项目一　LED 矩阵广告牌的制作

项目简介

图 1- 1　日常生活中的 LED 广告牌

太阳刚落山，闹市的街头华灯闪烁，多么美丽的灯光，流光溢彩、姹紫嫣红；人们醉心于霓虹灯闪烁下的都市。这些光彩夺目的灯光效果其实都可以用单片机技术来实现。图 1- 1 为日常生活中用到的各种各样 LED 广告牌。

本项目主要任务：做一个属于自己的 LED 矩阵广告牌。制作 LED 广告牌首先得制作硬件电路；接着学习让一只 LED 闪烁；然后学习让 8 只 LED 流水显示，最后学习控制 32 只 LED 组成的矩阵广告牌。其流程如图 1- 2 所示。

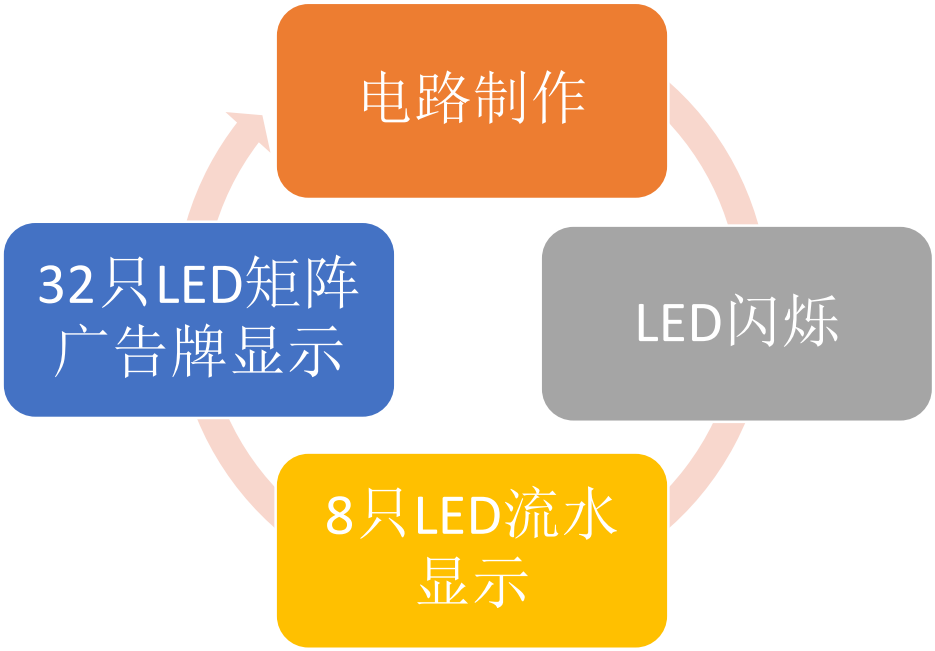

图 1- 2　LED 矩阵广告牌任务分解

任务一　单片机最小系统 PCB 硬件制作

一、任务要求

要学习单片机，首先得搭建硬件电路，然后才能编写程序和做各种实验。本书中附送了一块 PCB 总板。其中有一块单片机最小系统主机模块（后简称主机模块）。其实物如图 1- 3 所示。主机模块提供电源接口，串行接口，ISP 下载接口，兼容 51 系列单片机的最小系统电路，复位电路，电源指示灯、P0 口上拉电阻等多种电路功能。简单的说，就是把供电与烧录程序这些硬件都集成在了这块小板子上。由于主机模块中电路在每个模块上都可以公用的，所以本书中所有功能电路都可以通过接口转接的方式由最小系统板控制。

本项目中第一个任务就是完成单片机最小系统的制作。

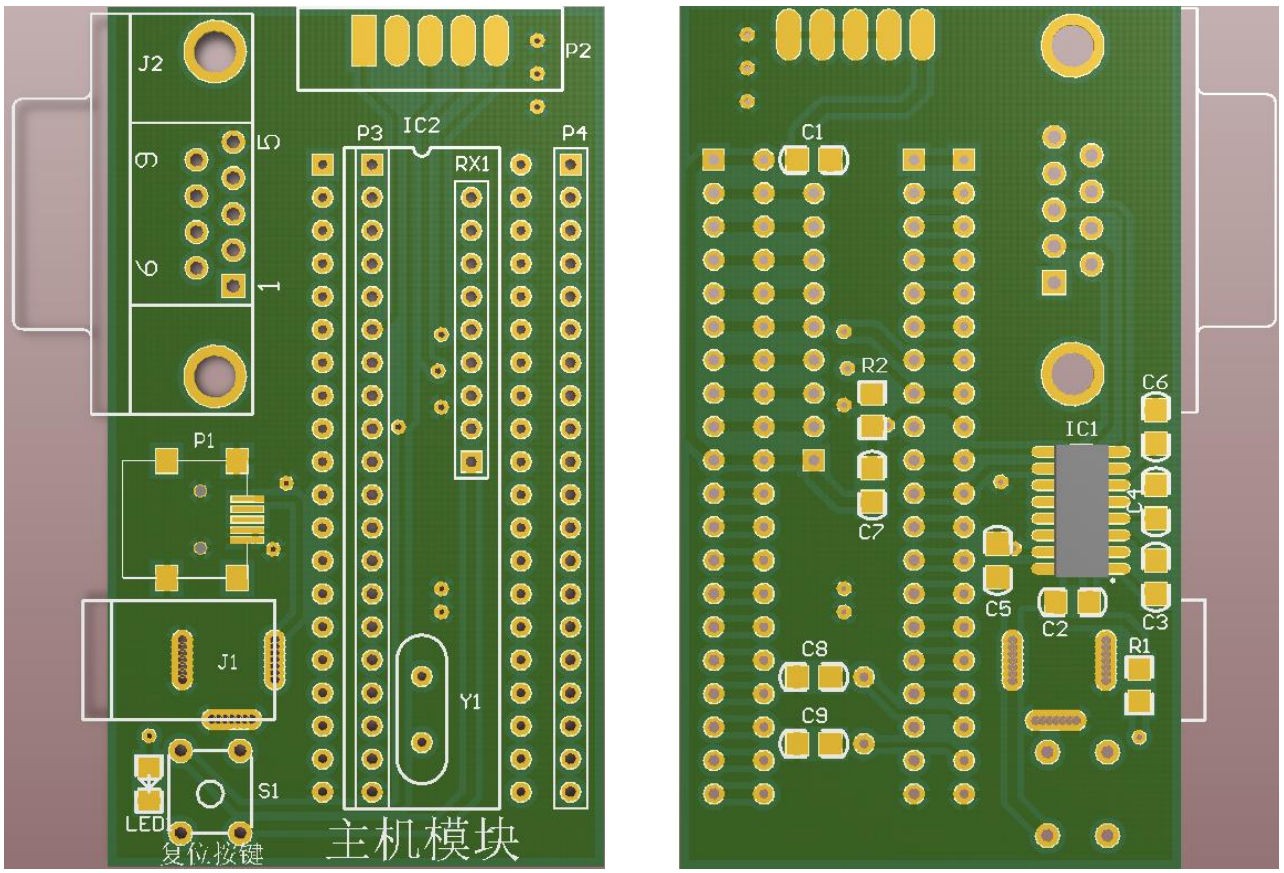

图 1- 3　单片机最小系统模块正面与背面 PCB 图

本任务要求完成主机模块电路板焊接。需要注意的是：单片机最小系统模块采用器件双面安装工艺其焊接实物如图 1- 4 所示。主机模块中排阻 RX1、晶振 Y1 与集成电路底座 IC1 采用重叠插装工艺如图 1- 5 所示。

需要根据电路原理图及装配图完成实物 PCB 的焊接。实物如图 1- 4 所示。

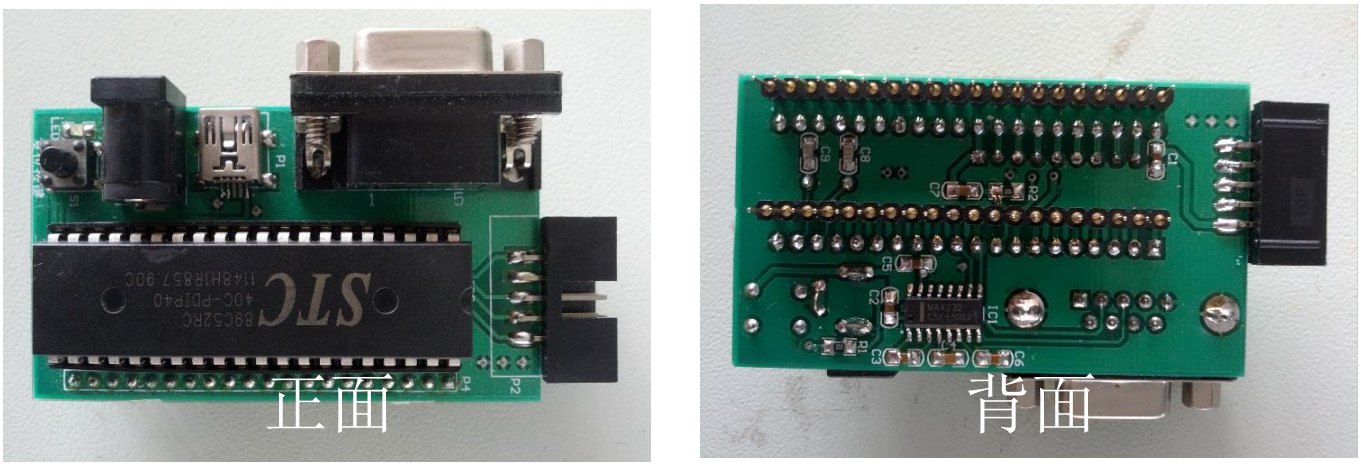

图 1- 4　主机模块实物效果示意图

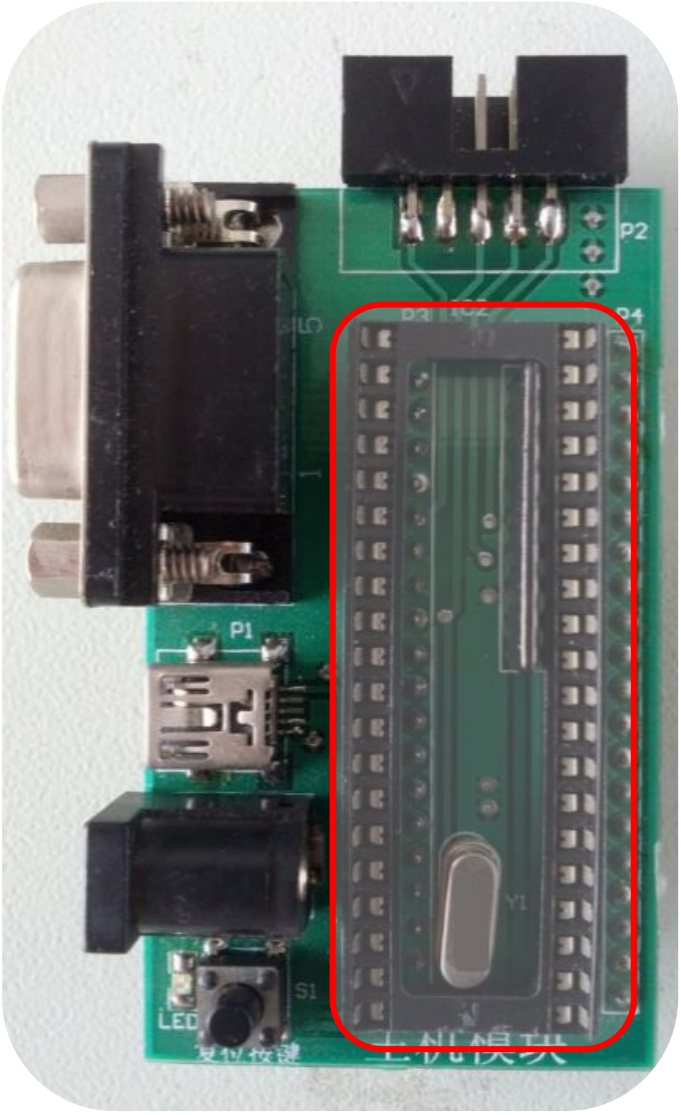

图 1- 5　RX1（P0 上拉）、Y1（晶振）与单片机底座采用重叠插装工艺

二、相关知识

要完成印制电路的焊接，首先得了解用到的这些元器件。下面就本次任务需要用到的元器件做一个简单介绍。

1、STC90C52RC 单片机或 51 兼容单片机

图 1- 6　STC90C52RC 单片机实物图

本书中使用单片机为 STC90C52RC，此单片机是台湾宏晶科技生产的增强型单片机，其实物图如图 1- 8 所示。STC90C52RC 是一款高速/低功耗/超强抗干扰的新一代 8051 单片机，指令代码完全兼容传统 8051,但速度快 8-12 倍。本书选型使用此款单片机是因为其烧录程序较为方便，可以使用串口直接下载程序。同时，本书中主机模块还提供了 ISP 下载接口，若使用 ATMEL 公司的 AT89S52 系列单片机也可以通过此接口烧录程序。

1.1　STC90C52RC 引脚及其功能

STC90C52RC 系列的单片机封装形式有 LQFP44、LQFP48、PLCC44、QFN40 和 PDIP40。本书使用 PDIP40 封装的 STC90C52RC 单片机，其管脚结构图如图 1-7 所示。

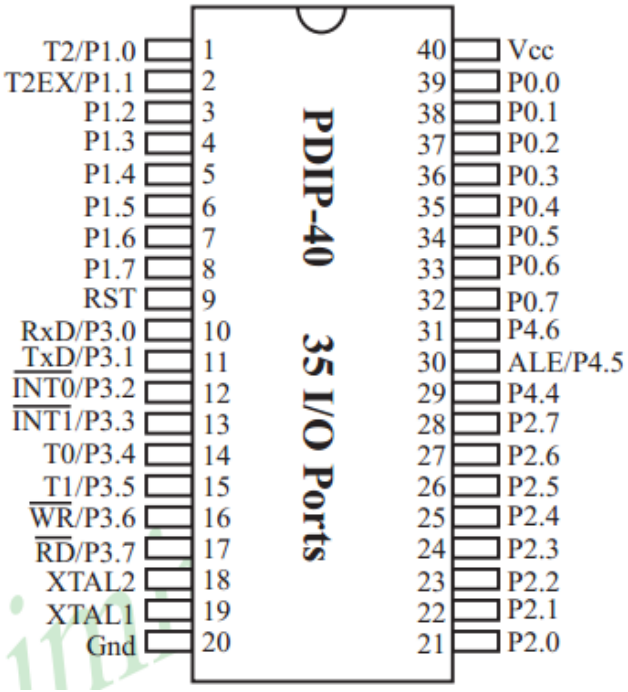

图 1- 7　STC90C52RC 引脚图

2、MAX232

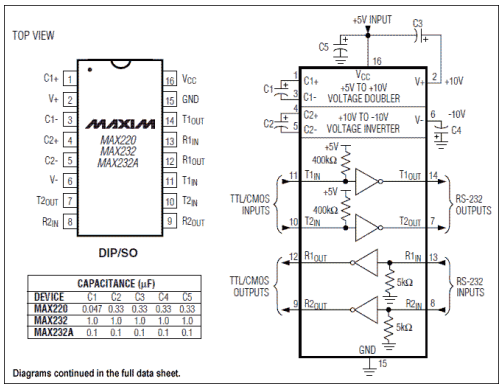

图 1- 8　MAX232 及其管脚图

MAX232 芯片是 MAXIM 公司专为 RS-232 标准串口设计的单电源电平转换芯片，使用+5v 单电源供电，其主要特点：1、符合所有的 RS-232C 技术标准，2、只需要单一+5V 电源供电，3、片载电荷泵具有升压、电压极性反转能力，能够产生+10V 和-10V 电压 V+、V-，4、功耗低，典型供电电流 5mA，5、内部集成 2 个 RS-232C 驱动器，6、高集成度，片外最低只需 4 个电容即可工作。MAX232 焊接在主机模块 PCB 背面。如图 1- 4 右图所示。

3、电容（贴片）

电容器通常简称其为电容，用字母 C 表示。电容器，顾名思义，是“装电的

容器”，是一种容纳电荷的器件。电容是电子设备中大量使用的电子元件之一，广泛应用于电路中的隔直通交，耦合，旁路，滤波，调谐回路，能量转换，控制等方面。本项目中贴片电容主要做滤波作用，为 0805 封装。如图 1- 9 所示。

4、电阻（贴片）

图 1- 9　贴片电容

图 1- 10　贴片电阻

电阻器在日常生活中一般直接称为电阻。是一个限流元件，电阻器的阻值是固定的一般是两个引脚，将电阻接在电路中后，它可限制通过它所连支路的电流大小。阻值不能改变的称为固定电阻器。本项目中使用贴片电阻，其封装统一为 0805。如图 1- 10 所示。

5、MINI-USB 接口（贴片）

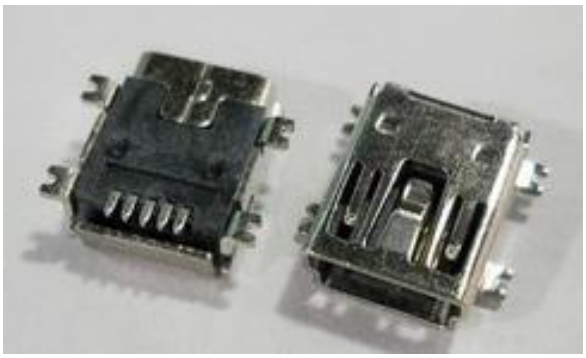

图 1- 11　MINI-USB

MINI-USB 接口是一种规范接口，标准 USB 接口是计算机都具备的接口。本项目中主机模块通过它从计算机 USB 口获取 5V 直流电。这样在调试程序时就不需要外接电源了。但是需要注意，此接口只能提供 500mA 电流，所以若需要调试大功率电路，需要使用外接电源。

6、直流电源接口

图 1- 12　直流电源接口（5.5mm*2.1mm）

主机模块的另一路电源接口，此接口外接开关电源。在 usb 电源供电不足时可通过此接口从外接开关电源取电。适用于（5.5mm*2.1mm）标准接口的 5V 开关电源。其中 5.5 表示接口外径，2.1 表示接口内径。

7、晶体振荡器

图 1- 13　晶振

石英晶体振荡器是一种高精度和高稳定度的振荡器，被广泛应用于数电或控制芯片等各类振荡电路中，以及通信系统中用于频率发生器、为数据处理设备产生时钟信号和为特定系统提供基准信号。本模块用它组成时钟振荡电路为单片机提供工作时钟。

8、排阻

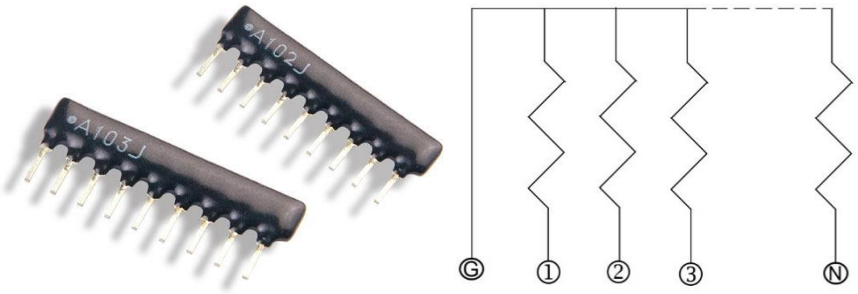

图 1- 14　RX8 排阻

排阻，就是若干个参数完全相同的电阻，它们的一个引脚都连到一起，作为公共引脚，其余引脚正常引出如图 1-14 所示。所以如果一个排阻是由 n 个电阻构成的，那么它就有 n+1 只引脚，一般来说，文字面左侧的那个是公共引脚。它在排阻上一般用一个圆点标出来。焊接时注意公共脚插入方孔内（1 号脚）。

9、40 脚底座

图 1- 15　40 脚底座

图 1- 4 中我们能看到在它的上方插入了一片 40 脚的芯片（单片机），为了编程和更换方便单片机不直接焊接在板子上，而是焊接图 1-15 所示的底座。底座的缺口在插装时注意要和 PCB 上印制的 IC1 的缺口一致。

10、轻触按键

轻触按键是按键产品下属的一款分类产品，它其实相当于是一种电子开关，只要轻轻的按下按键开关接通，松开时开关就断开连接，实现原理主要是通过轻触按键内部的金属弹片受力弹动来实现接通和断开的。其内部接构如图 1-16 所示。从图中可以看出，1,2 脚相连，3,4,脚相联，但按下按键时对角两个触点导通，反之对角两触点断开。

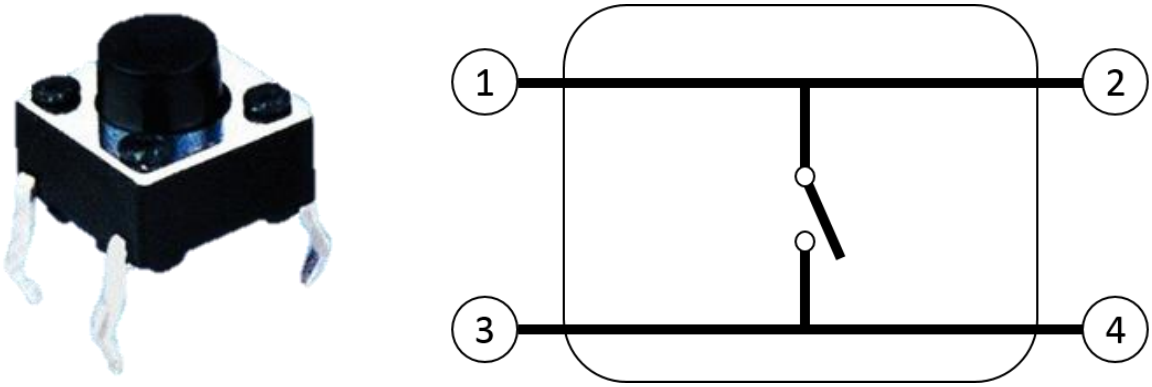

图 1- 16　轻触按键与其内部结构示意图

11、10P 牛角底座

图 1- 17　10P 牛角底座

主机模块用它做 ISP 下载的接口以兼容 AT89SXX 系列单片机。若主机模块中不需要使用 AT89SXX 系列单片机时，此接口可以不焊接。

12、板插串口母头（直角母头）

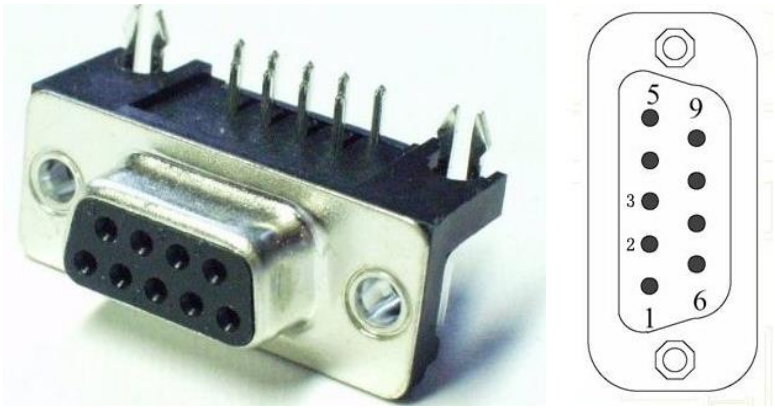

图 1- 18　直角串口母头

串行接口简称串口，也称串行通信接口（通常指 COM 接口），是采用串行通信方式的扩展接口。一条信息的各位数据被逐位按顺序传送的通讯方式称为串行通讯。串行通讯的特点是：数据位的传送，按位顺序进行，最少只需一根传输

线即可完成；成本低但传送速度慢。串行通讯的距离可以从几米到几千米；根据信息的传送方向，串行通讯可以进一步分为单工、半双工和全双工三种。其 2 脚为 RXD，3 脚为 TXD，5 脚为 GND。主机模块用它来连接 PC 实现对 STC 的单片机进行编程烧录。程序烧录后，单片机也可以通过此接口与 PC 主机进行通讯。

