51 单片机使用手册

51 单片机使用手册

DIY-51

V2.0 – 2010.1

亿学通™ 北京海淀上地七街国际创业园 2 号院 C 座 812

TEL: 86-10-62669059 *E-mail:61mcu@61mcu.com* *http://www.61mcu.com*

目 录

开发板介绍

1.1 简介

本开发板以 STC89C52 单片机为核心，通过功能齐全优秀的一系列实验内容充分表现和融合了单片机的各功能单元，并通过跳线的方式预留所有接口，配合板上蜂窝板达到用户自由定制的目的。STC89C52 是一种低功耗、高性能 CMOS8 位微控制器，具有 8K 在系统可编程 Flash 存储器。全面兼容 AT89S51/52，并且具备 ISP 在线下载程序功能。

本开发板主要适合以下用户：

1．适合大一、大二电子专业同学电子实习焊接使用；

2．适合高年级同学单片机课程设计、毕业设计使用；

3．也可以作为高校电子、自动化及相关专业的实验设备

1.2 特点

1．由前入深的演示程序，只要用户学习过汇编或 C 语言就能轻松入门；

2．通过跳线使演示学习与开发设计集为一体，方便了功能扩展；

3．USB 和 5V 电源两种供电选择方式，方便了学习开发；

4．MCU 使用的是课堂教学通用的 MCS51 单片机，和教学保持一致性；

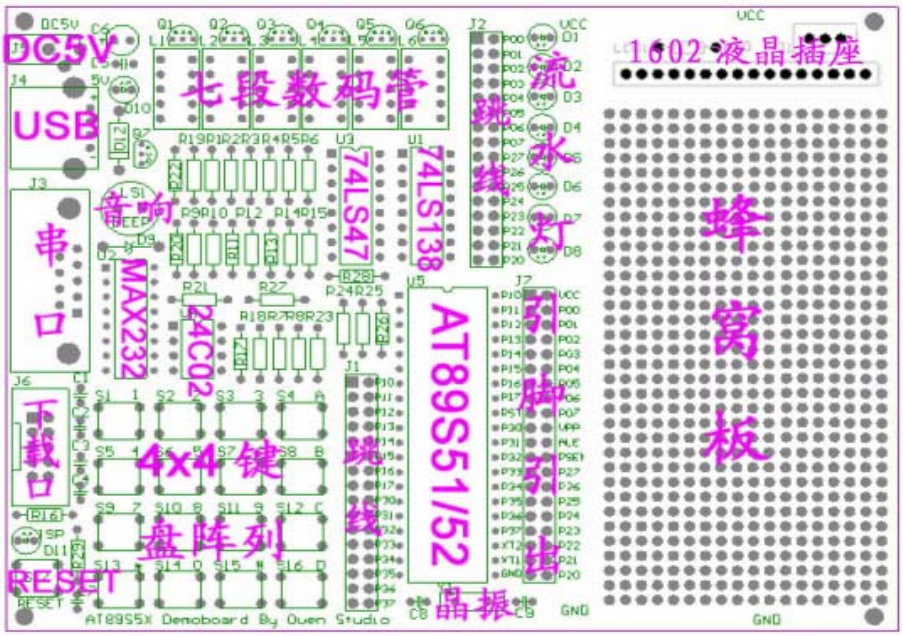
5．光盘附带资料、软件丰富，方便自学和提高；

6．开发套件，价格低廉，资源丰富；

7．尺寸:14cm*12cm。

1.3 布局和结构

1．开发板结构布局



2. 开发板主要器件清单及描述

器件	描述
STC89C52	8051 内核；ISP 在线编程；40 引脚，4 个 GPIO，3 个 Timer，1 个 UART，5 个中断源。
24C02	I^2C 总线接口 E^2PROM，容量 256 字节。
74LS138	三八译码器
74LS47	BCD 码到七段数码管转换器
MAX232	串口收发器，TTL/RS-232 电平转换器。

1.4 实验内容

1. I/O 口的操作

2.定时器及中断实验

3. 4x4 键盘阵列扫描实验

4. 9 首流行歌曲的音乐实验

5.七段数码管实验

6. 5 种流水灯演示实验（包括复杂霓虹灯控制）

7. PWM 实验

8.I2C 接口的 E2PROM 读写实验

9.串口通信实验

10.*1602 液晶显示与控制（选配）

1.5 演示范例描述

在光盘：开发板资料\程序资料\演示程序(Keil)带有一个演示程序，可以测试开发板所有硬件功能。可以使用 STC-ISP 软件将演示程序文件夹中的.hex 文件，下载到单片机中。

演示程序功能如下：

上电后系统等待用户输入密码登录。此时可按 A 键查询储存在 24C02 中的 6 位密码，也可修改登陆密码，按 B 键将屏幕中的 6 位数字设为密码。按 C 键清屏。当输入密码正确后则进入系统，如不正确则清屏等待用户重新输入。

进入系统后 6 个数码管开始计时，初始时间为 10:00:00。同时循环播放 9 首音乐，发光二极管也按预设的流水灯自动延时程序作循环演示。其中有 5 种流水灯方案，包括水滴方案、环扫方案、渐明渐暗方案、慢闪、快闪方案。在此主系统中，按不同键对应不同的功能。

