

通用 IO 口 (GPIO) 例程实验

1.GPIO功能描述

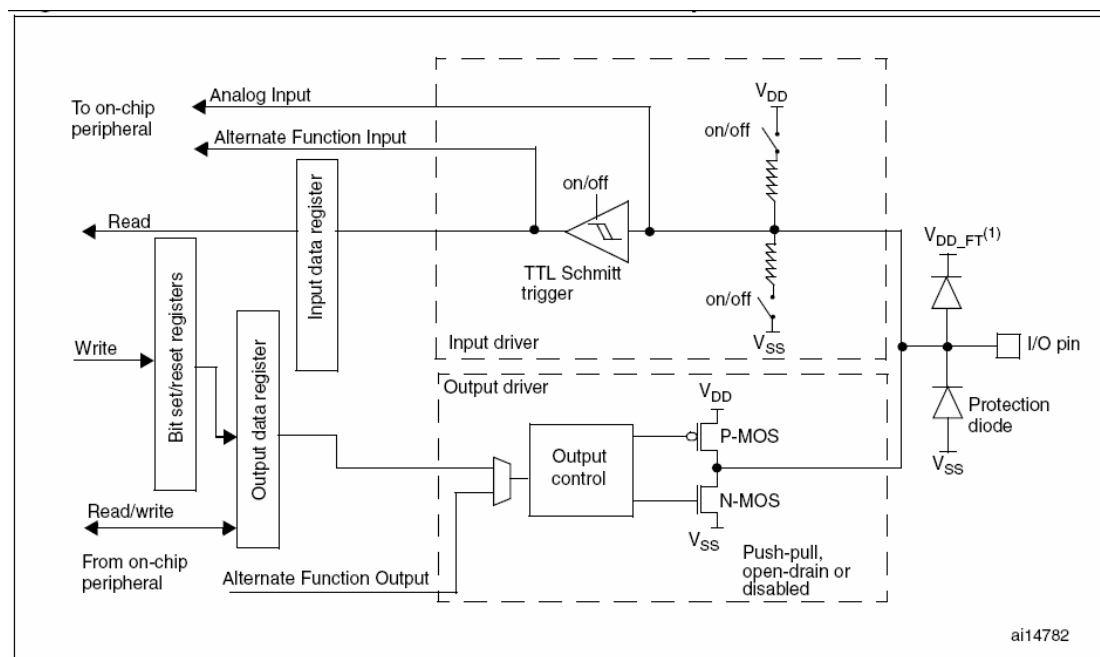
每个GPIO 端口有两个32 位配置寄存器(GPIOx_CRL, GPIOx_CRH), 两个32 位数据寄存器(GPIOx_IDR, GPIOx_ODR), 一个32 位置位/复位寄存器(GPIOx_BSRR), 一个16 位复位寄存器(GPIOx_BRR)和一个32 位锁定寄存器(GPIOx_LCKR)。

根据数据手册中列出的每个I/O 端口的特定硬件特征, GPIO 端口的每个位可以由软件分别配置成多种模式。

- 输入浮空
- 输入上拉
- 输入下拉
- 模拟输入
- 开漏输出
- 推挽式输出
- 推挽式复用功能
- 开漏复用功能

每个I/O 端口位可以自由编程, 然而I/O 端口寄存器必须按32 位字被访问(不允许半字或字节访问)。GPIOx_BSRR 和GPIOx_BRR 寄存器允许对任何GPIO 寄存器的读/更改的独立访问; 这样, 在读和更改访问之间产生IRQ 时不会发生危险。

