应用笔记

AN022—C8051F02X系列带注释“C”例程

相关器件

此应用笔记适用于下列器件：
C8051F020，C8051F021，C8051F022，C8051F023

引言

本应用笔记包含用C语言编写的例程代码是使用C8051F02X系列器件应用软件开发的起点。

按外设划分的程序索引

下面的简要描述提供了附件程序的索引（按外设划分）

ADC0举例

下面是ADC0应用例程

“ADC0_Buf1.c”

此程序为使用ADC0的例程，在中断模式使用定时器3溢出作为开始启动信号并采样AIN0<NUM_SAMPLES>次，将结果存储在XDATA空间。一旦<NUM_SAMPLES>次被采集，采样值从UART0传输。一旦传输结束，另一个数据采样次数<NUM_SAMPLES>将被采集并重复此处理过程。

“ADC0_Int1.c”

此程序为使用ADC0的例程，在中断模式使用定时器3溢出作为开始启动信号测量片内温度传感器的输出。ADC0的转化结果经计算的温度从UART0传输。

“ADC0_Int2m.c”

此程序为使用ADC0的例程，在中断模式使用定时器3溢出作为开始启动信号测量从AIN0到AIN7的电压和温度传感器。ADC0转换结果经计算从UART0传输。

“ADC0_OSA1.c”

此程序为使用ADC0的例程，在中断模式使用定时器3溢出作为开始启动信号测量片内温度传感器的输出。ADC0 结果经过简单的“Integrate-and-dump”处理方法过滤“integrate/decimate”比率由<INT_DEC>给

Silabs Integrated Products, Inc.
4301 Westbank Drive
Suite B-100
Austin, TX 78746
www.silabs.com

新华龙电子有限公司
深圳市福田区华强北路现代之窗大厦 A座 13F C室(518013)
Tel:(0755) 83645240 83645242 Fax: 83645243
电邮:shenzhen@xhl.com.cn
网址www.xhl.com.cn

出。ADC0转换的结果经计算得温度值从UART0传输。

“ADC0_Poll1.c”

此例程说明了ADC0在查询模式的操作。ADC0设置为写AD0BUSY作为其转换开始信号，并测量片内温度传感器的输出。温度传感器的输出转换成摄氏度并从UART0传输。

DAC0举例

下面是DAC0的使用例程。如果有必要很容易将其变成DAC1的例程。

“DAC0_DTMF1.c”

次例程源代码在ADC0输出双音多频音调。根据常数<SAMPLERATED>确定的速率定时更新DAC0的输出，并使用定时器4管理和定时。

振荡器举例

下面是配置内部和外部振荡器的例程。

“OSC_Cry1.c”

此例程说明如何配置外部振荡器驱动22.1184Mhz晶体，及如何选择此外部振荡器作为系统时钟。同时使能丢失时钟检测器复位功能。假定22.1184Mhz晶体连接在XTAL1和XTAL2之间。

“OSC_Int1.c”

此例程说明如何配置内部部振荡器到最大频率（16Mhz）同时使能丢失时钟检测器复位功能。

定时器举例

“Timer0_Poll1.c”

此例程说明在查询模式使用定时器0实现1ms分辨率延时计数器。

程序代码

“ADC0_Buf1.c”

```
//-----------------------------------------------------------------------------
// ADC0_Buf1.c
//-----------------------------------------------------------------------------
// 版权所有Silabs集成产品有限公司 2001
//
// 作者: BW
// 日期: 2001年8月27日
//此程序为使用ADC0的例程，在中断模式使用定时器3溢出作为开始启动信号并采样//AIN0<NUM_SAMPLES>
次，将结果存储在XDATA空间。一旦<NUM_SAMPLES>次被采集，采样值从//UART0传输。一旦传输结束，另
```

```
一个数据采样次数<NUM_SAMPLES>将被采集并重复处理过程。
//
//  假定一个22.1184Mhz晶体连接在XTAL1和XTAL2之间。
//
//用全局常量SYSCLK存储系统时钟频率。用全局常量BAUDRATE存储目标UART波特率。
//用全局常量SAMPLERATE存储ADC0采样速率。每批收集的采样次数存储在<NUM_SAMPLES>。
//在使用C8051F02x器件的4096字节XRAM时，
//采样次数<NUM_SAMPLES>的最大值为2048 （假定未使用外扩RAM接口）。
//
// 目标器件: C8051F02x
// 链接工具: KEIL C51 6.03 / KEIL EVAL C51
//
//-----------------------------------------------------------------------------
// 包含文件
//-----------------------------------------------------------------------------
#include <c8051f020.h> // SFR声明
#include <stdio.h>
//-----------------------------------------------------------------------------
// C8051F02X的16位SFR定义
//-----------------------------------------------------------------------------
sfr16 DP = 0x82; // 数据指针
sfr16 TMR3RL = 0x92; // 定时器3重装值
sfr16 TMR3 = 0x94; // 定时器3计数器
sfr16 ADC0 = 0xbe; // ADC0数据
sfr16 ADC0GT = 0xc4; // ADC0大于窗口
sfr16 ADC0LT = 0xc6; // ADC0小于窗口
sfr16 RCAP2 = 0xca; // 定时器2捕捉/重装
sfr16 T2 = 0xcc; // 定时器2
sfr16 RCAP4 = 0xe4; // 定时器4捕捉/重装
sfr16 T4 = 0xf4; // 定时器4
sfr16 DAC0 = 0xd2; // DAC0数据
sfr16 DAC1 = 0xd5; // DAC1数据
 //-----------------------------------------------------------------------------
// 全局常量
//-----------------------------------------------------------------------------
#define SYSCLK 22118400 // 系统时钟频率（Hz）
#define BAUDRATE 115200 // UART波特率（bps）
#define SAMPLERATE0 50000 // ADC0采样频率（Hz）
#define NUM_SAMPLES 2048 // ADC0采样次数
#define TRUE 1
#define FALSE 0
sbit LED = P1^6; // LED='1' 意为开
sbit SW1 = P3^7; // SW1='0' 意为按压开关
//-----------------------------------------------------------------------------
// 函数原型
//-----------------------------------------------------------------------------
void SYSCLK_Init (void);
void PORT_Init (void);
void UART0_Init (void);
void ADC0_Init (void);
void Timer3_Init (int counts);
```

```
void ADC0_ISR (void);
//-----------------------------------------------------------------------------
// 全局变量
//-----------------------------------------------------------------------------
xdata unsigned samples[NUM_SAMPLES]; // 存储ADC0结果数组
bit ADC0_DONE; // 当NUM_SAMPLES次被采集为真
//-----------------------------------------------------------------------------
// 主程序
//-----------------------------------------------------------------------------
void main (void) {
int i; // 循环计数器
WDTCN = 0xde; // 禁止看门狗定时器
WDTCN = 0xad;
SYSCLK_Init (); // 初始化振荡器
PORT_Init (); // 初始化数据交叉开关和通用IO口
UART0_Init (); // 初始化UART0
Timer3_Init (SYSCLK/SAMPLERATE0); // 初始化定时器3溢出作为ADC0采样率
ADC0_Init (); // 初始化ADC
EA = 1; // 允许全部中断
while (1) {
ADC0_DONE = FALSE;
LED = 1; // 在采样过程中点亮LED
EIE2 |= 0x02; // 允许ADC0中断
while (ADC0_DONE == FALSE); // 等待采样结果
// 上传采样值到UART0
LED = 0; // 上传其间关LED（灭）
for (i = 0; i < NUM_SAMPLES; i++) {
printf (“%u\n”, samples[i]);
}
printf (“\n”);
}
}
//-----------------------------------------------------------------------------
// 初始化化子程序
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
// 时钟初始化
//-----------------------------------------------------------------------------
//
// 此程序初始化系统时钟使用22.1184MHz晶体为时钟源
//
void SYSCLK_Init (void)
{
int i; // 延时计数器
OSCXCN = 0x67; // 开启外部振荡器（22.1184MHz晶体）
for (i=0; i < 256; i++) ; // 等待振荡器启振
while (!(OSCXCN & 0x80)) ; // 等待晶体振荡器稳定
OSCICN = 0x88; // 选择外部振荡器为系统时钟源并允许丢失时钟检测器
}
//-----------------------------------------------------------------------------
// IO口初始化
```

```
//-----------------------------------------------------------------------
//
// 配置数据交叉开关和通用IO口
//
void PORT_Init (void)
{
XBR0 = 0x04; // 使能UART0
XBR1 = 0x00;
XBR2 = 0x40; // 使能数据交叉开关和弱上拉
P0MDOUT |= 0x01; // 允许TX0为推挽输出
P1MDOUT |= 0x40; // 允许P1.6(LED)为推挽输出
}
//-----------------------------------------------------------------------
// UART0初始化
//-----------------------------------------------------------------------
//
// 配置UART0，使用定时器1为波特率发生器
//
void UART0_Init (void)
{
SCON0 = 0x50; // SCON0: 模式1, 8位UART, 使能RX
TMOD = 0x20; // TMOD: 定时器1, 模式2, 8位重装
TH1 = -(SYSCLK/BAUDRATE/16); // 根据波特率的值设定定时器1重装值
TR1 = 1; // 启动定时器1
CKCON |= 0x10; // 定时器1使用系统时钟作为时基
PCON |= 0x80; // SMOD00 = 1
TI0 = 1; // 表示TX0就绪
}
//-----------------------------------------------------------------------
// ADC0初始化
//-----------------------------------------------------------------------
//
// 配置ADC0，使用定时器3溢出作为转换开始信号,转换结束产生一个中断,
// 使用左对齐输出模式，
// 使能ADC转换结束中断中断。使能ADC0，但禁止ADC0转换结束中断。
//
void ADC0_Init (void)
{
ADC0CN = 0x05; // ADC0 禁止; 正常跟踪模式，定时器3溢出ADC0转换开始
//  ADC0数据左对齐
REF0CN = 0x07; // 使能温度传感器, 片内 VREF,和 VREF 输出缓冲器
AMX0SL = 0x00; // 选择AIN0作为ADC多路转换输出
ADC0CF = (SYSCLK/2500000) << 3; // ADC转换时钟2.5MHz
ADC0CF &= ~0x07; // PGA增益 = 1
EIE2 &= ~0x02; // 禁止ADC0中断
AD0EN = 1; // 使能ADC0
}
//-----------------------------------------------------------------------
// 定时器3初始化
//-----------------------------------------------------------------------
//
```

```
// 配置定时器3，自动重装，间隔由 <counts> 决定(不产生中断)，使用系统时钟为时基
//
void Timer3_Init (int counts)
{
TMR3CN = 0x02; // 停止定时器3; 清除TF3;
// 使用系统时钟作为时基
TMR3RL = -counts; // 初始化重装值
TMR3 = 0xffff; // 立即开始重装
EIE2 &= ~0x01; // 禁止定时器3中断
TMR3CN |= 0x04; // 启动定时器3
}
//-----------------------------------------------------------------------------
//中断服务程序
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
// ADC0中断服务程序
//-----------------------------------------------------------------------------
//
// ADC0转换结束中断服务程序
// 得到ADC0采样值并存储到全局数组<samples[]>,
// 并更新局部采样计数器 <num_samples>。 当 <num_samples> == <NUM_SAMPLES>时,
// 禁止ADC0 转换结束中断并置ADC0_DONE = 1。
//
void ADC0_ISR (void) interrupt 15 using 3
{
static unsigned num_samples = 0; // ADC0采样计数器
AD0INT = 0; // 清除ADC0转换结束标志
samples[num_samples] = ADC0; // 读和存储ADC0值
num_samples++; // 更新采样计数器
if (num_samples == NUM_SAMPLES) {
num_samples = 0; // 复位采样计数器
EIE2 &= ~0x02; // 禁止ADC0中断
ADC0_DONE = 1; // 设置DONE标志
}
}
```