1¯9 数字键：播放第 1¯9 首歌曲。

0 键：停止播放音乐。　　A 键：调整时钟秒值。

B 键：调整时钟分值。　　C 键：调整时钟时值。

D 键：开始/停止音乐自动循环播放。　　*键：开始/停止流水灯自动循环演示。

#键：流水灯方案切换。

程序下载方法介绍

将光盘中：“开发板资料\程序下载软件\STC 单片机程序下载软件”里面的“STC-ISP”进行解压；非安装版,自解压，直接执行 STC-ISP.exe 即可 。

STC_ISP 是宏晶科技公司提供的能直接在编程者电脑上使用的 ISP 在线下载方式，将用户程序目标代码（.hex 文件），下载进 STC 单片机的软件。

它可以通过普通的串口（COM）烧写单片机的程序。软件运行稳定。操作相对方便。

软件的操作界面如下：

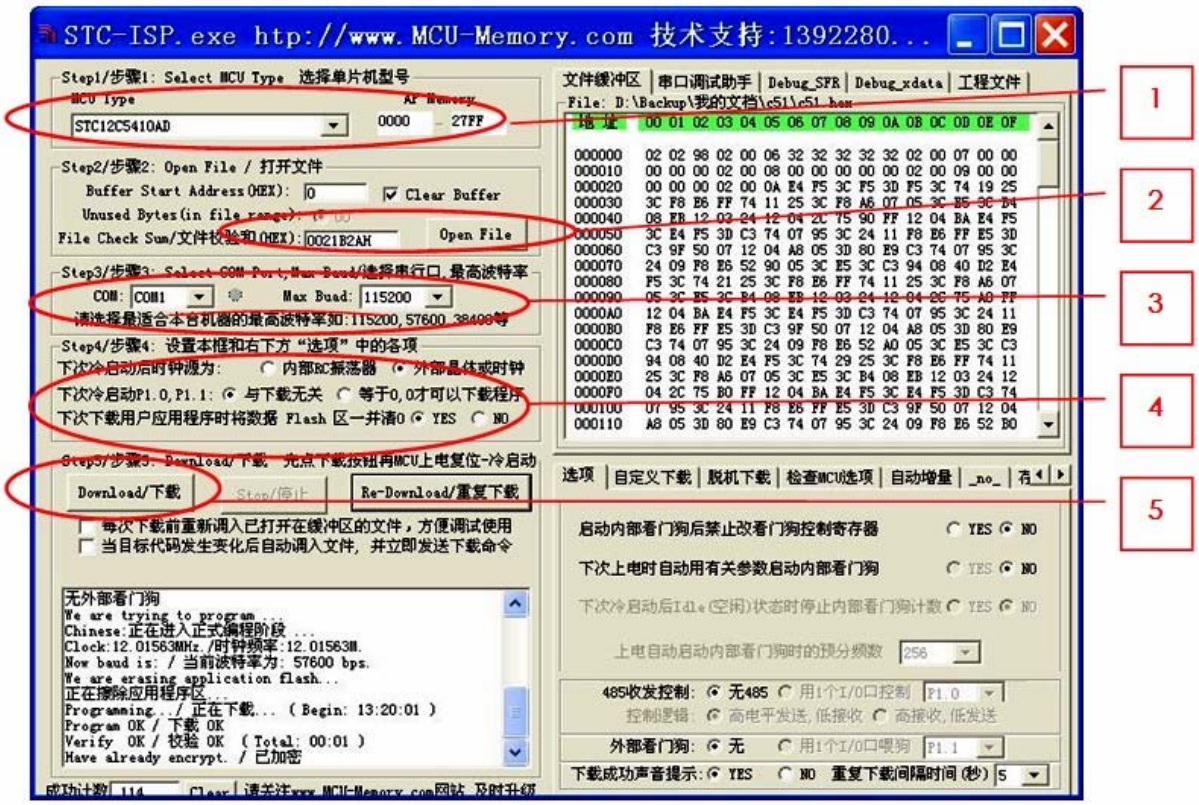


图 STC_ISP 软件界面

下面给出操作的具体步骤：

1、选择芯片类型：STC89C52

2、选择要烧写的 HEX 文件；

3、设置串行端口和波特率，这里要注意端口号，波特率的选择比较任意一般为 38400；

4、选择外部晶振；与下载无关；清 FLASH 区；

5、点击下载按钮后，打开单片机供电电源。在上电后单片机会自动进入编程状态。通过提示可以判断是否下载完成。

注意：

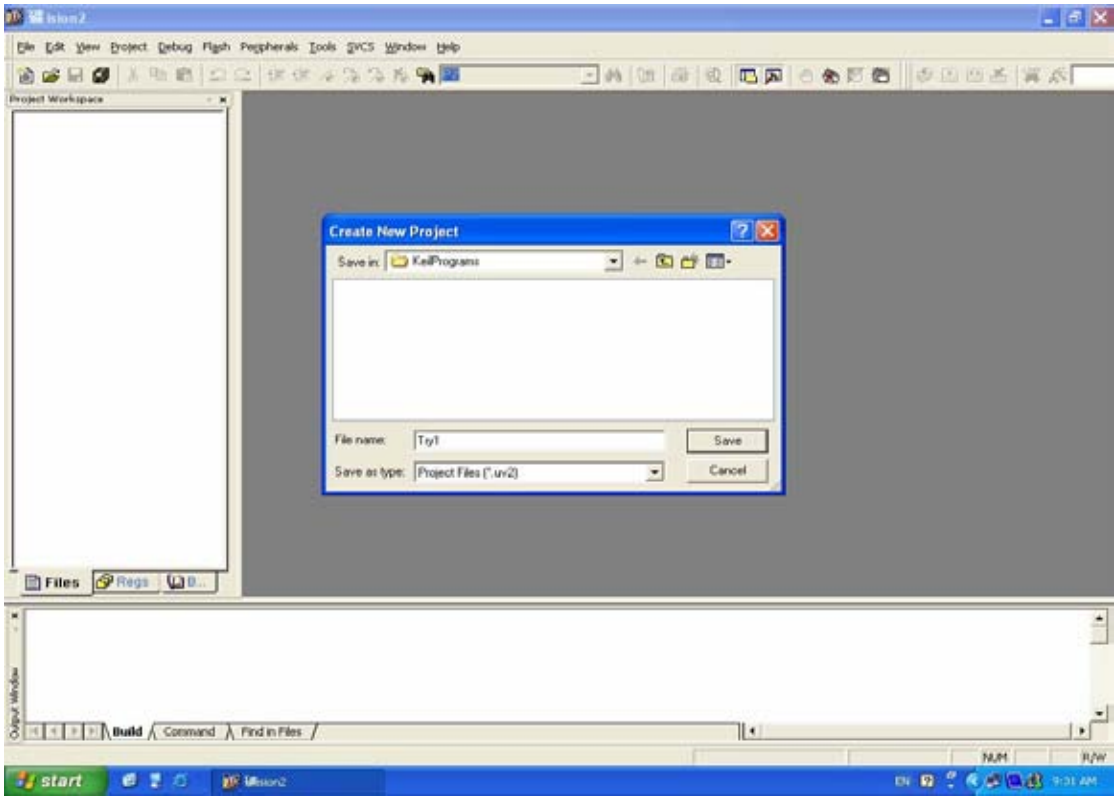
在选择芯片时一定要看清型号，很容易选错；

应选择外部晶振，如果选择内部晶振，单片机可以工作，但是程序中的定时会有变化；

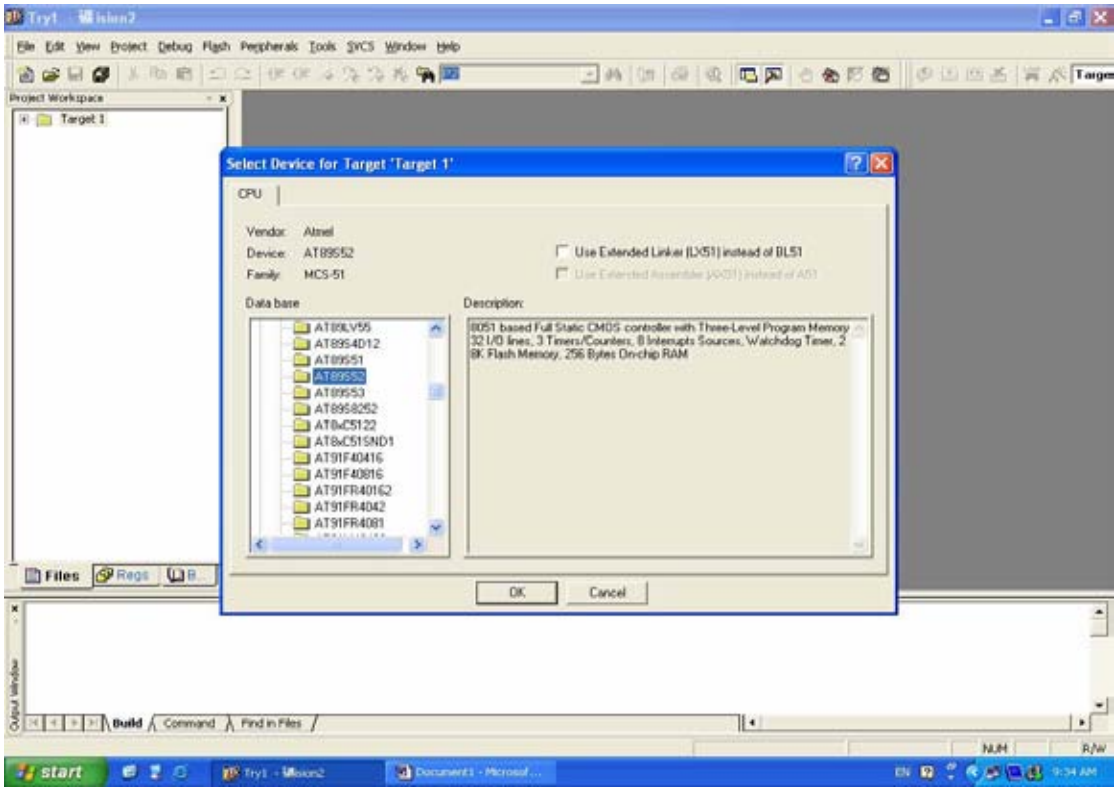
一定要先点编程按钮，然后再打开电源，否则下载不能成功。

Keil 使用简介

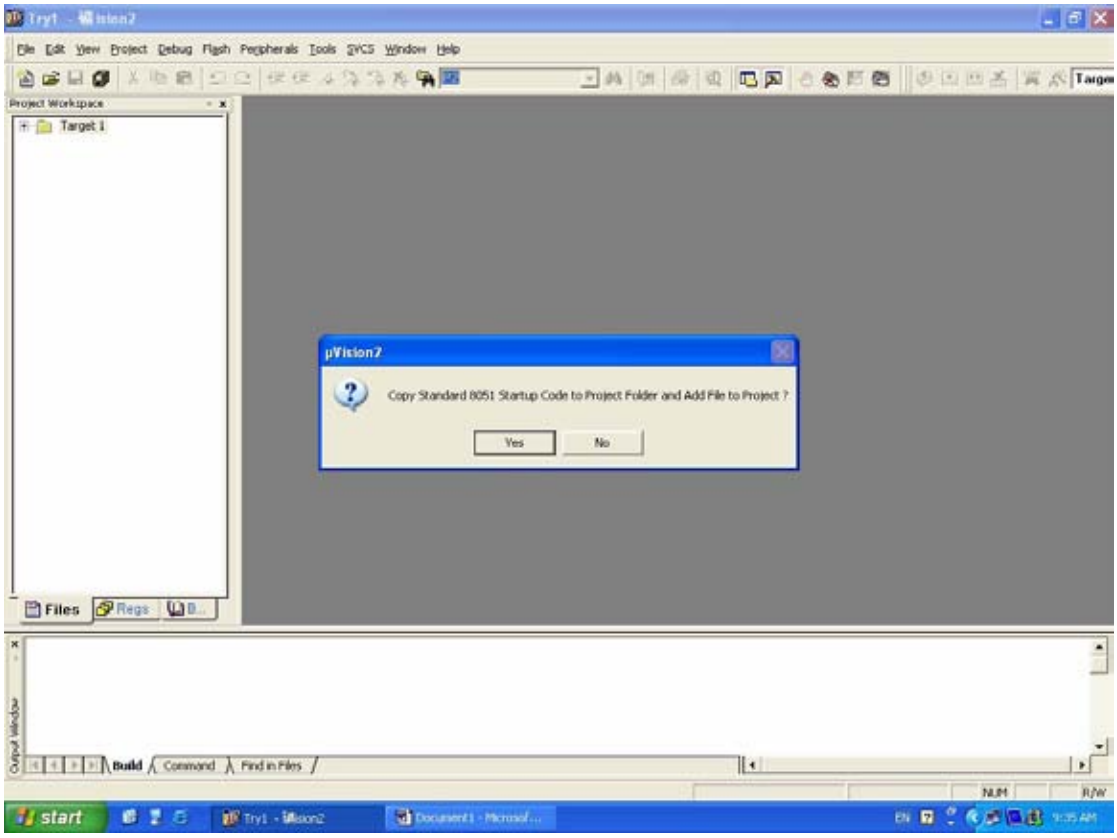
先建一个新的工程，保存到一个位置，如下图所示。



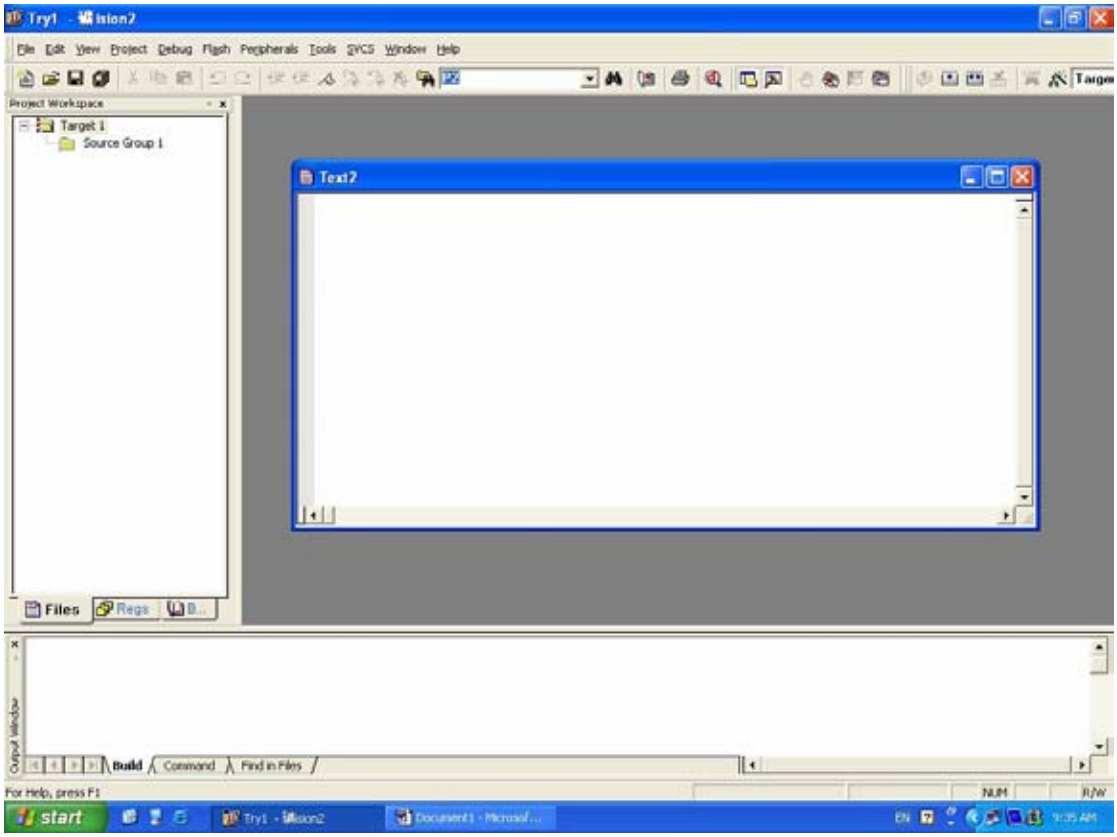
接下来会弹出如下对话框，让选择处理器，这里选择 89S52（因为没有 STC89C52 选项）。



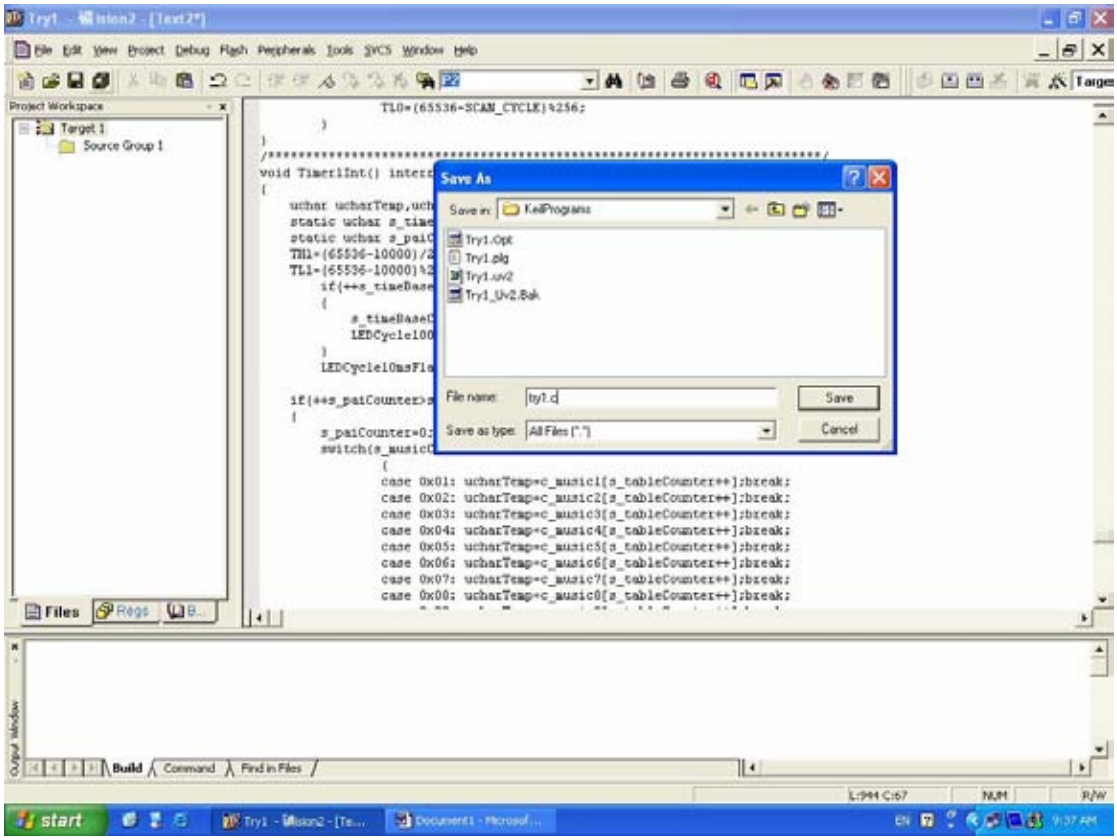
接下来会问是否把 Startup Code 加入到工程，选否即可。工程就建完了。



