### 19.7 A/D例程

本节描述了在微芯PIC32MX220F032B型芯片上的AD输入程序示例。示例中将AD采样的原始值通过SPI控制的7段LED数码显示器显示出来，显示范围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19-1 A/D引脚和SPI引脚选择硬件配置表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 功能符号 | 引脚号 | 复用端口选择指定功能所用代码 | 说明 |
| 1 | AN10 | 25 | ANSELBbits.ANSB14 = 1 | PORTB.14，使能为模拟输入 |
| 2 | SCK2 | 26 | 由SPI模块自动选择(SCK2只能选这个引脚) | SPI数据时钟 |
| 3 | SDO2 | 17 | PPSOutput(2, RPB8, SDO2) | SPI数据输出 |
| 4 | SLCK | 18 | PORTSetPinsDigitalOut(IOPORT\_B, BIT\_9) | 外部移位寄存器数据锁存 |

在便携式实验板上A/D输入通道设计了4路，即AN1～AN4，可以从J14接口从板外输入模拟量，见图19-5所示，也可以由OC1～OC4（即PWM1～PWM4脉宽调制）输出可变占空比的PWM波形，PWM波形经过RC滤波后形成可变的模拟信号，由A/D采样即可得到模拟信号的幅值，通过SPI接口送到LED数码管上显示出来。同时AN1～AN4引脚也可以作为开关量的通用输入输出引脚使用，这种使用方式可以将K1～K4按键开关读入到单片机中，见图19-4所示。

