实验四 模数和数模转换

杨庆龙
1500012956

2018.4.11

1 实验目的

- 了解模数和数模转换电路的原理和使用方法
- 掌握MCS-51系列单片机中定时器和计数器的使用方法
- 掌握使用示波器，信号源对单片机系统进行调试的方法

2 实验原理

2.1 数模转换器

2.1.1 控制

单片机系统中，有两个12位DAC和两个比较器，通过特殊功能寄存器DAC0CN实现控制,详见1

Table 1: DAC0CN寄存器结构

D7	D6	D5	D4	D3	D2	D1	D0
DAC0EN			DAC0MD1	DAC0MD0	DAC0DF2	DAC0DF1	DAC0DF0

- DAC0EN:使能设置，0禁止，1允许
- DAC0MD1-0:工作模式，00写入触发，01，10，11分别对应计数器3,4,2
- DAC0DF2-0:数据格式，000:右对齐，001右对齐左移一位..1xx左对齐

2.1.2 参考电压

单片机系统中，使用REF0CN寄存器进行参考电压控制，详见2

- AD0VRS:ADC0的参考电压，0为VREF0，1为DAC0输出
- AD1VRS:ADC1的参考电压，0为VREF1，1为AV+

Table 2: REF0CN寄存器结构

D7	D6	D5	D4	D3	D2	D1	D0
			AD0VRS	AD1VRS	TEMPE	BIASE	REFBE

- TEMPE:内部温度传感器,0禁止，1允许
- BIASE:偏置电压允许位，必须设为1
- REFBE:内部参考电压允许位，0禁止，1允许

2.2 模数转换器

单片机内部有一个片内ADC0，一个9通道输入多路选择开关和可编程增益放大器，提供100ksps下的真12位进度。

2.2.1 多路选择器

使用AMX0SL寄存器选择输入情况，0-7对应AIN0-AIN7，其余为温度传感器

2.2.2 时钟设置

ADC0使用ADC0CF寄存器设置SAR的时钟，详见3

Table 3: ADC0CF寄存器结构

D7	D6	D5	D4	D3	D2	D1	D0
ADC0SC4	ADC0SC3	ADC0SC2	ADC0SC1	ADC0SC0	AMP0GN2	AMP0GN1	AMP0GN0

- ADC0SC:设置时钟频率为 $\frac{SYSCLK}{CLK_{SAR0}} - 1$
- AMP0GN:设置内部电压增益，000,001,010,011为1，2，4，8，10x为16，11x为0.5

2.3 定时器2

定时器2共有三种工作方式:带捕获的16位定时器/计数器模式，带自装载的16位定时器/计数器，串口0波特率发生器

Table 4: T2CON寄存器结构

D7	D6	D5	D4	D3	D2	D1	D0
TF2	EXF2	RCLK0	TCLK0	EXEN2	TR2	C/T2	CP/RL2

2.4 定时器3

定时器3仅可工作在自装载模式下

Table 5: TMR3CN寄存器结构

D7	D6	D5	D4	D3	D2	D1	D0
TF3					TR3	T3M	T3XCLK

2.5 定时器4

定时器4和定时器2相同

Table 6: TMR3CN寄存器结构

D7	D6	D5	D4	D3	D2	D1	D0
TF3					TR3	T3M	T3XCLK

3 实验内容

3.1 数模转换

```
#include <C8051F020.h>
#include "../include/time.h"
#include "../include/adc.h"
#define SYSCLK 22118400
#define SAMPLERATE 10000

void main(void) {
  WDTCN = 0xDE;
  WDTCN = 0xAD;
   sysclk_init();
   dac0_init();
   int4_init(SYSCLK/SAMPLERATE);
  EA = 1;
   while(1);
}

void timer4_int(void) interrupt 16 {
   static unsigned phase = 0;
  DAC0 = phase;
   phase += 0x10;
  T4CON &= ~0x80;
}
```

