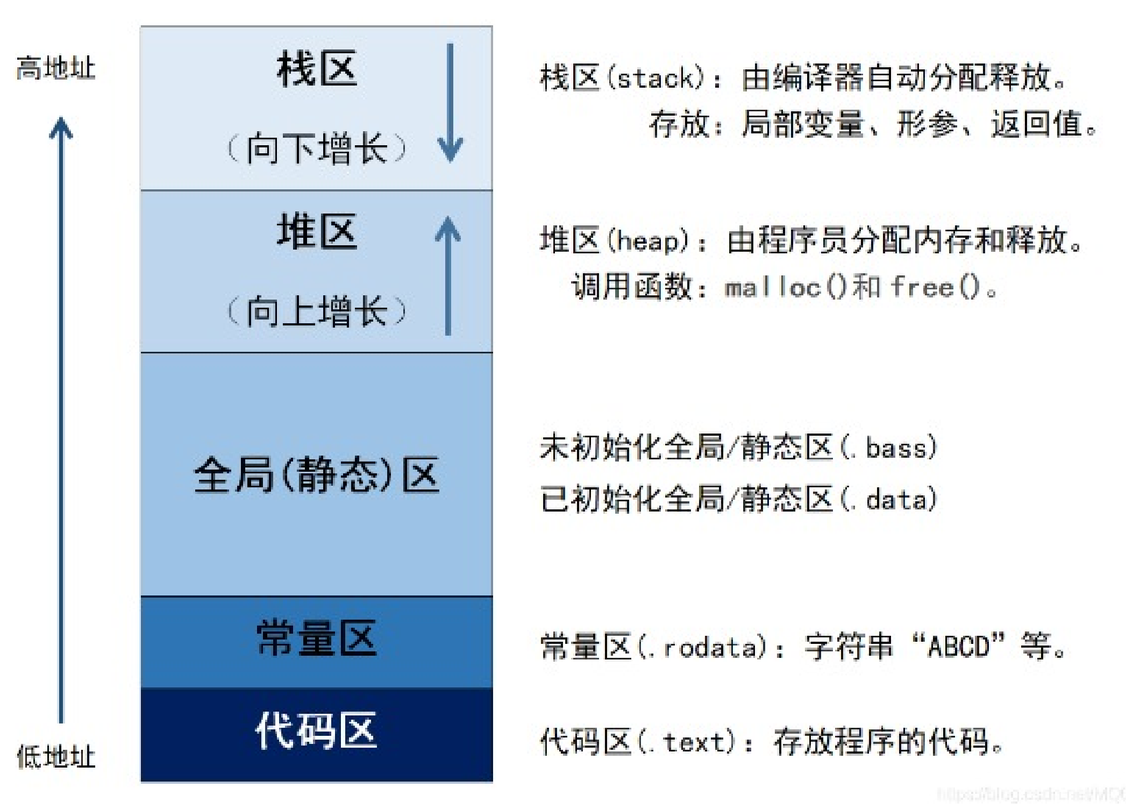
C语言：内存分配---栈区、堆区、全局区、常量区和代码区

原创 根号五 已于 2022-03-14 09:56:08 修改 9819 收藏 220 版权
分类专栏： C语言 文章标签： C语言 STM32 内存管理 嵌入式

一、C语言内存分区

C语言内存分区示意图如下：



栈区介绍

- **栈区**由编译器**自动分配释放**，由操作系统自动管理，无须手动管理。
- **栈区**上的内容只在函数范围内存在，当函数运行结束，这些内容也会**自动被销毁**。
- **栈区**按内存地址**由高到低**方向生长，其最大大小由编译时确定，速度快，但自由性差，最大空间不大。
- **栈区是先进后出**原则，即先进去的被堵在屋里的最里面，后进去的在门口，释放的时候门口的先出去。

存放内容

- 临时创建的**局部变量**和**const定义的局部变量**存放在**栈区**。
- 函数调用和返回时，其入口**参数**和**返回值**存放在**栈区**。

