STC 单片机教学系统编程
指导书 V1.0

目 录

1. 前言 ····· 1

2. 硬件电路介绍 ····· 2

3. 开发环境介绍 ····· 3

3.1 使用 KEIL C51 编译程序 ····· 3

3.2 程序烧写 ····· 7

4. 上机实验 ····· 10

4.1 数字输入输出实验（3 学时） ····· 10

4.1.1 I/O 基础知识 ····· 10

4.1.2 电路原理分析 ····· 11

4.1.4 程序示例：跑马灯的简单实现 ····· 12

4.1.5 实验任务 ····· 13

4.2 中断系统、定时器/计数器（3 学时） ····· 14

4.2.1 单片机的中断 ····· 14

4.2.2 定时器中断系统 ····· 15

4.2.3 程序示例：定时器实现跑马灯 ····· 17

4.2.4 实验任务 ····· 18

4.3 模拟接口 A/D 实验（3 学时） ····· 19

4.3.1 模数转换原理 ····· 19

4.3.2 程序示例：A/D 电压检测 ····· 20

4.3.4 实验任务 ····· 22

4.4 单片机综合实验（3 学时） ····· 23

5. 总结 ····· 25

6. 附录 ····· 26

6.1 中断寄存器 ····· 26

6.2 中断使能寄存器 ····· 26

6.4 中断优先寄存器 ····· 28

6.5 定时器/计数器中断控制寄存器 ····· 29

6.6 与 ADC 相关的寄存器 ····· 29

6.7 最小系统原理图 ····· 32

6.8 多功能板原理图…… 33

1. 前言

单片机是将运算核心、随机存储器、只读存储器、多种 I/O 端口、中断系统和定时器/计数器等功能模块集成到一块集成电路上构成的微型计算机系统，由于其体积小、结构简单、可靠性高的优点，被广泛应用于仪器仪表、家用电器、医用设备、航空航天、专用设备的智能化管理及过程控制等领域。因此单片机原理与接口技术课是一门应用性极强的基础课程, 是进一步学习其它控制类、电子类专业课程的前提和基础。

在进行上机实验之前，应当首先具备一定的模拟、数字电路基础理论知识，掌握 C 语言程序设计的基本技巧，而且对 51 内核单片机的性能和编程方式有初步认识。在实验过程中，可以将本书与课程教材配合学习，相互参考，并通过认真完成每次实验的编程任务，更快更好地掌握单片机应用知识和开发技能。

本书安排的实验内容均针对以 STC IAP15W4K58S4 为核心的实验电路板进行设计。全书在内容上可分为 5 个部分，第一部分介绍实验电路的基本特性和功能模块；第二部分介绍程序开发环境的配置与程序烧写流程；第三部分详细介绍了本学期 4 次实验课程的主要任务，其中每次实验都首先从 51 内核的基本原理出发，简单回顾相应的理论知识，并附上简短的学习例程。完成实验任务时应尽可能独立思考，并尝试在题目要求的基础上考虑程序流程更简单、执行效率更高的实现方法。指导书在附录中列出了部分可能用到的数据信息，供编写程序时查阅参考。

由于时间仓促和水平所限，实验指导书中可能存在一些错误和不妥之处，敬请批评指正。

2017 年 11 月

2. 硬件电路介绍

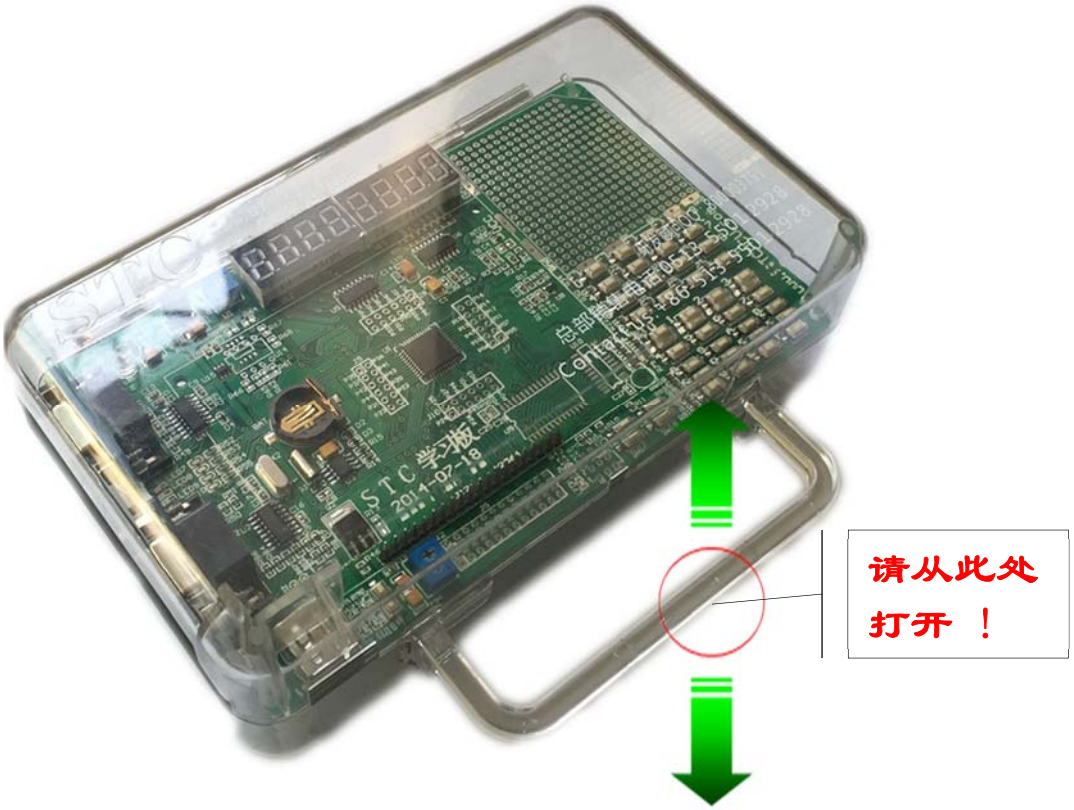


图 2.1 实验箱外观

在进行上机实验之前，**请双手握住实验箱的把手（上图红圈标示处），分别向两边推拉**。为避免锈蚀和染尘，每次上机实验结束后，应将电路板妥善放置在实验箱内，关闭保护外壳并扣紧。

实验电路板由 STC15W4K32S4 主控芯片和个外围电路模块构成，各模块的位置分布见图 2.2，主要功能模块包括：

- 8 位红色共阴数码管
- 红外接收、发送端口
- ADC 分压检测
- RTC 实时时钟（PCF8563）
- 中断 INT0、INT1 测试
- 外部 2.5V 参考电压源（TL431 或 CD431）
- 行列式矩阵扫描按键、普通按键
- 自定义实验万能板
- 热敏电阻测温接口（ADC）

- 使用 PWM 实现 DAC 输出端口
- 掉电检测、掉电唤醒测试

此外，实验电路板还提供以下数据通信接口和协议转换芯片：

- 外部扩展 RAM 接口
- LCD12864 接口
- USB、Micro-USB 接口
- 串口接口、RS232 电平转换芯片

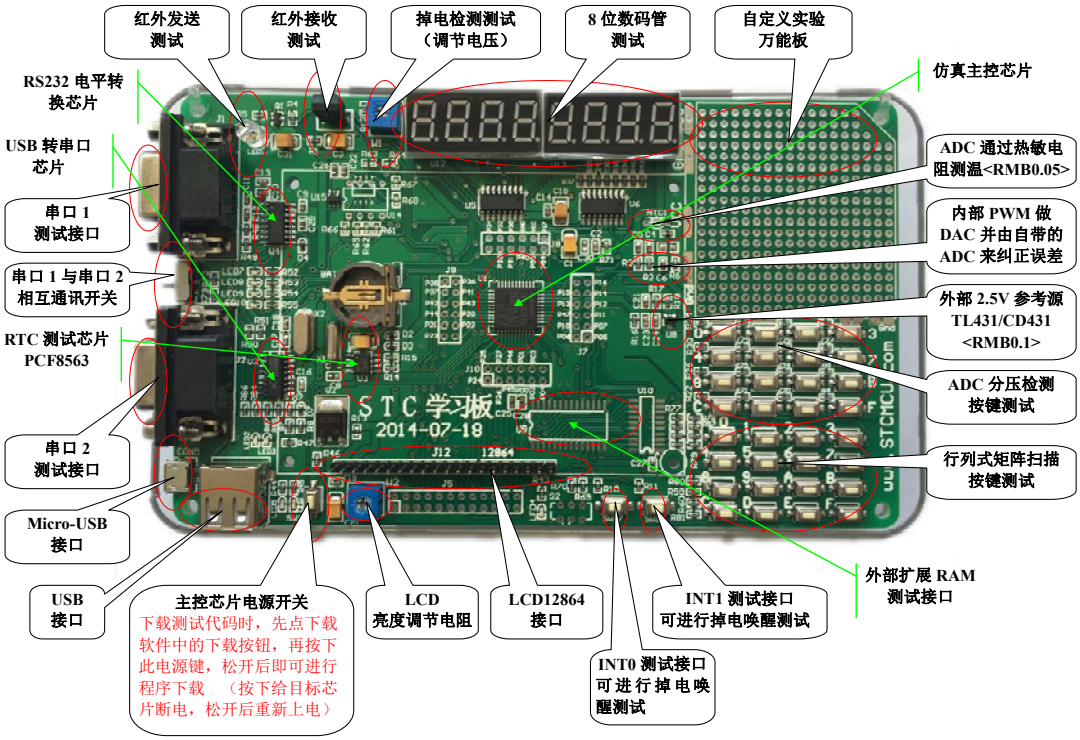


图 2.2 实验电路板布局图

实验电路板上设有主控芯片电源开关，该按钮被按下时主控芯片断电，松开后芯片重新上电。因此，在向实验电路板的主控芯片烧写程序时，应当首先点击烧写软件中的下载按钮，再按下主控芯片电源开关，松开后即可开始程序烧写。

3. 开发环境介绍

3.1 使用 Keil C51 编译程序

单片机开发通常使用 C 语言或汇编语言编写程序，并使用编译器将代码编译生成单片机可执行的二进制代码，再通过烧写软件将其写入单片机片内的 Flash 存储器，由单片机重新上电后自动执行。

Keil uVision4 是由美国 Keil Software 公司发布的单片机程序集成开发环境系列，其中 Keil C51 可用于进行 51 内核单片机的 C 语言程序设计。Keil 提供丰富的库函数和功能强大的集成开发调试工具，而且生成的目标代码效率高，大部分语句生成的汇编代码十分紧凑，容易理解。在开发大型软件时更能体现高级语言的优势。

