STM32库函数架构剖析

目录

STM32库函数

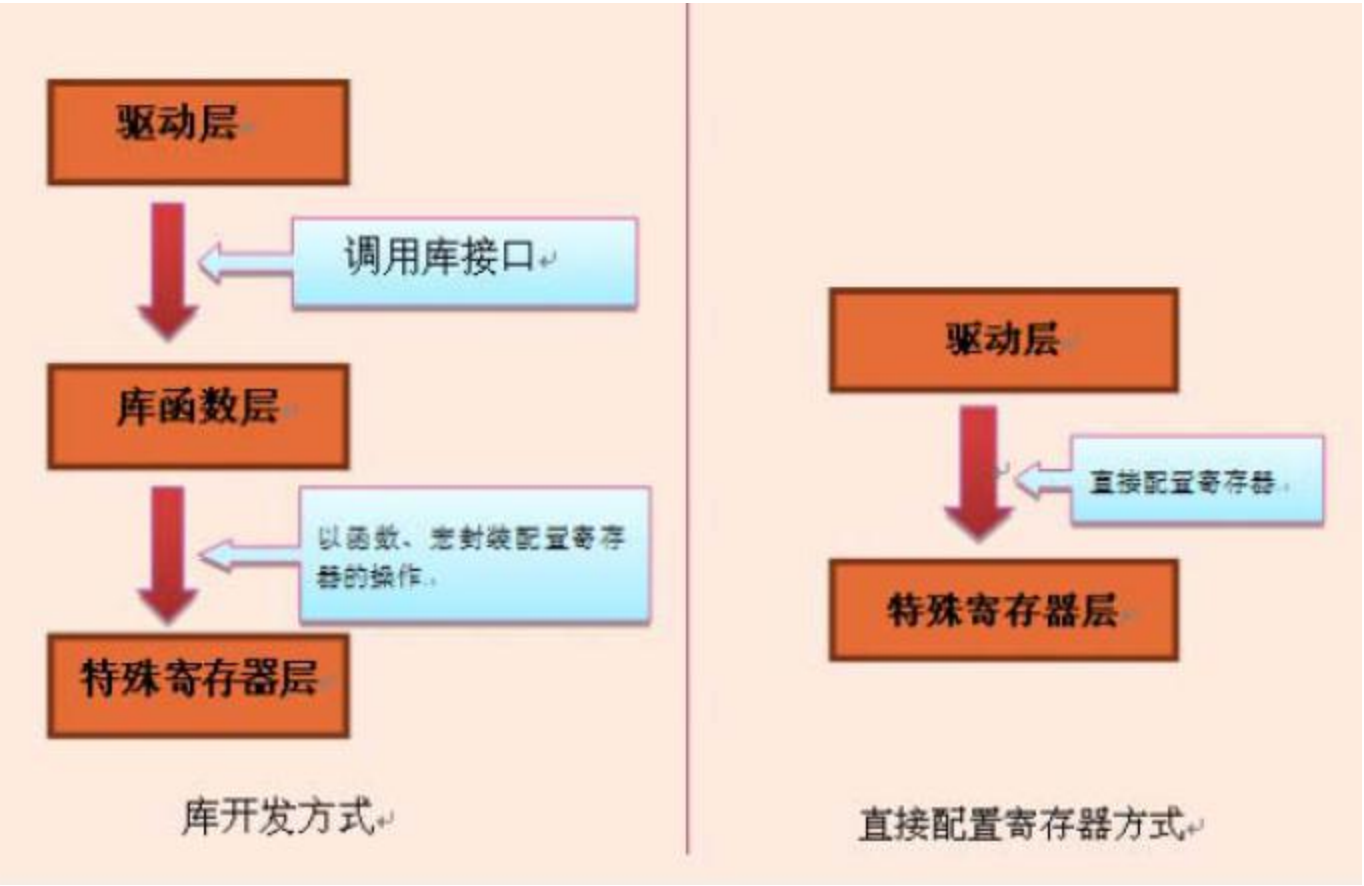
- STM32的库函数是ST公司已经封装好的一个软件封装库，也就是很多基础的代码，在开发产品的时候只需直接调用这个ST库函数的函数接口就可以完成一系列工作。

库函数的益处

- 在51单片机进行开发时，使用内部资源，查阅手册，配置寄存器。
- STM32库是由ST公司针对STM32提供的函数接口，即API(Application Program Interface), 开发者可调用这些函数接口来配置STM32的寄存器，使得开发人员得以脱离底层的寄存器操作。开发快速，易于阅读，维护成本低。
  - C语言-printf()函数使用

Cont.

