第5章 基于构件的汇编程序设计方法

5.1 构件及其设计方法

5.2 程序流程控制

5.3 汇编程序设计实例

5.4 实验二：基于构件方法的汇编程序设计

构件是软件开发中极其重要的概念，只要我们把软件作为一个工程来做，构件就是基础。这一章在分析构件化设计重要性和必要性的基础上，给出软件构件基本概念及构件设计的基本原则；给出程序顺序结构、分支结构、循环结构等流程控制基本方法及实例；给出基于构件方法的汇编程序设计实验等内容。

5.1 构件及其设计方法

构件（Component）是核心和基础，复用是必需的手段。随着微控制器及应用处理器内部Flash存储器、RAM容量的增大、外部模块内置化程度的提高，已经嵌入式系统的设计复杂性、设计规模及开发手段的变化，特别是嵌入式人工智能的需求，软硬件设计可重用性与可移植性得到更加重视，构件是重用与移植的基础与保障。

5.1.1 软件构件基本概念

软件构件（Software component）广义上的理解是：可复用的软件成分。

多种软件构件定义

面向构件程序设计工作组：软件构件是一种组装单元，它具有规范的接口规约和显式的语境依赖。软件构件可以被独立地部署并由第三方任意地组装。

卡内基梅隆大学软件工程研究所:构件是一个不透明的功能实体，能够被第三方组织，且符合一个构件模型。

国际上第一部软件构件专著的作者Szyperski:可单独生产、获取、部署的二进制单元，它们之间可以相互作用构成一个功能系统。

一般可以将软件构件定义为：在语义完整、语法正确情况下，具有可复用价值的单位软件是软件复用过程中可以明确辨别的成分；从程序角度上可以将构件看作是有一定功能、能够独立工作或协同其他构件共同完成的程序体。

引申阅读：[NATO 1991-1]软件重用-nato_standards_vol_1-P96.pdf
[NATO 1991-2]软件重用-nato_standards_vol_2-P65.pdf
[NATO 1991-3]软件重用-nato_standards_vol_3-P52.pdf
在电子资源“01-Informaiton”文件夹

5.1.2 构件设计基本原则

1. 构件设计的基本思想

底层构件是与硬件直接打交道的软件，它被组织成具有一定独立性的功能模块，由头文件和源程序文件两部分组成。

头文件：必要的引用文件、描述构件功能特性的宏定义语句以及声明对外接口函数。

源程序：构件的头文件、内部函数的声明、对外接口函数的实现。

在设计底层构件时，最关键的工作是要对构件的共性和个性进行分析，设计出合理的、必要的对外接口函数及其形参。尽量做到：当一个底层构件应用到不同系统中时，仅需修改构件的头文件，对于构件的源程序文件则不必修改或改动很小。

2. 构件设计的基本原则

良好的底层驱动构件具备如下特性：

（1）封装性。在内部封装实现细节，采用独立的内部结构以减少对外部环境的依赖。调用者只通过构件接口获得相应功能，内部实现的调整将不会影响构件调用者的使用。

（2）描述性。构件必须提供规范的函数名称、清晰的接口信息、参数含义与范围、必要的注意事项等描述，为调用者提供统一、规范的使用信息。

（3）可移植性。底层构件的可移植性是指同样功能的构件，如何做到不改动或少改动，而方便地移植到同系列及不同系列芯片内，减少重复劳动。

（4）可复用性。在满足一定使用要求时，构件不经过任何修改就可以直接使用。特别是使用同一芯片开发不同项目，底层驱动构件应该做到复用。可复用性使得高层调用者对构件的使用不因底层实现的变化而有所改变，可复用性提高了嵌入式软件的开发效率、可靠性与可维护性。不同芯片的底层驱动构件复用需在可移植性基础上进行。

为了使构件设计满足封装性、描述性、可移植性、可复用性的基本要求，嵌入式底层驱动构件的开发，应遵循层次化、易用性、鲁棒性及对内存的可靠使用原则。

（1）层次化原则。要求清晰地组织构件之间的关联关系。针对应用场景和服务对象，分层组织构件；在构件的层次模型中，上层构件可以调用下层构件提供的服务，同一层次的构件不存在相互依赖关系，不能相互调用。

（2）易用性原则。在于让调用者能够快速理解的构件提供服务的功能并进行使用。函数名简洁且达意；接口参数清晰，范围明确；使用说明语言精炼规范，避免二义性；在函数的实现方面，避免编写代码量过多。

（3）鲁棒性原则。为调用者提供安全的服务，避免在程序运行过程中出现异常状况。在函数实现的内部要有对输入参数的检测，对超出合法范围的输入参数进行必要的处理；使用分支判断时，确保对分支条件判断的完整性，对缺省分支进行处理。

（4）内存可靠使用原则。保证系统安全、稳定运行的一个重要的考虑因素。优先使用静态分配内存；不使用全局变量。

5.1.3 三类构件

1. 基础构件

基础构件是根据MCU内部功能模块的基本知识要素，针对MCU引脚功能或MCU内部功能，利用MCU内部寄存器，所制作的直接干预硬件的构件。

2. 应用构件

应用构件是使用基础构件并面向对象编程的构件。

3. 软件构件

软件构件是一个面向对象的具有规范接口和确定的上下文依赖的组装单元，它能够被独立使用或被其他构件调用。

5.1.4 基于构件的软件设计步骤

1. 构件测试

要进行具有相对完整功能的应用程序设计，必然要使用到构件。在构件使用之前，必须编写构件测试程序对构件进行测试。

2. 应用程序设计

汇编语言的应用程序设计可分为主程序及中断服务程序两个部分，一般过程有：分析问题、建立数学模型、确定算法、绘制程序流程图、内存空间分配、编写程序及程序整体测试。

（1）分析问题。分析问题就是将解决问题所需条件、原始数据、输入/输出信息、运行速度、运算精度和结果形式等搞清楚。

（2）建立数学模型。就是把问题数学化、公式化。

（3）确定算法。建立数学模型后，许多情况下还不能直接进行程序设计，需要确定符合微控制器运算的算法。

（4）绘制程序流程图。程序流程图是用箭头线段、框图及菱形图等绘制的一种图，它直接描述程序内容。

（5）内存空间分配。汇编语言的重要特点之一是能够直接用汇编指令或伪指令为数据或代码程序分配内存空间，当程序中没有指定分配存储空间时，系统会按约定方式分配存储空间。

