零死角玩转STM32—M3系列

什么是寄存器

淘宝：firestm32.taobao.com

论坛： www.firebbs.cn

扫描进入淘宝店铺

主讲内容

01 STM32长啥样

02 芯片里面有什么

03 存储器映射

04 寄存器映射

参考资料:《零死角玩转STM32》“什么是寄存器”章节

什么是寄存器

1、什么是储存器映射？

2、什么是寄存器映射？

STM32长啥样

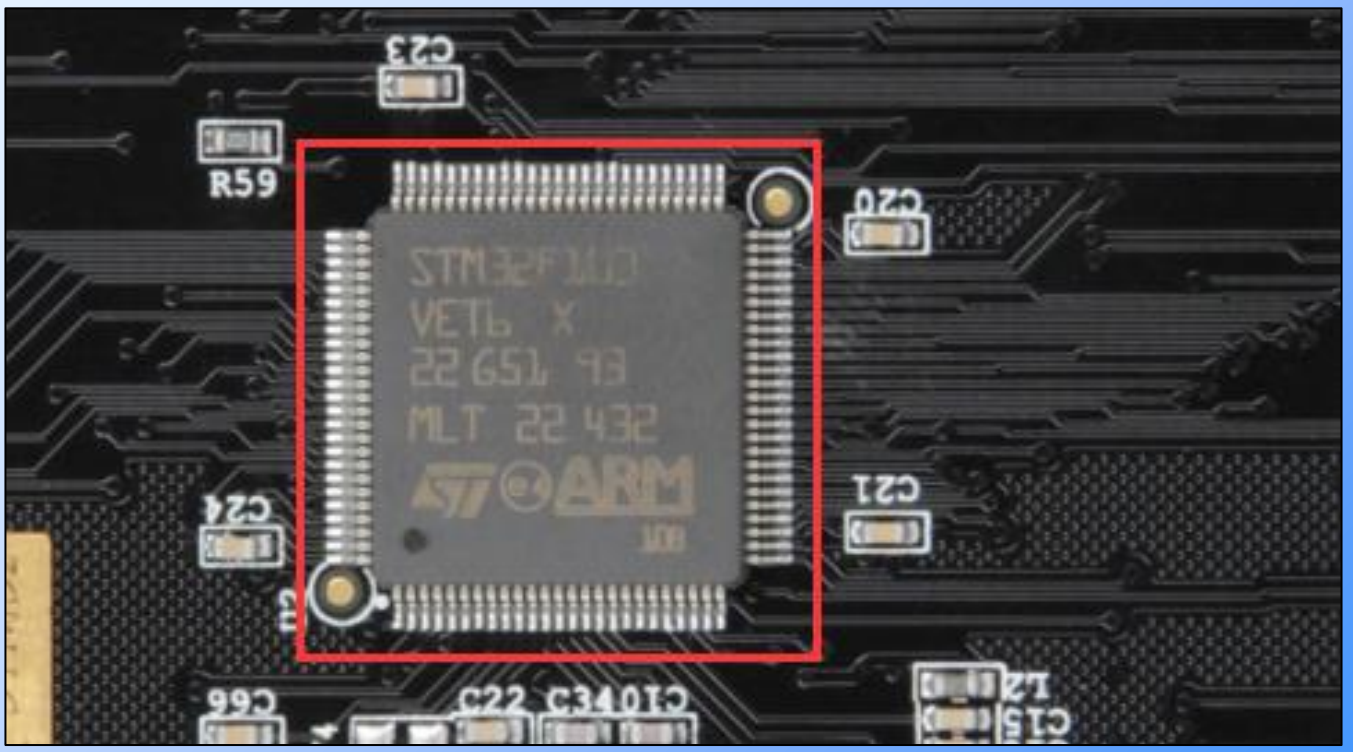
STM32F103系列芯片实物图

1、学会看丝印

2、懂得如何辨别正方向

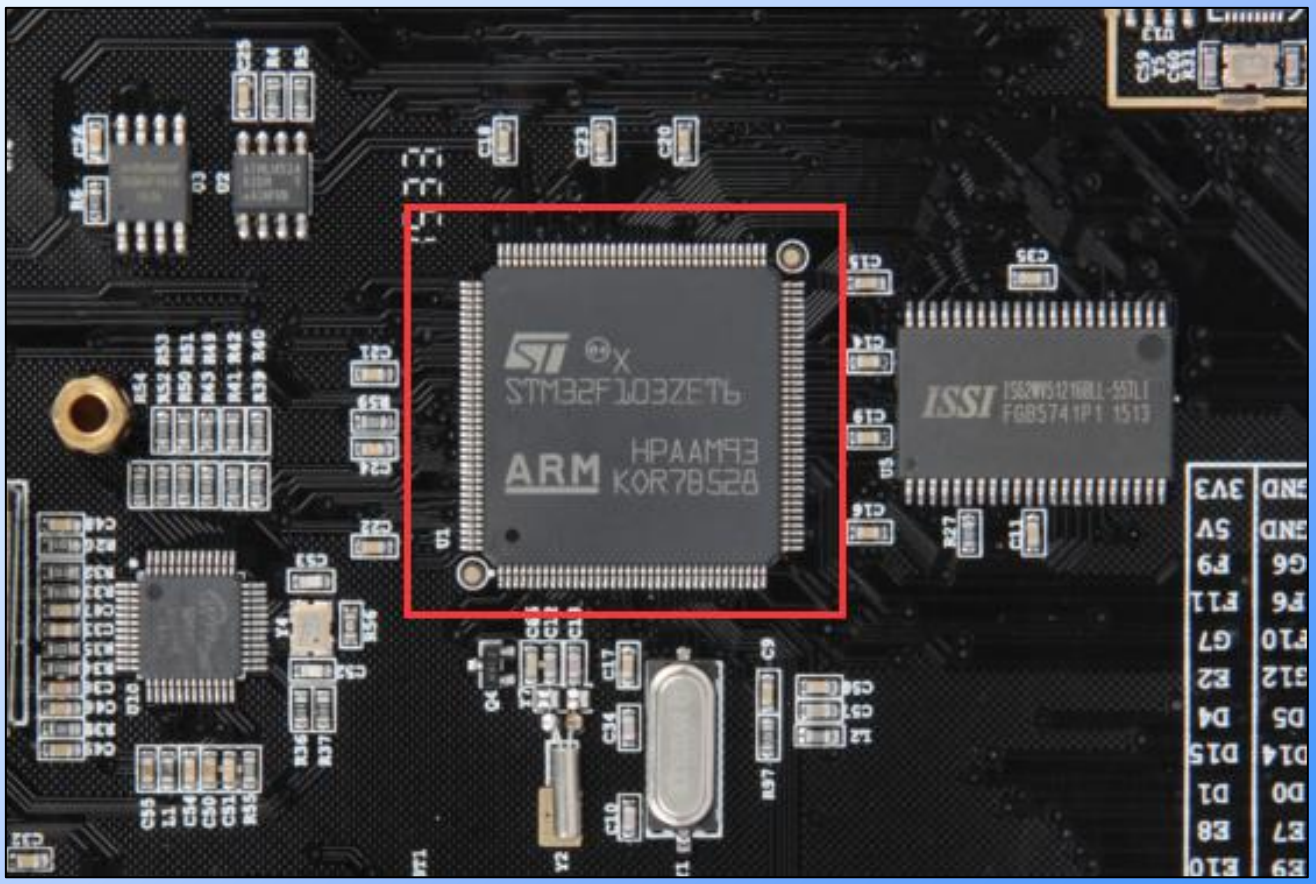
