8051单片机

C语言

软件设计的艺术

赖麒文 编著

科学出版社 文魁资讯股份有限公司

8051 单片机 C 语言
软件设计的艺术

赖麒文　编著

科学出版社

2002

内 容 简 介

本书主要介绍了 8051 单片机 C 语言软件设计的思维与解决方法。本书每一章都是一个精彩的例子，范例说明深入浅出。重点介绍软件的设计流程、软件的构思和解决方法。在实例中说明模块化程序设计的各种指令的应用，使用户可以更有效地学习。

本书适合于从事 8051 单片机应用设计的人员参考使用。

图字：01-2001-4800 号

图书在版编目（CIP）数据

8051 单片机 C 语言软件设计的艺术/赖麒文编著.—北京：科学出版社，2002

ISBN 7-03-010410-2

Ⅰ.8... Ⅱ.赖... Ⅲ.C 语言—软件设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2002）第 028788 号

科学出版社 出版

北京东黄城根北街16号

邮政编码:100717

http://www.sciencep.com

新蕾印刷厂 印刷

科学出版社发行 各地新华书店经销

*

2002 年 6 月第 一 版 开本：720×1000 1/16

2002 年 6 月第一次印刷 印张：42

印数：1—5 000 字数：836 000

定价:58.00 元

（如有印装质量问题，我社负责调换〈路通〉）

导　读

第 1 章：介绍如何输出方波信号，使喇叭发出声音的方法，包括发出“哔”声的函数和分别传递一个、二个及三个自变量的“哔”声函数，以及利用定时器产生方波信号而令喇叭发出“哔”声，并叙述音阶与频率的关系，以此作为演奏音乐的基础。

第 2 章：演奏音乐的程序由 main( )函数开始，将其所有函数定义在一个 main.c 的模块内，并分别以各种指令结构来循序渐进地介绍软件构建的思维与解决方法。

第 3 章：以模块化的设计方式将单独的一个 main.c 模块细分为 main.c 模块、initial.c 模块、delay.c 模块、music.c 模块以及其对应的包括文件，可以使程序易于了解，节省开发时间。而且，用范例来说明各种应用方法，以使读者建立整体思维，并进行有效的学习。

第 4 章：详细介绍如何利用定时器的中断方法来产生音阶的频率，并由 I/O 输出此方波信号而驱动喇叭发出正确的音阶。当连续产生各音符的音调频率时，则形成演奏音乐，并渐进式地说明什么样的设计方法是最好的。

第 5 章：音符的形成有两个要素：音调及音长，当音调以定时器中断方法来产生，音长是否也可以由定时器来产生呢？本章介绍如何利用 timer0 及 timer1 两个定时器中断方法来演奏音乐，并特别说明当音长计时中断时间太短时所造成的影响以及解决的方法。

第 6 章：说明音乐中“移调”的概念，分别以查表法和计算法来举例说明 D 大调、降 E 大调、F 大调、G 大调、降 A 大调、降 B 大调。并以 TACT 开关的按键动作来阐述移调的功能，而以外部中断的方法来达到音乐演奏中实时移调的功能。

第 7 章：介绍如何以按键开关来选曲，以“哔”声和 LED 闪烁方式作为选曲的提示动作，并以下列技巧来说明按键的处理方法：开关持续按着的重复动作、开关持续按着也只动作一次、消除按键弹跳波的程序规划、持续按键以延时方式来继续执行动作，及持续按键以定时器计时方式来继续执行动作。同时，通过此方式来培养读者软件设计的能力并使读者养成慎密的思维方式。

第 8 章：以 9 个按键开关分别代表 1～9 首的按键选曲，并介绍如何以 I/O 的方式、SCAN 的方式以及 ADC 的方式来检测按键动作，以及当微电脑 I/O 不敷使用时的解决方法。