（6）编写程序。汇编语言编程应按指令和伪指令的语法规则进行。

（7）程序整体测试。程序整体测试是程序设计的最后一步，也是非常重要的一步。通过整体测试，可以纠正程序的语法错误和语义错误，最后实现程序正确的运行。

5.2程序流程控制

程序流程控制主要有顺序结构、分支结构、循环结构。

5.2.1 顺序结构

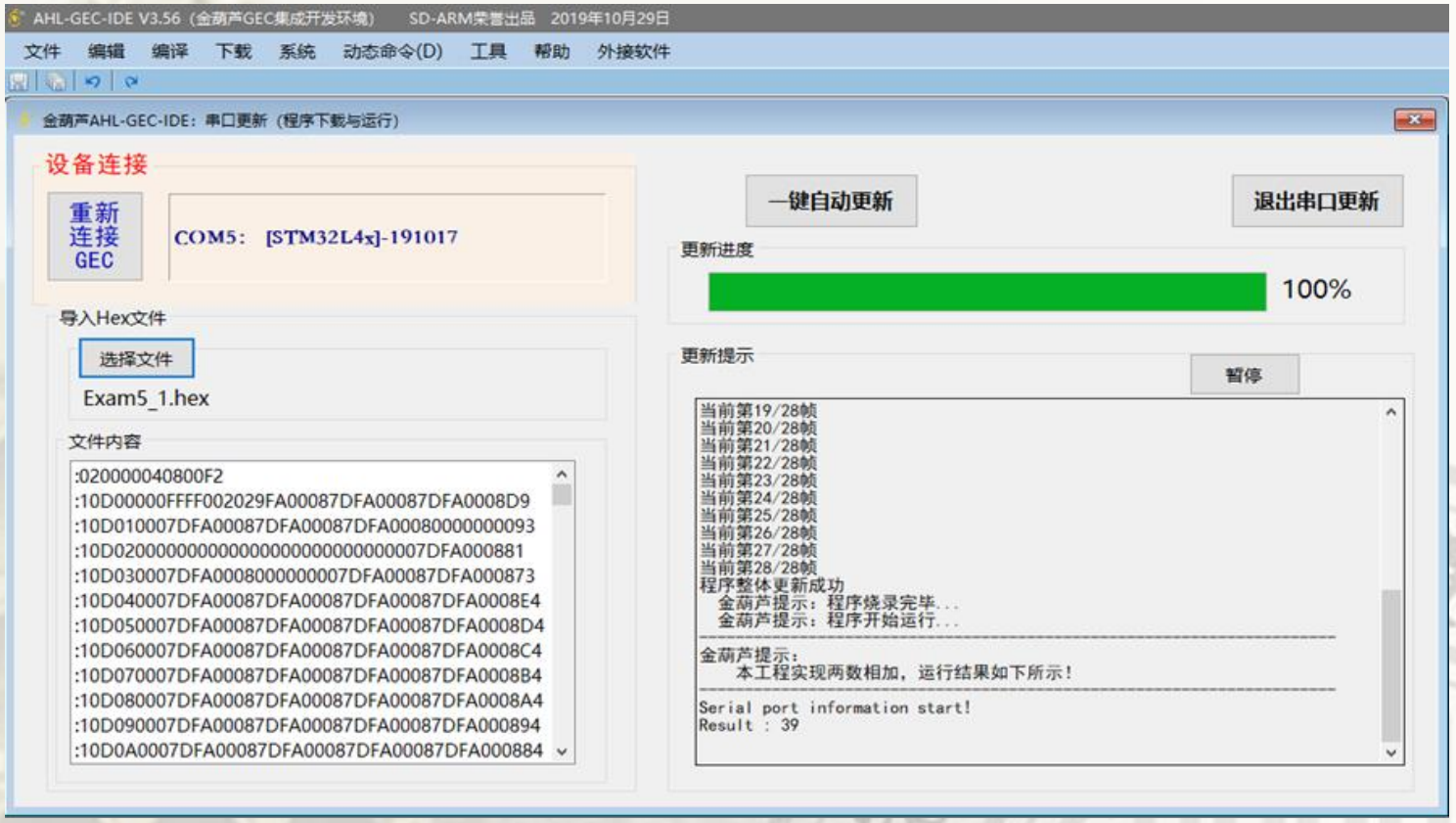
顺序结构程序的执行方式是“从头到尾”，逐条执行指令语句，直到程序结束，这是程序的最基本形式。

【例5-1】编程实现将两个寄存器相加得到的结果通过串口输出，假设两个寄存器存放的数据分别是15和24，参考程序见“Exam5_1”工程。（直接演示）

分析：假定r0寄存器存放数据15，r1寄存器存放数据24，可通过add指令实现两数相加，再调用串口构件函数输出结果即可，这里在打印结果前先输出字符串“Serial port information start!”和“Result：”，方便直观的看到结果现象，