STM32长啥样

指南者—STM32F103VET6



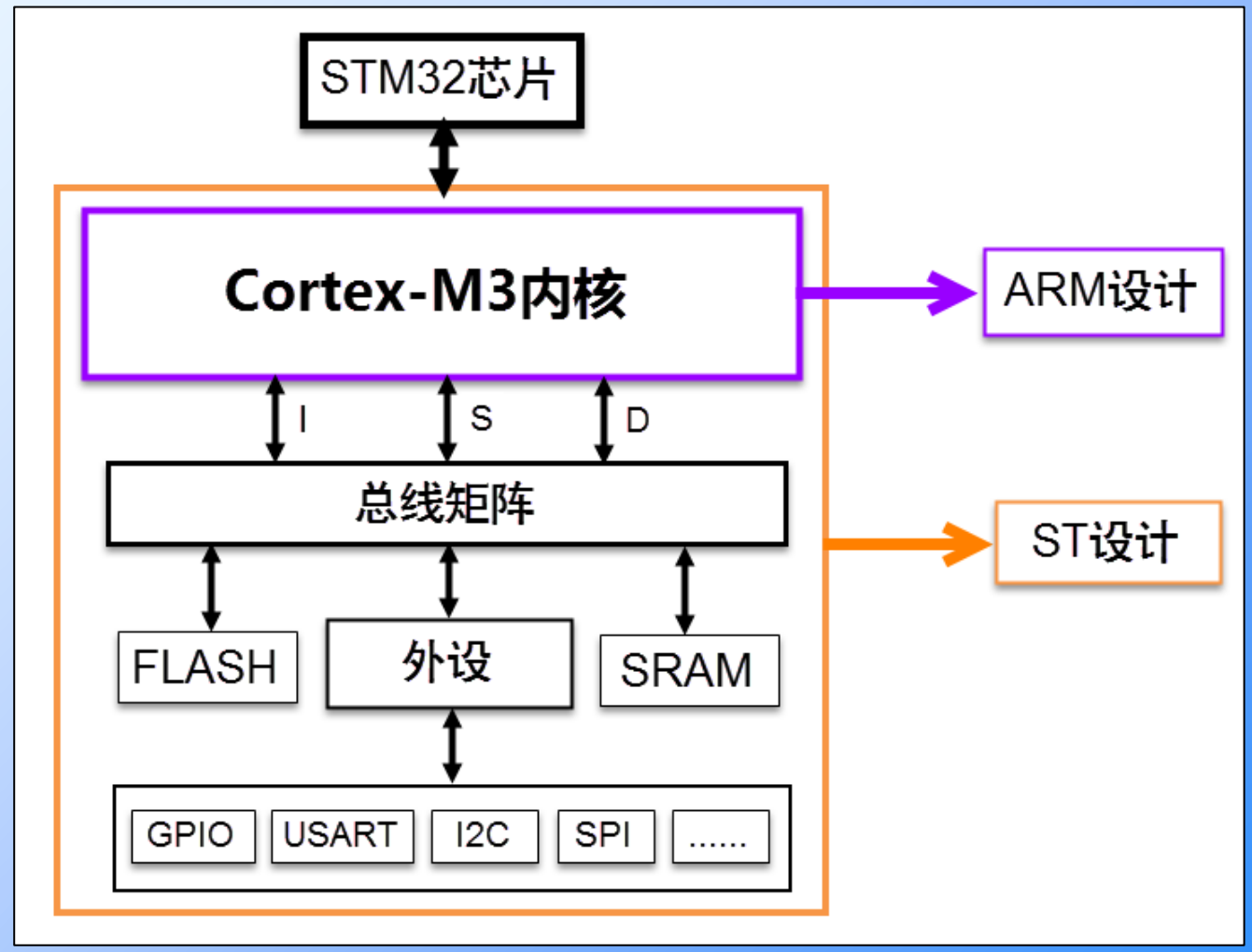
STM32长啥样

霸道—STM32F103ZET6



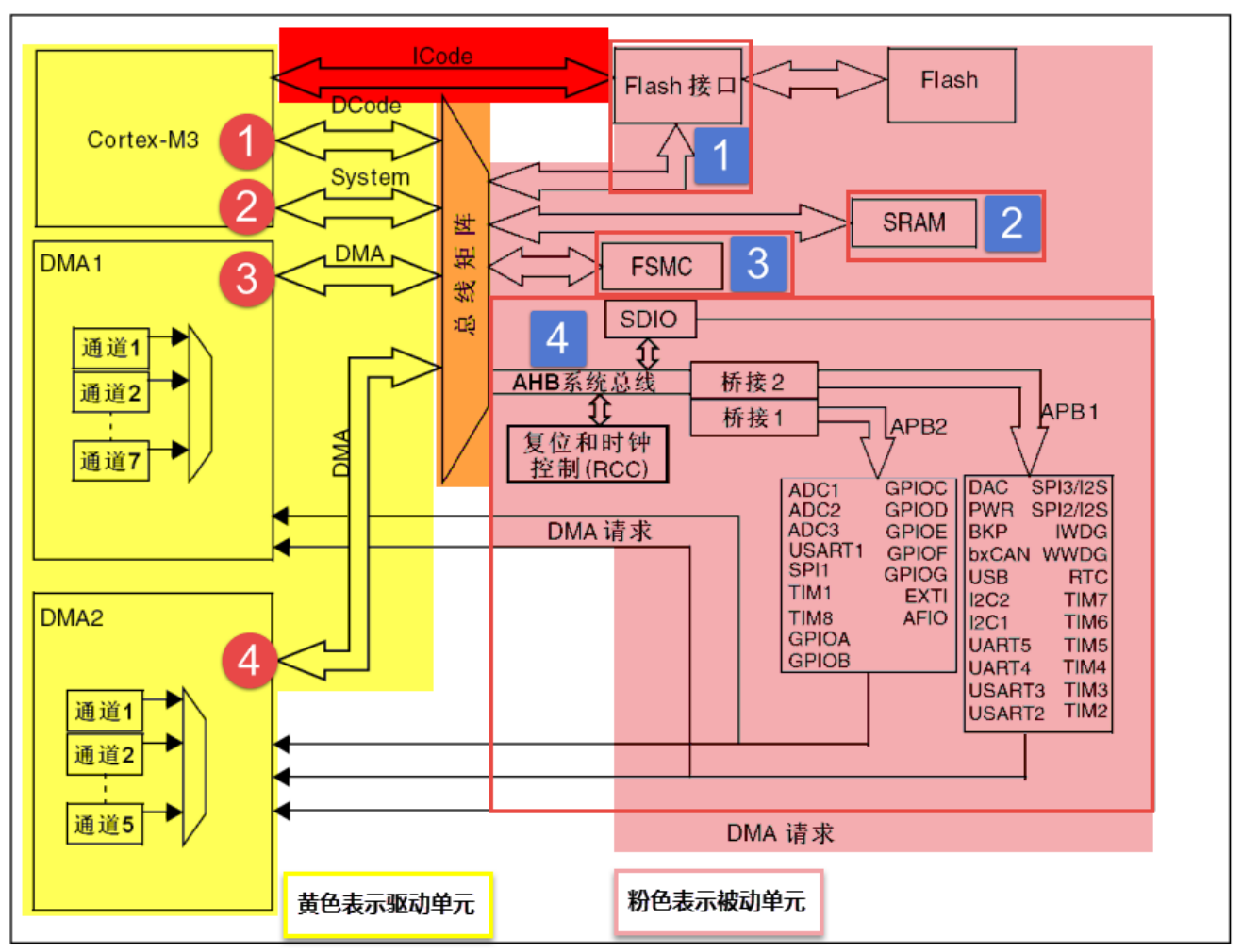
STM32内部有什么

STM32芯片架构简图



STM32内部有什么

STM32F10xx系统框图



存储器映射

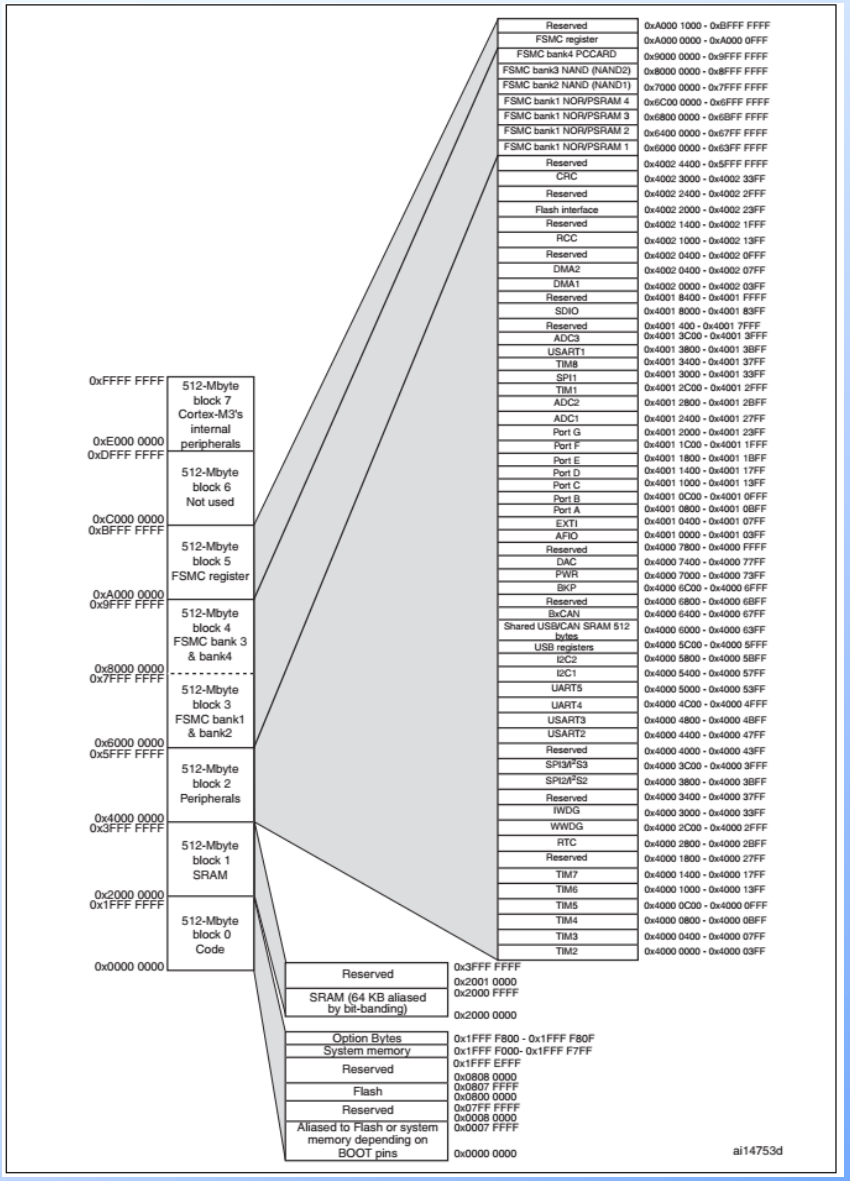
什么叫存储器映射？

存储器本身不具有地址信息，它的地址是由芯片厂商或用户分配，给存储器分配地址的过程就称为存储器映射。

