Test 2B 事件触发  
背景知识提要：  
1、 EVENT的基本知识  
typedef struct {  
        bool bAutoReset;  
        bool bIsSet; //事件当前状态  
    } event\_t;  
2、 Event的基本操作  
a) Set： SET\_EVENT (\_\_EVENT)  
b) Wait:  WAIT\_EVENT (\_\_EVENT), 返回值是bool，true表示检测到event被set了  
c) Reset:  RESET\_EVENT (\_\_EVENT);  
d) Init: INIT\_EVENT(\_\_EVENT, \_\_INIT\_VALUE,\_\_MANUAL)

我想，先想先思考几个关键问题：

到底什么是事件触发呢

一个事件有两个属性：bAutoReset和bIsSet

bool bAutoReset; //事件本身属性-事件自己自动触发，还是要别的事件来触发  
bool bIsSet; //事件当前状态

判断event\_t.bIsSet是不是true，即是不是已经被Set，，但这个bAutoReset是干什么用的呢，是不是。

对事件的基本操作----用宏来封装，

#define SET\_EVENT(\_\_EVENT)  do{\_\_EVENT -> bIsSet = true;}while(0)

#define WAIT\_EVENT(\_\_EVENT) do{ if(false != \_\_EVENT -> bIsSet)return true;}while(0)

#define RESET\_EVENT(\_\_EVENT) do{\_\_EVENT -> bIsSet = true; }while(0)

#defineINIT\_EVENT(\_\_EVENT,\_\_INIT\_VALUE,\_\_MANUAL)do{\_\_EVENT->bIsSet= \_\_INIT\_VALUE;\_\_EVENT-> bAutoReset = \_\_MANUAL ;}while(0)

上面用宏封装的好处：task\_print和task\_delay都可以使用上面的宏，传入的\_\_EVENT不同，即可对不同的事件进行基本操作。具有统一性，一般性。

INIT\_EVENT这里这个对事件的初始化，初始化了一些什么东西呢，

应该是一开始就调用，声明这个事件@s\_tEvent的初始值，即赋值：\_\_EVENT-> bIsSet = \_\_INIT\_VALUE，\_\_EVENT-> bAutoReset = \_\_MANUAL 。

以上是对test2b，事件触发的背景知识提要的 思考和理解，

然后来实现功能

遇到一个问题，这个事件，是指的哪个事件呢，哪个事件触发哪个事件

把打印看成一个事件，把延时也看成一个事件

每次不是很清晰的的时候，迷迷糊糊的写完代码后，才会有恍然大悟的感觉，层次明了的感觉。