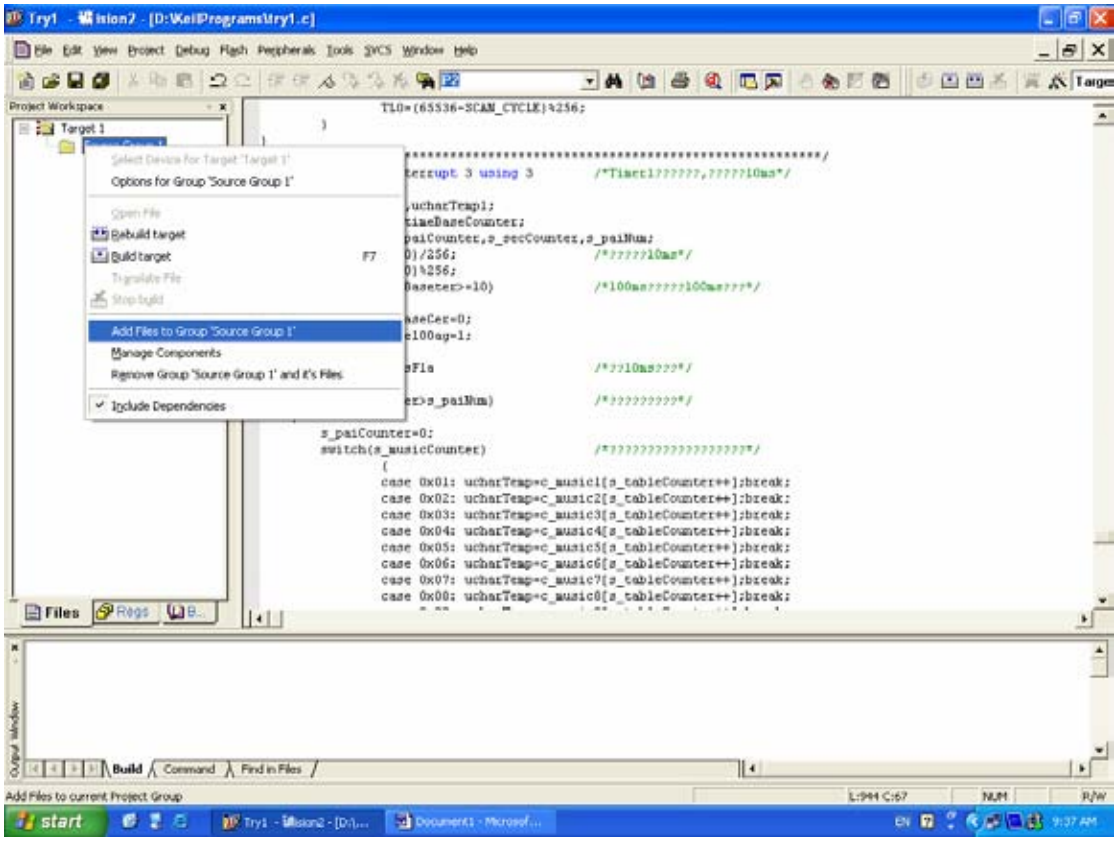
接下来可以新建一个文档用来编辑程序。

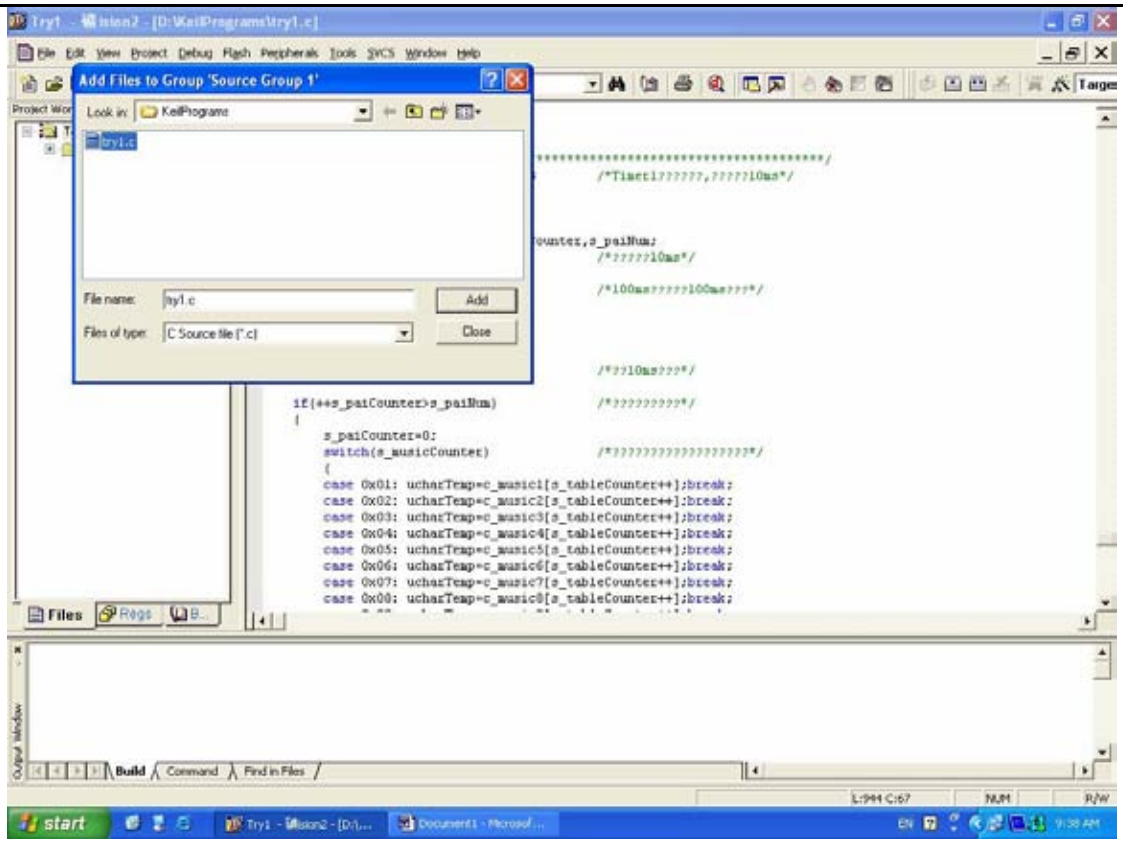


编辑完存为.asm（汇编源文件）或 .h（C 语言头文件）或.c（C 语言实现文件）即可。

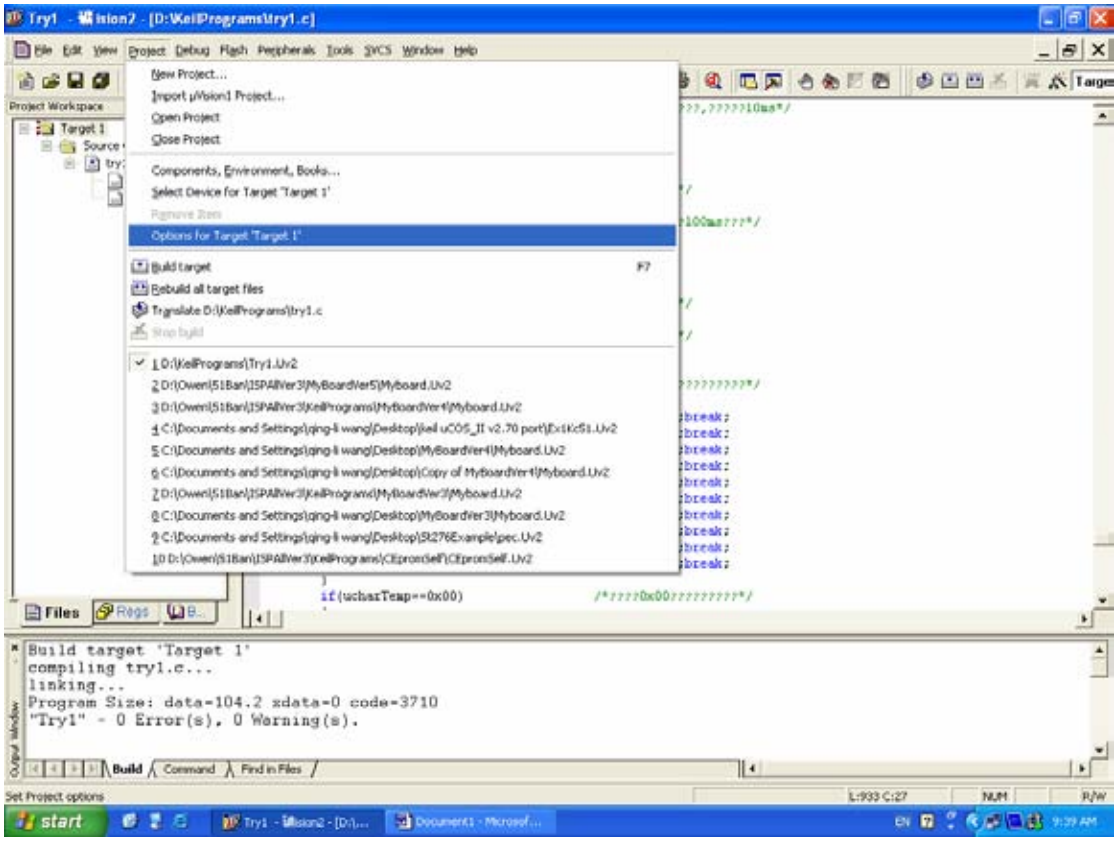


接下来把保存的.asm, .c 或.h 文件加入到工程里即可。如下两图所示。

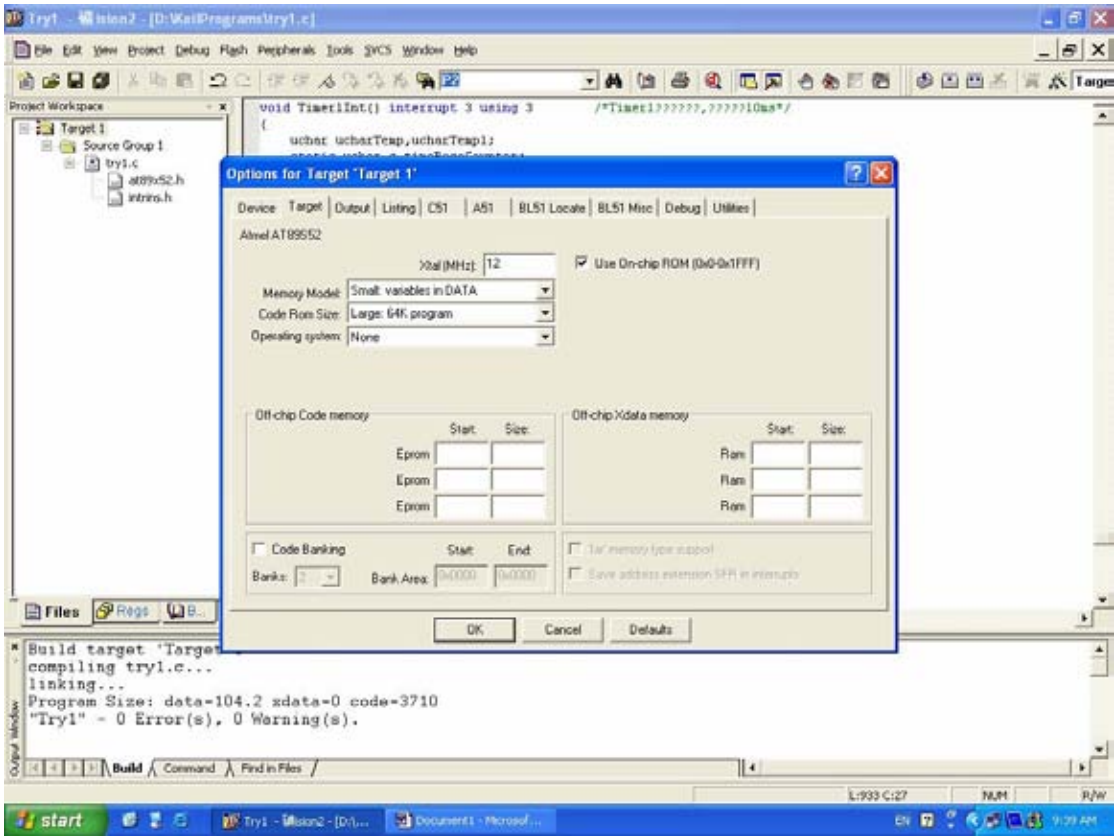




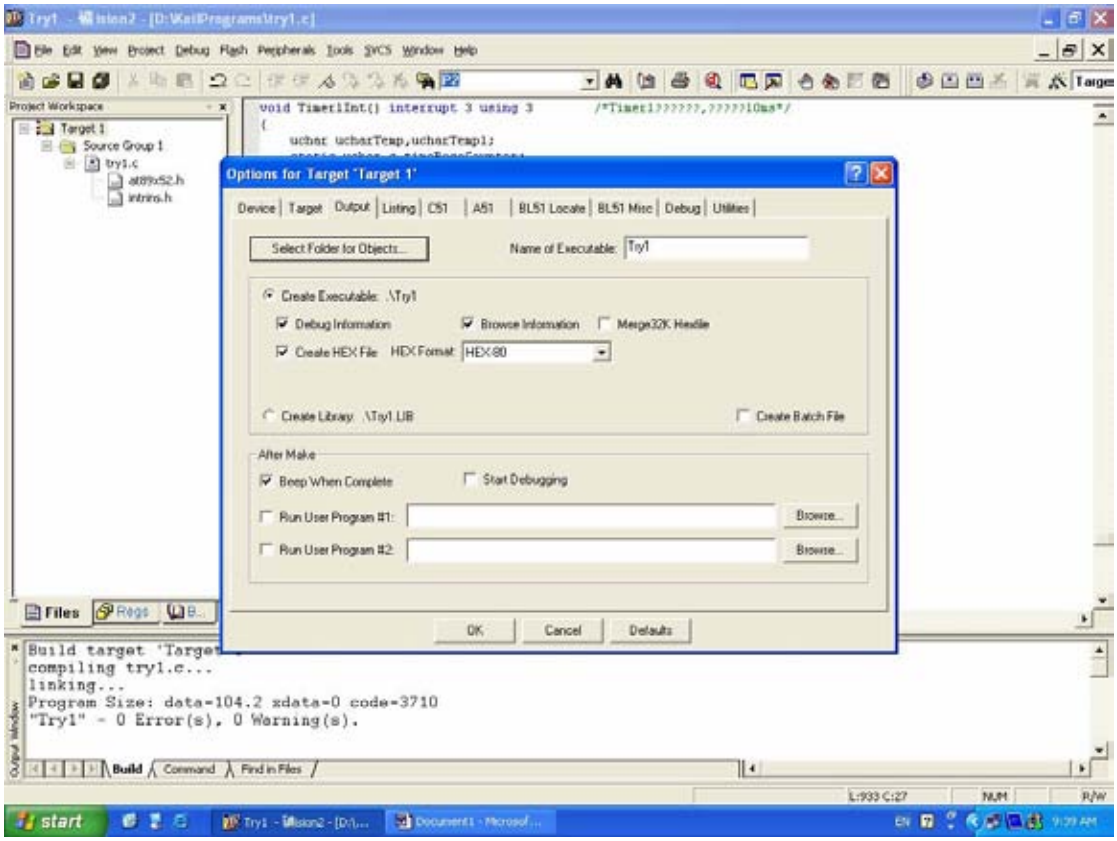
下面进行工程配置。点击 Project 菜单下的 Options for Target ‘Target 1’。



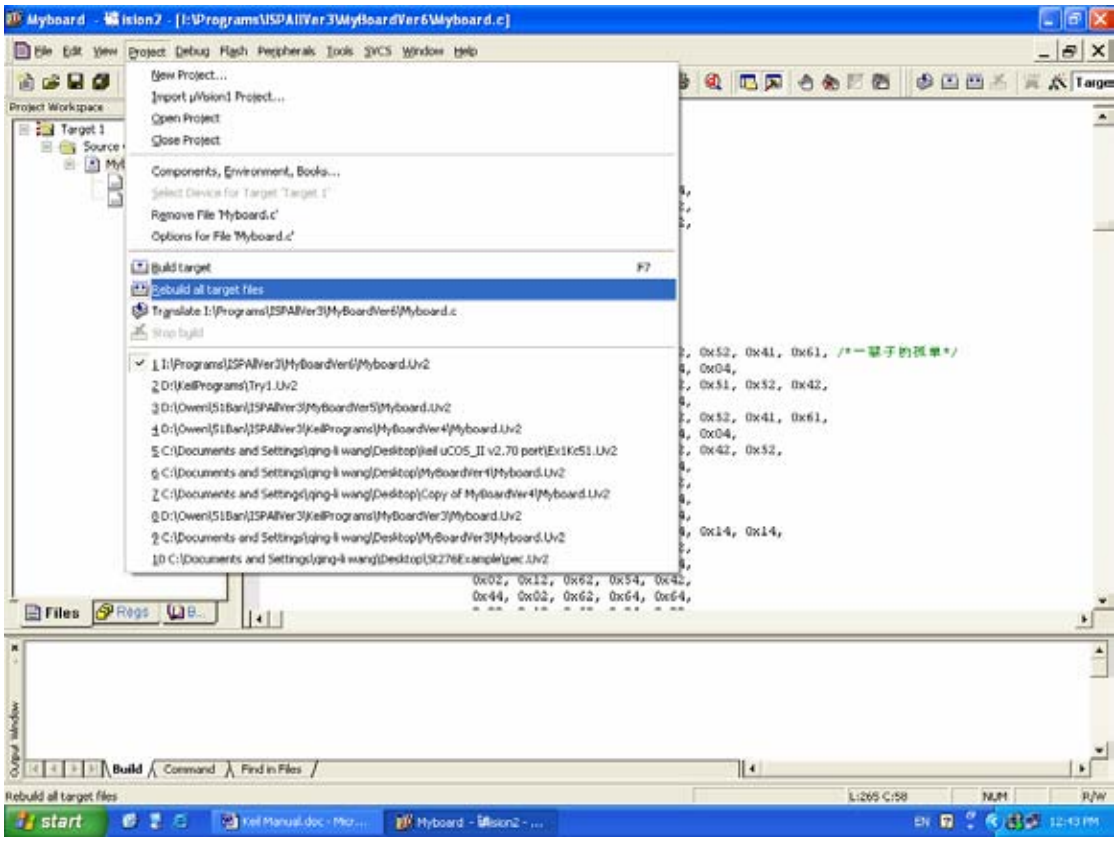
在弹出对话框的 Target 项里输入晶振为 12M，然后勾上 Use On-chip ROM。



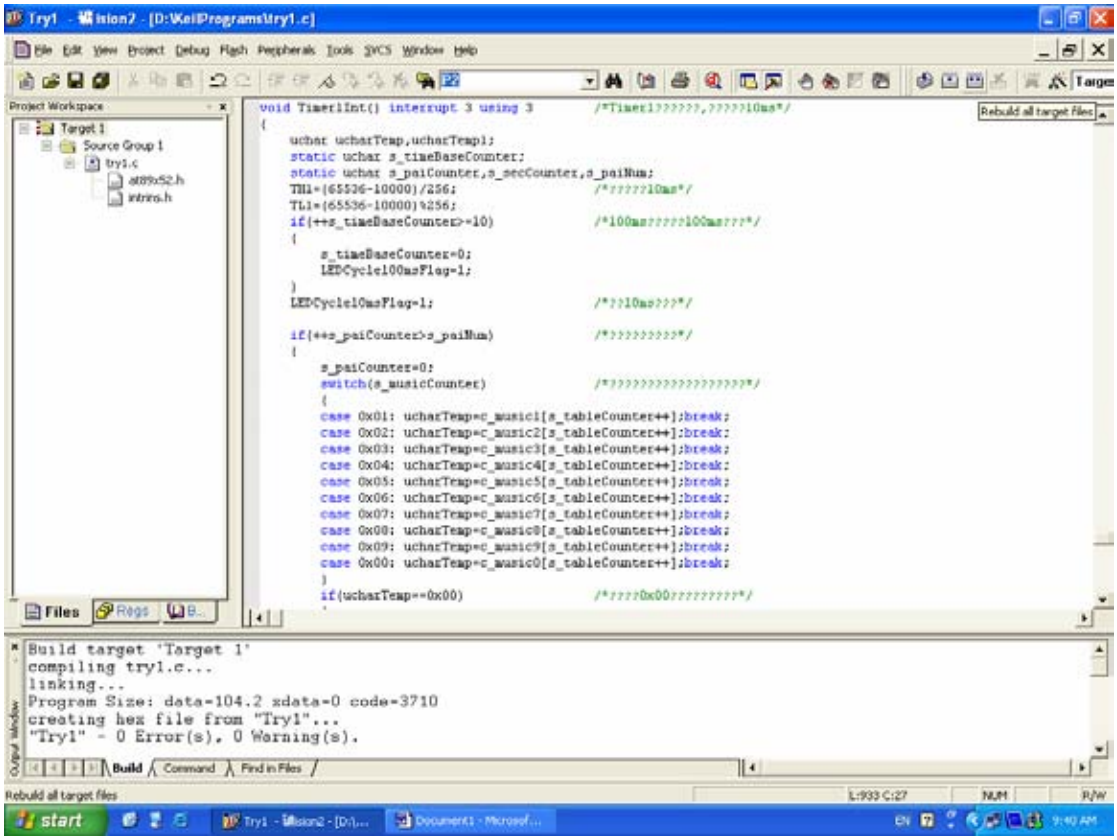
在 Output 项里勾上 Create HEX File。这就是产生要烧写的.hex 文件。