下图给出了一个5V兼容I/O端口位的基本结构。



(1) V_{DD_FT} 对5 伏兼容I/O 脚是特殊的, 它与V_{DD} 不同

端口位配置表

| 配置模式 | | CNF1 | CNF0 | MODE1 | MODE0 | PxODR寄存器 | |
|--------|----------------|------|------|------------------------|-------|----------|-----|
| 通用输出 | 推挽式(Push-Pull) | 0 | 0 | 01 10 11 见表12 | | 0 或 1 | |
| | 开漏(Open-Drain) | | 1 | | | 0 或 1 | |
| 复用功能输出 | 推挽式(Push-Pull) | 1 | 0 | | | | 不使用 |
| | 开漏(Open-Drain) | | 1 | | | 不使用 | |
| 输入 | 模拟输入 | 0 | 0 | 00 | | 不使用 | |
| | 浮空输入 | | 1 | | | 不使用 | |
| | 下拉输入 | 1 | 0 | | | 0 | |
| | 上拉输入 | | | | | 1 | |

输出模式位

| MODE[1:0] | 意义 |
|-----------|--------------|
| 00 | 保留 |
| 01 | 最大输出速度为10MHz |
| 10 | 最大输出速度为2MHz |
| 11 | 最大输出速度为50MHz |

2 通用I/O(GPIO)

复位期间和刚复位后，复用功能未开启，I/O 端口被配置成浮空输入模式(CNFX[1:0]=01b，MODEx[1:0]=00b)。复位后，JTAG 引脚被置于输入上拉或下拉模式：

- PA15：JTDI 置于上拉模式
- PA14：JTCK 置于下拉模式
- PA13：JTMS 置于上拉模式
- PB4：JNTRST 置于上拉模式

当作为输出配置时，写到输出数据寄存器上的值(GPIOx_ODR)输出到相应的I/O引脚。可以以推挽模式或开漏模式(当输出0时，只有N-MOS 被打开)使用输出驱动器。输入数据寄存器(GPIOx_IDR)在每个APB2 时钟周期捕捉I/O 引脚上的数据。所有GPIO 引脚有一个内部弱上拉和弱下拉，当配置为输入时，它们可以被激活也可以不被激活。

2.1 输入配置

当I/O 端口配置为输入时：

输出缓冲器被禁止

施密特触发输入被激活

根据输入配置(上拉，下拉或浮动)的不同，弱上拉和下拉电阻被连接

出现在I/O 脚上的数据在每个APB2 时钟被采样到输入数据寄存器

对输入数据寄存器的读访问可得到I/O 状态

2.2 输出配置

当I/O 端口被配置为输出时：

输出缓冲器被激活

- 开漏模式：输出寄存器上的0 激活N-MOS，而输出寄存器上的1 将端口置于高阻状态 (P-MOS 从不被激活)。
- 推挽模式：输出寄存器上的0 激活N-MOS，而输出寄存器上的1 将激活P-MOS。

施密特触发输入被激活

弱上拉和下拉电阻被禁止

出现在I/O 脚上的数据在每个APB2 时钟被采样到输入数据寄存器

在开漏模式时，对输入数据寄存器的读访问可得到I/O 状态

在推挽模式模式时，对输出数据寄存器的读访问得到最后一次写的值。

2.3 GPIO寄存器描述

2.3.1 端口配置低寄存器(GPIOx_CRL) (x=A..E)

偏移地址：0x00

复位值：0x4444 4444

| | | | | | | | | | | | | | | | |
|-----------|------------|-----------|------------|-----------|------------|-----------|------------|----|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CNF7[1:0] | MODE7[1:0] | CNF6[1:0] | MODE6[1:0] | CNF5[1:0] | MODE5[1:0] | CNF4[1:0] | MODE4[1:0] | | | | | | | | |
| rW | rW | rW | rW | rW | rW | rW | rW | rW | rW | rW | rW | rW | rW | rW | rW |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CNF3[1:0] | MODE3[1:0] | CNF2[1:0] | MODE2[1:0] | CNF1[1:0] | MODE1[1:0] | CNF0[1:0] | MODE0[1:0] | | | | | | | | |
| rW | rW | rW | rW | rW | rW | rW | rW | rW | rW | rW | rW | rW | rW | rW | rW |

