
HAL_Delay(100);
HAL_GPIO_TogglePin(GPIOF, GPIO_PIN_1);
HAL_Delay(100);
HAL_GPIO_TogglePin(GPIOF, GPIO_PIN_2);
HAL_Delay(100);
HAL_GPIO_TogglePin(GPIOF, GPIO_PIN_3);
HAL_Delay(100);
HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_13);
```

```
HAL_GPIO_TogglePin(GPIOF, GPIO_PIN_3);
HAL_Delay(100);
HAL_GPIO_TogglePin(GPIOF, GPIO_PIN_2);
HAL_Delay(100);
HAL_GPIO_TogglePin(GPIOF, GPIO_PIN_1);
HAL_Delay(100);
HAL_GPIO_TogglePin(GPIOF, GPIO_PIN_0);
HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_13);
```

7. 实验感悟和收获：
   通过本次实验，学习了利用 STM32CubeMX 建立工程文件，并利用图形界面配置引脚的方法。对 STM32 常用管脚的功能有了初步认识，对其使用的时钟有了基本的了解。
   学习了 GPIO 电平翻转函数 `HAL_GPIO_TogglePin` 的使用方法，以及延时函数 `HAL_Delay` 单位为 `ms`。

实验二 按键控制 LED 闪烁实验

1. 实验目的
   掌握 STM32CubeMX 使用方法、程序外设的配置和其他相关的操作。
   掌握在 MDK 的使用、程序的添加以及其他相关的操作。
   掌握 STM32 单片机的 GPIO 口的输入输出功能。实验设备
2. 实验设备
   PC 机一台
   实验平台一台
3. 实验内容
   利用实验平台的 LD1-LD4 和 4 和按键（皆在 LCD 屏幕下方），试编程实现 LETF、RIGHT、UP、DOWN、按键分别对应的 LD1、LD2、LD3、LD4 指示灯亮，其他指示灯灭。
4. 实验原理
   实验分析 PF0-PF3 为核心板上 LD1-LD4 的接口。将 4 个接口配置为推挽输出模式（Out_pp）
   LETF、RIGHT、UP、DOWN、按键分别占 PG14、PG13、PG15、PG12 按键为低电平有效（即按键按下时引脚电平状态为低电平）按键电路图和指示灯电路图如下。



5. 实验步骤

（1）建立一个新的工程，通过 STM32CubeMX 完成对外设和时钟的配置。

（2）生成工程文件并打开，在相应的位置添加程序。

（3）打开实验平台，对工程文件进行仿真下载，查看运行效果。

6. 实验代码