2. 堆区

堆区介绍

- **堆区**由程序员分配内存和释放。
- **堆区**按内存地址**由低到高**方向生长，其大小由系统内存/虚拟内存上限决定，速度较慢，但自由性大，可用空间大。

调用函数

- 用**malloc**等函数实现动态分布内存。

```
void *malloc(size_t);
```

  参数size_t是分配的字节大小。
  返回值是一个void*型的指针，该指针指向分配空间的首地址。
  （void *型指针可以任意转换为其他类型的指针）

- 用**free**函数进行内存释放，否则会造成内存泄漏。

```
void free(void * /*ptr*/);
```

  参数是开辟的内存的首地址。

3. 全局(静态)区

全局(静态)区介绍

- 通常是用于那些在编译期间就能确定存储大小的变量的存储区，但它用于的是在整个程序运行期间都可见的全局变量和静态变量。
- 全局区有 **.bss段** 和 **.data段**组成，可读可写。

.bss段

- 未初始化的**全局变量**和未初始化的**静态变量**存放在**.bss段**。
- 初始化为0的**全局变量**和初始化为0的**静态变量**存放在**.bss段**。
- .bss段不占用可执行文件空间，其内容由操作系统初始化。

.data段

- 已初始化的**全局变量**存放在.data段。
- 已初始化的**静态变量**存放在.data段。
- .data段占用可执行文件空间，其内容有程序初始化。

4. 常量区

- **字符串**、**数字**等常量存放在常量区。
- **const修饰的全局变量**存放在常量区。