| | |
|---------|---------------------------------|
| 位 | CNFy[1:0]: 端口x配置位(y = 0...7) |
| 31 : 30 | 软件通过这些位配置相应的I/O端口，请参考表11端口位配置表。 |
| 27 : 26 | 在输入模式(MODE[1:0]=00): |
| 23 : 22 | 00: 模拟输入模式 |
| 19 : 18 | 01: 浮空输入模式(复位后的状态) |
| 15 : 14 | 10: 上拉/下拉输入模式 |
| 11 : 10 | 11: 保留 |
| 7 : 6 | 在输出模式(MODE[1:0]>00): |
| 3 : 2 | 00: 通用推挽输出模式 |
| | 01: 通用开漏输出模式 |
| | 10: 复用功能推挽输出模式 |
| | 11: 复用功能开漏输出模式 |

| | |
|---------|---------------------------------|
| 位 | MODEy[1:0]: 端口x的模式位(y = 0...7) |
| 9 : 28 | 软件通过这些位配置相应的I/O端口，请参考表11端口位配置表。 |
| 25 : 24 | 00: 输入模式(复位后的状态) |
| 21 : 20 | 01: 输出模式，最大速度10MHz |
| 17 : 16 | 10: 输出模式，最大速度2MHz |
| 13 : 12 | 11: 输出模式，最大速度50MHz |
| 9 : 8 | |
| 5 : 4 | |
| 1: 0 | |

2.3.2 端口配置高寄存器(GPIOx_CRH) (x=A..E)

偏移地址：0x04

复位值：0x4444 4444

| | | | | | | | | | | | | | | | |
|------------|-------------|------------|-------------|------------|-------------|------------|-------------|----|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CNF15[1:0] | MODE15[1:0] | CNF14[1:0] | MODE14[1:0] | CNF13[1:0] | MODE13[1:0] | CNF12[1:0] | MODE12[1:0] | | | | | | | | |
| rW | rW | rW | rW | rW | rW | rW | rW | rW | rW | rW | rW | rW | rW | rW | rW |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CNF11[1:0] | MODE11[1:0] | CNF10[1:0] | MODE10[1:0] | CNF9[1:0] | MODE9[1:0] | CNF8[1:0] | MODE8[1:0] | | | | | | | | |
| rW | rW | rW | rW | rW | rW | rW | rW | rW | rW | rW | rW | rW | rW | rW | rW |

| | |
|-------|---------------------------------|
| 位 | CNFy[1:0]：端口x配置位(y = 8...15) |
| 31:30 | 软件通过这些位配置相应的I/O端口，请参考表11端口位配置表。 |
| 27:26 | 在输入模式(MODE[1:0]=00)： |
| 23:22 | 00：模拟输入模式 |
| 19:18 | 01：浮空输入模式(复位后的状态) |
| 15:14 | 10：上拉/下拉输入模式 |
| 11:10 | 11：保留 |
| 7:6 | 在输出模式(MODE[1:0]>00)： |
| 3:2 | 00：通用推挽输出模式 |
| | 01：通用开漏输出模式 |
| | 10：复用功能推挽输出模式 |
| | 11：复用功能开漏输出模式 |
| 位 | MODEy[1:0]：端口x的模式位(y = 8...15) |
| 9:28 | 软件通过这些位配置相应的I/O端口，请参考表11端口位配置表。 |
| 25:24 | 00：输入模式(复位后的状态) |
| 21:20 | 01：输出模式，最大速度10MHz |
| 17:16 | 10：输出模式，最大速度2MHz |
| 13:12 | 11：输出模式，最大速度50MHz |
| 9:8 | |
| 5:4 | |
| 1:0 | |

2.3.3 端口输入数据寄存器(GPIOx_IDR) (x=A..E)