库是架设在寄存器与用户驱动层之间的代码，向下处理与寄存器直接相关的配置，向上为用户提供配置寄存器的接口。

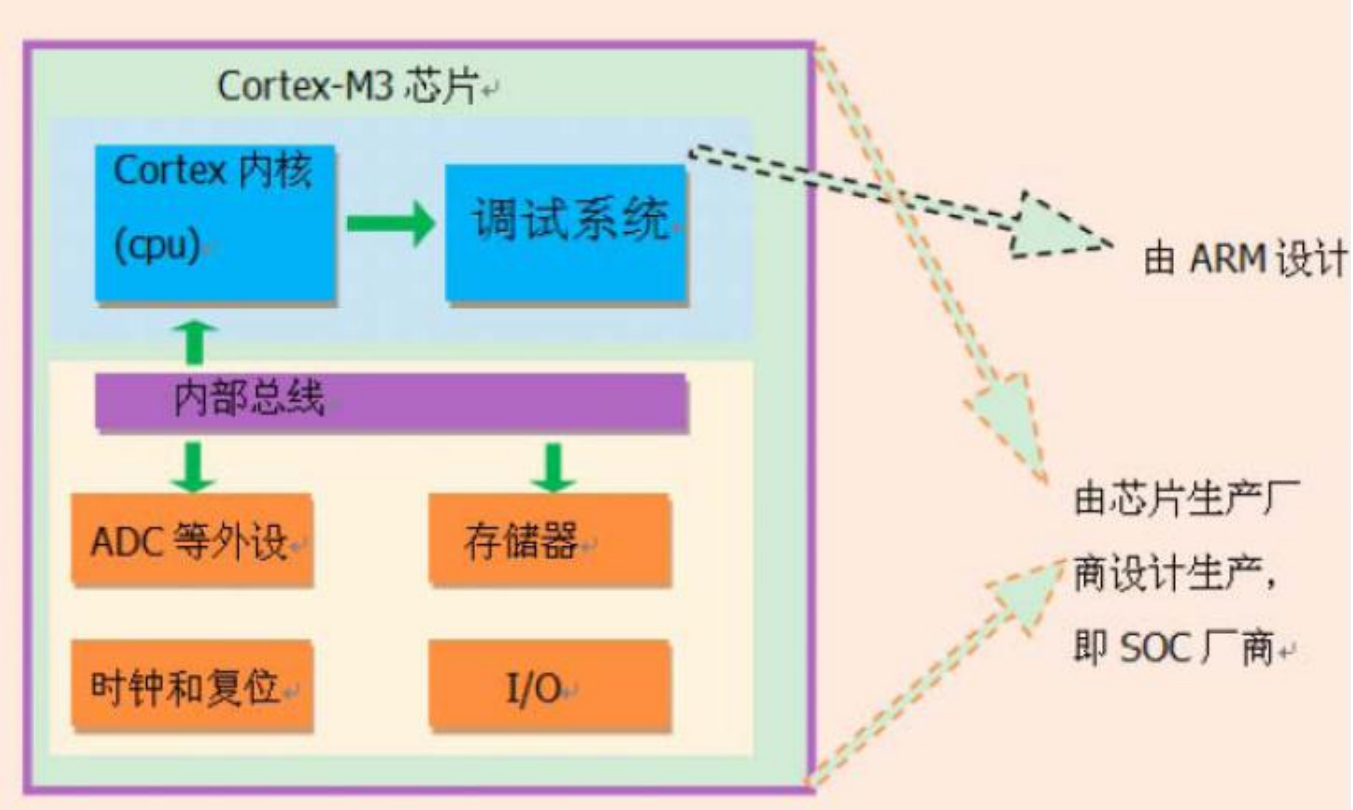


Cont.