然后点击 Project 菜单里的 build target 或 Rebuild all target files 以编译要烧写的.hex 文件。

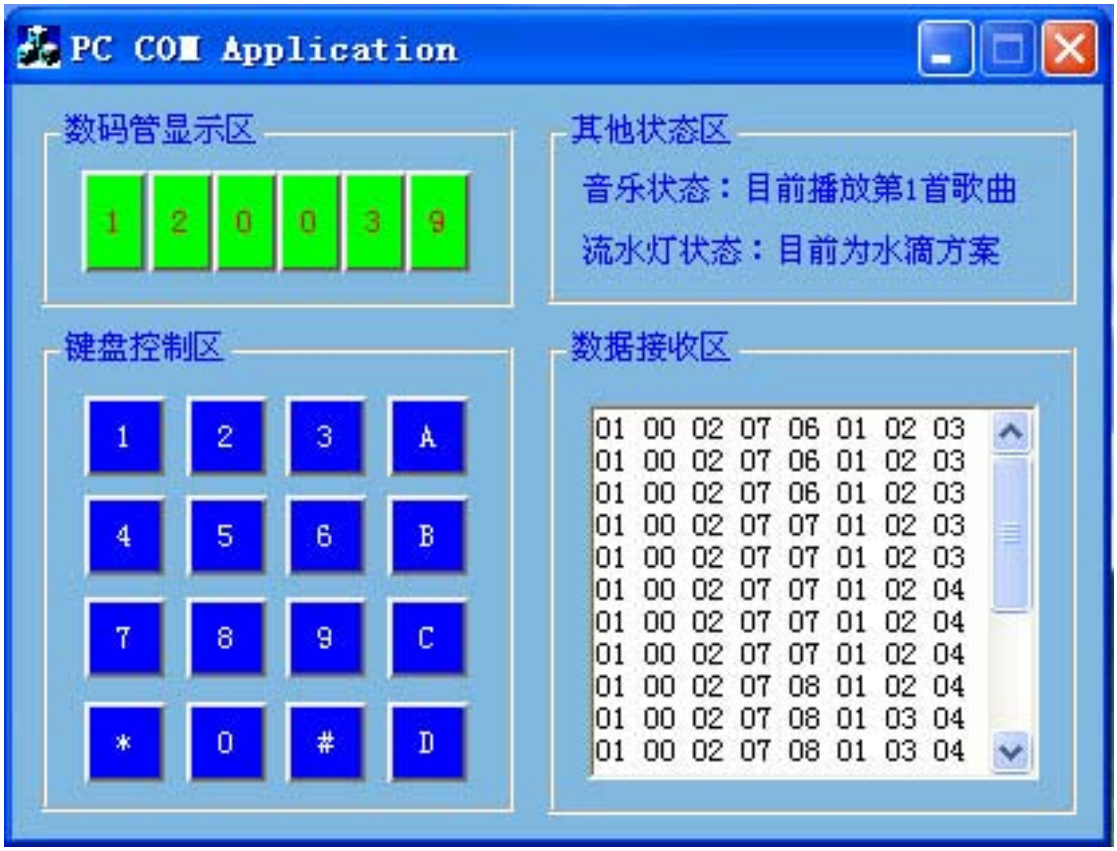


编译完会在下面 Output Window 里显示编译成功与否的信息和错误提示。



PC 机通信控制/状态显示软件功能描述

本开发板附带用于与开发板进行串口通信的 PC 机通信软件。本软件可以将开发板通过串口上传的信息进行解码，并将开发板上资源（包括数码管、音响、流水灯）的状态进行同步显示。同时还可以实现开发板上 4x4 键盘阵列的控制功能。另外还提供了数据接收区，以十六进制的形式把接收的所有数据进行显示以帮助分析。软件界面如下所示:



音乐合成原理

1.1 音频产生原理

1. 要产生音频脉冲，只要算出某一音频的周期（1/频率），然后将此周期除以 2 即为半周期的时间。利用定时器计时这个半周期时间，每当计时到后就将输出脉冲的 I/O 反相，然后重复计时此半周期时间再对 I/O 反相，就可以在 I/O 引脚上得到此频率的脉冲。
2. 利用 8051 的内部定时器使其工作在计数器模式 MODE1 下，改变计数值 TH0 及 TL0 以产生不同频率。
3. 例如频率为 523Hz，其周期 T=1/523=1912us，因此只要令计数器计时 956us/1us=956，在每次计数 956 次时将 I/O 反相，就可得到中音 1 (DO) (523Hz)。
4. C 调各音符频率与计数值 T 的对照表如下表所示:

表 6.1 C 调各音符频率与计数值 T 的对照表

音符	频率（Hz）	简谱码(T 值)	音符	频率（Hz）	简谱码（T 值）
低 1 DO	262	63628	# 4 FA#	740	64860
# 1 DO#	277	63731	中 5 SO	784	64898
低 2 RE	294	63835	# 5 SO#	831	64934
# 2 RE#	311	63928	中 6 LA	880	64968
低 3 M	330	64021	# 6	932	64994
低 4 FA	349	64103	中 7 SI	988	65030
# 4 FA#	370	64185	高 1 DO	1046	65058
低 5 SO	392	64260	# 1 DO#	1109	65085
# 5 SO#	415	64331	高 2 RE	1175	65110
低 6 LA	440	64400	# 2 RE#	1245	65134
# 6	466	64463	高 3 M	1318	65157
低 7 SI	494	64524	高 4 FA	1397	65178
中 1 DO	523	64580	# 4 FA#	1480	65198
# 1 DO#	554	64633	高 5 SO	1568	65217
中 2 RE	587	64684	# 5 SO#	1661	65235
# 2 RE#	622	64732	高 6 LA	1760	65252
中 3 M	659	64777	# 6	1865	65268
中 4 FA	698	64820	高 7 SI	1967	65283

1.2 简谱的单片机实现

1. 每个音符使用一个字节，字节的高 4 位代表音符的高低，低 4 位代表音符的节拍。如果 1 拍为 0.4 秒，1/4 拍就是 0.1 秒，只要设定延时时间就可以求得节拍的时间。假设 1/4 拍为 1DELAY，则 1 拍应为 4DELAY，依此类推。所以只要求得 1/4 拍的 DELAY 时间，其余的节拍就是它的倍数。下表为 1/4 和 1/8 节拍的时间设定。

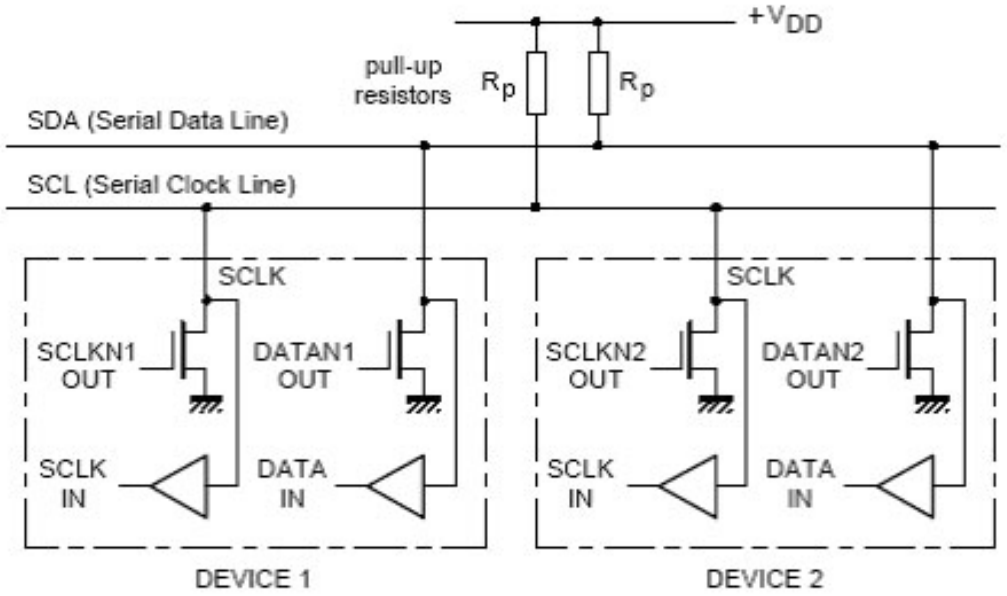
曲调值	DELAY（1/4 拍）	曲调值	DELAY（1/8 拍）
调 4/4	125 毫秒	调 4/4	62 毫秒
调 3/4	187 毫秒	调 3/4	94 毫秒
调 2/4	250 毫秒	调 2/4	125 毫秒

2. 建立音乐的步骤:

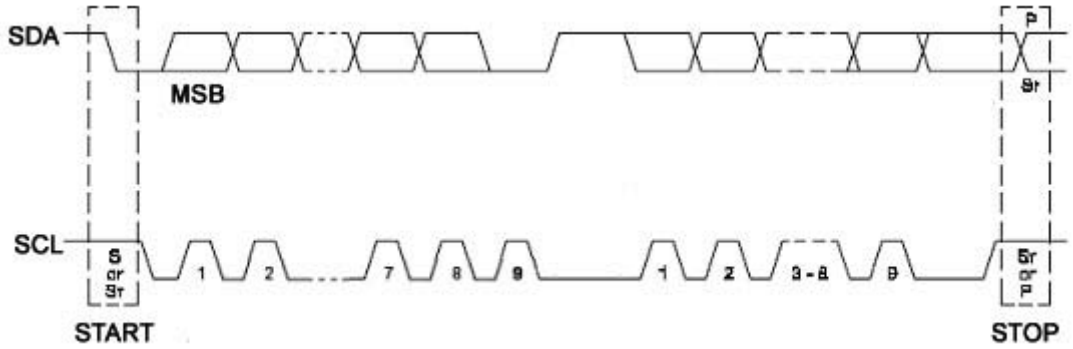
A. 先建立音频表，建立时可不必把所有音频都建在表里，可以从音乐里最低的音频开始建立。
B. 再按音乐的简谱和节拍建立音符表。音符表的高 4 位为音符的音频高低，低 4 位为节拍数。
C. 可以用延时或中断的方式产生 1/4 节拍或 1/8 节拍时间间隔。
D. 程序中从音符表的开始处依次读取音符并翻译成音频和节拍数，直到音乐结束。

I2C 总线简介

1. I²C 总线是 Philips 公司提出的串行总线，但现在实际上已成为一个国际标准，在超过 100 种不同的 IC 上实现并得到超过 50 家公司的许可。它只用两根总线（时钟线 SCL 和数据线 SDA）实现了多主的总线连接。它有 3 种模式：标准模式（100Kbits/s），快速模式（400Kbits/s）和高速模式（3.4Mbits/s），寻址方式有 7 位和 10 位方式。
2. SCL 和 SDA 都是双向线路，都通过一个上拉电阻连接到正的电源电压，如下图所示。当总线空闲时这两条线路都是高电平。



3. 完整的 I2C 数据传输如下图所示：



它包括起始条件、地址传送、数据传送、响应位、停止条件组成。起始条件是指在 SCL 线是高电平时，SDA 从高电平向低电平切换。停止条件是指当 SCL 是高电平时，SDA 从电平向高电平切换。接下来的一个字节包含 7 位地址和一位读/写控制位。接下来是主发到从或从发到主的数据，紧跟着的是响应位，由接收数据的设备发出。最后是停止位表示数据传输的完成。整个过程中传输的 1 或 0 是在 SCL 为高时 SDA 的电平决定的。而 SDA 线的高或低电平状态只有在 SCL 为低电平时才能改变。

