### 11.5 定时器定时应用例程

本节描述了在微芯PIC32MX220F032B型芯片上的定时器的示例代码。代码中，实现了秒表功能，通过两个按钮控制秒表的启动\停止、复位，通过SPI主控的LED数码管显示秒表值，秒表计时范围0~999.9秒。

### 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11-1 引脚选择硬件配置表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | SPI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 | PA.0，按钮：启动\暂停 |
| 5 | RA1 | 3 | ANSELAbits.ANSA1 = 0 | PA.1，按钮：复位 |

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11-4所示，采用10千欧的电阻与K1～K4开关串联的方案设计按键模块，当某按键断开的时候，其相应的输出信号呈现出高电平；当按键闭合时，呈现出低电平。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。

本示例中使用了K1和K2两个按钮，其中K1为秒表的“启动\暂停”，K2为秒表的“复位”。

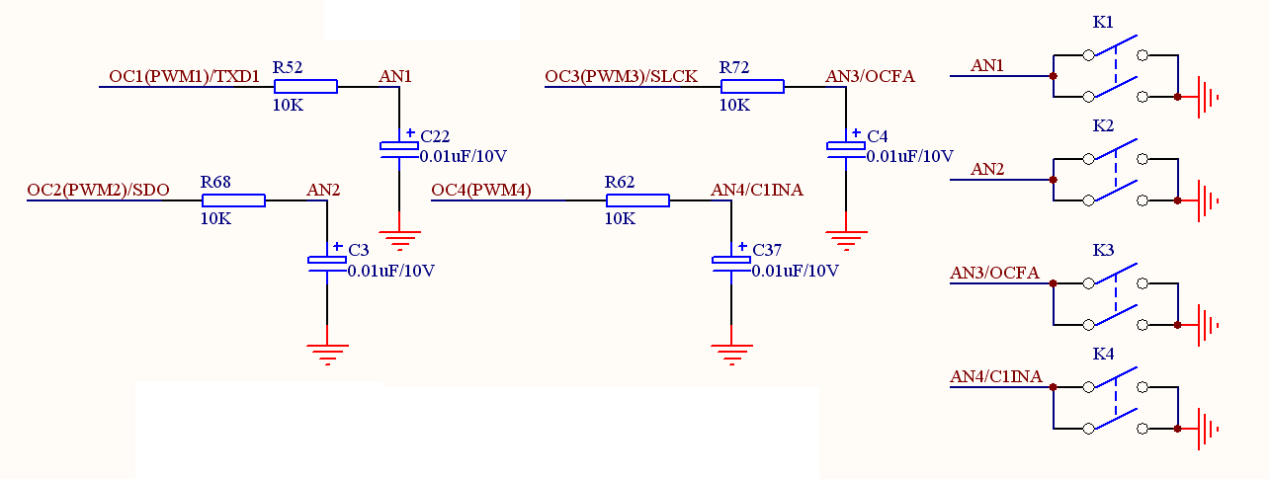


图11-4： 按键模块

七段数码管显示模块如图11-5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。

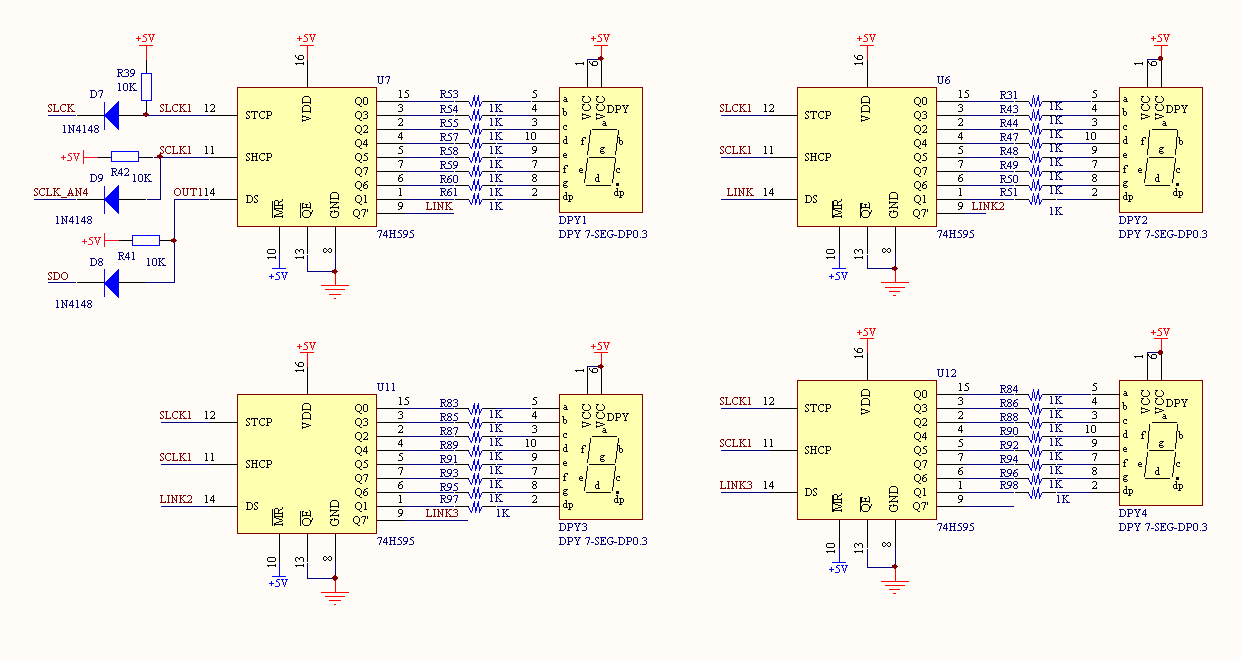


图11-5：3.3V输出电平转换到5V输入电平的转换电路及LED七段数码管驱动电路



图11-6：主函数流程框图

**1、主函数例程（程序流程框图见11-6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;  ButtonScan();  }  default:  break;  }  task ++;  if(task > 1) task = 0;  }  return 1;  } |



图11-7：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11-7所示）**

|  |
| --- |
| void Led()  {  unsigned char ledBuff[4] = {0x00, 0x00, 0x00, 0x00};  static unsigned char ledcnt[4]={0x00, 0x0A, 0x00, 0x00};  int i;  switch(op)  {  case null:  case reset:  for(i=0;i<4;i++)  {  ledcnt[i] = 0;  }  ledcnt[1] = 10;  break;  case start:  ledcnt[2] ++;  if(ledcnt[2] > 9)  {  ledcnt[2] = 0;  ledcnt[1] ++;  if(ledcnt[1] > 19)  {  ledcnt[1] = 10;  ledcnt[0] ++;  if(ledcnt[0] > 9)  {  ledcnt[0] = 0;  ledcnt[3] ++;  if(ledcnt[3] > 9)  {  ledcnt[3] = 0;  }  }  }  }  break;  case pause:  break;  default:  break;  }  for (i = 0; i < 4; i++)  ledBuff[i] = Led\_lib[ledcnt[i]];  SpiDoBurst(ledBuff, 4);  } |



图11-8：定时器中断函数流程框图

