# 使用信号量实现按键轮询检测

RealTouch 评估板 RT-Thread 入门文档

版本号：1.0.0

日期：2012/8/12

修订记录

| 日期 | 作者 | 修订历史 |
| --- | --- | --- |
| 2012/8/12 | prife | 创建文档 |

# 实验目的

使用了key.c 驱动，所以与程序与原文不同

- 了解信号量用作多个线程进行同步的使用方式
- 掌握轮询实现按键检测的用法
- 熟悉使用 RT-Thread 中信号量相关 API

# 硬件说明

本实验使用 RT-Thread 官方的 Realtouch 开发板作为实验平台。涉及到的硬件主要为：

- 串口 3，作为 rt_kprintf 输出。
- 按键 IO 口

具体请参见《Realtouch 开发板使用手册》

# 实验原理及程序结构

## 实验设计

本实验是在 RT-Thread 中通过轮询方式按键的检测功能。创建一个线程，以一定的时间间隔周期运行，扫描硬件系统中的物理按键，当检测到有按键按下时，发布一个按键信号量。创建另外一个线程来使用按键，为了简单起见，它将通过获取按键信号量，当取得这个信号量时，将按键信息打印出到串口。

## 源程序说明

本实验对应 kernel_sem_keypolling。

### 系统依赖

在 rtconfig.h 中需要开启

- #define RT_USING_SEMAPHORE
  此项必选，选择此项后才可以使用信号量相关 API。
- #define RT_USING_HEAP
  此项可选，开启此项可以创建动态信号量，如果使用静态信号量，则此项不是必要的
- #define RT_USING_CONSOLE

此项必须，本实验使用 rt_kpriintf 向串口打印按键信息，因此需要开启此项

## 主程序说明

在 applications/application.c 中，在系统 init 线程中完成按键 GPIO 初始化。参考原理图可知，这些按键已经在开发板上已经下拉模式，因此所有按键 GPIO 口都设置为悬空模式，这样当有键按下时，对应的 GPIO 口的电平就会为高，相关代码如下。

初始化按键 GPIO 口

```
#include "stm32f4xx.h"
#define KEY_PORT  GPIOF
#define KEY_PIN   (GPIO_Pin_6 | GPIO_Pin_7 | GPIO_Pin_8 | GPIO_Pin_9 | GPIO_Pin_10| GPIO_Pin_11)
#define KEY_CLCOK RCC_AHB1Periph_GPIOF

void rt_init_thread_entry(void* parameter)
{
//gpio init
    GPIO_InitTypeDef  GPIO_InitStructure;

    /* GPIOA GPIOB Periph clock enable */
    RCC_AHB1PeriphClockCmd(KEY_CLCOK , ENABLE);

    /* Configure Pin in input pushpull mode */
    GPIO_InitStructure.GPIO_Pin = KEY_PIN;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;//GPIO_Mode_OUT;
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
    GPIO_Init(KEY_PORT, &GPIO_InitStructure);
}
```

用Key_Init （） 函数代替

在 int rt_application_init()中初始化静态线程，名为“keyp”的 thread1。

初始化线程代码

```
    rt_thread_init(&thread1,
```

```
                "keyp", //producer
                rt_thread_entry1,
                RT_NULL,
                &thread1_stack[0],
                sizeof(thread1_stack),11,25);
rt_thread_startup(&thread1);
```

创建信号量代码如下所示。

创建信号量代码

```
static int key;          static Key_Def key;
static struct rt_semaphore sem;
int rt_application_init()
{
    rt_err_t result;

    result = rt_sem_init(&sem, "sem", 0, RT_IPC_FLAG_FIFO);
    f (result != RT_EOK)
    {
        rt_kprintf("error, init sem failed!\n");
        return 0;
    }
```

其处理线程处理函数如下，此线程周期启动，100ms 启动一次检测按键，当检测到按键按下后，延时 20ms 后再次检测按键以实现按键去抖。如果的确检测到按键，则发布信号量。

线程 1 代码

```
ALIGN(RT_ALIGN_SIZE)
static char thread1_stack[1024];
struct rt_thread thread1;

static void rt_thread_entry1(void* parameter)
{
    int temp;

    while (1)
    {                    key = KEY_Scan();
       key = GPIO_ReadInputData(KEY_PORT);
```

```
if (key!=KEYNULL)
{
    rt_thread_delay(RT_TICK_PER_SECOND / 50);
    rt_sem_release(&sem);
}
```

```
        if (key & KEY_PIN)
        {
            temp = key;
            rt_thread_delay(RT_TICK_PER_SECOND / 50);
            key = GPIO_ReadInputData(KEY_PORT);
            if (key == temp)
                rt_sem_release(&sem);
        }
        rt_thread_delay(RT_TICK_PER_SECOND/10);
    }
}
```

接下来在创建线程 2，通过获取按键信号量，来获得线程 1 中得到的按键，并将按键信息通过串口发送的 PC 机上。

线程 2 代码

```
ALIGN(RT_ALIGN_SIZE)
static char thread2_stack[1024];
struct rt_thread thread2;

static int key;
static void rt_thread_entry2(void* parameter)
{
    while (1)
    {
        rt_sem_take(&sem, RT_WAITING_FOREVER);
        if (key & KEY_PIN)
        {
            rt_kprintf("some keys has been pressed : %x\n", key);
        }
    }
}
```

# 编译调试及观察输出信息

编译请参见《RT-Thread 配置开发环境指南》完成编译烧录，参考《Realtouch 开发板使用手册》完成硬件连接，连接扩展板上的串口和 jlink。

运行后可以看到如下信息：

串口信息

```
 \ | /
- RT -     Thread Operating System
 / | \     1.1.0 build Aug 13 2012
 2006 - 2012 Copyright by rt-thread team
```

此时按下按键 S2 至 S7，即可看到如下信息。

串口信息

```
some keys has been pressed : 40
some keys has been pressed : 80
some keys has been pressed : 100
some keys has been pressed : 200
some keys has been pressed : 400
some keys has been pressed : 800
```

# 结果分析

整个程序运行过程中各个线程的状态变化：

Thread2 获取按键信号量，因为此时信号量中值为 0，Thread2 会被挂起。Thread1 周期检测按键，当检测到按键时，Thread1 发布信号量，此时有两种情况：

如果 Thread2 的优先级比 Thread1 的优先级高，此时 Thread1 会立刻被唤醒并执行，在其线程处理函数中，将按键信息发送至串口，while 循环，继续执行 rt_sem_take，此时信号量的值已经为 0，Thread1 再次被挂起。此时系统中优先级最高的线程为 Thread2，因此内核将调度此线程继续周期按键检测功能。

如果 Thread2 的优先级比 Thread1 的优先级高，Thread1 会从挂起态回到就绪态，但偶需 Thread2 优先级更高，因此 Thread2 继续运行，如果没有按键按下，Thread1 会调用 rt_threa_delay，Thread1 挂起，此时 Thread2 为系统优先级最高的线程，会被立即调度执行，在线程处理函数中打印 Thread1 获取到的按键信息。

# 总结

本实验的原理是使用信号量实现两个线程之间的同步。

本实验使用静态信号量以及静态线程来进行按键轮询检测，读者可以改成动态信号量以及动态线程来实现同样的功能。