第 9 章：介绍如何以 DIP 指拨开关作为选曲的方法。当利用 2P 的 DIP 开关时，其各自独立或相邻 I/O 作为选曲输入的软件构思与解决方法；当利用 4P 的 DIP 开关

可选曲 1～15 首曲目，其低 4 位或高 4 位作为选曲输入及起始键输入的软件构思与解决方法。

第 10 章：介绍当按下选曲键或数字键以七段显示器立即显示的方法，分别是以一个 I/O 端口直接驱动显示器，以 IC7447 来驱动显示器，以及以串行移位缓存器 IC4094 扩充为输出端口作为 IC7447 BCD 码的输入而驱动显示器。

第 11 章：分别以单曲循环、顺序播放、随机选曲及播放简介的演奏功能来说明如何加强系统的附加价值，以及随机选曲功能如何能以开关弹跳波的方式来达成。

第 12 章：介绍如何以 ADC 方式检测更多的按键，而将 LED 以更简单的 I/O 来控制的方法，并在演奏曲目时能立即显示出曲目编号；最后介绍如何将功能模式及曲目编号存储起来，以便重新开机后能立即显示最后输入的曲目编号，并详细介绍记忆 IC 及 IIC BUS 的控制与应用、其软件构建的思维与解决方法。

目　录

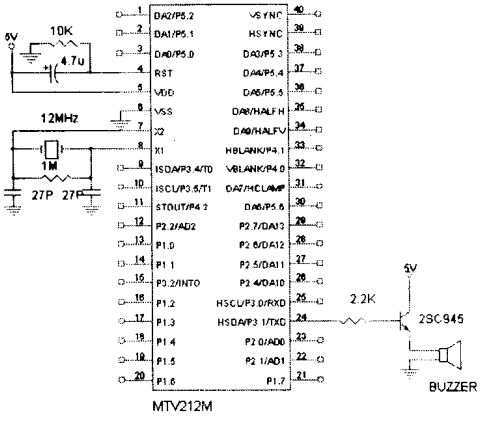
第 1 章　声音基础

1-1　Beep 无自变量

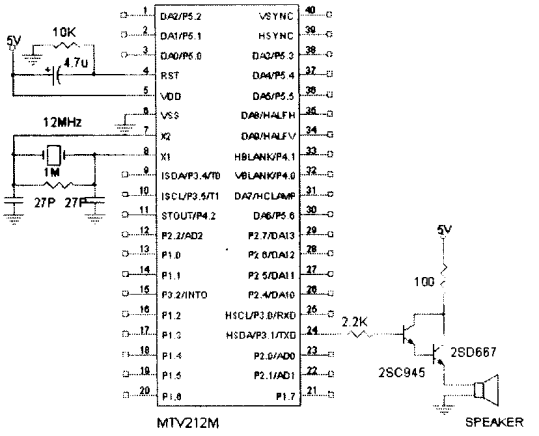
目的：通过 CPU 的 I/O 引脚以软件方式使喇叭发出声音。

电路图：驱动喇叭的线路，为了使其流过电流而控制喇叭的声音大小，可分为以下三种方法。

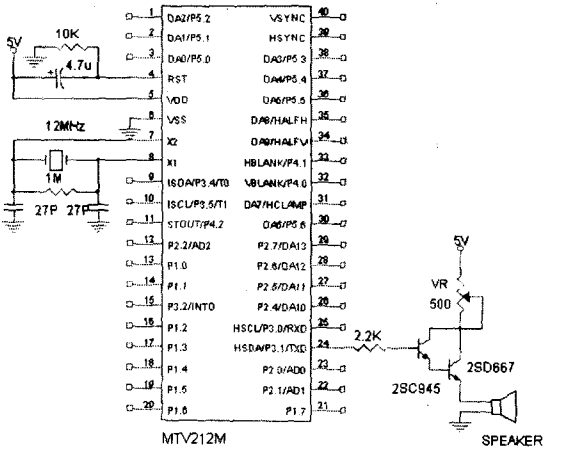
1. 晶体管方式：



2. 达灵顿大电流方式：



3. 可调电阻控制声音大小方式：



原理：其实原理很简单，只需令 I/O 引脚不断地产生方波，则晶体管就不断地 ON/OFF，其集电极电流也就不断地流经喇叭或截止不流过喇叭，而使得喇叭发出声音。也就是控制方波的频率，则喇叭就会发出此频率的声音，而声音的长短只要控制此方波频率的时间长短即可。