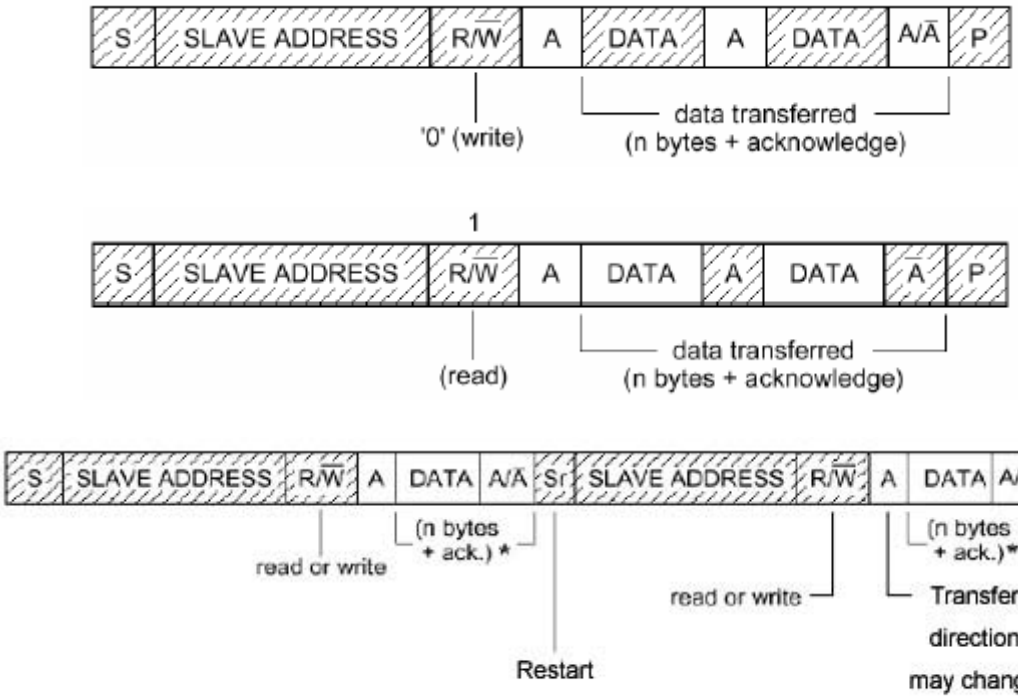
4. 三种典型的数据传输模式。

A. 主机只发送数据给从机指定地址。

B. 主机寻址从机，只从从机读取数据。

C. 以上两种的复合模式，即主机既想向从机写数据又需要从机反馈数据。

三种模式的示意图如下所示：



演示程序清单

```
/*
* Copyright (c) 2005, Owen Studio
* All rights reserved.
*
*  文件名称:MyBoard.c
*  当前版本:Version 1.0
*
*  本程序为通过调试的源程序
*
*  功能描述:上电后系统等待用户输入登陆密码，用户可查询储存在 24C02 中的
*  登陆密码(A 键),也可修改登陆密码(B 键), 当输入登陆密码后按 D 键，如果密
*  码正确则进入系统。进入系统后 6 个数码管开始计时，同时循环播放 4 首音
*  乐，发光二极管也按预定流水灯程序循环闪动。此时按 1-9 键则播放第 1-9 首
*  音乐。按 0 键则停止播放。按 D 键开始/停止音乐自动播放。按*键开始/停止
*  流水灯自动演示。按#键切换流水灯方案臘键则跳到下一首音乐，按 A 键、B
*  键、C 键则分别对时、分、秒进行调节。
*
*另外本程序集成了与 PC 通信的串口服务程序，所有按键的功能也可以通过
* PC 机的控制软件实现。状态也可以反馈到此控制软件上。
*
*  资源配置:Timer0 用于为系统提供时基；Timer1 用于音乐播放；Timer2 用于串*  口波特率发生器。3 个中断
源，其中两个为 Timer0 和 Timer1 中断，另一个为  *  串口中断，主要完成与 PC 机的通信。
*/
/*************************************************************************/
//////////////////////////////////////////////Beginning of Program/////////////////////////////////////////////
/*************************************************************************/
#include "at89x52.h"
#include <intrins.h>
#define    nop    _nop_()
#define    uchar unsigned char
#define    uint    unsigned int
#define    PAI_BASE 15
#define    MUSIC_NUM 4
#define    SCAN_CYCLE 500
/*************************************************************************/
/*Constant Tables*/
static const char c_keyCode[16]={0x01,0x02,0x03,0x0a,                /*键盘 1,2,3,A*/
                                 0x04,0x05,0x06,0x0b,       /*键盘 4,5,6,B*/
```

```
                                                    0x07,0x08,0x09,0x0c,      /*键盘 7,8,9,C*/
                                                    0x0e,0x00,0x0f,0x0d};                  /* 键 盘
*,0,#,D*/
    const uint code c_musicFreq[]={64260,64400,64524,64580,        /*低音 5,6,7,中音 1*/
                                                    64684,64777,64820,64898,        /*中音 2,3,4,5*/
                                                    64968,65030,65058,65110,        /*中音 6,7,高音 1,2*/
                                                    65157,65178,65217};                /*高音 3,4,5*/
    const char code c_music1[]={        /*生日快乐歌*/
                0x82,0x01,0x81,0x94,0x84,
                                    0xb4,0xa4,0x04,
                                    0x82,0x01,0x81,0x94,0x84,
                                    0xc4,0xb4,0x04,
                                    0x82,0x01,0x81,0xf4,0xd4,
                                    0xb4,0xa4,0x94,
                                    0xe2,0x01,0xe1,0xd4,0xb4,
                                    0xc4,0xb4,0x04,
                                    0x00};
    const char code c_music2[]={        /*两只蝴蝶*/
    0x62, 0x52,
                                    0x64, 0x64, 0x02, 0x52, 0x62, 0x52,
                                    0x44, 0x44, 0x04, 0x22, 0x42,
                                    0x54, 0x52, 0x62, 0x52, 0x42, 0x22, 0x42,
                                    0x14, 0x14, 0x04, 0x62, 0x52,
                                    0x64, 0x64, 0x02, 0x52, 0x62, 0x52,
                                    0x44, 0x44, 0x04, 0x22, 0x42,
                                    0x54, 0x52, 0x62, 0x52, 0x42, 0x22, 0x42,
                                    0x54, 0x54, 0x04, 0x62, 0x82,
                                    0x84, 0x84, 0x02, 0x82, 0x92, 0x82,
                                    0x64, 0x64, 0x04, 0x52, 0x52,
                                    0x54, 0x52, 0x62, 0x52, 0x42, 0x22, 0x91, 0x41,
                                    0x42, 0x42, 0x44, 0x44, 0x44,
                                    0x02, 0x82, 0x82, 0x92,
                                    0xb2, 0xa2, 0xa2, 0x92, 0x64, 0x52, 0x62,
                                    0x64, 0x64, 0x02, 0x62, 0x62, 0x82,
                                    0x94, 0x94, 0x02, 0x22, 0x62, 0x52,
                                    0x54, 0x54, 0x04, 0x62, 0x82,
                                    0x82, 0x62, 0x84, 0x02, 0xb2, 0xb2, 0xa2,
                                    0x92, 0xa2, 0x64, 0x02, 0x92, 0x92, 0xa2,
                                    0x92, 0x82, 0x62, 0x52, 0x52, 0x62, 0x52, 0x62,
                                    0x84, 0x84, 0x02, 0x82, 0x82, 0x92,
```

```
                                0x92, 0x82, 0x62, 0x52, 0x52, 0x12, 0x12, 0x22,
                                0x44, 0x44, 0x44, 0x44,
                                0x00};
const char code c_music3[]={        /*老鼠爱大米*/
0x04, 0x42, 0x41, 0x51,
                                0x62, 0x62, 0x52, 0x41, 0x51, 0x54, 0x02, 0x41, 0x51,
                                0x62, 0x62, 0x52, 0x42, 0x44, 0x02, 0x11, 0x21,
                                0x14, 0x11, 0x11, 0x21, 0x41, 0x44, 0x02, 0x42,
                                0x42, 0x42, 0x42, 0x41, 0x51, 0x54, 0x42, 0x51, 0x61,
                                0x62, 0x62, 0x82, 0x91, 0x51, 0x54, 0x62, 0x51, 0x41,
                                0x42, 0x42, 0x52, 0x61, 0x81, 0x84, 0x02, 0x81, 0x81,
                                0x92, 0x41, 0x41, 0x42, 0x61, 0x01, 0x52, 0x42, 0x42, 0x51, 0x41,
                                0x44, 0x44, 0x44, 0x62, 0x61, 0x81,
                                0x82, 0x81, 0x81, 0x82, 0x81, 0x91, 0x94, 0x62, 0x52,
                                0x42, 0x42, 0x42, 0x51, 0x61, 0x64, 0x02, 0x61, 0x81,
                                0x82, 0x82, 0x82, 0x91, 0xb1, 0xb2, 0x92, 0x82, 0x62,
                                0x52, 0x42, 0x42, 0x22, 0x54, 0x02, 0x61, 0x81,
                                0x82, 0x82, 0x81, 0x92, 0x81, 0x94, 0x62, 0x52,
                                0x42, 0x42, 0x42, 0x51, 0x61, 0x64, 0x02, 0x61, 0x81,
                                0x82, 0x82, 0x82, 0x91, 0xb1, 0xb2, 0x92, 0x82, 0x61, 0x51,
                                0x52, 0x52, 0x54, 0x54, 0x41, 0x62, 0x71,
                                0x62, 0x52, 0x54, 0x54, 0x62, 0x51, 0x41,
                                0x44, 0x62, 0x51, 0x41, 0x44, 0x62, 0x52,
                                0x62, 0x42, 0x62, 0x91, 0x81, 0x84, 0x02, 0x61, 0x81,
                                0x92, 0x91, 0x81, 0x82, 0x91, 0x81, 0x84, 0x61, 0x51, 0x42,
                                0x52, 0x51, 0x61, 0x52, 0x41, 0x51, 0x41, 0x53, 0x62, 0x51, 0x41,
                                0x44, 0x62, 0x51, 0x41, 0x44, 0x62, 0x52,
                                0x62, 0x42, 0x61, 0x92, 0x81, 0x84, 0x02, 0x61, 0x81,
                                0x92, 0x92, 0xb2, 0x81, 0x91, 0x82, 0x01, 0x81, 0x61, 0x51, 0x41,
                                0x54, 0x51, 0x41, 0x51, 0x61, 0x54, 0x62, 0x62,
                                0x54, 0x02, 0x42, 0x44, 0x44,
                                    0x00};
const char code c_music4[]={        /*梦醒时分*/
0x04, 0x02, 0x41, 0x51, 0x62, 0x61, 0x61, 0x62, 0x52,
                                0x62, 0x81, 0x61, 0x62, 0x61, 0x81, 0x92, 0x92, 0x92, 0xa2,
                                0x82, 0x62, 0x62, 0x62, 0x61, 0x51, 0x42, 0x41, 0x41, 0x42, 0x22,
                                0x62, 0x81, 0x41, 0x44, 0x51, 0x52, 0x01, 0x52, 0x42,
                                0x81, 0x61, 0x52, 0x52, 0x41, 0x51, 0x62, 0x61, 0x61, 0x62, 0x52,
                                0x62, 0x81, 0x61, 0x62, 0x61, 0x81, 0x92, 0x91, 0x91, 0x92, 0xa2,
                                0x82, 0x62, 0x61, 0x81, 0x81, 0x81, 0x92, 0xb2, 0xb2, 0x92,
```

```
                                0x72, 0x52, 0x52, 0x81, 0x81, 0x92, 0xb2, 0xb2, 0x92,
                                0xb2, 0xc2, 0xc4, 0xc4, 0x01, 0x81, 0x81, 0x81,
                                0xd2, 0xd2, 0xd2, 0xd2, 0xd2, 0xc1, 0xc1, 0xc2, 0xb1, 0xa1,
                                0xb2, 0xb2, 0xb2, 0xc2, 0x62, 0x82, 0x82, 0x81, 0x81,
                                0x92, 0xb1, 0xb1, 0xb2, 0xb2, 0xc2, 0x91, 0xb1, 0xb2, 0xb1, 0xb1,
                                0xc2, 0xc1, 0xc1, 0xc2, 0xb2, 0xd2, 0xc1, 0x81, 0x81, 0x81, 0x81, 0x81,
                                0xd2, 0xd2, 0xd2, 0xf2, 0xd2, 0xc1, 0xc1, 0xc2, 0xb1, 0xa1,
                                0xb2, 0xb2, 0xb2, 0xc2, 0x62, 0x82, 0x82, 0x81, 0x81,
                                0x92, 0xb1, 0xb1, 0xb2, 0xb1, 0xb1, 0xc2, 0x91, 0xb1, 0xb2, 0xb1, 0xb1,
                                0xc2, 0xc1, 0xc1, 0xc2, 0xb2, 0xd4, 0xc2, 0x01, 0xb1,
                                0xb4, 0xb4, 0xb4, 0xb4,
                                0x00};
const char code c_music5[]={        /*大长今*/
0x54, 0x64, 0x64,
                                0x64, 0x02, 0x52, 0x44,
                                0x24, 0x44, 0x44,
                                0x51, 0x43, 0x44, 0x44,
                                0x54, 0x64, 0x64,
                                0x64, 0x02, 0x82, 0x64,
                                0x64, 0x54, 0x64,
                                0x64, 0x64, 0x64,
                                0x84, 0x94, 0x94,
                                0x94, 0x84, 0x64,
                                0x64, 0x84, 0x94,
                                0x81, 0x91, 0x82, 0x84, 0x84,
                                0x54, 0x64, 0x64,
                                0x54, 0x02, 0x62, 0x64,
                                0x54, 0x64, 0x24,
                                0x41, 0x23, 0x94, 0x94,
                                0x24, 0x24, 0x24,
                                0x54, 0x64, 0x64,
                                0x64, 0x02, 0x52, 0x44,
                                0x24, 0x44, 0x44,
                                0x51, 0x43, 0x44, 0x44,
                                0x54, 0x64, 0x64,
                                0x64, 0x02, 0x82, 0x64,
                                0x64, 0x54, 0x64,
                                0x64, 0x64, 0x64,
                                0x84, 0x94, 0x94,
                                0x94, 0x02, 0x82, 0x64,
```

0x64, 0x84, 0x94,
0x84, 0x84, 0x84,
0x54, 0x64, 0x64,
0x54, 0x02, 0x62, 0x64,
0x54, 0x64, 0x24,
0x41, 0x23, 0x24, 0x24,
0x24, 0x24, 0x24,
0x54, 0x02, 0x42, 0x24,
0x54, 0x02, 0x42, 0x24,
0x54, 0x64, 0x44,
0x54, 0x02, 0x62, 0x84,
0x94, 0x02, 0x82, 0x64,
0x54, 0x02, 0x42, 0x24,
0x24, 0x14, 0x24,
0x41, 0x23, 0x24, 0x24,
0x24, 0x24, 0x24,
0x14, 0x14, 0x14,
0x14, 0x02, 0x22, 0x42, 0x52,
0x62, 0x52, 0x44, 0x02, 0x22,
0x44, 0x44, 0x54,
0x64, 0x64, 0x64,
0x64, 0x64, 0x84,
0x94, 0x02, 0xb2, 0x94,
0x84, 0x84, 0x84,
0x84, 0x61, 0x81, 0x92, 0x04,
0x92, 0x82, 0x64, 0x02, 0x52,
0x64, 0x02, 0x52, 0x62, 0x52,
0x44, 0x44, 0x44,
0x41, 0x23, 0x24, 0x44,
0x54, 0x54, 0x54,
0x54, 0x64, 0x64,
0x44, 0x41, 0x23, 0x24,
0x24, 0x24, 0x24,
0x00};

