零死角玩转STM32—M4系列

什么是寄存器

淘宝：firestm32.taobao.com

论坛： www.chuxue123.com

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？

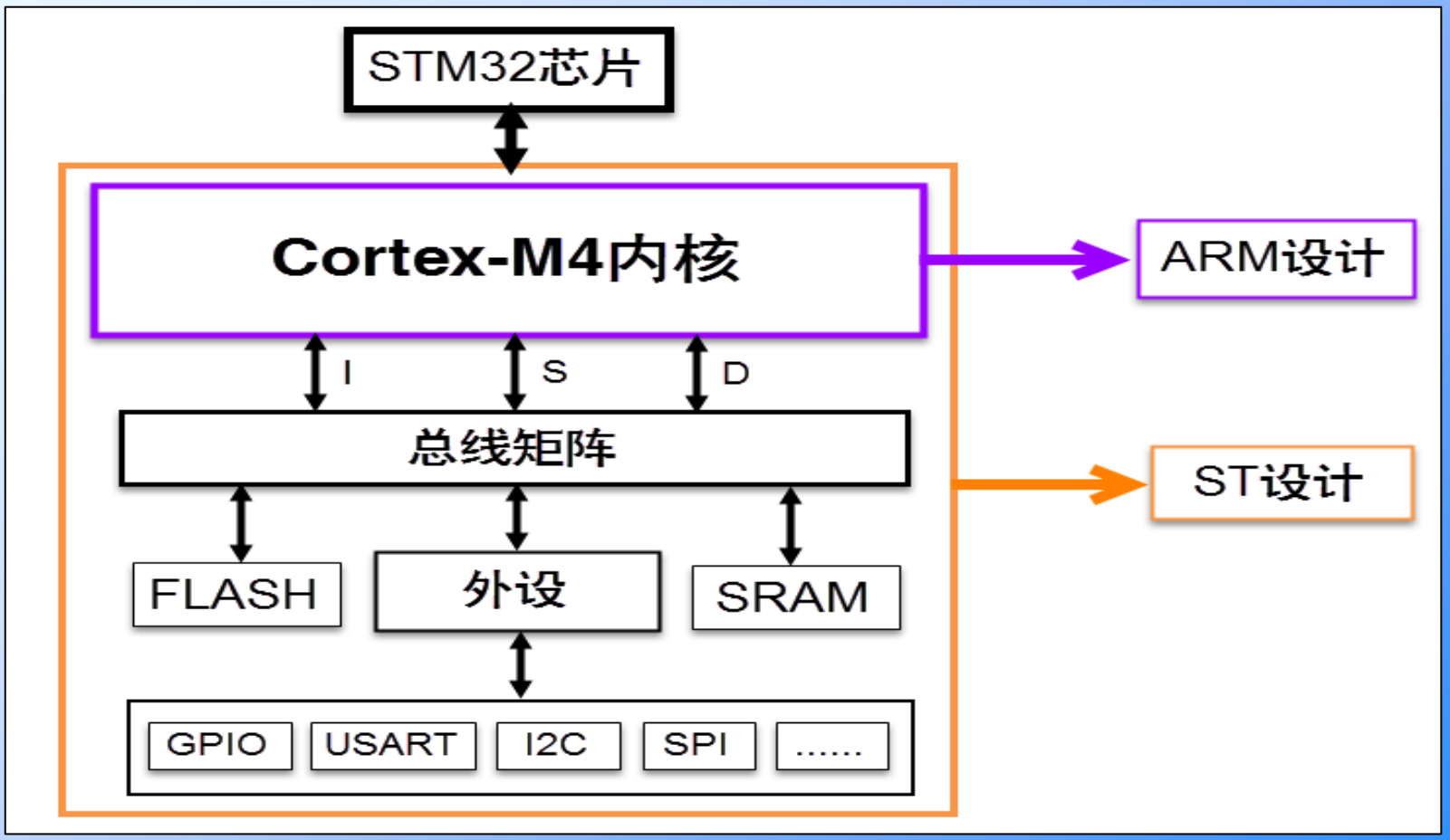
STM32长啥样

STM32F429IGT6芯片实物图

1、学会看丝印
2、懂得如何辨别正方向

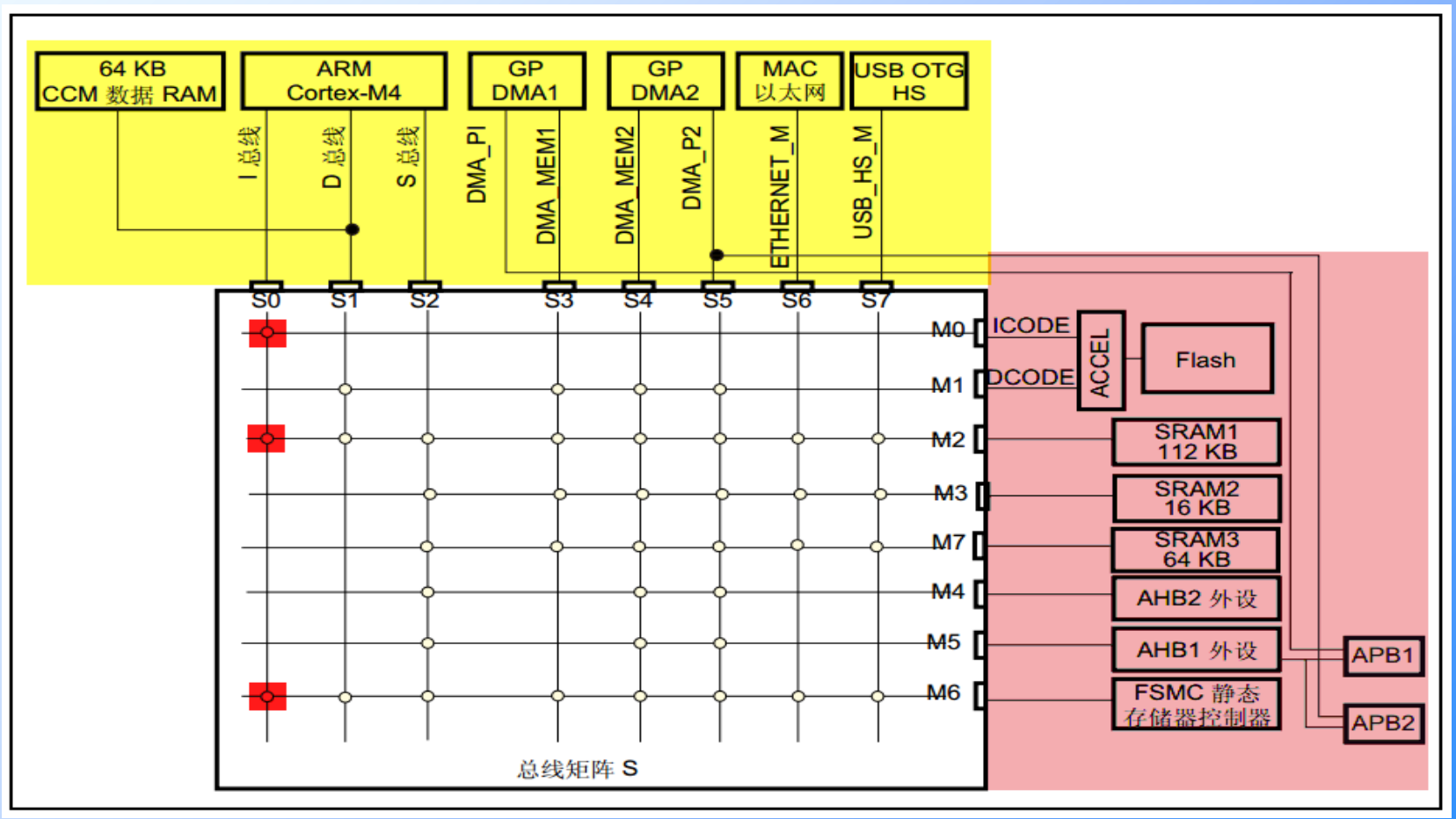
