定时器超时

RealTouch 评估板 RT-Thread 入门文档

版本号：1.0.0

日期：2012/8/14

修订记录

日期	作者	修订历史
2012/8/14	bloom5	创建文档

实验目的

- 熟悉了解定时器超时的应用场景

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，需要连接 JTAG 扩展板

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

本实验的主要设计目的是帮助读者了解定时器超时的应用场景,具体是消息队列，关于消息队列内容，可以回看第二章。请读者注意，本实验本身不具有实际的工程参考价值，只是帮助读者快速了解相关 API 的用法。

源程序说明

本实验对应 1_kernel_timer_timeout

系统依赖

在 rtconfig.h 中需要开启

- #define RT_USING_HEAP

此项可选，开启此项可以创建动态线程和动态信号量，如果使用静态线程和静态信号量，则此项不是必要的

- #define RT_USING_CONSOLE

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

- #define RT_USING_TIMER_SOFT

此项必须，本实验使用软件定时器，因此必须开启此项

- #define RT_USING_MESSAGEQUEUE

此项必须，本实验使用了消息队列进行信息传递，因此必须打开此项

主程序说明

这个程序会创建 1 个动态线程，这个线程会从消息队列中收取消息；另外通过定时器超时函数向消息队列发送消息

全局变量

```
static rt_thread_t tid = RT_NULL;

/* 消息队列控制块 */
static struct rt_messagequeue mq;
/* 消息队列中用到的放置消息的内存池 */
static char msg_pool[2048];

/* 定时器的控制块 */
static struct rt_timer timer;
/*消息的参数，确定消息的顺序*/
static rt_uint16_t no = 0;
```

在初始化函数 rt_application_init()中，初始化了消息队列，并创建了一个动态线程。

初始化函数

```
 rt_err_t result;

    /* 初始化消息队列 */
    result = rt_mq_init(&mq, "mqt",
        &msg_pool[0], /* 内存池指向 msg_pool */
        128 - sizeof(void*), /* 每个消息的大小是 128 - void* */
        sizeof(msg_pool), /* 内存池的大小是 msg_pool 的大小 */
        RT_IPC_FLAG_FIFO); /* 如果有多个线程等待，按照先来先得到的方法分配消息 */

    if (result != RT_EOK)
    {
        rt_kprintf("init message queue failed.\n");
        return -1;
    }

    /* 创建线程 */
    tid = rt_thread_create("t",
        thread_entry, RT_NULL, /* 线程入口是 thread_entry, 入口参数是 RT_NULL */
        1024, 8, 50);
    if (tid != RT_NULL)
        rt_thread_startup(tid);
```

```
    return 0;
```

在线程入口函数 thread_entry()中我们初始化了定时器，定时时间为 1000 os tick，在开启定时器后就准备就收消息队列中的消息，等待没有超时失败，为一直等待直到消息队列中有消息。

```
  char buf[64];
    rt_err_t result;

    /* 初始化定时器 */
    rt_timer_init(&timer, "timer",  /* 定时器名字是 timer1 */
        timer_timeout, /* 超时时回调的处理函数 */
        RT_NULL, /* 超时函数的入口参数 */
        1000, /* 定时长度，以 OS Tick 为单位，即 1000 个 OS Tick */
        RT_TIMER_FLAG_PERIODIC); /* 周期性定时器 */

    rt_timer_start(&timer);

    while (1)
    {
        rt_memset(&buf[0], 0, sizeof(buf));

        /* 从消息队列中接收消息 */
        result = rt_mq_recv(&mq, &buf[0], sizeof(buf),
RT_WAITING_FOREVER);
        if (result == RT_EOK)
        {
            rt_kprintf("recv msg: %s\n", buf);
        }
        else if (result == -RT_ETIMEOUT)
        {
            rt_kprintf("recv msg timeout\n");
        }
    }
```

在超时函数 timer_timeout()中，让它超时即向消息队列中发送一个消息。

```
  char buf[32];
    rt_uint32_t length;

    length = rt_snprintf(buf, sizeof(buf), "message %d", no++);
    rt_mq_send(&mq, &buf[0], length);
```

上面代码中出现的 snprintf()，可以将可变参数按照制定的格式，转化为字符串形式，存在 buffer 中。

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：

```
 \ | /
- RT -     Thread Operating System
 / | \     1.1.0 build Aug 14 2012
 2006 - 2012 Copyright by rt-thread team
recv msg: message 0
recv msg: message 1
recv msg: message 2
recv msg: message 3
recv msg: message 4
recv msg: message 5
recv msg: message 6
recv msg: message 7
recv msg: message 8
recv msg: message 9
recv msg: message 10
recv msg: message 11
recv msg: message 12
recv msg: message 13
recv msg: message 14
recv msg: message 15
...
```

结果分析

有结果很好发现，定时器每 1000 os tick 超时一次，在超时函数中发送消息给消息队列，而线程则是从消息队列中获取消息将之打印出来。如果将 rT_mq_receive()函数的最后一个参数由 RT_WAITING_FOREVER 改为小于 1000 的数，那么你会发现出现

```
recv msg timeout
```

定时器设定为1000才会向消息队列中发送一个消息，而接收超时设定小于1000的话，必然会有接收超时到的时候定时器的超时函数还未向消息队列发送消息的情况。所以在发送和接收消息的时间上可能需要注意一下。