**综合实验报告**

一、实验目的

1. 掌握 µCOS-II 中任务管理的基本方法，熟练使用 µCOS-II 任务管理基本系统调用。
2. 掌握uC/OS-II操作系统下使用信号量解决任务之间的同步互斥问题。
3. 掌握嵌入式实时操作系统 µC/OS 中时间中断的使用情况。
4. 掌握uC/OS-II操作系统下任务间通讯的方法。
5. 加强综合处理问题的能力。

二、实验内容

结合内容1和内容2，修改程序，创建三个任务，两个按键任务一个响应任务。有键按下即发送消息邮箱。任一消息均控制响应任务运行。两个按键任务中任意一个按键按下，则发送一个消息邮箱给响应任务。若按下是数字（0-9），则在串口打印输出，并在数码管显示；如为非数字（其他），则分别对应LED灯的闪烁、步进电机的正转，反转，停止、加速和减速。

三、实验步骤

1. 参考Eg\_kbd.apj，Eg\_stepper.apj，Eg\_timer.apj，了解如何通过按键控制LED的闪烁和对步进电机的控制。
2. 参考Eg3.aws了解如何通过邮箱通信控制两个任务之间的通信。
3. 修改程序，使一个任务中能响应按键，使数字能在串口打印；并能控制LED灯的闪烁和对步进电机的控制；同时发数据到邮箱。另一个任务通过OSMboxPend()函数等待一个邮箱中的消息，如果邮箱中没有可用的消息，调用任务就被挂起，直到邮箱中有了消息或者等待超时即执行User\_SEG\_Blink()把按键值在数码管上显示出来。

四、实验核心代码（如是在已有程序上的修改，直接写出修改部分程序）。

开头定义：

int seg=0;

OS\_EVENT \*Mbox1;

在void User\_SEG\_Blink(void)中：

\*((unsigned char \*)0x10000004) = seg\_value[**seg**];

在void TaskLED(void \*Id)中：

char Msg[100];

INT8U err;

int nCount = 0;

…………

for (;;) {

INT8U ch; //相应按键的值

ch = Key\_GetKey();

if(ch == 0)

continue;

switch( ch )

{

case '1':

uHALr\_printf("\r1");

seg=1;

sprintf(Msg, "TaskSEG %d", nCount++); //发送数据到邮箱

OSMboxPost(Mbox1, Msg);

break;

…………

case 'A':

uHALr\_printf("\rA\n");

User\_LED\_Blink();

seg=10;

sprintf(Msg, "TaskSEG %d", nCount++);

OSMboxPost(Mbox1, Msg);

break;

case 'C': // 加速

uHALr\_printf("\r加速");

DRVStepperSpeedUp();

seg=12;

sprintf(Msg, "TaskSEG %d", nCount++);

OSMboxPost(Mbox1, Msg);

break;

case 'D': // 减速

uHALr\_printf("\r减速");

DRVStepperSpeedDown();

seg=13;

sprintf(Msg, "TaskSEG %d", nCount++);

OSMboxPost(Mbox1, Msg);

break;

case 'E': // 正反转控制

if(direct == STEP\_MOTOR\_CLOCKWISE)

{

uHALr\_printf("\r正转");

direct = STEP\_MOTOR\_ANTICLOCKWISE;

seg=14;

sprintf(Msg, "TaskSEG %d", nCount++);

OSMboxPost(Mbox1, Msg);

}

else

{

uHALr\_printf("\r反转");

direct = STEP\_MOTOR\_CLOCKWISE;

seg=14;

sprintf(Msg, "TaskSEG %d", nCount++);

OSMboxPost(Mbox1, Msg);

}

DRVStepperSetDirect(direct);

break;

case 'F': //使能控制

if(benable == STEP\_MOTOR\_ENABLE)

{

uHALr\_printf("\r停止");

benable = STEP\_MOTOR\_DISABLE;

seg=15;

sprintf(Msg, "TaskSEG %d", nCount++);

OSMboxPost(Mbox1, Msg);

}

else

{

uHALr\_printf("\r启动");

benable = STEP\_MOTOR\_ENABLE;

seg=15;

sprintf(Msg, "TaskSEG %d", nCount++);

OSMboxPost(Mbox1, Msg);

}

DRVStepperControl(benable);

break;

void TaskSEG(void \*Id)

{

char \*Msg;

INT8U err;

for (;;) {

Msg = (char \*)OSMboxPend(Mbox1, 0, &err);

uHALr\_printf("Task2() called\n")：

OSSchedLock();

sprintf(print\_buf, "Task%c() turned\n", \*(char \*)Id);

uHALr\_printf(print\_buf);

User\_SEG\_Blink();

OSSchedUnlock();

OSTimeDly(10)；

}

}

void Main(void)

Mbox1 = OSMboxCreate((void \*)0);

五、实验结果及分析

一个任务中能响应按键，使数字能在串口打印并把按键对应值赋给seg；通过case语句控制LED灯的闪烁和对步进电机的控制；同时发数据到邮箱。另一个任务通过OSMboxPend()函数等待一个邮箱中的消息，如果邮箱中没有可用的消息，调用任务就被挂起，直到邮箱中有了消息或者等待超时即执行User\_SEG\_Blink()通过在另一个任务里的赋值的seg查阅数码管显示数组把按键值在数码管上显示出来。