顺序执行程序流程图：

顺序执行结果：



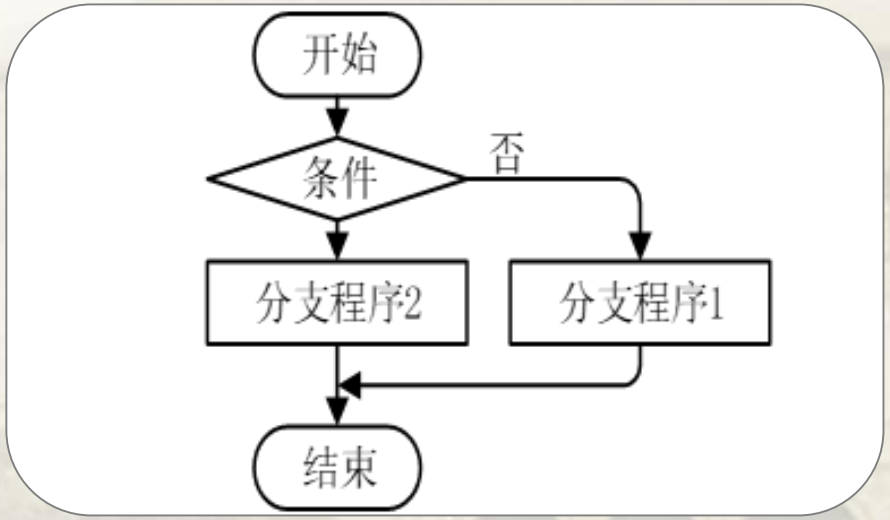
5.2.1 顺序结构——代码

```
        mov r0,#33
        mov r1,#24
        add r1,r0,r1        //将结果直接存入到r1,作为printf的参数
//      mov r1,r0
        ldr r0,=data_format
        bl printf
        mov r2,#33
        mov r3,#25
        add r1,  r2,r3        //
        ldr r0,=data_format     //此句不能省略，因为前面的printf调用会修改r0
        bl printf
```

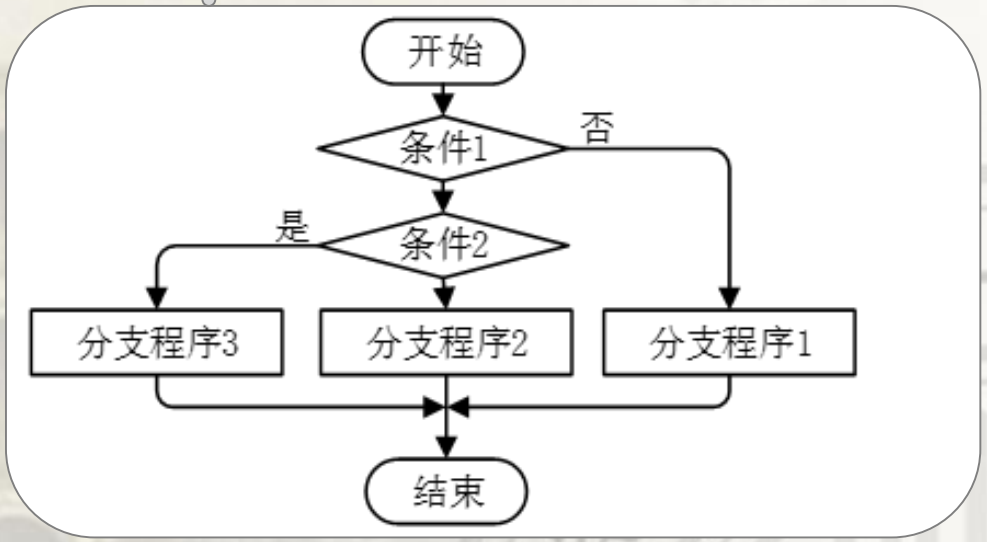
5.2.2 分支结构

分支结构程序是利用条件转移指令，使程序执行到某一指令后，根据条件是否满足，来改变程序执行的次序，这类程序使计算机有了判断作用。分支结构分为单分支结构和多分支结构。