12、接口圆脚插针

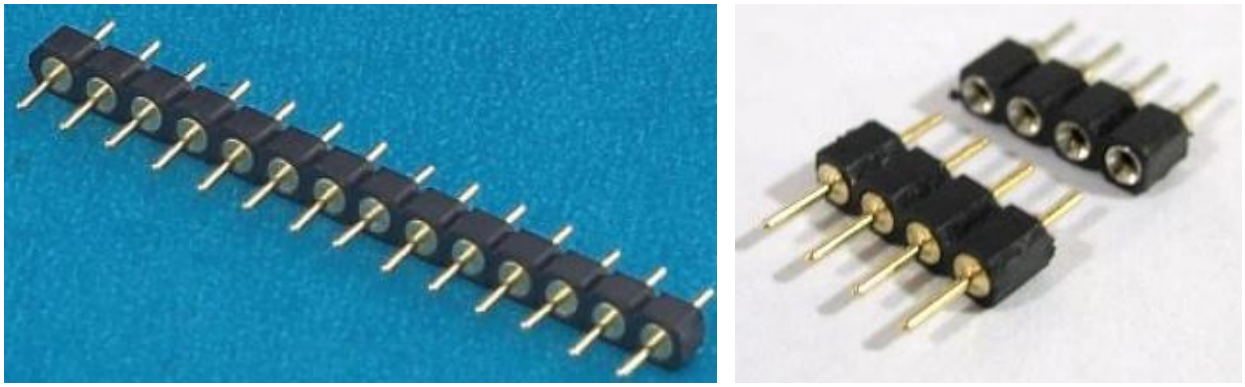

图 1- 19　圆脚插针（100mail）

主机模块中有两排 20 脚插针，通过这两排插针使主机板与各种功能电路板相连接。其实物示意图如图 1- 19 所示。

三、操作训练

请按照电路原理图与 PCB 装配图焊接主机模块电路板。其 PCB 板装接工艺要求如下：

1、装接工艺要求：

1.1　贴片元件的安装工艺

贴片电阻、贴片电容、贴片 LED 水平安装，贴紧印制板。电阻标号方向应一致。

1.2　贴片集成电路安装工艺

RS232 等贴片集成电路水平安装，贴紧印制板，安装时请注意集成电路方向，避免装反。

1.3　直插集成电路安装工艺

直插式集成电路安装，请先安装好底座，再将集成电路插入。安装时请注意

集成电路底座方向与装配图一致，避免装反。

1.4 直插元件安装工艺

串口接头、圆脚排针、晶振、排阻等直插式元件请直立安装，插到底。

1.5 10P 牛角底座安装工艺

若系统中需要焊接 10P 牛角底座，请将牛角底座横向，电路板接口位置插入牛角底座两排排针之间。然后双面焊接。注意：牛角底座缺口面朝向应与 PCB 正面一致。

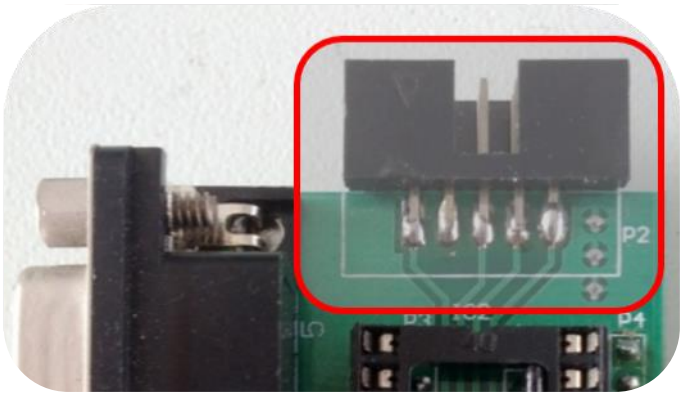

图 1- 20 牛角插座横向安装示意图

1.6 其他工艺要求

所有插入焊盘孔的元件引脚及导线均采用直脚焊，剪脚留头在焊面以上 0.5mm~1mm，未述之处均按常规工艺。

2、电路原理图

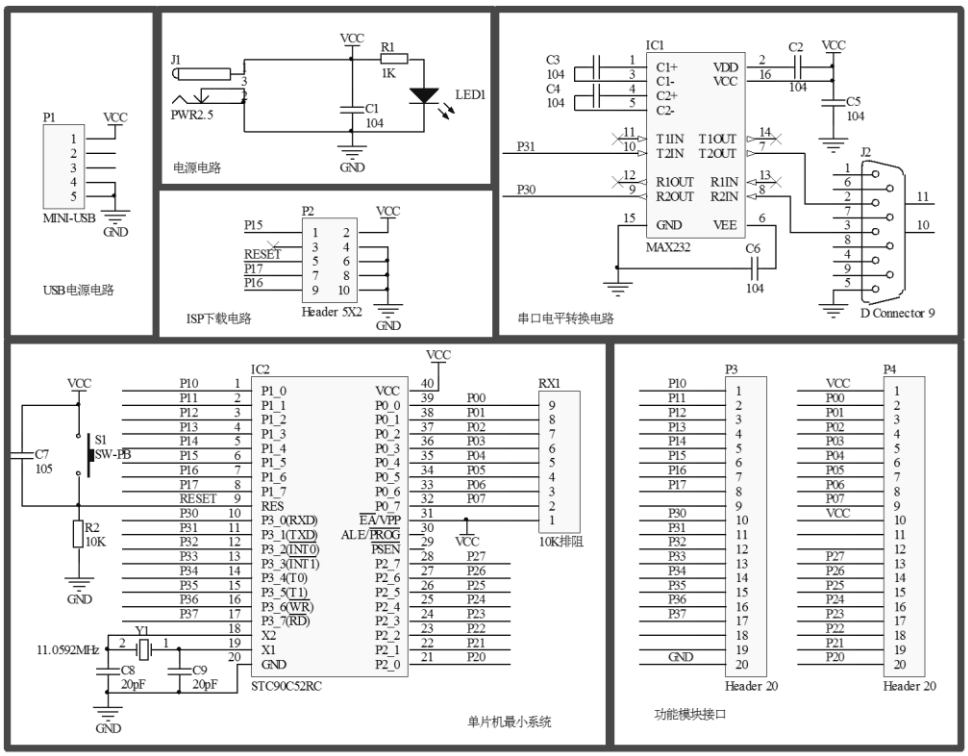

图 1- 21　主机模块原理图

3、PCB 板装配图

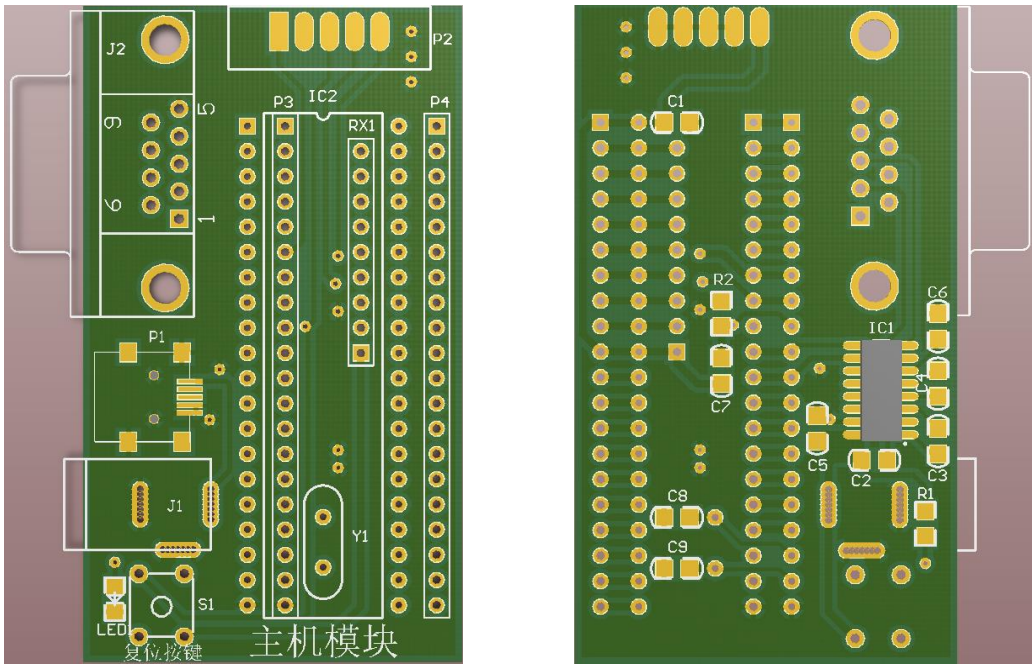

图 1- 22　主机模块装配图

4、元器件清单

元器件清单如下表所示：

表 1- 1　主机模块元器件清单

序号	元件名	参数	标号	封装	数量	备注
1	电容	104	C1~C6	0805C	6	
2	电容	105	C7	0805C	1	
3	电容	20P	C8、C9	0805C	2	
4	电阻	1K	R1	0805C	1	
5	电阻	2K	R2	0805C	1	
6	发光二极管	0805 红光	LED1	0805C	1	
7	集成电路	MAX232	IC1	SOIC127P600-16M	1	
8	单片机底座	DIP40	IC2	DIP40	1	
9	单片机	STC90C52RC	IC2	DIP40	1	
10	晶振	11.0592	Y1	X1	1	
11	排阻	10K	RX1	SIP9	1	
12	迷你 USB	miniUSB	P1	MINI-USB 母头	1	
13	牛角插座	10P	P2	10P	1	可不焊
14	串口	DB-9 母头	J2	DB-9	1	
15	电源接口	5.5x2.1 母头	J1	PWR2.1	1	
16	轻触按键	6mmX6mmX6	S1	6X6X6 按键	1	
17	20 脚插针	引脚间距 100mai	P3、P4	SIP20	2	

5、电路调试与程序下载

电路完成后，怎么才能知道所制作的 PCB 电路板是否制作成功呢。我们可以通过 STC 官方的程序烧录软件来验证此模块是否能正常工作。

5.1　STC-ISP 烧录软件获取

请至 STC 官网（http://www.mcu-memory.com/）下载最新版 PC 端 ISP 控制软件”STC-ISP”。其图标如图 1-23 所示。截止本书完成 STC-ISP 最新版为 V6.52B。

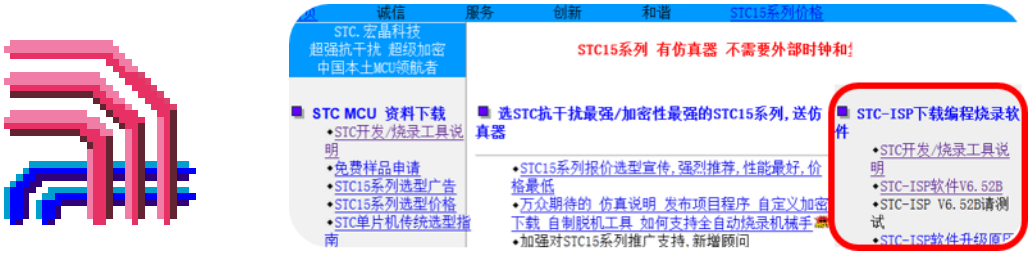

图 1- 23　STC-ISP 软件图标及官网下载界面

5.2　STC-ISP 单片机烧写软件使用

双击运行下载的 STC-ISP.EXE 软件。由于 STC-ISP 软件需要管理员权限才能运

行所以双击后可能会跳出用户账户对话框，点“是”授予管理员权限（仅 WIN7 系统）。之后进入 STC-ISP 软件界面，如图 1- 24 所示。

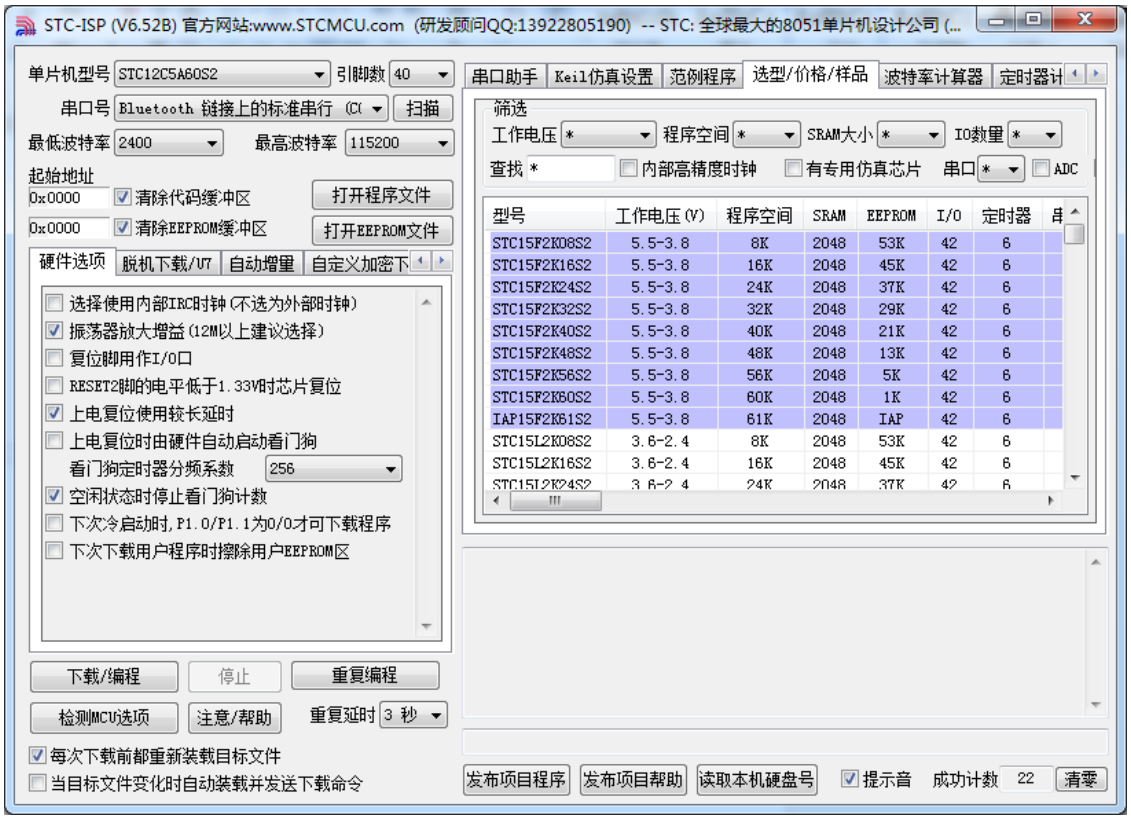

图 1- 24　STC-ISP 软件界面

（1）芯片选择

在单片机型号下拉菜单中找到 STC90C52RC 系列下的 STC90C52RC 单片机。如图 1- 25 所示。

（2）检测芯片

在软件“串口号”下拉菜单中选择当前 PC 与单片机通讯的串口（一般台式机默认为 COM1）。然后再下方最低波特率选择 2400，为确保通讯正常，选择最高波特率为 9600（此数值设置过高可能会导致通讯不成功）。然后点击“检测 MCU 选项”按钮，如图 1- 25 所示。之后软件提示“正在检测目标单片机…”。然后用串口线把电脑串口与主机模块串口连接起来。当主机模块接入 5V 电源后，若硬件正常，系统会显示“正在握手…”，“握手通过…”等字样。并在最后显示当前单片机型号。若型号信息为“STC90C52RC”说明单片机时钟电路与串口电路正确。主机模块 PCB 功能正常。

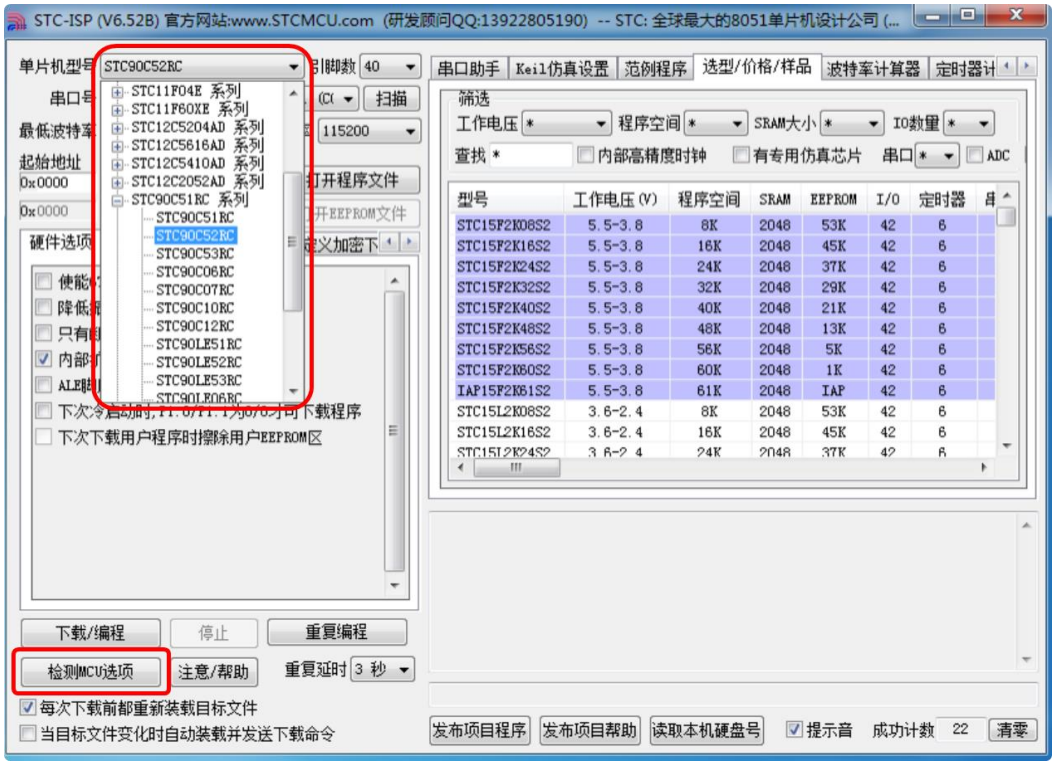

图 1- 25　单片机选型与自动检测 MCU

5.3　KEIL 软件获取

本步骤主要测试能否通过 STC-ISP 软件把程序文件（*.HEX）烧写到单片机当中。要烧写程序首先得产生能够被单片机识读的“*.HEX”文件。HEX 文件的生成必须要用到程序编写软件 KEIL。本书中使用 KEIL V7.01 版系统。可以通过登陆 KEIL 官方网站（http://www.keil.com）获取并下载 KEIL 安装文件。下载完成后，请运行安装文件，并按提示完成安装操作。

5.4　KEIL 软件打开范例工程“HELLO”

KEIL 软件安装完成后。点击开始菜单或桌面 KEIL 图标运行软件。其图标如图 1-26 所示。打开软件后，点击“Project”（工程）选项中的“Open Project”（打开工程）选项，如图 1-27 所示。找到 KEIL 自带的“HELLO”例程（默认安装情况下所在位置为 C:\KEIL\C51\EXAMPLES\HELLO\HELLO.uv2）。

图 1- 26　KEIL 文件图标

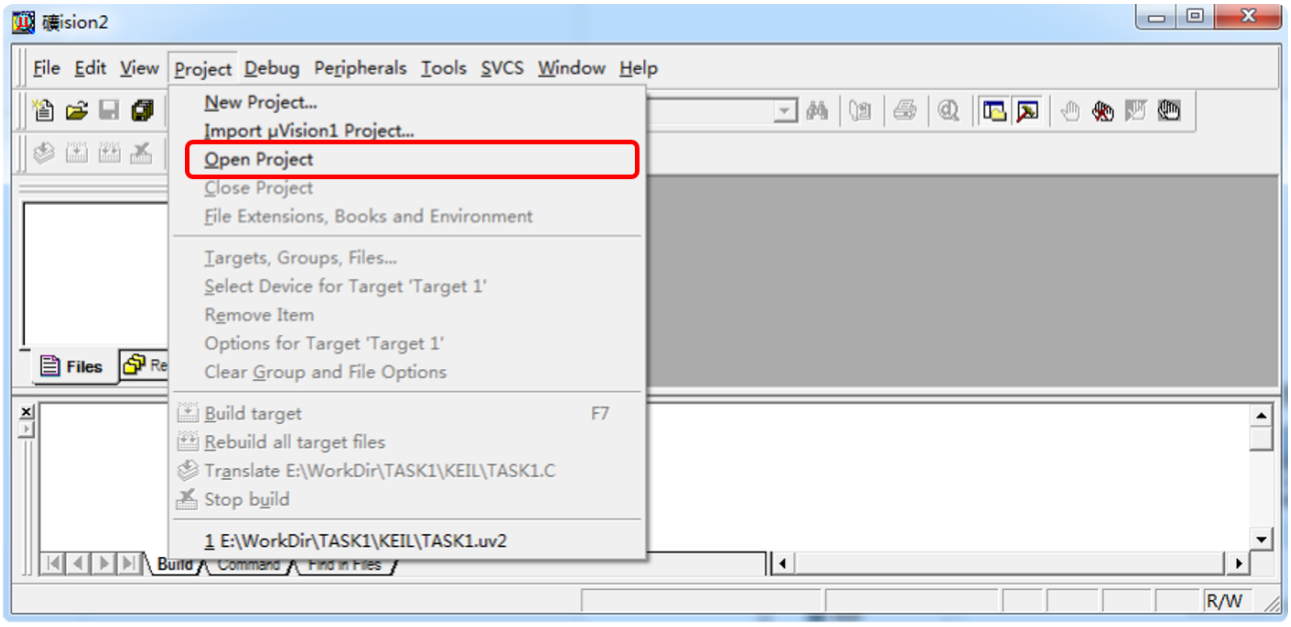

图 1- 27　KEIL 打开工程

工程打开后，左方会有 HELLO.C 文件，说明工程载入成功，下一步就是利用这个工程生成 HEX 文件了。

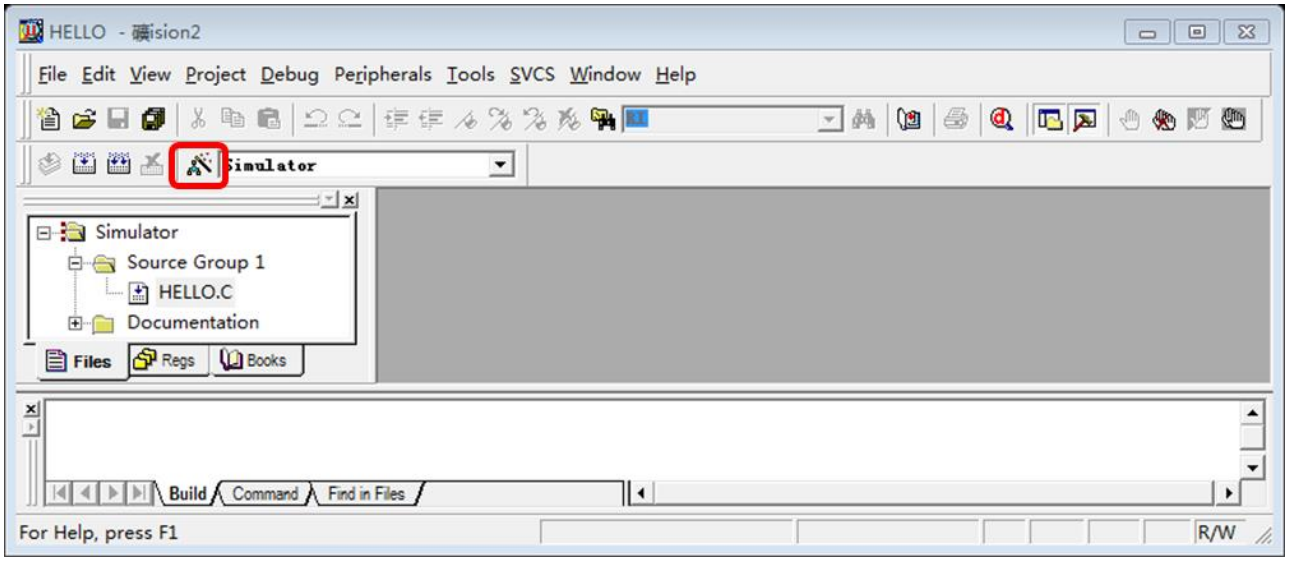

图 1- 28　成功打开 HELLO 工程

5.5　利用 HELLO 范例工程生成 HEX 文件

点击图 1- 28 中红色方框中的魔棒图标，进入工程设置选项菜单。勾选此菜单的 Output 菜单卡，选中 Create HEX File 选项。如图 1- 29 所示。然后点击 图标，就能生成当前范例工程 HELLO 的 HEX 程序文件。此 HEX 文件可以被烧录到单片机当中。生成文件成功，软件会在编译提示中显示” creating hex file from "HELLO"...”的字样。如图 1- 30 所示。下面就要做程序烧录了。

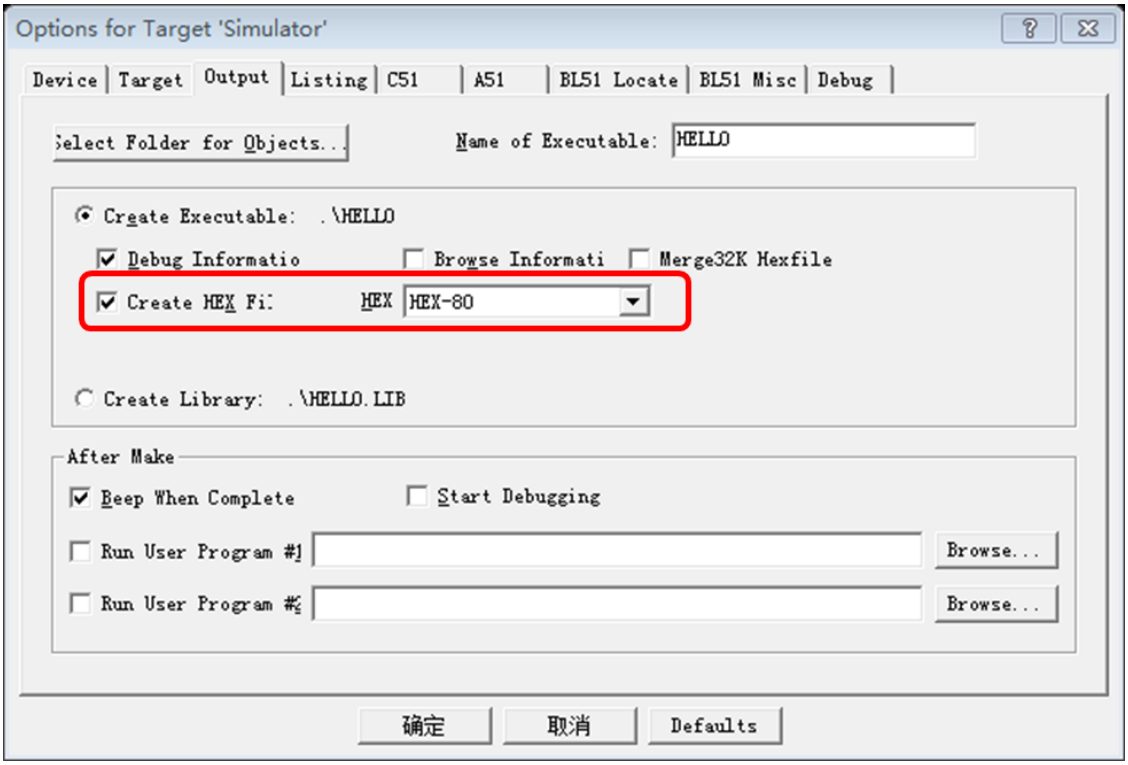

图 1- 29　勾选 Create HEX File 选项

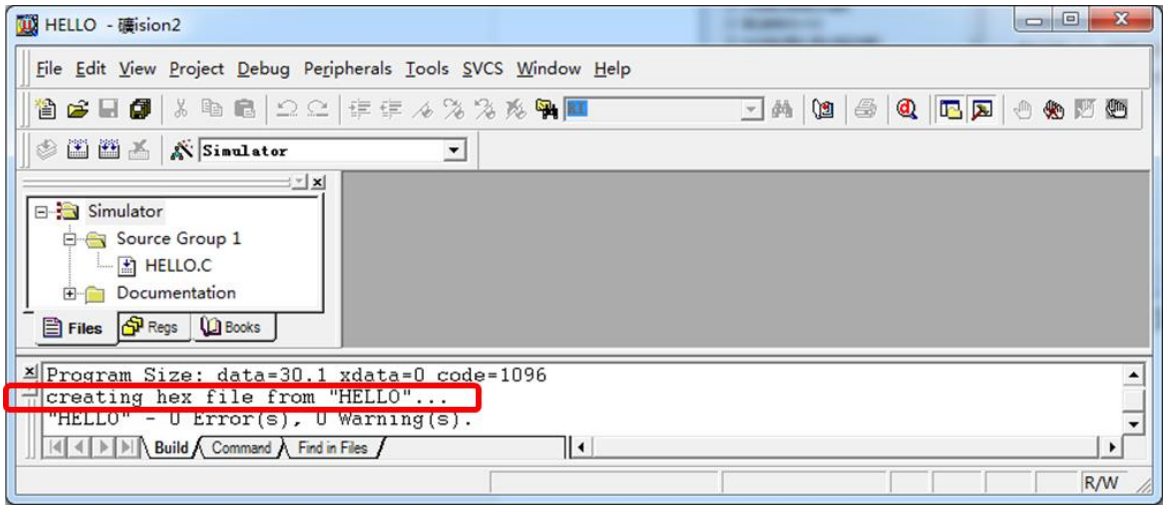

图 1- 30　创建 HEX 文件提示

5.6　把 HEX 文件烧写到单片机中

重新打开 STC-ISP 软件，点击打开程序按钮，在刚才的 HELLOW 文件夹中找到并选中 HELLOW.HEX 文件。按下“下载/编程”按钮开始编程。如果主机模块焊接正确的话，很快软件就能提示烧写完成。如图 1-31 所示。若不能下载正确，请检查硬件电路是否存在元器件缺焊虚焊的情况，是否装反或装错。