实验室的计算机已经安装了 Keil C51 集成开发环境，下面介绍创建工程项目的操作步骤。

第一步 启动开发环境

双击 Keil C51 程序的图标（如图 3.1）运行 Keil C51 程序。

图 3.1 Keil C51 图标

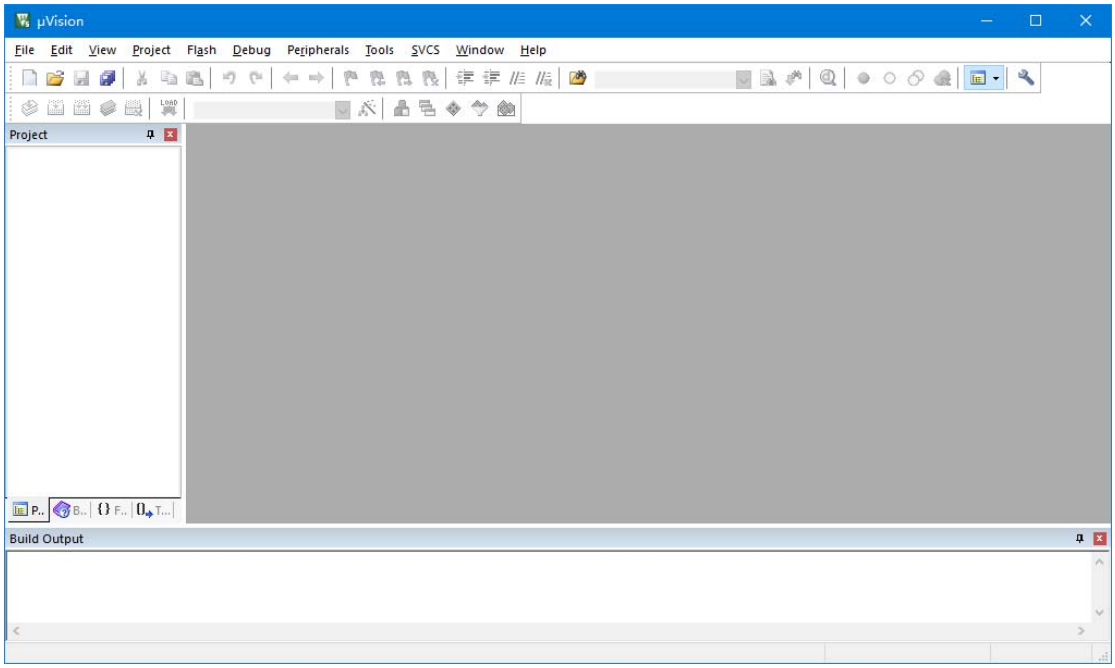


图 3.2 Keil C51 主界面

第二步 新建项目

使用 Keil 编写程序需要首先新建一个项目，单击 **Project→New Project…**（如图 3.3），在弹出的对话框中，选择新项目存放的地方及文件名。

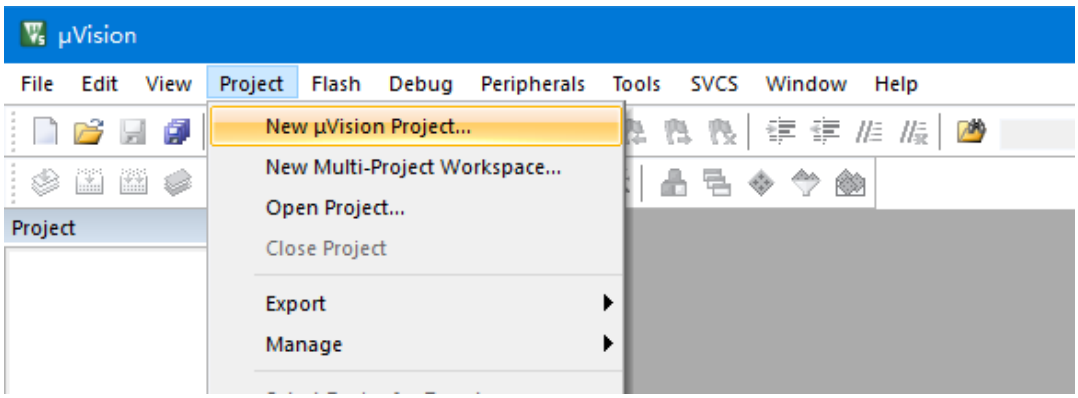


图 3.3 “项目”菜单

在弹出的目标设备选项框中，选择 **STC MCU Database**，找到 **STC15W4K32S4**（见图 3.4），弹出的对话框点击“是”，完成一个新项目的创建。

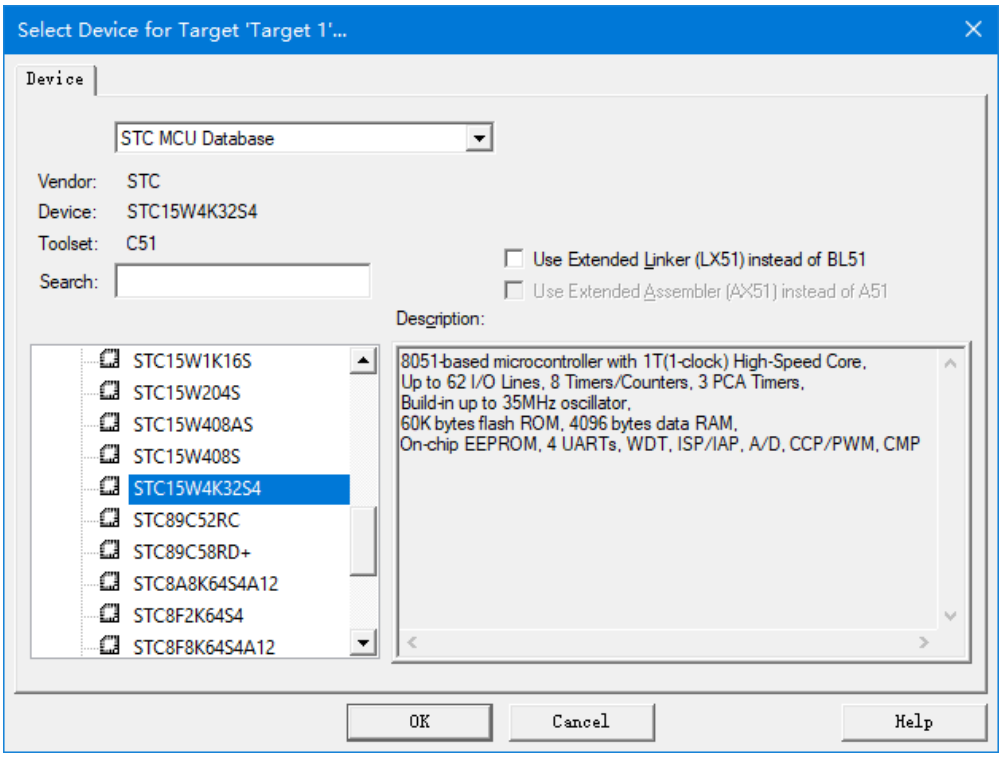


图 3.4 目标设备选项框

第三步 修改项目属性

通过菜单 **Project→Options for Target ’target1’**，或者快捷键 **Alt+F7**，打开项目属性对话框。

在“Output”选项卡中，勾选“Create HEX File”选项（见图 3.5）。选中该项后，Keil C51 将在项目编译完成后自动生成.hex 格式的目标文件，用于程序烧写，点击“OK”按钮保存设置。

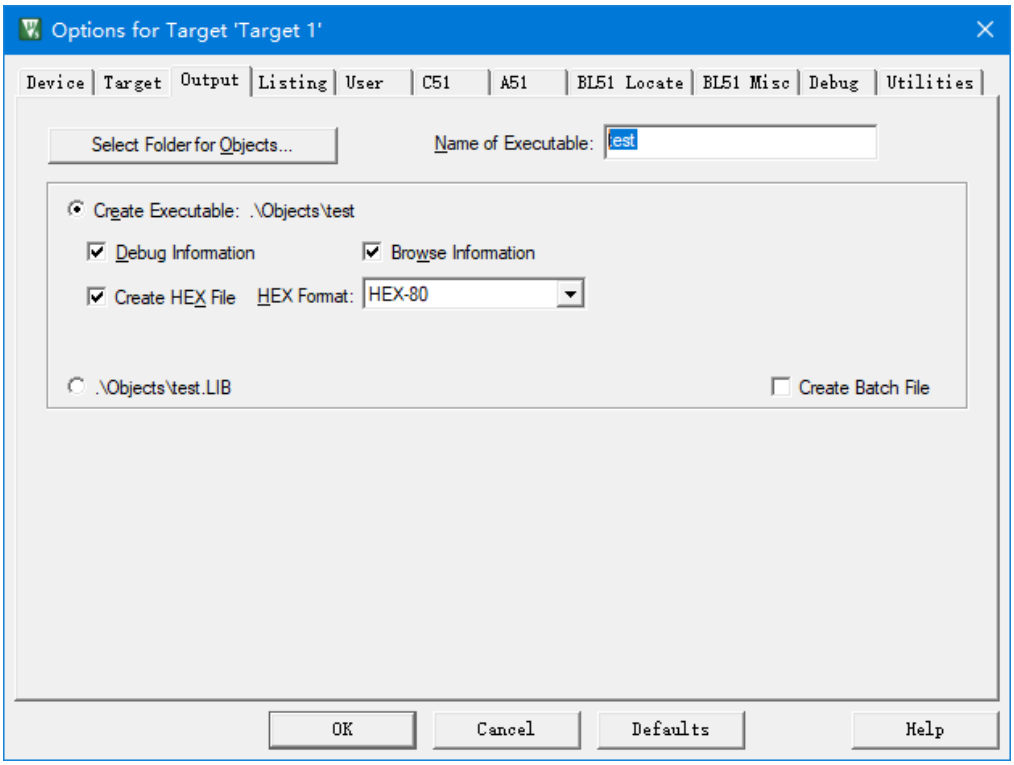


图 3.5 输出选项设置页面

第四步 添加程序文件

在主界面左侧的 project 侧边栏中，对 **Source Group 1** 单击右键，选择 **Add New Item to Group ‘Source Group 1’**。

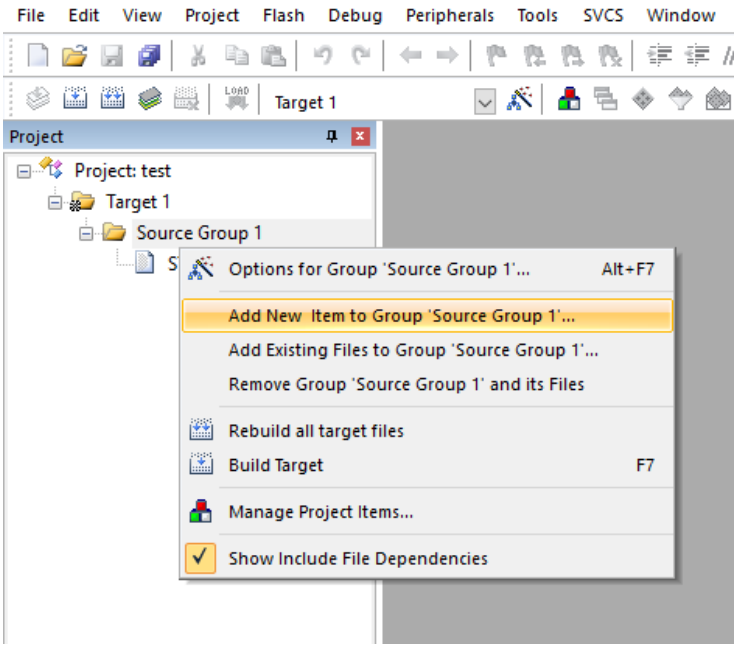


图 3.6 Source Group 右键菜单

在弹出的窗口中（见图 3.7），选择新建的文件类型，并在下方的 Name 输入框中输入文件名（不带文件名后缀），点击 **Add** 按钮即可新建文件并添加到当前项目。

