### 13.8 由PWM输出构成 DA模拟量输出和将其采样的AD例程

本节描述了在微芯PIC32MX220F032B型芯片上的DA-AD综合示例。通过PWM输出占空比从0~100%渐变循环变化的数字信号，该信号通过RC滤波后，接入10位AD接口AN1～AN4，并将AD采样结果通过SPI接口输出给8段数码管显示（0～1023）。

适用范围：本节所描述的代码适用于PIC32MX220F032B型芯片（28 引脚SOIC封装），对于其他型号或封装的芯片，未经测试，不确定其可用性。

表13-2 SPI引脚和A/D引脚选择硬件配置表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 功能描述 | 引脚号 | 复用端口选择指定功能所用代码 | 说明 |
| 1 | SCK2 | 26 | 由SPI模块自动选择(SCK2只能选这个引脚) | SPI数据时钟 |
| 2 | SDO2 | 17 | PPSOutput(2, RPB8, SDO2) | SPI数据输出 |
| 3 | SLCK | 18 | PORTSetPinsDigitalOut(IOPORT\_B, BIT\_9) | 外部移位寄存器数据锁存 |
| 4 | RPB7 | 16 | RPB7Rbits.RPB7R = 0b0101 | 复用引脚RPB7，配置为OC1输出 |
| 5 | AN0 | 2 | ANSELAbits.ANSA0 = 1 | PORTA.0，使能为模拟通道0 |

基于PWM的D/A变换电路：采用PIC32MX220F032B型芯片所具有的脉宽调制PWM通道输出可变占空比的PWM波形，将PWM波形通过RC滤波电路将其形成大小可调的模拟量，即可实现D/A转换。如图13-6所示。

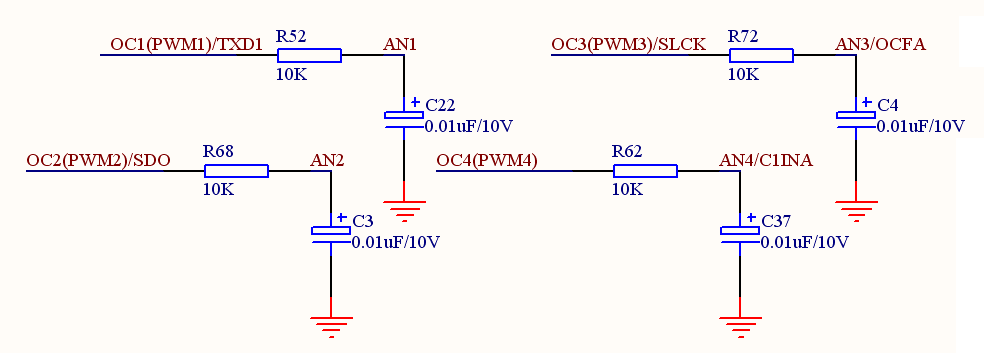
图13-6：PWM滤波电路

图13-6中所示的为较简单的RC电路构成的一阶滤波电路，电阻的阻值采用10千欧，电容的大小采用0.01微法/10V，滤波电路的输出引脚AN1～AN4与芯片的模拟输入引脚连接，可以用万用表直流电压档检测D/A输出的电压值，也可以用A/D转换模块将其电压值转换成数字量送到LED七段数码显示器显示出转换值。



图13-7：主函数流程框图

**1、主函数例程（程序流程框图见图13-7所示）**

|  |
| --- |
| int **main**(void)  {  int i,ads;  SYSTEMConfig(SYS\_FREQ, SYS\_CFG\_WAIT\_STATES | SYS\_CFG\_PCACHE);  INTDisableInterrupts();  INTConfigureSystem(INT\_SYSTEM\_CONFIG\_MULT\_VECTOR);  SpiInitDevice();  AD10init();  PWMinit();  Timer1Init();  INTEnableInterrupts();  while(1)  {  if(ADS\_flag > 0)  {  ADS\_flag = 0;  adrst[adptr] = AD10Sample();  adptr++;  if(adptr > 15)  {  adptr = 0;  ads = 0;  for(i=0;i<16;i++)  ads += adrst[i];  ads = ads >> 4;  AD10DispRst(ads);  }  }  }  return 0;  } |