地址偏移：0x08

复位值：0x0000 0000

| | | | | | | | | | | | | | | | |
|-------|-------|-------|-------|-------|-------|------|------|------|------|------|------|------|------|------|------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| 保留 | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IDR15 | IDR14 | IDR13 | IDR12 | IDR11 | IDR10 | IDR9 | IDR8 | IDR7 | IDR6 | IDR5 | IDR4 | IDR3 | IDR2 | IDR1 | IDR0 |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

| | |
|--------|--|
| 位31:16 | 保留，始终读为0。 |
| 位15:0 | IDRy[15:0]：端口输入数据(y = 0...15) 这些位为只读并只能以字(16位)的形式读出。读出的值为对应I/O口的状态。 |

2.3.4 端口输出数据寄存器(GPIOx_ODR) (x=A..E)

地址偏移：0Ch

复位值：00000000h

| | | | | | | | | | | | | | | | |
|-------|-------|-------|-------|-------|-------|------|------|------|------|------|------|------|------|------|------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| 保留 | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ODR15 | ODR14 | ODR13 | ODR12 | ODR11 | ODR10 | ODR9 | ODR8 | ODR7 | ODR6 | ODR5 | ODR4 | ODR3 | ODR2 | ODR1 | ODR0 |
| rW | rW | rW | rW | rW | rW | rW | rW | rW | rW | rW | rW | rW | rW | rW | rW |

| | |
|--------|--|
| 位31:16 | 保留，始终读为0。 |
| 位15:0 | ODRy[15:0]：端口输出数据(y = 0...15) 这些位可读可写并只能以字(16位)的形式操作。 注：对GPIOx_BSRR(x = A...E)，可以分别地对各个ODR位进行独立的设置/清除。 |

2.3.5 端口位设置/复位寄存器(GPIOx_BSRR) (x=A..E)

地址偏移：0x10

复位值：0x0000 0000

| | | | | | | | | | | | | | | | |
|------|------|------|------|------|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| BR15 | BR14 | BR13 | BR12 | BR11 | BR10 | BR9 | BR8 | BR7 | BR6 | BR5 | BR4 | BR3 | BR2 | BR1 | BR0 |
| W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BS15 | BS14 | BS13 | BS12 | BS11 | BS10 | BS9 | BS8 | BS7 | BS6 | BS5 | BS4 | BS3 | BS2 | BS1 | BS0 |
| W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| | |
|--------|---|
| 位31:16 | BRy: 清除端口x的位y (y = 0...15) 这些位只能写入并只能以字(16位)的形式操作。 0: 对对应的ODRy位不产生影响 1: 清除对应的ODRy位为0 注: 如果同时设置了BSy和BRy的对应位, BSy位起作用。 |
| 位15:0 | BSy: 设置端口x的位y (y = 0...15) 这些位只能写入并只能以字(16位)的形式操作。 0: 对对应的ODRy位不产生影响 1: 设置对应的ODRy位为1 |

2.3.6 端口位复位寄存器(GPIOx_BRR) (x=A..E)

地址偏移: 0x14

复位值: 0x0000 0000

| | | | | | | | | | | | | | | | |
|------|------|------|------|------|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| 保留 | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BR15 | BR14 | BR13 | BR12 | BR11 | BR10 | BR9 | BR8 | BR7 | BR6 | BR5 | BR4 | BR3 | BR2 | BR1 | BR0 |
| W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| | |
|--------|--|
| 位31:16 | 保留。 |
| 位15:0 | BRy: 清除端口x的位y (y = 0...15) 这些位只能写入并只能以字(16位)的形式操作。 0: 对对应的ODRy位不产生影响 1: 清除对应的ODRy位为0 |

2.3.7 端口配置锁定寄存器(GPIOx_LCKR) (x=A..E)

当执行正确的写序列设置了位16(LCKK)时,该寄存器用来锁定端口位的配置。位[15:0]用于锁定GPIO 端口的配置。在规定的写入操作期间,不能改变LCKP[15:0]。当对相应的端口位执行了LOCK 序列后,在下次系统复位之前将不能再更改端口位的配置。每个锁定位锁定控制寄存器(CRL, CRH)中相应的4 个位。