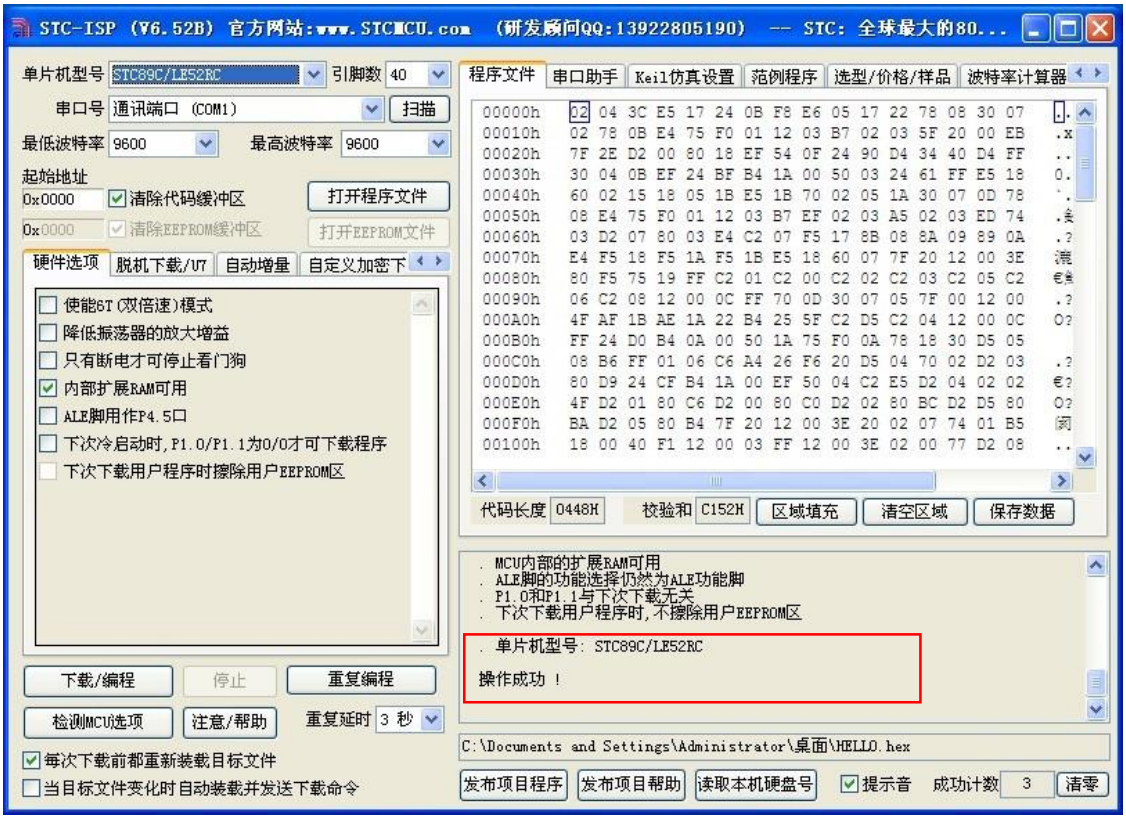

图 1- 31　HELLOW.HEX 程序成功下载

四、任务评价

表 1- 2　任务一完成情况汇总表

项目名称			评价时间		年　月　日		
小组名称		小组成员					
评价内容	评价要求	权重	评价标准	学生自评得分	小组评价得分	教师评价得分	合计
职业与安全意识	1．工具摆放、操作符合安全操作规程；2．遵守纪律、爱惜设备和器材、工位整洁；3．具有团队协作精神。	10%	好（10） 较好（8） 一般（6） 差（<6）				
印制线路板插件	1、插件正确，电阻标志方向一致。 2、元器件高度符合工艺要求，元器件装插平整。	15%	好（15） 较好（12） 一般（9） 差（<9）				
印制线路板焊接	焊点光亮、焊料适量，无虚焊、漏焊、假焊、搭锡、溅锡、铜箔起翘等现象,无毛刺、孔隙留脚长度，焊面以上0.5mm~1mm。	60%	好（60） 较好（45） 一般（30） 差（<30）				
印制线路板的总装	装配正确，无烫伤、划伤工件和导线，紧固件装配牢固可靠，导线剥头尺寸符合工艺要求绝缘恢复良好。	15%	好（15） 较好（12） 一般（9） 差（<6）				
教师签名			学生签名			总分	
项目评价=学生自评（0.2）+小组评价（0.3）+教师评价（0.5）							

五、知识拓展

“HELLO”工程为一个简单的串口发送程序，其功能主要是向上位机发送“hello world”字符。但是此范例工程使用芯片工作在 16MHZ 晶振下 1200 波特率。如果想要让上位机接收并显示正确单片机发送的字符，可以在 HELLO.C 文件修改代码。

“TH1=0xfd;”//使单片机产生 9600bps 的波特率

然后编译并把生成的 HEX 文件烧写入单片机中。上电后，通过 STC-ISP 软件中自带的串口助手就能源源不断的接收到单片机传送来的“hello world”字符了。

其程序修改位置与串口助手接收效果如图 1- 32 所示：

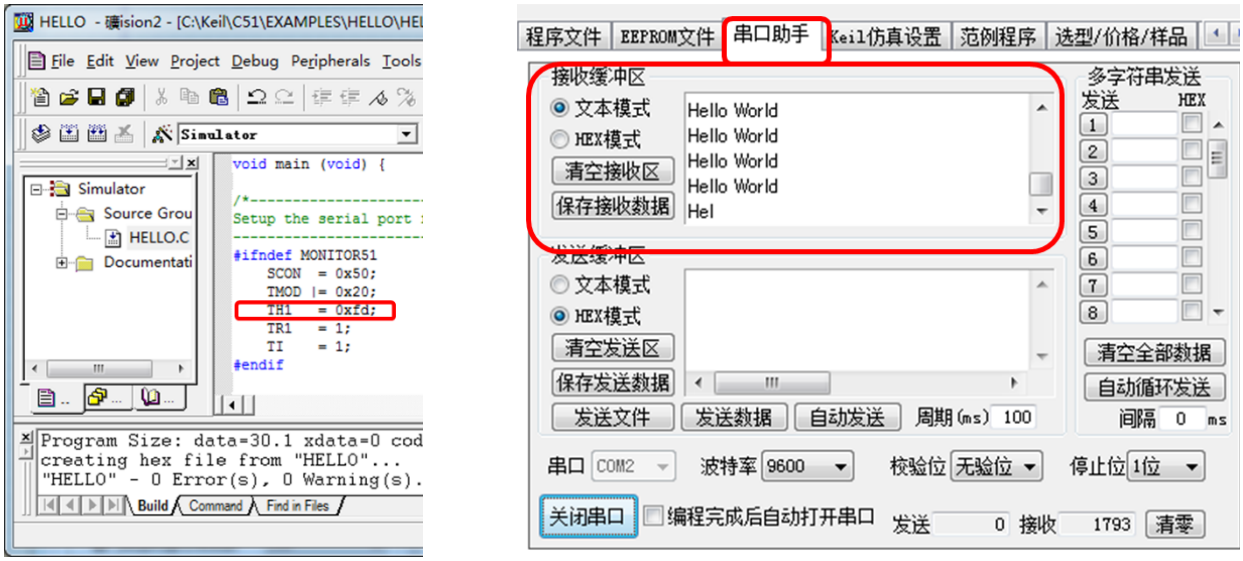

图 1- 32　修改及串口接收 HELLO WORLD 成功图示

六、思考与练习

请写一份装配报告，注明安装中遇到的问题与思考。

任务二　LED 矩阵广告牌 PCB 硬件制作

一、任务要求

任务一中已经完成了单片机主机模块的焊接工作。接下进入 LED 矩阵电路的搭建环节。在随书附送的 PCB 中，取出矩阵广告模块。其 PCB 板如图 1-33 所示。

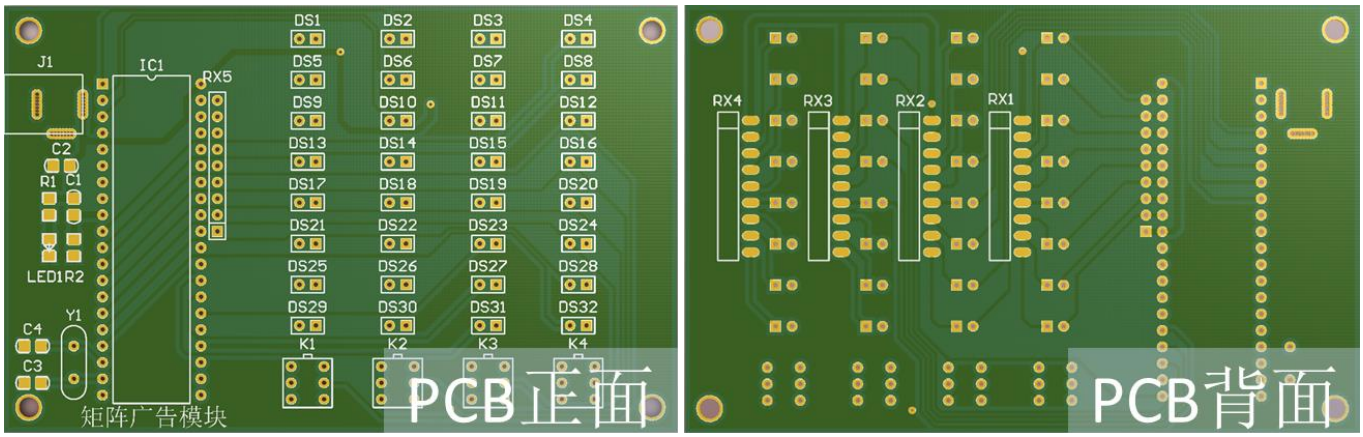

图 1-33　矩阵广告模块 PCB 实物图

本次任务要求完成矩阵广告模块电路板焊接。需要注意的是：由于我们使用最小系统板进行程序编写与调试，所以 PCB 正面左侧的单片机最小系统可以不焊，只需在 IC1 位置焊上底座就可以了如所示。若不使用主机模块，这些元器件需要焊接。

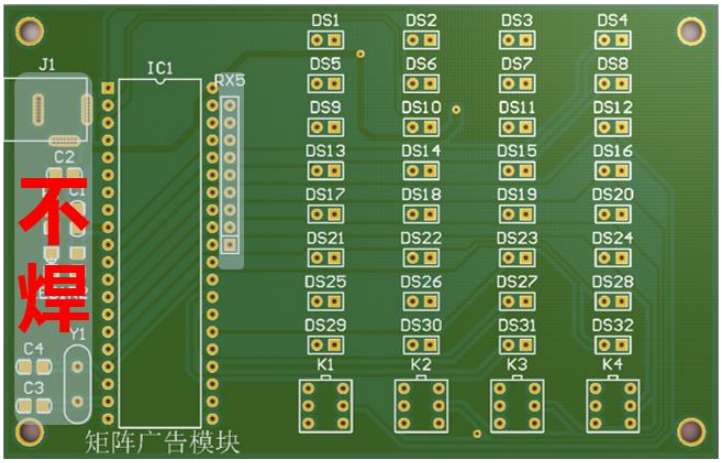

图 1-34　非必焊区域示意图

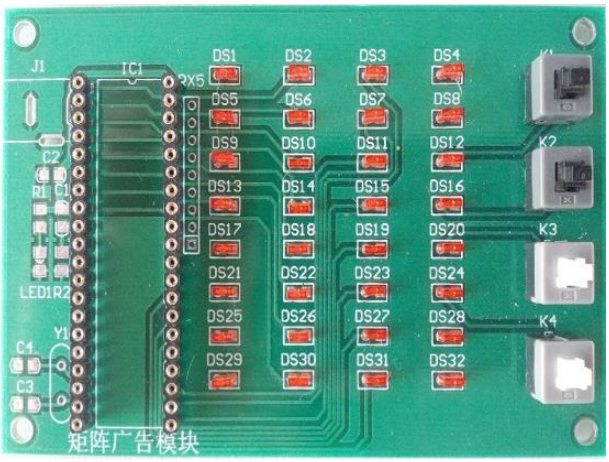

图 1-35　实物示意图（与新版不符）

图 1-35 为实物演示图，此图仅仅作为参考。本项目中新版 PCB 中为 LED 灯显示清晰所以把按键在挪到了 PCB 下方。本任务要求学生根据电路原理图及装配图装配要求完成矩阵广告模块实物 PCB 的焊接。

二、相关知识

要完成焊接任务，首先需要了解一下本项目中需要用到的元器件。下面就本次任务需要用到的未介绍过的元器件做一个简单介绍。

1、直插式发光二极管（红色）

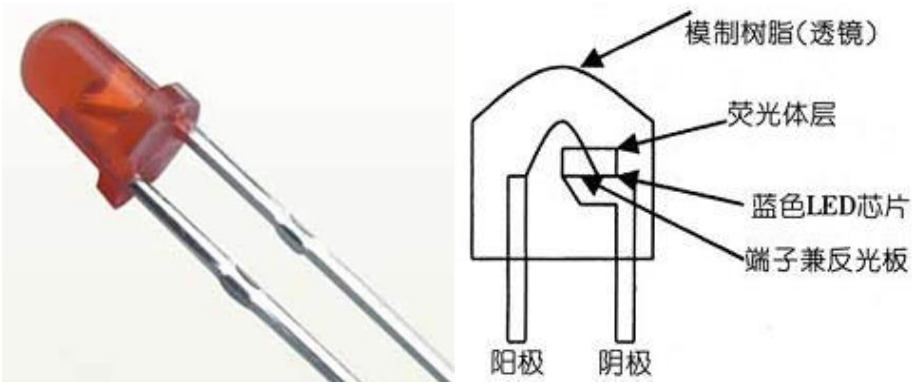

图 1- 36 发光二极管示意图及管脚图

发光二极管简称为 LED。由镓（Ga）与砷（AS)、磷（P）的化合物制成的二极管，当电子与空穴复合时能辐射出可见光，因而可以用来制成发光二极管。在电路及仪器中作为指示灯，或者组成文字或数字显示。

新出厂的发光二极管一只引脚长，一只短，长的那只为其正极，插装时注意将长脚插进方孔里。

2、6 脚按键（自锁）

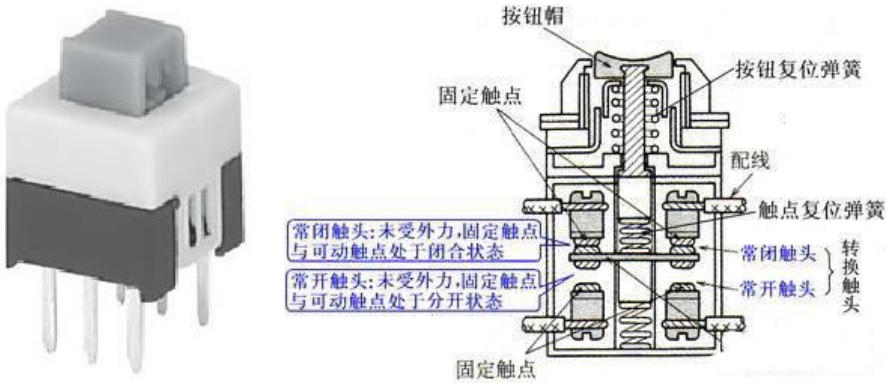

图 1- 37　自锁按键及内部原理示意图

相当于一个双刀双掷开关，一排的 3 个引脚为一组开关。中间的引脚和其左侧或右侧的引脚接通对应于按键处于按下或弹开状态。另外的 3 个引脚是另一个开关。其实物图如图 1- 37 所示。

3、6 脚按键（无自锁）

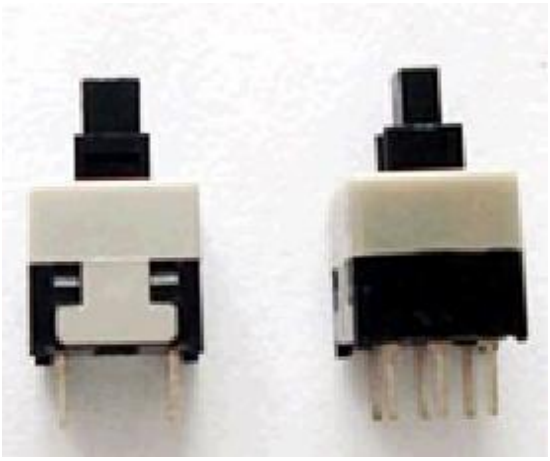

图 1- 38　无自锁按键

与自锁按键类似，区别在于按下的状态不能保持住。即没有卡子固定状态。

6 脚按键在焊接时需要注意，其有线条突起的哪一面应与 PCB 上按键示意图中突起位置相对应。

4、集成电路底座

本项目需要用到一种集成电路底座 40 脚的单片机底座。为了使电气连接更加可靠，项目中使用了圆孔 IC 集成电路底座，其实物如图 1- 39 所示。

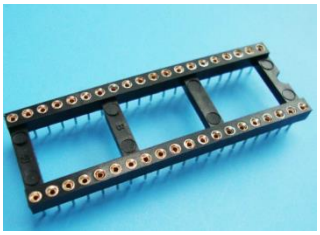

图 1- 39　圆孔集成电路底座

其余器件在主机模块中已经介绍过，这里就不再介绍了。认识了装接要使用的器件，就能开始实战操作了。

三、操作训练

请按照电路原理图与 PCB 装配图焊接 PCB 板。其 PCB 板装接工艺要求如下：

1、装接工艺要求：

1.1 贴片排阻的安装工艺

本项目中排阻安装方式较为特别，贴片安装在 PCB 板背面。安装时请注意排阻有标识圆点的 1 号脚应安装于 PCB 元器件示意图中有线条标识的一端。切勿装反。

1.2 直插元件安装工艺

发光二极管，6 脚按键等直插式元件请直立安装，其中 6 脚按键安装时需插到底，LED 发光二极管采取直立安装，其底面统一高出印制板 5mm。

1.3 其他工艺要求

所有插入焊盘孔的元件引脚及导线均采用直脚焊，剪脚留头在焊面以上 0.5mm~1mm，未述之处均按常规工艺。

2、电路原理图

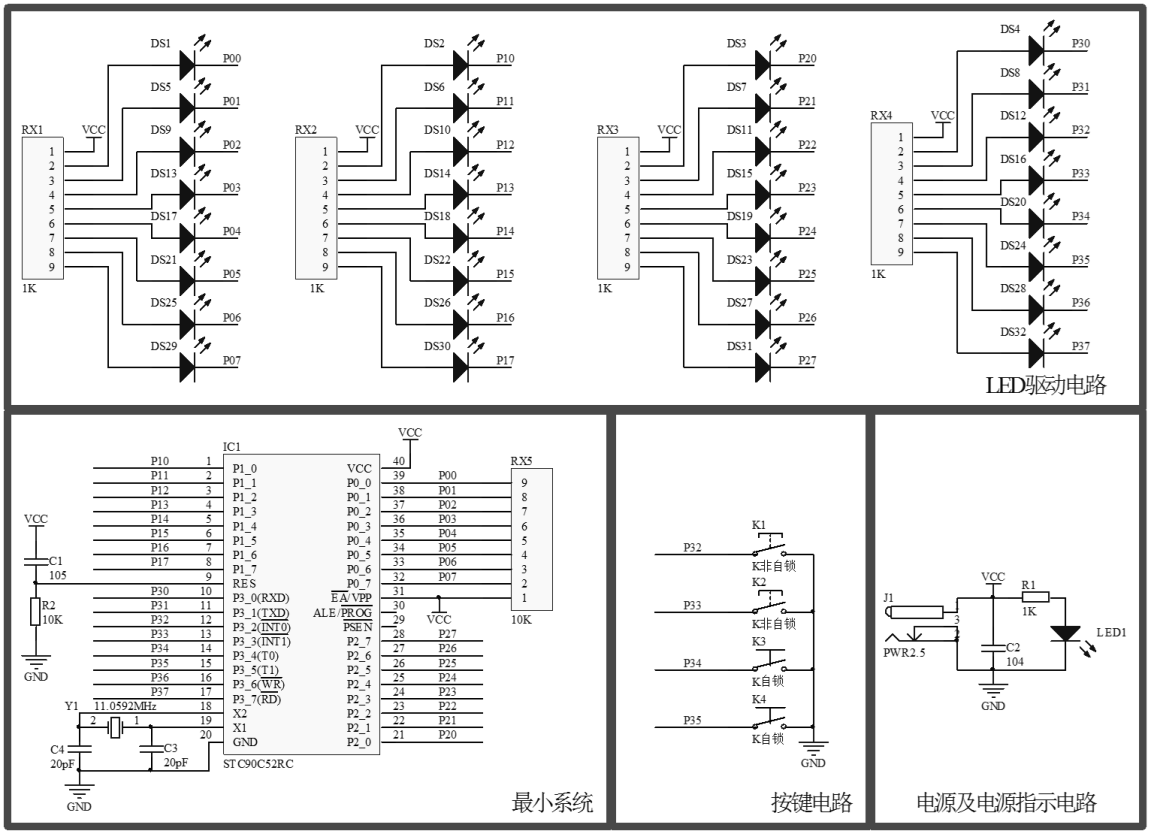

图 1- 40 矩阵广告模块原理图

3、矩阵广告模块 PCB 装配图

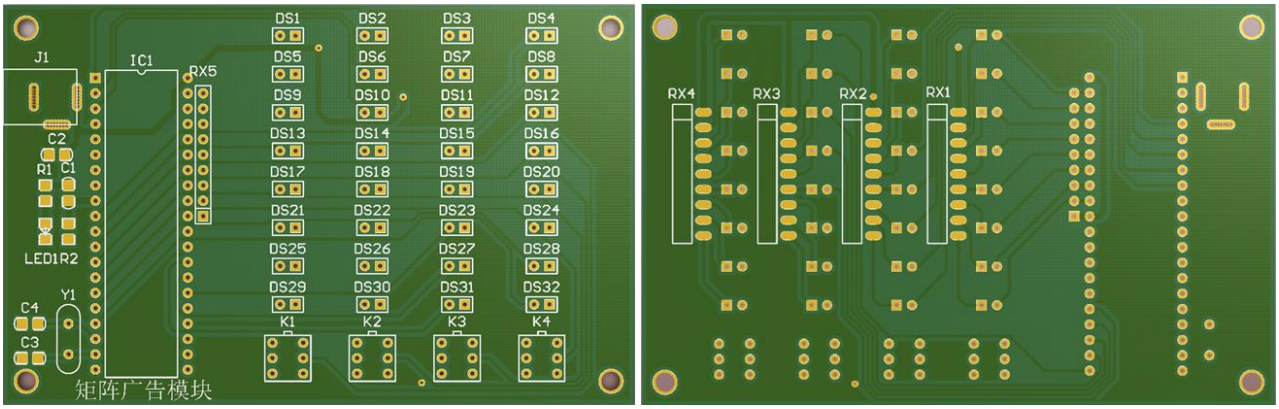

图 1- 41　矩阵广告模块装配图

4、元器件清单

表 1- 3　矩阵广告模块元器件清单

序号	元件名	参数	标号	封装	数量	备注
1.	LED	扁平红	DS1~DS32	SIP2	32	
2.	排阻	1K	RX1~RX4	SIP9	4	
3.	按键	自锁	K1、K2	6 脚按键	2	
4.	按键	非自锁	K3、K4	6 脚按键	2	
5.	电阻	2K	R2	0805C	1	
6.	单片机底座	PDIP40	IC2	DIP40	1	

若使用单片机主机模块进行编程调试，下表中器件不需安装。若直接安装单片机在 PCB 板上，则表 1- 4 中元件必须安装。

表 1- 4　矩阵广告模块非必装配元器件

序号	元件名	参数	标号	封装	数量	备注
7.	电容	104	C4	0805C	1	不焊
8.	电容	105	C5	0805C	1	不焊
9.	电容	20pF	C8, C9	0805C	2	不焊
10.	电阻	1K	R1	0805R	1	不焊
11.	电阻	10K	R2	0805R	1	不焊
12.	排阻	10K	RX1	SIP-9	1	不焊
13.	发光二级管	红色	LED1	LED-0805	1	不焊
14.	晶振	11.0592MHz	Y2	X1	1	不焊
15.	电源接口	PWR2.5	J1	POWER 接口 1	1	不焊

四、任务评价

表 1- 5　任务二完成情况汇总表

项目名称			评价时间		年　月　日		
小组名称		小组成员					
评价内容	评价要求	权重	评价标准	学生自评得分	小组评价得分	教师评价得分	合计
职业与安全意识	1．工具摆放、操作符合安全操作规程；2．遵守纪律、爱惜设备和器材、工位整洁；3．具有团队协作精神。	10%	好（10） 较好（8） 一般（6） 差（<6）				
印制线路板插件	1、插件正确，电阻标志方向一致。 2、元器件高度符合工艺要求，元器件装插平整。	15%	好（15） 较好（12） 一般（9） 差（<9）				
印制线路板焊接	焊点光亮、焊料适量，无虚焊、漏焊、假焊、搭锡、溅锡、铜箔起翘等现象,无毛刺、孔隙留脚长度，焊面以上0.5mm~1mm。	60%	好（60） 较好（45） 一般（30） 差（<30）				
印制线路板的总装	装配正确，无烫伤、划伤工件和导线，紧固件装配牢固可靠，导线剥头尺寸符合工艺要求绝缘恢复良好。	15%	好（15） 较好（12） 一般（9） 差（<6）				
教师签名			学生签名			总分	
项目评价=学生自评（0.2）+小组评价（0.3）+教师评价（0.5）							