const char code c_music6[]={ /*漫步人生路*/
0x24, 0x42, 0x52, 0x64, 0x62, 0x92,
0x84, 0x62, 0x52, 0x64, 0x02, 0x62,
0x72, 0x62, 0x62, 0x52, 0x52, 0x42, 0x41, 0x31, 0x22,
0x44, 0x02, 0x52, 0x34, 0x34,
0x24, 0x42, 0x52, 0x64, 0x62, 0x92,

```
                                0xa4, 0x82, 0x62, 0x64, 0x02, 0x62,
                                0x72, 0x62, 0x52, 0x42, 0x32, 0x42, 0x52, 0x62,
                                0x42, 0x32, 0x22, 0x12, 0x24, 0x24,
                                0x42, 0x32, 0x42, 0x52, 0x64, 0x52, 0x62,
                                0x74, 0x74, 0x72, 0x81, 0x71, 0x62, 0x52,
                                0x64, 0x74, 0x84, 0x02, 0x62,
                                0x94, 0x92, 0x82, 0x94, 0x92, 0xb2,
                                0x92, 0x82, 0x82, 0x92, 0x64, 0x52, 0x42,
                                0x54, 0x52, 0x42, 0x54, 0x42, 0x42, 0x82,
                                0x62, 0x62, 0x52, 0x42, 0x34, 0x62, 0x62,
                                0x94, 0x92, 0x82, 0x94, 0x92, 0xb2,
                                0x92, 0x82, 0x82, 0x92, 0x64, 0x52, 0x42,
                                0x54, 0x42, 0x82, 0x64, 0x52, 0x42,
                                0x52, 0x61, 0x51, 0x42, 0x32, 0x24, 0x24,
                                0x00};
const char code c_music7[]={    /*最浪漫的事*/
0x02, 0x12, 0x12, 0x22,
                                0x44, 0x42, 0x42, 0x42, 0x22, 0x22, 0x62,
                                0x64, 0x64, 0x02, 0x12, 0x24,
                                0x42, 0x42, 0x42, 0x42, 0x44, 0x62, 0x82,
                                0x84, 0x84, 0x84, 0x02, 0x82,
                                0x92, 0x82, 0x92, 0x82, 0x94, 0x82, 0x51, 0x61,
                                0x62, 0x51, 0x41, 0x44, 0x02, 0x42, 0x42, 0x22,
                                0x44, 0x42, 0x22, 0x42, 0x64, 0x52,
                                0x54, 0x54, 0x54, 0x54,
                                0x32, 0x12, 0x22, 0x42,
                                0x42, 0x42, 0x42, 0x42, 0x42, 0x42, 0x22, 0x52,
                                0x52, 0x62, 0x64, 0x62, 0x12, 0x22, 0x42,
                                0x42, 0x42, 0x42, 0x42, 0x44, 0x62, 0x82,
                                0x84, 0x84, 0x84, 0x82, 0x82,
                                0x92, 0x82, 0x92, 0x82, 0x94, 0x92, 0x82,
                                0x52, 0x62, 0x52, 0x44, 0x22, 0x22, 0x42,
                                0x14, 0x62, 0x52, 0x44, 0x22, 0x22,
                                0x41, 0x51, 0x42, 0x44, 0x44, 0x44,
                                0x82, 0x92, 0xb2,
                                0x94, 0x92, 0x82, 0x92, 0x82, 0x62, 0x82,
                                0x84, 0x84, 0x02, 0x82, 0x92, 0xb2,
                                0x94, 0x92, 0x82, 0x92, 0x82, 0x92, 0x42,
                                0x44, 0x44, 0x02, 0x42, 0x52, 0x62,
                                0x52, 0x52, 0x52, 0x42, 0x52, 0x42, 0x22, 0x62,
```

```
                                0x62, 0x52, 0x54, 0x02, 0x82, 0x92, 0x62,
                                0x84, 0x82, 0x62, 0x82, 0x62, 0x64,
                                0x92, 0x82, 0x92, 0x82, 0x62, 0x82, 0x81, 0x92, 0xb1,
                                0x94, 0x92, 0x82, 0x92, 0x82, 0x62, 0x82,
                                0x84, 0x84, 0x02, 0x82, 0x92, 0xb2,
                                0x94, 0x92, 0x82, 0x92, 0x82, 0x92, 0x42,
                                0x44, 0x44, 0x02, 0x42, 0x52, 0x62,
                                0x74, 0x72, 0x82, 0x92, 0x92, 0x82, 0x92,
                                0x92, 0xb2, 0x92, 0x92, 0x92, 0xb2, 0x92, 0x82,
                                0x84, 0x84, 0x82, 0x42, 0x92, 0x82,
                                0x84, 0x84, 0x82, 0x92, 0xb2, 0x01, 0x61,
                                0x52, 0x61, 0x51, 0x44, 0x44, 0x44,
                                0x00};
const char code c_music8[]={        /*1000 年以后*/
0xb2, 0x41, 0x51, 0x62, 0x81, 0x91, 0x84, 0x84,
                                0x72, 0x61, 0x41, 0x42, 0x61, 0x51, 0x54, 0x54,
                                0xb2, 0x41, 0x51, 0x62, 0x92, 0xa1, 0x91, 0x82, 0x82, 0x61, 0x91,
                                0x94, 0x94, 0xa1, 0xa1, 0xa1, 0xb1, 0xc1, 0xb1, 0xa2,
                                0xb2, 0xa1, 0xb1, 0xb1, 0xd2, 0x01, 0xa2, 0x91, 0xa1, 0xa1, 0x81, 0x82,
0x91, 0x91, 0xa1, 0xb1, 0xc1, 0xb1, 0xc1, 0xe1, 0xd1, 0xc1, 0xc1, 0xc1, 0xc2, 0xc1, 0xb1,
                                0xb2, 0xa1, 0xb1, 0xb1, 0xd2, 0x01, 0xa2, 0x91, 0xa1, 0xa1, 0x81, 0x82,
                                0x92, 0xb4, 0x02, 0xb1, 0xb1, 0xb1, 0xc1, 0xd2, 0xc1, 0xb1,
                                0xb2, 0xc2, 0xc4, 0x02, 0xa1, 0xb1, 0xb1, 0xc2, 0x01,
                                0xd2, 0xc1, 0xb1, 0xb1, 0x82, 0x01, 0xc4, 0x02, 0xd1, 0xc1,
                                0xb2, 0xa1, 0x91, 0x92, 0x62, 0xa4, 0x02, 0xb1, 0xa1,
                                0x92, 0xa1, 0xb1, 0xb1, 0xc2, 0x01, 0x82, 0xc1, 0xd1, 0xd2, 0x01, 0xd1,
                                0xe2, 0xd2, 0xb1, 0xc2, 0xc1, 0xc2, 0x01, 0xa1, 0xb1, 0xc2, 0x01,
                                0xd2, 0xc1, 0xb1, 0xb1, 0x82, 0x01, 0xc4, 0xc1, 0xc1, 0xd1, 0xc1,
                                0xb2, 0xa2, 0x91, 0xc2, 0x01, 0xa4, 0x02, 0xb1, 0xa1,
                                0x92, 0x81, 0x91, 0x91, 0xb1, 0x82, 0x82, 0xc1, 0xb1, 0xb2, 0xd1, 0xd1,
                                0xe1, 0xd1, 0xc4, 0xb1, 0xc1, 0xd2, 0xc1, 0xb1, 0xd2, 0xc1, 0x81,
                                0x82, 0xb2, 0xb4, 0xb4, 0xb4,
                                0x00};
const char code c_music9[]={        /*一辈子的孤单*/
0x02, 0x11, 0x11, 0x62, 0x52, 0x52, 0x41, 0x61,
                                0x62, 0x41, 0x51, 0x54, 0x54, 0x04,
                                0x02, 0x11, 0x11, 0x61, 0x52, 0x51, 0x52, 0x42,
                                0x52, 0x62, 0x64, 0x64, 0x04,
                                0x02, 0x11, 0x11, 0x62, 0x52, 0x52, 0x41, 0x61,
                                0x62, 0x41, 0x51, 0x54, 0x54, 0x04,
```

```
                                0x02, 0x51, 0x41, 0x62, 0x52, 0x42, 0x52,
                                0x42, 0x42, 0x44, 0x44, 0x44,
                                0x04, 0x62, 0x52, 0x52, 0x42,
                                0x42, 0x52, 0x54, 0x54, 0x54,
                                0x04, 0x51, 0x63, 0x54, 0x44,
                                0x41, 0x11, 0x21, 0x11, 0x14, 0x14, 0x14,
                                0x04, 0x62, 0x52, 0x52, 0x42,
                                0x42, 0x52, 0x54, 0x54, 0x54,
                                0x02, 0x12, 0x62, 0x54, 0x42,
                                0x44, 0x02, 0x62, 0x64, 0x64,
                                0x02, 0x12, 0x62, 0x84, 0x92,
                                0x62, 0x52, 0x54, 0x54, 0x54,
                                0x02, 0x12, 0x62, 0x54, 0x42,
                                0x54, 0x02, 0x62, 0x64, 0x64,
                                0x02, 0x12, 0x62, 0x52, 0x52, 0x42,
                                0x51, 0x62, 0x51, 0x54, 0x54, 0x04,
                                0x02, 0x51, 0x41, 0x62, 0x52, 0x42, 0x52,
                                0x42, 0x42, 0x44, 0x44, 0x44,
                                0x02, 0x12, 0x62, 0x52, 0x52, 0x01, 0x41,
                                0x51, 0x62, 0x51, 0x54, 0x54, 0x54,
                                0x02, 0x12, 0x62, 0x52, 0x52, 0x01, 0x41,
                                0x52, 0x62, 0x64, 0x64, 0x64,
                                0x02, 0x12, 0x61, 0x52, 0x01, 0x52, 0x42,
                                0x51, 0x62, 0x51, 0x54, 0x54, 0x54,
                                0x02, 0x12, 0x61, 0x52, 0x01, 0x52, 0x42,
                                0x52, 0x62, 0x64, 0x64, 0x64,
                                0x02, 0x12, 0x62, 0x82, 0x82, 0x92,
                                0x62, 0x52, 0x54, 0x54, 0x54,
                                0x02, 0x12, 0x62, 0x52, 0x52, 0x42,
                                0x52, 0x62, 0x64, 0x64, 0x64,
                                0x02, 0x12, 0x62, 0x52, 0x52, 0x42,
                                0x62, 0x52, 0x54, 0x54, 0x54,
                                0x02, 0x51, 0x41, 0x62, 0x52, 0x42, 0x52,
                                0x42, 0x42, 0x44, 0x44, 0x44,
                                0x00};
const char code c_music0[]={0x00};
/*Constant Tables Over*/
/*************************************************************************/
/*System Variables*/
uchar i;
```

```
uint keyFlag;
uchar hour=10;min=0;sec=0;
uchar DisplayArray[6];
uchar password[6];
static uchar LEDFlag=0x00;
bit LEDCycle100msFlag=0,LEDCycle500usFlag=0,LEDCycle10msFlag=0;
bit LEDAutoFlag=1,LEDKeyFlag=0;
static uchar s_musicCounter=0x00,s_tableCounter;
bit musicAutoFlag=1,musicKeyFlag=0;
uchar Int0Flag,TH0Global,TL0Global;
uchar UARTKeyFlag=0xaa,modifyTimeFlag=0;;
/*System Variables Over*/
/*********************************************************************/
/*System Functions*/
void IdleForLogin();                    /*登陆系统前的密码检测模块*/
void Login(void);                       /*系统登录模块*/
void MainSystem(void);                  /*系统主程序*/
void ScanKeys(void);                    /*扫描键盘模块*/
uchar KeyJudge(void);                   /*按键预处理*/
void KeyHandle(uchar keyIndex);         /*按键处理模块*/
void LEDHandle(void);                   /*流水灯方案选择*/
void AutoShow(void);                    /*开始流水灯自动循环演示*/
void StopAutoShow(void);                /*停止流水灯自动循环演示*/
void DropDown(void);                    /*水滴方案*/
void CircleFlash(void);                 /*环扫方案*/
void GradualFlash(void);                /*渐明渐暗方案*/
void SlowFlash(void);                   /*慢闪方案*/
void FastFlash(void);                   /*快闪方案*/
void ScrShift(uchar keyIndex);          /*六位数码管顺序左移*/
void ClearScr(void);                    /*清屏*/
void ReadPassword();                    /*从 24C02 中读取密码*/
void WritePassword();                   /*将屏幕的六位数字作为密码存入 24C02 中*/
void StatusSend(void);                  /*下位机状态反馈（串口）*/
void Delay (uint value);                /*10us 延时*/
void ModifyTime(void);                  /*调整时间*/
void ScanSeg7();                        /*数码管扫描显示*/
/*System Functions Over*/
/*********************************************************************/
/*IIC Varialbes & Functions*/
sbit SCL=P3^4;                          /*定义 IIC 总线的时钟线*/
```

```
sbit SDA=P3^5;                          /*定义 IIC 总线的数据线*/
void StartIIC();                        /*发送 IIC 总线启动时序*/
void StopIIC();                         /*发送 IIC 总线停止时序*/
void IICClockDelay();                   /*IIC 总线时钟电平延时*/
void IICAck(bit ackValue);              /*MCU 对 24C02 的应答函数*/
uchar IICRcvByte();                     /*MCU 从 24C02 接收一个字节*/
bit IICSendByte(uchar byteData);        /*MCU 向 24C02 发送一个字节*/
/*MCU 向 24C02 的指定地址写入若干个字节数据函数*/
bit IICWrite(uchar slaveAddress,uchar subAddress,uchar *dataPointer,uchar num);
/*MCU 从 24C02 的指定地址度去若干个字节数据函数*/
bit IICRead(uchar slaveAddress,uchar subAddress,uchar *dataPointer,uchar num);
/*IIC Varialbes & Functions Over*/
/*****************************************************************************/
/*****************************************************************************/
void main(void)                         /*主程序入口*/
{
TMOD=0x11;                              /*Timer0 工作在 MODE1,16 位定时器*/
                                        /*Timer1 工作在 MODE2,自动重新装载模式*/
                                        /*|          定时器 1 |定时器 1          |*/
                                        /*| GATE|C/T| M1 | M0 | GATE|C/T| M1 | M0 |*/
TH0=(65536-SCAN_CYCLE)/256;             /*设定 Timer0 每隔 500us 中断一次*/
    TL0=(65536-SCAN_CYCLE)%256;
    TR0=1;                             /*启动 Timer0*/
    TH1=(65536-10000)/256;              /*设定 Timer1 每 10ms 中断一次*/
    TL1=(65536-10000)%256;
T2CON=0x38;                             /*令定时器 2 作为串口波特率发生器*/
                                        /*|TF2|EXF2|RCLK|TCLK|EXEN2|TR2|C/T2|CP/RL2|*/
TL2=0xd9;                               /*设定串口波特率为 9600，晶振为 12MHz*/
TH2=0xff;
RCAP2L=0xd9;                            /*根据串口波特率 9600 设定自动重载寄存器*/
RCAP2H=0xff;
    TR2=1;                              /*启动定时器 2*/
SCON=0x50;                              /*设定 UART 工作在 MODE3 模式，可收发数据*/
                                        /*| SM0| SM1| SM2| REN| TB8| RB8| TI | RI |*/
    IE=0x9f;                            /*使能总中断、Timer0,1 中断、外部中断 1,2*/
                                        /*| EA | - | ET2| ES | ET1| EX1| ET0| EX0|*/
    IP=0x02;                            /*优先 Timer0 中断*/
                                        /*| - | - | PT2| PS | PT1| PX1| PT0| PX0|*/
    ReadPassword();                     /*先把 24C02 中的密码读到密码缓冲区中*/
    ClearScr();                         /*预清屏*/
```

```
        P3_3=1;                                 /*关喇叭*/
    P2=0x00;                                    /*熄灭 8 个发光二极管*/
        while(1)
    {
    IdleForLogin();
    }
    }
    /*********************************************************************/
    void IdleForLogin()                         /*登陆系统前的密码检测模块*/
    {
        static uint s_UARTSendCounter;
        uchar keyPtr;
        keyPtr=KeyJudge();                      /*键盘扫描判断*/
    if(keyPtr!=0x55)
        {
    switch(c_keyCode[keyPtr])
            {
    case 0x0a: ReadPassword();                  /*按 A 键：读取 24C02 中的密码并显示*/
                                for(i=0;i<5;i++)        /*显示密码 1 秒后恢复*/
                            {
                                Delay(20000);
                            }
                                ClearScr();
                                break;
            case 0x0b: WritePassword();             /*按 B 键：将屏幕中的六位数字设为密码*/
                                        ClearScr();
                                    break;
            case 0x0c: ClearScr();                  /*按 C 键：清屏*/
                                    break;
            case 0x0d: Login();                     /*按 D 键：比较密码，如果密码正确则进入系统*/
break;
            case 0x0e: break;                       /*按*键：保留*/
    case 0x0f:    break;                        /*按#键：保留*/
            default:    ScrShift(keyPtr);           /*以上均不是则为数字键，六位数码管顺序左移*/
                                break;
            }
            UARTKeyFlag=0xaa;                   /*清串口命令标志位*/
    }
    if(++s_UARTSendCounter>200)                 /*定期上传系统状态*/
  {
```

```
s_UARTSendCounter=0;
StatusSend();
}
}
/*********************************************************************/
void Login(void)                          /*系统登录模块*/
{
    for(i=0;i<6;i++)                      /*比较密码是否正确*/
    {
        if(DisplayArray[i]!=password[i])
        {
                return;
}
    }
    TR1=1;                                /*启动系统时基 Timer1，中断周期 10ms*/
ClearScr();                               /*清屏*/
s_musicCounter=0x01;
LEDFlag=0x01;
    ModifyTime();                         /*初始设定时间*/
    while (1)
    {
MainSystem();                             /*进入系统主程序*/
}
}
/*********************************************************************/
void MainSystem(void)                     /*系统主程序*/
{
    static uint s_UARTSendCounter;
uchar keyPtr;
    keyPtr=KeyJudge();                    /*键盘扫描判断*/
if(keyPtr!=0x55)                          /*如果按某键则进行相应处理*/
    {
KeyHandle(keyPtr);
    }
LEDHandle();                              /*流水灯处理模块*/
if(++s_UARTSendCounter>200)               /*定期上传系统状态*/
{
s_UARTSendCounter=0;
StatusSend();
}
```

```
if(modifyTimeFlag==1)
{
ModifyTime();                                   /*根据新秒值调整时间*/
modifyTimeFlag=0;
}
}
/*************************************************************************/
void ScanKeys(void)                             /*扫描键盘模块*/
{
uchar scanValue=0xef,iScan,scanTempUchar;
    for(iScan=0;iScan<4;iScan++)
    {
        P1=scanValue;                           /*发扫描值*/
        scanTempUchar=P1;                       /*读入扫描值*/
        if((scanTempUchar&0x01)==0x00)          /*第一列被按*/
        {
            keyFlag|=(0x0001<<(iScan*4+0));     /*置此键的标志位*/
        }
        else
        {
            keyFlag&=~(0x0001<<(iScan*4+0));    /*清此键的标志位*/
        }
        if((scanTempUchar&0x02)==0x00)          /*第二列被按*/
        {
            keyFlag|=(0x0001<<(iScan*4+1));     /*置此键的标志位*/
        }
        else
        {
            keyFlag&=~(0x0001<<(iScan*4+1));    /*清此键的标志位*/
        }
        if((scanTempUchar&0x04)==0x00)          /*第三列被按*/
        {
            keyFlag|=0x0001<<(iScan*4+2);       /*置此键的标志位*/
        }
        else
        {
            keyFlag&=~(0x0001<<(iScan*4+2));    /*清此键的标志位*/
        }
        if((scanTempUchar&0x08)==0x00)          /*第四列被按*/
        {
```

```
                keyFlag|=0x0001<<(iScan*4+3);                    /*置此键的标志位*/
            }
            else
            {
                keyFlag&=~(0x0001<<(iScan*4+3));                 /*清此键的标志位*/
            }
            scanValue=scanValue<<1|0x01;                         /*置下一次扫描值*/
        }
    }
    /*************************************************************************/
    uchar KeyJudge(void)                                         /*按键预处理*/
    {
        uchar j,counterKeyPressedNum;
        uchar keyRet=0x55;                                       /*初始按键返回码设定为无按码 0x55*/
        uint uintTemp;
        ScanKeys();                                              /*扫描键盘*/
        if(keyFlag!=0)
        {
            Delay(1000);                                         /*延时 10ms 以消除抖动*/
            ScanKeys();
            P1=0x0f;
    while((P1&0x0f)!=0x0f);                                      /*没松开按键就等按键松开*/
            counterKeyPressedNum=0;
            for(j=0;j<16;j++)
            {
                uintTemp=((uint)0x0001)<<j;
                if((keyFlag&uintTemp)==uintTemp)                 /*依次检测键盘扫描标志 KeyFlag*/
                {                                                /*如某位置一则按键个数寄存器加一*/
                    counterKeyPressedNum++;
                        keyRet=j;
                }
            }
            if(counterKeyPressedNum>1)                           /*如果不止一个键被按则返回无按码 0x55*/
            {
                return 0x55;
            }
    else                                                         /*某键被按，返回此键的标识码*/
    {
    return(keyRet);
    }
```

```
}
if(UARTKeyFlag!=0xaa)
{
return(UARTKeyFlag);
}
return(0x55);
}
/*********************************************************************/
void KeyHandle(uchar keyIndex)              /*按键处理模块*/
{
    switch(c_keyCode[keyIndex])          /*按数字键 N：循环播放第 N 首音乐*/
    {
case 0x01: s_musicCounter=1; musicAutoFlag=0; s_tableCounter=0x00; break;
case 0x02: s_musicCounter=2; musicAutoFlag=0; s_tableCounter=0x00; break;
case 0x03: s_musicCounter=3; musicAutoFlag=0; s_tableCounter=0x00; break;
case 0x04: s_musicCounter=4; musicAutoFlag=0; s_tableCounter=0x00; break;
case 0x0a: sec++;                                   /*按 A 键：调整时钟秒值*/
                    ModifyTime();
                break;
    case 0x0b: min++;                               /*按 B 键：调整时钟分值*/
ModifyTime();
                 break ;
    case 0x0c: hour++;                              /*按 C 键：调整时钟时值*/
                    ModifyTime();
                break;
    case 0x0d: if(musicKeyFlag==1)                  /*按 D 键：开始/停止音乐自动循环播放*/
                     {
                         s_musicCounter=1;
                    musicAutoFlag=1;
                         musicKeyFlag=~musicKeyFlag;
                }
                    else
                    {
                         s_musicCounter=10;
                    musicAutoFlag=0;
                    musicKeyFlag=~musicKeyFlag;
                     }
                     s_tableCounter=0x00;
                 break;
    case 0x0e: if(LEDKeyFlag==1)                    /*按*键：开始/停止流水灯自动演示*/
```

```
                {
                AutoShow();
                LEDKeyFlag=~LEDKeyFlag;
            }
            else
                {
                StopAutoShow();
                    LEDKeyFlag=~LEDKeyFlag;