- 千人大项目的开发，任务分配问题。
  - 底层的硬件级，寄存器的读写封装等
  - 根据底层这群人提供的接口，同步做二次开发。
- 库函数的理念完全被广泛应用于各种实际的项目和产品中，协同工作，大项目工作合理展开。

CMSIS标准

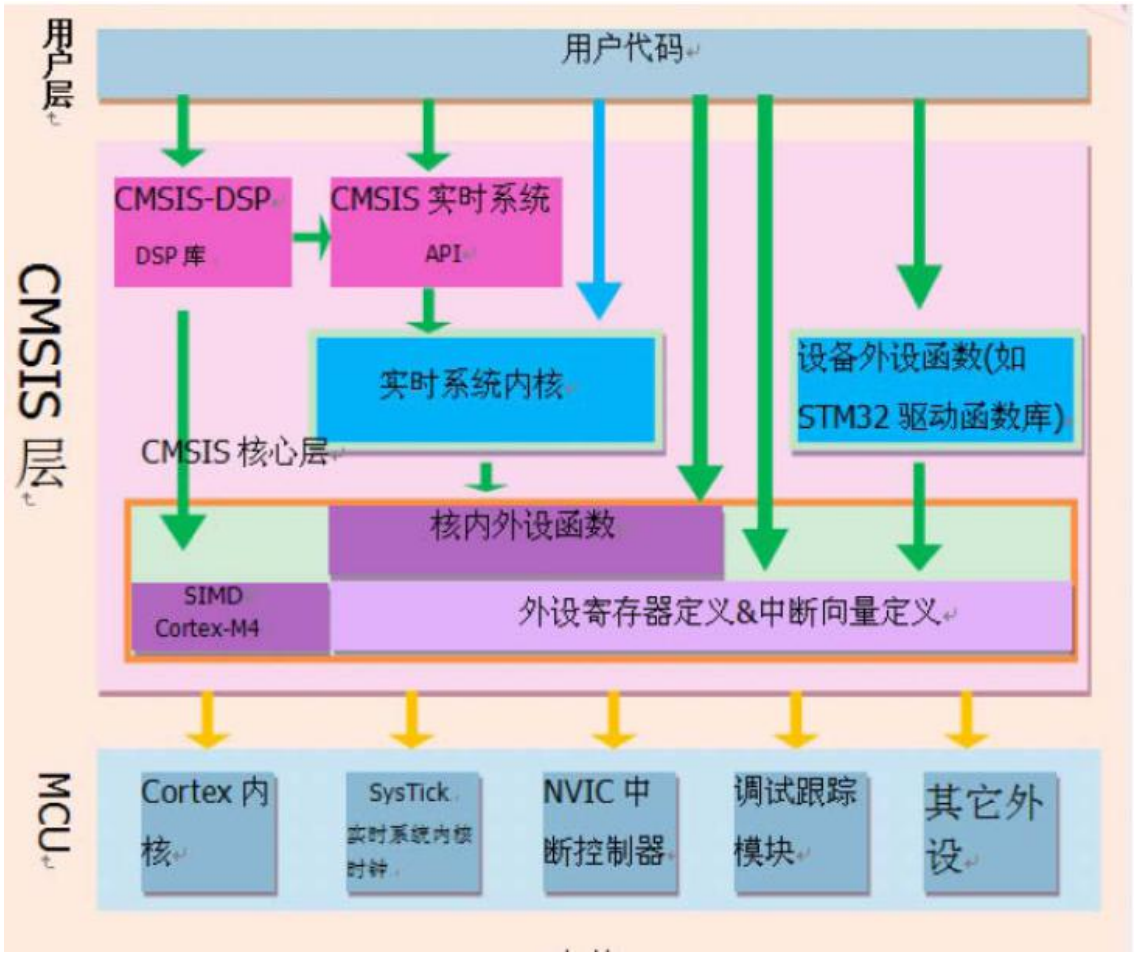
- ST公司生产的STM32芯片采用的是Cortex-M3或Cortex-M4内核。内核是由ARM公司设计的一个处理器体系架构。内核是整个微控制器的CPU.



Cont.

- 基于Cortex的某系列芯片采用的内核都是相同的，区别主要为核外的片上外设的差异，这些差异却导致软件在同内核，不同外设的芯片上移植困难。为了解决不同的芯片厂商生产的Cortex微控制器软件的兼容性问题，ARM与芯片厂商建立 了CMSIS标准（Cortex MicroController Software Interface Standard)--------实际为软件抽象层

Cont.