3.2 模数转换

使用ADC进行模数转换，并将结果输出到数码管显示，参考程序如下。

```
#include <C8051F020.h>
```

```
#include "../include/time.h"
#include "../include/adc.h"
#include "../include/keyboard.h"

#define SYSCLK 22118400
#define SAMPLERATE 10000

unsigned int adc_result;

void main() {
  WDTCN = 0xDE;
  WDTCN = 0xAD;
  sysclk_init();
  int3_init(SYSCLK/SAMPLERATE);
  adc0_init(SYSCLK/2500000);
  display_port_init();
  EA = 1;
  EIE2 |= 0x02;
  while(true) {
    unsigned char i;
    unsigned j;
    j = adc_result;
    for(i = 0;i < 4;i ++) {
      digital_number = digital_trans[j & 0xF];
      delay(1);
      digital_selecte = digital_index[i];
      j = j >> 4;
      delay(1000);
    }
  }
}

void adc0_int(void) interrupt 15 {
  static unsigned int count = 0;
  count ++;
  AD0INT = 0;
  if(count >= SAMPLERATE/2) {
    adc_result = ADC0;
    count = 0;
  }
}
```

3.3 模数数模联调

3.3.1 转换输出

模数转换从信号源输入，将转换结果再通过数模转换输出，输出结果用示波器查看。

```
#include <C8051F020.h>
#include "../../includes/time.h"
#include "../../includes/adc.h"
#define SAMPLERATE 10000

unsigned adc_result;

void main(void) {
  WDTCN = 0xDE;
  WDTCN = 0xAD;
   adc_result = 0x60;
   sysclk_init();
   dac0_init();
   adc0_init(SYSCLK/2500000);
   int4_init(SYSCLK/SAMPLERATE);
   int3_init(SYSCLK/SAMPLERATE);
   EIE2 |= 0x02;
  EA = 1;
   while(1);
}

void timer4_int(void) interrupt 16 {
        int i;
        i = 0;
        i ++;
  DAC0 = adc_result;
  T4CON &= ~0x80;
}

void adc_int(void) interrupt 15 {
        int i;
        i = 0;
        i ++;
        AD0INT = 0;
    adc_result = ADC0;
}
```

3.3.2 音频回放

音谱输入已经接到AIN1上(使用AMX0SL选择)，音谱输出用DAC1驱动(低位0xD5,高位0xD6)。

```
#include <C8051F020.h>
```

```
#include "../../includes/time.h"
#include "../../includes/adc.h"
#include "../../includes/display.h"
#define SAMPLERATE 8000
#define STATE_REC 1
#define STATE_PLAY 2
//#define BUF_LEN 16384
#define BUF_LEN 16384
unsigned int xdata buf[BUF_LEN] _at_ (0x0000);
unsigned int point;
unsigned int state;

void main(void) {
  WDTCN = 0xDE;
  WDTCN = 0xAD;
   sysclk_init();
   state = STATE_REC;
   point = 0;
   dac1_init();
   display_port_init();
   adc0_init(SYSCLK/2500000,0x01);
   int4_init(SYSCLK/SAMPLERATE);
   int3_init(SYSCLK/SAMPLERATE);
   EIE2 |= 0x02;
   EA = 1;
   T4CON &= ~0x04;
   while(1);
}

void timer4_int(void) interrupt 16 {
   if(state == STATE_PLAY) {
     DAC1 = buf[point++];
     if(point >= BUF_LEN){
       point = 0;
       state = STATE_REC;
           TMR3CN |= 0x04;
           T4CON &= ~0x04;
     }
   }
   T4CON &= ~0x80;
}

void adc_int(void) interrupt 15 {
         AD0INT = 0;
         if(state == STATE_REC) {
       buf[point++] = ADC0;
```

```
        if(point >= BUF_LEN) {
          point = 0;
          state = STATE_PLAY;
                  TMR3CN &= ~0x04;
                  T4CON |= 0x04;
        }
    }
}
```

4 思考题

- CPU在每个程序周期都会检查中断寄存器，当发现中断时，会将目前运行的程序停止，并保护现场。之后依据中断编号到中断向量表中读取相应的中断处理程序地址，之后跳转到相应地址，进行中断处理。处理结束后，恢复现场，继续处理原程序。
  使用汇编语言时，先写好中断处理程序，再将该处理程序的起始地址填入中断向量表即可。
  跳转到中断1的代码如下
  ORG 03H
  JMP INT1

- 因为ADC和DAC要面对不同的采样的情况，所以相应地也需要多种触发方式，以满足具体使用环境的要求。使用计时器溢出触发的好处有两点，一个是两次采样的时间间隔与CPU运行状态无关，采样频率稳定。另一个好处是使用计数器进行采样间隔调整可以实现高精度采样频率。

- C51中的sfr16使用操作两个8bit特殊功能寄存器的方法实现对一个16bit特殊功能寄存器的操作。指定的变量地址为16位中的低8位起始地址，读操作时先读低字节，再读高字节，写操作时先写高字节，后写低字节。