“ADC0_Int1.c”

```
//-----------------------------------------------------------------------------
// ADC0_int1.c
//-----------------------------------------------------------------------------
// 版权所有Silabs集成产品有限公司 2001
//
// 作者: BW
// 日期: 2001年8月18日
//
// 此程序为ADC0的应用例程，在中断模式使用定时器3溢出
// 作为开始转换信号来测量片内温度传感器。
// ADC0的结果经过计算得到温度值从UART0传输。
//
// 假设在XTAL1和XTAL2之间连接一个22.1184MHz晶体。
//
// 系统时钟频率存储在全局常量SYSCLK，目标UART波特率存储在全局常量BAUDRATE。
```

```
// ADC0采样率存储在全局常量SAMPLERATE0。
//
// 目标器件: C8051F02x
// 链接工具： KEIL C51 6.03 / KEIL EVAL C51
//
//-----------------------------------------------------------------------------
// 包含文件
//-----------------------------------------------------------------------------
#include <c8051f020.h> // SFR声明
#include <stdio.h>
//-----------------------------------------------------------------------------
// C8051F02X的16位SFR定义
//-----------------------------------------------------------------------------
sfr16 DP = 0x82; // 数据指针
sfr16 TMR3RL = 0x92; // 定时器3重装值
sfr16 TMR3 = 0x94; // 定时器3计数器
sfr16 ADC0 = 0xbe; // ADC0数据
sfr16 ADC0GT = 0xc4; // ADC0大于窗口
sfr16 ADC0LT = 0xc6; // ADC0小于窗口
sfr16 RCAP2 = 0xca; // 定时器2捕捉/重装
sfr16 T2 = 0xcc; // 定时器2
sfr16 RCAP4 = 0xe4; // 定时器4捕捉/重装
sfr16 T4 = 0xf4; // 定时器4
sfr16 DAC0 = 0xd2; // DAC0数据
sfr16 DAC1 = 0xd5; // DAC1数据
//-----------------------------------------------------------------------------
// 全局常量
//-----------------------------------------------------------------------------
#define SYSCLK 22118400 // 系统时钟频率（Hz）
#define BAUDRATE 9600 // UART波特率（bps）
#define SAMPLERATE0 50000 // ADC0采样频率（Hz）
sbit LED = P1^6; // LED='1' 意为ON
sbit SW1 = P3^7; // SW1='0' 意为按下开关
//-----------------------------------------------------------------------------
// 函数原型
//-----------------------------------------------------------------------------
void SYSCLK_Init (void);
void PORT_Init (void);
void UART0_Init (void);
void ADC0_Init (void);
void Timer3_Init (int counts);
void ADC0_ISR (void);
//-----------------------------------------------------------------------------
// 全局变炼
//-----------------------------------------------------------------------------
long result; // ADC0 十选一值
//-----------------------------------------------------------------------------
// 主程序
//-----------------------------------------------------------------------------
void main (void) {
long temperature; // 温度为百分之一精度（度C）
```

```
int temp_int, temp_frac; // 温度的整数和小数部分
WDTCN = 0xde; // 禁止看门狗定时器
WDTCN = 0xad;
SYSCLK_Init (); // 初始化振荡器
PORT_Init (); // 初始化数据交叉开关和通用IO口
UART0_Init (); // 初始化UART0
Timer3_Init (SYSCLK/SAMPLERATE0); // 初始化定时器3溢出作为采样速率
ADC0_Init (); // 初始化ADC
AD0EN = 1; // 使能ADC
EA = 1; // 允许全局中断
while (1) {
EA = 0; // 禁止中断
temperature = result; // 从全局变量得到ADC值
EA = 1; // 重新允许中断
// 计算温度误差为百分之一度
temperature = temperature - 41380;
temperature = (temperature * 100L) / 156;
temp_int = temperature / 100;
temp_frac = temperature - (temp_int * 100);
printf (“Temperature is %+02d.%02d\n”, temp_int, temp_frac);
}
}
//-----------------------------------------------------------------------------
// 初始化子程序
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
// 系统时钟初始化
//-----------------------------------------------------------------------------
//
// 此程序初始化系统时钟，使用22.1184MHz晶体作为时钟源
//
void SYSCLK_Init (void)
{
int i; // 延时计数器
OSCXCN = 0x67; // 起动外部振荡器晶体为 22.1184MHz
for (i=0; i < 256; i++) ; // 等待振荡器启动(>1ms)
while (!(OSCXCN & 0x80)) ; // 等待晶体振荡器稳定
OSCICN = 0x88; // 选择外部振荡器作为系统时钟源并使能丢失时钟检测器
}
//-----------------------------------------------------------------------------
// IO口初始化
//-----------------------------------------------------------------------------
//
// 配置数据交叉开关和通用IO口
//
void PORT_Init (void)
{
XBR0 = 0x04; // 使能UART0
XBR1 = 0x00;
XBR2 = 0x40; // 允许数据交叉开关和弱上拉
P0MDOUT |= 0x01; // 允许TX0为推挽输出
```

```
P1MDOUT |= 0x40; // 允许P1.6(LED)为推挽输出
}
//-------------------------------------------------------------------------
// UART0初始化
//-------------------------------------------------------------------------
//
//设置UART0, 用定时器1为波特率发生器
//
void UART0_Init (void)
{
SCON0 = 0x50; // SCON0: 模式1, 8位UART, 允许RX
TMOD = 0x20; // TMOD: 定时器1, 模式2, 8位重装
TH1 = -(SYSCLK/BAUDRATE/16); // 按波特率设置定时器1重装值
TR1 = 1; // 起动定时器1
CKCON |= 0x10; // 定时器1使用系统时钟为时基
PCON |= 0x80; // SMOD00 = 1
TI0 = 1; // 表示TX0就绪
}
//-------------------------------------------------------------------------
// ADC0初始化
//-------------------------------------------------------------------------
//
// 配置ADC0，使用定时器3溢出作为转换源,转换结束产生中断
// 使用左对齐输出模式，使能ADC转换结束中断。禁止ADC
//
void ADC0_Init (void)
{
ADC0CN = 0x05; // 禁止ADC0; 正常跟踪模式
// 定时器3溢出ADC0开始转换，ADC0数据左对齐
REF0CN = 0x07; // 使能温度传感器, 片内 VREF和VREF 输出缓冲器
AMX0SL = 0x0f; // 选择温度传感器作为ADC为多路模拟输出
ADC0CF = (SYSCLK/2500000) << 3; // ADC 转换时钟=2.5MHz
ADC0CF |= 0x01; // PGA增益=2
EIE2 |= 0x02; // 允许ADC中断
}
//-------------------------------------------------------------------------
// 定时器3初始化
//-------------------------------------------------------------------------
//
// 配置定时器3为自动重装，时间间隔由<counts> 制定(不产生中断)
// 使用系统时钟作为时基.
//
void Timer3_Init (int counts)
{
TMR3CN = 0x02; // 停止定时器3; 清除TF3;
// 使用系统时钟作为时基
TMR3RL = -counts; // 初始化重装值
TMR3 = 0xffff; // 设置立即重装值
EIE2 &= ~0x01; // 禁止定时器3中断
TMR3CN |= 0x04; // 起动定时器3
}
```

```
//-----------------------------------------------------------------------------
// 中断服务程序
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
// ADC0中断服务程序
//-----------------------------------------------------------------------------
//
// ADC0转换结束中断服务程序
// 我们将ADC0取样结果存储在全局变量<result>.
//
void ADC0_ISR (void) interrupt 15
{
AD0INT = 0; // ADC转换结束清除标志
result = ADC0; // 读ADC值
}
```