地址偏移: 0x18

复位值: 0x0000 0000



| | |
|--------|---|
| 位31:17 | 保留。 |
| 位16 | <p>LCKK：锁键</p> <p>该位可随时读出，它只可通过锁键写入序列修改。</p> <p>0：端口配置锁键位激活</p> <p>1：端口配置锁键位被激活，下次系统复位前GPIOx_LCKR寄存器被锁住。</p> <p>锁键的写入序列：</p> <p>写1 -> 写0 -> 写1 -> 读0 -> 读1</p> <p>最后一个读可省略，但可以用来确认锁键已被激活。</p> <p>注：在操作锁键的写入序列时，不能改变LCK[15:0]的值。</p> <p>操作锁键写入序列中的任何错误将不能激活锁键。</p> |
| 位15:0 | <p>LCKy：端口x的锁位y (y = 0...15)</p> <p>这些位可读可写但只能在LCKK位为0时写入。</p> <p>0：不锁定端口的配置</p> <p>1：锁定端口的配置</p> |

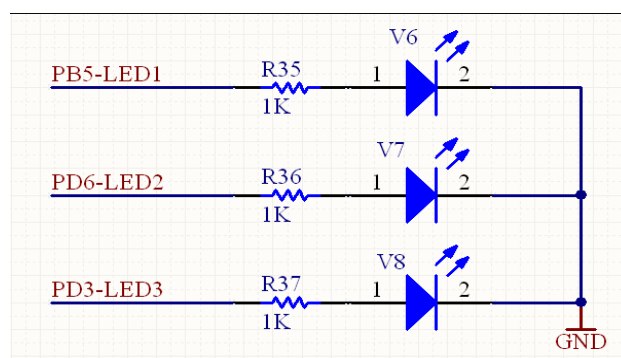
3. 应用实例

3.1. 设计要求

开发板上有3个蓝色状态指示灯V6,V7,V8 ,使它们有规律的点亮 ,具体顺序如下 :V6亮->V7亮->V8亮，如此反复。

3.2 硬件电路设计

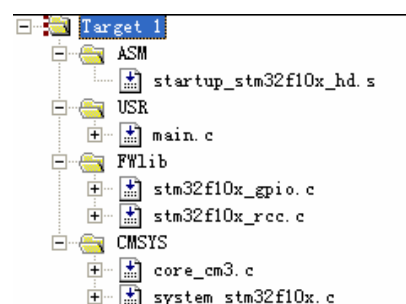
在开发板上V6、V7、V8分别与MCU的PB5、PD6、PD3相连，如下图所示



3.3 软件程序设计

根据任务要求，程序内容主要包括：
初始化后，LED依次点亮。

整个工程包含4类源文件：



ASM--startup_stm32f10x_hd.s 由于奋斗板采用的是STM32F103大存储器芯片，因此采用STM32标准库自带的大存储器芯片启动代码，这个文件已经配置好了初始状态，以及中断向量表。可以直接在工程里使用，如果你以后的应用中采用了中存储器或者小存储器STM32芯片，可以将启动代码换为startup_stm32f10x_md.s 或者 startup_stm32f10x_ld.s。

FWLIB--stm32f10x_gpio.c ST公司的标准库，包含了关于对通用IO口设置的函数。

stm32f10x_rcc.c ST公司的标准库，包含了关于对系统时钟设置的函数。

CMSYS—是关于CORETEX-M3平台的系统函数及定义

USER—main.c 例程的主函数。

RCC_Configuration() 完成对系统时钟的设置，例程中通过系统时钟设置函数，外接晶振采用8Mhz，经过片内频率合成，9倍频，设置为72MHz的时钟。

LED_Config() 对控制3个LED指示灯的IO口进行了初始化，将3个端口配置为推挽上拉输出，口线速度为50Mhz。

在配置某个口线时，首先应对它所在的端口的时钟进行使能。否则无法配置成功，由于3个控制口用到了端口B和端口D，因此要对这两个端口的时钟进行使能，
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB | RCC_APB2Periph_GPIOD ,
ENABLE);