                }
            break;
    case 0x0f: if(++LEDFlag>5)                  /*按#键：流水灯方案循环切换*/
                {
                    LEDFlag=1;
                }
            break;
    default:    break;
    }
}
/*********************************************************************/
void LEDHandle(void)                            /*流水灯方案选择*/
{
    switch(LEDFlag)                             /*判断该演示的流水灯方案*/
{
    case 0x01: DropDown(); break;               /*水滴方案*/
case 0x02: CircleFlash(); break;                /*环扫方案*/
case 0x03: GradualFlash(); break;               /*渐明渐暗方案*/
case 0x04: SlowFlash(); break;                  /*慢闪方案*/
case 0x05: FastFlash(); break;                  /*快闪方案*/
    }
}
/*********************************************************************/
void AutoShow(void)                             /*开始流水灯自动循环演示*/
{
    LEDFlag=1;
    LEDAutoFlag=1;
}
/*********************************************************************/
void StopAutoShow(void)                         /*停止流水灯自动循环演示*/
{
    LEDAutoFlag=0;
```

```
}
/*********************************************************************/
void DropDown(void)                                    /*水滴方案*/
{
    static uchar s_dropdownCounter=0,s_dropdown_i=7,s_dropdown_OverCounter;
    uchar rest;
    if(LEDCycle100msFlag==1)                           /*判断 100ms 是否到*/
{
LEDCycle100msFlag=0;
        rest=s_dropdownCounter%8;
P2=~(0x01<<rest) & (0xff>>(7-s_dropdown_i));           /*使水滴流到某个灯并且*/
                                                       /*让流到底的灯常亮*/
if(s_dropdownCounter>=(8-s_dropdown_i)*7)               /*判断一个水滴流程是否完成*/
{
s_dropdownCounter+=8-s_dropdown_i;                     /*每流完一个流程下一个水滴的*/
s_dropdown_i--;                                        /*流程就要相应减少一个灯*/
                    if(s_dropdown_i==0)                /*整个水滴流程完了吗*/
                    {
                        s_dropdown_i=7;
                        s_dropdownCounter=0;
                        if(++s_dropdown_OverCounter>=1)      /*一个完整水滴流程完了之后*/
{                                                            /*如果自动演示功能打开则跳到下一方案*/
                            s_dropdown_OverCounter=0;
                            if(LEDAutoFlag==1)
                            {
                                LEDFlag++;
                             }
                         }
                     }
}
else
{
s_dropdownCounter++;
}
}
}
/*********************************************************************/
void GradualFlash(void)                                /*渐明渐暗方案*/
{
    static uchar s_gradualFlash_i=10,s_flashCounter,s_gradualFlash_OverCounter;
```

```
static bit s_gradualFlash_i_Flag=0;
    if(LEDCycle500usFlag==1)                    /*判断 500us 是否到*/
{
LEDCycle500usFlag=0;
if(((++s_flashCounter)%10)!=0)                  /*PWM 周期为 10*500us=5ms，判断其到否*/
{
if((s_flashCounter%10)<s_gradualFlash_i)        /*在 PWM 周期内按设定*/
{                                               /*占空比调节灯的亮灭*/
                P2=0xff;
            }
            else
            {
P2=0x00;
            }
}
else                                            /*PWM 周期到了*/
{
            P2=0xff;
            if((s_flashCounter/10)>=10)         /*改变 PWM 占空比前先判断延时 10*5ms*/
{                                               /*=50ms 到否，没到在此周期内占空比保持不变*/
                s_flashCounter=0;
if(s_gradualFlash_i_Flag==0)                    /*需要渐明则调高占空比*/
                {
s_gradualFlash_i--;
                }
                else                            /*需要渐暗则调低占空比*/
                {
                    s_gradualFlash_i++;
}
if(s_gradualFlash_i==0 || s_gradualFlash_i>=10)
                {                               /*如果渐明或渐暗完毕则交换*/
                    s_gradualFlash_i_Flag=~s_gradualFlash_i_Flag;
                    if(++s_gradualFlash_OverCounter>=6)     /*渐明渐暗 6/2=3 个周期后*/
{                                                           /*如果自动演示功能打开则跳到下一方案*/
s_gradualFlash_OverCounter=0;
                    if(LEDAutoFlag==1)
                        {
                            LEDFlag++;
                        }
                    }
```

```
                    }
                }
    }
    }
    }
    /*********************************************************************/
    void CircleFlash(void)                                  /*环扫方案*/
    {
        static uchar s_circleFlashCounter,s_circle_OverCounter;
        if(LEDCycle100msFlag==1)                            /*判断 100ms 是否到*/
    {
    LEDCycle100msFlag=0;
    P2=~(0x01<<s_circleFlashCounter);                       /*让某个灯亮应该令相应引脚为低*/
    if(++s_circleFlashCounter>=8)                           /*8 个灯都亮过以后再重新开始*/
    {
    s_circleFlashCounter=0;
                    if(++s_circle_OverCounter>=5)           /*循环 5 遍后如果自动演示*/
    {                                                       /*功能打开则跳到下一方案*/
                        s_circle_OverCounter=0;
    if(LEDAutoFlag==1)
    {
                            LEDFlag++;
                    }
    }
    }
    }
    }
    /*********************************************************************/
    void SlowFlash(void)                                    /*慢闪方案*/
    {
        static uchar s_slowFlashCounter;
        if(LEDCycle100msFlag==1)                            /*判断 100ms 是否到*/
    {
    LEDCycle100msFlag=0;
    if((++s_slowFlashCounter%5)==0)                         /*判断 5*100ms=500ms 是否到*/
    {
                    P2=~P2;                                 /*每 500ms 灯亮灭改变一次*/
    }
    if(s_slowFlashCounter/5>=6)                             /*慢闪 6/2=3 个周期以后如果自动*/
    {                                                       /*演示功能打开则跳到下一方案*/
```

```
s_slowFlashCounter=0;
                        if(LEDAutoFlag==1)
                        {
                            LEDFlag++;
                        }
}
}
}
/*********************************************************************/
void FastFlash(void)                                    /*快闪方案*/
{
    static uchar s_fashFlashCounter;
    if(LEDCycle100msFlag==1)                            /*判断 100ms 是否到*/
{
LEDCycle100msFlag=0;
P2=~P2;                                                 /*每 100ms 灯亮灭改变一次*/
if(++s_fashFlashCounter>=30)                            /*快闪 30/2=15 个周期以后如果自动*/
{                                                       /*演示功能打开则跳到下一方案*/
s_fashFlashCounter=0;
if(LEDAutoFlag==1)
                        {
                            LEDFlag=1;
                        }
}
}
}
/*********************************************************************/
void ScrShift(uchar keyIndex)                           /*六位数码管顺序左移*/
{
    for(i=5;i>0;i--)
    {
        DisplayArray[i]=DisplayArray[i-1];
    }
    DisplayArray[0]=c_keyCode[keyIndex];
}
/*********************************************************************/
void ClearScr(void)                                     /*清屏*/
{
    for(i=0;i<6;i++)
```

```
    {
        DisplayArray[i]=0x00;
    }
}
/*****************************************************************/
void ReadPassword()                                 /*从 24C02 中读取密码*/
{
  uchar i;
   IICRead(0xa0,0,password,6);
   for(i=0;i<6;i++)
   {
        DisplayArray[i]=password[i];
   }
}
/*****************************************************************/
void WritePassword()                                /*将屏幕的六位数字作为密码存入 24C02 中*/
{
  IICWrite(0xa0,0,DisplayArray,6);
}
/*****************************************************************/
void StatusSend(void)                               /*下位机状态反馈（串口）*/
{
    IE&=0xef;                                       /*发送状态之前先关闭串口中断*/
    for(i=6;i>0;i--)
{
SBUF=DisplayArray[i-1];                             /*发送屏幕时间*/
while(TI!=1);
TI=0;
}
SBUF=s_musicCounter;
while(TI!=1);
TI=0;
SBUF=LEDFlag;
while(TI!=1);
TI=0;
IE|=0x10;                                           /*发送状态后再打开串口中断*/
}
/*****************************************************************/
void Delay (unsigned int value)                     /*10us 延时*/
{
```

```
    while (value!=0)
{
value--;
}
}
/*********************************************************************/
void ModifyTime(void)                          /*调整时间*/
{
    uchar ucharTemp;
    ucharTemp=sec;
    sec%=60;                                   /*秒除 60 取余数为 sec*/
    min+=ucharTemp/60;                         /*分加上秒的进位先暂存于 min*/
ucharTemp=min;
    min%=60;                                   /*分再除 60 取余数为 min*/
    hour=(hour+ucharTemp/60)%24;               /*时加上分的进位除 24 取余数为 hour 时*/
    DisplayArray[0]=(sec%10);                  /*将时间存入显示缓冲区*/
    DisplayArray[1]=(sec/10);
    DisplayArray[2]=(min%10);
    DisplayArray[3]=(min/10);
    DisplayArray[4]=(hour%10);
    DisplayArray[5]=(hour/10);
}
/*********************************************************************/
void ScanSeg7()                                /*数码管扫描显示*/
{
    static uchar s_scanCounter;
    s_scanCounter++;
if(s_scanCounter>5)
{
  s_scanCounter=0;                             /*ptr1>5 表示 6 个数码管都已经扫描过*/
    }
    P0=s_scanCounter|(DisplayArray[s_scanCounter]<<4);
}
/*********************************************************************/
void IICClockDelay()                           /*IIC 总线时钟电平延时*/
{
    nop;nop;nop;nop;nop;nop;
}
/*********************************************************************/
void IICAck(bit ackValue)                      /*MCU 对 24C02 的应答/不应答函数*/
```

```
{
if(ackValue==0)                                    /*ackValue=0 表示不应答*/
        SDA=0;
    else                                           /*ackValue=1 表示应答*/
        SDA=1;
    IICClockDelay();
    SCL=1;
    IICClockDelay();
    SCL=0;
    IICClockDelay();
}
/*****************************************************************************/
void StartIIC()                                    /*发送 IIC 总线启动时序*/
{
    SCL=1;
IICClockDelay();
SDA=1;
IICClockDelay();
    SDA=0;
IICClockDelay();
SCL=0;
IICClockDelay();
}
/*****************************************************************************/
void StopIIC()                                     /*发送 IIC 总线停止时序*/
{
    SDA=0;
IICClockDelay();
SCL=1;
IICClockDelay();
SDA=1;
IICClockDelay();
SCL=1;
IICClockDelay();
}
/*****************************************************************************/
bit IICSendByte(uchar byteData)                    /*向 24C02 发送一个字节数据或地址函数*/
{                                                  /*返回值为发送成功与否标志*/
    bit ackFlag;
    for(i=0;i<8;i++)                               /*要传送的数据长度为 8 位*/
```

```
    {
        if((byteData<<i)&0x80)              /*依次判断待发送位高低*/
            SDA=1;
        else
            SDA=0;
        IICClockDelay();
        SCL=1;                              /*置时钟线为高，通知 24C02 开始接收数据位*/
        IICClockDelay();
        SCL=0;
IICClockDelay();
    }
    SDA=1;                                  /*8 位数据发送完后释放数据线，准备接收应答位*/
    IICClockDelay();
    SCL=1;
    IICClockDelay();
    if(SDA==1)
        ackFlag=0;                          /*24c02 无应答*/
    else
        ackFlag=1;                          /*数据成功发送*/
    SCL=0;
    IICClockDelay();
return(ackFlag);                            /*返回 24C02 应答标志*/
}
/*****************************************************************************/
uchar   IICRcvByte()                        /*从 24C02 读一个字节数据函数，返回值为读入的该字节*/
{
    uchar retbyteData=0x00;                 /*置返回值初值为 0x00*/
    for(i=0;i<8;i++)
    {
        SDA=1;
        IICClockDelay();
        SCL=1;                              /*拉高时钟线后读取数据线电平*/
        IICClockDelay();
retbyteData=retbyteData<<1;                 /*返回值依次左移一位*/
if(SDA==1)                                  /*读取数据线电平至于返回值空出的一位*/
{
retbyteData+=1;
}
SCL=0;                                      /*拉低释放时钟线*/
IICClockDelay();
```

```
    }
    return(retbyteData);
}
/*****************************************************************/
/*向 24C02 的指定地址写入若干个数据函数*/
bit IICWrite(uchar slaveAddress,uchar subAddress,uchar *dataPointer,uchar num)
{
    bit retBit;
    StartIIC();                                 /*启动总线*/
    retBit=IICSendByte(slaveAddress);           /*发送器件地址*/
    if(retBit==0)
        return(0);
    retBit=IICSendByte(subAddress);             /*发送器件子地址*/
    if(retBit==0)
        return(0);
    for(i=0;i<num;i++)
    {
        retBit=IICSendByte(*(dataPointer+i));   /*发送数据*/
        if(retBit==0)
            return(0);
    }
    StopIIC();                                  /*结束总线*/
    return(0);
}
/*****************************************************************/
/*从 24C02 的指定地址读入若干个数据函数*/
bit IICRead(uchar slaveAddress,uchar subAddress,uchar *dataPointer,uchar num)
{
    bit retBit;
    StartIIC();                                 /*启动总线*/
    retBit=IICSendByte(slaveAddress);           /*发送器件从地址*/
    if(retBit==0)
        return(0);
    retBit=IICSendByte(subAddress);             /*发送器件子地址*/
    if(retBit==0)
        return(0);
    StopIIC();                                  /*结束总线*/
    StartIIC();                                 /*重新启动总线*/
    retBit=IICSendByte(slaveAddress+1);
    if(retBit==0)
```

```
            return(0);
        for(i=0;i<num-1;i++)
        {
            *dataPointer=IICRcvByte();                  /*接收数据*/
            IICAck(0);                                  /*发送应答位*/
            dataPointer++;
        }
        *dataPointer=IICRcvByte();                      /*接收数据*/
        IICAck(1);
        StopIIC();                                      /*结束总线*/
        return(1);
}
/*************************************************************************/
void Timer0Int() interrupt 1 using 2                    /*每隔 500us 中断扫描一次数码管*/
{
        LEDCycle500usFlag=1;                            /*置 500us 标志位*/
        ScanSeg7();
        if(Int0Flag==1)                                 /*如果需要产生音乐则按音频改变 Timer0 的中断周期*/
{
            P3_7=~P3_7;                                 /*每次中断翻转 P3.7 的电平以产生相应频率的方波*/
TH0=TH0Global;
TL0=TL0Global;
}
else                                                    /*如不需产生音乐则设置 Timer0 的中断周期为 500us*/
{
TH0=(65536-SCAN_CYCLE)/256;
TL0=(65536-SCAN_CYCLE)%256;
}
}
/*************************************************************************/
void Timer1Int() interrupt 3 using 3                    /*Timer1 中断服务程序，中断周期为 10ms*/
{
        uchar ucharTemp,ucharTemp1;
        static uchar s_timeBaseCounter;
        static uchar s_paiCounter,s_secCounter,s_paiNum;
        temp=65536-10000;                               /*中断周期为 10ms*/
        TH1=temp>>8;                                    /*temp/256 相当于右移 8 位*/
        TL1=temp-((temp>>8)<<8);                        /*temp%256*/
if(++s_timeBaseCounter>=10)                             /*100ms 到了则设置 100ms 标志位*/
{
```

```
s_timeBaseCounter=0;
LEDCycle100msFlag=1;
}
LEDCycle10msFlag=1;                                    /*设置 10ms 标志位*/

    if(++s_paiCounter>s_paiNum)                        /*音乐一拍的时间到了*/
    {
        s_paiCounter=0;
        switch(s_musicCounter)                         /*根据当前音乐编号查找下一个音乐标识码*/
{
                case 0x01: ucharTemp=c_music1[s_tableCounter++];break;
                case 0x02: ucharTemp=c_music2[s_tableCounter++];break;
                case 0x03: ucharTemp=c_music3[s_tableCounter++];break;
                case 0x04: ucharTemp=c_music4[s_tableCounter++];break;
                case 0x0a: ucharTemp=c_music10[s_tableCounter++];break;
}
if(ucharTemp==0x00)                                    /*如果查到 0x00 则表示本首歌曲结束*/
{
                Int0Flag=0;                            /*产生音乐频率标志置零*/
                temp=65536-SCAN_CYCLE;
                TH0temp=temp>>8;                       /*temp/256 相当于右移 8 位*/
                TL0temp=temp-((temp>>8)<<8);           /*temp%256*/
            TH0=TH0temp;
            TL0=TL0temp;
                s_paiNum=4*PAI_BASE;                   /*本首歌曲结束后停顿*/
                                                       /*4*PAI_BASE*10ms=600ms 再继续播放*/
                P3_7=1;                                /*把喇叭关掉*/
                s_tableCounter=0;
                if(musicAutoFlag==1)                   /*如自动循环播放功能打开则跳到下一首歌曲*/
                {
                    if(++s_musicCounter>MUSIC_NUM)
                    {
                        s_musicCounter=0x01;
                    }
                }
}
else                                                   /*如果查到非 0x00 则表示是音乐标识码*/
{
                s_paiNum=(ucharTemp&0x0f)*PAI_BASE;    /*标识码的低四位为节拍码*/
                ucharTemp1=(ucharTemp&0xf0)>>4;        /*标识码的高四位为音频码*/
```

```
                    if(ucharTemp1==0)                    /*如果音频码为零则表示不发音*/
                    {
                        Int0Flag=0;
                        temp=65536-SCAN_CYCLE;
                        TH0temp=temp>>8;                 /*temp/256 相当于右移 8 位*/
                        TL0temp=temp-((temp>>8)<<8);     /*temp%256*/
                TH0=TH0temp;
                TL0=TL0temp;
                    }
                    else                                 /*如果音频码不为零则按此改变 Timer0 的中断周期*/
                    {
                        temp=c_musicFreq[ucharTemp1-1];
                        TH0temp=temp>>8;                 /*temp/256 相当于右移 8 位*/
                        TL0temp=temp-((temp>>8)<<8);     /*temp%256*/
                TH0=TH0temp;
                TL0=TL0temp;
                        TH0Global=TH0;                   /*保存新的代表音频的 Timer0 周期值*/
                        TL0Global=TL0;
                        Int0Flag=1;
}
}
        }

        if(++s_secCounter>=100)                          /*如果 Timer1 中断 100 次则表示 1s 到了*/
{
s_secCounter=0;
            sec++;                                       /*秒加一*/
            modifyTimeFlag=1;
        }
        TF1=0;
}
/*************************************************************************/
void UARTInt() interrupt 4 using 1                       /*串口中断服务程序*/
{
        if(TI==1)                                        /*是发送中断则返回*/
{
TI=0;
return;
}
else                                                     /*接收到数据*/
```

```
{
uchar rcv;
        while(RI!=1);
{
RI=0;                                                    /*请接收标志位以激活下次串口中断*/
                    rcv=SBUF;
                    switch(rcv)
  {
  case 48: UARTKeyFlag=0x0d;      /*收到'0'*/
                                  break;
  case 49: UARTKeyFlag=0x00;      /*收到'1'*/
                                  break;
  case 50: UARTKeyFlag=0x01;      /*收到'2'*/
                                  break;
  case 51: UARTKeyFlag=0x02;      /*收到'3'*/
                                  break;
                    case 52: UARTKeyFlag=0x04;      /*收到'4'*/
                                  break;
                    case 53: UARTKeyFlag=0x05;      /*收到'5'*/
                                  break;
                    case 54: UARTKeyFlag=0x06;      /*收到'6'*/
                                  break;
                    case 55: UARTKeyFlag=0x08;      /*收到'7'*/
                                  break;
                    case 56: UARTKeyFlag=0x09;      /*收到'8'*/
                                  break;
                    case 57: UARTKeyFlag=0x0a;      /*收到'9'*/
                                  break;
                    case 65: UARTKeyFlag=0x03;      /*收到'A'*/
                                  break;
                    case 66: UARTKeyFlag=0x07;      /*收到'B'*/
                                  break;
                    case 67: UARTKeyFlag=0x0b;      /*收到'C'*/
                                  break;
                    case 68: UARTKeyFlag=0x0f;      /*收到'D'*/
                                  break;
                    case 69: UARTKeyFlag=0x0c;      /*收到'E'*/
                                  break;
                    case 70: UARTKeyFlag=0x0e;      /*收到'F'*/
                                  break;
```

```
                    }
    }
    }
    }
/*********************************************************************/
////////////////////////////////////////////////End of Program/////////////////////////////////////////////////
/*********************************************************************/
```