“ADC0_Int2m.c”

```
//-----------------------------------------------------------------------------
// ADC0_int2m.c
//-----------------------------------------------------------------------------
// 版权所有Silabs集成产品有限公司 2001
//
// 作者: BW
// 日期: 2001年8月25日
//
// 此程序为ADC0的应用例程，在中断模式使用定时器3溢出作为开始转换信号
// 测量AIN0到AIN7的电压和温度传感器。
// 转换结果经过计算所得电压从UART0传输
//
// 假设在XTAL1和XTAL2之间接22.1184MHz晶体
//
// 系统时钟频率存储在全局常量SYSCLK. 目标UART波特率存储在全局常量BAUDRATE.
// ADC0采样率存储在全局常量SAMPLERATE0.电压参考值存储在VREF0，
//
// 目标器件: C8051F02x
// 链接工具: KEIL C51 6.03 / KEIL EVAL C51
//
//-----------------------------------------------------------------------------
// 包含文件
//-----------------------------------------------------------------------------
#include <c8051f020.h> // SFR声明
#include <stdio.h>
//-----------------------------------------------------------------------------
// C8051F02X的16位SFR定义
//-----------------------------------------------------------------------------
sfr16 DP = 0x82; // 数据指针
sfr16 TMR3RL = 0x92; // 定时器3重装值
sfr16 TMR3 = 0x94; // 定时器3计数器
sfr16 ADC0 = 0xbe; // ADC0数据
sfr16 ADC0GT = 0xc4; // ADC0大于窗口
sfr16 ADC0LT = 0xc6; // ADC0小于窗口
sfr16 RCAP2 = 0xca; // 定时器2捕捉/重装
```

```
sfr16 T2 = 0xcc; // 定时器2
sfr16 RCAP4 = 0xe4; // 定时器4捕捉/重装
sfr16 T4 = 0xf4; // 定时器4
sfr16 DAC0 = 0xd2; // DAC0数据
sfr16 DAC1 = 0xd5; // DAC1数据
//-----------------------------------------------------------------------------
// 全局常量
//-----------------------------------------------------------------------------
#define SYSCLK 22118400 // 系统时钟频率（Hz）
#define BAUDRATE 9600 // UART波特率（bps）
#define SAMPLERATE0 50000 // ADC0采样频率（Hz）
#define VREF0 2430 // VREF 参考电平为mV
sbit LED = P1^6; // LED=’1’ 意为ON
sbit SW1 = P3^7; // SW1=’0’ 意为按下开关
//-----------------------------------------------------------------------------
// 函数原型
//-----------------------------------------------------------------------------
void SYSCLK_Init (void);
void PORT_Init (void);
void UART0_Init (void);
void ADC0_Init (void);
void Timer3_Init (int counts);
void ADC0_ISR (void);
//-----------------------------------------------------------------------------
// 全局变量
//-----------------------------------------------------------------------------
long result[9]; // AIN0-7和温度传感器输出结果
//-----------------------------------------------------------------------------
// 主程序
//-----------------------------------------------------------------------------
void main (void) {
long voltage; // 电压以mV为单位
int i; // 循环计数器
WDTCN = 0xde; // 禁止看门狗定时器
WDTCN = 0xad;
SYSCLK_Init (); // 初始化振荡器
PORT_Init (); // 初始化数据交叉开关和通用IO
UART0_Init (); // 初始化UART0
Timer3_Init (SYSCLK/SAMPLERATE0); // 初始化定时器3溢出作为采样率
ADC0_Init (); // 初始化ADC
AD0EN = 1; // 允许ADC
EA = 1; // 允许所有中断
while (1) {
for (i = 0; i < 9; i++) {
EA = 0; // 禁止中断
voltage = result[i]; // 从全局变量取得ADC值
EA = 1; // 重新使能中断
// 计算电压（mV）
voltage = voltage * VREF0;
voltage = voltage >> 16;
printf (“Channel ‘%d’ voltage is %ldmV\n”, i, voltage);
```

```
}
}
}
//-----------------------------------------------------------------------------
// 子程序初始化
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
// 系统时钟初始化
//-----------------------------------------------------------------------------
//
// 此程序初始化系统时钟，使用22.1184MHz晶体作为系统时钟
//
void SYSCLK_Init (void)
{
int i; // 延时计数器
OSCXCN = 0x67; // 起动外部振荡器（22.1184MHz晶体）
for (i=0; i < 256; i++) ; // 等待振荡器启动 (>1ms)
while (!(OSCXCN & 0x80)) ; // 等待晶体振荡器稳定
OSCICN = 0x88; // 选择外部振荡器作为系统时钟源并允许丢失时钟检测器
}
//-----------------------------------------------------------------------------
// IO口初始化
//-----------------------------------------------------------------------------
//
// 配置数据交叉开关和通用IO口
//
void PORT_Init (void)
{
XBR0 = 0x04; // 使能UART0
XBR1 = 0x00;
XBR2 = 0x40; // 使能数据交叉开关和弱上拉
P0MDOUT |= 0x01; // 允许TX0为推挽输出
P1MDOUT |= 0x40; // 允许P1.6(LED)为推挽输出
}
//-----------------------------------------------------------------------------
// UART0初始化
//-----------------------------------------------------------------------------
//
// 配置UART0，使用定时1为波特率发生器。
//
void UART0_Init (void)
{
SCON0 = 0x50; // SCON0: 模式1, 8位UART, 允许RX
TMOD = 0x20; // TMOD: 定时器1, 模式2, 8位重装
TH1 = -(SYSCLK/BAUDRATE/16); // 按波特率设置定时器1重装值
TR1 = 1; // 起动定时器1
CKCON |= 0x10; // 定时器1使用系统时钟为时基
PCON |= 0x80; // SMOD00 = 1
TI0 = 1; // 表示TX0就绪
//-----------------------------------------------------------------------------
// ADC0初始化
```

```
//-----------------------------------------------------------------------
//
// 配置ADC0，使用定时器3 溢出作为转换源，转换结束产生中断，使用左对齐输出模式
// 使能ADC转换结束中断。禁止ADC 。
//注意：使能低功率跟踪模式，保证当改变通到时的跟踪次数最少
//
void ADC0_Init (void)
{
ADC0CN = 0x45; // ADC0 禁止；低功率跟踪模式
// 当定时器3溢出时ADC0转换开始；ADC0 数据左对齐
REF0CN = 0x07; // 使能温度传感器，片内VREF和VREF输出缓冲器
AMX0SL = 0x00; // 选择AIN0为ADC多路模拟输出
ADC0CF = (SYSCLK/2500000) << 3; // ADC转化时钟=2.5MHz
ADC0CF &= ~0x07; // PGA增益=1
EIE2 |= 0x02; // 允许ADC中断
}
//-----------------------------------------------------------------------
// 定时器3初始化
//-----------------------------------------------------------------------
//
// 配置定时器3，自动重装时间间隔由<counts>制定(不产生中断)
// 使用系统时钟作为时基.
//
void Timer3_Init (int counts)
{
TMR3CN = 0x02; // 停止定时器3；清除TF3;
// 使用系统时钟作为时基
TMR3RL = -counts; //初始化重装值
TMR3 = 0xffff; // 设置为立即重装
EIE2 &= ~0x01; // 禁止定时器3中断
TMR3CN |= 0x04; // 起动定时器3
}
//-----------------------------------------------------------------------
// 中断服务程序
//-----------------------------------------------------------------------
//-----------------------------------------------------------------------
// ADC0中断服务程序
//-----------------------------------------------------------------------
//
// ADC0转换结束中断服务程序
// 得当ADC0采样值并存储在全局数组 <result>.
// 同时选择下一个通道转换。
//
void ADC0_ISR (void) interrupt 15
{
static unsigned char channel = 0; // ADC多路模拟通道(0-8)
AD0INT = 0; // 清除ADC转换结束标志
result[channel] = ADC0; // 读ADC值
channel++; // 改变通道
if (channel == 9) {
channel = 0;
```

```
}
AMX0SL = channel; // 设置多路模拟转换器到下一个通道
}
```