STM32内部有什么

STM32芯片架构简图



STM32内部有什么

STM32F42xx和F43xx器件的总线接口



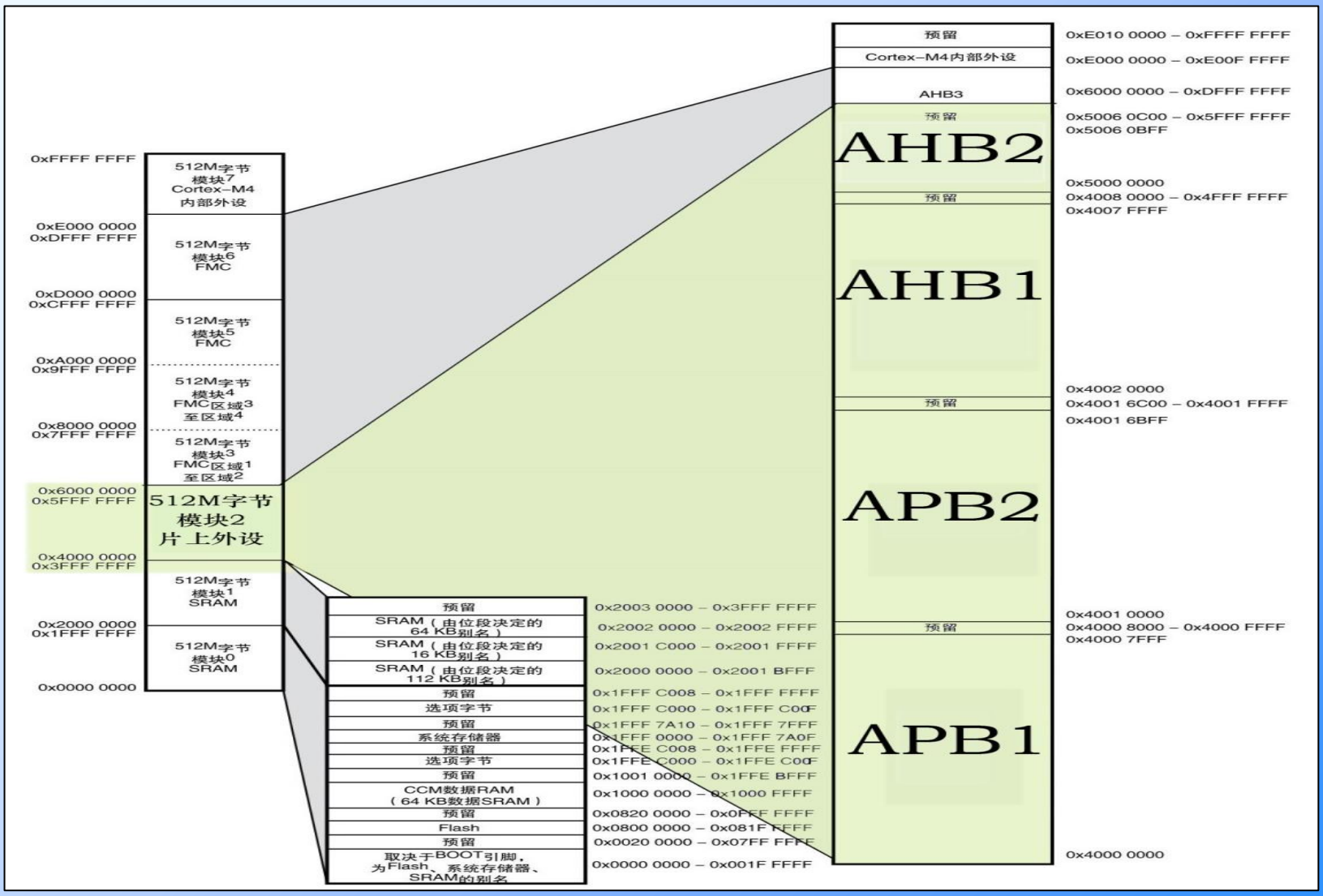
存储器映射

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。

序号	用途	地址范围
Block 0	SRAM（FLASH）	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	FMC的bank1 ~ bank2	0x6000 0000 ~ 0x7FFF FFFF(512MB)
Block 4	FMC的bank3 ~ bank4	0x8000 0000 ~ 0x9FFF FFFF(512MB)
Block 5	FMC	0xA000 0000 ~ 0xCFFF FFFF(512MB)
Block 6	FMC	0xD000 0000 ~ 0xDFFF FFFF(512MB)
Block 7	Cortex-M4内部外设	0xE000 0000 ~ 0xFFFF FFFF(512MB)