单分支程序流程图

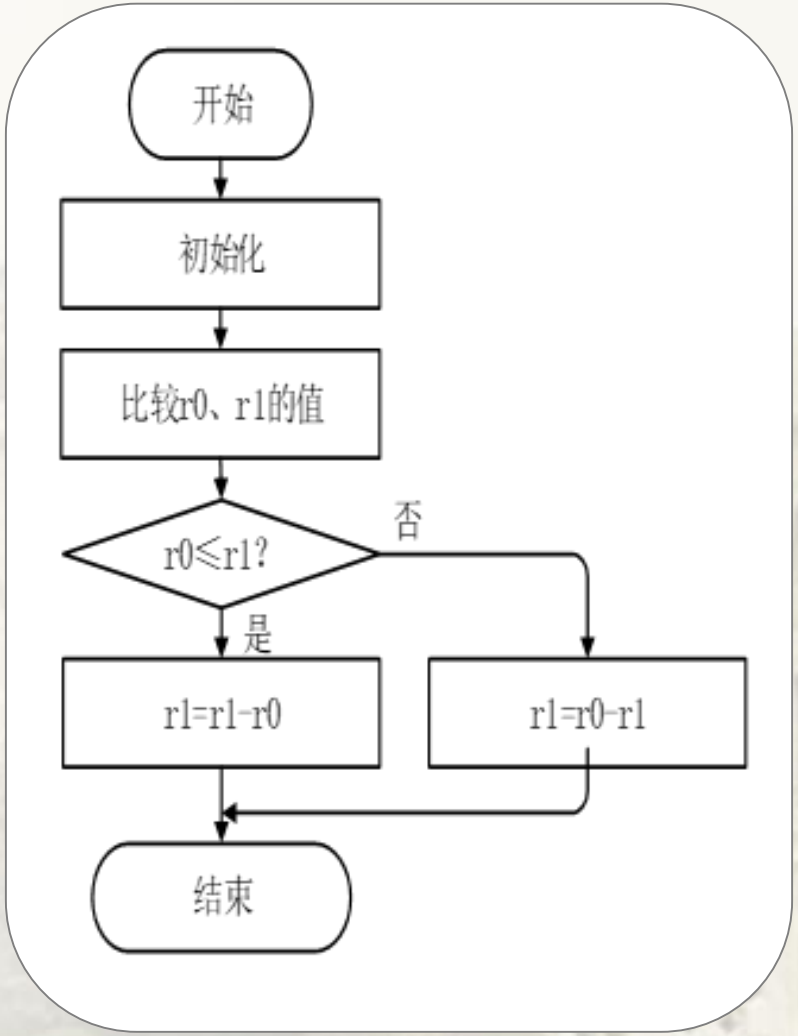


双分支程序流程图

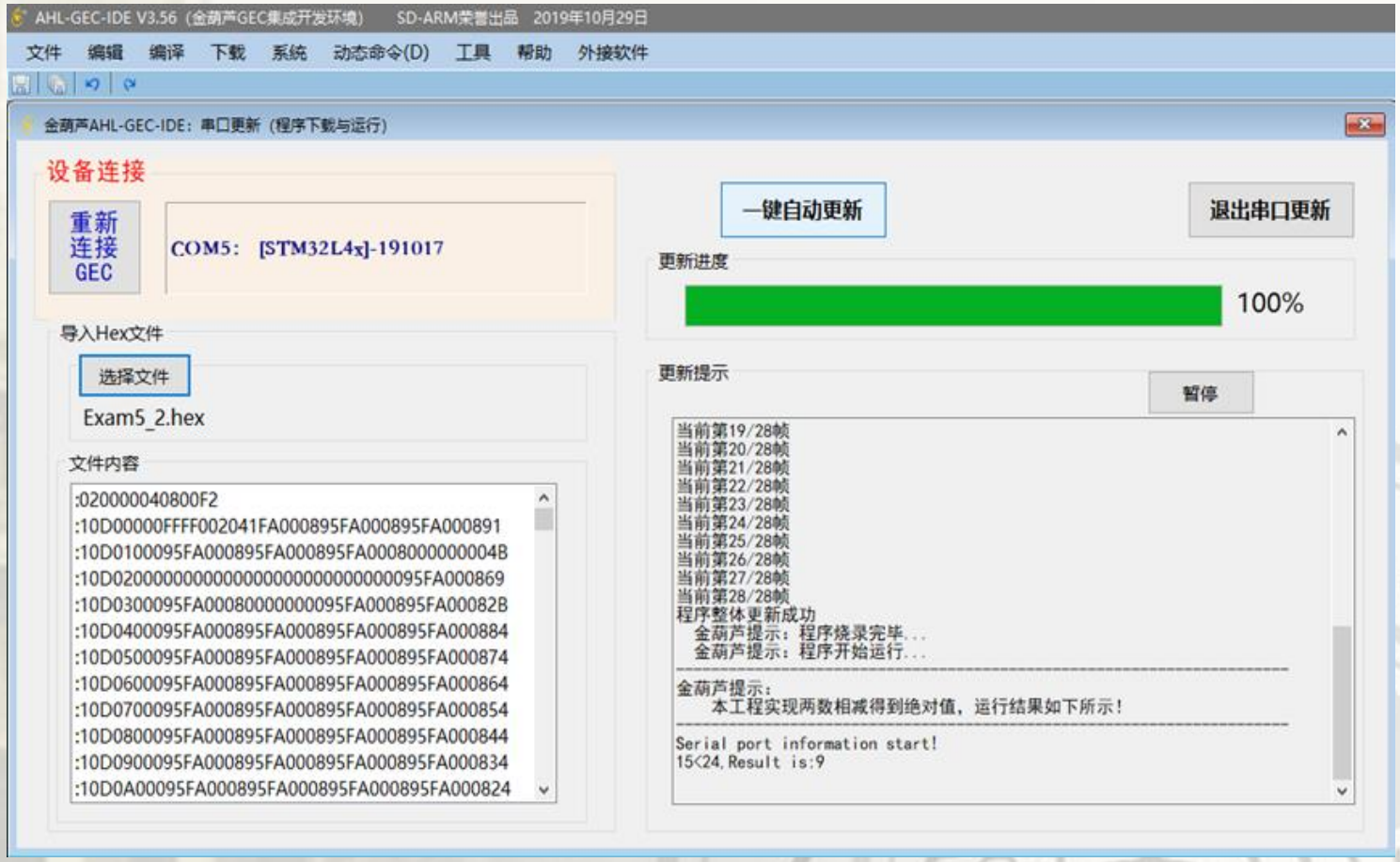


【例5-2】编程实现寄存器A和寄存器B两个无符号数之差的绝对值，将结果用串口输出，假设两寄存器存放的数据分别是15和24，参考程序见“Exam5_2”工程。

求绝对值程序流程图：



求绝对值执行结果：



5.2.2 分支结构——代码

```
//(2.4.2)计算结果并用串口输出
        mov r1,#24                          //r2=15
        mov r2,#39                          //r3=24
        //mov r1,r2
 //(2.4.3)比较r2、r3的值大小
        cmp r1,r2
        bls low         //若r2≤r3，跳转到low执行
        //bcs high      //若r2>r3，跳转到high执行
```

5.2.2 分支结构——代码

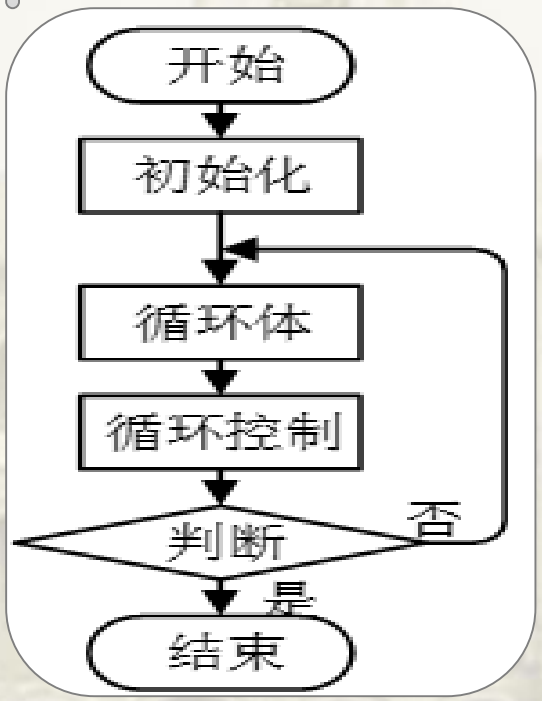
```
high:
        sub r3,r1,r2
        ldr r0, =string_control_1
        bl  printf
        b main_loop        //继续循环

low:
        sub r3,r2,r1
        ldr r0, =string_control_2
        bl  printf
```

```
sub r4,r2,r3
mov r2,r3
mov r3,r4
```