“ADC0_OSA1.c”

```
//-----------------------------------------------------------------------------
// ADC0_OSA1.c
//-----------------------------------------------------------------------------
// 版权所有Silabs集成产品有限公司 2001
//
// 作者: BW
// 日期: 2001年8月18日
//
// 此程序是ADC0应用例程，在中断模式使用定时器3溢出作为转换开始信号，
// 测量片内温度传感器输出。
// ADC0结果经简单的“integrate-and-dump” 处理，
//“integrate/decimate”率由常量INT_DEC给出
// ADC结果经计算得出温度从UART0传输
//
// 假设在XTAL1和XTAL2之连接22.1184MHz晶体
//
// 系统时钟频率存储在全局常量SYSCLK。目标UART波特率存储在全局常量BAUDRATE.
// ADC0采样率存储在全局常量SAMPLERATE0.
//
// 目标器件: C8051F02x
// 链接工具: KEIL C51 6.03 / KEIL EVAL C51
//-----------------------------------------------------------------------------
// 包含文件
//-----------------------------------------------------------------------------
#include <c8051f020.h> // SFR声明
#include <stdio.h>
//-----------------------------------------------------------------------------
// C8051F02X的16位SFR定义
//-----------------------------------------------------------------------------
sfr16 DP = 0x82; // 数据指针
sfr16 TMR3RL = 0x92; // 定时器3重装值
sfr16 TMR3 = 0x94; // 定时器3计数器
sfr16 ADC0 = 0xbe; // ADC0数据
sfr16 ADC0GT = 0xc4; // ADC0大于窗口
sfr16 ADC0LT = 0xc6; // ADC0小于窗口w
sfr16 RCAP2 = 0xca; // 定时器2捕捉/重装
sfr16 T2 = 0xcc; // 定时器2
sfr16 RCAP4 = 0xe4; // 定时器4捕捉/重装
sfr16 T4 = 0xf4; // 定时器4
sfr16 DAC0 = 0xd2; // DAC0数据
sfr16 DAC1 = 0xd5; // DAC1数据
//-----------------------------------------------------------------------------
// 全局常量
//-----------------------------------------------------------------------------
#define SYSCLK 22118400 // 系统时钟频率（Hz）
#define BAUDRATE 9600 // UART波特率（bps）
#define SAMPLERATE0 50000 // ADC0 采样频率（Hz）
```

```
#define INT_DEC 256 // 累计和十取一比率
sbit LED = P1^6; // LED=’1’ 意为ON
sbit SW1 = P3^7; // SW1=’0’ 意为按下开关
//-----------------------------------------------------------------------------
// 函数原型
//-----------------------------------------------------------------------------
void SYSCLK_Init (void);
void PORT_Init (void);
void UART0_Init (void);
void ADC0_Init (void);
void Timer3_Init (int counts);
void ADC0_ISR (void);
//-----------------------------------------------------------------------------
// 全局变量
//-----------------------------------------------------------------------------
long result; // ADC0十取一值
//-----------------------------------------------------------------------------
// 主程序
//-----------------------------------------------------------------------------
void main (void) {
long temperature; // 精度为百分之一的温度 C
int temp_int, temp_frac; // 温度的整数和小数部分
WDTCN = 0xde; // 禁止看门狗定时器
WDTCN = 0xad;
SYSCLK_Init (); // 初始化振荡器
PORT_Init (); // 初始化数据交叉开关和通用IO口
UART0_Init (); // 初始化UART0
Timer3_Init (SYSCLK/SAMPLERATE0); // 初始化定时器3溢出为采样速率
ADC0_Init (); // 初始化ADC
AD0EN = 1; // 使能ADC
EA = 1; // 使能所有中断
while (1) {
EA = 0; // 禁止中断
temperature = result;
EA = 1; // 重使能中断
// 计算温度（百分之一精度）
temperature = temperature - 41380;
temperature = (temperature * 100L) / 156;
temp_int = temperature / 100;
temp_frac = temperature - (temp_int * 100);
printf (“Temperature is %+02d.%02d\n”, temp_int, temp_frac);
LED = ~SW1; // LED 反映开关状态
}
}
//-----------------------------------------------------------------------------
// 子程序初始化
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
// 系统时钟初始化
//-----------------------------------------------------------------------------
//
```

```
// 此程序初始化系统时钟，使用22.1184MHz晶体作为系统时钟源
//
void SYSCLK_Init (void)
{
int i; // 延时计数器
OSCXCN = 0x67; // 起动外部振荡器（22.1184MHz晶体）
for (i=0; i < 256; i++) ; // 等待振荡器启动(>1ms)
while (!(OSCXCN & 0x80)) ; // 等待晶体振荡器稳定
OSCICN = 0x88; // 选择外部振荡器作为系统时钟源并允许丢失时钟检测器
}
//-----------------------------------------------------------------------------
// IO口初始化
//-----------------------------------------------------------------------------
//
// 配置数据交叉开关和通用IO口
//
void PORT_Init (void)
{
XBR0 = 0x04; // 使能UART0
XBR1 = 0x00;
XBR2 = 0x40; // 使能数据交叉开关和弱上拉
P0MDOUT |= 0x01; // 使能TX0推挽输出
P1MDOUT |= 0x40; // 使能P1.6(LED)为推挽输出
}
//-----------------------------------------------------------------------------
// UART0初始化
//-----------------------------------------------------------------------------
//
// 配置UART0，使用定时器1作为波特率发生器
//
void UART0_Init (void)
{
SCON0 = 0x50; // SCON0: 模式1, 8位UART, 使能RX
TMOD = 0x20; // TMOD: 定时器1, 模式2, 8位重装
TH1 = -(SYSCLK/BAUDRATE/16); // 按波特率设置T1重装值
TR1 = 1; // 起动定时器1
CKCON |= 0x10; // 定时器1使用系统时钟作为时基
PCON |= 0x80; // SMOD00=1
TI0 = 1; // 表示TX0就绪
}
//-----------------------------------------------------------------------------
// ADC0初始化
//-----------------------------------------------------------------------------
//
// 配置ADC0，使用定时器3溢出作为转换源，转换结束产生中断，
// 使用左对齐输出模式，允许ADC转换结束中断。不使用时禁止ADC
//
void ADC0_Init (void)
{
ADC0CN = 0x05; // ADC0禁止；正常跟踪
// mode；定时器3溢出ADC0转换开始，ADC0数据是左对齐
```

```
REF0CN = 0x07; // 允许温度传感器，片内VREF和VREF输出缓冲器
AMX0SL = 0x0f; // 选择温度传感器作为ADC多路模拟转换输出
ADC0CF = (SYSCLK/2500000) << 3; // ADC转换时钟=2.5MHz
ADC0CF |= 0x01; // PGA增益=2
EIE2 |= 0x02; // 允许ADC中断
}
//-------------------------------------------------------------------------
// 定时器3初始化
//-------------------------------------------------------------------------
//
// 配置定时器3,自动重装间隔由<counts>指定 (不产生中断)使用系统时钟为时基
//
void Timer3_Init (int counts)
{
TMR3CN = 0x02; // 停止定时器3; 清除 TF3;
// 使用系统时钟为时基
TMR3RL = -counts; // 初始化重装值
TMR3 = 0xffff; // 设置为立即重装
EIE2 &= ~0x01; // 禁止定时器3中断
TMR3CN |= 0x04; // 起动定时器3
}
//-------------------------------------------------------------------------
// 中断服务程序
//-------------------------------------------------------------------------
//-------------------------------------------------------------------------
// ADC0中断服务程序
//-------------------------------------------------------------------------
//
// ADC0转换结束中断服务程序
// 得到ADC0采样值, 将它加到运行总数<accumulator>中,
// 局部十取一计数器 <int_dec>减一。当<int_dec>为0时, 在全局变量<result>放置十取一的// 结果。
//
void ADC0_ISR (void) interrupt 15
{
static unsigned int_dec=INT_DEC; // 合计/十取一计数器,
// 当int_dec = 0时记入新值

static long accumulator=0L;
AD0INT = 0; // 请除ADC转换结束标志
accumulator += ADC0; // 读ADC值并加到运行总数中
int_dec--; // 更新十取一计数器
if (int_dec == 0) { // 如果为0记入结果
int_dec = INT_DEC; // 复位计数器
result = accumulator >> 8;
accumulator = 0L; // 复位accumulator
}
}
```