五、思考与练习

请写一份装配报告，注明安装中遇到的问题与思考。

任务三　LED 闪烁

一、任务要求

任务说明：

生活中让一盏灯亮起来，合上开关，灯就亮起来了。本任务要求我们尝试使用单片机技术作控制让 LED 闪烁。

任务要求：

这个任务将要求使用单片机控制点亮矩阵广告模块 32 只 LED（发光二极管）的 DS30 亮并闪烁。

二、相关知识

1、硬件知识

要让指定的 LED 灯闪起来，得先了解一下 LED 硬件电路是如何搭建的。LED

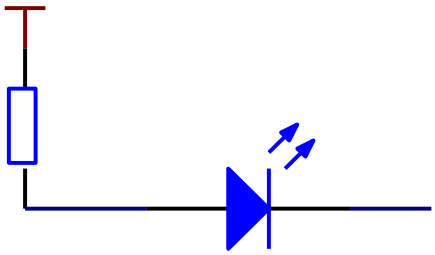

图 1- 42　LED 驱动电路

从上述驱动电路中可以看出，假设想让接在 P1.7 口的 DS30 光二极管亮起来，那么我们只要把 P1.7 口的电平变为低电平就可以了；相反，如果要接在 P1.7 口的 DS30 发光二极管熄灭，就要把 P1.7 口的电平变为高电平就可以。那么如何让单片机 IO 口产生高低电平成了本任务的关键问题。这个事情得交给软件来做。

2、软件知识

从上文表表述看来实现单片机点亮 LED 灯很简单，但是我们不能说 P1.7 你变低，它就变低了。因为单片机听不懂我们的汉语，只能接受二进制的“1、0……”

机器代码。我们又怎样来使单片机按我们的意思去工作呢？为了让单片机工作，只能将程序写为二进制代码交给其执行。早期单片机开发人员就是使用人工编写的二进制代码或汇编语言交给单片机去工作的。今天，我们不必用烦人的二进制去编写程序，完全可以将我们容易理解的“程序语言”通过“翻译”软件“翻译”成单片机所需的二进制代码，然后交给单片机去执行。这里的“程序语言”目前主要有汇编语言和 C 语言两种；在这里我们所说的“翻译”软件，一般都叫它为“编译器”，将“程序语言”通过编译器产生单片机的二进制代码的过程叫编译。接下来让我们了解一下本书中所介绍的 C51 编译软件 KEIL V7.01。

2.1 KEIL 的使用

任务一中已经介绍了 KEIL 的安装与如何在一个项目中生成 HEX 文件。今天的任务就是了解如何使用 KEIL 建立工程并点亮一个 LED 灯。

（1）添加 STC 单片机型号至 KEIL 软件中

在使用 KEIL 软件之前，首先需要通过 STC-ISP 软件，把 STC 单片机型号添加入 KEIL 软件中。具体方法如下：打开 STC-ISP 软件，选择“Keil 仿真设置”子菜单，点击“添加 MCU 型号到 Keil 中”按钮即可。

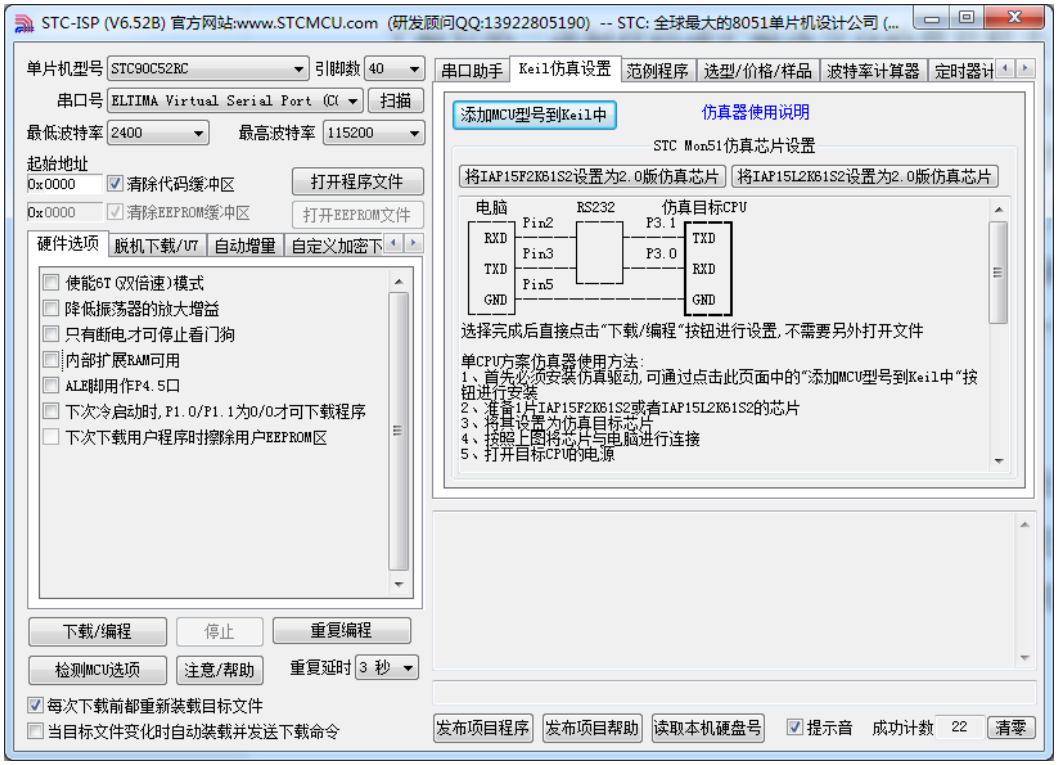

图 1- 43　添加 STC 单片机型号到 KEIL 软件中

（2）在 KEIL 中新建 STC90C52RC 单片机工程

打开 KEIL 软件。点击 Project 菜单下的 New Project 菜单。在弹出窗口输入项目名与项目路径。（注意：尽量为本工程建立一个文件夹以方便管理）本任务为 LED 点亮实验，所以本任务起名为 LED_ON_OFF。

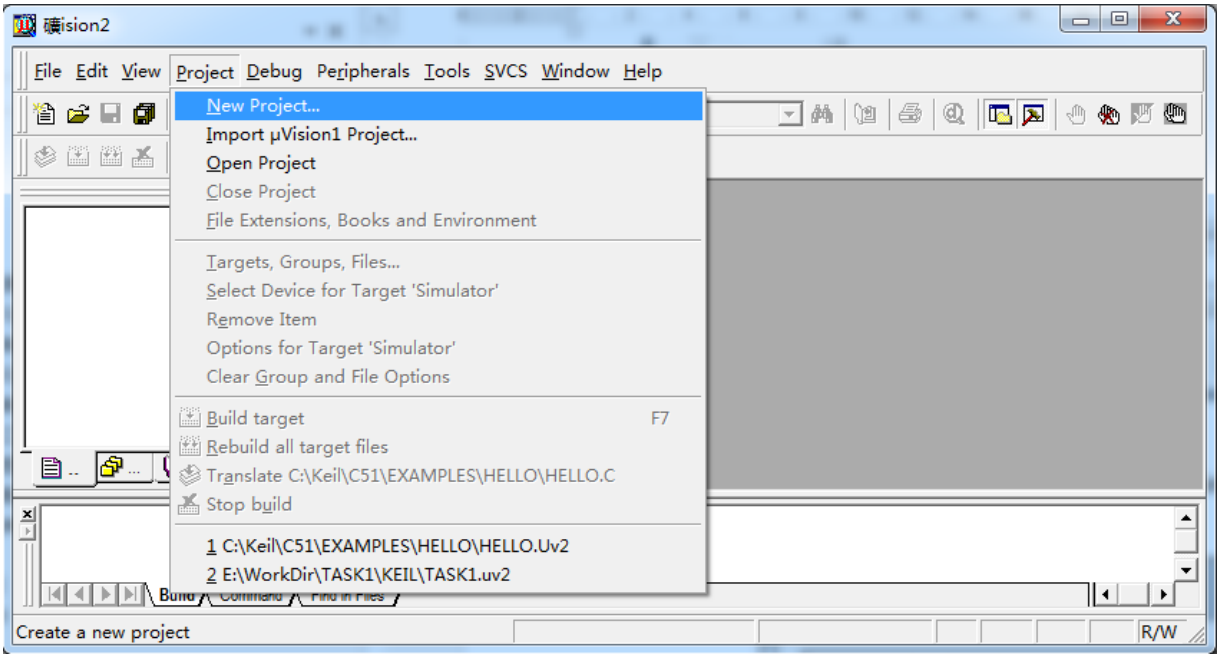

图 1- 44　创建工程

新建项目后会跳出选择 CPU 数据型号对话框，如图 1- 45 所示。选中”STC MCU Database”选项然后点击“确定”按钮。然后再芯片选择对话框中选择本项目使用的单片机芯片“STC90C52RC”如图 1- 46 所示。点击确定之后，这个新的工程就建立好了。

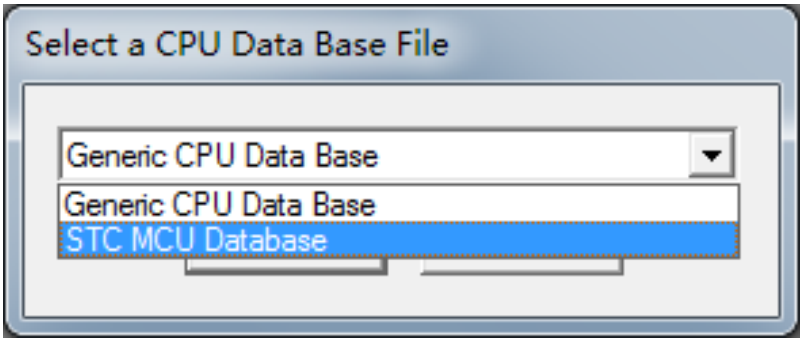

图 1- 45　CUP 数据类型选择

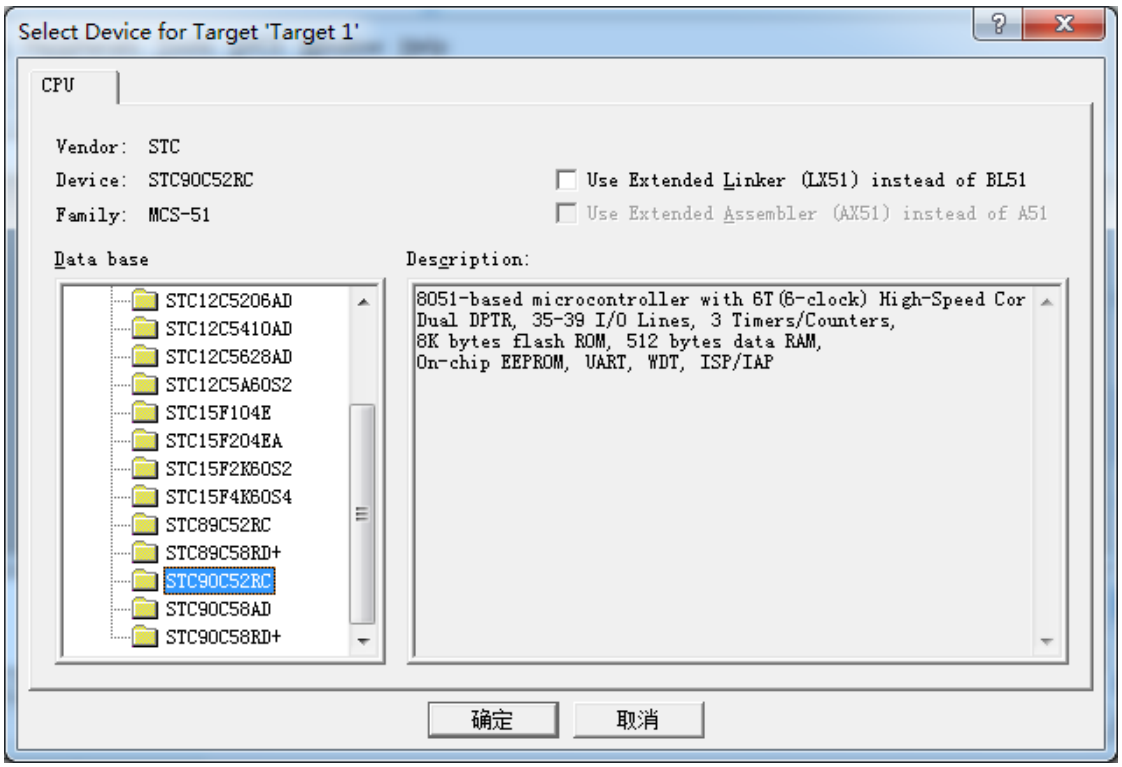

图 1- 46　芯片类型选择

（3）新建文档

点击 File 菜单下的 New 菜单或者直接点击图标建立新文档；点击 File 菜单下的 Save 菜单或者直接点击图标来保存文件；在弹出窗口输入文件名（后缀为.c，本任务为 LED_ON_OFF.C）。

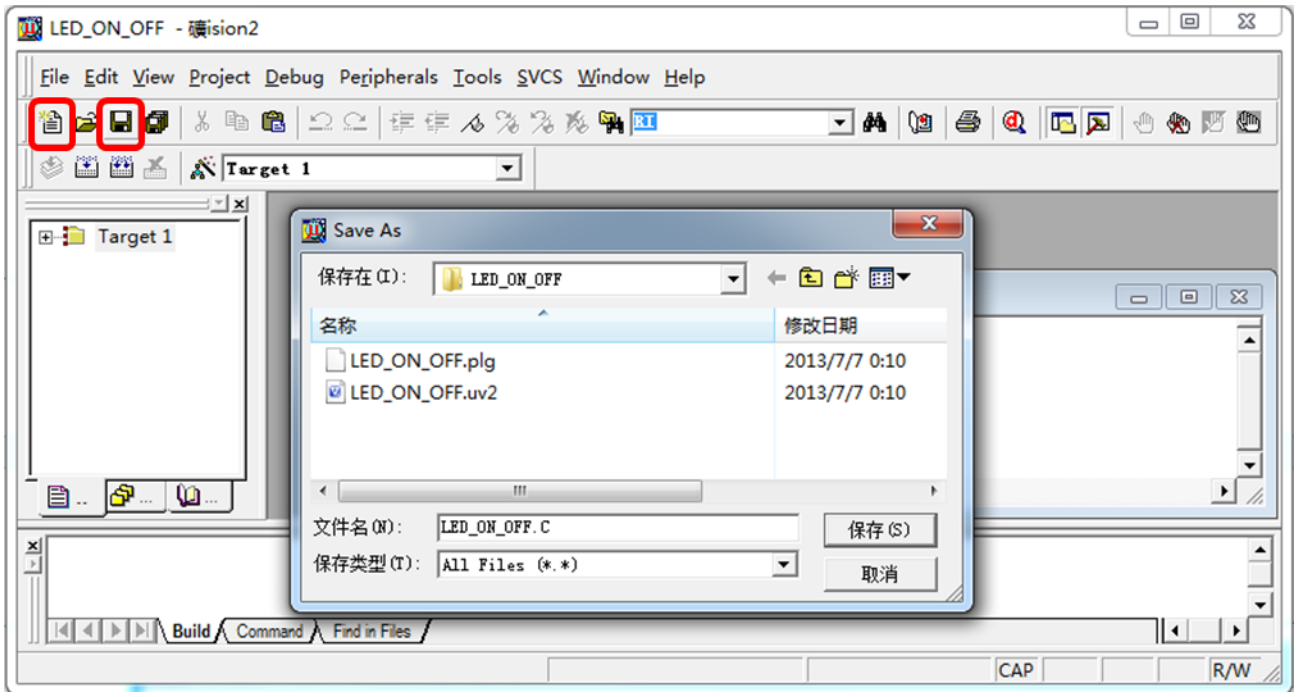

图 1- 47　保存 C 文件

（4）将文档加入至工程

在窗口左侧 Project WorkSpace 窗内右击 Source Group；在弹出菜单中选择 Add File To Group；在弹出窗口选择刚刚您保存的文档。

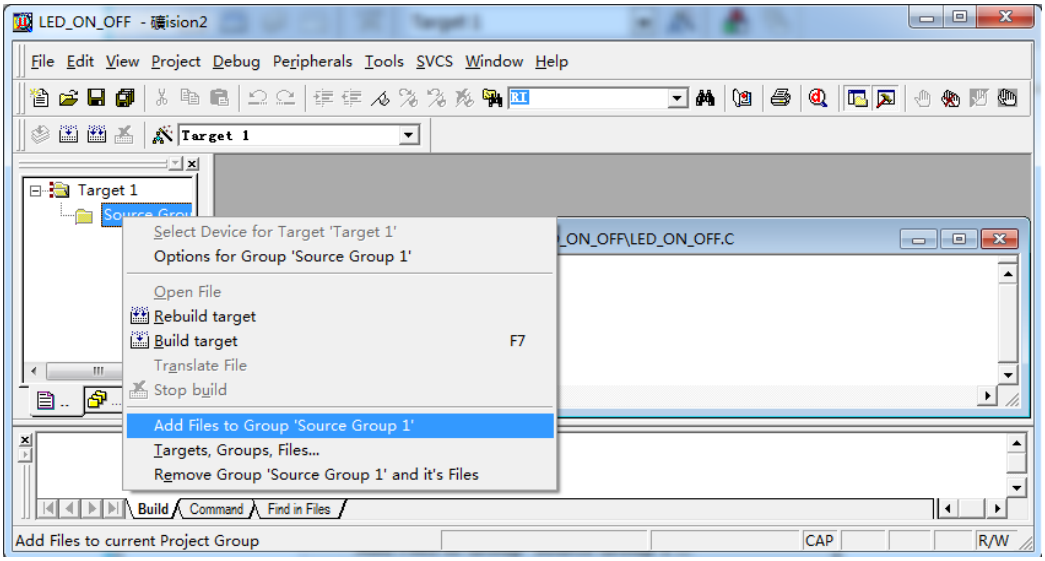

图 1- 48　添加文件到工程

（5）编写程序

在 C 文件编辑窗口（图 1-49）中输入编写的程序。然后点击 编译图标就能执行程序的编译操作了。

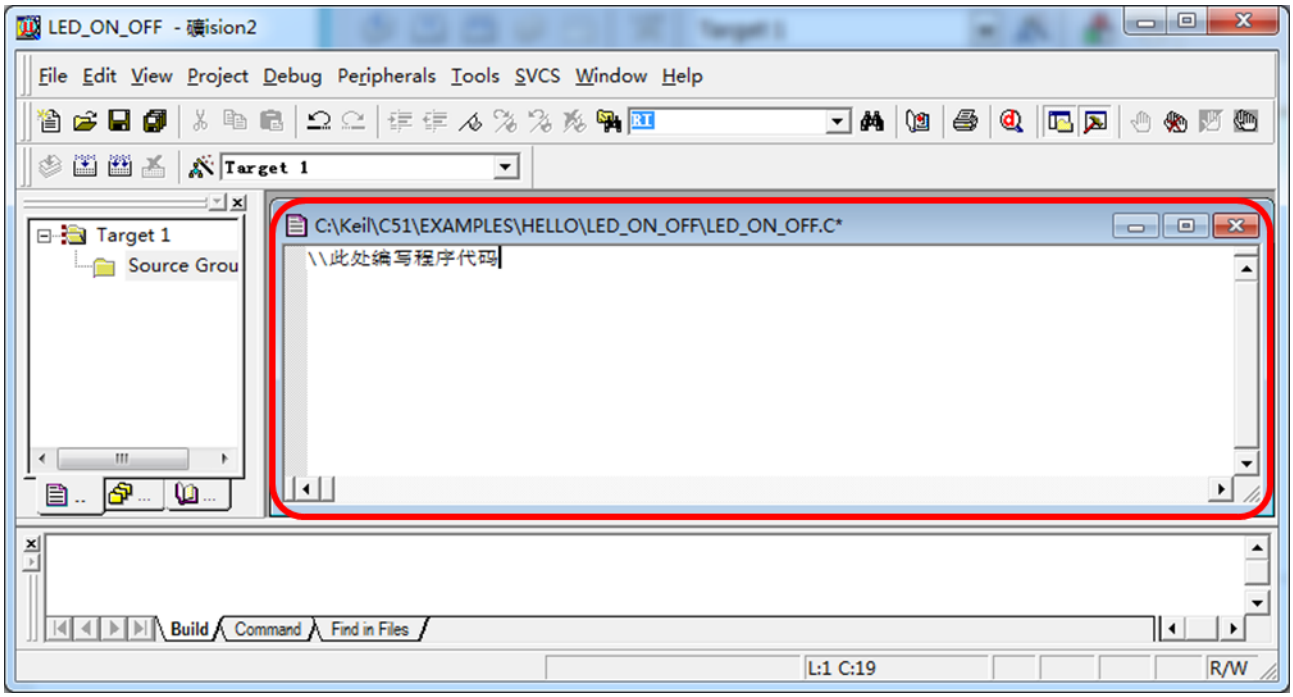

图 1- 49　程序编写位置示意图

（6）设置输出 HEX 文件

点击 Project 菜单下的 Option for Target 菜单或直接点击魔棒图标。在弹出窗口点选 Output 页，在 Create HEX file 前打钩。

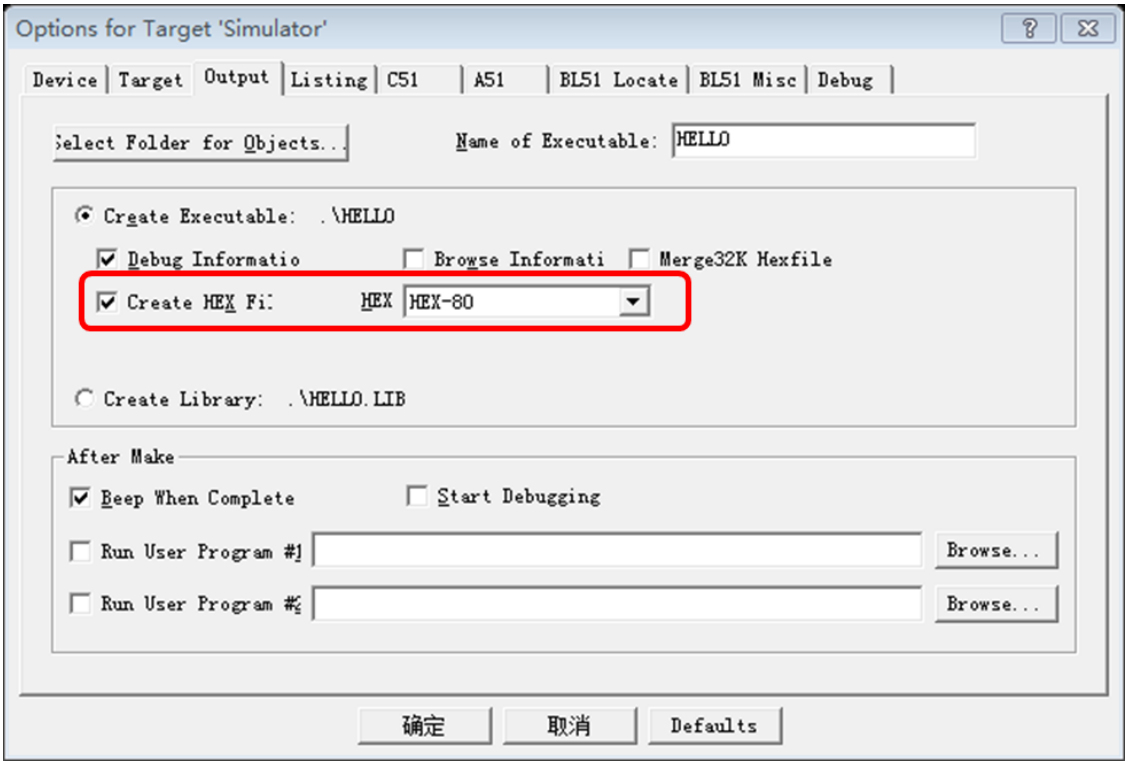

图 1- 50　选择创建 16 进制文件

（7）编译创建 HEX 文件

点击 Project 菜单下的 Build 菜单或直接点击 图标编创建 HEX 文件。至此就能够生成由你输入的程序所翻译成的机器代码（HEX 文件）了。

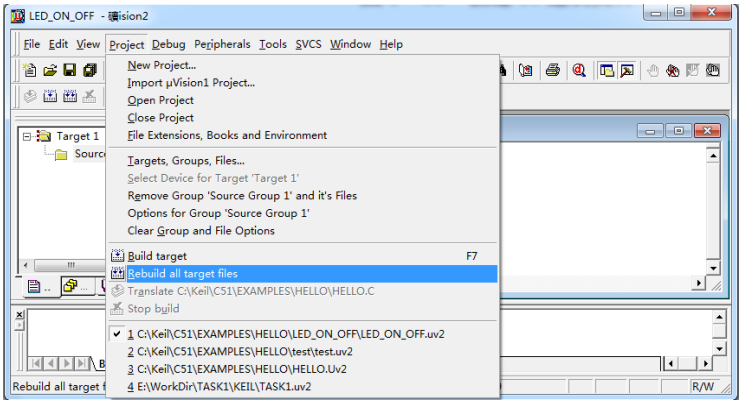

图 1- 51　创建 HEX 文件

那么单片机的语言代码怎么写呢？接着往下看。

2.2　程序编写知识

了解了硬件知识和 KEIL 软件的应用，接下来的任务就是要把我们的要求翻

译成单片机能够明白的指令。通过输入单片机能够了解的指令，控制单片机来按照指令工作。控制单片机的指令有两种，一种是汇编语言，一种是 C 语言。这里我们采用移植性较高的 C 语言来控制单片机。

（1）C 语言程序构架

首先，来了解一下单片机 C 语言程序的构架。图 1-52 所示为单片机 C 语言程序基本构架。

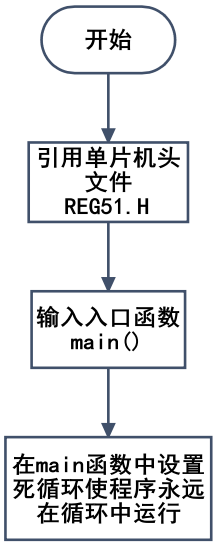

图 1- 52　C 语言构架

单片机 C 语言基本构架

```
#include <reg51.h>//引用 51 单片机头文件
void main(void)//主程序 main 函数
{
/*在主程序中设置死循环程序，保证主程序的运行。*/
    while(1)//while(1)死循环
    {
        /*在此处添加控制程序*/
    }
}
```