程序中对各状态LED进行了预定义：

```
#define LED1_ON GPIO_SetBits(GPIOB, GPIO_Pin_5);
#define LED1_OFF GPIO_ResetBits(GPIOB, GPIO_Pin_5);

#define LED2_ON GPIO_SetBits(GPIOD, GPIO_Pin_6);
#define LED2_OFF GPIO_ResetBits(GPIOD, GPIO_Pin_6);
```

```
define LED3_ON GPIO_SetBits(GPIOD, GPIO_Pin_3);
#define LED3_OFF GPIO_ResetBits(GPIOD, GPIO_Pin_3);
```

GPIO_SetBits(x,x); GPIO_ResetBits(x,x); 是标准库stm32f10x_gpio.c中的函数，用于对某口线置位或复位。

初始化完成后，进入大循环，执行功能。

```
while (1){
    LED1_ON; LED2_OFF; LED3_OFF;          //V6 亮 V7,V8灭
    Delay(0xAFFFF);
    LED1_OFF; LED2_ON; LED3_OFF;          //V7 亮 V6,V8灭
    Delay(0xAFFFF);
    LED1_OFF; LED2_OFF; LED3_ON;          //V8 亮 V6,V7灭
```




```
    Delay(0xAFFFF);
}
```

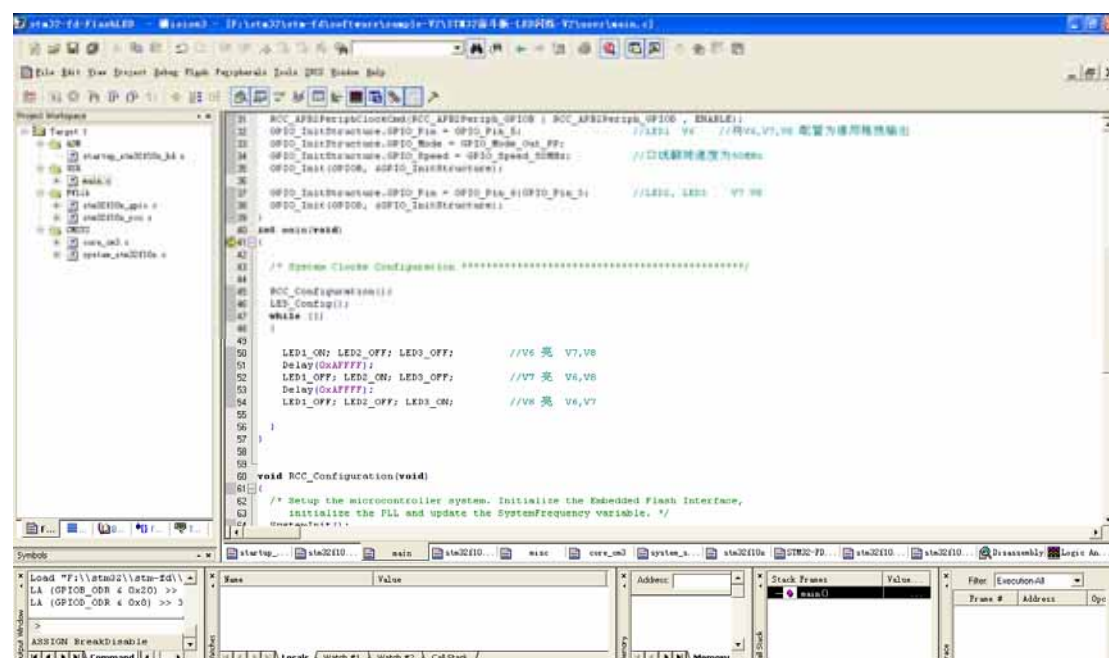
按  编译工程，完成后会提示如下。


```
Build target 'Target 1'
linking...
Program Size: Code=2308 RO-data=336 RW-data=24 ZI-data=512
FromELF: creating hex file...
".\Obj\STM32-FD-FLASHLED.axf" - 0 Error(s), 0 Warning(s).
```