图13-8：PWMinit函数流程框图 图13-9：定时器中断函数流程框图 图13-10：AD采样函数流程框图

**2、PWMinit函数例程（程序流程框图见图13-8所示）**

|  |
| --- |
| void **PWMinit**()  {  //PWM引脚关联  RPB7Rbits.RPB7R = 0b0101; //PWM1  //PWM1初始化  OC1CON = 0x0000; // Turn off OC1 while doing setup.  OC1RS = pwm1; // Initialize secondary Compare Register  OC1CON = 0x0006; // Configure for PWM mode  //定时器2周期设定+开启  PR2 = PWM\_PR; // Set period  T2CONSET = 0x8000; // Enable Timer2  //PWM1开启  OC1CONSET = 0x8000; // Enable OC1  } |

**3、定时器中断函数例程（程序流程框图见图13-9所示）**

|  |
| --- |
| // Configure the Timer 1 interrupt handler  void \_\_ISR(\_TIMER\_1\_VECTOR, ipl2) **Timer1Handler**(void)  {  // Clear the interrupt flag  INTClearFlag(INT\_T1);    ADS\_cnt++;  if(ADS\_cnt > 10) //0.01s  {  ADS\_cnt = 0;  ADS\_flag = 1;  }  pwm\_cnt++;  if(pwm\_cnt > 1)  {  pwm\_cnt = 0;  if(pwm1\_d == 0)  {  pwm1 ++;  if(pwm1 > DUTYMAX )  {  pwm1 = DUTYMAX;  pwm1\_d = 1;  }  }  else  {  if(pwm1 == 0)  {  pwm1 = 0;  pwm1\_d = 0;  }  else  pwm1 --;  }  OC1RS = pwm1;  }  } |

**4、AD采样函数例程（程序流程框图见图13-10所示）**

|  |
| --- |
| UINT16 **AD10Sample**(void)  {  AD1CON1bits.ASAM = 1; // 自动采样：31个Tad后自动转换  while (!AD1CON1bits.DONE); // 等待转换完成  AD1CON1bits.ASAM = 0; // 结束本次采样/转换操作  return ADC1BUF0; //返回采样结果  } |