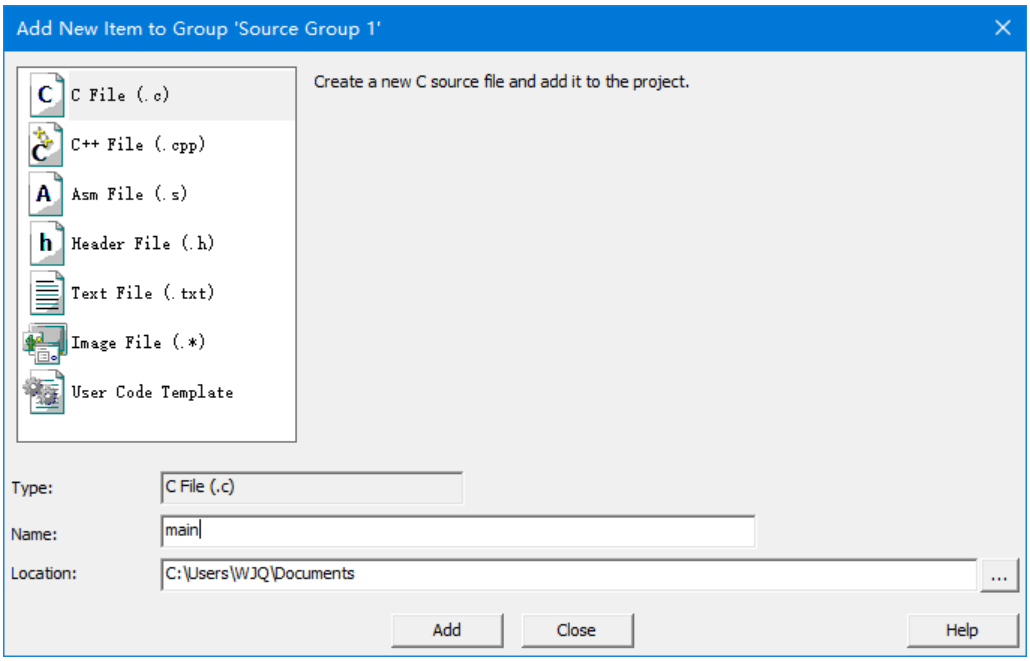


图 3.7 文件新增窗口

第五步 编译

当程序代码编写完成后，需要将 C 语言程序代码编译为二进制代码。通过菜单 **Project→Build target**，或者单击工具栏上的按钮，或者使用快捷键 **F7**，即可执行编译命令。如果编译过程出错，相应的错误信息会出现在下方的编译输出框。

如果编译出错，应当根据提示，修改源文件，然后再次尝试编译，直至成功生成.hex 文件。编译成功后，可将生成的.hex 文件通过烧写软件下载到单片机中。

3.2 程序烧写

程序编译完成后，需要借助专用的烧写程序，通过串口通讯的方式，按照一定读写时序向芯片发送清除、烧写、校验命令，对单片机片上的 Flash 程序储存区进行烧写。下面介绍使用 STC-ISP（V6.86F）进行程序烧写的具体步骤。

在烧写程序前，请使用 USB 数据线，将实验电路板与个人电脑的 USB 端口相连。然后打开 stc-isp.exe 程序（图标见图 3.8），图 3.9 为程序运行后的界面。

图 3.8 烧写程序图标

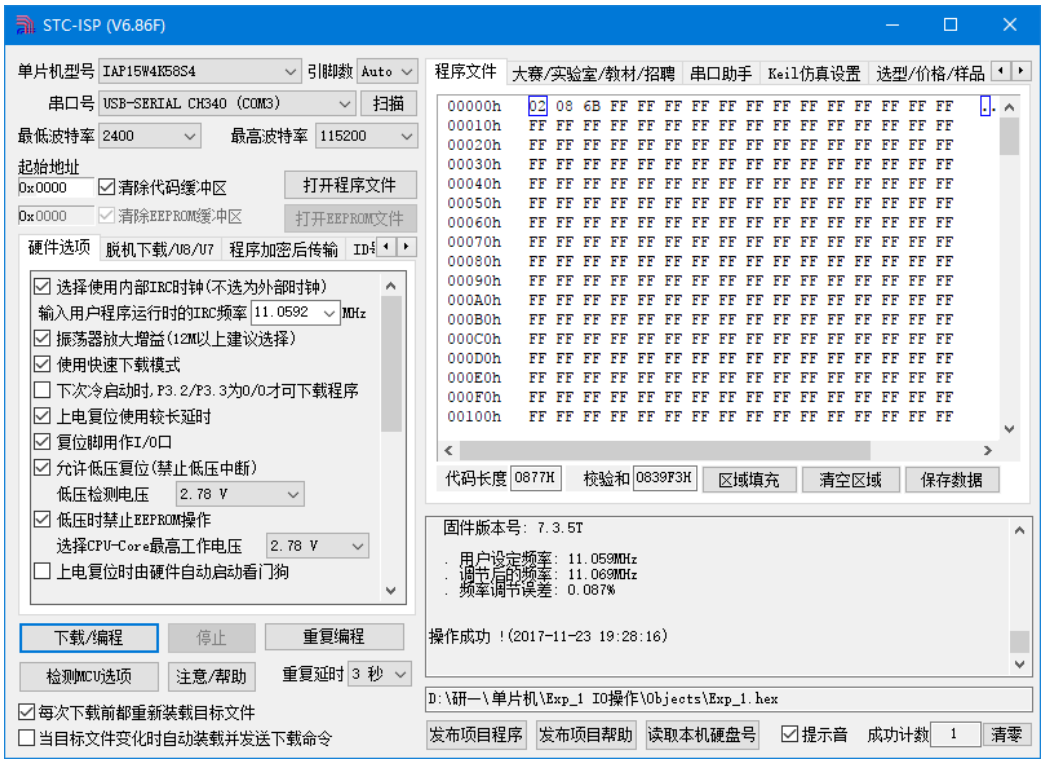


图 3.9 烧写程序界面

第一步 参数配置

在程序左上角的**单片机型号**选项，选择“**IAP15W4K58S4**”，并根据实际情况选择合适的**通讯串口**，可使用 Windows 系统的设备管理器查看当前使用的端口号。设置**程序运行 IRC 频率**为单片机采用的工作晶振频率 **11.0592MHz**，其它设置维持默认状态即可。

第二步 读取二级制代码文件

点击位于程序左侧的中间位置的“**打开程序文件**”按钮，在弹出的通用对话框中选择使用 Keil C51 编译生成的.hex 文件。文件打开后，可以再右侧区域查看程序文件的二进制编码和代码长度。

第三步 程序烧写

点击程序左下角的“**下载/编程**”按钮，然后**按下主控芯片电源开关**（见图 2.2），松开后即可开始程序烧写，烧写结束后将在右下方的文本框中显示“操作成功”的提示。

如果程序无法正确烧写，请按下列步骤进行检查：

(1) 重复复位上电操作，适当延长主控芯片电源开关的按下时间；

(2) 检查 USB 数据线两端是否正常连接到电脑和电路板；

(3) 检查烧写软件的各项设置是否正确（晶振频率、串口通讯端口等）；

如果执行经过以上排查过程后故障仍未消除，请向助教反映或更换实验仪器。

4. 上机实验

本学期将安排 4 次上机实验，共计 12 学时。前 3 次实验分别围绕输入输出端口、中断与定时/计时器、数模/模数转换设计实验内容，最后一次上机实验前应复习回顾相关的理论基础，并仔细阅读指导书中的实验步骤和注意事项，并根据实验要求编写程序。在完成实验任务时，建议首先查看实验板电路图，熟悉单片机引脚与外围电路器件之间的连接关系，并在必要时翻阅数据手册上的寄存器信息。

4.1 数字输入输出实验（3 学时）

4.1.1 I/O 基础知识

STC15 系列单片机可工作在弱上拉（准双向口）、强上拉（推挽输出）、高阻输入（悬空）、开漏 4 种工作模式，通过 PxM0 和 PxM1 两个控制位决定（详细设置见表 4.1）。在上电复位后，默认设置为弱上拉（准双向口）工作模式，每个 I/O 引脚的驱动能力可达 20mA。

表 4.1 STC15 系列单片机 I/O 工作模式设置

PxM1[7:0]	PxM0[7:0]	I/O 模式
0	0	准双向口：灌电流 20mA、拉电流 270μA
0	1	推挽输出：强上拉，可达 20mA，需加限流电阻
1	0	高阻状态
1	1	开漏输出（OD）：内部上拉电阻断开，需外加上拉电阻

下面简要介绍准双向口工作模式内部连接原理，电路原理图见图 4.1。

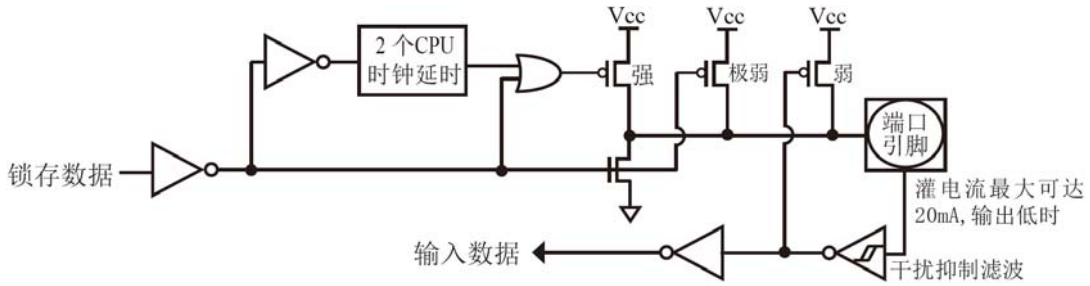


图 4.1 准双向口工作模式内部连接

准双向口（弱上拉）输出类型可同时用作输出和输入功能而无需重新配置，

这是因为当端口输出设为高电平时驱动能力很弱，允许外部电路将其拉低。当引脚输出为低电平时，驱动能力很强，可吸收相当大的电流。准双向口有 3 个上拉晶体管适应不同的需要。