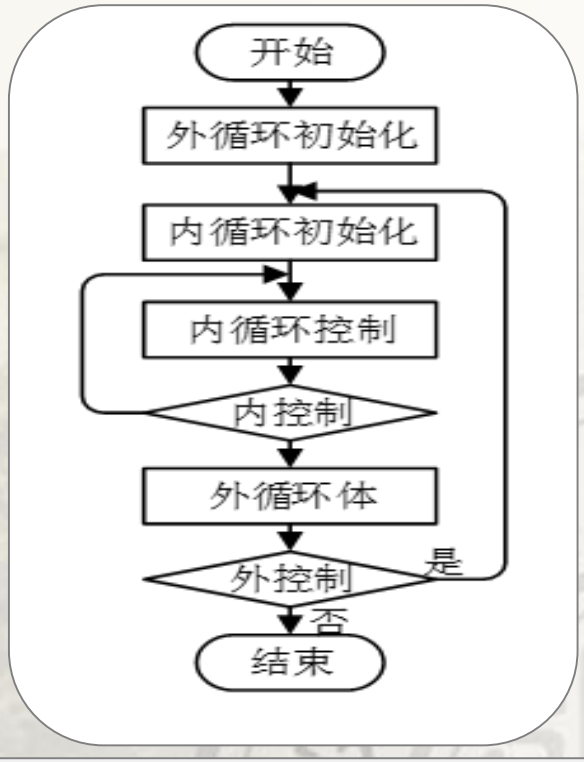
5.2.3 循环结构

循环结构程序是强制CPU重复执行某一指令系列（程序段）的一种程序结构形式，凡是要重复执行的程序段都可以按循环结构设计。循环程序一般由四部分组成：初始化、循环体、循环控制和循环结束处理。