用户层
用户代码
CMSIS 层
CMSIS-DSP
DSP 库
CMSIS 实时系统
API
实时系统内核
CMSIS 核心层
设备外设函数(如
STM32 驱动函数库)
核内外设函数
SIMD
Cortex-M4
外设寄存器定义&中断向量定义
MCU
Cortex 内
核
SysTick
实时系统内核
时钟
NVIC 中
断控制器
调试跟踪
模块
其它外
设

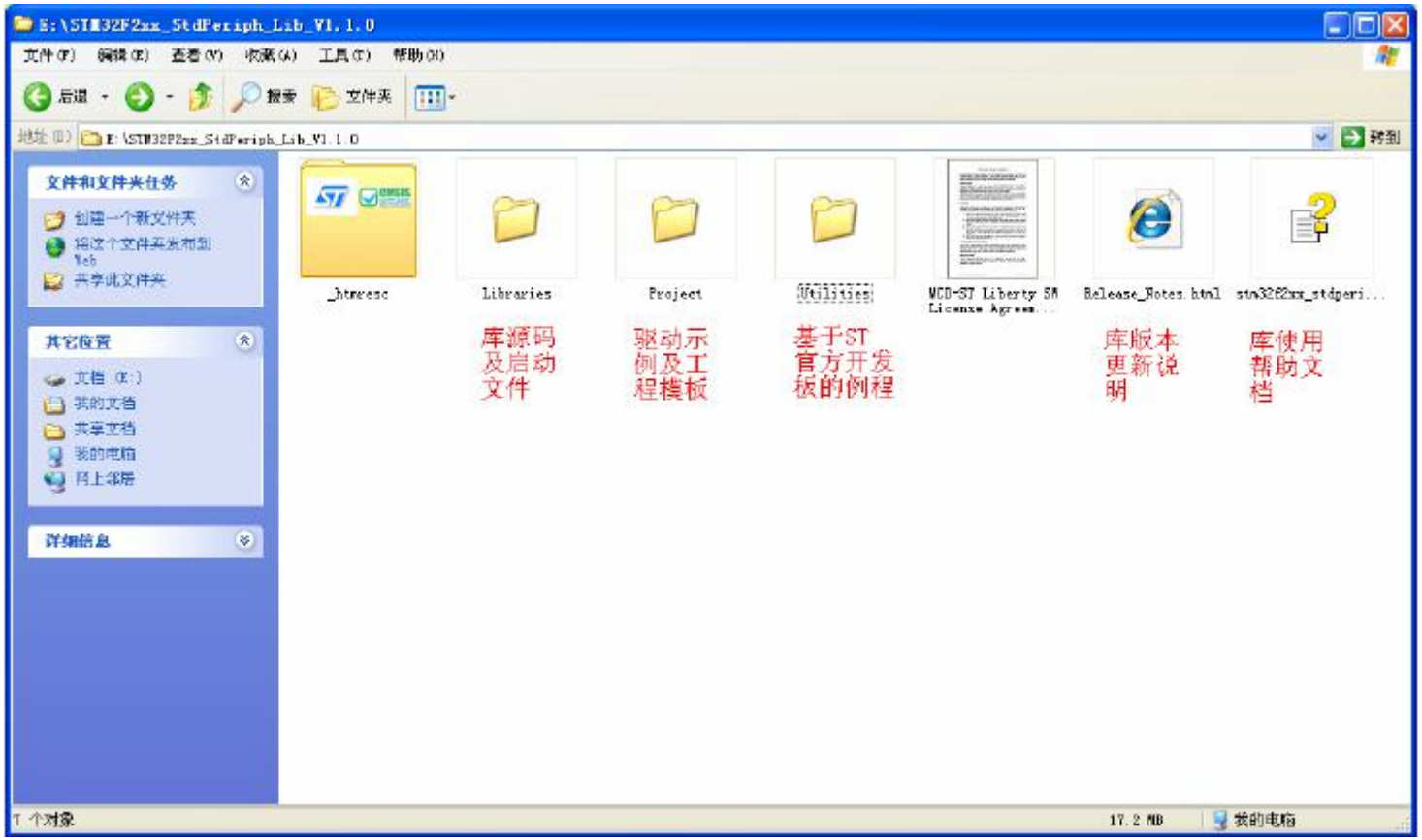
CMSIS标准

- CMSIS标准中最主要的为CMSIS核心层，包括：
  - 内核函数层：其中包含用于访问内核寄存器的名称、地址定义，主要由ARM公司提供
  - 设备外设访问层：提供了片上的核外外设的地址和中断定义，主要由芯片生产商提供。
- CMSIS层位于硬件层与操作系统或用户层之间，提供了与芯片生产商无关的硬件抽象层，可以为接口外设、实时操作系统提供简单的处理器软件接口，屏蔽了硬件差异。这对软件移植好处极大。STM32的库，就是按照CMSIS标准建立。