在 3 个上拉晶体管中，有 1 个上拉晶体管称为“弱上拉”，当端口寄存器为 1 且引脚本身也为 1 时打开。此上拉提供基本驱动电流使准双向口输出为 1。如果一个引脚输出为 1 而由外部装置下拉到低电平时，弱上拉关闭而“极弱上拉”维持闭合状态，为了把引脚强拉为低电平，外部装置必须有足够的灌电流能力使引脚上的电压降到门槛电压以下。对于 5V 单片机，“弱上拉”晶体管的电流约 250uA；对于 3.3V 单片机，“弱上拉”品体管的电流约 150uA。

4.1.2 电路原理分析

实验电路板上一共设置了 4 盏可供开发者自行控制的 LED 指示灯（其余指示灯均用于各类功能模块的工作状态指示），编号分别为 LED7~LED10，与单片机引脚 P1.7、P1.6、P4.7、P4.6 相连，连接关系见下图。

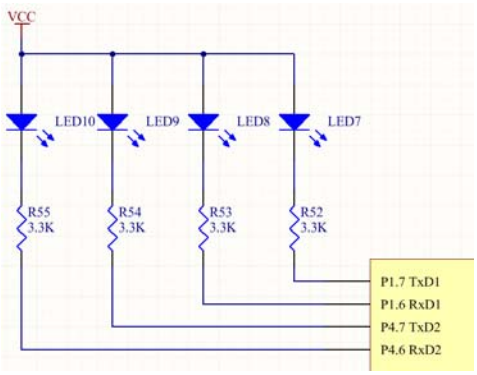


图 4.2 实验电路板 LED 接线图

在实验电路中（引脚电路见图 4.2）使用引脚直接驱动 LED 时，可以设置为弱上拉模式，当引脚输出低电平时，LED 经串联电阻分压后获得 1.7V 左右的压降，指示灯亮起。

需要注意的是，实验板使用的 STC15 系列单片机引脚最高灌电流为 20mA，与 STC89 系列单片机的 6mA 限值相比有了明显提升，可以驱动较大的电流，足够直接与二极管相连。

实验板上还配有 3641AS 红色 4 位共阴数码管，一共需要引出 16 组连线用于控制 8 位段码和 8 位位码。由于单片机引脚资源有限而且便于布线，实验板使用了两片级联的 74HC595 芯片（3 态 8 位串联输入并联输出移位寄存器），其串联输入端与单片机 P4.0 相连，并联输出端的高 8 位作为位码，低 8 位作为段码，

连接关系图见下图。

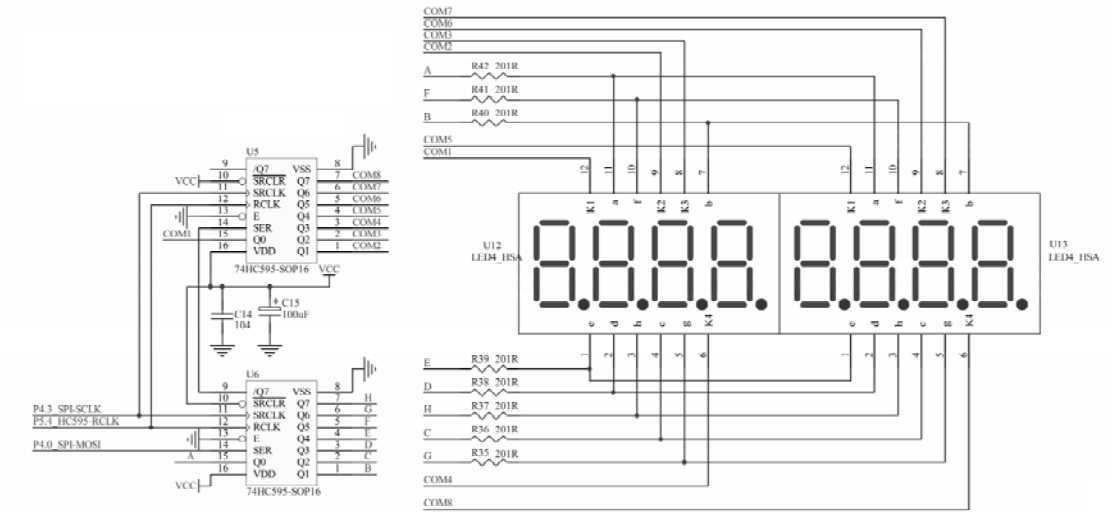


图 4.3 数码管电路原理图

4.1.4 程序示例：跑马灯的简单实现

本次实验主要涉及基本 I/O 口的输出功能，示例程序使用 P0 口的 4 个端口控制实验板上的 4 个指示灯，使之循环点亮。

程序分析：根据电路原理图中 LED 灯与单片机引脚的连接关系，可知 LED7、LED8、LED9、LED10 依次对应 P1.7、P1.6、P4.7、P4.6。为了实现四盏 LED 循环点亮，程序需要在主循环中依次对上述 4 个引脚输出低电平，点亮 LED，待延时结束后输出高电平，使灯光熄灭。

程序代码：

```
#define    MAIN_Fosc          11059200L     //定义主时钟
#include  "STC15Fxxxx.H"

void    delay_ms(u16 ms);

//  主函数
void main(void){

        //  设置为准双向口
        P0M1 = 0; P0M0 = 0;
        P1M1 = 0; P1M0 = 0;
        P2M1 = 0; P2M0 = 0;
        P3M1 = 0; P3M0 = 0;
        P4M1 = 0; P4M0 = 0;
        P5M1 = 0; P5M0 = 0;
        P6M1 = 0; P6M0 = 0;
        P7M1 = 0; P7M0 = 0;
```

```
    while(1){
        P17 = 0;
        delay_ms(500);

        P17 = 1;
        P16 = 0;
        delay_ms(500);

        P16 = 1;
        P47 = 0;
        delay_ms(500);

        P47 = 1;
        P46 = 0;
        delay_ms(500);

        P46 = 1;
    }
}

// 延时函数
void delay_ms(u16 ms){

    u16 i;
    do{
        i = MAIN_Fosc / 13000;
        while(--i)    ;
    }while(--ms);
}
```

4.1.5 实验任务

- **实验目的：**

  1） 熟悉基于 Keil C 的单片机编程环境以及程序开发流程；

  2） 了解基本 I/O 输入输出的硬件原理；

  3） 学习掌握单片机的基本 I/O 编程技术。

- **实验内容：**

1） 基本实验 1：循环扫描实验板上 2 个按键的状态，并在数码管上显示被按下按键的编号；

2） 基本实验 2：循环检测按键的状态，两个数码管显示按键按下的次数。初始状态数码管显示“00”，按键每按下一次显示数字加 1，直到增加到“20”的时候，当按键被再次按下时，显示数字清零；

3） 基本实验 3：循环检测按键 17 和 18 的状态，两个数码管显示按键按下的次数。初始状态数码管显示“10”，按键 17 每按下一次显示数字减 1，按键 18

每按下一次显示数字加 1。如果增加到“20”的时候又按下按键 18，则显示数字清零，如果减小到“00”按下按键 17，显示数字变为“20”。

4.2 中断系统、定时器/计数器（3 学时）

4.2.1 单片机的中断

STC15W4K32S4 系列单片机提供了 21 个中断请求源，详细列表见表 4.2，其中常用的中断有外部中断 0（INT0）、定时器 0 中断、外部中断 1（INTl）、定时器 1 中断、串口 1 中断、AD 转换中断等。

外部中断 0（INTO）和外部中断 1（INT1）既可上升沿触发，又可下降沿触发。请求两个外部中断的标志位是位于寄存器 TCON 中的 IE0/TCON.1 和 IE1/TCON.3。当外部中断服务程序被响应后，中断标志位 IEO 和 IE1 会自动被清 0。外部中断 0（INT0）和外部中断 1（INT1）还可以用于将单片机从掉电模式唤醒。

表 4.2 中断触发表

中断源	中断名称	触发行为	中断号
INT0	外部中断 0	（IT0=1）：下降沿；（IT0=0）：上升沿和下降沿均可	0
Timer0	定时器 0 中断	溢出	1
INT1	外部中断 1	（IT0=1）：下降沿；（IT0=0）：上升沿和下降沿均可	2
Timer1	定时器 1 中断	溢出	3
UART1	串口 1 中断	发送或接收完成	4
ADC	AD 转换中断	A/D 转换完成	5
LVD	低电压中断	电源电压下降到低于 LVD 检测电压	6
UART2	串口 2	发送或接收完成	8
SPI	SPI 中断	数据传输完成	9
$\overline{\text{INT2}}$	外部中断 2	下降沿	10
$\overline{\text{INT3}}$	外部中断 3	下降沿	11
UART3	串口 3 中断	发送或接收完成	17

UART4	串口 4 中断	发送或接收完成	18
Timer3	定时器 3 中断	溢出	19
Timer4	定时器 4 中断	溢出	20
Comparator	比较器中断	比较器比较结果由低变高或高变低	21

单片机每个中断源是否允许触发中断是由**中断允许寄存器 IE** 控制的，该寄存器可按位寻址，其格式可参见附录。IE 寄存器中的 **EA** 位是总中断使能控制位，当 EA=1 时，单片机内核开放中断。此外，各中断源还受各自的**中断允许控制位**控制，例如计时器 0 中断控制位为 ET0。因此在使用中断前，应当将上述两种控制位都设为 1。

4.2.2 定时器中断系统

实验板使用的 STC15 系列单片机具有 T/C0、T/C1、T/C2、T/C3、T/C4 共 5 个 16 位定时/计数器，其中每个定时器有模式 0-模式 3 四种工作模式，其中最常用的是工作模式 0。与 STC89 系列单片机相比，STC89 系列只有 3 个定时/计数器。而且 STC15 系列的工作模式 0 是 16 位自动重载模式，而 STC89 系列是 13 位自动重载模式。下面分别介绍这四种工作模式。

- **工作模式 0：16 位自动重载**

图 4.4 为 STC15 系列单片机 T/C0 在工作模式 0 下的逻辑电路图。当 GATE=0（TMOD.3）时，只要 TR0=1，定时器就开始计数。GATE=1 时，允许从 INT0 引脚输入的外部信号作为 T/C0 的时钟脉冲，使用此功能可实现脉冲宽度测量。

当$C/\overline{T}$=0 时，多路开关连接到系统时钟分频器输出，T/C0 对内部系统时钟计数，T/C0 作为定时器使用。当$C/\overline{T}$=1 时，多路开关连接到外部脉冲输入 P3.4/T0，T/C0 作为计数器使用。

STC15 系列单片机的定时器有 12T 模式和 1T 模式两种计数速率。在 12T 模式下，定时器每 12 个时钟周期加 1，与传统 8051 单片机相同；1T 模式下，每个时钟周期加 1，速度是传统 8051 单片机的 12 倍。T0 的速率由特殊功能寄存器 AUXR 中的 T0x12 决定，如果 T0x12=0，T0 则工作在 12T 模式，否则工作在 1T 模式。