控制单片机的程序，不需要循环运行的写在 while(1)之前，需要循环运行的写在 while(1)的大括号中。

（2）IO 口控制

在单片机中，任意一个 IO 端口通过等于符号“=”赋值为 0 就能控制其转变为低电平。同理，赋值为 1 就能控制器转变为高电平。

小知识点：如何控制单片机 IO 口电平

控制单片机的 IO 口，在 C 语言中，很方便。只需要用“=”运算符就行。

“=”运算符：这个运算符是赋值运算符，它的作用是把“=”号右边的值赋给“=”号左边的变量。

比如，我们想让单片机的 P0 口全部置为低电平，只需要写程序 P0=0;就可以了。

【注意】：C51 语法中，在所有的单片机独立语句中，都需要在末尾加上冒号，表示此语句结束。

所以需要控制单片机的某个 IO 口点亮，只需把对应 IO 口通过“=”符号赋值为 0 就行了。但是，由于我们常用的 REG51.H 头文件中没有包含单个 IO 口的位地址定义，所以需要使用 sbit 定义 IO 口的位地址。

单片机 LED 点亮程序

```
#include <reg51.h>//引用 51 单片机头文件
sbit   P1_7=P1^7;//定义 P1.7 口位寻址名称为 P1_7。此名称可以更改。
void main(void)//主程序 main 函数
{
/*在主程序中设置死循环程序，保证主程序的运行。*/
      while(1)//主程序死循环
      {
            /*在此处添加控制程序*/
            P1_7=0;//点亮 P1.0 上的 LED 灯，即 DS30 点亮。
      }
}
```

三、操作训练

1、任务分析

本任务将要求让矩阵广告模块中的一只 LED（发光二极管 DS30）闪烁。矩阵广告模块的 LED 正极经过限流电阻接到了电源的正极；其负极接到了单片机的 IO 口；要让 LED 发光就需要 LED 的负极接地也就是对应的 IO 口输出低电平。以 P1.0 口连接的 LED 为例，要点亮 LED 就需要让单片机的 P1.7 口输出数据 0。

然而要让 LED0 闪烁，就需要用单片机使 LED 高一段时间，低一段时间。这一段时间叫做延时。延时如何获得呢？让单片机不停的做别的工作就行了，51 单片在 11.0592MHZ 晶振下执行一个简单指令约等于 1μs。所以理论上只要编写 1000000 条程序（例如指令 P1_7=0;）就能让单片机运行约 1 秒的时间。但是实际中这样的代码量太大，而且相同的重复指令很可能会被 KEIL 编译器给优化去除。达不到延时 1 秒的目的。那么该怎么办呢？

通过定义一个变量，让它进行减法操作 50000 次，就能让单片机工作一段时间。这个时间，就是延时。

单片机延时程序

```
unsigned int i;
i=50000;
while(i--);//i 每次减一，直至减到 0 为止，经过 50000 次减法操作后，单片机延时一段时间。
```

知识点：“while”循环指令：这个指令是常用的判断循环指令。我们这里只使用 while(1){程序内容}的形式，让程序永远在 while(1)所包含的大括号之中循环运行。

while 一般有两种形式

形式 1：While(判断条件){执行语句}，先进行判断，而运行执行语句。执行语句运行完毕，自动返回继续判断 while 中的条件是否符合，符合的话，继续运行执行语句，不符合，则退出循环。

形式 2：do{执行语句}While(判断条件)，执行效果是先运行执行语句，再进行 while 条件判断，如果符合条件，则返回继续执行 do 后的执行语句，由此形成循环。

PS:while(1){}表示判断条件一直为 1（C 语言中 1 为真）所以一直在这个循环中重复运行。

2、主函数流程图

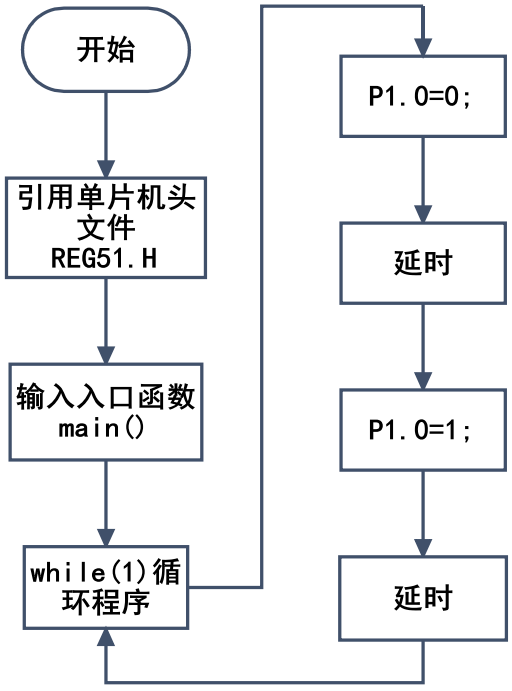

图 1- 53　任务三主程序流程图

3、参考程序

LED 闪烁参考程序 LED_ON_OFF.C

```
#include <reg51.h>//引用 51 单片机头文件
sbit   P1_7=P1^7;//定义 P1.7 口位寻址名称为 P1_7。此名称可以更改。
void main(void)//主程序 main 函数
{
      unsigned int i;
/*在主程序中设置死循环程序，保证主程序的运行。*/
      while(1)//主程序死循环
      {
            /*在此处添加控制程序*/
            P1_7=0;//点亮 P1.7 上的 LED 灯，即 DS30 点亮。
            i=50000;while(i--);//延时一段时间
            P1_7=1;//使 P1.7 上 LED 灯 DS30 熄灭。
            i=50000;while(i--);//延时一段时间
      }
}
```

4、任务实施

4.1 建立工程

打开 keil 软件，通过菜单“Project”，新建立一个工程”new Project”项目 LED_ON_OFF，然后再新建一个文件名为 LED_ON_OFF.C 的源程序文件，将上面的参考程序输入并保存。然后将 LED_ON_OFF.C 文件添加入本项目中。

4.2 编译并生成 HEX

单击”Build target”按钮，对源程序进行编译和链接，产生 HEX 文件。

4.3 烧录芯片

打开 STC-ISP 下载软件，选中单片机型号为“STC90C52RC”,选择最低波特率为 2400 最高波特率为 115200（为默认值一般不需修改）。点击打开程序文件按钮，在刚才建立的 LED_ON_OFF 工程文件夹中生成的 LED_ON_OFF.HEX 文件。然后点击“下载/编程”按钮。最后使用串口线连接 PC 与主机模块，然后给主机模

块通电，等待电脑端串口与单片机握手成功。就能把程序载入到单片机中去了。

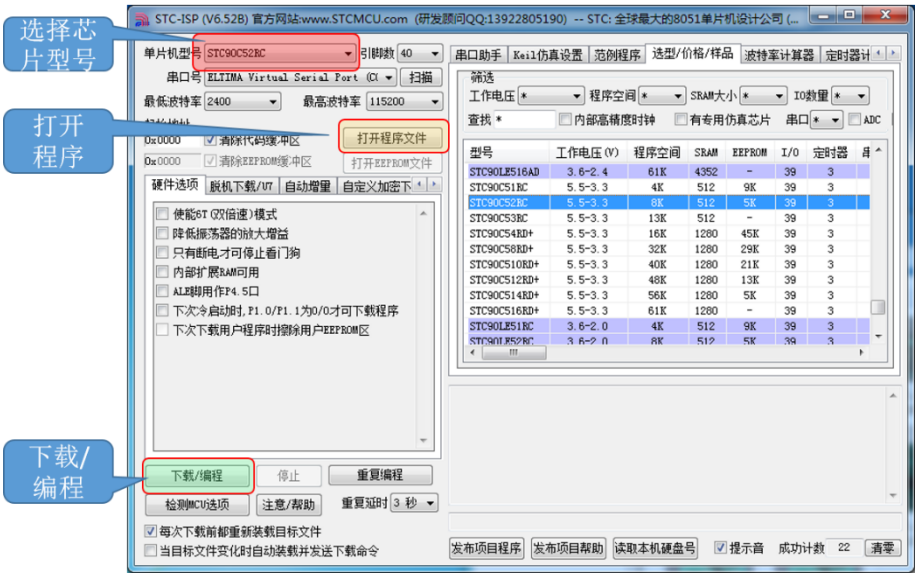

图 1- 54　STC 单片机程序烧写示意图

4.4　硬件调试

程序下载完成后，把单片机最小系统接入电路中，调试电路使其能正常工作，使 LED 闪烁。

四、任务评价

表 1- 6　任务三完成情况评价表

项目名称			评价时间		年　月　日		
小组名称		小组成员					
评价内容	评价要求	权重	评价标准	学生自评得分	小组评价得分	教师评价得分	合计
职业与安全意识	1．工具摆放、操作符合安全操作规程；2．遵守纪律、爱惜设备和器材、工位整洁；3．具有团队协作精神。	10%	好（10） 较好（8） 一般（6） 差（<6）				
模块的布局和布线工艺	1．模块布局合理，模块的选择应符合模块的要求。	15%	好（15） 较好（12） 一般（9） 差（<9）				
项目功能测试	1．编写的程序能成功编译；2．程序能正确烧写到芯片中；3．指定 LED 闪烁。	60%	好（60） 较好（45） 一般（30） 差（<30）				
问题与思考	不用#includ“reg51.h”，而用 sfr P1= 0x90;代替可以不可以	15%	好（15） 较好（12） 一般（9） 差（<6）				
教师签名			学生签名			总分	
项目评价=学生自评（0.2）+小组评价（0.3）+教师评价（0.5）							

五、知识拓展

1、预处理

在编译环境对源程序进行编译前，先对程序中的预处理命令进行处理，然后将处理结果和源程序一起进行编译。预处理命令通常只进行一些符号的处理，其并不执行具体的单片机硬件操作。C51 语言中的预处理命令包括文件包含指令、宏定义指令和条件编译指令等，还有其他一些指令在程序调试时使用。C51 语言中提供了各种预处理命令，其作用类似于汇编程序中的伪指令。一般来说，在对 C51 源程序进行编译前，编译器需要先对程序中的预处理命令进行处理，然后将预处理的结果和源代码一并进行编译，最后产生目标代码。预处理命令通常只进行一些符号的处理，其并不执行具体的硬件操作。为了与 C51 源代码中的程序语句相区别，预处理命令前要加一个“#”。C51 语言中的预处理命令如表 1-7 所示。

表 1- 7　C51 的预处理命令

预处理命令	用途
#define	用于宏定义
#error	用于程序调试
#include	用于文件包含
#if	用于条件编译
#else	用于条件编译
#elif	用于多种条件编译选择
#endif	用于条件编译
#ifdef	用于条件编译
#ifndef	用于条件编译
#undef	用于宏定义
#line	用于更改行号
#pragma	用于传送控制指令

1.1　文件包含指令

文件包含指令，即#include 命令，通常位于 C51 源程序的开头，利用#include

命令可以将其他的文件引入当前的 C51 源文件。其中被包含的文件通常是头文件、宏定义等。使用文件包含指令，有利于更好地调试 C51 源文件。当需要调试修改文件时，只要修改某一包含文件即可，而无需对所有文件进行修改。

在 C51 语言中，文件包含指令的一般形式如下：

#include "头文件.h"

#include <头文件.h>

#include 宏定义标识符

其中，“#include”表示文件包含指令、双引号或尖括号括起来的文件名是要引入的源文件。典型的文件包含指令示例如下：

```
#include "myfile.h"                                    //引用自定义文件 myfile
#include <studio.h>                                    //引用库函数文件 studio
#include <reg51.h>                                     //引用寄存器文件
#define MATH_FILE "C\keil\inc\math1.h"//宏定义自定义文件 MATH_FILE
#include MATH_FILE                                     //引用自定义文件 MATH_FILE
```

在 Keil 集成开发环境中，C51 标准库提供了许多包含文件，即 C51 的头文件。这些文件存放在目录 Keil\C51\INC 文件夹及其子目录下。这些头文件包含常数、宏定义、类型定义和函数原型等。

1.2 宏定义

宏定义指令是指用一些标识符作为宏名，来代替其他一些符号或者常量的预处理命令。使用宏定义指令，可以减少程序中字符串输入的工作量，而且可以提高程序的可移植性。

宏名既可以是字符串或常数，也可以是带参数的宏。宏定义指令可分为带参数的宏定义和不带参数的宏定义。下面分别介绍用于宏定义的一些预处理命令。

#define 命令用于定义一个宏名。宏名是一个标识符，在源代码中遇到该标识符时，均以宏定义的串的内容代替该标识符。ANSI 标准宏将定义的标识符称为“宏名”，而用定义的内容代替宏名的过程称为“宏替换”。#define 命令用于定义宏名时，既可以带参数，也可以不带参数，下面分别介绍这两种情况。

（1）不带参数的宏定义

不带参数的宏定义一般形式为: ＃define 标识符 字符串。

例如：＃ define PI 3.1415926

宏定义的作用是在本程序文件中用指定的标识符 PI 来代替“3.1415926”这个字符串，在编译预处理时，将程序中在该命令以后出现的所有的 PI 都用“3.1415926”代替。这种方法使用户能以一个简单的名字代替一个长的字符串。这个标识符（名字）称为“宏名”在预编译时将宏名替换成字符串的过程称为“宏展开”。

（2）带参数的宏定义

带参数的宏定义一般形式为:＃define 宏名（参数表）字符串。字符串中包含在括弧中所指定的参数例如

#define S(a,b) a*b

对带实参的宏（如 S（3,2)），按＃define 命令行中指定的字符串从左到右进行置换。若串中包含宏中的形参（如 a、b)，则将程序中相应的实参（可以是常量、变量或表达式代替形参。如果宏定义中的字符串中的字符不是参数字符（如 a * b 中的*号)，则保留。这样就形成了置换的字符串。

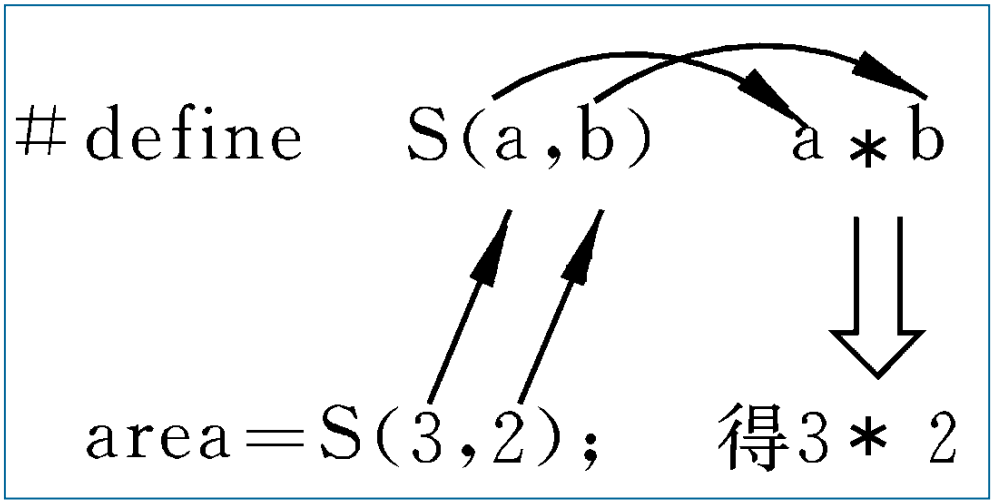

图 1- 55 宏替换

2、标识符和关键字

标识符和关键字是一种编程语言最基本的组成部分，C51 语言同样支持自定义的标识符以及系统保留的关键字。在进行 C51 程序设计时，需要了解标识符

和关键字的使用规则。

- 标识符常用来声明某个对象的名称，如变量和常量的声明、数组和结构的声明、自定义函数的声明以及数据类型的声明等。
- 在 C51 语言中，标识符可以由字母、数字（0～9）和下划线“_”组成，最多可支持 32 个字符。并且，C51 标识符第一个字符必须是字母或者下划线“_”。例如“ut1”、“ch_1”等，都是正确的。而“5count”则是错误的标识符。另外，C51 的标识符区分大小写，例如“count1”和“COUNT1”代表两个不同的标识符。
- 关键字是 C51 语言重要的组成部分，是 C51 编译器已定义保留的专用特殊标识符，有时也称为保留字。这些关键字通常有固定的名称和功能，如 int、float、if、for、do、while、case 等。

本程序主要就是访问 P1.7；语句“sbit LED7=P1^7;”用 sbit 去定义 P1 口的最高位，而 P1 的定义在头文件 reg51.h 有定义——“sfr P1 = 0x90;”（要在自己写的文件中使用到 reg51.h 文件里的内容，可以使用预处理命令：#include “reg51.h”）,这样我们就可以在主函数 main（）中直接使用 P17 这个标识符了既：“P17=0;”，这样接在 P1.7 的 LED 就能发光了。

上文所提到的“sbit”和“sfr”叫做关键字——是 C51 保留的特殊标识符，有时又称为保留字，它们具有固定名称和含义。C51 还有其他的关键字在稍后的项目和任务中将向您介绍。“P1”和“P17”叫做标识符——用来标识变量名、符号常量名、函数名、数组名、类型名等的有效字符序列，标识符的命名要满足以下规则：1）一个标识符由字符串、数字和下划线等组成；2）第一个字符必须是字母或下划线；（通常以下划线开头的标识符是编译系统专用的，因此在编写 C 语言源程序时一般不要使用以下划线开头的标识符，而将下划线用作分段符）。3）标识符的长度不要超过 32 个字符。（C51 编译器规定标识符最长可达 255 个字符，但只有前面 32 个字符在编译时有效，因此在编写源程序时标识符的长度不要超过 32 个字符。）4）C 语言是大小写英文字母是不同。（如 sec（秒）和 SEC 是两个完全不同的标识符。）5）关键字（保留字）不能作为标识符。例：以下是合法的标识符：atep1, Delay_1_ms, scan_twice, u2_wait_for_me, a1；以下是不合法的标识符：2_w： 只能以英文字母或下划线开头，wait!：!不能使用，

sfr 保留字不能作为标识符。

3、sbit 定义特殊功能寄存器的位变量

sbit 的作用是定义特殊功能寄存器的位变量。bit 和 sbit 都是 C51 扩展的变量类型。其区别为：bit 是定义普通位寻址区的位变量的，而 sbit 是定义特殊功能寄存器（sfr）中位变量的。

典型应用是：sbit P0_0=P0^0;//即定义 P0_0 为 P0 口的第 1 位，以便进行位操作。

在 C 语言里，如果直接写 P1.0，C 编译器并不能识别，而且 P1.0 也不是一个合法的 C 语言变量名，所以得给它另起一个名字，这里起的名为 P1_0，可是 P1_0 是不是就是 P1.0 呢？你这么认为，C 编译器可不这么认为，所以必须给它们建立联系，这里使用了 Keil C 的关键字 sbit 来定义，sbit 的用法有三种：

第一种方法：sbit 位变量名=地址值

第二种方法：sbit 位变量名=SFR 名称^变量位地址值

第三种方法：sbit 位变量名=SFR 地址值^变量位地址值

如定义 PSW 中的 OV 可以用以下三种方法：

sbit OV=0xd2　　（1）说明：0xd2 是 OV 的位地址值

sbit OV=PSW^2　　（2）说明：其中 PSW 必须先用 sfr 定义好

sbit OV=0xD0^2　　（3）说明：0xD0 就是 PSW 的地址值

因此这里用 sbit P1_0=P1^0;就是定义用符号 P1_0 来表示 P1.0 引脚，如果愿意也可以起 P10 一类的名字，只要下面程序中也随之更改就行了。

【注意】：sbit 只能再 c 语言结构的最外面定义，不能定义在子程序中。

4、编译错误

初学者在刚开始学习编程时经常会遇到编译不能通过的问题。如图 1-56 所示。这主要是因为程序中有语法错误。下面把日常会遇到的一些错误罗列了出来，供刚刚开始学习单片机的初学者参考。初学者尤其要注意的是少些分号或括号的问题。

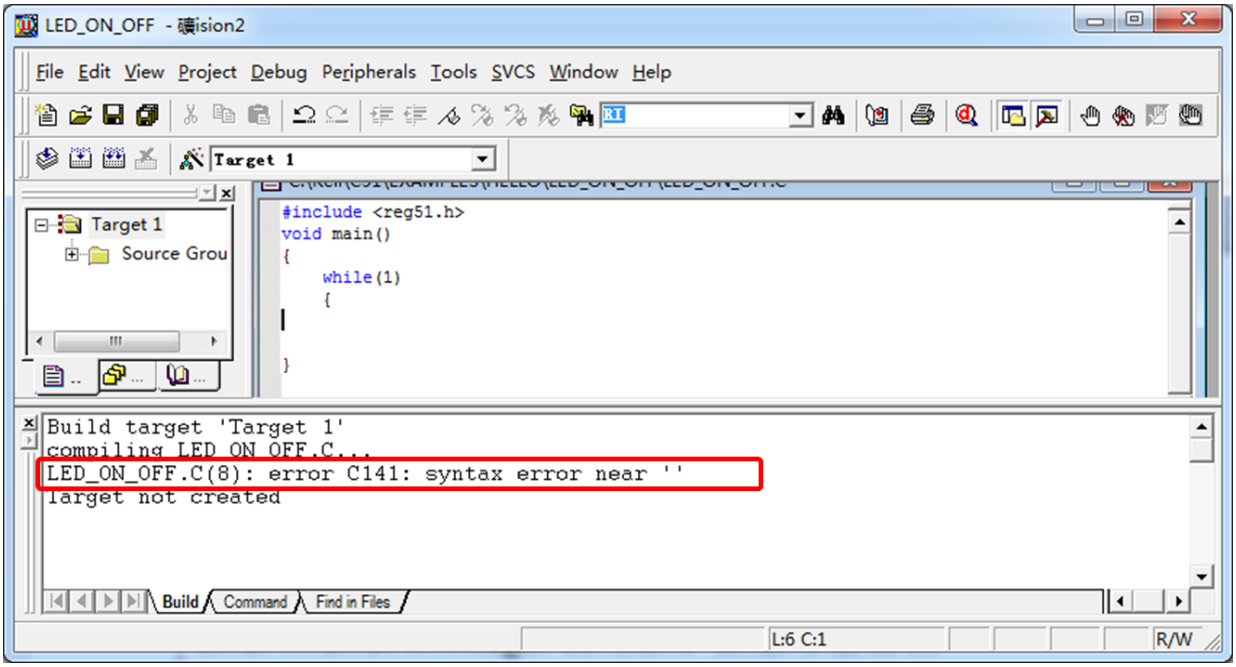

图 1- 56　编译错误演示

KEIL 编译错误
错误代码及错误信息　错误释义 error 1: Out of memory　内存溢出 error 2: Identifier expected　缺标识符 error 3: Unknown identifier　未定义的标识符 error 4: Duplicate identifier　重复定义的标识符 error 5: Syntax error　语法错误 error 6: Error in real constant　实型常量错误 error 7: Error in integer constant　整型常量错误 error 8: String constant exceeds line　字符串常量超过一行 error 10: Unexpected end of file　文件非正常结束 error 11: Line too long　行太长 error 12: Type identifier expected　未定义的类型标识符 error 13: Too many open files　打开文件太多 error 14: Invalid file name　无效的文件名 error 15: File not found　文件未找到 error 16: Disk full　磁盘满 error 17: Invalid compiler directive　无效的编译命令 error 18: Too many files　文件太多 error 19: Undefined type in pointer def　指针定义中未定义类型 error 20: Variable identifier expected　缺变量标识符 error 21: Error in type　类型错误 error 22: Structure too large　结构类型太长

KEIL 编译错误

error 23: Set base type out of range 集合基类型越界
error 24: File components may not be files or objectsfile 分量不能是文件或对象
error 25: Invalid string length 无效的字符串长度
error 26: Type mismatch 类型不匹配
error 27：error 27：Invalid subrange base type 无效的子界基类型
error 28：Lower bound greater than upper bound 下界超过上界
error 29：Ordinal type expected 缺有序类型
error 30：Integer constant expected 缺整型常量
error 31：Constant expected 缺常量
error 32：Integer or real constant expected 缺整型或实型常量
error 33：Pointer Type identifier expected 缺指针类型标识符
error 34：Invalid function result type 无效的函数结果类型
error 35：Label identifier expected 缺标号标识符
error 36：BEGIN expected 缺 BEGIN
error 37：END expected 缺 END
error 38：Integer expression expected 缺整型表达式
error 39：Ordinal expression expected 缺有序类型表达式
error 40：Boolean expression expected 缺布尔表达式
error 41：Operand types do not match 操作数类型不匹配
error 42：Error in expression 表达式错误
error 43：Illegal assignment 非法赋值
error 44：Field identifier expected 缺域标识符
error 45：Object file too large 目标文件太大
error 46：Undefined external 未定义的外部过程与函数
error 47：Invalid object file record 无效的 OBJ 文件格式
error 48：Code segment too large 代码段太长
error 49：Data segment too large 数据段太长
error 50：DO expected 缺 DO
error 51：Invalid PUBLIC definition 无效的 PUBLIC 定义
error 52：Invalid EXTRN definition 无效的 EXTRN 定义
error 53: Too many EXTRN definitions 太多的 EXTRN 定义
error 54：OF expected 缺 OF
error 55：INTERFACE expected 缺 INTERFACE
error 56：Invalid relocatable reference 无效的可重定位引用
error 57：THEN expected 缺 THEN
error 58：TO or DOWNTO expected 缺 TO 或 DOWNTO

KEIL 编译错误

error 59：Undefined forward 提前引用未经定义的说明
error 61：Invalid typecast 无效的类型转换
error 62：Division by zero 被零除
error 63：Invalid file type 无效的文件类型
error 64：Cannot read or write variables of this type 不能读写此类型变量
error 65：Pointer variable expected 缺指针类型变量
error 66：String variable expected 缺字符串变量
error 67：String expression expected 缺字符串表达式
error 68：Circular unit reference 单元 UNIT 部件循环引用
error 69：Unit name mismatch 单元名不匹配
error 70：Unit version mismatch 单元版本不匹配
error 71：Internal stack overflow 内部堆栈溢出
error 72：Unit file format error 单元文件格式错误
error 73：IMPLEMENTATION expected 缺 IMPLEMENTATION
error 74：Constant and case types do not match 常量和 CASE 类型不匹配
error 75：Record or object variable expected 缺记录或对象变量
error 76：Constant out of range 常量越界
error 77：File variable expected 缺文件变量
error 78：Pointer expression expected 缺指针表达式
error 79：Integer or real expression expected 缺整型或实型表达式
error 80：Label not within current block 标号不在当前块内
error 81：Label already defined 标号已定义
error 82：Undefined label in preceding statement part 在前面未定义标号
error 83：Invalid @ argument 无效的@参数
error 84：UNIT expected 缺 UNIT
error 85: ";" expected 缺“;”
error 86： ":" expected 缺“:”
error 87： "," expected 缺“,”
error 88： "(" expected 缺“(”
error 89： ")" expected 缺“)”
error 90： "=" expected 缺“=”
error 91： ":=" expected 缺“:=”
error 92： "[" or "(." Expected 缺“[”或“(.”
error 93: "]" or ".)" expected 缺“]”或“.)”
error 94： "." expected 缺“.”
error 95: ".." expected 缺“..”