表格 6-1 存储器功能分类

序号	用途	地址范围
Block 0	Code	0x0000 0000 ~ 0x1FFF FFFF(512MB)
Block 1	SRAM	0x2000 0000 ~ 0x3FFF FFFF(512MB)
Block 2	片上外设	0x4000 0000 ~ 0x5FFF FFFF(512MB)
Block 3	FSMC 的 bank1 ~ bank2	0x6000 0000 ~ 0x7FFF FFFF(512MB)
Block 4	FSMC 的 bank3 ~ bank4	0x8000 0000 ~ 0x9FFF FFFF(512MB)
Block 5	FSMC 寄存器	0xA000 0000 ~ 0xCFFF FFFF(512MB)
Block 6	没有使用	0xD000 0000 ~ 0xDFFF FFFF(512MB)
Block 7	Cortex-M3 内部外设	0xE000 0000 ~ 0xFFFF FFFF(512MB)

存储器映射图



具体的可参考

《STM32F103xCDE_数据手册》

4-memory mapping章节

寄存器映射

让GPIOB端口的16个引脚输出高电平，要怎么实现？

通过绝对地址访问内存单元

```
1 // GPIOB 端口全部输出 高电平
2 *(unsigned int*)(0x40010C0C) = 0xFFFF;
```

1、0X40010C0C 是GPIOB输出数据寄存器ODR的地址，如何找到？

2、(unsigned int*)的作用是什么？

2、学会使用C语言的 * 号

寄存器映射

通过寄存器别名方式访问内存单元

```
1 // GPIOB 端口全部输出 高电平
2 #define GPIOB_ODR     (unsignedint*)(0x40010C0C)
3 * GPIOB_ODR = 0xFF;
```

为了方便操作，我们干脆把指针操作“*”也定义到寄存器别名里面

```
1 // GPIOB 端口全部输出 高电平
2 #define GPIOB_ODR    *(unsignedint*)(0x40010C0C)
3 GPIOB_ODR = 0xFF;
```