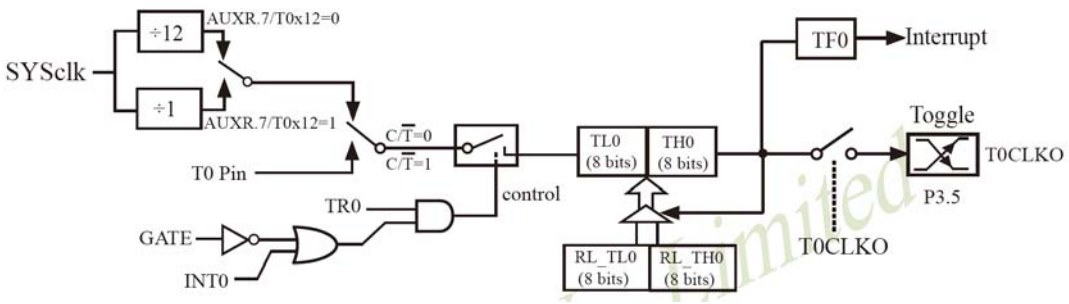


图 4.4 定时器/计数器 0 的模式 0 逻辑电路图

- **工作模式 1：16 位单次计数**

工作模式 1 与工作模式 0 的计数空间相同，但在工作模式 1 下，定时/计数器发生溢出后不会自动装载。计数空间由存放低 8 位的 TL 寄存器和存放高 8 位数据的 TH 寄存器组成。

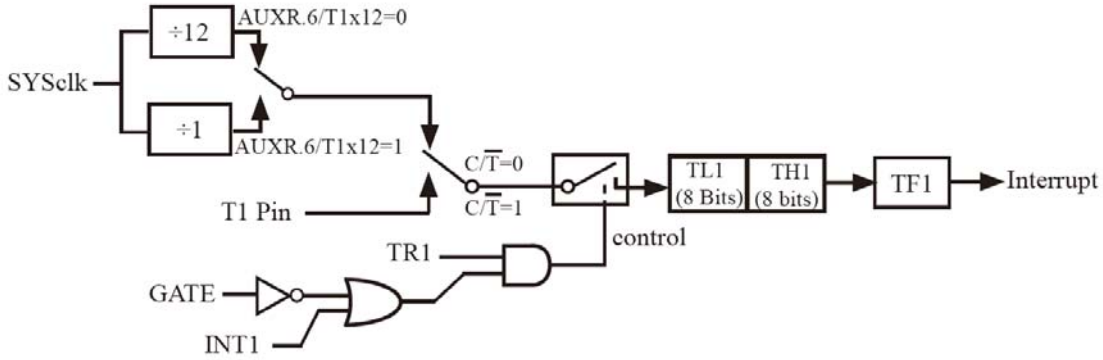


图 4.5 定时器/计数器 1 的模式 1 逻辑电路图

- **工作模式 2：8 位自动重载**

模式 2 的计数空间只使用了 TL 寄存器的 8 位，因此每个循环周期最多只能计数 256 次。TH 寄存器作为预置数寄存器，当溢出发生后，单片机将 TH 寄存器中的预置数通过硬件电路存入 TL 寄存器。

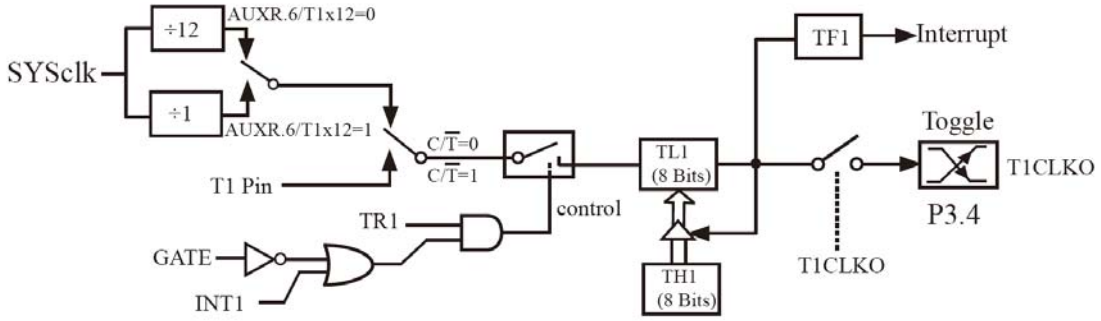


图 4.6 定时器/计数器 1 的模式 2 逻辑电路图

- **工作模式 3：16 位自动重载且不可屏蔽中断**

（略）

4.2.3 程序示例：定时器实现跑马灯

定时器中断实验：本次实验主要学习定时器中断程序的编写方法。示例程序采用定时器中断的方法，实现跑马灯程序。

程序分析：程序实现原理与实验 1 基本一致，但延时改为使用定时器中断实现。通过配置令定时/计数器 0 工作在 0 模式（16 位预置数自动重装），当定时/计数的计数溢出发生时，会暂时停止当前进程并转入中断程序。与此同时，通过内部计数器通过硬件电路设置为预置数，开始下一轮定时/计数。通过合理设置溢出周期的长度，可实现 1 秒左右的延时。

```
#define     MAIN_Fosc           11059200L//定义主时钟
#include    "STC15Fxxxx.H"

u8 num, count;

// 主函数
void main(){
        u16 TC0_Count;

        //设置为准双向口
        P1M1 = 0; P1M0 = 0;
        P4M1 = 0; P4M0 = 0;

        //配置定时器 0
        TMOD = 0x00; // 定时器 0 工作在 0 模式：16 位自动重装
        TCON = 0x10; // 打开定时器中断 0
        AUXR &= ~0x80;      // 打开 12 分频
        TC0_Count = MAIN_Fosc / 12 / 100; //溢出周期为 100Hz
        TH0 = (u8)((65536UL - TC0_Count) / 256);
        TL0 = (u8)((65536UL - TC0_Count) % 256);

        ET0 = 1;      // T/C0 中断使能
        EA = 1;     // 中断总使能置位
```

```
        num = 0; count = 0;
        while(1);
}

// 中断响应程序
void on_TC0_Overflow(void) interrupt TIMER0_VECTOR{

        if(count++ < 100) return;
        else count = 0;

        switch(num){
        case 0: P46 = 1; P17 = 0; break;
        case 1: P17 = 1; P16 = 0; break;
        case 2: P16 = 1; P47 = 0; break;
        case 3: P47 = 1; P46 = 0; break;
        }

        num = (num + 1) % 4;
}
```

4.2.4 实验任务

- **实验目的：**

  1） 熟悉 Keil C 整体编程环境以及 MCU 程序开发流程；

  2） 了解的中断系统、定时器/计数器工作原理；

  3） 学习中断系统、定时器/计数器功能的编程技术。

- **实验内容：**

1） 基本实验 1：使用定时器产生 1 秒钟的时间间隔，循环点亮 4 盏 LED。初始状态下 4 盏 LED 熄灭，按下按键 17 后启动定时器，1 秒钟后第 1 盏 LED 点亮；再过 1 秒钟，前第 2 盏 LED 点亮；以此类推，第 4 秒全部熄灭，重复上述过程。在启动定时器的状态下，按下按键 18 定时器停止；再次按下按键 17，可重新启动定时器。

2） 基本实验 2：使用按键 17 来触发定时器，按下按键 17 后定时器开始工作，并由数码管实时显示定时器的溢出次数。溢出 10 次后，定时器停止。按键 17 再次按下后，重复上述过程，当溢出 100 次后清零，并重新开始计数；

3） 基本实验 3：使用定时器产生循环周期为 1 秒钟的定时器中断，使用中断控制 8 位数码管的显示内容。第 1 秒后，左侧第一位数码管显示 1；第 2 秒后，左侧两位数码管分别显示 12；第 3 秒后，左侧三位数码管分别显示 123,，以此类推，第 8 秒后，全部数码管熄灭，重复上述过程。

4.3 模拟接口 A/D 实验（3 学时）

4.3.1 模数转换原理

STC15 系列单片机 ADC 是逐次比较型 ADC，由多路选择开关、比较器、逐次比较寄存器、10 位 DAC、转换结果寄存器（ADC_RES 和 ADC_RESL）以及 ADCCONTR 构成。AD 转换开始后，单片机内部的 ADC 模块，从最高位（MSB）开始，按顺序对每一输入电压与内置 D/A 转换器输出进行逐次比较，使转换所得的数字量逐次逼近输入模拟量对应值，这种类型的 AD 转换器具有速度高，功耗低的优点。

图 4.7 是 STC15 系列单片机的 ADC 逻辑图，从图中可以看出，ADC0-7 的模拟量输入通过模拟多路开关传送到比较器，在比较器中将数/模转换器（DAC）提供的模拟量与输入的模拟量进行比较，将比较结果保存到逐次比较寄存器作为转换结果。A/D 转换结束后，最终的转换结果保存到结果奇存器 ADC_RES 和 ADC_RESL，同时，ADC 控制奇存器 ADCCONTR 中的 A/D 转换结束标志位 ADC_FLAG 置位，以供程序查询。

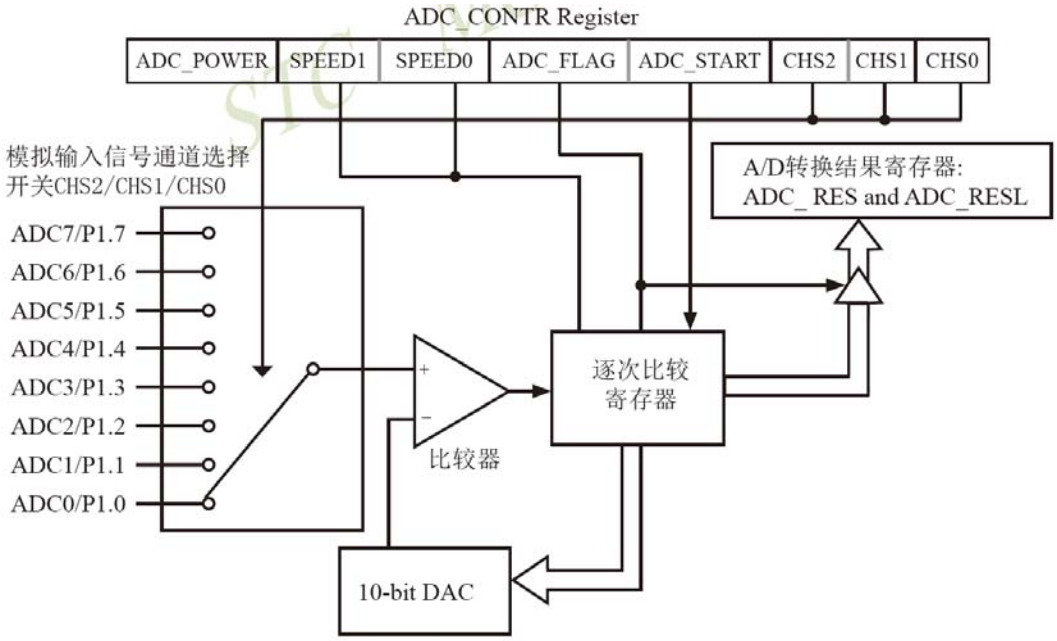


图 4.7 STC15 系列单片机 ADC 逻辑电路

如果设置 ADRJ=0，可以使用下面公式计算 10 位转换结果：

$$(\text{ADC_RES}[7:0], \text{ADC_RESL}[1:0]) = 1024 \times \frac{V_{in}}{V_{CC}}$$