```
1.  /* USER CODE BEGIN WHILE */
2.    while (1)
3.    {
4.          if(HAL_GPIO_ReadPin(GPIOG,GPIO_PIN_14)==0)
5.          {
6.          HAL_GPIO_WritePin(GPIOF,GPIO_PIN_0,GPIO_PIN_RESET);
7.          HAL_GPIO_WritePin(GPIOF,GPIO_PIN_1,GPIO_PIN_SET);
8.          HAL_GPIO_WritePin(GPIOF,GPIO_PIN_2,GPIO_PIN_SET);
9.          HAL_GPIO_WritePin(GPIOF,GPIO_PIN_3,GPIO_PIN_SET);
10.         }else if(HAL_GPIO_ReadPin(GPIOG,GPIO_PIN_13)==0)
11.         {
12.         HAL_GPIO_WritePin(GPIOF,GPIO_PIN_0,GPIO_PIN_SET);
13.         HAL_GPIO_WritePin(GPIOF,GPIO_PIN_1,GPIO_PIN_RESET);
14.         HAL_GPIO_WritePin(GPIOF,GPIO_PIN_2,GPIO_PIN_SET);
15.         HAL_GPIO_WritePin(GPIOF,GPIO_PIN_3,GPIO_PIN_SET);
16.         }else if(HAL_GPIO_ReadPin(GPIOG,GPIO_PIN_15)==0)
17.         {
18.         HAL_GPIO_WritePin(GPIOF,GPIO_PIN_0,GPIO_PIN_SET);
19.         HAL_GPIO_WritePin(GPIOF,GPIO_PIN_1,GPIO_PIN_SET);
20.         HAL_GPIO_WritePin(GPIOF,GPIO_PIN_2,GPIO_PIN_RESET);
21.         HAL_GPIO_WritePin(GPIOF,GPIO_PIN_3,GPIO_PIN_SET);
22.         }else if(HAL_GPIO_ReadPin(GPIOG,GPIO_PIN_12)==0)
23.         {
24.         HAL_GPIO_WritePin(GPIOF,GPIO_PIN_0,GPIO_PIN_SET);
25.         HAL_GPIO_WritePin(GPIOF,GPIO_PIN_1,GPIO_PIN_SET);
26.         HAL_GPIO_WritePin(GPIOF,GPIO_PIN_2,GPIO_PIN_SET);
27.         HAL_GPIO_WritePin(GPIOF,GPIO_PIN_3,GPIO_PIN_RESET);
28.         }
29.
30. /* USER CODE END WHILE */
```

7. 实验感悟和收获：

添加用户代码时必须添加在“/* USER CODE BEGIN...*/”至“/* USER CODE END... */” 之间。这样 CUBEMX 工程文件可以重复配置，并生成相关代码，而不影响用户原来写好的代码。

通过本次实验，学习了利用按键控制 LED 灯亮灭的方法，学会了常用的输入输出操作函数 HAL_GPIO_WritePin 和 HAL_GPIO_ReadPin.

实验三 外部中断实验

1. 实验目的

   掌握 STM32CubeMX 使用方法、程序外设的配置，外部中断设置。

   掌握在 MDK 的使用、程序的添加以及其他相关的操作。

   熟悉 STM32 单片机的中断机制。

2. 实验设备

   PC 机一台

   实验平台一台

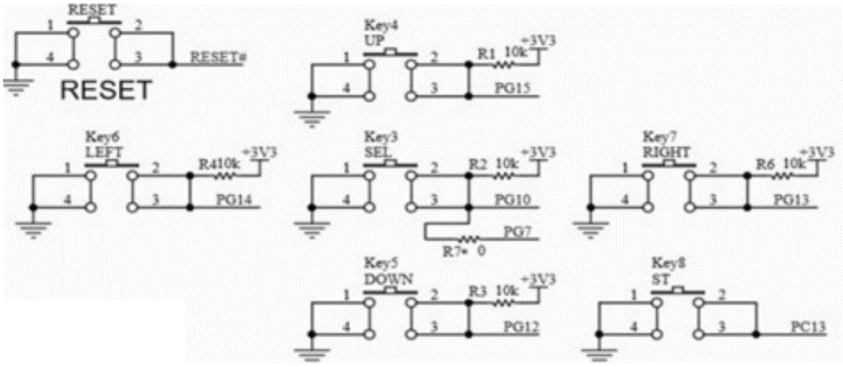
3. 实验内容

   利用实验平台 LD1 和 UP 按键（皆在 LCD 下方在），试用**外部中断功能**编程实现按键 UP 按键按下时 LD1 亮灭状态切换。

4. 实验原理

   LD1 由 PF0 控制输出。在 GPIO 初始化时，需要将这个引脚配置为推挽输出。UP 按键占用的是 PG15 引脚，按键为**低电平有效**（即按键按下时引脚电平为低电平）。

   按键电路图如下图所示。



   外部中断占 EXTI15 口，按键 GPIO 需要设置为内部上拉模式。具体外部中断原理请参考相关教材或手册。

5. 实验步骤

   （1）建立一个新的工程，通过 STM32CubeMX 完成对外设和时钟的配置。

   （2）生成工程文件并打开，在相应的位置添加程序。

   （3）打开实验平台，对工程文件进行仿真下载，查看运行效果。

6. 实验代码

   中断回调函数