- 程序运行期间，**常量区**的内容不可以被修改。

5. 代码区

- 程序执行代码存放在代码区，其值不能修改（若修改则会出现错误）。
- 字符串常量和define定义的常量也有可能存放在代码区。

二、STM32存储器分配

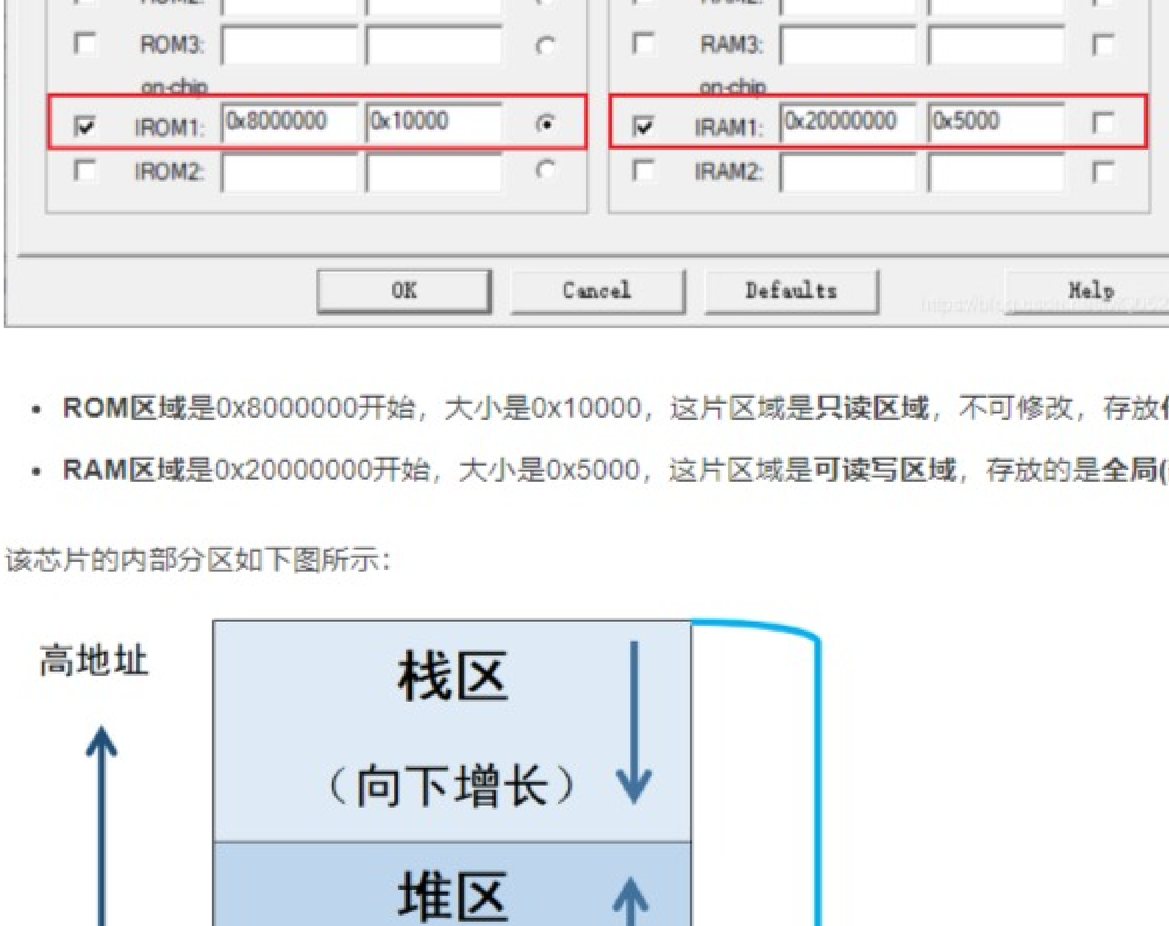
1. 随机存储器—RAM

- RAM是与CPU直接交换数据的内部存储器，也叫**主存**（内存）。
- 它可以**随时读写**，而且速度很快，通常作为操作系统或其他正在运行中的程序的临时数据存储媒介。
- 当电源关闭时RAM不能保留数据（**掉电数据消失**哦）如果需要保存数据，就必须把它们写入一个长期的存储设备中（例如硬盘）。

2. 只读存储器—ROM

- ROM所存数据，一般是装入整机前事先写好的，整机工作过程中只能读出，而不像随机存储器那样能快速地、方便地加以改写。
- ROM所存数据稳定，断电后所存数据也不会改变。

本文使用是STM32F103芯片，keil V5环境下默认的内存配置见下图：



STMicroelectronics STM32F103C8
Xtal (MHz): 8.0
Operating system: None
System Viewer File: STM32F103xx.svd
☐ Use Custom File
Code Generation
ARM Compiler: Use default compiler version 5
☐ Use Cross-Module Optimization
☑ Use MicroLIB ☐ Big Endian