如果设置 ADRJ=1，可以使用下面公式计算 10 位转换结果：

$$(\mathrm{ADC_RESL}[1:0], \mathrm{ADC_RES}[7:0]) = 1024 \times \frac{V_{in}}{V_{CC}}$$

式中，V_{in}为模拟输入通道的输入电压，V_{CC}为单片机电源电压（使用单片机电源电压作为模拟参考电压）。

4.3.2 程序示例：A/D 电压检测

下面给出使用单片机片上 ADC 模块测量工作电源电压 Vcc 的程序，经实际测试，输出数值约为 4.86，电压读数随 USB 接口的松紧变化而上下浮动。

程序分析：使用 ADC 通道 0 检测电源电压，并采用查询法逐次查询 ADC 转换结果，然后控制数码管最右侧三位显示电压数值。

```
#define MAIN_Fosc           11059200L
#include    "STC15Fxxxx.H"


#define DIS_DOT             0x20
#define DIS_BLACK  0x10
#define DIS_            0x11

#define P1n_pure_input(bitn)            P1M1 |=    (bitn),P1M0 &= ~(bitn)
#define LED_TYPE 0x00           //数码管类型, 0x00 共阴, 0xFF 共阳

#define     Timer0_Reload   (65536UL -(MAIN_Fosc / 1000))         //Timer 0  中断频率, 1000 次/秒

u8 code t_display[]={
        0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F,0x77,0x7C,0x39,0x5E,0x79,0x71, // 0-9 A-F
        0x00,0x40,0x76,0x1E,0x70,0x38,0x37,0x5C,0x73,0x3E,0x78,0x3d,0x67,0x50,0x37,0x6e, //全黑  - H J K L
N o P U t G Q r M y
        0xBF,0x86,0xDB,0xCF,0xE6,0xED,0xFD,0x87,0xFF,0xEF,0x46      //0. 1. 2. 3. 4. 5. 6. 7. 8. 9. -1
};
u8 code T_COM[]={0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80};            //位码

sbit   P_HC595_SER      = P4^0;   //pin 14      SER         data input
sbit   P_HC595_RCLK     = P5^4;  //pin 12      RCLk        store (latch) clock
sbit   P_HC595_SRCLK = P4^3;   //pin 11      SRCLK       Shift data clock

u8     LED8[8];            //显示缓冲
u8     display_index;    //显示位索引
bit    B_1ms;                      //1ms 标志

u8     msecond;
u16    Get_ADC10bitResult(u8 channel);//channel = 0~7


void main(void){

        u8     i;
        u16    j;
```

```
	//设置为准双向口
	P0M1 = 0; P0M0 = 0;
	P1M1 = 0; P1M0 = 0;
	P2M1 = 0; P2M0 = 0;
	P3M1 = 0; P3M0 = 0;
	P4M1 = 0; P4M0 = 0;
	P5M1 = 0; P5M0 = 0;
	P6M1 = 0; P6M0 = 0;
	P7M1 = 0; P7M0 = 0;

	display_index = 0;
	P1ASF = 0;	//对内部基准做 ADC
	ADC_CONTR = 0xE0;			//90T, ADC power on

	AUXR = 0x80;	//定时器 0 设为：不分频, 16 位自动重载
	TH0 = (u8)(Timer0_Reload / 256);
	TL0 = (u8)(Timer0_Reload % 256);
	ET0 = 1;	//Timer0 interrupt enable
	TR0 = 1;	//Tiner0 run
	EA = 1;		//打开总中断

	for(i=0; i<8; i++)LED8[i] = 0x10; //上电消隐

	while(1){
		if(B_1ms){//1ms 到

			B_1ms = 0;
			if(++msecond >= 200){	//200ms 到

				msecond = 0;
				for(j=0,i=0; i<16; i++)	j += Get_ADC10bitResult(0);	// 读 内 部 基 准 ADC,
P1ASF=0, 读 0 通道
				j = (u32)128000UL*16 / j;	// ADC = 1024 * Uref / Ux, 则 Ux = 1024 * Uref / ADC =
1024 * 1.25 / ADC = 1280 / ADC

				LED8[5] = j / 100 + DIS_DOT;	//显示 MCU 电压值, 计算时放大了 100 倍, 电压有
两位小数.
				LED8[6] = (j % 100) / 10;
				LED8[7] = j % 10;
			}
		}
	}
}

// 查询法读一次 ADC 结果.
u16	Get_ADC10bitResult(u8 channel){ //channel = 0~7
	ADC_RES = 0;
	ADC_RESL = 0;

	ADC_CONTR = (ADC_CONTR & 0xe0) | 0x08 | channel;	//启动 ADC
	NOP(4);

	while((ADC_CONTR & 0x10) == 0)	;	//等待 ADC 转换结束
	ADC_CONTR &= ~0x10;		//清除 ADC 结束标志
	return	(((u16)ADC_RES << 2) | (ADC_RESL & 3));
}
```

```
void Send_595(u8 dat){
	u8	i;
	for(i=0; i<8; i++){
		dat <<= 1;
		P_HC595_SER	= CY;
		P_HC595_SRCLK = 1;
		P_HC595_SRCLK = 0;
	}
}

void DisplayScan(void){

	Send_595(~LED_TYPE ^ T_COM[display_index]);				//输出位码
	Send_595( LED_TYPE ^ t_display[LED8[display_index]]);	//输出段码

	P_HC595_RCLK = 1;
	P_HC595_RCLK = 0;					//锁存输出数据
	if(++display_index >= 8)	display_index = 0;	//8 位结束回 0
}

void timer0 (void) interrupt TIMER0_VECTOR{

	DisplayScan();	//1ms 扫描显示一位
	B_1ms = 1;		//1ms 标志
}
```

4.3.4 实验任务

- **实验目的：**

  1）熟悉 Keil C 整体编程环境以及 MCU 程序开发流程；

  2）了解 A/D 模块的工作原理以及编程结构；

  3）学习 A/D 模块的编程技术。

- **实验内容：**

1） 基本实验 1：使用单片机内部的 A/D 转换功能实现 ADC 键盘，键盘一共有 4 个有效按钮，分别对应 4 盏 LED。当按下按钮 1 时，切换第 1 盏 LED 灯的熄灭或亮起状态；其他按钮的控制规律与此类似。（提示：可用 P1ASF = 0x10; 设置引脚 P1.4 为 ADC 模拟输入端口）

2） 基本实验 2：设计程序实现 ADC 键盘，键盘一共有 10 个有效按钮，分别对应 0-9 共 10 个数字，用户按下按键后，数码管显示相应数字。

3） 基本实验 3：设计程序实现 ADC 键盘，令数码管高四位与低四位作为两个独立计数器的显示器。初始状态下，两个计数器都是 0，数码管显示“0000 0000”，

按键 0、1 分别为高四位的加、减按键，按键 2、3 分别为低四位的加、减按键。当计数器为 0 时，减按键无效。

4.4 单片机综合实验（3 学时）

- **实验目的：**

1）熟悉 Keil C51 编程开发环境和单片机程序的开发流程；

2）考察综合运用单片机知识，解决较复杂问题的能力。

3）自行完成设计性实验，培养独立创新意识。

- **实验内容：**

1） 反应测试游戏

请使用 2 个按键、3 位数码管、1 盏 LED 指示灯，实现反映测试游戏功能。

游戏规则如下：设置 1 个准备按钮、1 个停止按钮。游戏开始时按住准备按钮，等待一段时间后（由程序随机确定），点亮 LED 灯并立即开始计时，3 位数码管动态显示持续时间。用户应迅速释放被按下的准备按钮，按下停止按钮。当停止键被按下时，计数立即停止，数码管中显示的数字保持不变作为反应时间。当准备按钮被再次按下时，计时显示清零，游戏重新开始。

2） 乒乓球游戏

请使用 4 个按键、2 位数码管、4 盏 LED 指示灯，实现兵乓球游戏功能。

4 个按键分别定义为“左击球”、“右击球”、“开始比赛”、“游戏重置”；亮起的 LED 表示乒乓球的位置，数码管显示当前双方的比分。

游戏规则如下：按下 1 个按键后，游戏开始，此时如果按动左击球键，则最左边的 LED 亮起，并且从左边开始不断向右边移动，当最右侧的 LED 亮起时，如果此时右击球键按下，则 LED 从右边又不断向左边移动，如果提前按下了“右击球”按钮，或者 LED 灯到达左边之后还没按下“右击球”按钮，则左边数码管加 1。同理如果左边出现了失误，右侧数码管加 1。

每局游戏结束后，再次按下“开始比赛”键，重新开始新一轮比赛，但比分保持。在球打向右边的过程中，“左击球”键没有作用，反之亦然。在游戏过程中如果按下了“游戏重置”键则游戏停止并且比分清零。

有能力的同学可以考虑在每一次球被击打后，小幅提高球的移动速度，以增加游戏的对抗性和趣味性。

3） 自行设计实验

要求使用中断、AD 转换和 I/O 引脚基本操作，具有一定的趣味性并保证一定的实现难度。

5. 总结

单片机原理实验课程是电子科学、测控技术及自动化等专业学生的重要实践课程，也是培养动手能力、创新能力、适应能力、团队合作等综合能力的教学环节。在完成实验任务的过程中，需要综合运用电路基础、模拟电子技术、数字电子技术、计算机软件应用技术的知识，才能掌握单片机的各功能部件及控制方式。学习单片机原理和技术，除了认识最基本的电路结构之外，还应当主动了解单片机的用途与应用前景，并熟练掌握单片机系统的设计与开发方法。