存储器映射图


0xFFFF FFFF
512M字节 模块7 Cortex-M4 内部外设
0xE000 0000
0xDFFF FFFF
512M字节 模块6 FMC
0xD000 0000
0xCFFF FFFF
512M字节 模块5 FMC
0xA000 0000
0x9FFF FFFF
512M字节 模块4 FMC区域3至区域4
0x8000 0000
0x7FFF FFFF
512M字节 模块3 FMC区域1至区域2
0x6000 0000
0x5FFF FFFF
512M字节 模块2 片上外设
0x4000 0000
0x3FFF FFFF
512M字节 模块1 SRAM
0x2000 0000
0x1FFF FFFF
512M字节 模块0 SRAM
0x0000 0000
预留
SRAM（由位段决定的64 KB别名）
SRAM（由位段决定的16 KB别名）
SRAM（由位段决定的112 KB别名）
预留
选项字节
预留
系统存储器
预留
选项字节
预留
CCM数据RAM（64 KB数据SRAM）
预留
Flash
预留
取决于BOOT引脚，为Flash、系统存储器、SRAM的别名
0x2003 0000 – 0x3FFF FFFF
0x2002 0000 – 0x2002 FFFF
0x2001 C000 – 0x2001 FFFF
0x2000 0000 – 0x2001 BFFF
0x1FFF C008 – 0x1FFF FFFF
0x1FFF C000 – 0x1FFF C00F
0x1FFF 7A10 – 0x1FFF 7FFF
0x1FFF 0000 – 0x1FFF 7A0F
0x1FFE C008 – 0x1FFE FFFF
0x1FFE C000 – 0x1FFE C00F
0x1001 0000 – 0x1FFE BFFF
0x1000 0000 – 0x1000 FFFF
0x0820 0000 – 0x0FFF FFFF
0x0800 0000 – 0x081F FFFF
0x0020 0000 – 0x07FF FFFF
0x0000 0000 – 0x001F FFFF
预留
Cortex-M4内部外设
AHB3
预留
AHB2
预留
AHB1
预留
APB2
预留
APB1
0xE010 0000 – 0xFFFF FFFF
0xE000 0000 – 0xE00F FFFF
0x6000 0000 – 0xDFFF FFFF
0x5006 0C00 – 0x5FFF FFFF
0x5006 0BFF
0x5000 0000
0x4008 0000 – 0x4FFF FFFF
0x4007 FFFF
0x4002 0000
0x4001 6C00 – 0x4001 FFFF
0x4001 6BFF
0x4001 0000
0x4000 8000 – 0x4000 FFFF
0x4000 7FFF
0x4000 0000