“ADC0_Poll1.c”

```
//-------------------------------------------------------------------------
// ADC0_Poll1.c
//-------------------------------------------------------------------------
```

```
// 版权所有Silabs集成产品有限公司 2001
//
// 作者: BW
// 日期: 2001年8月18
//
// 此程序示范了ADC0的查询操作模式。ADC0配置为写AD0BUSY作为转换开始源
// 测量片内温度传感器。温度传感器的输出转换成摄氏度由UART0传输
// 假设在XTAL1和XTAL2之间连接22.1184MHz晶体。
//
// 系统时钟频率存储在全局常量SYSCLK。目标器件UART波特率存储在全局常量BAUDRATE。
//
// 目标器件: C8051F02x
// 链接工具: KEIL C51 6.03 / KEIL EVAL C51
//
//-----------------------------------------------------------------------------
// 包含文件
//-----------------------------------------------------------------------------
#include <c8051f020.h> // SFR 声明
#include <stdio.h>
//-----------------------------------------------------------------------------
// C8051F02X的16位SFR定义
//-----------------------------------------------------------------------------
sfr16 DP = 0x82; // 数据指针
sfr16 TMR3RL = 0x92; // 定时器3重装值
sfr16 TMR3 = 0x94; // 定时器3计数器
sfr16 ADC0 = 0xbe; // ADC0数据
sfr16 ADC0GT = 0xc4; // ADC0大于窗口
sfr16 ADC0LT = 0xc6; // ADC0小于窗口
sfr16 RCAP2 = 0xca; // 定时器2捕捉/重装
sfr16 T2 = 0xcc; // 定时器2
sfr16 RCAP4 = 0xe4; // 定时器4捕捉/重装
sfr16 T4 = 0xf4; // 定时器4
sfr16 DAC0 = 0xd2; // DAC0数据
sfr16 DAC1 = 0xd5; // DAC1数据
//-----------------------------------------------------------------------------
//全局常量
//-----------------------------------------------------------------------------
#define SYSCLK 22118400 //系统时钟频率（Hz）
#define BAUDRATE 9600 // UART波特率（bps）
sbit LED = P1^6; // LED='1' 意为ON
sbit SW1 = P3^7; // SW1='0' 意为按下开关
//-----------------------------------------------------------------------------
// 函数原型
//-----------------------------------------------------------------------------
void SYSCLK_Init (void);
void PORT_Init (void);
void UART0_Init (void);
void ADC0_Init (void);
//-----------------------------------------------------------------------------
// 全局变量
//-----------------------------------------------------------------------------
```

```
//-----------------------------------------------------------------------------
// 主程序
//-----------------------------------------------------------------------------
void main (void) {
long temperature; // 温度（百分之一精度）度C
int temp_int, temp_frac; // 温度的整数和小数部分
WDTCN = 0xde; // 禁止看门狗定时器
WDTCN = 0xad;
SYSCLK_Init (); // 初始化振荡器
PORT_Init (); // 初始化数据交叉开关和通用IO口
UART0_Init (); // 初始化UART0
ADC0_Init (); // 初始化和使能ADC
while (1) {
AD0INT = 0; //清除转换结束标记
AD0BUSY = 1; // 开始转换
while (AD0INT == 0); // 等待转换结束
temperature = ADC0; // 读ADC0数据
// 计算温度（精度为百分之一度）
temperature = temperature - 41380;
temperature = (temperature * 100L) / 156;
temp_int = temperature / 100;
temp_frac = temperature - (temp_int * 100);
printf (“Temperature is %+02d.%02d\n”, temp_int, temp_frac);
}
}
//-----------------------------------------------------------------------------
// 子程序初始化
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
// 系统时钟初始化
//-----------------------------------------------------------------------------
//
// 此程序初始化系统时钟，使用22.1184MHz晶体作为时钟源
//
void SYSCLK_Init (void)
{
int i; // 延时计数器
OSCXCN = 0x67; // 起动外部振荡器（22.1184MHz晶体）
for (i=0; i < 256; i++) ; // 等待振荡器启动 (>1ms)
while (!(OSCXCN & 0x80)) ; // 等待晶体振荡器稳定
OSCICN = 0x88; // 选择外部振荡器作为系统时钟源并使能丢失时钟检测器
}
//-----------------------------------------------------------------------------
// IO口初始化
//-----------------------------------------------------------------------------
//
// 配置数据交叉开关和通用IO口
//
void PORT_Init (void)
{
XBR0 = 0x04; // 使能UART0
```

```
XBR1 = 0x00;
XBR2 = 0x40; // 使能数据交叉开关和弱上拉
P0MDOUT |= 0x01; // 允许TX0为推挽输出
P1MDOUT |= 0x40; // 允许P1.6(LED)为推挽输出
}
//-----------------------------------------------------------------------------
// UART0初始化
//-----------------------------------------------------------------------------
//
// 配置UART0，使用定时器1产生波特率
//
void UART0_Init (void)
{
SCON0 = 0x50; // SCON0: 模式 1,8位UART,允许RX
TMOD = 0x20; // TMOD: 1定时器, 模式2, 8位重装
TH1 = -(SYSCLK/BAUDRATE/16); // 按波特率设置定时器1重装值
TR1 = 1; // 起动定时器1
CKCON |= 0x10; // 定时器1使用系统时钟为时基
PCON |= 0x80; // SMOD00=1
TI0 = 1; // 表示就绪
}
//-----------------------------------------------------------------------------
// ADC0初始化
//-----------------------------------------------------------------------------
//
// 配置ADC0，使用AD0BUSY作为转换源，使用左对齐输出模式,
// 使用正常跟踪模式，测量片内温度传感器器输出
// 禁止ADC0转换结束中断和ADC0窗口比较器中断。
//
void ADC0_Init (void)
{
ADC0CN = 0x81; // ADC0使能;正常跟踪模式
//  当写AD0BUSY时ADC0转换开始；ADC0数据左对齐
REF0CN = 0x07; // 使能温度传感器，片内VREF和VREF输出缓冲器
AMX0SL = 0x0f; // 选择温度传感器作为ADC多路模拟转换器输出
ADC0CF = (SYSCLK/2500000) << 3; // ADC转换时钟=2.5MHz
ADC0CF |= 0x01; // PGA增益=2
EIE2 &= ~0x02; // 禁止ADC0 EOC中断
EIE1 &= ~0x04; // 禁止ADC0窗口比较器中断
}
//-----------------------------------------------------------------------------
```