循环程序结构流程图

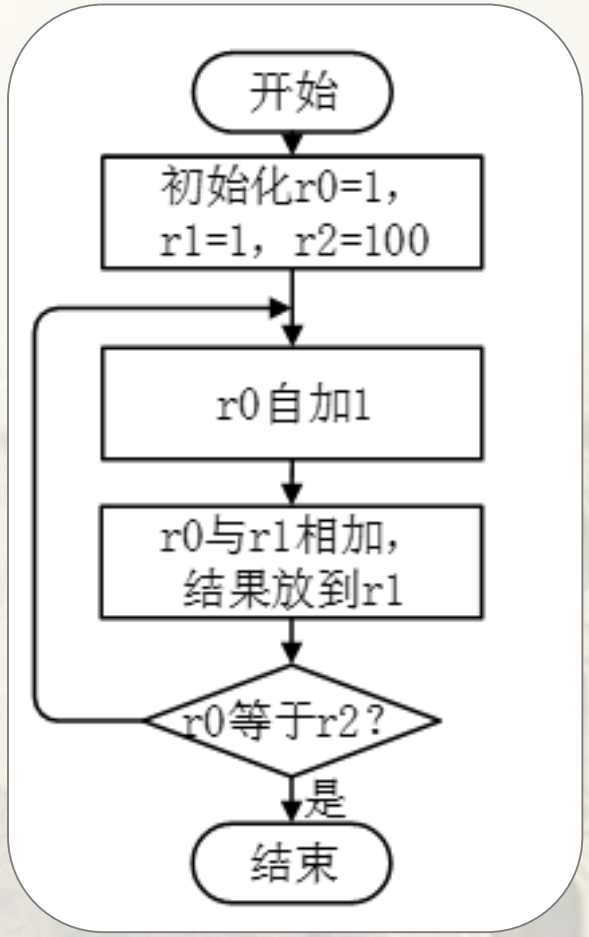


循环程序结构流程图

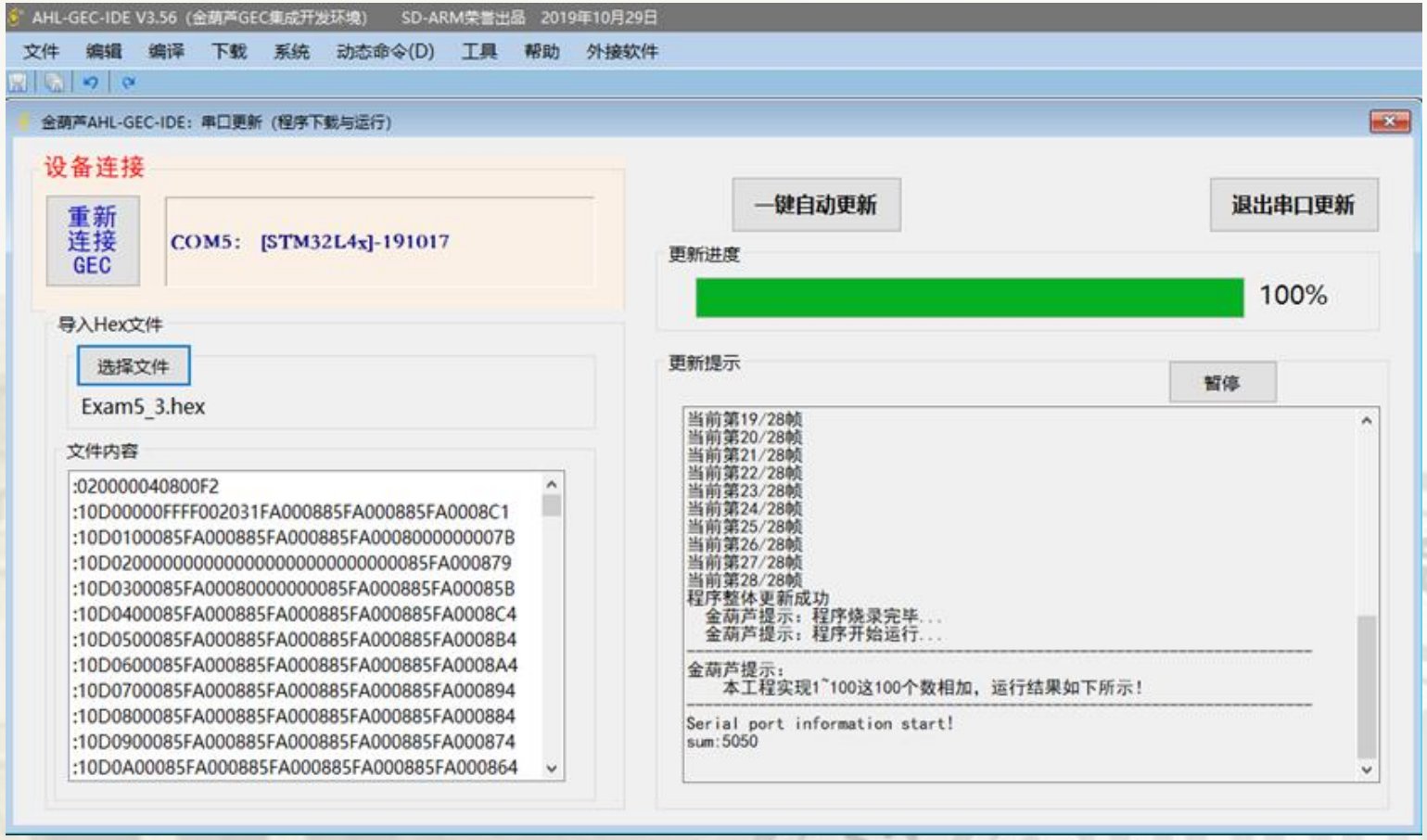


【例5-3】编程实现sum=1+2+3+4+…+100，并将sum的值通过串口输出，参考程序见“Exam5_3”工程。

循环控制流程图



循环控制执行结果



【练习5-2】打开“Exam5_3”工程，修改程序实现求5！，并通过printf输出结果。

5.3 汇编程序设计实例

本节以数制转换、冒泡排序为例，并通过调用printf构件、LCD构件进行过程及结果显示。

5.3.1 数制转换程序设计

【例5-4】编写程序将十进制数转换成二进制、八进制、十六进制，结果通过串口输出。假设需要转换的十进制数是95，参考程序见“Exam5_4”工程。

分析：十进制转换成二进制可通过每次移动一位数字再与“1”进行逻辑“与”运算，依次得到存储在寄存器中的最后一位数据，最后通过串口输出每一位即可。十进制转化为八进制同上。十进制转化成十六进制可通过每次移动四位数字再与“1”进行逻辑“与”运算，依次得到存储在寄存器中的后四位数据，若数据大于10，则参照ASCII表可知字符“10”和“A”相差55，在“10”的ASCII值的基础上加上55输出即可，

5.3.1 数制转换程序设计

1、汇编语言如何定义数组

```
.global array,count                    //声明所定义的数组和数组长度是全局变量
.global array1,count                   //声明所定义的数组和数组长度是全局变量
.global array2,count                   //声明所定义的数组和数组长度是全局变量
.section .data
.align 1
array:                                 //定义保存数制转换后的数组
.byte 0,0,0,0,0,0,0,0
.align 1
```

5.3.1 数制转换程序设计

2、汇编语言如何声明函数以及对函数进行外部声明

```
.type convert_binary function          //声明convert_binary为函数类型
.global convert_binary                 //将convert_binary定义成全局函数
.type convert_hexade function
.global convert_hexade
.type convert_octal function
.global convert_octal
```

5.3.1 数制转换程序设计

3、数值转换函数功能解析

3-1：convert_binary:

参数：r0---要转换的十进制数

r1---存放转换后的二进制位串的首地址，高位在前低位在后。

例如：13转换成二进制数为:b00001101

3-2：convert_hexade:

参数：r0---要转换的十进制数

r1---存放转换后的十六进制位串的首地址，高位在前低位在后。

例如：13转换成二进制数为:0x0D

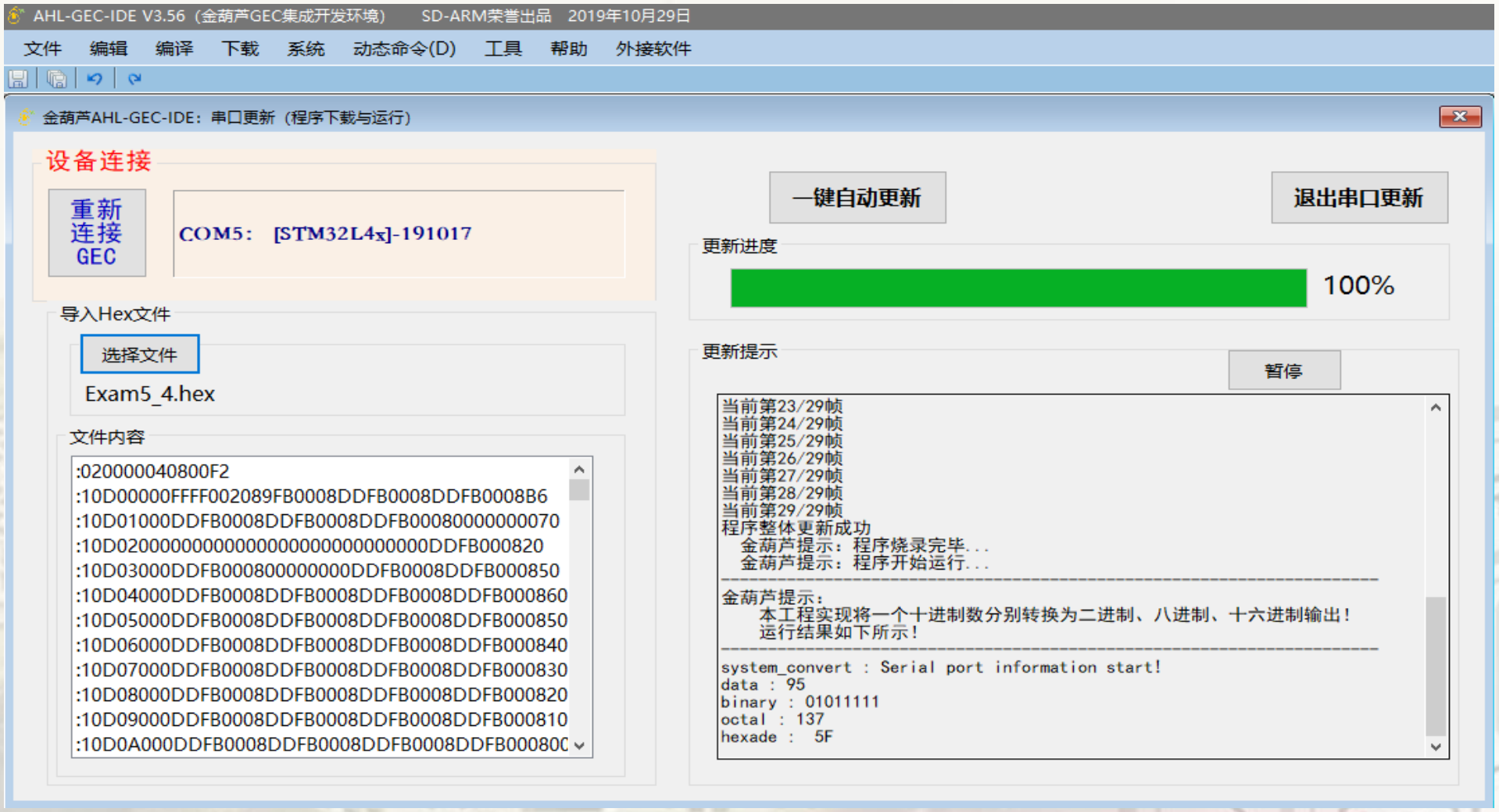
3-3：convert_octal:

参数：r0---要转换的十进制数

r1---存放转换后的十六进制位串的首地址，高位在前低位在后。

例如：13转换成二进制数为:0x0D

进制转换程序执行结果：


AHL-GEC-IDE V3.56（金葫芦GEC集成开发环境） SD-ARM荣誉出品 2019年10月29日
文件 编辑 编译 下载 系统 动态命令(D) 工具 帮助 外接软件
金葫芦AHL-GEC-IDE：串口更新（程序下载与运行）
设备连接
重新连接GEC
COM5： [STM32L4x]-191017
一键自动更新
退出串口更新
更新进度
100%
导入Hex文件
选择文件
Exam5_4.hex
文件内容
:020000040800F2
:10D00000FFFF002089FB0008DDFB0008DDFB0008B6
:10D01000DDFB0008DDFB0008DDFB00080000000070
:10D020000000000000000000000000000DDFB000820
:10D03000DDFB000800000000DDFB0008DDFB000850
:10D04000DDFB0008DDFB0008DDFB0008DDFB000860
:10D05000DDFB0008DDFB0008DDFB0008DDFB000850
:10D06000DDFB0008DDFB0008DDFB0008DDFB000840
:10D07000DDFB0008DDFB0008DDFB0008DDFB000830
:10D08000DDFB0008DDFB0008DDFB0008DDFB000820
:10D09000DDFB0008DDFB0008DDFB0008DDFB000810
:10D0A000DDFB0008DDFB0008DDFB0008DDFB00080C
更新提示
暂停
当前第23/29帧
当前第24/29帧
当前第25/29帧
当前第26/29帧
当前第27/29帧
当前第28/29帧
当前第29/29帧
程序整体更新成功
金葫芦提示：程序烧录完毕...
金葫芦提示：程序开始运行...
金葫芦提示：
本工程实现将一个十进制数分别转换为二进制、八进制、十六进制输出！
运行结果如下所示！
system_convert : Serial port information start!
data : 95
binary : 01011111
octal : 137
hexade : 5F

5.3.2 冒泡排序程序设计

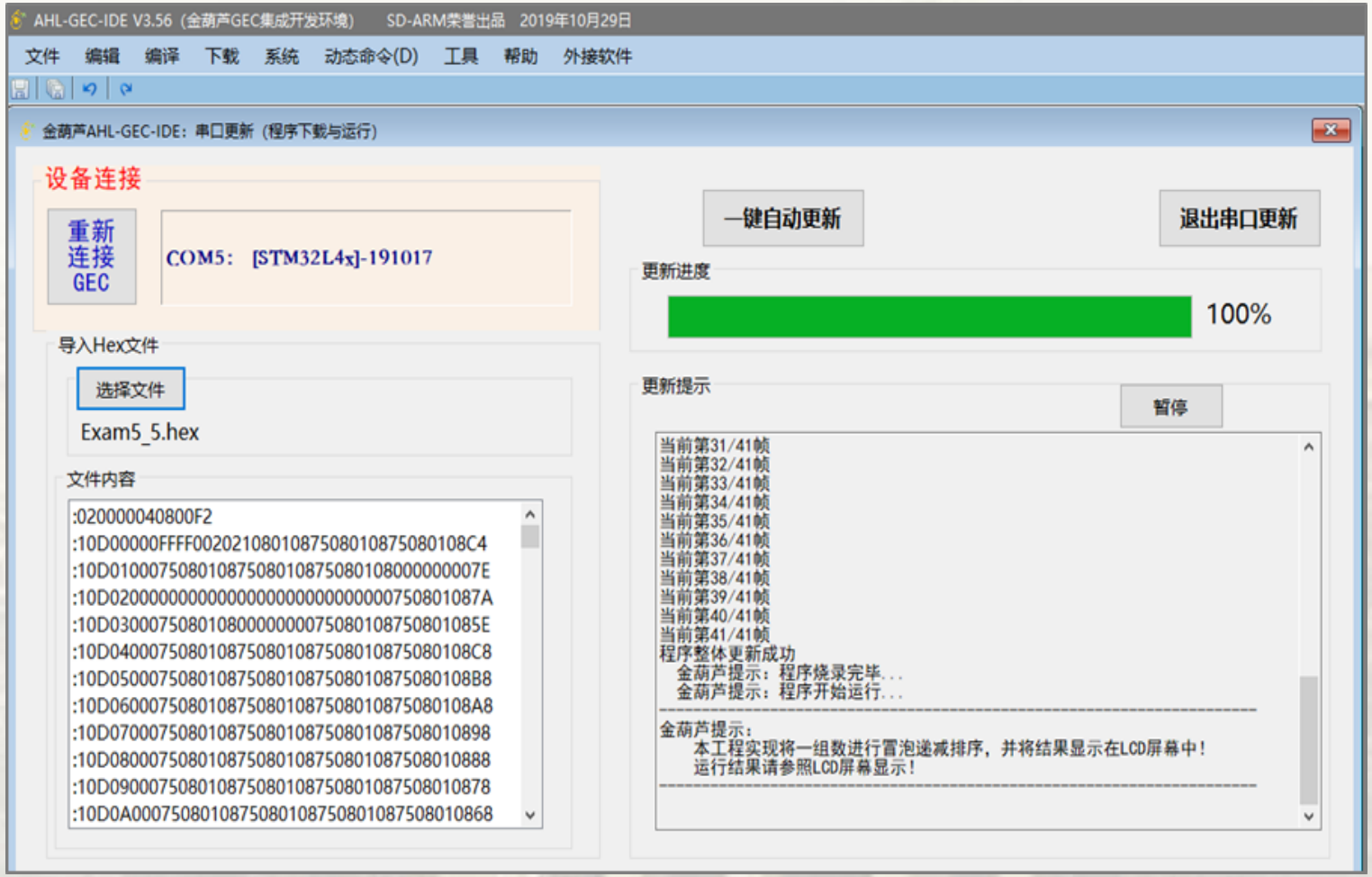
【例5-5】输入一组数据，编程实现冒泡递减排序效果，并通过LCD屏幕显示输出结果。假设待排序的一组数是：12，15，8，14，16，10，2，30参考程序见“Exam5_5”工程。