什么是寄存器

什么是寄存器？

给有特定功能的内存单元取一个别名，这个别名就是我们经常说的寄存器，这个给已经分配好地址的有特定功能的内存单元取别名的过程就叫寄存器映射。

什么叫存储器映射？

给存储器分配地址的过程叫存储器映射，再分配一个地址叫重映射。

STM32寄存器映射

STM32寄存器映射

总线基地址（总线是什么）

- 1. 总线基地址

表格 6-5 总线基地址

总线名称	总线基地址	相对外设基地址的偏移
APB1	0x4000 0000	0x0
APB2	0x4001 0000	0x0001 0000
AHB	0x4001 8000	0x0001 8000

STM32寄存器映射

GPIO基地址（外设是什么）

表格 6-6 外设 GPIO 基地址

外设名称	外设基地址	相对 APB2 总线的地址偏移
GPIOA	0x4001 0800	0x0000 0800
GPIOB	0x4001 0C00	0x0000 0C00
GPIOC	0x4001 1000	0x0000 1000
GPIOD	0x4001 1400	0x0000 1400
GPIOE	0x4001 1800	0x0000 1800
GPIOF	0x4001 1C00	0x0000 1C00
GPIOG	0x4001 2000	0x0000 2000

STM32寄存器映射

GPIOB端口的寄存器列表

表格 6-7 GPIOB 端口的 寄存器地址列表

寄存器名称	寄存器地址	相对 GPIOB 基址的偏移
GPIOB_CRL	0x4001 0C00	0x00
GPIOB_CRH	0x4001 0C00	0x04
GPIOB_IDR	0x4001 0C00	0x08
GPIOB_ODR	0x4001 0C00	0x0C
GPIOH_BSRR	0x4001 0C00	0x10
GPIOH_BRR	0x4001 0C00	0x14
GPIOH_LCKR	0x4001 0C00	0x18

STM32寄存器映射

GPIOx端口数据输出寄存器ODR描述

1 端口位设置/清除寄存器(GPIOx_BSRR) (x=A..E)

2 地址偏移：0x10

复位值：0x0000 0000

3

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

4

位31:16	**BRy**: 清除端口x的位y (y = 0…15) (Port x Reset bit y) 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) (Port x Set bit y) 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

C语言对寄存器的封装

STM32寄存器映射

总线和外设基址宏定义

代码 6-4 总线和外设基址宏定义

```
1  /* 外设基地址 */
2  #define PERIPH_BASE           ((unsigned int)0x40000000)
3
4  /* 总线基地址 */
5  #define APB1PERIPH_BASE       PERIPH_BASE
6  #define APB2PERIPH_BASE       (PERIPH_BASE + 0x00010000)
7  #define AHBPERIPH_BASE        (PERIPH_BASE + 0x00020000)
8
9
10 /* GPIO 外设基地址 */
11 #define GPIOA_BASE            (APB2PERIPH_BASE + 0x0800)
12 #define GPIOB_BASE            (APB2PERIPH_BASE + 0x0C00)
13 #define GPIOC_BASE            (APB2PERIPH_BASE + 0x1000)
14 #define GPIOD_BASE            (APB2PERIPH_BASE + 0x1400)
15 #define GPIOE_BASE            (APB2PERIPH_BASE + 0x1800)
16 #define GPIOF_BASE            (APB2PERIPH_BASE + 0x1C00)
17 #define GPIOG_BASE            (APB2PERIPH_BASE + 0x2000)
18
19
20 /* 寄存器基地址，以 GPIOB 为例 */
21 #define GPIOB_CRL             (GPIOB_BASE+0x00)
22 #define GPIOB_CRH             (GPIOB_BASE+0x04)
23 #define GPIOB_IDR             (GPIOB_BASE+0x08)
24 #define GPIOB_ODR             (GPIOB_BASE+0x0C)
25 #define GPIOB_BSRR            (GPIOB_BASE+0x10)
26 #define GPIOB_BRR             (GPIOB_BASE+0x14)
27 #define GPIOB_LCKR            (GPIOB_BASE+0x18)
```

STM32寄存器映射

让PB0输出低/高电平，要怎么实现？

```
#define PERIPH_BASE       ((unsigned int)0x40000000)
#define APB2PERIPH_BASE   (PERIPH_BASE + 0x00010000)
#define GPIOB_BASE        (APB2PERIPH_BASE + 0x0C00)
#define GPIOB_ODR         *(unsignedint*)(GPIOB_BASE+0x0C)

// PB0输出输出低电平
GPIOB_ODR &= ~(1<<0);

// PB0输出输出高电平
GPIOB_ODR |= (1<<0);
```

