3.22 国赛2015 任务二

2017年3月23日 15:51

1. 所遇问题

- 题3要求按下LED模块的SW1按键后，转换为手动控制模式，即按下SW1按键后转换D5灯的亮灭状态，不再受接收到的AD值控制。这看起来很好写，不过给出的部分源码中，接收无线信息处理部分(sapi.c)和LED模块(Enddev2.c)是分开来的，我们就需要在这两个文件中共用一个全局变量去存储控制模式。一开始我一直在找Enddev2.c的头文件，发现找不到就想着自己写一个，但是换个思考方式，为啥不能把这个全局变量定义写在sapi.c中，然后在sapi.h的声明它呢，这样我们一样可以在Enddev2.c中使用这个变量啊。如果对C语言中两个文件共用一个全局变量不熟，去学习一下extern。参考代码见下：

sapi.c | sapi.h | ZMain.c | Coord1.c | Enddev1.c | Enddev2.c | sensor.c

```
743 * @param   task_id  - The OSAL assigned task ID.
744 * @param   events - events to process.  This is a bit map
745 *                   contain more than one event.
746 *
747 * @return  none
748 */
749 unsigned char flag = 0; //1为自动控制
750 unsigned char flag1 = 0; //接收消息次数
751 uint8 hhhh[12] = {0XFE,0X00,0X00,0X00,0X00,0X00,0X00,0X00,
752 UINT16 SAPI_ProcessEvent( byte task_id, UINT16 events )
753 {
```

sapi.c | sapi.h | ZMain.c | Coord1.c | Enddev1.c | Enddev2.c | sensor.c

```
427
428 extern unsigned char flag;
```

sapi.c | sapi.h | ZMain.c | Coord1.c | Enddev1.c | Enddev2.c | sensor.c

```
627 }
628
629 #pragma vector = P1INT_VECTOR
630 __interrupt void P1INT(void)
631 {
632   if(P1IFG & 0x04)
633   {
634     flag = 1; //关闭自动控制  flag来自sapi
635     P1_3 = !P1_3; //反转LED灯状态
636     P1IFG &= ~0x04;
637   }
638   P1IF = 0;
639 }
640
```

- 题3中还要求要求要将LED节点模块D5灯状态每5秒发送至串口，一开始我傻了没想到可以直接判断P1_3的值即可，弄得最初的代码是通过AD值大小去判断D5灯状态，太傻了。
- 一开始我将接收到的AD值在判断是否为自动控制模式下才赋值给数组，这样是不对的，应该只要在接收到光电传感器的信息之后就应该赋值。

2. 参考代码

a. 题3

i. Coord1

```
652 /*********************************************************************************
653  * @fn          channel_panid_init
654  *
655  * @brief       对信道和PANID判断并设置
656  *
657  * @param       none
658  *
659  * @return      none
660  */
661 void ChannelPanidInit (void)
662 {
663   /* user code start */
664   uint8 panid[2];
665   uint8 channel = 11;
666   zb_Readpandid(panid);
667   if(panid[0] == 0x87 && panid[1] == 0x80 && channel == zb_Readchannel())
668   {
669     panid[0] = 0x87;
670     panid[1] = 0x80;
671     zb_Writepandid(panid);
672     zb_Writechannel(channel);
673     zb_SystemReset();
674   }
675   /* user code end */
676 }
677 /*********************************************************************************
```

ii. Enddev1(省略了组网代码，见上)

```
601 static void sendDummyReport(void)
602 {
603   /* user code start */
604   uint8 buf[2];
605   uint16 ad;
606   ad = get_guangdian_ad();
607   buf[0] = '1';
608   buf[1] = ad;
609   zb_SendZigbeeDatas(buf,2);
610   /* user code end*/
611 }
```

iii. Enddev2(省略了组网代码，见上)

```
600 static void sendDummyReport(void)
601 {
602   /* user code start */
603   uint8 buf[2];
604   buf[0] = 'L';
605   if(P1_3 == 1) //如果灯亮则发送01
606   {
607     buf[1] = 0x01;
608     zb_SendZigbeeDatas(buf,2);
609   }
610   else
611   {
612     buf[1] = 0x00;
613     zb_SendZigbeeDatas(buf,2);
614   }
615   /* user code end*/
616 }
617 void initP1(void);  //必须要声明
618 __interrupt void P1INT(void); //必须要声明
619
620 void initP1(void) //P1中断初始化
621 {
622    IEN2 |= 0x10;
623    P1IEN |= 0x04;
624    PICTL |= 0x02;
625    EA = 1;
626 }
627
628 #pragma vector = P1INT_VECTOR
629 __interrupt void P1INT(void)
630 {
631   if(P1IFG & 0x04)
632   {
633     flag = 1; //关闭自动控制  flag来自sapi
634     P1_3 = !P1_3; //反转LED灯状态
635     P1IFG &= ~0x04;
636   }
637   P1IF = 0;
638 }
```

iv. Sapi

```
749 unsigned char flag = 0; //1为自动控制
750 unsigned char flag1 = 0; //接收消息次数
751 uint8 hhhh[12] = {0XFE,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0XFF}; //全局数组用来存储发送串口的数据
752 UINT16 SAPI_ProcessEvent( byte task_id, UINT16 events )
753 {
754   osal_event_hdr_t *pMsg;
```

```
775         case AF_INCOMING_MSG_CMD:
776           pMSGpkt = (afIncomingMSGPacket_t *) pMsg;
777           SAPI_ReceiveDataIndication( pMSGpkt->srcAddr.addr.shortAddr, pMSGpkt->clusterId,
778                                       pMSGpkt->cmd.DataLength, pMSGpkt->cmd.Data);
779           /* user code start */
780           uint8 inf[2];
781           osal_memcpy(inf,pMSGpkt->cmd.Data,2);
782           if(flag == 0 && inf[0] == '1') //flag为0时是自动控制
783           {
784             if(inf[1] < 0x09)
785               P1_3 = 1;
786             else
787               P1_3 = 0;
788           }
789           else if(inf[0] == '1')
790              hhhh[1] = inf[1]; //当接收到了光电发送的数据则将AD值赋值给数组
791
792           if(inf[0] == 'L')
793               hhhh[3] = inf[1];  //将接收到的LED状态赋值给数组
794
795           if(++flag1 == 2)  //接收到两次消息则发送数据至串口
796           {
797             HalUARTWrite(HAL_UART_PORT_0, hhhh, 12);
798             flag1 = 0;
799           }
800           /* user code end */
```