软件构建的思维与解决方法

测试喇叭发出声音的软件程序，为了进行音乐演奏，必须将喇叭通过软件方法来发声。除了必须将 CPU 进行初始化外（只要以微电脑进行控制的，都必须进行 CPU 的初始化，以稳定 CPU 状态的功能），在 main( )主程序中每秒钟发出一声“哔”并一直重复下去，而延迟程序有三层 for 循环并通过自变量来达到可任意延迟多少时间。而 Beep( )函数的实现方法：即将 I/O 先设定为“1”电位，此时晶体管导通，电流将流过喇叭，并以 for 循环作短暂延迟后，再将 I/O 清除为“0”电位，此时晶体管为截止状态，电流将不会流过喇叭，同样作短暂延迟，这样作 17 次就会发出“哔”声了。而在程序前面必须包括 define.h 文件，以便可以使用自定义类型来代替数据类型的声明，例如：Bit、Bool、Byte、Word、Long，以及内存类型的转换字符，如：DATA、IDATA、PDATA、XDATA、RDATA 等，而为了调用其他函数，则应在程序前面先进行函数的声明，程序如下：

```
/*****************************/
/* include files             */
/*****************************/
#include  "define.h"
#include  "mtv212m.h"
```

```
void PowerOnInit(void);
void DelayX10ms(Word count);
void Beep(void);

/*****************************/
void Main( void )
{
    PowerOnInit();

    while( 1 )
    {
       Beep( );
       DelayX10ms(100);
    }
}

/*****************************/
void PowerOnInit(void)
{
    IE  = 0;              //disable all interrupt
    PSW = 0;              //bank 0

    IP  = 0x0b;      //hi priority:int0,timer0,timer1

    TMOD= 0x11;           //set timer1,timer0 mode

    TR0 = 0;         //stop timer0
    TR1 = 0;              //stop timer1
    IT0 = 1;              //set int0:falling eage trigger

    EX0 = 0;              //disable int0  interrupt
    ET1 = 0;              //disable timer1 interrupt
    ET0 = 0;              //disable timer0 interrupt

    EA  = 1;              //enable all interrupt gate
}

/*****************************/
void DelayX10ms(Word count)
{
    Word i,j,k;

    for(i=0; i<count; i++)
       for(j=0; j<10; j++)
```

```
        for(k=0; k<120; k++)
           ;
}

/*****************************/
void Beep(void)
{
   Byte  j,k;

   for(j=0; j<17; j++)
   {
      P3_1 = 1;
      for(k=0; k<200; k++)
         ;
      P3_1 = 0;
      for(k=0; k<200; k++)
         ;
   }
}
```

包括文件 define.h 的内容主要为自定义类型，bit 数据类型可以使用 Bit 或 Bool 声明，unsigned char 数据类型可以使用 Byte 声明，unsigned int 数据类型可以使用 Word 声明，unsigned long 数据类型可以用 Long 声明，这样有利于缩短文字，同时能望文生义。并定义常数 DATA 以内存类型 data 来转换；常数 IDATA 以内存类型 idata 来转换；常数 PDATA 以内存类型 pdata 来转换；常数 XDATA 以内存类型 xdata 来转换；常数 RDATA 以内存类型 code 来转换，其定义如下：

```
#ifndef   _DEFINE_H
#define   _DEFINE_H

//declare
typedef   bit              Bit;
typedef   bit              Bool;
typedef   unsigned char    Byte;
typedef   unsigned int     Word;
typedef   unsigned long    Long;

#define   DATA          data
#define   IDATA         idata
#define   PDATA         pdata
#define   XDATA         xdata
#define   RDATA         code

#endif
```