分析：

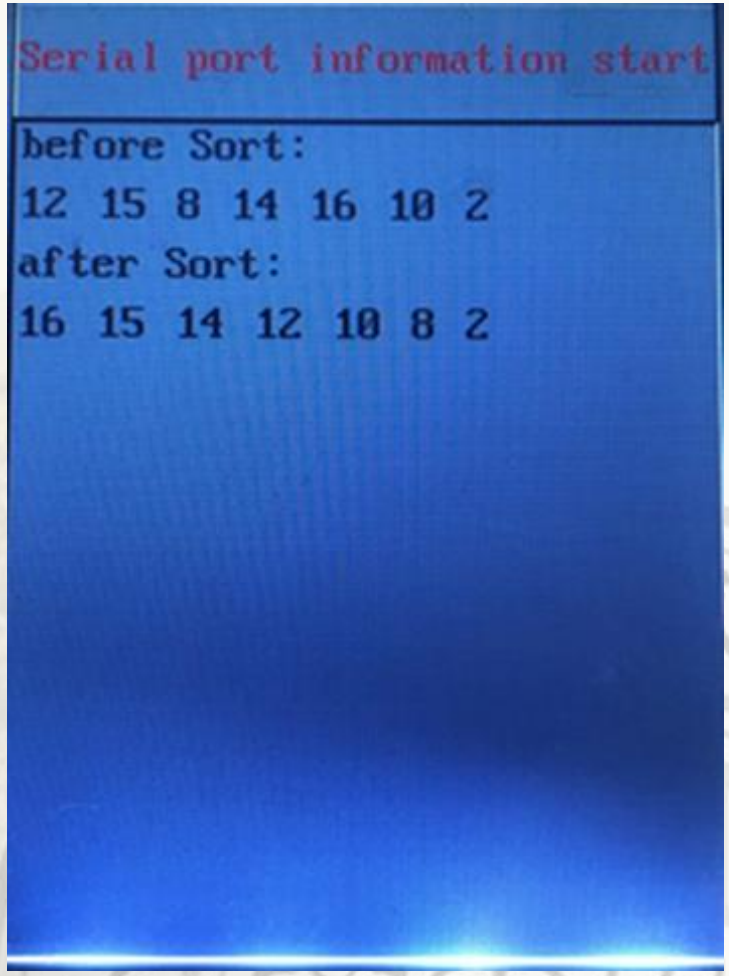
（1）硬件接法：用SWD连接目标套件和PC机的USB端口；LCD屏幕的8个针脚与底板标有“彩色LCD”字样处相连。

（2）这里需要定义一个数据段来存储数据，可以设定存取每个数据都占用一个字节；排序过程主要分为外循环和内循环两个程序段，每执行完一次内循环，外循环需要循环的次数减1，当外循环需要循环的次数为0时，说明排序完成。

冒泡排序串口输出提示信息：



LCD屏幕显示冒泡排序程序执行结果



5.4 实验二：基于构件方法的汇编程序设计

1. 实验目的

（1）对构件基本应用方法有更进一步的认识，初步掌握基于构件的汇编设计与运行。
（2）理解汇编语言中顺序结构、分支结构和循环结构的程序设计方法。
（3）理解和掌握汇编跳转指令的使用方法和场合。
（4）掌握硬件系统的软件测试方法，初步理解printf输出调试的基本方法。

2. 实验准备

3. 参考样例

参照“Exam5_5”工程。该程序通过申请一段绝对地址空间用来存储一组数据，并通过编程实现冒泡从大到小的排序，最后用串口输出排序后的结果。

4. 实验过程或要求

（1）验证性实验
（2）设计性实验
在验证性实验的基础上，自行编程实现100以内的奇数相加所得到的和，最后通过串口输出该结果。（3）进阶实验★
利用“Exam5_5”工程提供的一组数据，也可自行定义一组数据，采用选择排序算法对这组数据进行从小到大排序。

5. 实验报告要求

6. 实践性问答题

（1）若要通过串口输出冒泡排序结果，该如何实现。
（2）若要实现冒泡排序从小到大的顺序，则应修改哪些语句。

作业：1~6