// DAC0_DTMF1.c

```
//-----------------------------------------------------------------------------
//版权所有Silabs集成产品有限公司 2001
//
// 作者: BW
// 日期: 2001年8月27日
//
// 目标器件: C8051F02x
// 链接工具: KEIL C51
//
```

```
// 描述:
// 次例程源代码在ADC0输出双音多频音调（DTMF）。根据常数<SAMPLERATED>
// 确定的速率定时更新DAC0的输出，并使用定时器4管理和定时。
//
// 输出频率与16位相位加法器成比例
// 在每一个DAC更新周期，相位加法器的值加到连续相位累加器<phase_accumulator>,
// 高位用于访问正弦查找表
//
//-----------------------------------------------------------------------------
// 包含文件
//-----------------------------------------------------------------------------
#include <c8051f020.h> // SFR声明
//-----------------------------------------------------------------------------
// C8051F02X的16位SFR定义
//-----------------------------------------------------------------------------
sfr16 DP = 0x82; // 数据指针
sfr16 TMR3RL = 0x92; // 定时器3重装值
sfr16 TMR3 = 0x94; // 定时器3计数器
sfr16 ADC0 = 0xbe; // ADC0数据
sfr16 ADC0GT = 0xc4; // ADC0大于窗口
sfr16 ADC0LT = 0xc6; // ADC0小于窗口
sfr16 RCAP2 = 0xca; // T2捕捉/重装
sfr16 T2 = 0xcc; // 定时器2
sfr16 RCAP4 = 0xe4; // 定时器4捕捉/重装
sfr16 T4 = 0xf4; // 定时器4
sfr16 DAC0 = 0xd2; // DAC0数据
sfr16 DAC1 = 0xd5; // DAC1数据
//-----------------------------------------------------------------------------
// 全局常量
//-----------------------------------------------------------------------------
#define SYSCLK 22118400 // 系统时钟频率（Hz）
#define SAMPLERATED 100000L // DAC更新率（Hz）
#define phase_precision 65536 // 相位累加器的范围
// DTMF相位加法器值是基于SAMPLERATED和<phase_precision>
#define LOW697697 * phase_precision / SAMPLERATED
#define LOW770 770 * phase_precision / SAMPLERATED
#define LOW852 852 * phase_precision / SAMPLERATED
#define LOW941 941 * phase_precision / SAMPLERATED
#define HI1209 1209 * phase_precision / SAMPLERATED
#define HI1336 1336 * phase_precision / SAMPLERATED
#define HI1477 1477 * phase_precision / SAMPLERATED
#define HI1633 1633 * phase_precision / SAMPLERATED
//-----------------------------------------------------------------------------
// 函数原型
//-----------------------------------------------------------------------------
void main (void);
void SYSCLK_Init (void);
void PORT_Init (void);
void Timer4_Init (int counts);
void Timer4_ISR (void);
//-----------------------------------------------------------------------------
```

```
// 全局变量
//-----------------------------------------------------------------------------
unsigned phase_add1; //保存低音调相位加法器
unsigned phase_add2; //保存低音调相位加法器
bit tone1_en; // 为1使能音调1
bit tone2_en; // 为1使能音调2
char code SINE_TABLE[256] = {
0x00, 0x03, 0x06, 0x09, 0x0c, 0x0f, 0x12, 0x15,
0x18, 0x1c, 0x1f, 0x22, 0x25, 0x28, 0x2b, 0x2e,
0x30, 0x33, 0x36, 0x39, 0x3c, 0x3f, 0x41, 0x44,
0x47, 0x49, 0x4c, 0x4e, 0x51, 0x53, 0x55, 0x58,
0x5a, 0x5c, 0x5e, 0x60, 0x62, 0x64, 0x66, 0x68,
0x6a, 0x6c, 0x6d, 0x6f, 0x70, 0x72, 0x73, 0x75,
0x76, 0x77, 0x78, 0x79, 0x7a, 0x7b, 0x7c, 0x7c,
0x7d, 0x7e, 0x7e, 0x7f, 0x7f, 0x7f, 0x7f, 0x7f,
0x7f, 0x7f, 0x7f, 0x7f, 0x7f, 0x7f, 0x7e, 0x7e,
0x7d, 0x7c, 0x7c, 0x7b, 0x7a, 0x79, 0x78, 0x77,
0x76, 0x75, 0x73, 0x72, 0x70, 0x6f, 0x6d, 0x6c,
0x6a, 0x68, 0x66, 0x64, 0x62, 0x60, 0x5e, 0x5c,
0x5a, 0x58, 0x55, 0x53, 0x51, 0x4e, 0x4c, 0x49,
0x47, 0x44, 0x41, 0x3f, 0x3c, 0x39, 0x36, 0x33,
0x30, 0x2e, 0x2b, 0x28, 0x25, 0x22, 0x1f, 0x1c,
0x18, 0x15, 0x12, 0x0f, 0x0c, 0x09, 0x06, 0x03,
0x00, 0xfd, 0xfa, 0xf7, 0xf4, 0xf1, 0xee, 0xeb,
0xe8, 0xe4, 0xe1, 0xde, 0xdb, 0xd8, 0xd5, 0xd2,
0xd0, 0xcd, 0xca, 0xc7, 0xc4, 0xc1, 0xbf, 0xbc,
0xb9, 0xb7, 0xb4, 0xb2, 0xaf, 0xad, 0xab, 0xa8,
0xa6, 0xa4, 0xa2, 0xa0, 0x9e, 0x9c, 0x9a, 0x98,
0x96, 0x94, 0x93, 0x91, 0x90, 0x8e, 0x8d, 0x8b,
0x8a, 0x89, 0x88, 0x87, 0x86, 0x85, 0x84, 0x84,
0x83, 0x82, 0x82, 0x81, 0x81, 0x81, 0x81, 0x81,
0x80, 0x81, 0x81, 0x81, 0x81, 0x81, 0x82, 0x82,
0x83, 0x84, 0x84, 0x85, 0x86, 0x87, 0x88, 0x89,
0x8a, 0x8b, 0x8d, 0x8e, 0x90, 0x91, 0x93, 0x94,
0x96, 0x98, 0x9a, 0x9c, 0x9e, 0xa0, 0xa2, 0xa4,
0xa6, 0xa8, 0xab, 0xad, 0xaf, 0xb2, 0xb4, 0xb7,
0xb9, 0xbc, 0xbf, 0xc1, 0xc4, 0xc7, 0xca, 0xcd,
0xd0, 0xd2, 0xd5, 0xd8, 0xdb, 0xde, 0xe1, 0xe4,
0xe8, 0xeb, 0xee, 0xf1, 0xf4, 0xf7, 0xfa, 0xfd
};
//-----------------------------------------------------------------------------
// 主程序
//-----------------------------------------------------------------------------
void main (void) {
WDTCN = 0xde; // 禁止看门狗
WDTCN = 0xad;
SYSCLK_Init ();
PORT_Init ();
REF0CN = 0x03; // 使能内部VREF发生器
DAC0CN = 0x97; // 使能DAC0为左对齐模式
// 使用定时器4更新进度表
```

```
Timer4_Init(SYSCLK/SAMPLERATED); // 初始化定时器T4产生DAC0进度表
tone1_en = 1; // 低组音调使能
tone2_en = 1; // 高组音调使能
phase_add1 = LOW697;
phase_add2 = HI1633;
EA = 1; // 允许全部中断
while (1); //
}
//-----------------------------------------------------------------------------
// 初始化程序
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
// 系统时钟初始化
//-----------------------------------------------------------------------------
//
// 此程序初始化系统时钟，使用22.1184MHz晶体作为时钟源
//
void SYSCLK_Init (void)
{
int i; // 延时计数器
OSCXCN = 0x67; // 启动外部振荡器（22.1184MHz晶体）
for (i=0; i < 256; i++) ; // 等待振荡器启动
while (!(OSCXCN & 0x80)) ; // 等待振荡器稳定
OSCICN = 0x88; // 选择外部振荡器作为系统时钟源并允许丢失时钟检测器
}
//-----------------------------------------------------------------------------
// IO口初始化
//-----------------------------------------------------------------------------
//
// 设置数据交叉开关和通用IO口
//
void PORT_Init (void)
{
XBR0 = 0x00;
XBR1 = 0x00;
XBR2 = 0x40; // 允许数据交叉开关和弱上拉
P1MDOUT |= 0x40; // 允许P1.6(LED)为推挽输出
}
//-----------------------------------------------------------------------------
// 定时器T4初始化
//-----------------------------------------------------------------------------
// 此程序初始化定时器T4，自动重装模式在指定的时间间隔<counts>内产生中断
//
void Timer4_Init (int counts)
{
T4CON = 0; // 停止定时器；设置为自动重装模式
CKCON |= 0x40; // T4M = ‘1’; 定时器T4计数系统时钟
RCAP4 = -counts; // 设置重装值
T4 = RCAP4;
EIE2 |= 0x04; // 允许定时器T4中断
T4CON |= 0x04; // 启动定时器T4
```

```
}
//-----------------------------------------------------------------------------
// 中断处理器
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
// 定时器T4中断服务程序
//-----------------------------------------------------------------------------
//
// 此中断服务程序是有定时器T4溢出引起。定时器设置为自动重装模式，
//和用来确定DAC输出取样率的时间
//注意：在调用中断服务程序其间向DAC0H写数值实际是在下一个定时器溢出时传送到ADC0
//
void Timer4_ISR (void) interrupt 16 using 3
{
static unsigned phase_acc1 = 0; // 保存低音调相位累加器
static unsigned phase_acc2 = 0; // 保存高音调相位累加器
char temp1; // 表结果的临时值
char temp2;
char code *table_ptr;
T4CON &= ~0x80; // 清除定时器T4溢出标志
table_ptr = SINE_TABLE;
if ((tone1_en) && (tone2_en)) {
phase_acc1 += phase_add1; // 更新相位累加器1（低音调)
temp1 = *(table_ptr + (phase_acc1 >> 8));
phase_acc2 += phase_add2; // 更新相位累加器2(高音调)
// 读表值
temp2 = *(table_ptr + (phase_acc2 >> 8));
// 现在更新DAC值。注意：与0x80异或将双极查表转换成单极数量
DAC0H = 0x80 ^ ((temp1 >> 1) + (temp2 >> 1));
} else if (tone1_en) {
phase_acc1 += phase_add1; //更新相位累加器1（低音调）
//读表值
temp1 = *(table_ptr + (phase_acc1 >> 8));
//现在更新DAC值。注意：与0x80异或将双极查表转换成单极数量
DAC0H = 0x80 ^ temp1;
} else if (tone2_en) {
phase_acc2 += phase_add2; //更新相位累加器1（高音调）
//读表值
temp2 = *(table_ptr + (phase_acc2 >> 8));
//现在更新DAC值。注意：与0x80异或将双极查表转换成单极数量
DAC0H = 0x80 ^ temp2;
}
}
```