包括文件 mtv212m.h 的内容是 8052 CPU 各缓存器地址的定义和缓存器中的每一个位地址定义以及 I/O 的符号定义，将每一个 I/O 引脚以符号来定义。例如：P1_0 代表 I/O P1.0、P2_3 代表 I/O P2.3、P3_5 代表 I/O P3.5 等，其定义如下：

```
/*---8052 sfr address declare---*/
sfr  ACC      = 0xE0;
sfr  B        = 0xF0;
sfr  PSW      = 0xD0;
sfr  SP       = 0x81;
sfr  DPL      = 0x82;
sfr  DPH      = 0x83;
sfr  P0       = 0x80;
sfr  P1       = 0x90;
sfr  P2       = 0xA0;
sfr  P3       = 0xB0;
sfr  IE       = 0xA8;
sfr  IP       = 0xB8;
sfr  PCON     = 0x87;
sfr  TCON     = 0x88;
sfr  TMOD     = 0x89;
sfr  TL0      = 0x8A;
sfr  TL1      = 0x8B;
sfr  TH0      = 0x8C;
sfr  TH1      = 0x8D;
sfr  T2CON    = 0xC8;
sfr  RCAP2L   = 0xCA;
sfr  RCAP2H   = 0xCB;
sfr  TL2      = 0xCC;
sfr  TH2      = 0xCD;
sfr  SCON     = 0x98;
sfr  SBUF     = 0x99;

/*---PSW---*/
sbit CY       = 0xD7;
sbit AC       = 0xD6;
sbit F0       = 0xD5;
sbit RS1      = 0xD4;
sbit RS0      = 0xD3;
sbit OV       = 0xD2;
sbit P        = 0xD0;
/*---TCON---*/
sbit TF1      = 0x8F;
sbit TR1      = 0x8E;
sbit TF0      = 0x8D;
```

```
sbit TR0      = 0x8C;
sbit IE1      = 0x8B;
sbit IT1      = 0x8A;
sbit IE0      = 0x89;
sbit IT0      = 0x88;
/*---T2CON---*/
sbit TF2      = 0xCF;
sbit EXF2     = 0xCE;
sbit RCLK     = 0xCD;
sbit TCLK     = 0xCC;
sbit EXEN2    = 0xCB;
sbit TR2      = 0xCA;
sbit CT2      = 0xC9;
sbit CPRL2    = 0xC8;
/*---PCON---*/
sbit SMOD     = 0x8e;
sbit GF1      = 0x8A;
sbit GF0      = 0x89;
sbit PD       = 0x88;
sbit IDL      = 0x87;
/*---SCON---*/
sbit SM0      = 0x9F;
sbit SM1      = 0x9E;
sbit SM2      = 0x9D;
sbit REN      = 0x9C;
sbit TB8      = 0x9B;
sbit RB8      = 0x9A;
sbit TI       = 0x99;
sbit RI       = 0x98;
/*---IE---*/
sbit EA       = 0xAF;
sbit ET2      = 0xAD;
sbit ES       = 0xAC;
sbit ET1      = 0xAB;
sbit EX1      = 0xAA;
sbit ET0      = 0xA9;
sbit EX0      = 0xA8;
/*---IP---*/
sbit PT2      = 0xBD;
sbit PS       = 0xBC;
sbit PT1      = 0xBB;
sbit PX1      = 0xBA;
sbit PT0      = 0xB9;
sbit PX0      = 0xB8;
```

```
/*---P0---*/
sbit P0_0      = P0 ^ 0;
sbit P0_1      = P0 ^ 1;
sbit P0_2      = P0 ^ 2;
sbit P0_3      = P0 ^ 3;
sbit P0_4      = P0 ^ 4;
sbit P0_5 = P0 ^ 5;
sbit P0_6      = P0 ^ 6;
sbit P0_7      = P0 ^ 7;

/*---P1---*/
sbit P1_0 = P1 ^ 0;
sbit P1_1 = P1 ^ 1;
sbit P1_2 = P1 ^ 2;
sbit P1_3      = P1 ^ 3;
sbit P1_4      = P1 ^ 4;
sbit P1_5 = P1 ^ 5;
sbit P1_6      = P1 ^ 6;
sbit P1_7      = P1 ^ 7;

/*---P2---*/
sbit P2_0 = P2 ^ 0;
sbit P2_1 = P2 ^ 1;
sbit P2_2      = P2 ^ 2;
sbit P2_3 = P2 ^ 3;
sbit P2_4 = P2 ^ 4;
sbit P2_5      = P2 ^ 5;
sbit P2_6      = P2 ^ 6;
sbit P2_7      = P2 ^ 7;

/*---P3---*/
sbit P3_0 = P3 ^ 0;
sbit P3_1      = P3 ^ 1;
sbit P3_2      = P3 ^ 2;
sbit P3_3      = P3 ^ 3;
sbit P3_4      = P3 ^ 4;
sbit P3_5      = P3 ^ 5;
sbit P3_6      = P3 ^ 6;
sbit P3_7      = P3 ^ 7;
```