Read/Only Memory Areas

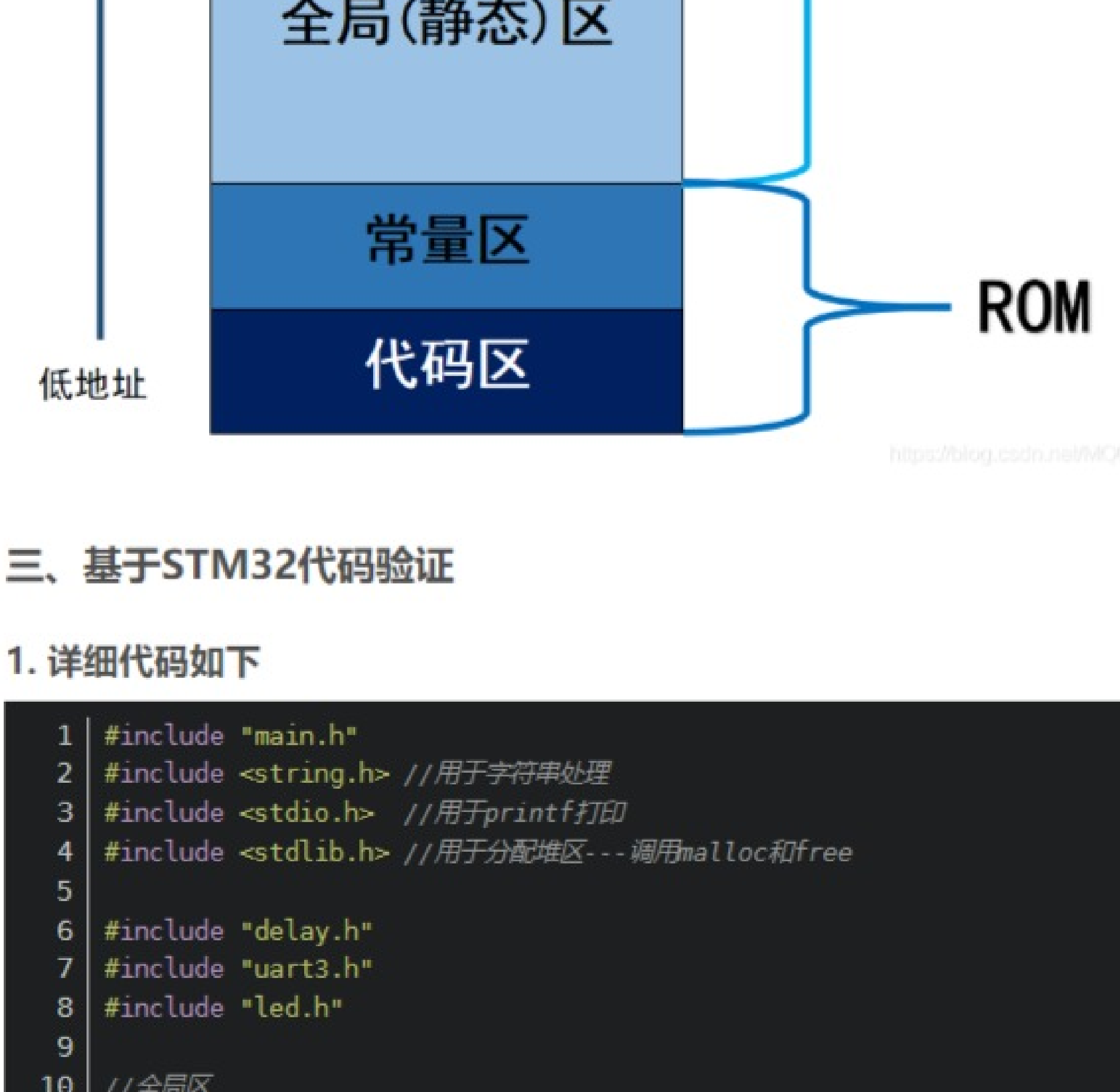
default	off-chip	Start	Size	Startup
☐	ROM1:			○
☐	ROM2:			○
☐	ROM3:			○
	on-chip			
☑	IROM1:	0x8000000	0x10000	●
☐	IROM2:			○

Read/Write Memory Areas

default	off-chip	Start	Size	NoInit
☐	RAM1:			☐
☐	RAM2:			☐
☐	RAM3:			☐
	on-chip			
☑	IRAM1:	0x20000000	0x5000	☐
☐	IRAM2:			☐

- **ROM区域**是0x8000000开始，大小是0x10000，这片区域是**只读区域**，不可修改，存放**代码区**和**常量区**。
- **RAM区域**是0x20000000开始，大小是0x5000，这片区域是**可读写区域**，存放的是**全局(静态)区**、**堆区**和**栈区**。