6. 附录

6.1 中断寄存器

符号	描述	地址	位地址及符号 MSB							LSB	复位值
IE	Interrupt Enable	A8H	EA	ELVD	EADC	ES	ET1	EX1	ET0	EX0	0000 0000B
IE2	Interrupt Enable 2	AFH	-	ET4	ET3	ES4	ES3	ET2	ESPI	ES2	x000 0000B
INT_CLKO AUXR2	外部中断允许和时钟输出寄存器	8FH	-	EX4	EX3	EX2	MCKO_S2	T2CLKO	T1CLKO	T0CLKO	x000 0000B
IP	Interrupt Priority Low	B8H	PPCA	PLVD	PADC	PS	PT1	PX1	PT0	PX0	0000 0000B
IP2	2rd Interrupt Priority Low register	B5H	-	-	-	PX4	PPWMFD	PPWM	PSPI	PS2	xx00 0000B
TCON	Timer Control register	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	0000 0000B
SCON	Serial Control	98H	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI	0000 0000B
S2CON	Serial 2/ UART2 Control	9AH	S2SM0	-	S2SM2	S2REN	S2TB8	S2RB8	S2TI	S2RI	0100 0000B
S3CON	串口3控制寄存器	ACH	S3SM0	S3ST3	S3SM2	S3REN	S3TB8	S3RB8	S3TI	S3RI	0000 0000B
S4CON	串口4控制寄存器	84H	S4SM0	S4ST4	S4SM2	S4REN	S4TB8	S4RB8	S4TI	S4RI	0000 0000B
T4T3M	T4和T3的控制寄存器	D1H	T4R	T4_C/$\overline{T}$	T4x12	T4CLKO	T3R	T3_C/$\overline{T}$	T3x12	T3CLKO	0000 0000B
PCON	Power Control register	87H	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL	0011 0000B
ADC_CONTR	ADC control register	BCH	ADC_POWER	SPEED1	SPEED0	ADC_FLAG	ADC_START	CHS2	CHS1	CHIS0	0000 0000B
SPSTAT	SPI Status register	CDH	SPIF	WCOL	-	-	-	-	-	-	00xx xxxxB
CCON	PCA Control Register	D8H	CF	CR	-	-	-	CCF2	CCF1	CCF0	00xx x000B
CMOD	PCA Mode Register	D9H	CIDL	-	-	-	CPS2	CPS1	CPS0	ECF	0xxx 0000B
CCAPM0	PCA Module 0 Mode Register	DAH	-	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0	x000 0000B
CCAPM1	PCA Module 1 Mode Register	DBH	-	ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1	x000 0000B
CCAPM2	PCA Module 2 Mode Register	DCH	-	ECOM2	CAPP2	CAPN2	MAT2	TOG2	PWM2	ECCF2	x000 0000B
AUXR	辅助寄存器	8EH	T0x12	T1x12	UART_M0x6	T2R	T2_C/$\overline{T}$	T2x12	EXTRAM	S1ST2	0000 0001B
CMPCR1	比较器控制寄存器1	E6H	CMPEN	CMPIF	PIE	NIE	PIS	NIS	CMPOE	CMPRES	0000 0000B

6.2 中断使能寄存器

IE：中断允许寄存器（可位寻址）

SFR name	Address	bit	B7	B6	B5	B4	B3	B2	B1	B0
IE	A8H	name	EA	ELVD	EADC	ES	ET1	EX1	ET0	EX0

EA：CPU的总中断允许控制位，EA=1，CPU开放中断，EA=0，CPU屏蔽所有的中断申请。EA的作用是使中断允许形成多级控制。即各中断源首先受EA控制;其次还受各中断源自己的中断允许控制位控制。

ELVD：低压检测中断允许位，ELVD=1，允许低压检测中断，ELVD=0，禁止低压检测中断。

EADC：A/D转换中断允许位，EADC=1，允许A/D转换中断，EADC=0，禁止A/D转换中断。

ES： 串行口1中断允许位，ES=1，允许串行口1中断，ES=0，禁止串行口1中断。

ET1：定时/计数器T1的溢出中断允许位，ET1=1，允许T1中断，ET1=0，禁止T1中断。

EX1：外部中断1中断允许位，EX1=1，允许外部中断1中断，EX1=0，禁止外部中断1中断。

ET0：T0的溢出中断允许位，ET0=1允许T0中断，ET0=0禁止T0中断。

EX0：外部中断0中断允许位，EX0=1允许中断，EX0=0禁止中断。

IE2：中断允许寄存器（不可位寻址）

SFR name	Address	bit	B7	B6	B5	B4	B3	B2	B1	B0
IE2	AFH	name	-	ET4	ET3	ES4	ES3	ET2	ESPI	ES2

ET4：定时器4的中断允许位。

ET4=1,允许定时器4产生中断；

ET4=0,禁止定时器4产生中断

ET3：定时器3的中断允许位。

ET3=1,允许定时器3产生中断；

ET3=0,禁止定时器3产生中断

ES4：串行口4中断允许位。

ES4=1，允许串行口4中断；

ES4=0，禁止串行口4中断

ES3：串行口3中断允许位。

ES3=1，允许串行口3中断；

ES3=0，禁止串行口3中断。。

ET2：定时器2的中断允许位。

ET2=1,允许定时器2产生中断；

ET2=0,禁止定时器2产生中断

ESPI：SPI中断允许位。

ESPI=1，允许SPI中断；

ESPI=0，禁止SPI中断。

ES2：串行口2中断允许位。

ES2=1，允许串行口2中断；

ES2=0，禁止串行口2中断。

6.4 中断优先寄存器

IP：中断优先级控制寄存器（可位寻址）

SFR name	Address	bit	B7	B6	B5	B4	B3	B2	B1	B0
IP	B8H	name	PPCA	PLVD	PADC	PS	PT1	PX1	PT0	PX0

PPCA: PCA中断优先级控制位。
当PPCA=0时，PCA中断为最低优先级中断(优先级0)
当PPCA=1时，PCA中断为最高优先级中断(优先级1)

PLVD: 低压检测中断优先级控制位。
当PLVD=0时，低压检测中断为最低优先级中断(优先级0)
当PLVD=1时，低压检测中断为最高优先级中断(优先级1)

PADC: A/D转换中断优先级控制位。
当PADC=0时，A/D转换中断为最低优先级中断(优先级0)
当PADC=1时，A/D转换中断为最高优先级中断(优先级1)

PS: 串口1中断优先级控制位。
当PS=0时，串口1中断为最低优先级中断(优先级0)
当PS=1时，串口1中断为最高优先级中断(优先级1)

PT1: 定时器1中断优先级控制位。
当PT1=0时，定时器1中断为最低优先级中断(优先级0)
当PT1=1时，定时器1中断为最高优先级中断(优先级1)

PX1: 外部中断1优先级控制位。
当PX1=0时，外部中断1为最低优先级中断(优先级0)
当PX1=1时，外部中断1为最高优先级中断(优先级1)

PT0: 定时器0中断优先级控制位。
当PT0=0时，定时器0中断为最低优先级中断(优先级0)
当PT0=1时，定时器0中断为最高优先级中断(优先级1)

PX0: 外部中断0优先级控制位。
当PX0=0时，外部中断0为最低优先级中断(优先级0)

STC MCU Limited

6.5 定时器/计数器中断控制寄存器

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：T1溢出中断标志。T1被允许计数以后，从初值开始加1计数。当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”（也可由查询软件清“0”）。

TR1：定时器1的运行控制位。

TF0：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（也可由查询软件清0）。

TR0：定时器0的运行控制位。

IE1：外部中断1（INT1/P3.3）中断请求标志。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中断源类型选择位。IT1=0，INT1/P3.3引脚上的上升沿或下降沿信号均可触发外部中断1。IT1=1，外部中断1为下降沿触发方式。

IE0：外部中断0（INT0/P3.2）中断请求标志。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。

IT0：外部中断0中断源类型选择位。IT0=0，INT0/P3.2引脚上的上升沿或下降沿均可触发外部中断0。IT0=1，外部中断0为下降沿触发方式。

6.6 与 ADC 相关的寄存器

符号	描述	地址	位地址及其符号 MSB							LSB	复位值
P1ASF	P1 Analog Function Configure register	9DH	P17ASF	P16ASF	P15ASF	P14ASF	P13ASF	P12ASF	P11ASF	P10ASF	0000 0000B
ADC_CONTR	ADC Control Register	BCH	ADC_POWER	SPEED1	SPEED0	ADC_FLAG	ADC_START	CHS2	CHS1	CHS0	0000 0000B
ADC_RES	ADC Result high	BDH									0000 0000B
ADC_RESL	ADC Result low	BEH									0000 0000B
CLK_DIV PCON2	时钟分频寄存器	97H	MCKO_S1	MCKO_S0	ADRJ	Tx_Rx	MCLKO_2	CLKS2	CLKS1	CLKS0	0000 0000B
IE	Interrupt Enable	A8H	EA	ELVD	EADC	ES	ET1	EX1	ET0	EX0	0000 0000B
IP	Interrupt Priority Low	B8H	PPCA	PLVD	PADC	PS	PT1	PX1	PT0	PX0	0000 0000B

ADC_CONTR：ADC控制寄存器

SFR name	Address	bit	B7	B6	B5	B4	B3	B2	B1	B0
ADC_CONTR	BCH	name	ADC_POWER	SPEED1	SPEED0	ADC_FLAG	ADC_START	CHS2	CHS1	CHS0

对ADC_CONTR寄存器进行操作，建议直接用MOV赋值语句，不要用‘与’和‘或’语句。

ADC_POWER: ADC 电源控制位。

0：关闭ADC 电源；

1：打开A/D转换器电源.

建议进入空闲模式和掉电模式前，将ADC电源关闭，即ADC_POWER =0，可降低功耗。启动A/D转换前一定要确认A/D电源已打开，A/D转换结束后关闭A/D电源可降低功耗，也可不关闭。初次打开内部A/D转换模拟电源，需适当延时，等内部模拟电源稳定后，再启动A/D转换。

建议启动A/D转换后，在A/D转换结束之前，不改变任何I/O口的状态，有利于高精度A/D转换,如能将定时器/串行口/中断系统关闭更好。

SPEED1，SPEED0：模数转换器转换速度控制位

SPEED1	SPEED0	A/D转换所需时间
1	1	90个时钟周期转换一次，CPU工作频率27MHz时，A/D转换速度约300KHz（=27MHz÷90）
1	0	180个时钟周期转换一次
0	1	360个时钟周期转换一次
0	0	540个时钟周期转换一次

ADC_FLAG: 模数转换器转换结束标志位,当A/D转换完成后，ADC_FLAG = 1，要由软件清0。不管是A/D 转换完成后由该位申请产生中断，还是由软件查询该标志位A/D转换是否结束,当A/D转换完成后，ADC_FLAG = 1，一定要软件清0。

ADC_START:模数转换器(ADC)转换启动控制位，设置为“1”时，开始转换,转换结束后为0。

CHS2/CHS1/CHS0：模拟输入通道选择，CHS2/CHS1/CHS0

CHS2	CHS1	CHS0	Analog Channel Select（模拟输入通道选择）
0	0	0	选择 P1.0 作为A/D输入来用
0	0	1	选择 P1.1 作为A/D输入来用
0	1	0	选择 P1.2 作为A/D输入来用
0	1	1	选择 P1.3 作为A/D输入来用
1	0	0	选择 P1.4 作为A/D输入来用
1	0	1	选择 P1.5 作为A/D输入来用
1	1	0	选择 P1.6 作为A/D输入来用
1	1	1	选择 P1.7 作为A/D输入来用

ADC转换结果调整寄存器位——ADRJ位于寄存器CLK_DIV/PCON中，用于控制ADC转换结果存放的位置。

Mnemonic	Add	Name	B7	B6	B5	B4	B3	B2	B1	B0	Reset Value
CLK_DIV (PCON2)	97H	时钟分频寄存器	MCKO_S1	MCKO_S0	ADRJ	Tx_Rx	MCLKO_2	CLKS2	CLKS1	CLKS0	0000,x000