寄存器映射

让GPIOH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 // GPIOH 端口全部输出 高电平
2 *(unsigned int*)(0x40021C14) = 0xFFFF;
```

1、0X40021C14 是GPIOH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 // GPIOH 端口全部输出 高电平
2 #define GPIOH_ODR     (unsignedint*)(0x40021C14)
3 * GPIOH_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 // GPIOH 端口全部输出 高电平
2 #define GPIOH_ODR    *(unsignedint*)(0x40021C14)
3 GPIOH_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
AHB1	0x4002 0000	0x0002 0000
AHB2	0x5000 0000	0x1000 0000
AHB3	0x6000 0000	已不属于片上外设

STM32寄存器映射

GPIO基地址

外设名称	外设基地址	相对AHB1总线的地址偏移
GPIOA	0x4002 0000	0x0
GPIOB	0x4002 0400	0x0000 0400
GPIOC	0x4002 0800	0x0000 0800
GPIOD	0x4002 0C00	0x0000 0C00
GPIOE	0x4002 1000	0x0000 1000
GPIOF	0x4002 1400	0x0000 1400
GPIOG	0x4002 1800	0x0000 1800
GPIOH	0x4002 1C00	0x0000 1C00

STM32寄存器映射

GPIOH端口的寄存器列表

寄存器名称	寄存器地址	相对GPIOH基址的偏移
GPIOH_MODER	0x4002 1C00	0x00
GPIOH_OTYPER	0x4002 1C04	0x04
GPIOH_OSPEEDR	0x4002 1C08	0x08
GPIOH_PUPDR	0x4002 1C0C	0x0C
GPIOH_IDR	0x4002 1C10	0x10
GPIOH_ODR	0x4002 1C14	0x14
GPIOH_BSRR	0x4002 1C18	0x18
GPIOH_LCKR	0x4002 1C1C	0x1C
GPIOH_AFRL	0x4002 1C20	0x20
GPIOH_AFRH	0x4002 1C24	0x24

STM32寄存器映射

GPIOH端口的寄存器列表

寄存器名称	寄存器地址	相对GPIOH基址的偏移
GPIOH_MODER	0x4002 1C00	0x00
GPIOH_OTYPER	0x4002 1C04	0x04
GPIOH_OSPEEDR	0x4002 1C08	0x08
GPIOH_PUPDR	0x4002 1C0C	0x0C
GPIOH_IDR	0x4002 1C10	0x10
GPIOH_ODR	0x4002 1C14	0x14
GPIOH_BSRR	0x4002 1C18	0x18
GPIOH_LCKR	0x4002 1C1C	0x1C
GPIOH_AFRL	0x4002 1C20	0x20
GPIOH_AFRH	0x4002 1C24	0x24

STM32寄存器映射

GPIOx端口数据输出寄存器ODR描述

GPIO 端口输出数据寄存器 (GPIOx_ODR) (x = A..I) 1

GPIO port output data register

偏移地址：0x14 2

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
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

3

位 31:16 保留，必须保持复位值。

位 15:0 **ODRy[15:0]**：端口输出数据 (Port output data) (y = 0..15)

这些位可通过软件读取和写入。

注意：对于原子置位/复位，通过写入 GPIOx_BSRR 寄存器，可分别对 ODR 位进行置位和复位 (x = A..I)。

4

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

```
 1 /* 外设基地址 */
 2 #define PERIPH_BASE           ((unsigned int)0x40000000)
 4 /* 总线基地址 */
 5 #define APB1PERIPH_BASE       PERIPH_BASE
 6 #define APB2PERIPH_BASE       (PERIPH_BASE + 0x00010000)
 7 #define AHB1PERIPH_BASE       (PERIPH_BASE + 0x00020000)
 8 #define AHB2PERIPH_BASE       (PERIPH_BASE + 0x10000000)
10 /* GPIO外设基地址 */
11 #define GPIOA_BASE            (AHB1PERIPH_BASE + 0x0000)
12 #define GPIOB_BASE            (AHB1PERIPH_BASE + 0x0400)
13 #define GPIOC_BASE            (AHB1PERIPH_BASE + 0x0800)
14 #define GPIOD_BASE            (AHB1PERIPH_BASE + 0x0C00)
15 #define GPIOE_BASE            (AHB1PERIPH_BASE + 0x1000)
16 #define GPIOF_BASE            (AHB1PERIPH_BASE + 0x1400)
17 #define GPIOG_BASE            (AHB1PERIPH_BASE + 0x1800)
18 #define GPIOH_BASE            (AHB1PERIPH_BASE + 0x1C00)
20 /* 寄存器基地址，以GPIOH为例 */
21 #define GPIOH_MODER             (GPIOH_BASE+0x00)
22 #define GPIOH_OTYPER            (GPIOH_BASE+0x04)
23 #define GPIOH_OSPEEDR           (GPIOH_BASE+0x08)
24 #define GPIOH_PUPDR             (GPIOH_BASE+0x0C)
25 #define GPIOH_IDR               (GPIOH_BASE+0x10)
26 #define GPIOH_ODR               (GPIOH_BASE+0x14)
27 #define GPIOH_BSRR              (GPIOH_BASE+0x18)
28 #define GPIOH_LCKR              (GPIOH_BASE+0x1C)
29 #define GPIOH_AFRL              (GPIOH_BASE+0x20)
30 #define GPIOH_AFRH              (GPIOH_BASE+0x24)
```

STM32寄存器映射

让PH10输出低/高电平，要怎么实现？