库目录及文件

- 一般库目录：
- Libraries文件夹下是驱动库的源代码及启动文件。
  - 内核与外设的库文件存放于CMSIS文件夹中。
  - 驱动在STM32F4xx_StdPeriph_Driver
- Project文件夹下是用驱动库写的例子跟一个工程模板。

Cont.


E:\STM32F2xx_StdPeriph_Lib_V1.1.0
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退 搜索 文件夹
地址(D) E:\STM32F2xx_StdPeriph_Lib_V1.1.0
转到
文件和文件夹任务
创建一个新文件夹
将这个文件夹发布到 Web
共享此文件夹
其它位置
文档 (E:)
我的文档
共享文档
我的电脑
网上邻居
详细信息
_htmresc
Libraries
Project
Utilities
MCD-ST Liberty SA License Agreem...
Release_Notes.html
stm32f2xx_stdperi...
库源码及启动文件
驱动示例及工程模板
基于ST官方开发板的例程
库版本更新说明
库使用帮助文档
7 个对象
17.2 MB
我的电脑

core_cm3.h

- CMSIS标准的核内设备函数层M3核通用的源文件core_cm3.c和头文件core_cm3.h,它们的作用为那些采用Cortex-M3核设计SOC的芯片商设计的芯片外设提供一个进入M3内核的接口。这两个文件在其他公司的M3系列芯片也是相同的。程序具体如何实现，目前不用管。

✓ C语言复习

- 位操作
- define宏定义关键词
- ifdef条件编译
- extern变量申明
- typedef类型别名
- 结构体
- static关键字

✓ C语言复习

■ 位操作：6种位操作运算符

运算符	含义	运算符	含义
&	按位与	~	取反
|	按位或	<<	左移
^	按位异或	>>	右移

GPIOA->ODR=(((uint32_t)0x01)<<pinpos;
GPIOA->ODR=0x0030;设置第pinpos位为1.

TIMx->SR = (uint16_t)~TIM_FLAG;
TIMx->SR=0xfff7;设置第3位为0；

✓ C语言复习

■ define宏定义关键词

define是C语言中的预处理命令，它用于宏定义，可以提高源代码的可读性，为编程提供方便。

常见的格式：

#define 标识符　字符串

“标识符”为所定义的宏名。“字符串”可以是常数、表达式、格式串等。

例如：

#define SYSCLK_FREQ_72MHz 72000000

定义标识符SYSCLK_FREQ_72MHz的值为72000000。

✓ C语言复习

■ ifdef条件编译

单片机程序开发过程中，经常会遇到一种情况，当满足某条件时对一组语句进行编译，而当条件不满足时则编译另一组语句。条件编译命令最常见的形式为：

```
#ifdef 标识符
程序段1
#else
程序段2
#endif
```

例如：

```
#ifdef STM32F10X_HD
大容量芯片需要的一些变量定义
#end
```

✓ C语言复习

■ extern变量申明

C语言中*extern*可以置于变量或者函数前，以表示变量或者函数的定义在别的文件中，提示编译器遇到此变量和函数时在其他模块中寻找其定义。

这里面要注意，对于extern申明变量可以多次，但定义只有一次。

✓ C语言复习

- extern变量申明

main.c文件

```
u8 id;//定义只允许一次
main()
{
id=1;
printf("d%",id);//id=1
test();
printf("d%",id);//id=2
}
```

test.c文件

```
extern u8 id;

void test(void){
id=2;
}
```

✓ C语言复习

■ typedef类型别名

定义一种类型的别名，而不只是简单的宏替换。可以用作同时声明指针型的多个对象。

```
typedef unsigned   char      uint8_t;
typedef unsigned short  int uint16_t;
typedef unsigned   int       uint32_t;
typedef unsigned  __int64  uint64_t;
```