ADRJ：ADC转换结果调整

0：ADC_RES[7:0]存放高8位ADC结果，ADC_RESL[1:0]存放低2位ADC结果

1：ADC_RES[1:0]存放高2位ADC结果，ADC_RESL[7:0]存放低8位ADC结果

4. A/D转换结果寄存器ADC_RES、ADC_RESL

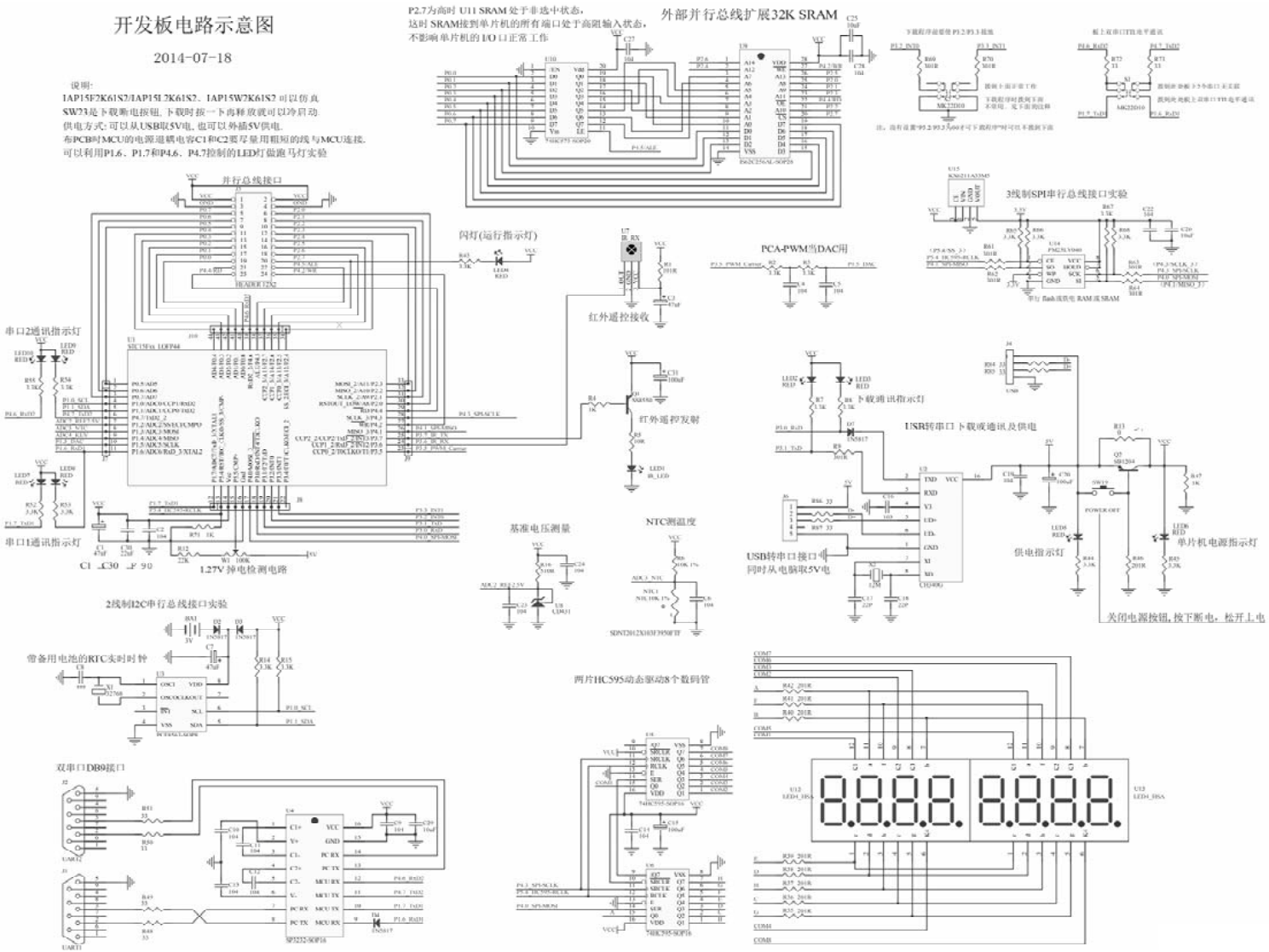
特殊功能寄存器ADC_RES和ADC_RESL寄存器用于保存A/D转换结果，其格式如下：

Mnemonic	Add	Name	B7	B6	B5	B4	B3	B2	B1	B0
ADC_RES	BDh	A/D转换结果寄存器高								
ADC_RESL	BEh	A/D转换结果寄存器低								
CLK_DIV (PCON2)	97H	时钟分频寄存器	MCKO_S1	MCKO_S0	ADRJ	Tx_Rx	MCLK0_2	CLKS2	CLKS1	CLKS0

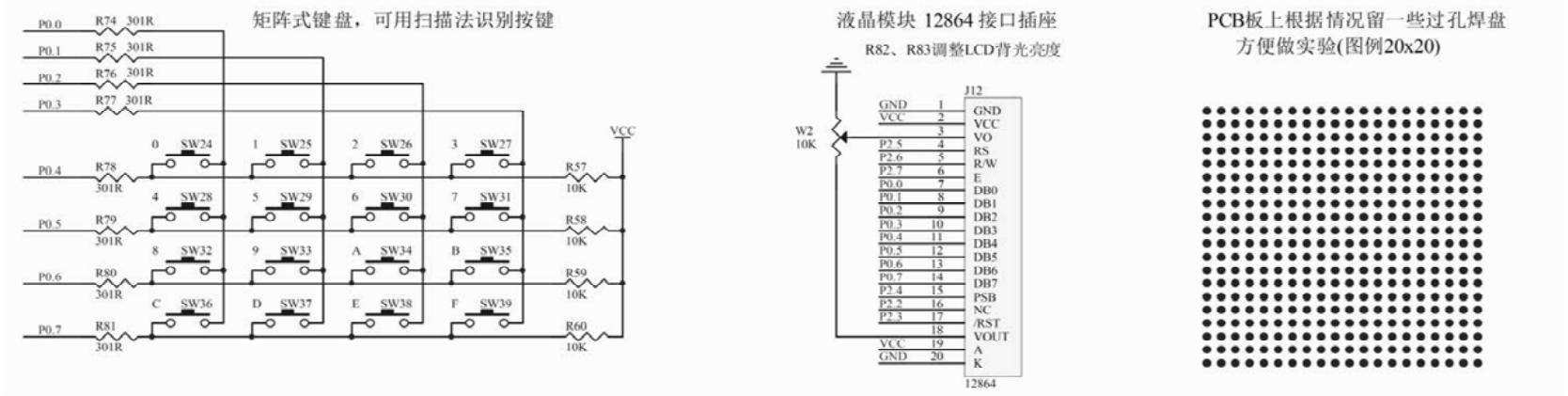
CKKO_DIV寄存器的ADRJ位是A/D转换结果寄存器(ADC_RES,ADC_RESL)的数据格式调整控制位.
当ADRJ=0时，10位A/D转换结果的高8位存放在ADC_RES中，低2位存放在ADC_RESL的低2位中。

Mnemonic	Add	Name	B7	B6	B5	B4	B3	B2	B1	B0
ADC_RES	BDh	A/D转换结果寄存器高8位	ADC_RES9	ADC_RES8	ADC_RES7	ADC_RES6	ADC_RES5	ADC_RES4	ADC_RES3	ADC_RES2
ADC_RESL	BEh	A/D转换结果寄存器低2位	-	-	-	-	-	-	ADC_RES1	ADC_RES0
CLK_DIV (PCON2)	97H	时钟分频寄存器			ADRJ = 0					

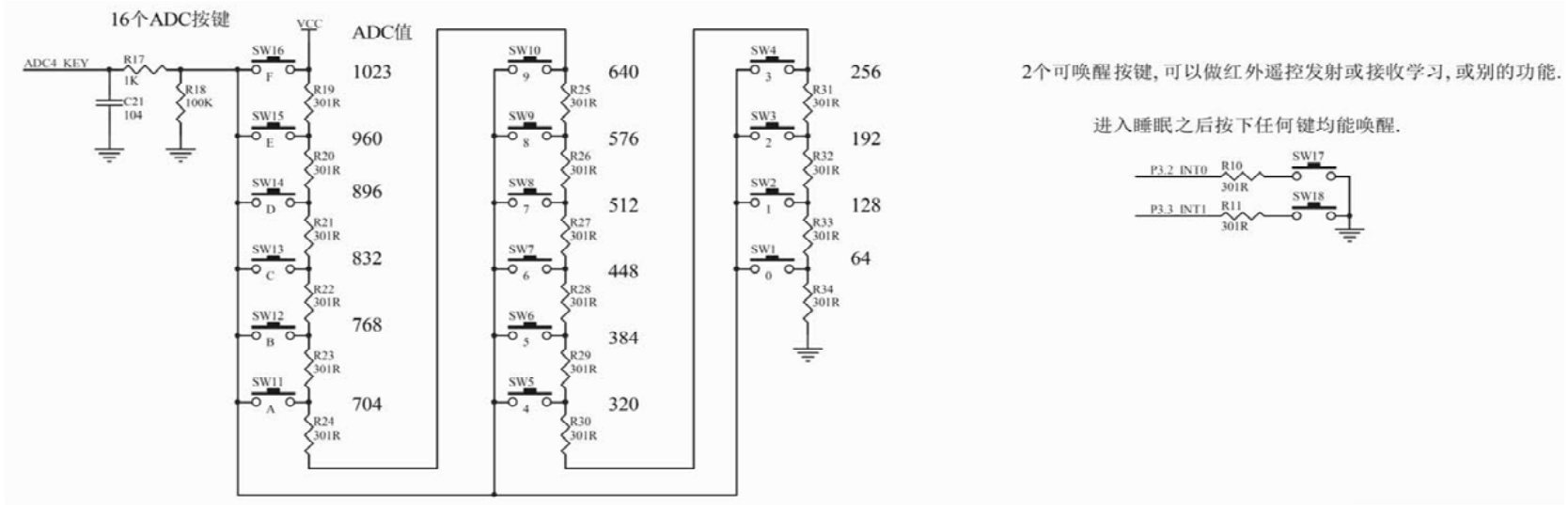
6.7 最小系统原理图


开发板电路示意图
2014-07-18
外部并行总线扩展32K SRAM
并行总线接口
闪灯(运行指示灯)
PCA-PWM当DAC用
3线制SPI串行总线接口实验
串口2通讯指示灯
红外遥控接收
红外遥控发射
串口1通讯指示灯
1.27V掉电检测电路
基准电压测量
NTC测温度
USB转串口接口
同时从电脑取5V电
供电指示灯
单片机电源指示灯
关闭电源按钮，按下断电，松开上电
2线制I2C串行总线接口实验
带备用电池的RTC实时时钟
双串口DB9接口
两片HC595动态驱动8个数码管

6.8 多功能板原理图



读ADC键方法:

每隔10ms左右读一次ADC值,并且保存最后3次的读数,其变化比较小时再判断键.

判断键有效时, 允许一定的偏差, 比如+-16个字的偏差.