该芯片的内部分区如下图所示：



三、基于STM32代码验证

1. 详细代码如下

```
#include "main.h"
#include <string.h> //用于字符串处理
#include <stdio.h>  //用于printf打印
#include <stdlib.h> //用于分配堆区---调用malloc和free

#include "delay.h"
#include "uart3.h"
#include "led.h"

//全局区
        int     q1;     //未初始化全局变量
static  int     q2;     //未初始化静态变量
const   int     q3;     //未初始化只读变量

        int   m1=1;     //已初始化全局变量
static  int   m2=2;     //已初始化静态变量

//常量区
const   int   m3=3;     //已初始化只读变量

int main(void)
{
    SystemCoreClockUpdate(); //设置系统时钟为72M
    LED_GPIO_Config();
    Uart3_init();

    while(1)
    {
        //栈区
        int      mq1;           //未初始化局部变量
        int     *mq2;           //未初始化局部指针变量

        int     mq3=3;      //已初始化局部变量
        char    qq[10] = "hello"; //已初始化局部数组

        const   int   mq4;          //未初始化局部只读变量
        const   int   mq5=3;        //已初始化局部只读变量

        //堆区
        int *p1 = malloc(4);    //已初始化局部指针变量p1
        int *p2 = malloc(4);    //已初始化局部指针变量p2

        //全局区
        static  int     mp1;            //未初始化局部静态变量
        static  int   mp2=2;        //已初始化局部静态变量

        //常量区
        char *vv = "I LOVE YOU";//已初始化局部指针变量
        char *mq = "5201314";

        printf("\n栈区-变量地址\n");
        printf("未初始化局部变量          :0x%p\r\n",&mq1);
        printf("未初始化局部指针变量      :0x%p\r\n",&mq2);
        printf("已初始化局部变量          :0x%p\r\n",&mq3);
        printf("已初始化局部数组          :0x%p\r\n", qq );

        printf("未初始化局部只读变量      :0x%p\r\n",&mq4);
        printf("已初始化局部只读变量      :0x%p\r\n",&mq5);

        printf("\n堆区-动态申请地址\r\n");
        printf("已初始化局部int型指针变量p1   :0x%p\r\n", p1);
        printf("已初始化局部int型指针变量p2   :0x%p\r\n", p2);

        printf("\n全局区-变量地址\n");
        printf("未初始化全局变量      :0x%p\r\n",&q1);
        printf("未初始化静态变量      :0x%p\r\n",&q2);
        printf("未初始化只读变量      :0x%p\r\n",&q3);

        printf("已初始化全局变量      :0x%p\r\n",&m1);
        printf("已初始化静态变量      :0x%p\r\n",&m2);

        printf("未初始化局部静态变量   :0x%p\r\n",&mp1);
        printf("已初始化局部静态变量   :0x%p\r\n",&mp2);

        printf("\n常量区地址\n");
        printf("已初始化只读变量        :0x%p\r\n",&m3);
        printf("已初始化局部指针变量    :0x%p\r\n",vv );
        printf("已初始化局部指针变量    :0x%p\r\n",mq );

        printf("\n代码区地址\n");
        printf("程序代码区main函数入口地址    :0x%p\n", main);

        led485_flicker();
        delay_ms(1000);

        free(p1);
        free(p2);
    }
}
```

2. 运行结果如下

```
栈区-变量地址
未初始化局部变量          :0x20000654
未初始化局部指针变量      :0x20000650
已初始化局部变量          :0x2000064c
已初始化局部数组          :0x20000640
未初始化局部只读变量      :0x2000063c
已初始化局部只读变量      :0x20000638

堆区-动态申请地址
已初始化局部指针变量p1    :0x20000060
已初始化局部指针变量p2    :0x20000068

全局区-变量地址
未初始化全局变量          :0x20000014
未初始化静态变量          :0x20000018
未初始化只读变量          :0x2000001c
已初始化全局变量          :0x20000020
已初始化静态变量          :0x20000024
未初始化局部静态变量      :0x20000028
已初始化局部静态变量      :0x2000002c

常量区地址
已初始化全局只读变量      :0x080011a4
已初始化局部指针变量      :0x08000e78
已初始化局部指针变量      :0x08000e84

代码区地址
程序代码区main函数入口地址    :0x08000d6d
```