**3、定时器中断函数例程（程序流程框图见11-8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: Timer.c  \* 定时器上实现的秒表程序：计时范围0~999.9秒  \* 按钮功能定义：  \* Button1：启动/暂停  \* Button2：复位  \*/  #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 = OFF, 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 BTN\_DELAY 5 //1\*5=5ms  #define SYS\_FREQ (48000000L)  typedef enum \_OP{  null,  start,  pause,  reset  }OP;  unsigned int led\_cnt=0,btn\_cnt=0,led\_flag=1,btn\_flag=0;  OP op=null;  //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**()  {  unsigned char ledBuff[4] = {0x00, 0x00, 0x00, 0x00};  static unsigned char ledcnt[4]={0x00, 0x0A, 0x00, 0x00};  int i;  switch(op)  {  case null:  case reset:  for(i=0;i<4;i++)  {  ledcnt[i] = 0;  }  ledcnt[1] = 10;  break;  case start:  ledcnt[2] ++;  if(ledcnt[2] > 9)  {  ledcnt[2] = 0;  ledcnt[1] ++;  if(ledcnt[1] > 19)  {  ledcnt[1] = 10;  ledcnt[0] ++;  if(ledcnt[0] > 9)  {  ledcnt[0] = 0;  ledcnt[3] ++;  if(ledcnt[3] > 9)  {  ledcnt[3] = 0;  }  }  }  }  break;  case pause:  break;  default:  break;  }  for (i = 0; i < 4; i++)  ledBuff[i] = Led\_lib[ledcnt[i]];  SpiDoBurst(ledBuff, 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);  }  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; //Button1  ANSELAbits.ANSA1 = 0; //Button2  }  void ButtonScan(void)  {  static int btn1=0,btn2=0,btn3=0;  if(PORTAbits.RA0 == 0)  {  btn1 ++;  if(btn1 == BTN\_DELAY) //Button1 Pressed  {  switch(op)  {  case null:  case pause:  case reset:  op = start;  break;  case start:  op = pause;  break;  default:  break;  }  }  }  else  btn1 = 0;  if(PORTAbits.RA1 == 0) //Button2 Pressed  {  btn2++;  if(btn2 == BTN\_DELAY)  {  op = reset;  }  }  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;  ButtonScan();  }  default:  break;  }  task ++;  if(task > 1) task = 0;  }  return 1;  } |