KEIL 编译错误

error 96：Too many variables 变量太多
error 97：Invalid FOR control variable 无效的 FOR 循环控制变量
error 98：Integer variable expected 缺整型变量
error 99：Files and procedure types are not allowed here 该处不允许文件和过程类型
error 100：String length mismatch 字符串长度不匹配
error 101：Invalid ordering of fields 无效域顺序
error 102：String constant expected 缺字符串常量
error 103：Integer or real variable expected 缺整型或实型变量
error 104：Ordinal variable expected 缺有序类型变量
error 105：INLINE error INLINE 错误
error 106：Character expression expected 缺字符表达式
error 107：Too many relocation items 重定位项太多
error 108：Overflow in arithmetic operation 算术运算溢出
error 112：CASE constant out of range CASE 常量越界
error 113：Error in statement 表达式错误
error 114：Cannot call an interrupt procedure 不能调用中断过程
error 116：Must be in 8087 mode to compile this 必须在 8087 模式编译
error 117：Target address not found 找不到目标地址
error 118：Include files are not allowed here 该处不允许 INCLUDE 文件
error 119：No inherited methods are accessible here 该处继承方法不可访问
error 121：Invalid qualifier 无效的限定符
error 122：Invalid variable reference 无效的变量引用
error 123：Too many symbols 符号太多
error 124：Statement part too large 语句体太长
error 126：Files must be var parameters 文件必须是变量形参
error 127：Too many conditional symbols 条件符号太多
error 128：Misplaced conditional directive 条件指令错位
error 129：ENDIF directive missing 缺 ENDIF 指令
error 130：Error in initial conditional defines 初始条件定义错误
error 131：Header does not match previous definition 和前面定义的过程或函数不匹配
error 133：Cannot evaluate this expression 不能计算该表达式
error 134：Expression incorrectly terminated 表达式错误结束
error 135：Invalid format specifier 无效格式说明符
error 136：Invalid indirect reference 无效的间接引用

KEIL 编译错误

error 137：Structured variables are not allowed here 该处不允许结构变量
error 138：Cannot evaluate without System unit 没有 System 单元不能计算
error 139：Cannot access this symbol 不能存取符号
error 140：Invalid floating point operation 无效的符号运算
error 141：Cannot compile overlays to memory 不能编译覆盖模块至内存
error 142：Pointer or procedural variable expected 缺指针或过程变量
error 143：Invalid procedure or function reference 无效的过程或函数调用
error 144：Cannot overlay this unit 不能覆盖该单元
error 146：File access denied 不允许文件访问
error 147：Object type expected 缺对象类型
error 148：Local object types are not allowed 不允许局部对象类型
error 149：VIRTUAL expected 缺 VIRTUAL
error 150: Method identifier expected 缺方法标识符
error 151：Virtual constructors are not allowed 不允许虚构造函数
error 152：Constructor identifier expected 缺构造函数标识符
error 153：Destructor identifier expected 缺析构函数标识符
error 154：Fail only allowed within constructors 只能在构造函数内使用 Fail 标准过程
error 155：Invalid combination of opcode and operands 操作数与操作符无效组合
error 156：Memory reference expected 缺内存引用指针
error 157：Cannot add or subtract relocatable symbols 不能加减可重定位符号
error 158：Invalid register combination 无效寄存器组合
error 159：286/287 instructions are not enabled 未激活 286/287 指令
error 160：Invalid symbol reference 无效符号指针
error 161：Code generation error 代码生成错误
error 162：ASM expected 缺 ASM
error 166：Procedure or function identifier expected 缺过程或函数标识符
error 167：Cannot export this symbol 不能输出该符号
error 168：Duplicate export name 外部文件名重复
error 169：Executable file header too large 可执行文件头太长
error 170：Too many segments 段太多

六、思考与练习

1、简述 C51 程序的一般组成？

2、练习编程点亮 32 只 LED 中的任意一只或几只。

任务四　手动控制 LED 的亮灭

一、任务要求

使用矩阵广告模块中设置的按键控制 LED 的亮灭，任务分两个层次，第一层次使用带自锁的按键，按键按下时 LED 发光，按键弹开时 LED 熄灭。第二层次使用无自锁按键，每有一次按键动作 LED 的状态改变一次。

二、相关知识

1、硬件知识

矩阵广告模块中按键的连接如图 1- 57 所示，P3 口的 P3.2、P3.3、P3.4 和 P3.5 口接有 4 只按键（K1、K2、K3、K4），其中 P3.2、P3.3 口接有非自锁按键，P3.4 和 P3.5 口接自锁按键，按键按下 IO 口接地，数据值为“0”，弹开时芯片内部上拉接电源，数据值为“1”。

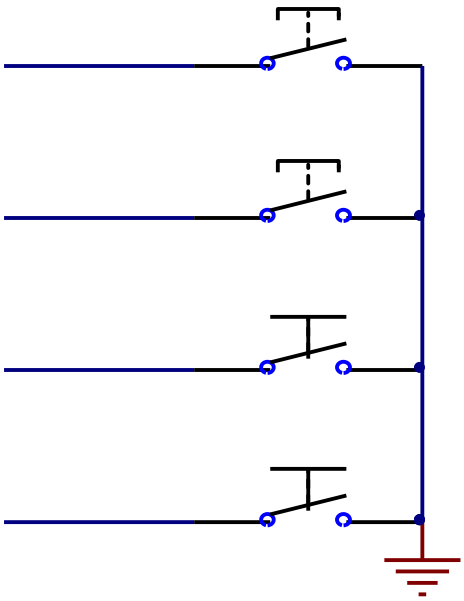

图 1- 57

2、软件防抖动

通常的按键所用开关为机械弹性开关，当机械触点断开、闭合时，由于机

械触点的弹性作用，一个按键开关在闭合时不会马上稳定地接通，在断开时也不会一下子断开。因而在闭合及断开的瞬间均伴随有一连串的抖动，为了不产生这种现象而作的措施就是按键消抖。抖动时间的长短由按键的机械特性决定，一般为 5ms～10ms。这是一个很重要的时间参数，在很多场合都要用到。

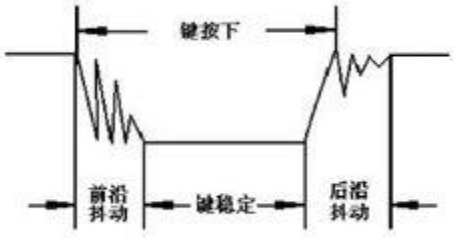

图 1- 58　按键抖动信号

按键稳定闭合时间的长短则是由操作人员的按键动作决定的，一般为零点几秒至数秒。键抖动会引起一次按键被误读多次。为确保单片机对键的一次闭合仅作一次处理，必须去除键抖动。在键闭合稳定时读取键的状态，并且必须判别到键释放稳定后再作处理。

按键按下时为数据值为“0”，弹开时数据值为“1”。如此我们分析完整的编程方案可以如下：判断按键是否有变成“0”即前沿抖动，然后软件延时 10ms，这时再次判断按键是否为“0”，是“0”则认定按键按下进入稳定，接着判断按键是否有变成“1”即后沿抖动，然后软件延时 10ms，这时再次判断按键是否为“1”，是“1”则认定完成一次按键动作。参考代码如下：

按键动作判别子程序

```
sbit BT=P3^4;
bit S=1;//按键稳态标记
void delay()//延时 10ms
{
    unsigned char i=50,j=100;
    while(i--)
    {
        j=100;
        while(j--);
    }
}
unsigned char PushB()//按键检测程序，有按键操作函数将返回 1，否则返回 0
```

```
{
    if(S)//按键稳定状态判断
    {
        if(BT==0)//按键按下判断
        {
            delay();//防按下抖动
            if(BT==0)S=0;
            else S=1;
        }
        return 0;
    }
    else
    {
         if(BT==1)按键弹开
        {
            S=1;
            delay();
            if(BT==1)return 1;
        }
        return 0;
    }
}
```

判断按键的状态我们使用了 if 语句，if 语句是选择结构常用的语句。If 选择结构如图 1- 59 所示。

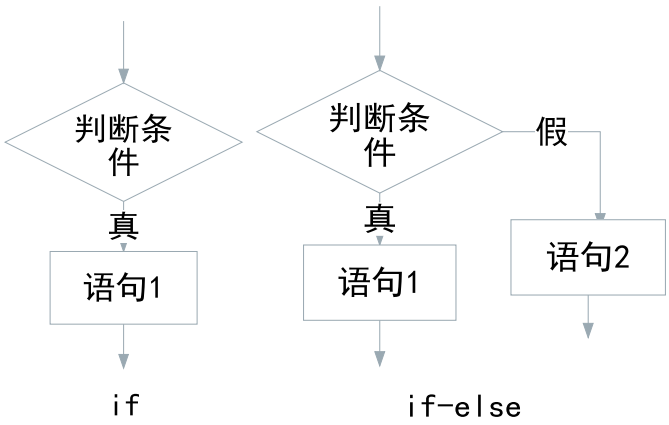

图 1- 59　if 选择结构

三、操作训练

1、任务分析

要实现按键控制 LED 的亮灭对于低层次的程序编制我们使用带自锁的按键，按键按下的状态对应的 IO 口接地，数据为“0”；按键弹开状态对应的 IO 口上拉接电源正极，数据为“1”；若输入的按键不带自锁我们则需要用到相关知识里提到的防抖动，我们可以使用按键判别函数。

2、主函数流程图

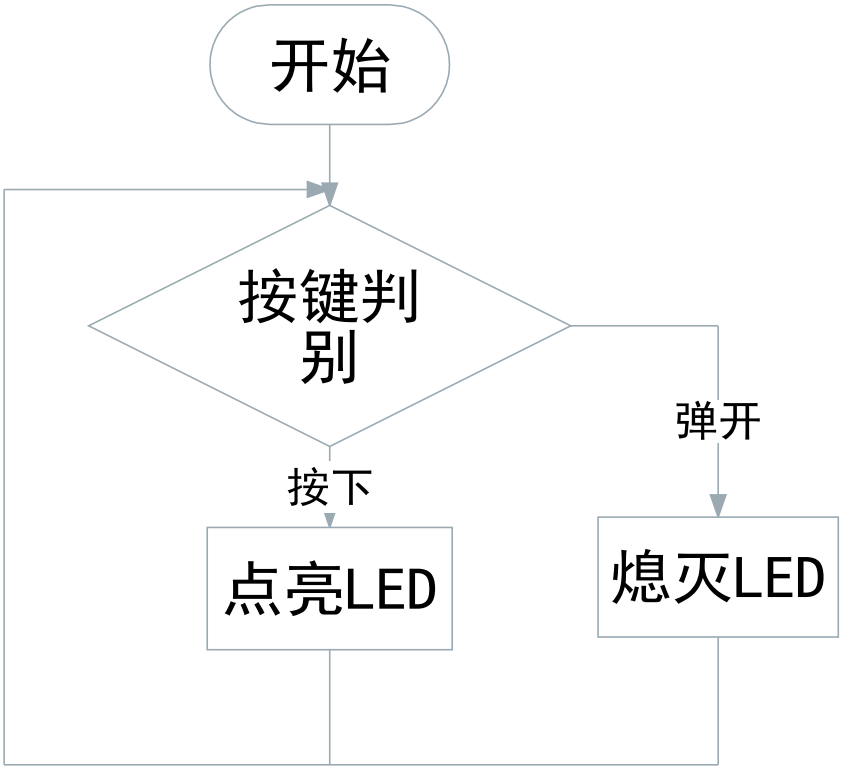

图 1- 60　　自锁按键控制 LED 流程图

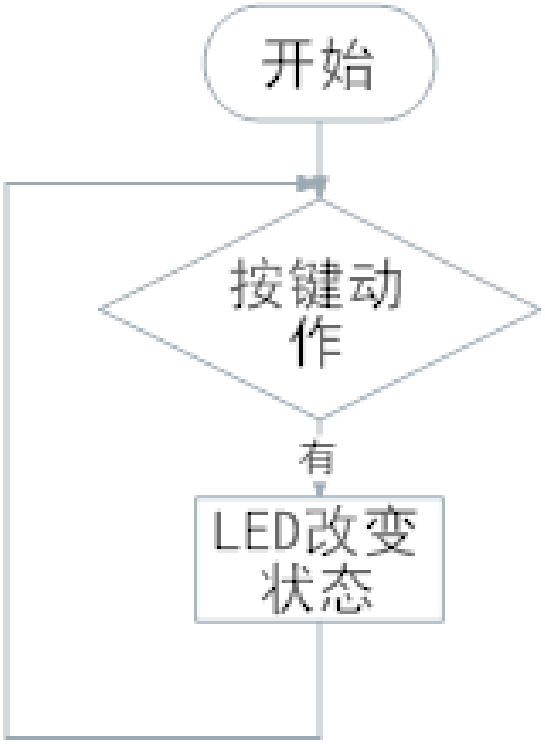

图 1- 61　非自锁按键控制 LED 流程图

3、参考程序

按键（自锁）手动控制 LED 亮灭程序　ZSAJ.c

```
#include <reg51.h>//文件包含指令
sbit LED=P1^7;//变量申明语句
sbit BT =P3^2; //变量申明语句
main()//主函数
{
        while(1)//主函数循环
        {
            if(BT==1)//按键弹开判别“是”分支
                LED=0;//LED 点亮
            else //“否”分支
                LED=1; //LED 点亮
        }
}
```

按键（非自锁）手动控制 LED 亮灭程序 FZSAJ.c

```
#include <reg51.h>//文件包含指令
sbit LED=P1^7;//变量申明
sbit BT =P3^4; //变量申明
bit S=1; //变量申明
void delay()//自定义函数：延时函数
{
    unsigned char i=50,j=100; //变量申明
    while(i--)//循环
    {
        j=100;//变量赋值
        while(j--);//循环
    }
}
unsigned char PushB()//自定义函数：按键动作检测
{
    if(S)//判断“是”分支（按键上一稳定状态为弹开？）
    {
        if(BT==0) //判断“是”分支（当前按键按下？）
        {
            delay();//延时 10ms 下降沿防抖
            if(BT==0)S=0;
// 判断“是”分支（当前按键按下），更新按键稳定状态为按下
            else S=1;
//判断“否”分支（当前按键弹开），更新按键稳定状态为弹开
        }
    }
    else//判断“否”分支（按键上一稳定状态为按下？）
    {
        if(BT==1) //判断“是”分支（当前按键弹开）
        {
            delay();//延时 10ms 上升沿防抖
            if(BT==1) S=1，return 1;
//判断“是”分支（当前按键弹开），更新按键稳定状态为弹开,返回 1
        }
    }
```

```
    return 0;//返回 0
}
main()//主函数
{
        while(1)//主函数循环
        {
            if(PushB())//按键动作判别（有动作）
                LED=!LED;//LED 状态翻转
      }
}
```

4、程序说明

函数 delay()延时 10ms，用于防止按键抖动导致单片机误动作，函数 PushB()用于检测按键是否有真正的按键动作——按下并释放。

5、任务实施

5.1 建立工程

打开 keil 软件，通过菜单“Project”，新建立一个工程”new Project”项目 SKLED，然后再新建文件名为 ZSAJ.C 和 FZSAJ.C 的源程序文件，将上面的参考程序输入并保存。分别将 ZSAJ.C 和 FZSAJ.C 文件添加入本项目中。

5.2 编译并生成 HEX

单击”Build target”按钮，对源程序进行编译和链接，分别产生两个 HEX 文件。

5.3 烧录芯片

打开 STC-ISP 下载软件，选中单片机型号为“STC90C52RC”,选择最低波特率为 2400 最高波特率为 115200。点击打开程序文件按钮，在刚才建立的 SKLED 工程文件夹中生成的 SKLED.HEX 文件。然后点击“下载/编程”按钮。最后给最小系统通电，等待电脑端串口与单片机握手成功。就能把程序载入到单片机中去了。

5.4 硬件调试

程序下载完成后，把单片机最小系统接入电路中，调试电路使其能正常工作，LED 受控。

四、任务评价

表 1- 8　任务四完成情况评价表

项目名称			评价时间		年　月　日		
小组名称		小组成员					
评价内容	评价要求	权重	评价标准	学生自评得分	小组评价得分	教师评价得分	合计
职业与安全意识	1．工具摆放、操作符合安全操作规程；2．遵守纪律、爱惜设备和器材、工位整洁；3．具有团队协作精神。	10%	好（10） 较好（8） 一般（6） 差（<6）				
印制线路板调试	1．能够根据任务要求进行印制线路板的调试；2．根据任务要求，能够对对应电路部分进行硬件电路的检查与故障检修。	15%	好（15） 较好（12） 一般（9） 差（<9）				
项目功能测试	1．编写的程序能成功编译；2．程序能正确烧写到芯片中；3．完成对 LED 的手动控制。	60%	好（60） 较好（45） 一般（30） 差（<30）				
问题与思考	1．怎讲用 4 个按键分别控制 4 个 LED；	15%	好（15） 较好（12） 一般（9） 差（<6）				
教师签名			学生签名			总分	
项目评价=学生自评（0.2）+小组评价（0.3）+教师评价（0.5）							

五、知识拓展

非自锁按键的应用还有一种：辨别按键长按还是短按。试试一直按住电脑的键盘，就会发现这样的应用。如何来实现这样的应用呢？其实就是在判断按键处于按下状态时进行计时，可以设置一个变量进行累加，当变量大于某一数值时就认定为长按按键。

按键动作判别子程序

```
sbit BT=P3^4;
bit S=1;//按键稳态标记
void delay()//延时 10ms
{
    unsigned char i=50,j=100;
    while(i--)
    {
        j=100;
        while(j--);
    }
}
unsigned char count;
unsigned char PushB()//按键检测程序，短按按键返回 1，长按按键返回 2，否则返回 0
{
    if(S)//按键稳定状态判断
    {
        if(BT==0)//按键按下判断
        {
            delay();//防按下抖动
            if(BT==0)S=0;
            else S=1;
        }
        return 0;
    }
    else
    {
        count++;
```

```
            if(count>200)
            {
                count=0;
                return 2;
            }
            if(BT==1)按键弹开
            {
                S=1;
                delay();
                if(BT==1)return 1;
            }
            return 0;
        }
}
```

六、思考与练习

练习按键控制 32 只 LED 中指定的一只或多只。

任务五　8 位 LED 流水灯

一、任务要求

本任务要求同学们完成矩阵广告模块中 8 只 LED（以第四列为例）自上往下依次点亮的显示控制。即点亮第一盏 LED 一段时间后关闭它，再点亮第二盏，依次往下，最后一盏点亮后一段时间关闭，再点亮第一盏，如此一直循环。

二、相关知识

1、硬件知识

先了解一下第四列 8 只 LED 是如何驱动的？如图 1- 62 所示，8 只 LED 的正极分别连接到排阻上，排阻的公共端连接到电源 VCC；8 只 LED 的负极分别连接到 P3 口的 8 位，自上而下第 1 只（DS4）连接到 P3.0；第 2 只（DS8）连接到 P3.1；第 3 只（DS12）连接到 P3.2；第 4 只（DS16）连接到 P3.3；第 5 只（DS20）连接到 P3.4；第 6 只（DS24）连接到 P3.5；第 7 只（DS28）连接到 P3.6；第 8 只（DS32）连接到 P3.7。

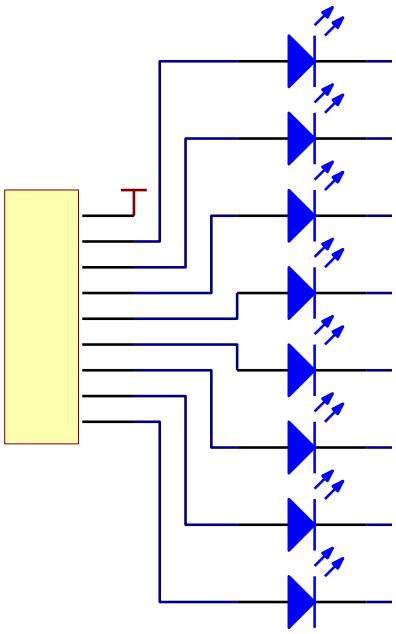

图 1- 62　第四列驱动电路

2、软件知识

任务三中我们学会了如何控制 LED 的亮灭，比如，本任务中我们首先要让 DS4 点亮，DS8、DS12、DS16、DS20、DS24、DS28 和 DS32 熄灭那么 P3 口每一位对应的数据如表 1- 9

表 1- 9　P3 口数据 1

P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0
1	1	1	1	1	1	1	0

这样的数据在 C 语言中怎么表示呢？我们习惯用十六进制表示为“0xfe”。

2.1 程序编写知识

（1）常量的表示

在任务三中我们用语句“P1_7=1;”来控制 P1.7 口输出高电平，其中的数据“1”就是一种整形常量。而本任务要表示的数据要复杂一点如二进制数据“11111110”在 C 语言中可以用三种方式表示，十进制、八进制和十六进制。整形常数在 C 语言里的表示如表 1- 10 所示。我们常用的是十进制和十六进制。

表 1- 10　常数的表示

整形常量类型	表示形式	示例
十进制数	以非 0 开始的数来表示	220，-560，45900
八进制数	以 0 开始的数来表示	06，0106，0578
十六进制数	以 0X 或 0x 开始的数来表示	0X0D，0XFF，0x4e

（2）进制转换

二进制转换成十进制的方法是将二进制数按权展开式相加。如：

（1011.11）2=$1\times2^3+0\times2^2+1\times2^1+1\times2^0+1\times2^{-1}+1\times2^{-2}$

=8+0+4+2+1+0.5+0.25

=11.75D

十六进制转换成十进制的方法是将十六进制数按权展开式相加。如：

B2AH=$11\times16^2+2\times16^1+10\times16^0=11\times256+32+10$

=2858D。

十进制转换成十六进制的方法是整数部分：除 16 取余倒记；小数部分：乘 16 取整顺记。如：

98D=62H

二进制与十六进制转换的方法是四位一组，对应填写。如：

100 1111 1110B=4 F E H

（3）数组

根据上面所述的知识我们可以把本任务中 P3 口输出的数据需要归纳如表 1-11 所示。

表 1- 11　P3 口输出数据表

序号	LED 状态自下往上	二进制	十进制	十六进制
1	灭灭灭灭 灭灭灭亮	11111110	254	0xfe
2	灭灭灭灭 灭灭亮灭	11111101	253	0xfd
3	灭灭灭灭 灭亮灭灭	11111011	251	0xfb
4	灭灭灭灭 亮灭灭灭	11110111	247	0xf7
5	灭灭灭亮 灭灭灭灭	11101111	239	0xef
6	灭灭亮灭 灭灭灭灭	11011111	223	0xdf
7	灭亮灭灭 灭灭灭灭	10111111	191	0xbf
8	亮灭灭灭 灭灭灭灭	01111111	127	0x7f

这样的 8 个数在数学里我们称之为数列：0xfe、0xfd、0xfb、0xf7、0xef、0xef、0xdf、0xdf、0xbf、0x7f。在 C 语言中我们用数组来表示。

数组是把若干具有相同数据类型的变量按有序的形式组织起来的集合。其中，数组中的每个变量称为数组元素。在 C51 语言中，使用数组前必须先进行定义，即数组的类型说明。数组定义的一般形式为：类型说明符 数组名 [常量表达式]，……；

数组元素，即数组中的变量，是组成数组的基本单元。在 C51 中，数组元素是变量，其标识方法为数组名后跟一个下标。数组元素通常也称为下标变量。数组元素表示的一般形式为：数组名[下标]

小知识点：数组的申明和使用

我们在单片机 C 语言程序基本构架里的第一行（#include <reg51.h>//引用 51 单片机头文件）和第二行（void main(void)//主程序 main 函数）之间插入一

行写上数组的定义：

unsigned char DataForLED[]={0xfe,0xfd,0xfb,0xf7,0xef,0xef,0xdf,0xdf,0xbf,0x7f};

利用这些数据控制 P3 口的 LED，可以直接用赋值语句，即“P3= DataForLED[n];”，其中的 n 为下标，可以是常数也可以是变量。

【注意】：数组的起始元素是 0 元素，上文所述的数组起始元素是 DataForLED[0]。

三、操作训练

1、任务分析

任务要求点亮第一盏 LED 一段时间后关闭它再点亮第二盏，依次往下，最后一盏点亮后一段时间关闭再点亮第一盏，如此一直循环。

从上文的描述得知，我们需要做的就是先把数组的 0 元素赋值给 P3 口，然后延时，接着把数组的 1 元素赋值给 P3 口，然后延时……这样把数组的 8 个元素依次赋值个 P3 口。我们可以利用一个变量作为取数组元素时的下标，并且让这个变量每延时一次加 1，同时控制其取值范围：0 到 7（模 8），因为我们的数组只有 8 个元素。

至于延时的方法我们依旧应用任务三里所述的方法。

2、主函数流程图

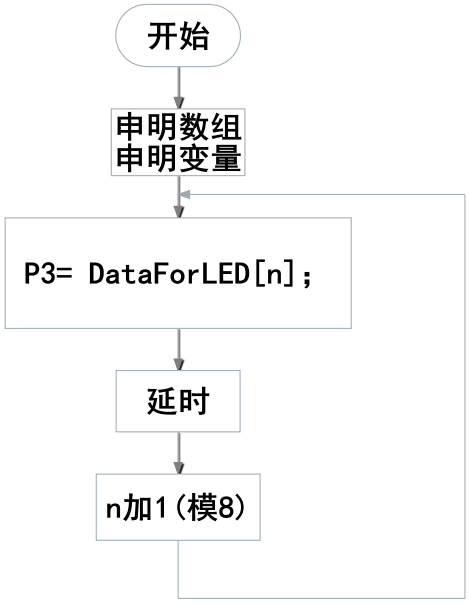

图 1- 63　8 位 LED 流水灯

3、参考程序

8 位 LED 流水灯 8LEDLSD.C

```
#include <reg51.h>//引用 51 单片机头文件
unsigned char DataForLED[]={0xfe,0xfd,0xfb,0xf7,0xef,0xef,0xdf,0xdf,0xbf,0x7f};//定义数组存放 P3 口 LED 驱动数据
void main(void)//主程序 main 函数
{
    unsigned int i;
    unsigned char n;
/*在主程序中设置死循环程序，保证主程序的运行。*/
    while(1)//主程序死循环
    {
        /*在此处添加控制程序*/
        P3= DataForLED[n]；//控制 P3 口的 LED 灯。
        i=50000;while(i--);//延时一段时间
        n++;//数组下标变量自加 1。
        if(n>7);n=0;//控制下标变量取值范围：0 到 7
    }
}
```