```
1.  void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
```

```
{
    if(GPIO_Pin==KEY_UP_Pin) //KEY_UP PRESSED
    {
        if(HAL_GPIO_ReadPin(KEY_UP_GPIO_Port,KEY_UP_Pin)==GPIO_PIN_RESET)
        {
            HAL_GPIO_TogglePin(LED1_GPIO_Port,LED1_Pin);
        }
    }
}
```

7. 实验心得体会

本次实验是利用按键中断实现 LED 灯的亮灭，该中断为下降沿触发，即按下按键的一瞬间触发中断。在 HAL 库中，中断运行结束后不会立刻退出，而是会先进入相对应的中断回调函数，处理该函数中的代码之后，才会退出中断，所以在 HAL 库中我们一般将中断需要处理代码放在中断回调函数中，中断回调函数为：HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)，此函数就是外部中断处理函数。当触发中断后，进行电平翻转的操作，实现用外部中断来控制 LED 亮灭的功能。

实验四 定时器定时应用实验

1. 实验目的

掌握 STM32CubeMX 使用方法、程序外设的配置，定时器的设置。

掌握在 MDK 的使用、程序的添加以及其他相关的操作。

熟悉 STM32 单片机的定时器及其编程。

2. 实验设备

PC 机一台

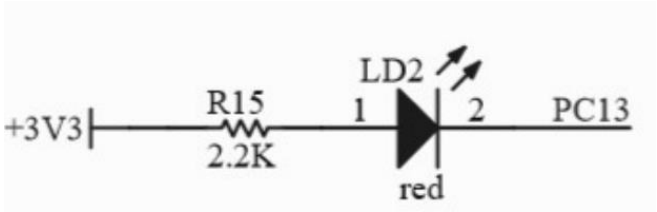
实验平台一台

3. 实验内容

利用实验平台核心板 PC13 指示灯的 1s 闪烁功能

4. 实验原理

PC13 为核心板上 LD2 灯的接口。将 PC13 配置为集电极开路输出模式（out_pp），设置定时器功能，实现指示灯的闪烁。PC13 指示灯电路如下图所示。



STM32F4 芯片具有 14 个定时器，含有 2 个 32 位高级定时器 TIM1 和 TIM8，10 个 16 位通用定时器，2 个基本定时器 TIM6 和 TIM7。

可以采用通过定时器 TIM2 来完成实验的定时功能。通用定时器包含了一个 16 位或 32 位的自动装载计数器，该计数器由可编程预分频驱动。他们可以通过多种用途，包括测量输入信号的脉冲宽度或者生成输出波形。使用定时器预分频和 RCC 时钟控制控制分频器，可将脉冲宽度和波形周期从几微妙调制到几毫秒。

5. 实验步骤

（1）建立一个新的工程，通过 STM32CubeMX 完成对外设和时钟的配置。

（2）生成工程文件并打开，在相应的位置添加程序。

（3）打开实验平台，对工程文件进行仿真下载，查看运行效果。

6. 实验代码

在主程序中添加如下代码，启动定时器中断功能。

```
/* USER CODE BEGIN 2 */
HAL_TIM_Base_Start_IT(&htim2);
/* USER CODE END 2 */
```

创建定时器中断回调函数，在此回调函数中，实现指示灯端口状态的控制功能，如：

```
/* USER CODE BEGIN 4 */
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if(htim->Instance==TIM2)
    {
        HAL_GPIO_TogglePin(LED_GPIO_Port,LED_Pin);
    }
}
/* USER CODE END 4 */
```

下载并执行程序，观察核心板指示灯的变化。

7. 实验心得体会

本次实验是利用定时器实现 LED 闪烁功能，其中关于定时器的配置为：选择内部时钟源：84M，分频值为 8400，定时器周期 5000，得到的更新中断的时间为 500ms。

实验五 DAC 基本实验

1. 实验目的

掌握 STM32CubeMX 使用方法、程序外设的配置，DAC 的工作原理及设置方法。

掌握在 MDK 的使用、程序的添加以及其他相关的操作。

学会使用 STM32 的 DAC 功能。

2. 实验设备

PC 机一台

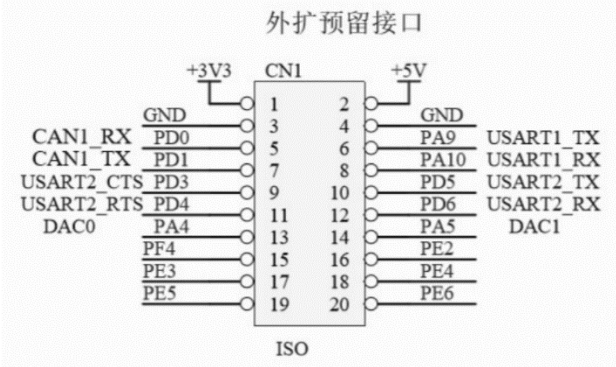
实验平台一台

3. 实验内容

试编程使用 DAC 功能实现输出为三角波。

4. 实验原理

STM32F407 处理的 DA 外设可通过 PA4（DAC0 通道）输出，也可以使用 PA5（DAC1 通道）输出。关于 STM32F407 外设原理，请详见芯片编程手册。平台主控板中，CN1 接口引出可做此实验的 PA4 或 PA5 引脚，如下图所示。本次实验采用 PA5 输出三角波。



5. 实验步骤

（1）建立一个新的工程，通过 STM32CubeMX 完成对外设和时钟的配置。

（2）生成工程文件并打开，在相应的位置添加程序。

（3）打开实验平台，对工程文件进行仿真下载，查看运行效果。

6. 实验代码

```
1.  /* USER CODE BEGIN 2 */
2.   HAL_TIM_Base_Start(&htim2);
3.   HAL_DAC_Start(&hdac, DAC_CHANNEL_2);
4.  /* USER CODE END 2 */
```

7. 实验心得体会

在本次综合性实验中，我学习了用 STM32CubeMX 来配置管脚、生成工程文件，配合 MDK-ARM 来编写代码的流程，同时深切体会到 HAL 库编程的优越性，对嵌入式开发的流程更加的熟悉。