STM32寄存器映射

使用结构体封装寄存器列表？

代码 6-6 使用结构体对 GPIO 寄存器组的封装

```
1  typedef unsigned           int uint32_t;  /*无符号 32 位变量*/
2  typedef unsigned short     int uint16_t;  /*无符号 16 位变量*/
3
4  /* GPIO 寄存器列表 */
5  typedef struct {
6      uint32_t CRL;      /*GPIO 端口配置低寄存器      地址偏移: 0x00 */
7      uint32_t ORH;      /*GPIO 端口配置高寄存器      地址偏移: 0x04 */
8      uint32_t IDR;      /*GPIO 数据输入寄存器        地址偏移: 0x08 */
9      uint32_t ODR;      /*GPIO 数据输出寄存器        地址偏移: 0x0C */
10     uint32_t BSRR;     /*GPIO 位设置/清除寄存器     地址偏移: 0x10 */
11     uint32_t BRR;      /*GPIO 端口位清除寄存器      地址偏移: 0x14 */
12     uint16_t LCKR;     /*GPIO 端口配置锁定寄存器    地址偏移: 0x18 */
13 } GPIO_TypeDef;
```

STM32寄存器映射

使用结构体指针访问寄存器

代码 6-7 通过结构体指针访问寄存器

```
1 GPIO_TypeDef  * GPIOx;          //定义一个 GPIO_TypeDef 型结构体指针 GPIOx
2 GPIOx = GPIOB_BASE;             //把指针地址设置为宏 GPIOH_BASE 地址
3 GPIOx->IDR = 0xFFFF;
4 GPIOx->ODR = 0xFFFF;
5
6
7 uint32_t temp;
8 temp = GPIOx->IDR;               //读取 GPIOB_IDR 寄存器的值到变量 temp 中
```

STM32寄存器映射

定义GPIO端口基地址指针

代码 6-8 定义好 GPIO 端口首地址址针

```
1  /*使用 GPIO_TypeDef 把地址强制转换成指针*/
2  #define GPIOA                ((GPIO_TypeDef *) GPIOA_BASE)
3  #define GPIOB                ((GPIO_TypeDef *) GPIOB_BASE)
4  #define GPIOC                ((GPIO_TypeDef *) GPIOC_BASE)
5  #define GPIOD                ((GPIO_TypeDef *) GPIOD_BASE)
6  #define GPIOE                ((GPIO_TypeDef *) GPIOE_BASE)
7  #define GPIOF                ((GPIO_TypeDef *) GPIOF_BASE)
8  #define GPIOG                ((GPIO_TypeDef *) GPIOG_BASE)
9  #define GPIOH                ((GPIO_TypeDef *) GPIOH_BASE)
10
11
12
13 /*使用定义好的宏直接访问*/
14 /*访问 GPIOB 端口的寄存器*/
15 GPIOB->BSRR = 0xFFFF;          //通过指针访问并修改 GPIOB_BSRR 寄存器
16 GPIOB->CRL = 0xFFFF;           //修改 GPIOB_CRL 寄存器
17 GPIOB->ODR =0xFFFF;           //修改 GPIOB_ODR 寄存器
18
19 uint32_t temp;
20 temp = GPIOB->IDR;            //读取 GPIOB_IDR 寄存器的值到变量 temp 中
21
22 /*访问 GPIOA 端口的寄存器*/
23 GPIOA->BSRR = 0xFFFF;
24 GPIOA->CRL = 0xFFFF;
25 GPIOA->ODR =0xFFFF;
26
27 uint32_t temp;
28 temp = GPIOA->IDR;            //读取 GPIOA_IDR 寄存器的值到变量 temp 中
```

STM32寄存器映射

这里我们仅是以GPIO这个外设为例，给大家讲解了C语言对寄存器的封装。以此类推，其他外设也同样可以用这种方法来封装。

好消息是，这部分工作都由固件库帮我们完成了，这里我们只是分析了下这个封装的过程，让大家知其然，也只其所以然。

零死角玩转STM32—M3系列

THANKS

论坛： www.firebbs.cn

淘宝：firestm32.taobao.com

扫描进入淘宝店铺