4、程序说明

程序中“n++”中“++”是一种特殊的算术运算符，它的是将变量自加以后的值重新赋值给该变量，相当于“n=n+1”。

主函数死循环内 P3 口将依次获得数组中的元素从而实现 8 只 LED 流水显示。

5、任务实施

5.1 建立工程

打开 keil 软件，通过菜单“Project”，新建立一个工程”new Project”项目 8LEDLSD，然后再新建一个文件名为 8LEDLSD.C 的源程序文件，将上面的参考程序输入并保存。然后将 8LEDLSD.C 文件添加入本项目中。

5.2 编译并生成 HEX

单击”Build target”按钮，对源程序进行编译和链接，产生 HEX 文件。

5.3 烧录芯片

打开 STC-ISP 下载软件，选中单片机型号为“STC90C52RC”,选择最低波特率为 2400 最高波特率为 115200（为默认值一般不需修改）。点击打开程序文件按钮，在刚才建立的 8LEDLSD 工程文件夹中生成的 8LEDLSD.HEX 文件。然后点击“下载/编程”按钮。最后使用串口线连接 PC 与主机模块，然后给主机模块通电，等待电脑端串口与单片机握手成功。就能把程序载入到单片机中去了。

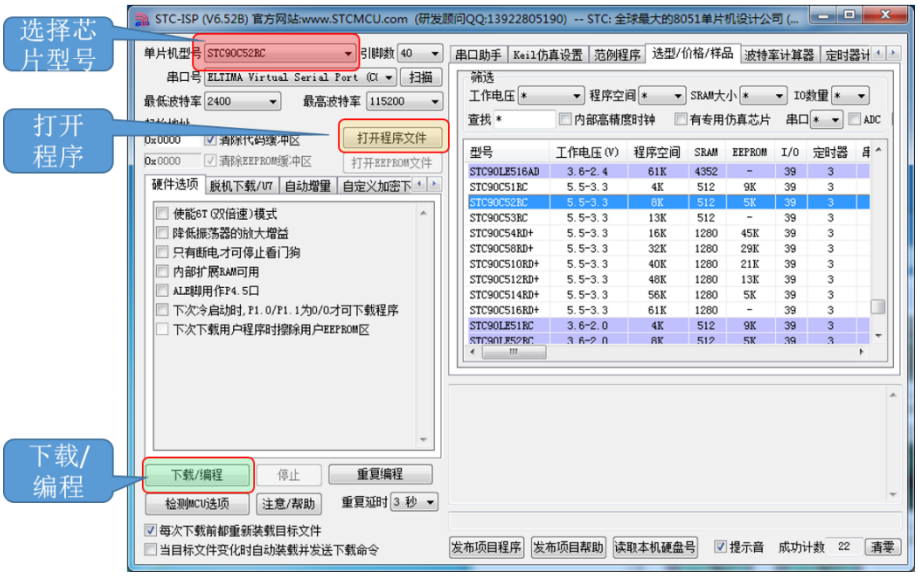

图 1- 64 STC 单片机程序烧写示意图

5.4 硬件调试

程序下载完成后，把单片机最小系统接入电路中，调试电路使其能正常工作，使 8 只 LED 流水显示。

四、任务评价

表 1- 12　任务五完成情况评价表

项目名称			评价时间		年　月　日		
小组名称		小组成员					
评价内容	评价要求	权重	评价标准	学生自评得分	小组评价得分	教师评价得分	合计
职业与安全意识	1．工具摆放、操作符合安全操作规程；2．遵守纪律、爱惜设备和器材、工位整洁；3．具有团队协作精神。	10%	好（10） 较好（8） 一般（6） 差（<6）				
模块的布局和布线工艺	1．模块布局合理，模块的选择应符合模块的要求。	15%	好（15） 较好（12） 一般（9） 差（<9）				
项目功能测试	1．编写的程序能成功编译；2．程序能正确烧写到芯片中；3．第四列 8 只 LED 流水显示。	60%	好（60） 较好（45） 一般（30） 差（<30）				
问题与思考	程序中数组下标变量进行了取值范围的控制，若不控制会有什么后果？	15%	好（15） 较好（12） 一般（9） 差（<6）				
教师签名			学生签名			总分	
项目评价=学生自评（0.2）+小组评价（0.3）+教师评价（0.5）							

五、知识拓展

本任务中 8 只 LED 流水显示的驱动数据我们预先计算好了，并它们放在申明了数组中；其实这 8 个数据的数值具有一定的规律，找出这个规律，并用表达式写出了，我们就可以让单片机自己把数据计算出来，不需要我们预先计算好。而这个规律就是这 8 个数据的数值等于二进制数“11111110”循环左移 n 位，n 取值为 0、1、2、3、4、5、6、7；刚好是参考程序里数组下标变量相同。

让数据循环左移可以使用库函数：_crol_（）；它的函数原型在头文件 INTRINS.H 里。使用时我们首先包含头文件 INTRINS.H（#include <INTRINS.H>）,然后调用库函数（P3=_crol_(0xfe,n);）即可。

六、思考与练习

练习使用库函数：_crol_（）；实现本任务；

练习实现 16 只 LED 流水显示。

任务六　32 位 LED 闪烁

一、任务要求

与任务三的要求不同，本任务要求控制的是矩阵广告模块中的 32 只 LED，同时还要求每次点亮和熄灭的时间都是 1 秒钟，这里所述的 1 秒是精确的一秒。

二、相关知识

1、硬件知识

1、1　32 只 LED 驱动电路

首先看看矩阵广告模块中 32 只 LED 是如何驱动的，其驱动电路原理图如图 1-65 所示。32 只 LED 被分成了 8 行 4 列，每一列得 8 只 LED 的正极都通过排阻连接到电源 VCC。其负极接到单片机的 IO 口，其第一行 4 只 DS1、DS2、DS3、DS4 分别连接到 P0.0、P1.0、P2.0、P3.0；第二行 DS5、DS6、DS7、DS8 分别连接到 P0.1、P1.1、P2.1、P3.1；……构成 8 行 4 列的矩阵。

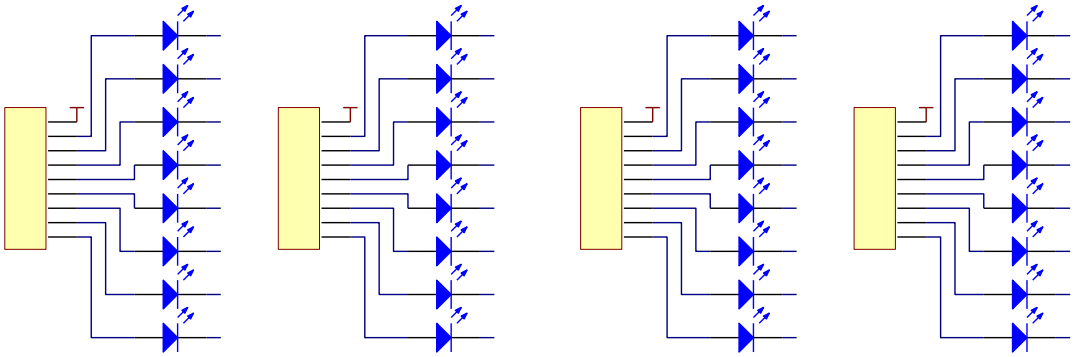

图 1-65　32 只 LED 驱动电路

闪烁就是 LED 间隔的亮灭。32 只全亮就是单片机 32 个 IO 口均输出“0”；32 只全灭就是单片机 32 个 IO 口均输出“1”。这样的 IO 控制相信大家已经学会了。

本任务的关键是产生 1 秒的时间间隔以控制点亮时间。单片机是个复杂的时序电路，它是在时钟的控制下进行工作的，这个时钟就是我们的系统时钟其频率为 11.0592MHz（主机模块采用的是 11.0592 的晶振）。这样的频率是每秒钟变化 11059200 次。我们只要能够对系统时钟进行计数，每计满 11059200 就表示

时间过去了 1 秒。

幸运的，51 单片机拥有 2 个这样的计数器，下面我们来了解一下：

1、2　定时器/计数器

STC 单片机有 2 个定时器，定时器 0 和定时器 1。与定时器/计数器相关的寄存器如表 1- 13 所示。

表 1- 13

符号	描述	地址	位地址及其符号 MSB							LSB	复位值
TCON	定时器控制寄存器	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	0000 0000B
TMOD	定时器模式寄存器	89H	GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0	0000 0000B
TL0	Timer Low 0	8AH									0000 0000B
TL1	Timer Low 1	8BH									0000 0000B
TH0	Timer High 0	8CH									0000 0000B
TH1	Timer High 1	8DH									0000 0000B

定时器/计数器控制寄存器 TCON

TCON 为定时器/计数器 T0、T1 的控制寄存器，同时也锁存 T0、T1 溢出中断源和外部请求中断源等，TCON 格式如下：

TCON：定时器/计数器中断控制寄存器　(可位寻址)

SFR name	Address	bit	B7	B6	B5	B4	B3	B2	B1	B0
TCON	88H	name	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1: 定时器/计数器 T1 溢出标志。T1 被允许计数以后，从初值开始加 1 计数。当最高位产生溢出时由硬件置“1”TF1，向 CPU 请求中断，一直保持到 CPU 响应中断时，才由硬件清“0”TF1（TF1 也可由程序查询清“0”）。

TR1: 定时器 T1 的运行控制位。该位由软件置位和清零。当 GATE（TMOD.7）=0，TR1=1 时就允许 T1 开始计数，TR1=0 时禁止 T1 计数。当 GATE（TMOD.7）=1，TR1=1 且 INT1 输入高电平时，才允许 T1 计数。

TF0：定时器/计数器 T0 溢出中断标志。T0 被允许计数以后，从初值开始加 1 计数，当最高位产生溢出时，由硬件置“1”TF0，向 CPU 请求中断，一直保持 CPU 响应该中断时，才由硬件清“0”TF0（　　TF0 也可由程序查询清“0”）。

TR0: 定时器 T0 的运行控制位。该位由软件置位和清零。当 GATE（TMOD.3）=0，TR0=1 时就允许 T0 开始计数，TR0=0 时禁止 T0 计数。当 GATE（TMOD.3）=1，TR1=0 且 INT0 输入高电平时，才允许 T0 计数。

IE1：外部中断 1 请求源（INT1/P3.3）标志。IE1=1，外部中断向 CPU 请求中

断，当 CPU 响应该中断时由硬件清“0”IE1。

IT1：外部中断 1 触发方式控制位。IT1=0 时，外部中断 1 为低电平触发方式，当 INT1（P3.3）输入低电平时，置位 IE1 。采用低电平触发方式时，外部中断源（输入到 INT1）必须保持低电平有效，直到该中断被 CPU 响应，同时在该中断服务程序执行完之前，外部中断源必须被清除(P3.3 要变高)，否则将产生另一个中断。当 IT1=1 时，则外部中断 1(INT1)端口由“1”→“0”下降沿跳变，激活中断请求标志位 IE1 ，向主机请求中断处理。

IE0：外部中断 0 请求源（INT0/P3.2）标志。IE0=1 外部中断 0 向 CPU 请求中断，当 CPU 响应外部中断时，由硬件清“0”IE0（边沿触发方式）。

IT0：外部中断 0 触发方式控制位。IT0=0 时，外部中断 0 为低电平触发方式，当 INT0（P3.2）输入低电平时，置位 IE0。采用低电平触发方式时，外部中断源（输入到 INT0）必须保持低电平有效，直到该中断被 CPU 响应，同时在该中断服务程序执行完之前，外部中断源必须被清除（P3.2 要变高），否则将产生另一个中断。当 IT0=1 时，则外部中断 0(INT0)端口由“1”→“0”下降沿跳变，激活中断请求标志位 IE1 ，向主机请求中断处理。

定时器/计数器工作模式寄存器 /计数器工作模式寄存器 TMOD

定时和计数功能由特殊功能寄存器 TMOD 的控制位 C/T 进行选择，TMOD 寄存器的各位信息如下表所列。可以看出，2 个定时/计数器有 4 种工作模式，通过 TMOD 的 M1 和 M0 选择。2 个定时/计数器的模式 0、1 和 2 都相同，模式 3 不同，各模式下的功能如下所述。

寄存器 TMOD 各位的功能描述

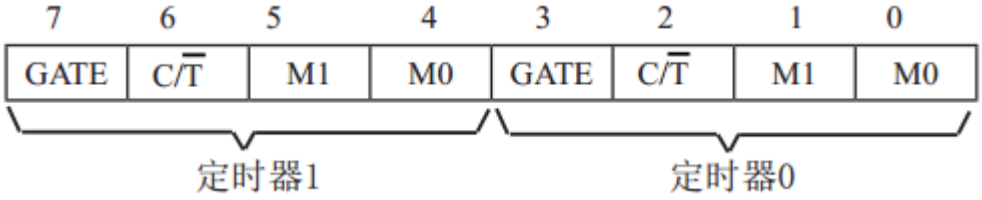

表 1- 14 寄存器 TMOD

位	符号	功能
TMOD.7/	GATE	TMOD.7 控制定时器 1,置 1 时只有在 INT1 脚为高及 TR1 控制位置 1 时才可打开定时器/计数器 1。
TMOD.3/	GATE	TMOD.3 控制定时器 0,置 1 时只有在 INT0 脚为高及 TR0 控制位

		置 1 时才可打开定时器/计数器 0。
TMOD.6/	C/T	TMOD.6 控制定时器 1 用作定时器或计数器，清零则用作定时器(从内部系统时钟输入)，置 1 用作计数器(从 T1/P3.5 脚输入)
TMOD.2/	C/T	TMOD.2 控制定时器 0 用作定时器或计数器，清零则用作定时器 (从内部系统时钟输入)，置 1 用作计数器(从 T0/P3.4 脚输入)
TMOD.5/ TMOD.4	M1、M0	定时器定时器/计数器 1 模式选择
TMOD.1/ TMOD.0	M1、M0	定时器/计数器 0 模式选择

定时器定时器/计数器 1 模式选择:

0 0　　13 位定时器/计数器，兼容 8048 定时模式，TL1 只用低 5 位参与分频，TH1 整个 8 位全用。

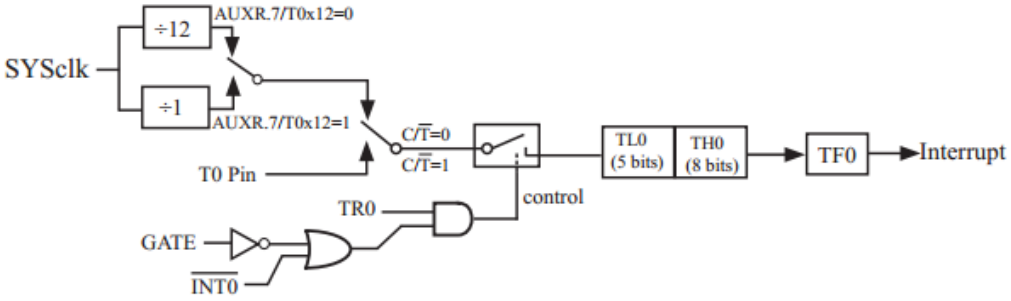

图 1- 66　模式 0 框图

0 1　　16 位定时器/计数器，TL1、TH1 全用。

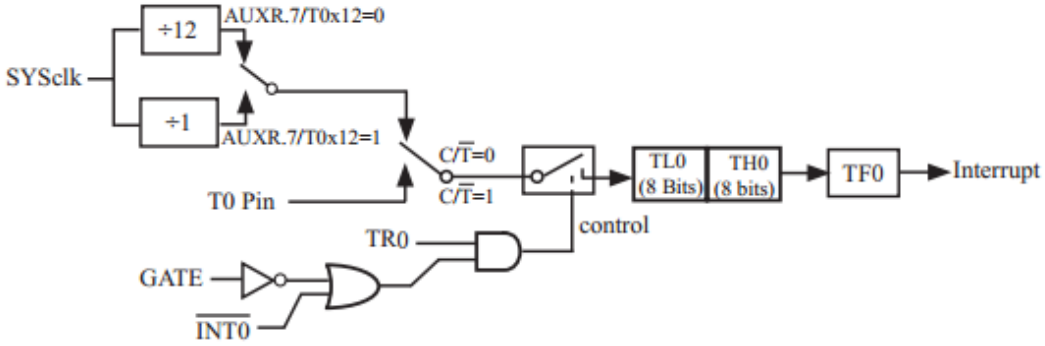

图 1- 67　模式 1 框图

1 0　　8 位自动重装载定时器,当溢出时将 TH1 存放的值自动重装入 TL1。

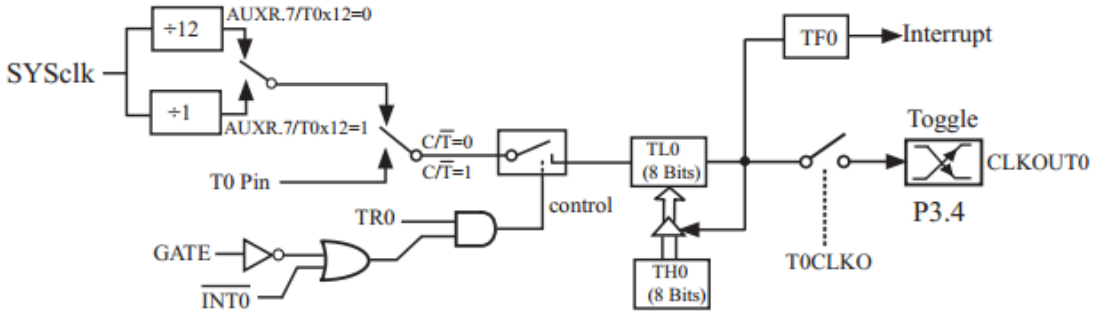

图 1- 68　模式 2 框图

1 1　对定时器 1，在模式 3 时，定时器 1 停止计数，效果与将 TR1 设置为 0 相同。对定时器 0，此模式下定时器 0 的 TL0 及 TH0 作为 2 个独立的 8 位计数器。下图为模式 3 时的定时器 0 逻辑图。TL0 占用定时器 0 的控制位：C/T、GATE、TR0、INT0 及 TF0。TH0 限定为定时器功能（计数器周期），占用定时器 1 的 TR1 及 TF1。此时，TH0 控制定时器 1 中断。模式 3 是为了增加一个附加的 8 位定时器/计数器而提供的，使单片机具有三个定时器/计数器。模式 3 只适用于定时器/计数器 0，定时器 T1 处于模式 3 时相当于 TR1=0，停止计数，而 T0 可作为两个定时器用。

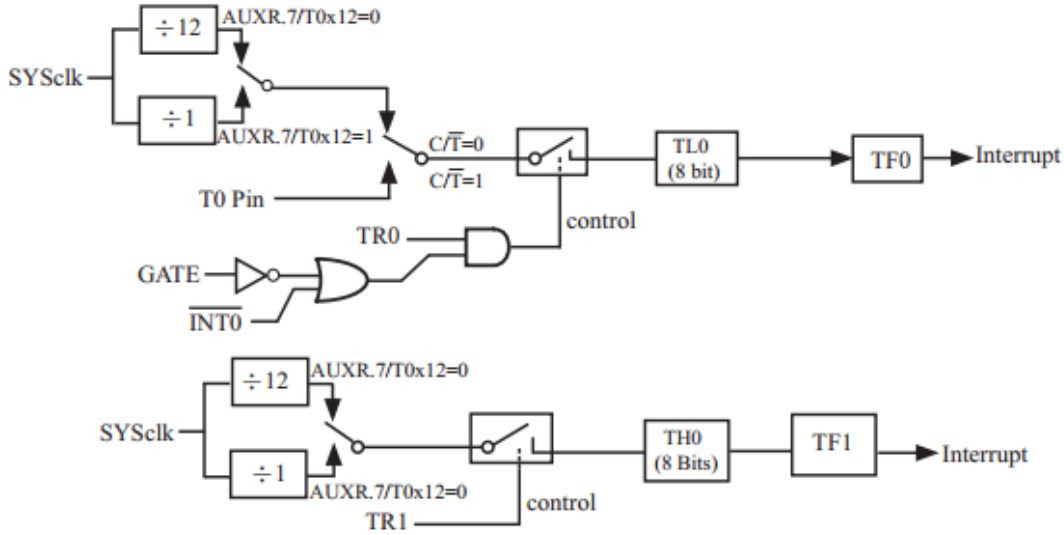

图 1- 69　模式 3 框图

2、软件知识

综上所述，产生 1 秒信号，只要能够对系统时钟进行计数，每计满 11059200 就表示时间过去了 1 秒。我们的定时器均可以对系统时钟进行计数，如何设置定时器必须通过代码告诉单片机。

2、1　定时器初始化

定时器/计数器工作方式设定，通过对寄存器 TMOD 的赋值实现定时器/计数器工作方式设定，我们要设定定时器 0 为 16 位定时器可以使用语句“TMOD=1；”

定时器初值设置，为兼容传统 51 我们采用 12T 模式，定时器每 12 个系统时钟计数一次，即计满 921600（11059200/12）次就满 1 秒。16 位计数最大值是 65535，我们设计让定时器从 56320（65536-9216）开始计数，到计数器计满溢出共经过 9216 次计数（即 0.01 秒），然后记录溢出 100 次就满 1 秒了。这时定时器是从 56320 开始加法计数的，56320 就是计数初值，即十六进制数 0xdc00，用语句“TH0=0xdc；TL0=0；”来实现。

定时器开启设置，通过对定时器/计数器控制寄存器 TCON 的 TR0 进行设置。使用语句“TR0=1；”即可。

定时器初始化代码

```
TMOD=1；//16 定时器
TH0=0xdc；
TL0=0；//定时初值设定
TR0=0；//定时器开启
```

2、2　定时器溢出查询

T0 被允许计数以后，从初值开始加 1 计数，计数最大产生溢出时，由硬件置“1”TF0。我们可以判断 TF0 是否为“1”监测定时器是否产生溢出。判断 TF0 是否为“1”，可以使用 if 语句，

定时器溢出查询代码

```
if(TF0)
{
    TF0=0;
    TH0=0xdc;
    /*此处插入定时器溢出处理代码*/
}
```

三、操作训练

1、任务分析

上文的定时器定时值设置成 0.01 秒，我们可以使用一个变量来记录定时器溢出了几次，即时间经过了几个 0.01 秒，计满 100 次就是 1 秒了。在主函数中在判断是否经过了 1 秒，每经过 1 秒将 LED 状态改变一次。

2、主函数流程图

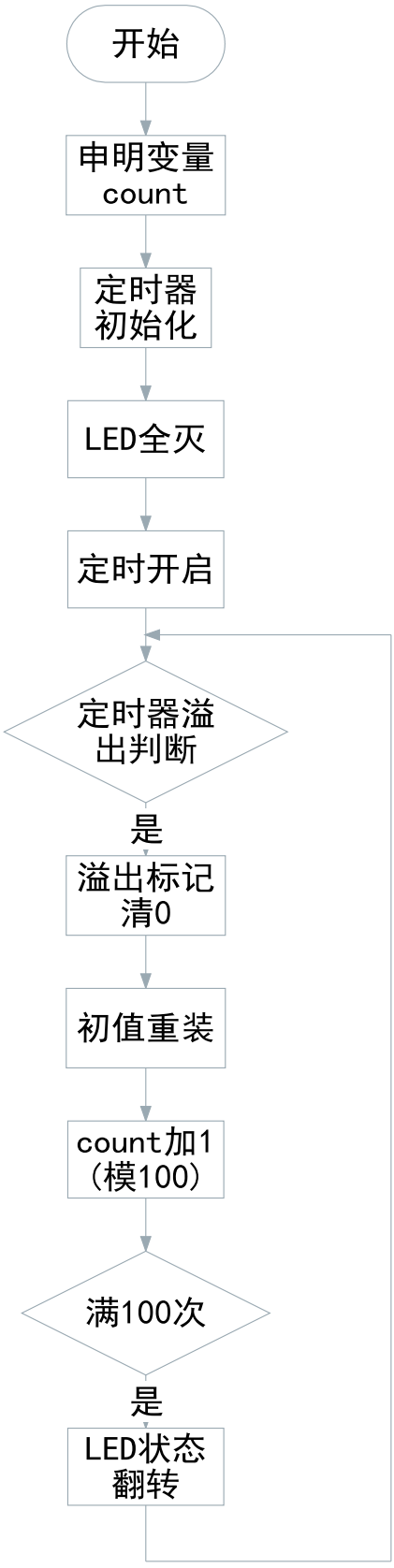

图 1- 70　32 只 LED 闪烁流程图

3、参考程序

32 位 LED 闪烁 32LEDSS.C

```
#include <reg51.h>//引用 51 单片机头文件
void main(void)//主程序 main 函数
{
	unsigned char count;
	TMOD=1；
	TH0=0xdc；
	TL0=0；
	TR0=0；
	P0=0xff;
	P1=0xff;
	P2=0xff;
	P3=0xff;
/*在主程序中设置死循环程序，保证主程序的运行。*/
	while(1)//主程序死循环
	{
		/*在此处添加控制程序*/
		if(TF0)
		{
			TF0=0;
			TH0=0xdc;
			/*此处插入定时器溢出处理代码*/
			count++
			if(count>99)
			{
				P0=~P0;
				P1=~P1;
				P2=~P2;
				P3=~P3;
				count=0；
			}
		}
	}
}
```

4、程序说明

程序中运算符“~”为按位取反运算，将 IO 口寄存器数据按位取反实现 LED 状态翻转。

查询定时器 0 溢出标记 TF0 时一定要对溢出标记清 0，因为使用查询方式时硬件不会清 0 所以程序里要清 0。

5、任务实施

5.1 建立工程

打开 keil 软件，通过菜单“Project”，新建立一个工程”new Project”项目 32LEDSS，然后再新建一个文件名为 32LEDSS.C 的源程序文件，将上面的参考程序输入并保存。然后将 32LEDSS.C 文件添加入本项目中。

5.2 编译并生成 HEX

单击  ”Build target”按钮，对源程序进行编译和链接，产生 HEX 文件。

5.3 烧录芯片

打开 STC-ISP 下载软件，选中单片机型号为“STC90C52RC”,选择最低波特率为 2400 最高波特率为 115200（为默认值一般不需修改）。点击打开程序文件按钮，在刚才建立的 32LEDSS 工程文件夹中生成的 32LEDSS.HEX 文件。然后点击“下载/编程”按钮。最后使用串口线连接 PC 与主机模块，然后给主机模块通电，等待电脑端串口与单片机握手成功。就能把程序载入到单片机中去了。