“OSC_Cry1.c”

```
//-----------------------------------------------------------------------------
// OSC_CRY1.c
//-----------------------------------------------------------------------------
//版权所有Silabs集成产品有限公司 2001
//
// 作者：BW
// 日期：2001年8月25日
```

```
//
// 此例程说明如何配置外部振荡器驱动22.1184Mhz晶体，
// 及如何选择此外部振荡器作为系统时钟。
// 同时使能丢失时钟检测器复位功能。
// 假定一个22.1184Mhz晶体连接在XTAL1和XTAL2之间。
//
// 系统时钟频率存储器在全局常量SYSCLK
//
// 目标器件: C8051F02x
// 链接工具: KEIL C51 6.03 / KEIL EVAL C51
//
//-----------------------------------------------------------------------------
// 包含文件
//-----------------------------------------------------------------------------
#include <c8051f020.h> // SFR 声明
//-----------------------------------------------------------------------------
// C8051F02X的16位SFR定义
//-----------------------------------------------------------------------------
sfr16 DP = 0x82; // 数据指针
sfr16 TMR3RL = 0x92; // 定时器T3重装值
sfr16 TMR3 = 0x94; // 定时器T3计数器
sfr16 ADC0 = 0xbe; // ADC0数据
sfr16 ADC0GT = 0xc4; // ADC0大于窗口
sfr16 ADC0LT = 0xc6; // ADC0小于窗口
sfr16 RCAP2 = 0xca; // 定时器T2捕捉/重装
sfr16 T2 = 0xcc; // 定时器T2
sfr16 RCAP4 = 0xe4; // 定时器T4捕捉/重装
sfr16 T4 = 0xf4; // 定时器T4
sfr16 DAC0 = 0xd2; // DAC0数据
sfr16 DAC1 = 0xd5; // DAC1数据
//-----------------------------------------------------------------------------
// 全局常量
//-----------------------------------------------------------------------------
#define SYSCLK 22118400 // 系统时钟频率（Hz）
sbit LED = P1^6; // LED='1' 意为ON
sbit SW1 = P3^7; // SW1='0' 意为按下开关
//-----------------------------------------------------------------------------
// 函数原型
//-----------------------------------------------------------------------------
void SYSCLK_Init (void);
//-----------------------------------------------------------------------------
// 全局变量
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
// 主程序
//-----------------------------------------------------------------------------
void main (void) {
WDTCN = 0xde; // 禁止看门狗
WDTCN = 0xad;
SYSCLK_Init (); // 初始化振荡器
while (1);
```

```
}
//-----------------------------------------------------------------------------
// 主程序初始化
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
// 系统时钟初始化
//-----------------------------------------------------------------------------
//
// 此程序初始化系统时钟，使用22.1184MHz晶体作为时钟源
//
void SYSCLK_Init (void)
{
int i; // 延时计数器
OSCXCN = 0x67; // 起动外部振荡器（22.1184MHz晶体）
for (i=0; i < 256; i++) ; // 等待振荡器启动
while (!(OSCXCN & 0x80)) ; // 等待晶体振荡器稳定
OSCICN = 0x88; // 选择外部振荡器为系统时钟源并使能丢失时钟检测器
}
```