✓ C语言复习

■ 结构体：构造类型

```
Struct 结构体名{
成员列表1;
成员变量2;
…
}变量名列表;
```

在结构体申明的时候可以定义变量，也可以申明之后定义，方法是：
Struct 结构体名字 结构体变量列表；

✓ C语言复习

- 结构体作用：

  同一个类型可以用数组，不同类型可以用结构体组织。

  结构体可扩展性强。

  举例说明：

  void GPIO_Init(GPIO_TypeDef* GPIOx, GPIO_InitTypeDef* GPIO_InitStruct)

◆ C语言关键字： static

- Static申明的局部变量，存储在静态存储区。
- 它在函数调用结束之后，不会被释放。它的值会一直保留下来。
- 所以可以说static申明的局部变量，具有记忆功能。

◆每次调用**getValue**函数之后，返回值是多少？

```
int getValue(void)
{
  int  flag=0;
  flag++;
  return flag;
}
```

```
int getValue(void)
{
  static int  flag=0;
  flag++;
  return flag;
}
```

■ 51中映射方法：

sfr P0 =0x80;//P0映射到地址0x80

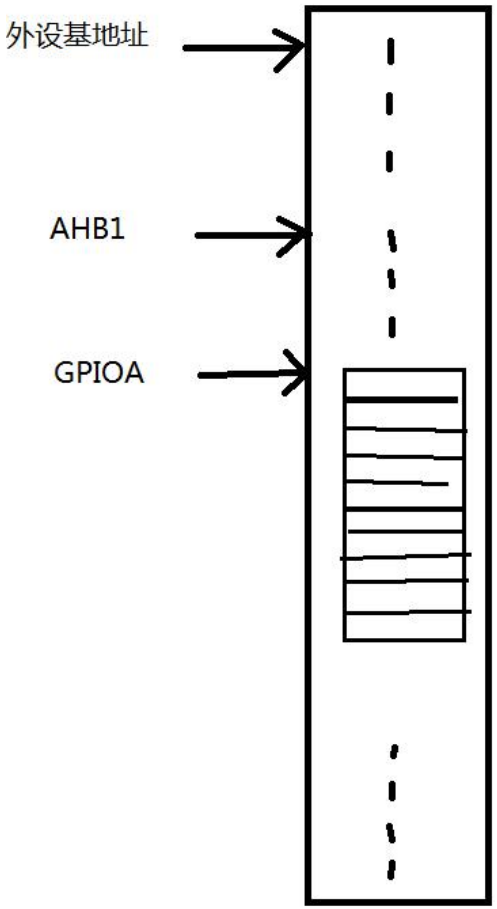
P0=0x00//寄存器地址0x80赋值0x00

■ STM32中操作：

GPIOA->ODR=0x00000000;

值0x00000000是怎么赋值给了GPIOA的ODR寄存器地址的呢？

也就是说GPIOA->ODR这种写法，是怎么与GPIOA的ODR寄存器地址映射起来的？



#define PERIPH_BASE ((uint32_t)0x40000000)

#define AHB1PERIPH_BASE (PERIPH_BASE + 0x00020000)

#define GPIOA_BASE (AHB1PERIPH_BASE + 0x0000)

#define GPIOA ((GPIO_TypeDef *) GPIOA_BASE)

```
typedef struct
{
  __IO uint32_t MODER;
  __IO uint32_t OTYPER;
  __IO uint32_t OSPEEDR;
  __IO uint32_t PUPDR;
  __IO uint32_t IDR;
  __IO uint32_t ODR;
  __IO uint16_t BSRRL;
  __IO uint16_t BSRRH;
  __IO uint32_t LCKR;
  __IO uint32_t AFR[2];
} GPIO_TypeDef;
```

◆ STM32F40x系列GPIOA寄存器地址映射

偏移	寄存器
0x00	GPIOA_MODER
0x04	GPIOA_OTYPER
0x08	GPIOA_OSPEEDER
0x0C	GPIOA_PUPDR
0x10	GPIOA_IDR
0x14	GPIOA_ODR
0x18	GPIOA_BSRR
0x1c	GPIOA_LCKR
0x20	GPIOA_AFRL
0x24	GPIOA_AFRH

表 4.6.1 GIPOA 寄存器地址偏移表