#### 10.6 开关量输入按键例程

本节描述了在微芯PIC32MX220F032B型芯片上的数字IO输入示例。通过按键连接IO输入，用SPI主控输出8段数码管（LED）显示3个按键计数值，同时用一个LED显示秒定时计数。

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。

表10-3输入引脚硬件配置表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 功能描述 | 引脚号 | 端口复用选择指定功能 | 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 | RA0 | 2 | ANSELAbits.ANSA0 = 0 | PORTA.0，按键K1 |
| 5 | RA1 | 7 | ANSELBbits.ANSB3 = 0 | PORTB.3，按键K4 |

开关量输入按键电路电位变化中断编程，在硬件系统中，通过按键开通或闭合来产生高或低电平，从而实现控制信号的键入。如图10-4所示，采用4.7千欧的电阻与SW-PB限位开关串联的方案设计按键模块，当某按键断开的时候，其相应的输出信号（K1和K4）呈现出高电压；当按键闭合时，输出信号则呈现出低电压。其输出信号直接与PIC32MX输入输出端口相连，提供相应的控制信息。

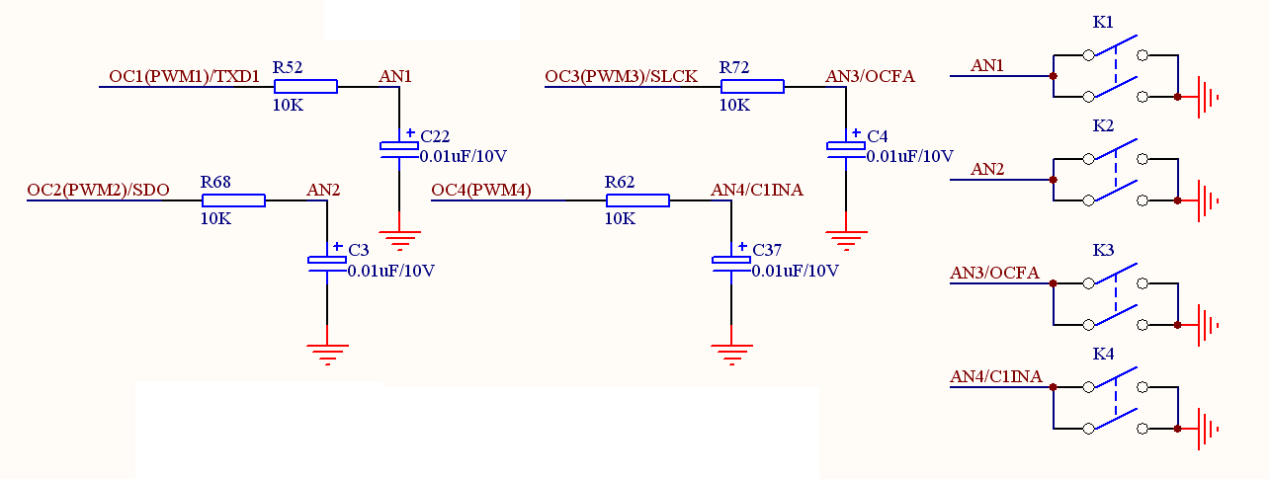


图10-4：按键电路

七段数码管显示模块如图10-5所示，采用PIC32MX的SPI口传送数据，并通过74HC595芯片驱动七段数码管进行显示。

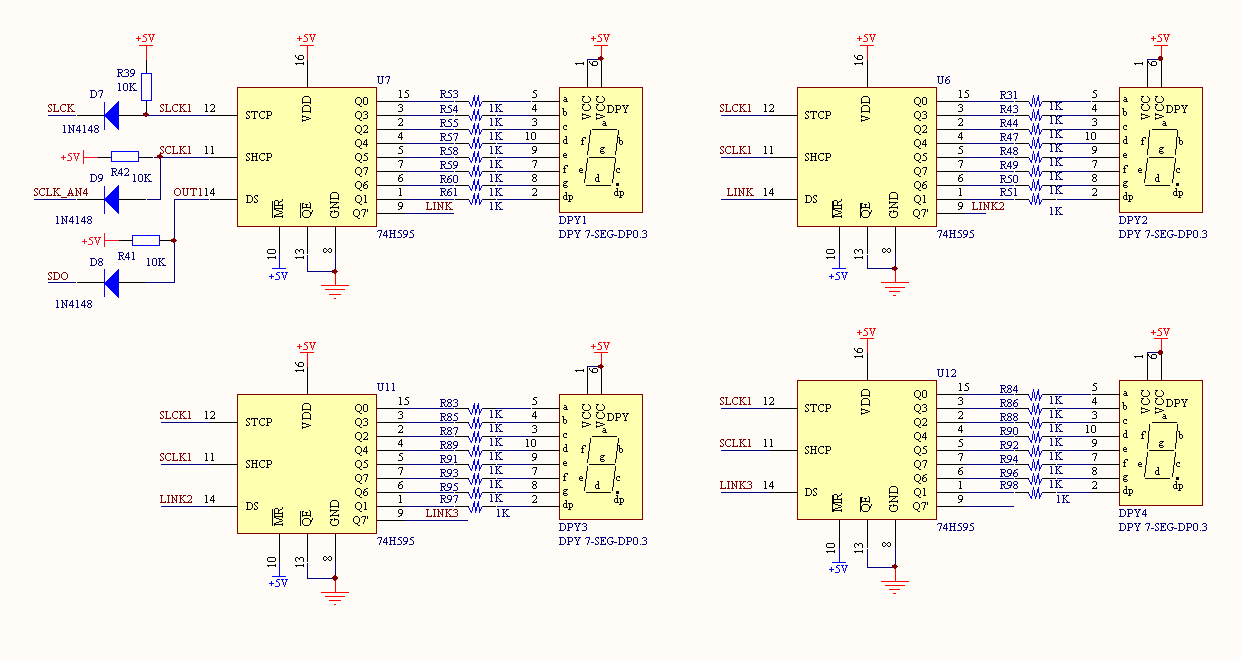


图10-5：3.3V输出电平转换到5V输入电平的转换电路及LED七段数码管驱动电路



图10-6：主函数流程框图

**1、主函数例程（程序流程框图见图10-6所示）**

|  |
| --- |