### 附件：代码

|  |
| --- |
| /\*  \* File: DAAD\_Example.c  \*/  #include <plib.h>  // Configuration Bit settings  // SYSCLK = 48 MHz (8MHz Crystal / FPLLIDIV \* FPLLMUL / FPLLODIV)  // PBCLK = 48 MHz (SYSCLK / FPBDIV)  // Primary Osc w/PLL (XT+,HS+,EC+PLL)  // WDT OFF  #pragma config FPLLMUL = MUL\_24, FPLLIDIV = DIV\_2, FPLLODIV = DIV\_2, FWDTEN = OFF  #pragma config POSCMOD = HS, FNOSC = FRCPLL, FPBDIV = DIV\_1  #pragma config FUSBIDIO = OFF //FUSBIDIO为端口控制  #pragma config JTAGEN = OFF //JTAG disable  // Period needed for timer 1 to trigger an interrupt every 0.1 second  // (48MHz PBCLK / 1 = 48000000KHz Timer 1 clock)  #define PERIOD 48000 //48000/48000000 = 0.001s = 0.2ms  #define BTN\_DELAY 2 //2\*5=10ms  #define SYS\_FREQ (48000000L)  #define PWM\_PR 0x0FFF  #define DUTYMAX 0x0DFF  UINT16 pwm1=0,pwm1\_d=0,pwm\_cnt=0;  unsigned int ADS\_cnt=0,ADS\_flag=1;  unsigned int adrst[16]={0},adptr=0;  //数码管显示数据 分别代表0~9,0.~9.FSEt-yno  unsigned char Led\_lib[] = {0x42, 0xf3, 0x86, 0xa2, 0x33, 0x2a, 0x0a, 0xf2, 0x02, 0x22, 0x40, 0xf1, 0x84, 0xa0, 0x31, 0x28, 0x08, 0xf0, 0x00, 0x20, 0x1e, 0x0e, 0x0f, 0xbf, 0x23, 0x9b, 0x8b}; //小LED字库  void **SpiInitDevice**() {  // 8 bits/char, input data sampled at end of data output time  SpiOpenFlags oFlags = SPI\_OPEN\_MSTEN | SPI\_OPEN\_CKP\_HIGH | SPI\_OPEN\_MODE8 | SPI\_OPEN\_ON;  PORTSetPinsDigitalOut(IOPORT\_B, BIT\_9);  PPSOutput(2, RPB8, SDO2); // Set RB8 pin as output for SDO2  // Open SPI module, use SPI channel 2, use flags set above, Divide Fpb by 6  SpiChnOpen(2, oFlags, 6);  }  void **SpiDoBurst**(unsigned char \*pBuff, unsigned char Len) {  if (pBuff) {  unsigned int i;  PORTClearBits(IOPORT\_B, BIT\_9);  for (i = 0; i < Len; i++) {  SpiChnPutC(2, pBuff[i]);  }  PORTSetBits(IOPORT\_B, BIT\_9);  }  }  void **AD10init**(void)  {  ANSELAbits.ANSA0 = 1; // PORTA.0选择为AN0模拟输入  AD1CON1 = 0x00E0; // 自动采样  AD1CHS = 0x00000000; // CH0和CH1均使用AN0通道  AD1CSSL = 0;  AD1CON3 = 0x0203; // 采样时间 = 2Tad  AD1CON2 = 0x6004; // 选择 VREF+ 和 VREF- 作为参考  // 采样2次后产生中断信号  AD1CON1bits.ADON = 1; // 开启AD  }  UINT16 **AD10Sample**(void)  {  AD1CON1bits.ASAM = 1; // 自动采样：31个Tad后自动转换  while (!AD1CON1bits.DONE); // 等待转换完成  AD1CON1bits.ASAM = 0; // 结束本次采样/转换操作  return ADC1BUF0; //返回采样结果  }  void **AD10DispRst**(UINT16 rst)  {  static BYTE spibuff[4];  spibuff[2] = Led\_lib[rst % 10]; //个位  spibuff[1] = Led\_lib[(rst / 10) % 10]; //十位  spibuff[0] = Led\_lib[(rst / 100) % 10]; //百位  spibuff[3] = Led\_lib[rst / 1000]; //千位  SpiDoBurst(spibuff,4);  }  void **Timer1Init**()  {  // Timer1@1ms  OpenTimer1(T1\_ON | T1\_SOURCE\_INT | T1\_PS\_1\_1, PERIOD);  // Set up the timer interrupt with a priority of 2  INTEnable(INT\_T1, INT\_ENABLED);  INTSetVectorPriority(INT\_TIMER\_1\_VECTOR, INT\_PRIORITY\_LEVEL\_2);  INTSetVectorSubPriority(INT\_TIMER\_1\_VECTOR, INT\_SUB\_PRIORITY\_LEVEL\_0);  }  // Configure the Timer 1 interrupt handler  void \_\_ISR(\_TIMER\_1\_VECTOR, ipl2) **Timer1Handler**(void)  {  // Clear the interrupt flag  INTClearFlag(INT\_T1);  ADS\_cnt++;  if(ADS\_cnt > 10) //0.01s  {  ADS\_cnt = 0;  ADS\_flag = 1;  }  pwm\_cnt++;  if(pwm\_cnt > 1)  {  pwm\_cnt = 0;  if(pwm1\_d == 0)  {  pwm1 ++;  if(pwm1 > DUTYMAX )  {  pwm1 = DUTYMAX;  pwm1\_d = 1;  }  }  else  {  if(pwm1 == 0)  {  pwm1 = 0;  pwm1\_d = 0;  }  else  pwm1 --;  }  OC1RS = pwm1;  }  }  void **PWMinit**()  {  //PWM引脚关联  RPB7Rbits.RPB7R = 0b0101; //PWM1  //PWM1初始化  OC1CON = 0x0000; // Turn off OC1 while doing setup.  OC1RS = pwm1; // Initialize secondary Compare Register  OC1CON = 0x0006; // Configure for PWM mode  //定时器2周期设定+开启  PR2 = PWM\_PR; // Set period  T2CONSET = 0x8000; // Enable Timer2  //PWM1开启  OC1CONSET = 0x8000; // Enable OC1  }  int **main**(void)  {  int i,ads;  SYSTEMConfig(SYS\_FREQ, SYS\_CFG\_WAIT\_STATES | SYS\_CFG\_PCACHE);  INTDisableInterrupts();  INTConfigureSystem(INT\_SYSTEM\_CONFIG\_MULT\_VECTOR);  SpiInitDevice();  AD10init();  PWMinit();  Timer1Init();  INTEnableInterrupts();  while(1)  {  if(ADS\_flag > 0)  {  ADS\_flag = 0;  adrst[adptr] = AD10Sample();  adptr++;  if(adptr > 15)  {  adptr = 0;  ads = 0;  for(i=0;i<16;i++)  ads += adrst[i];  ads = ads >> 4;  AD10DispRst(ads);  }  }  }  return 0;  } |