1-2　Beep 自变量 X1

目的：通过传递声音的频率，自变量可弹性地控制喇叭“哔”声的高低。

软件构建的思维与解决方法

利用传入自变量的方式来取代程序中固定的数值，将使程序本身更具弹性，更适合作各种不同的应用。例如：要产生三种不同的声调来表示三种不同的状况提示时，利用不同的自变量内容即可达到要求，而无需再重新编写程序，这样可大大缩小整体的程序内存空间，所以利用自变量的传递确实是很好且实用的方法。在 Beep( ) 函数中的 tone 变量，即是此函数用来接收调用函数的传递值，经过运算处理后形成产生 I/O P3.1 “1” 电位及 “0” 电位的时间，也就是方波的频率参数，当 tone 内容不同时则方波的频率也会不同，则发出声音的音调也会不同，从而达到要求，其程序如下：

```
/*****************************/
/* include files             */
/*****************************/
#include  "define.h"
#include  "mtv212m.h"

void PowerOnInit(void);
void DelayX10ms(Word count);
void Beep(Byte tone);

/*****************************/
void Main( void )
{
   PowerOnInit();

   while( 1 )
   {
      Beep(5);
      DelayX10ms(100);
   }
}

/*****************************/
void PowerOnInit(void)
{
   IE  = 0;              //disable all interrupt
   PSW = 0;              //bank 0

   IP  = 0x0b;           //hi priority:int0,timer0,timer1

   TMOD= 0x11;           //set timer1,timer0 mode
```

```
    TR0 = 0;              //stop timer0
    TR1 = 0;              //stop timer1
    IT0 = 1;              //set int0:falling eage trigger

    EX0 = 0;              //disable int0   interrupt
    ET1 = 0;              //disable timer1 interrupt
    ET0 = 0;              //disable timer0 interrupt

    EA  = 1;              //enable all interrupt gate
}

/*****************************/
void DelayX10ms(Word count)
{
    Word i,j,k;

    for(i=0; i<count; i++)
       for(j=0; j<10; j++)
          for(k=0; k<120; k++)
             ;
}

/*****************************/
void Beep(Byte tone)
{
    Byte  j,k,spfreq;

    spfreq  = (1000/tone)/2;

    for(j=0; j<17; j++)
    {
       P3_1 = 1;
       for(k=0; k<spfreq; k++)
          ;
       P3_1 = 0;
       for(k=0; k<spfreq; k++)
          ;
    }
}
```