| int **main**(void)  {  int task=0;  SYSTEMConfig(SYS\_FREQ, SYS\_CFG\_WAIT\_STATES | SYS\_CFG\_PCACHE);  INTDisableInterrupts();  INTConfigureSystem(INT\_SYSTEM\_CONFIG\_MULT\_VECTOR);  SpiInitDevice();  BtnInit();  Timer1Init();  INTEnableInterrupts();  while(1)  {  switch(task)  {  case 0:  if(led\_flag > 0)  {  led\_flag = 0;  Led();  }  break;  case 1:  if(btn\_flag > 0)  {  btn\_flag = 0;  Button();  }  default:  break;  }  task ++;  if(task > 1) task = 0;  }  return 1;  } |



图10-7：数码管显示函数流程框图

**2、数码管显示函数例程（程序流程框图见图10-7所示）**

|  |
| --- |
| void **Led**()  {  static unsigned char ledBuff[4] = {0x00, 0x00, 0x00, 0x00};  static int led = 0,ledt=0;  int i;  SpiDoBurst(ledBuff, 4);  ledt ++;  if(ledt > 9)  {  ledt = 0;  led++;  if (led > 9) led = 0;  }  for (i = 0; i < 3; i++)  ledBuff[i] = Led\_lib[BtnCnt[i]];  ledBuff[3] = Led\_lib[led];  } |