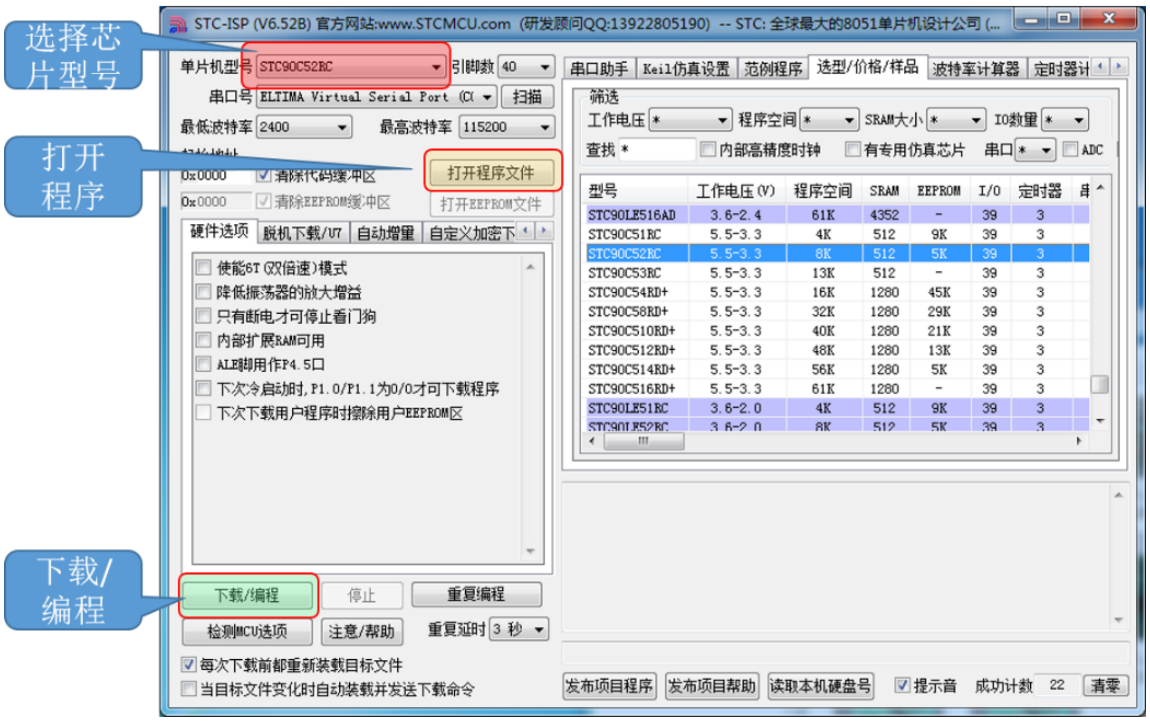

图 1- 71　STC 单片机程序烧写示意图

5.4　硬件调试

程序下载完成后，把单片机最小系统接入电路中，调试电路使其能正常工作，使 32 只 LED 闪烁间（隔 1 秒）显示。

四、任务评价

表 1- 15　任务六完成情况评价表

项目名称			评价时间		年　月　日		
小组名称		小组成员					
评价内容	评价要求	权重	评价标准	学生自评得分	小组评价得分	教师评价得分	合计
职业与安全意识	1．工具摆放、操作符合安全操作规程；2．遵守纪律、爱惜设备和器材、工位整洁；3．具有团队协作精神。	10%	好（10） 较好（8） 一般（6） 差（<6）				
模块的布局和布线工艺	1．模块布局合理，模块的选择应符合模块的要求。	15%	好（15） 较好（12） 一般（9） 差（<9）				
项目功能测试	1．编写的程序能成功编译；2．程序能正确烧写到芯片中；3．32 只 LED 闪烁间隔 1 秒。	60%	好（60） 较好（45） 一般（30） 差（<30）				
问题与思考	闪烁时能否实现 16 只亮的同时另 16 之灭；	15%	好（15） 较好（12） 一般（9） 差（<6）				
教师签名			学生签名			总分	
项目评价=学生自评（0.2）+小组评价（0.3）+教师评价（0.5）							

五、知识拓展

定时器初值计算我们预先计算好了，然后赋值给定时器计数器 THx 和 TLx。我们可以总结归纳前文中初值计算的过程不难发现，若我们知道系统时钟频率 FOSC 单位 Hz，定时器为 16 位，设计定时值 x 单位毫秒。则初值 TxMS=（65535-FOSC*x/12/1000）；这样我们也可以让单片机来计算初值。

定时器初值计算代码

```
#define FOSC 11059200L//11.0592MHz
#define X 10//定时值 10ms（0.01s）
#define TXMS (65536-FOSC*X/12/1000)；//16 位定时器
TH0=TXMS〉〉8；
TL0=TXMS；
```

六、思考与练习

练习调整 LED 闪烁的速度，本任务的速度是 2 秒一次，练习实现 0.5 秒闪烁一次或 1 秒闪烁一次。

任务七　LED 矩阵广告牌

一、任务要求

这个任务我们做一个 LED 矩阵广告牌，设计首先 32 只 LED 依次点亮（由第一列从上往下，接着第二列由下往上，然后第三列由上往下，最后第四列有下往上）；然后 32 只 LED 每 2 秒闪烁一次，共闪烁 10 次秒；最后第一行四只 LED 同时点亮，依次自上往下点亮 8 行 LED 的一行。上述 3 种等效交替进行循环不止。

二、相关知识

在任务四中，主函数不停地察看 TF0，就好比一直盯住手机看有没有来电，如果有更重要的事件要做就不可能一直盯住手机，还好我们的手机有铃声提醒我们，有来电时我们中断手头的事接电话，接完电话继续做事。单片机也可以这样，实现中断。

1、硬件知识

1、1　中断

51 系列单片机具有 5 个中断源，两级中断优先级，具有完善的指令控制能力。使用单片机的中断系统可以很方便地完成各种外部硬件响应的操作。

通常按照中断源的不同，大致可以分为三类：外部中断源、定时中断源和串行中断源。在 8051 单片机中共有 5 个中断源，包括 2 个外部中断源、2 个定时中断源和 1 个串行中断源。

51 系列单片机的 5 个中断请求，分别对应一个中断请求标志位。当中断发生时，将置位相应的中断请求标志位，并向 CPU 提出请求。这里，特殊功能寄存器 TCON 控制外部中断和定时器溢出中断，特殊功能寄存器 SCON 控制串行接口中断。

对于 51 系列单片机的 5 个中断请求，分别对应一个中断允许或禁止标志位，其均由中断允许控制寄存器 IE 控制。用户可以在程序中使用中断允许标志位来允许或者禁止相应的中断请求。

中断允许控制寄存器 IE 的位定义格式，如图 1- 72 所示。其字节地址为 A8H，

可以位寻址。其中 D7 位为总控制位，D0~D4 分别控制各个中断请求，D5 和 D6 位未定义。

	D7	D6	D5	D4	D3	D2	D1	D0	可位寻址
位地址	AF	AE	AD	AC	AB	AA	A9	A8	
位符号	EA	—	—	ES	ET1	EX1	ET0	EX0	

图 1- 72　中断允许控制寄存器 IE 格式

51 系列单片机对中断的处理分为 4 步：中断响应、中断处理、中断请求的撤离和中断返回。1．中断的响应:是指在单片机主程序运行过程中，如果遇到中断请求，在满足中断响应条件的情况下，CPU 对该中断做出的响应。2．中断的处理:8051 单片机 CPU 对中断处理的过程可以分为两类，一个是单片机硬件自动完成的部分，另一个是 C51 软件处理的部分。3．中断请求的撤离:中断请求的撤离主要保证对于一次中断信号只执行一次中断响应。单片机响应某个中断后，应及时将中断请求标志 TCON 或 SCON 中对应的标志位清除，否则会导致一个中断信号触发多次中断响应。4．中断的返回:中断的返回是指中断服务例程结束后，返回主程序的过程。在 8051 单片机中，中断服务例程的返回比较简单，直接在中断服务程序最后面加上一个中断返回指令 RETI 即可。而如果使用 C51 语言进行程序设计，则将由编译系统完成此工作。

2、软件知识

如图 1- 73 所示为 51 单片机的中断控制系统示意图。要让单片机中断系统工作我们需要进行设置。即设置让图 1- 73 中的那些开关闭合？我们要对寄存器 TCON、IE 和 IP 进行设置。其中 TCON 和 IE 的各位可以直接访问，即可以直接用赋值如：“EA=0；”。

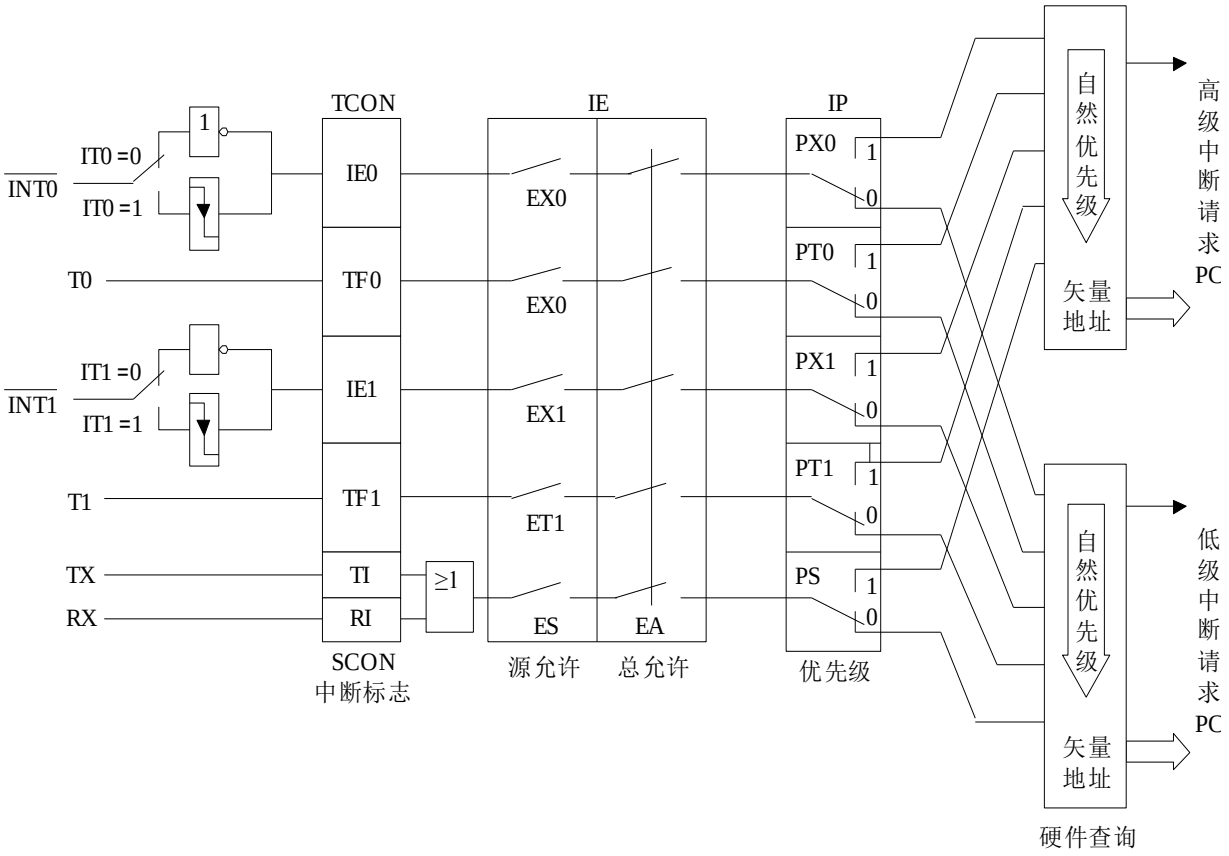

图 1- 73　中断控制系统示意图

我们这个任务设计开启定时器 0 中断，任务里不涉及其他中断源，所以 IP 不需要进行设置。

中断设置

```
ET0=1;
EA=1;
```

单单允许中断还不够，我们还得告诉单片机，中断该如何响应。这样的响应处理程序在 C 语言被写为中断函数。本任务需定义定时器 0 中断函数。

定时器 0 中断函数

```
void Timer0() interrupt 1
{
    TH0=0xdc;
    //TL0=0;//初值重装
    /*此处插入定时器中断处理代码*/
}
```

小知识点：中断函数

中断函数的具体形式如下：

void funcname() interrupt n [using m]

其中 n、m 为常量。n 表示中断源，取值 0、1、2、3、4；m 表示工作寄存器组，取值 0、1、2、3；

n=0：外部中断 0；n=1：定时器 0；n=2 外部中断 1；n=3：定时器 1；n=4：串行口中断。

【注意】：using m 为指定工作寄存器组。C51 语法中，可以不指定工作寄存器组。但不指定工作寄存器组，中断时，默认的工作寄存器组就会被推入堆栈，这将需要 32 个处理周期。

三、操作训练

1、任务分析

我们用中断服务函数负责产生 1 秒时间间隔标记，主函数负责在时间间隔标记有效时改变 IO 的输出值从而实现所要灯光效果。首先分析定时器中断的任务和设置

1.1 定时器中断初始化设置

设置定时器 0 为 16 位定时器，定时值为 10ms，允许定时器 0 中断，并开启定时器 0。

定时器中断初始化函数

```
#define FOSC 11059200L//11.0592MHz
#define X 10//定时值 10ms
#define TXMS (65536-FOSC*X/12/1000);//16 位定时器初值计算
void T0int()
{
    TMOD=1;//16 位定时方式
    TH0=TXMS>>8;
    TL0=TXMS;//初值设置
    ET0=1;
    EA=1;//中断允许
    TR0=1;//定时器 0 开启
```

```
}
```

1.2 定时器 0 中断函数

中断函数要完成初值重装，产生 1 秒间隔标记信号。

定时器中断函数

```
#define FOSC 11059200L//11.0592MHz
#define X 10//定时值 10ms
#define TXMS (65536-FOSC*X/12/1000);//16 位定时器初值计算
bit F_1S=0;// 1 秒间隔标记
unsigned char count;
void Timer0() intrrupt 1
{
    TH0=TXMS>>8;
    TL0=TXMS;//初值重装
    count++;
    if(count>99)
    {
        F_1S=1;// 1 秒间隔标记有效
    }
}
```

1.3 LED 驱动数据

任务四中 LED 自上而下点亮的驱动数据是：0xfe、0xfd、0xfb、0xf7、0xef、0xef、0xdf、0xdf、0xbf、0x7f；自下而上数据正好是上述数据排序相反；闪烁的数据是 0xff 和 0；同行点亮实质上是 P0、P1、P2 和 P3 口的数据同为自上而下的驱动数据。整个矩阵广告牌的显示状态有 8+8+8+8+20+8=60 个；第 1 到 8 是 P0 口输出自上而下点亮的驱动数据，第 9 到 16 是 P1 口输出自下而上点亮的驱动数据，第 17 到 24 是 P2 口输出自上而下点亮的驱动数据，第 25 到第 32 是 P3 口输出自下而上点亮的驱动数据；第 33、35、…51 是 P0、P1、P2 和 P3 口均输出全 1，第 34、36、…52 是 P0、P1、P2 和 P3 口均输出全 0；第 53 到 60 是 P0、P1、P2 和 P3 口均输出自上而下点亮的驱动数据。

上述的控制可以判断显示状态数，根据其不同取值执行不同的程序代码，会

发现共有 7 个分支，这是一种多分支机构，可以用开关语句来实现。C 语言提供了一种用于多分支选择的 switch 语句， 其一般形式为：

```
switch(表达式){
        case 常量表达式 1:  语句 1;
        case 常量表达式 2:  语句 2;
        …
        case 常量表达式 n:  语句 n;
        default:语句 n+1;
}
```

其语义是：计算表达式的值。 并逐个与其后的常量表达式值相比较，当表达式的值与某个常量表达式的值相等时， 即执行其后的语句，然后不再进行判断，继续执行后面所有 case 后的语句。如表达式的值与所有 case 后的常量表达式均不相同时，则执行 default 后的语句。

2、主函数流程图

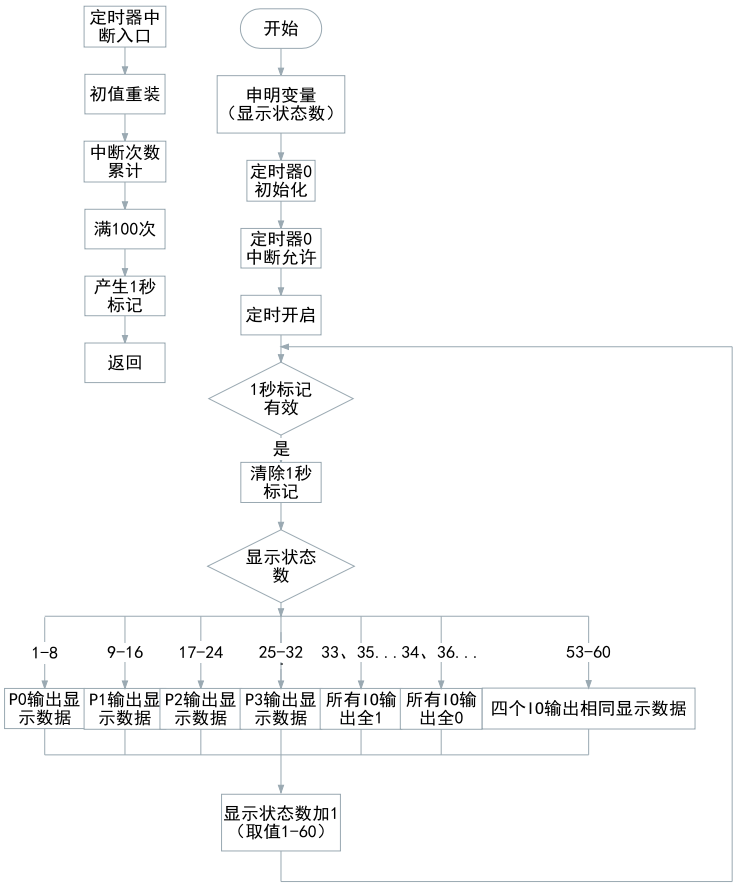

图 1- 74　　LED 矩阵广告牌流程图

3、参考程序

LED 矩阵广告牌 LEDGGP.C

```
#include <reg51.h>
unsigned char DataForLED[]={0xfe,0xfd,0xfb,0xf7,0xef,0xef,0xdf,0xdf,0xbf,0x7f};//定义数组存放LED 驱动数据
#define FOSC 11059200L//11.0592MHz
#define X 10//定时值 10ms
#define TXMS (65536-FOSC*X/12/1000);//16 位定时器初值计算
void T0int()
{
    TMOD=1;//16 位定时方式
    TH0=TXMS>>8;
    TL0=TXMS;//初值设置
    ET0=1;
    EA=1;//中断允许
    TR0=1;//定时器 0 开启
}
bit F_1S=0;// 1 秒间隔标记
unsigned char count;
void Timer0() intrrupt 1
{
    TH0=TXMS>>8;
    TL0=TXMS;//初值重装
    count++;
    if(count>99)
    {
        F_1S=1;// 1 秒间隔标记有效
    }
}
void main()//主函数
{
    unsigned char n=1;
    TOint();//定时器中断初始化
    while(1)
    {
```

```
        if(F_1S)//判断 1 秒标记
        {
            F_1S=0;// 清除 1 秒标记
            switch(n)//开关（LED 广告牌显示状态数）
            {
                case 1:
                case 2:
                case 3:
                case 4:
                case 5:
                case 6:
                case 7:
                case 8:P0=DataForLED[n-1];break;//P0 口 LED 自上而下点亮

                case 9:
                case 10:
                case 11:
                case 12:
                case 13:
                case 14:
                case 15:
                case 16:P0=0xff;P1=DataForLED[16-n];break; //P1 口 LED 自下而上点亮

                case 17:
                case 18:
                case 19:
                case 20:
                case 21:
                case 22:
                case 23:
                case 24:P1=0xff;P2=DataForLED[n-17];break; //P2 口 LED 自上而下点亮

                case 25:
                case 26:
                case 27:
                case 28:
                case 29:
```

```
case 30:
case 31:
case 32:P3=0xff;P3=DataForLED[32-n];break; /P3 口 LED 自下而上点亮


case 33:
case 35:
case 37:
case 39:
case 41:
case 43:
case 45:
case 47:
case 49:
case 51:P0=P1=P2=P3=0xff;break;//所有 IO 全 1，LED 全灭

case 34:
case 36:
case 38:
case 40:
case 42:
case 44:
case 46:
case 48:
case 50:
case 52:P0=P1=P2=P3=0;break; //所有 IO 全 0，LED 全亮


case 53:
case 54:
case 55:
case 56:
case 57:
case 58:
case 59:
case 60:P0=P1=P2=P3=DataForLED[n-53];break;//依次点亮整行
```

```
                    default:
                        break;
                }
                n++;//显示状态数累加
                if(n>60)n=1; //显示状态数取值控制
            }
        }
}
```

4、程序说明

主函数死循环中核心的结构是多分支结构，由 switch 语句实现。

“break;”的作用是跳出当前语句块，程序中用于跳出开关语句块。

“P2=DataForLED[n-17];”当 n=17 到 24 时能将数组元素从 0 下标元素开始依次取到 7 下标元素，实现第三列 LED 自上而下依次点亮。

同样的“P0= DataForLED[n-1];”也能将数组元素从 0 下标元素开始依次取到 7 下标元素，实现第一列 LED 自上而下依次点亮。

“P3=DataForLED[32-n];”当 n=25 到 32 时能将数组元素从 7 下标元素开始依次取到 0 下标元素，实现第四列 LED 自下而上依次点亮。

同样“P1=DataForLED[16-n];”也能将数组元素从 7 下标元素开始依次取到 0 下标元素，实现第二列 LED 自下而上依次点亮。

5、任务实施

5.1 建立工程

打开 keil 软件，通过菜单“Project”，新建立一个工程”new Project”项目 LEDGGP，然后再新建一个文件名为 LEDGGP.C 的源程序文件，将上面的参考程序输入并保存。然后将 LEDGGP.C 文件添加入本项目中。

5.2 编译并生成 HEX

单击  ”Build target”按钮，对源程序进行编译和链接，产生 HEX 文件。

5.3 烧录芯片

打开 STC-ISP 下载软件，选中单片机型号为“STC90C52RC”,选择最低波特率

为 2400 最高波特率为 115200（为默认值一般不需修改）。点击打开程序文件按钮，在刚才建立的 LEDGGP 工程文件夹中生成的 LEDGGP.HEX 文件。然后点击“下载/编程”按钮。最后使用串口线连接 PC 与主机模块，然后给主机模块通电，等待电脑端串口与单片机握手成功。就能把程序载入到单片机中去了。

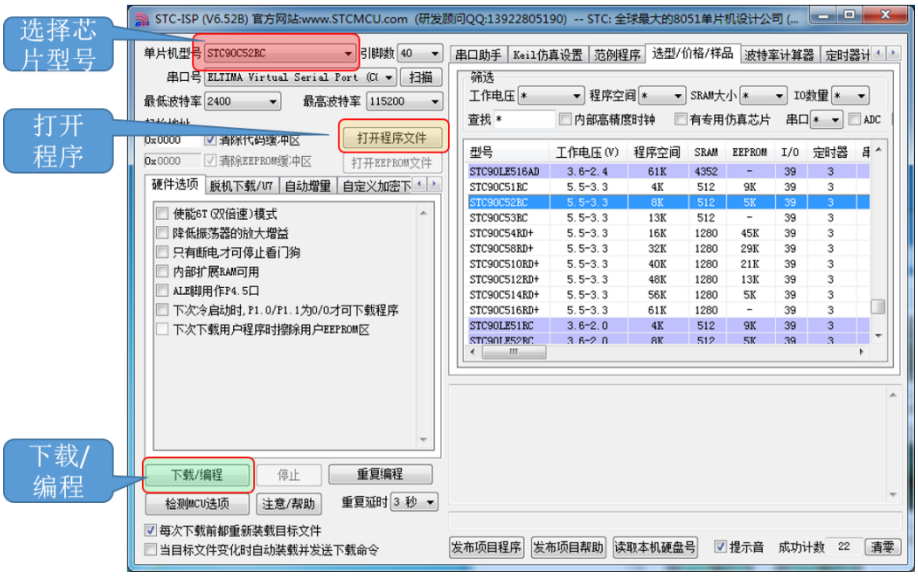

图 1- 75　STC 单片机程序烧写示意图

5.4　硬件调试

程序下载完成后，把单片机最小系统接入电路中，调试电路使其能正常工作，使广告牌实现预先设想的灯光效果。

四、任务评价

表 1- 16　任务七完成情况评价表

项目名称			评价时间		年　月　日		
小组名称		小组成员					
评价内容	评价要求	权重	评价标准	学生自评得分	小组评价得分	教师评价得分	合计
职业与安全意识	1．工具摆放、操作符合安全操作规程；2．遵守纪律、爱惜设备和器材、工位整洁；3．具有团队协作精神。	10%	好（10） 较好（8） 一般（6） 差（<6）				
模块的布局和布线工艺	1．模块布局合理，模块的选择应符合模块的要求。	15%	好（15） 较好（12） 一般（9） 差（<9）				
项目功能测试	1．编写的程序能成功编译；2．程序能正确烧写到芯片中；3．实现 3 种灯效。	60%	好（60） 较好（45） 一般（30） 差（<30）				
问题与思考	罗列其它形式的显示效果；	15%	好（15） 较好（12） 一般（9） 差（<6）				
教师签名			学生签名			总分	
项目评价=学生自评（0.2）+小组评价（0.3）+教师评价（0.5）							

五、知识拓展

程序中使用了 switch 现实多分支结构，其实用 if-else 嵌套结构也可以实现。switch 中一一列举所有 n 的 60 种取值；而 if-else 嵌套结构就不需要如此，这种结构仅需要将取值分成六段：1-8、9-16、17-24、25-32、33-52、53-60。

if-else 嵌套控制 IO

```
if(n<9)
{
	P0=DataForLED[n-1];
}
else if(n<17)
{
	P0=0xff;P1=DataForLED[16-n];
}
else if(n<25)
{
	P1=0xff;P2=DataForLED[n-17];
}
else if(n<33)
{
	P3=0xff;P3=DataForLED[32-n];
}
else if(n<53)
{
	P0=P1=P2=P3=n&1?0xff:0;//n 为奇数时输出 0xff，否则输出 0
}
else if(n<61)
{
	P0=P1=P2=P3=DataForLED[n-53];
}
```

六、思考与练习

练习:实现其他显示效果如中间向两边，两边向中间。

练习：将 32 只 LED 中外围的 20 只灯组成 A 组，剩下的 12 只组成 B 组，A 组顺时针依次点亮（从 DS1 开始），B 组逆时针依次点亮（从 DS6 开始）。