演示程序中 9 首流行音乐列表

1. 生日快乐歌
2. 两只蝴蝶
3. 老鼠爱大米
4. 梦醒时分
5. 大长今
6. 漫步人生路
7. 最浪漫的事
8. 一千年以后
9. 一辈子的孤单

附光盘中 51 单片机应用实例列表

51 控制硬盘.pdf

89C51 单片机 IO 口模拟串行通信的实现方法.pdf

从 MCS51 向 AVR 的快速转换.pdf

单片机是怎样在液晶上显示字符的.pdf

基于单片机的红外通讯设计.pdf

用 51 控制 1602LCD 液晶屏.pdf

USB2.0 的高速无线数传接收设备的数据接收存储方法.doc

USB 接口芯片 CH375 的原理及应用.doc

USB 电池充电电路.doc

一个单片机串行数据采集.doc

一种基于 AT89C2051 单片机的传呼机发码电路.doc

一种基于公用电话网的家电遥控系统.doc

一种纸币识别系统的设计.doc

关于单片机硬件抗干扰.doc

几个单片机应用实例.doc

单线数字温度传感器的原理与应用.doc

可充电电池技术和充电方法.doc

基于 51 单片机的车用数字仪表设计与实现.doc

基于 77E58 的高速行式热敏打印机控制板的研制.doc

基于 AVR 单片机的 SPWM 变频调速控制策略.doc

基于 GPRS 的嵌入式 Internet 设备.doc

基于打印机接口的语音型抢答器.doc

多网络智能远程遥控系统的设计与实现.doc

射频卡技术在数字式预付费电表系统中的应用.doc

带有串行接口的功率电能计量芯片 CS5460 及其应用.doc

步进电机小知识.doc

混合信号单片机 C8051F060 存储系统的编程.doc

源码公开的 TCPIP 协议栈在远程监测中的应用.doc

用 51 单片机控制 RTL8019AS 实现以太网通讯.doc

用 IP2022 实现支持 GPRS 的 GPS 系统.doc

电子车速里程表的单片机实现方案.doc

硬件设计的鸡毛蒜皮.doc

血压计的原理.doc

采用实时时钟芯片 DS1302.doc

非接触 IC 卡读写器的应用设计.doc

10 种软件滤波方法.doc

93c46 读写程序.doc

125KHz RFID 读写器的 FSK 解调器设计.doc

AT89C51 单片机在无线数据传输中的应用.doc

AT89C52 的智能无线安防报警器.doc

AT89C2051 控制 LMX2332 的频率合成器.doc

AT89C2051 设计的 PCAT 键盘.doc

AT89C2051 驱动步进电机的电路和源码.doc

CD4051 和 AD595 制作的温度采集仪.doc

CRC 算法原理及 C 语言实现.doc

I2C 总线数字电位器原理及与单片机的接口设计.doc

LIN 总线在车身控制中的应用.doc

LIN 网络技术与汽车电子控制.doc

MRFIC1502 在 GPS 接收器中的应用.doc

NE556 等制作的小功率同步电机调速器.doc

NOKIA 移动电话液晶显示模块 LPH7366 原理及其应用.doc

PC 机与单片机的通讯.doc

PC 机打印口与便携式数据采集系统接口设计.doc

RS232 到 485 电路.doc

RS-485 接口芯片介绍及应用中的有关问题.doc

TLC2543 在仪器仪表中的应用.doc

标准配置单

1. 51PCB 板及对应元器件　　一套

2. 232 串口线　　一条

3. USB 电源线　　一条

4. 元器件清单并电路图　　一份

5. 附光盘　　一张

（含程序烧写软件、开发板与 PC 机串口通信软件、开发板资料及相关文档等）

关于我们

公司简介

亿学通™ 电子是纳科斯科技旗下的高校事业部，成立于 2005 年底。专精于高校竞赛创新产品和设备的开发、生产及销售。和全国 1400 余所高校有着广泛的联系和合作。

亿学通™ 坚持“合适即最好”的经营理念，倡导“创新实践、学以致用”的教育理念，以市场需求为导向、以科技创新为依托，为客户研发最有针对性的产品和设备，让更多的同学能在亿学通™ 平台上得到锻炼和能力的提升是公司的服务宗旨。

现在产品种类已发展成“创新教学产品、机器人/智能车、电子实习套件、电子竞赛套件、配套电子模块、传感器模块等六大类近两百种产品。

公司具有专业的研发团队和第一流市场调研人员的。可以保证我们能够为客户提供最合适的产品。

远景规划

让一亿大学生在**亿学通™** 创新平台中受益。

经营理念

合适即最好！Be Appropriate to be Best!

注：为高校创新实践量身定制最合适的实践创新教育产品。

经营策略

易：寓教于乐、轻松掌握；
新：技术新潮、形式新颖；
精：树精品意识、创优秀品牌；
专：以专业的眼光做产品；以专一的态度做事情

联系方式

电话：**010-62669059**
邮箱：**61mcu@61mcu.com**
网址：**http://www.61mcu.com**
地址：北京市海淀区上地七街国际创业园 2 号院 C 座 812

账户信息

中国工商银行对公账户（对公业务窗口办理,汇率:1%,最高不超过 50 元,2-3 个工作日可以到帐）
户名：北京纳克斯科技发展有限公司
开户行：中国工商银行海淀支行营业部
帐号：**0200 0496 0920 0322 217**