单片机中的C语言

1 stdint.h

stdint.h是从C99中引进的一个标准C库的文件，用于统一类型的别名

```
   /* exact-width signed integer types */
typedef signed       char           int8_t  ;
typedef signed short int            int16_t ;
typedef signed       int            int32_t ;
typedef signed       __INT64        int64_t ;

   /* exact-width unsigned integer types */
typedef unsigned       char         uint8_t ;
typedef unsigned short int          uint16_t    ;
typedef unsigned       int          uint32_t    ;
typedef unsigned       __INT64      uint64_t    ;
```

2 位操作

运算符	含义	运算符	含义
&	按位与	~	按位取反
|	按位或	<<	左移
^	按位异或	>>	右移(正补0负补1)

如何给寄存器某个位幅值?

```
uint32_t    temp = 0 ;  //temp的二进制为0000_0000_0000_0000_0000_0000_0000_0000

//第一个栗子：将temp的位6赋值为1（先清零再置一）
/*****方法一*****/
temp &= 0xFFFFFFBF ;    //0xFFFFFFBF => 1111_1111_1111_1111_1111_1111_1011_1111
temp |= 0x00000040 ;    //0x00000040 => 0000_0000_0000_0000_0000_0000_0100_0000
/*****方法二*****/
temp &= ~(1<<6) ;
temp |= 1<<6    ;

//第二个栗子：按位异或用于控制位6翻转
temp ^= 1<<6 ;
```

3 宏定义

提高效率、可读性、易改性

```
/*
#define 标识符 字符串
标识符：宏定义的名字
字符串：常数、表达式、格式串等
*/
//举个栗子
#define PI          3.14159
#define HSE_VALUE   8000000U
```

带参数的宏定义

```
//栗子：LED亮灭
#define LED(x) do{ x?\
    GPIO_WritePin(LED_GPIO_PORT,LED_GPIO_PIN,GPIO_PIN_SET):\
    GPIO_WritePin(LED_GPIO_PORT,LED_GPIO_PIN,GPIO_PIN_RESET):\
}while(0)
```

4 条件编译

让编译器只对满足条件的代码进行编译，不满足条件的不参与编译

指令	作用
#if	编译预处理条件指令，类似if
#ifdef	判断某个宏是否已被定义
#ifndef	判断某个宏是否未被定义
#elif	若前面的条件不满足，则判断新条件，类似else if
#else	若前面的条件不满足，则执行后面的语句，类似else
#endif	#if、#ifdef、#ifndef的结束标志

```
//第一个栗子：头文件的条件编译，可以避免头文件被重复引用编译，减少编译时间
#ifndef _LED_H
#define _LED_H
#include "./SYSTEM/sys/sys.h"
//code...
//code...
//code...
#endif

//第二个栗子：代码的条件编译,是否执行code
#if SYS_SUPPORT_OS
//code...
//code...
//code...
#endif
```

5 extern声明

放在函数/变量前，表示此函数/变量在其他文件定义，以便本文件引用

```
extern  uint16_t    g_usart_rx_sta  ;   //让编译器去其他文件寻找这个变量
extern  void delay_us(uint32_t nus) ;
```

6 类名别名typedef

为现有数据类型创建一个新的名字，用来简化变量的定义

```
//typrdef   现有类型                  新名字
typedef     signed       char       int8_t  ;
typedef     signed short int        int16_t ;
typedef     signed       int        int32_t ;
```

类型别名应用

```
Struct GPIO_Typedef
{
    __IO uint32_t CRL ;
    __IO uint32_t CRH ;
    //...
};
Struct GPIO_Typedef gpiox
//采用typedef定义结构体类型
typedef struct
{
    __IO uint32_t CRL ;
    __IO uint32_t CRH ;
    //...
}GPIO_Typedef;
GPIO_Typedef gpiox
```

7 结构体

由若干基本数据类型集合组成的一种自定义数据类型

```
/*
struct 结构体名
{
    成员列表;
}变量名列表(可选);
*/

//举个栗子
struct student
{
    char    *name ; //姓名
    int     num   ; //学号
    int     age   ; //年龄
    char    group ; //所在学习小组
    float   score ; //成绩
}stu1,stu2;
```

结构体的定义和使用

```
//第一个栗子
```

```
struct student
{
    char    *name ; //姓名
    int     num   ; //学号
    int     age   ; //年龄
    char    group ; //所在学习小组
    float   score ; //成绩
};
struct student stu3,stu4 ;
stu3.name   =  "张三" ;
stu3.num    =  1      ;
stu3.age    =  18     ;
stu3.group  =  'A'    ;
stu3.score  =  80.9   ;

//第二个栗子：ST源码中使用类型别名
typedef struct
{
    uint32_t    Pin   ; //引脚号
    uint32_t    Mode  ; //工作模式
    uint32_t    Pull  ; //上下拉
    uint32_t    Speed ; //IO速度
}GPIO_IntiTypeDef;
```

8 指针

指针是内存的地址

指针变量是保存了指针的变量

```
/*
类型名 *指针变量名
*/
//举个栗子
char *p_str = "This is a test!" ;
//  *p_str：取p_str变量的值
//  &p_str：取p_str变量的地址
```

举个栗子

地址	变量	数组
0x0001		
0x0002	1	buf[0]
0x0003	3	
0x0004	5	
0x0005	7	
0x0006	9	buf[4]

```
uint8_t buf[5] = {1,3,5,7,9} ;
uint8_t *p_buf = buf ;       //地址为0x0002
//  *p_buf   = ?    1
//  p_buf[0] = ?    1
//  p_buf[1] = ?    3
p_buf++ ;                    //地址为0x0003
//  *p_buf   = ?    3
//  p_buf[0] = ?    3        //指针地址变为0x0003，所以buf[0]的值不再是1，而是3
```

指针使用的两大最常见问题： 1未分配内存 2越界使用

```
//第一个栗子：错误用法-未分配内存
char *p_buf ;
p_buf[0] = 100 ;
p_buf[1] = 120 ;
p_buf[2] = 150 ;
//第二个栗子：错误用法-越界使用
uint8_t buf[5] = {1,3,5,7,9} ;
uint8_t *p_buf = buf ;
p_buf[5] = 200 ; //5超出了0-4的范围
```