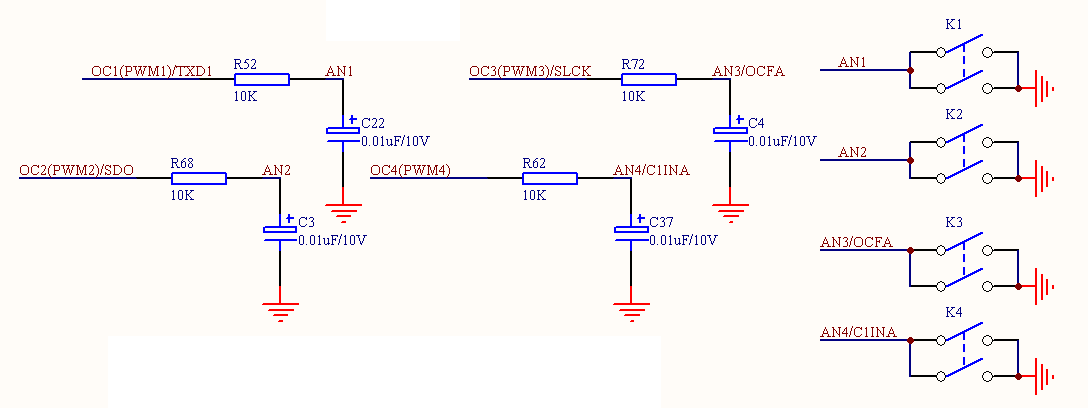


图19-4： A/D输入与按键接口复用电路图

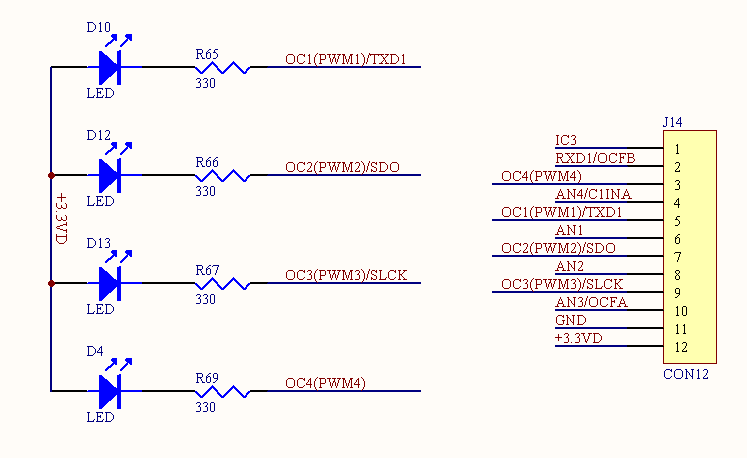


图19-5： A/D输入外接电路图

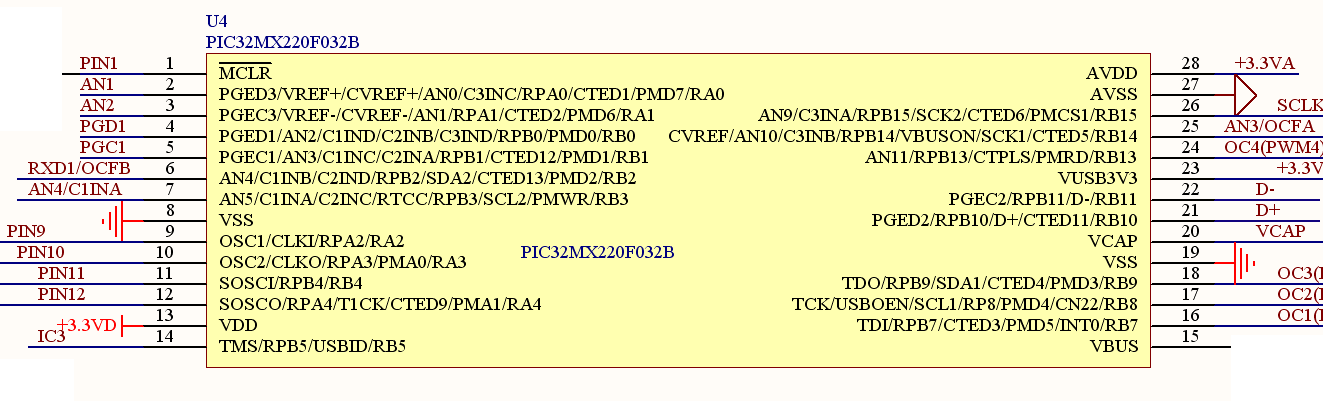


图19-6：A/D输入到芯片的引脚图

图19-7：主函数流程框图 图19-8：AD采样函数流程框图 图19-9：定时器中断函数流程框图

**1、主函数例程（程序流程框图见图19-7所示）**

|  |
| --- |
| int **main**(void)  {  SYSTEMConfig(SYS\_FREQ, SYS\_CFG\_WAIT\_STATES | SYS\_CFG\_PCACHE);  SpiInitDevice();  AD10init();  INTDisableInterrupts();  INTConfigureSystem(INT\_SYSTEM\_CONFIG\_MULT\_VECTOR);  Timer1Init();  INTEnableInterrupts();  while(1)  {  if(ADS\_flag > 0)  {  ADS\_flag = 0;  AD10DispRst(AD10Sample());  }  }  return 1;  } |

**2、AD采样函数例程（程序流程框图见图19-8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  } |

**3、定时器中断函数例程（程序流程框图见图19-9所示）**

|  |
| --- |
| void \_\_ISR(\_TIMER\_1\_VECTOR, ipl2) **Timer1Handler**(void)  {  // Clear the interrupt flag  INTClearFlag(INT\_T1);  ADS\_cnt++;  if(ADS\_cnt > 50) //0.05s  {  ADS\_cnt = 0;  ADS\_flag = 1;  }  } |

### 附件：代码

|  |
| --- |
| /\*  \* File: main.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  #pragma config CP = OFF  #pragma config DEBUG = ON  // 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 = 1ms  #define SYS\_FREQ (48000000L)  //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}; //LED字库  /\*-------LED段码分布图------  ---0---  | |  7 3  | |  ---6---  | |  5 2  | |  ---4--- 1  ----------------------------\*/  unsigned int ADS\_cnt=0,ADS\_flag=1;  unsigned int adrst=0;  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)  {  ANSELBbits.ANSB14 = 1; // PORTB.14选择为AN10模拟输入  AD1CON1 = 0x00E0; // 自动采样  AD1CHS = 0xA0A0000; // CH0和CH1均使用AN10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 > 50) //0.05s  {  ADS\_cnt = 0;  ADS\_flag = 1;  }  }  int **main**(void)  {  SYSTEMConfig(SYS\_FREQ, SYS\_CFG\_WAIT\_STATES | SYS\_CFG\_PCACHE);  SpiInitDevice();  AD10init();  INTDisableInterrupts();  INTConfigureSystem(INT\_SYSTEM\_CONFIG\_MULT\_VECTOR);  Timer1Init();  INTEnableInterrupts();  while(1)  {  if(ADS\_flag > 0)  {  ADS\_flag = 0;  AD10DispRst(AD10Sample());  }  }  return 1;  } |