图10-8：按键扫描函数流程框图

**3、按键扫描函数例程（程序流程框图见图10-8所示）**

|  |
| --- |
| void **Button**(void)  {  static int btn1=0,btn2=0,btn3=0;  if (PORTAbits.RA0 == 0)  {  btn1 ++;  if(btn1 == BTN\_DELAY)  {  BtnCnt[0]++;  if (BtnCnt[0] > 9)  BtnCnt[0] = 0;  }  }  else  btn1 = 0;  if (PORTBbits.RB3 == 0)  {  btn2++;  if(btn2 == BTN\_DELAY)  {  BtnCnt[1]++;  if (BtnCnt[1] > 9)  BtnCnt[1] = 0;  }  }  else  btn2 = 0;  } |



图10-9定时器中断函数流程框图

**4、定时器中断函数例程（程序流程框图见图10-9所示）**

|  |
| --- |
| void \_\_ISR(\_TIMER\_1\_VECTOR, ipl2) **Timer1Handler**(void)  {  // Clear the interrupt flag  INTClearFlag(INT\_T1);  led\_cnt++;  if(led\_cnt > 100) //0.1s  {  led\_cnt = 0;  led\_flag = 1;  }  btn\_cnt++;  if(btn\_cnt > 5) //5ms  {  btn\_cnt = 0;  btn\_flag = 1;  }  } |

### 附件：代码

|  |
| --- |
| /\*  \* File: SPIExample.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,FSOSCEN = OFF  #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 BTN\_DELAY 5 //1\*5=5ms  #define SYS\_FREQ (48000000L)  unsigned int led\_cnt=0,btn\_cnt=0,led\_flag=1,BtnCnt[]={0,0,0},btn\_flag=0;  //8段LED数码管字库：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  /\*-------LED段码分布图------  ---0---  | |  7 3  | |  ---6---  | |  5 2  | |  ---4--- 1  ----------------------------\*/  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 **Led**()  {  static unsigned char ledBuff[4] = {0x00, 0x00, 0x00, 0x00};  static int led = 0,ledt=0;  int i;  SpiDoBurst(ledBuff, 4);  ledt ++;  if(ledt > 9)  {  ledt = 0;  led++;  if (led > 9) led = 0;  }  for (i = 0; i < 3; i++)  ledBuff[i] = Led\_lib[BtnCnt[i]];  ledBuff[3] = Led\_lib[led];  }  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);  }  void \_\_ISR(\_TIMER\_1\_VECTOR, ipl2) **Timer1Handler**(void)  {  // Clear the interrupt flag  INTClearFlag(INT\_T1);  led\_cnt++;  if(led\_cnt > 100) //0.1s  {  led\_cnt = 0;  led\_flag = 1;  }  btn\_cnt++;  if(btn\_cnt > 5) //5ms  {  btn\_cnt = 0;  btn\_flag = 1;  }  }  void **BtnInit**()  {  ANSELAbits.ANSA0 = 0;  ANSELBbits.ANSB3 = 0;  }  void **Button**(void)  {  static int btn1=0,btn2=0,btn3=0;  if (PORTAbits.RA0 == 0)  {  btn1 ++;  if(btn1 == BTN\_DELAY)  {  BtnCnt[0]++;  if (BtnCnt[0] > 9)  BtnCnt[0] = 0;  }  }  else  btn1 = 0;  if (PORTBbits.RB3 == 0)  {  btn2++;  if(btn2 == BTN\_DELAY)  {  BtnCnt[1]++;  if (BtnCnt[1] > 9)  BtnCnt[1] = 0;  }  }  else  btn2 = 0;  }  int **main**(void)  {  int task=0;  SYSTEMConfig(SYS\_FREQ, SYS\_CFG\_WAIT\_STATES | SYS\_CFG\_PCACHE);  INTDisableInterrupts();  INTConfigureSystem(INT\_SYSTEM\_CONFIG\_MULT\_VECTOR);  SpiInitDevice();  BtnInit();  Timer1Init();  INTEnableInterrupts();  while(1)  {  switch(task)  {  case 0:  if(led\_flag > 0)  {  led\_flag = 0;  Led();  }  break;  case 1:  if(btn\_flag > 0)  {  btn\_flag = 0;  Button();  }  default:  break;  }  task ++;  if(task > 1) task = 0;  }  return 1;  } |