**哈尔滨工程大学信通学院**

**实习手册**

**学生姓名 付丁一**

**学 号 2018080405**

**实习单位 中国高科**

**实习班级**

**指导老师 李北明 (校内)**

**王亚涛 (校外)**

**二○二一年七月**

|  |  |  |  |
| --- | --- | --- | --- |
| **时间** | **7.12** | **实习内容** | **STM32的开发（1）** |
| **day6**  **今日收获**   1. clean.bat 解放双手 2. 复习寄存器操作 3. HAL库的使用    1. 配合CubeMX    2. 简单的GPIO配置    3. 更加简单的中断 4. 代码的移植    1. 使用HAL库，快速移植代码   **STM32的开发**   1. 寄存器 2. std库 3. HAL库   *今天主要任务*：用不同的方式点亮LED  注：  1. 因为STM32G0引脚比较少  2. STM32G0没有预留JTAG的口，如需要使用JTAG需要手动连杜邦线  3. 如果使用串口下载方式，个人认为比较繁琐  - 因为有STM32F1开发板，所以将单片机型号改为STM32F103RCT6.大部分代码相同，逻辑上全部类似；可以使用JTAG，方便程序调试。  **STM32寄存器开发方法**  *认识寄存器*  什么是寄存器？  在数字电子，尤其是计算中，硬件寄存器是通常由触发器组成的电路，通常具有许多类似于存储器的特性，例如：   * 一次读取或写入多个位的能力 * 使用地址以类似于内存地址的方式选择特定寄存器   然而，它们的显着特点是它们还具有超出普通内存的特殊硬件相关功能。所以，从不同的角度来看，硬件寄存器就像内存，具有额外的硬件相关功能；或者，内存电路就像只存储数据的硬件寄存器。  硬件寄存器用于软件和外设之间的接口。软件写入它们以向设备发送信息，并读取它们以从设备获取信息。一些硬件设备还包括软件不可见的寄存器，供其内部使用。  详见《微机原理与接口技术》  如何寻找 STM32F103RCT6 的寄存器              （P194）    并不觉得像STM32这类功能强大的单片机应当用寄存器的方式来开发。  以上课时用STM32G0编写的一段代码来解释。  C  #include <stm32g0xx.h> //这里存入了寄存器的定义表  /\*USER Define-----------------------------\*/  #define MY\_MODE\_OUTPUT\_PP 0  #define MY\_MODE\_OUTPUT\_OD 1  #define MY\_SPEED\_LOW 0  #define MY\_SLEED\_MID 1  #define MY\_SPEED\_HIGH 2  #define MY\_SPEED\_VERY\_HIGH 3  /\*Define End------------------------------\*/  //用户自己定义结构体，方便后续输入  typedef struct {  uint16\_t pin;  uint8\_t DirOrSpeed;//输入还是输出，速度  uint8\_t PP\_OD\_Pull;//输出模式  } myGPIO\_ST;  //延时函数  void delay(unsigned int a) {  while(a--) {  ;  }  }  void IO\_Init(GPIO\_TypeDef \*GPIOx,myGPIO\_ST\* st) {  uint8\_t temp;  if (st->DirOrSpeed > 0) { //不是输入模式。暂时只处理输出  GPIOx->ODR &= ~(0xf << st->pin\*4);  temp =(st->PP\_OD\_Pull<<2)|(st->DirOrSpeed);  GPIOx ->ODR |= temp << (st->pin\*4);  }  }  int main() {  RCC->IOPENR |= RCC\_IOPENR\_GPIOBEN;  myGPIO\_ST myGPIO;  myGPIO.pin = 0;  myGPIO.DirOrSpeed = MY\_SPEED\_LOW;  myGPIO.PP\_OD\_Pull = MY\_MODE\_OUTPUT\_PP;  IO\_Init(GPIOB,&myGPIO);  while(1) {  GPIOB->ODR &= ~(0x1<<(0\*1) | (1\*1));  delay(0xfffff);  GPIOB->ODR |= (0x1<<(0\*1) | (1\*1));  delay(0xfffff);  }  }  与之前接触过的std库函数相比，无论从可读性还是开发的速度和对开发者的要求上，库函数开发都略胜寄存器开发一筹。  **HAL库**    STM32Cube硬件抽象层（HAL），STM32抽象层嵌入式软件，可确保STM32微控制器上最大的可移植性。 HAL适用于所有硬件外围设备。他们隐藏MCU和外设复杂度到最终用户。  *从寄存器到HAL库*  寄存器操作让我想起刚接触单片机时，我开始使用MSP430的寄存器开发的方式，代码可读性比较差（即便当时写了简单的注释，现在读起来也比较吃力）；唯一的好处是让我对指针有了进一步的了解。   * *其实《微机原理》解释得还蛮清楚的*     *使用HAL函数点亮LED*  1. 使用CubeMX配置工程  详见day5日报   * 如何找到板卡上的LED2连接的引脚？   因为我使用的是正点原子F103RCT6的板卡，所以需要到厂家提供的开发板原理图中找。      2. 打开Keil进行main.c的编写  在main文件之中，只需要编写while(1)中的具体语句即可。  C  /\* Infinite loop \*/  /\* USER CODE BEGIN WHILE \*/  while (1)  {  HAL\_GPIO\_TogglePin(LED2\_GPIO\_Port,LED2\_Pin);  HAL\_Delay(1000);  /\* USER CODE END WHILE \*/      3. 下载与调试        配置好JTAG就可以烧录板卡了。      点击settings。    **注意**：   1. 在CubeMX生成的工程文件中，如需用户自己书写代码，必须要按照CubeMX分配好的空间来写。否则在下一次使用CubeMX进行新的配置时，用户的代码会丢失。 2. 翻转LED的电平后，记得延时。否则闪烁过快，看不到明显的现象。   *使用HAL函数利用按键控制LED电平翻转*  1. 使用CubeMX配置工程  首先，需要在原理图上找到KEY0对应的GPIO。    当然还有小灯。（同上个）    接下来就在CubeMX中配置这两个GPIO。其中，PC5为Input，PD2为Output。具体参数见图。        最后点击Generate即可。  2. 编写main.c  同样只需要编写while(1)里的代码即可。  C  /\* Infinite loop \*/  /\* USER CODE BEGIN WHILE \*/  while (1) {  if(KEY0 == 0){  HAL\_Delay(10);//软件消抖  while(!KEY0)//等待KEY无效  ;  HAL\_Delay(10);//软件消抖  HAL\_GPIO\_TogglePin(LED\_GREEN\_GPIO\_Port,LED\_GREEN\_Pin);//翻转电平  }  /\* USER CODE END WHILE \*/  3. 烧录程序  同样，连接好JTAG和板卡后，点击Translate-Bulid-Download即可成功烧录。 | | | |
| **指导教师** |  | **分数** |  |
| **实习评语:** | | | |