```
#define PERIPH_BASE       ((unsigned int)0x40000000)
#define AHB1PERIPH_BASE   (PERIPH_BASE + 0x00020000)
#define GPIOH_BASE        (AHB1PERIPH_BASE + 0x1C00)
#define GPIOH_ODR         *(unsignedint*)(GPIOH_BASE+0x14)

// PH10输出输出低电平
GPIOH_ODR &= ~(1<<10);

// PH10输出输出高电平
GPIOH_ODR |= (1<<10);
```

STM32寄存器映射

使用结构体封装寄存器列表？

```
typedef unsigned           int uint32_t; /*无符号32位变量*/
typedef unsigned short     int uint16_t; /*无符号16位变量*/

/* GPIO寄存器列表 */
typedef struct {
     uint32_t MODER;     /*GPIO模式寄存器                地址偏移: 0x00        */
     uint32_t OTYPER;    /*GPIO输出类型寄存器            地址偏移: 0x04        */
     uint32_t OSPEEDR;   /*GPIO输出速度寄存器            地址偏移: 0x08        */
     uint32_t PUPDR;     /*GPIO上拉/下拉寄存器           地址偏移: 0x0C        */
     uint32_t IDR;       /*GPIO输入数据寄存器            地址偏移: 0x10        */
     uint32_t ODR;       /*GPIO输出数据寄存器            地址偏移: 0x14        */
     uint16_t BSRRL;     /*GPIO置位/复位寄存器低16位部分 地址偏移: 0x18        */
     uint16_t BSRRH;     /*GPIO置位/复位寄存器高16位部分 地址偏移: 0x1A        */
     uint32_t LCKR;      /*GPIO配置锁定寄存器            地址偏移: 0x1C        */
     uint32_t AFR[2];    /*GPIO复用功能配置寄存器        地址偏移: 0x20-0x24   */
} GPIO_TypeDef;
```

STM32寄存器映射

使用结构体指针访问寄存器

```
GPIO_TypeDef  *GPIOx;              //定义一个GPIO_TypeDef类型的结构体指针GPIOx
GPIOx  =  GPIOH_BASE;              //把指针地址设置为宏GPIOH_BASE地址
GPIOx->BSRRL  =  0xFFFF;           //通过指针访问并修改GPIOH_BSRRL寄存器
GPIOx->MODER  =  0xFFFFFFFF;       //修改GPIOH_MODER寄存器
GPIOx->OTYPER  =0xFFFFFFFF;        //修改GPIOH_OTYPER寄存器


uint32_t  temp;
temp  =  GPIOx->IDR;               //读取GPIOH_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

```
/*使用GPIO_TypeDef把地址强制转换成指针*/
#define GPIOA                ((GPIO_TypeDef *) GPIOA_BASE)
#define GPIOB                ((GPIO_TypeDef *) GPIOB_BASE)
#define GPIOC                ((GPIO_TypeDef *) GPIOC_BASE)
#define GPIOD                ((GPIO_TypeDef *) GPIOD_BASE)
#define GPIOE                ((GPIO_TypeDef *) GPIOE_BASE)
#define GPIOF                ((GPIO_TypeDef *) GPIOF_BASE)
#define GPIOG                ((GPIO_TypeDef *) GPIOG_BASE)
#define GPIOH                ((GPIO_TypeDef *) GPIOH_BASE)
```

```
GPIOA->ODR = 0xFF;

GPIOB->ODR = 0xFF;

... ...
```

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—M4系列

THANKS

论坛： www.chuxue123.com

淘宝：firestm32.taobao.com

扫描进入淘宝店铺