“OSC_Int1.c”

```
//-----------------------------------------------------------------------------
// OSC_INT1.c
//-----------------------------------------------------------------------------
//版权所有Silabs集成产品有限公司 2001
//
// 作者: BW
// 日期: 2001年8月25日
//
// 此程序示例如何配置内部振荡器到最大频率(~16MHz).同时使能丢失时钟检测器复位功能
//
// 系统时钟频率存储在全局常量SYSCLK.
//
// 目标器件: C8051F02x
// 链接工具: KEIL C51 6.03 / KEIL EVAL C51
//
//-----------------------------------------------------------------------------
// 包含文件
//-----------------------------------------------------------------------------
#include <c8051f020.h> // SFR声明
//-----------------------------------------------------------------------------
// C8051F02X的16位SFR定义
//-----------------------------------------------------------------------------
sfr16 DP = 0x82; // 数据指针
sfr16 TMR3RL = 0x92; // 定时器T3重装值
sfr16 TMR3 = 0x94; // 定时T3计数器
sfr16 ADC0 = 0xbe; // ADC0数据
sfr16 ADC0GT = 0xc4; // ADC0大于窗口
sfr16 ADC0LT = 0xc6; // ADC0小于窗口
sfr16 RCAP2 = 0xca; // 定时器T2捕捉/重装
sfr16 T2 = 0xcc; // 定时器T2
sfr16 RCAP4 = 0xe4; // 定时器T4捕捉/重装
sfr16 T4 = 0xf4; // 定时器T4
```

```
sfr16 DAC0 = 0xd2; // DAC0数据
sfr16 DAC1 = 0xd5; // DAC1数据
//-----------------------------------------------------------------------------
// 全局常量
//-----------------------------------------------------------------------------
#define SYSCLK 16000000 // 系统时钟频率（Hz）
sbit LED = P1^6; // LED='1' 意为ON
sbit SW1 = P3^7; // SW1='0' 意为按下开关
//-----------------------------------------------------------------------------
// 函数原型
//-----------------------------------------------------------------------------
void SYSCLK_Init (void);
//-----------------------------------------------------------------------------
// 全局变量
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
// 主程序
//-----------------------------------------------------------------------------
void main (void) {
WDTCN = 0xde; // 禁止看门狗
WDTCN = 0xad;
SYSCLK_Init (); // 初始化振荡器
while (1);
}
//-----------------------------------------------------------------------------
// 子程序初始化
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
// 系统时钟初始化
//-----------------------------------------------------------------------------
//
// 此程序初始化内部振荡器到最大设置和选择内部振荡器作为系统时钟源
// 同时使能丢失时钟检测器复位功能。
//
void SYSCLK_Init (void)
{
OSCICN = 0x87; // 配置内部振荡器到最高频率选择内部振荡器作为系统时钟源
// 同时使能丢失时钟检测器复位
}
```

“Timer0_Poll1.c”

```
//-----------------------------------------------------------------------------
// Timer0_Poll1.c
//-----------------------------------------------------------------------------
//版权所有Silabs集成产品有限公司 2001
//
// 作者: BW
// 日期: 2001年8月27日
//
// 此程序是一个使用定时器T0在查询模式的例子，实现一个延时计数器（精度1ms）
//
// 假设在XTAL1和XTAL2之间接22.1184MHz晶体
```

```
//
// 系统时钟频率存储在全局常量SYSCLK.
//
// 目标器件: C8051F02x
// 链接工具: KEIL C51 6.03 / KEIL EVAL C51
//
//-----------------------------------------------------------------------------
// 包含文件
//-----------------------------------------------------------------------------
#include <c8051f020.h> // SFR声明
//-----------------------------------------------------------------------------
// C8051FF02x的16位SFR定义
//-----------------------------------------------------------------------------
sfr16 DP = 0x82; // 数据指针
sfr16 TMR3RL = 0x92; // 定时器T3重装值
sfr16 TMR3 = 0x94; // 定时齐器T3计数器
sfr16 ADC0 = 0xbe; // ADC0数据
sfr16 ADC0GT = 0xc4; // ADC0大于窗口
sfr16 ADC0LT = 0xc6; // ADC0小于窗口
sfr16 RCAP2 = 0xca; // 定时器T2捕捉/重装
sfr16 T2 = 0xcc; // 定时器T2
sfr16 RCAP4 = 0xe4; // 定时器T4 捕捉/重装
sfr16 T4 = 0xf4; // 定时器T4
sfr16 DAC0 = 0xd2; // DAC0数据
sfr16 DAC1 = 0xd5; // DAC1数据
//-----------------------------------------------------------------------------
// 全局常量
//-----------------------------------------------------------------------------
#define SYSCLK 22118400 // 系统时钟频率（Hz）
sbit LED = P1^6; // LED=’1’ 意为ON
sbit SW1 = P3^7; // SW1=’0’ 意为按下开关
//-----------------------------------------------------------------------------
// 函数原型
//-----------------------------------------------------------------------------
void SYSCLK_Init (void);
void PORT_Init (void);
void Timer0_Delay (int ms);
//-----------------------------------------------------------------------------
// 全局变量
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
// 主程序
//-----------------------------------------------------------------------------
void main (void) {
WDTCN = 0xde; // 禁止看门狗定时器
WDTCN = 0xad;
SYSCLK_Init (); // 初始化振荡器
PORT_Init (); // 初始化数据交叉开关和通用IO口
while (1) {
Timer0_Delay (100); // 延时100ms
LED = ~LED; // 改变LED状态
```

```
}
}
//-----------------------------------------------------------------------
// 初始化子程序
//-----------------------------------------------------------------------
//-----------------------------------------------------------------------
// 系统时钟初始化
//-----------------------------------------------------------------------
//
// 此程序初始化系统时钟使用22.1184MHz晶体作为时钟源
//
void SYSCLK_Init (void)
{
int i; // 延时计数器
OSCXCN = 0x67; // 启动外部振荡器22.1184MHz晶体
for (i=0; i < 256; i++) ; // 等待振荡器启动
while (!(OSCXCN & 0x80)) ; // 等待振荡器稳定
OSCICN = 0x88; // 选择内部振荡器为时钟源并使能丢失时钟检测器
}
//-----------------------------------------------------------------------
// IO口初始化
//-----------------------------------------------------------------------
//
// 配置数据交叉开关和通用IO口
//
void PORT_Init (void)
{
XBR0 = 0x00;
XBR1 = 0x00;
XBR2 = 0x40; // 使能数据交叉开关和弱上拉
P1MDOUT |= 0x40; // 使能P1.6(LED)为推挽输出
}
//-----------------------------------------------------------------------
// Timer0_Delay
//-----------------------------------------------------------------------
//
// 配置定时器T0延时.
//
void Timer0_Delay (int ms)
{
int i; // 毫秒计数器
TCON &= ~0x30; // 停止定时器T0并清除溢出标志
TMOD &= ~0x0f; // 配置定时器T0为16位模式
TMOD |= 0x01;
CKCON |= 0x08; // 定时器T0计数系统时钟
for (i = 0; i < ms; i++) { // 数毫秒
TR0 = 0; // 停定时器T0
TH0 = (-SYSCLK/1000) >> 8; // 设置定时器T0 1ms溢出
TL0 = -SYSCLK/1000;
TR0 = 1; // 启动定时器T0
while (TF0 == 0); // 等待溢出
```

```
TF0 = 0; // 清除溢出标志
}
}
```