# 1 主程序文件的编写规范

主文件包括总头文件includes.S和main.S。includes.S是main.S使用的头文件，main.S文件是工程任务的核心文件，main函数即在该文件中实现。在main函数中包含了一个永久循环，对具体事务过程的操作几乎都是添加在该主循环中。

## 1.1 include.s文件编写规范

include.S文件作为总头文件，内含main.S文件中需要的元素。Include.S文件中的内容包括：程序编码框架、包含文件的处理、数据定义等。

### 1.1.1 编码框架

在总头文件的开始应使用“#ifndef… #define … #endif”的编码结构，防止对该文件的重复包含。

### 1.1.2 包含文件

通过.include指令，将需要调用的构件的头文件包含在本文件中。在包含时，一般通过.include指令包含汇编头文件，通过#include指令包含C头文件。

### 1.1.3 数据定义及宏函数

在参数的定义区域实现数据的定义，一般需要使用的数据类型有常量、变量、未初始化量，对它们的定义分别在.section .rodata，.section .data，.section .bss段中实现，如果有特殊的数据要求，可以另行开辟其它类型的段。

## 1.2 main.S文件编写规范

main.S文件中应包含以下部分内容：包含include.S文件、定义主函数及在本文件中编写的子程序、main函数代码、子程序代码、结束符。

### 1.2.1 包含include.S文件

main.S通过.include命令包含include.S文件，即可调用所有不在main.S文件中定义的函数及变量。

### 1.2.2 定义函数

函数定义段用于声明主函数main，以及在main.S文件中编写的子程序，由.section .text语句开始，表示以下的代码放置在程序段中，将用于运行；对于main函数，应接着使用.global指令将其定义为全局变量，并使用.type指令将其定义为函数类型，使其在芯片初始化后能被立即调用；如果有其他子程序，可以按照同样方式定义。

### 1.2.3 main函数代码段

main函数代码段以标号main:开始，一般应包括初始化和主循环两部分。

### 1.2.4子程序代码段

子程序代码一般跟在main函数代码后，以标号*symbol*:开始。建议每个子程序不超过100行，若有更多的代码应单独存盘，单独测试。子程序段可以没有。

### 1.2.5 结束符

主文件的最后应加上.end伪指令，表示程序代码到此结束。

# 2 构件文件的编写规范

构件文件一般包括高层软件构件文件及底层驱动构件文件，每个构件由头文件（.inc）和源文件（.S）两个独立文件组成，放在以构件名命名的文件夹中。在使用时，调用者可以只看头文件中的信息而应用构件。

## 2.1 头文件编写规范

头文件描述了构件的接口，用户通过头文件获取构件服务。头文件中的内容包括：程序编码框架“#ifndef… #define … #endif”、包含文件的处理、数据定义及宏、设计服务接口等。

### 2.1.1 编码框架

编写每个构件的头文件时，应使用编码结构，防止对头文件的重复包含。

### 2.1.2 包含文件

包含文件命令为.include ，包含文件的语句统一安排在构件的头文件中，而在相应构件的源文件中仅包含本构件的头文件。将包含文件的语句统一置于构件的头文件中，使文件间的引用关系能够更加清晰地呈现。

### 2.1.3 数据定义及宏函数

汇编中使用.set或.equ伪指令实现数据定义，而用.macro伪指令抽象出一个函数，这两者都类似于C中的#define指令。所以，如果构件中需要使用到一些常量，可以使用数据定义为常量值提供有意义的别名；如果需要实现构件对外部请求服务的接口映射，可以使用宏函数，从而为程序移植提供灵活性。

### 2.1.4设计服务接口

构件通过服务函数接口提供完备而易用的服务，对外提供的服务接口在头文件应通过.global声明。同时应按照编码风格基本规范为函数接口添加函数头注释。通过对设备驱动功能进行研究可以发现，构件一般应提供的服务有初始化、向设备发送信息、从设备读取信息和调整设备工作属性等。

## 2.2 源文件的编写规范

为确保构件工作的独立性，实现构件高内聚、低耦合的设计要求，构件的实现内容应封装在源文件内部。源文件中的内容有：包含自身头文件、构件函数。

构件的源文件（.S）中，只允许一处使用.include包含自身头文件。需要包含的内容需在自身构件的头文件中包含，以便有统一、清晰的程序结构。

（1）汇编编程不够直观、阅读、理解困难，要给出完备简明的分段功能及重要语句的注释。

（2）程序中需要使用多个跳转语句时，不要上下交叉跳转，应尽量向下跳转，确保编程结构清晰。

（3）若要使用内部函数，则内部函数中要做好现场保护与恢复工作，调用程序无需考虑内部函数的现场保护与恢复问题，各司其职，职责分明，也便于出错调试。

## 2.3 构件函数调用规则

在调用构件函数的过程中，涉及到参数的传递及寄存器的使用。如果调用函数时随意使用不同的参数传递方式，或者在函数过程中随意使用寄存器，那么该构件函数就不具有通用性。在这里，依照ARM公司的aapcs标准，设定构件函数调用规则如下：

（1）调用函数需要传递参数时，通过寄存器R0～R3传递，如果参数超过4个，第5个开始的参数从数据栈传递，被调用的函数在返回前无须恢复寄存器R0～R3的内容。

（2）被调用函数在ARM状态下使用寄存器R4～R11来保存局部变量；Thumb状态下只能使用R4～R7来保存局部变量。被调用的函数应首先通过push指令保存这些寄存器，在返回前通过pop指令恢复这些寄存器的内容。

（3）寄存器SP用于数据栈指针，在进入子程序时的值和退出子程序时的值必须相等，在被调用函数中一般不要修改SP。

（4）寄存器LR用于保存子程序的返回地址，被调用的函数应首先通过push指令保存LR，然后LR可以在被调用函数中用作其它用途。

（5）寄存器PC是程序计数器，被调用函数最后通过pop{pc}指令返回。

（6）被调用函数如果有返回值，当结果为一个32位的整数时，可以通过寄存器R0返回；当结果为一个64位整数或两个32位整数时，可以通过R0和R1返回，依此类推。