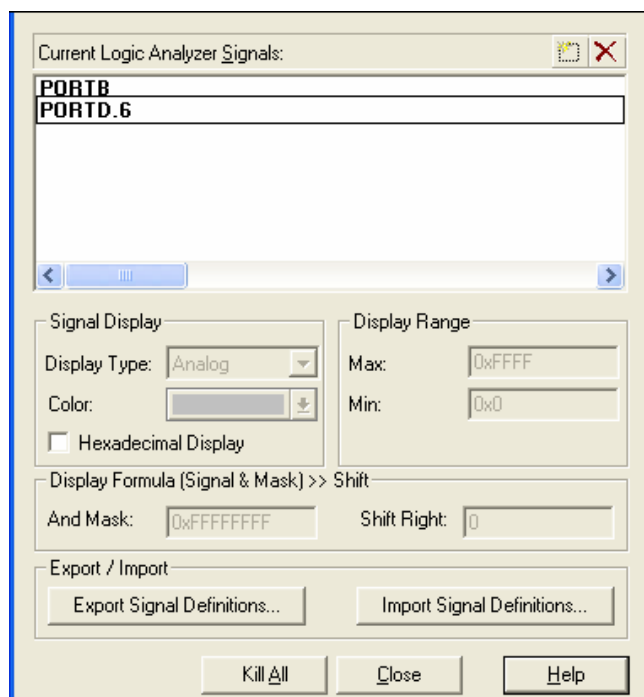
接下来可以下载或者仿真。

3.4 仿真与下载

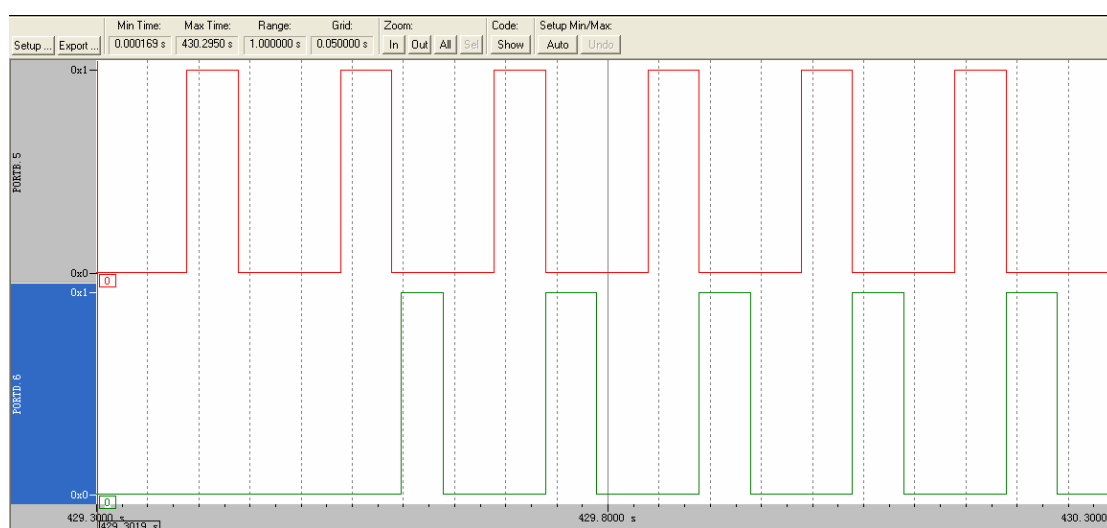
软件仿真：点  进入软件仿真界面。界面如下：



点  进入仿真逻辑逻辑分析仪，点SETUP，加入如下端口。



设置完成后，按  执行程序。如下显示



此图模拟了LED1，LED2的变化情况。

调试例程无问题后，可以通过JLINK V8或者串口将代码写入板子，具体的烧写步骤，参考奋斗板文档目录下的《奋斗版STM32开发板JTAG下载步骤》或者《奋斗版STM32开发板串口下载步骤》

至此，GPIO例程讲解完毕