Test 1B 子状态机  
背景知识提要：  
1、 引入子状态机的概念  
2、 引入简单子状态机概念  
介绍一个概念——简单状态机  
一般状态机会有几个状态  
on\_going，表示状态机正在执行，还没有获得任何结果  
cpl，表示状态机已经执行完成了（complete的缩写）  
err，表示状态机发生了意料之外的错误，状态机的功能不能正常继续，并强制停止了  
我们可以定义一个枚举类型：  
//! \name finit state machine state  
//! @{  
typedef enum {  
    fsm\_rt\_err          = -1,    //!< fsm error, error code can be get from other interface  
    fsm\_rt\_cpl          = 0,     //!< fsm complete  
    fsm\_rt\_on\_going     = 1,     //!< fsm on-going  
} fsm\_rt\_t;  
//! @}  
  
fsm\_rt\_t fsm\_example(void)  
{  
    ...  
    return fsm\_rt\_on\_going;  
}  
显然，当状态机函数拥有返回值以后，我们就可以实现子状态机调用了  
3、 引入状态迁移（Condition和Behavior） 概念。介绍状态迁移事件与状态的区别。

Cuiz:

状态迁移：包含两部分，一是条件，二是行为，满足条件后进行状态转换

状态迁移事件与状态的区别：状态迁移事件是一个动作，一个箭头；状态指时间上或空间上位置，地点，有现状态，下一状态，上一状态。  
4、 引入基本状态模型以及和代码的对应关系  
基本要求：  
1、 在以下代码和状态图环境下，编写子状态机Print和Delay，实现1A题目中同样的效果。  
  
static fsm\_rt\_t test1b(void);  
static fsm\_rt\_t print(void);  
static fsm\_rt\_t delay(void);  
  
void main(void)  
{  
    system\_init();  
    while(1) {  
        breath\_led();  
        test1b();  
    }  
}  
  
#define TEST1\_RESET\_FSM() \  
do {\  
s\_tState = 0;\  
} while(0)  
  
static fsm\_rt\_t test1b(void)  
{  
    static enum {  
        TEST1\_START    = 0,  
        TEST1\_PRINT,  
        TEST1\_DELAY  
    } s\_tState = TEST1\_START;  
  
    switch (s\_tState) {  
        case TEST1\_START:  
            s\_chState = TEST1\_PRINT;  
            break;  
        case TEST1\_PRINT:  
            if (fsm\_rt\_cpl == print()) {  
                s\_chState = TEST1\_DELAY;  
            }  
            break;  
        case TEST1\_DELAY:  
            if (fsm\_rt\_cpl == delay()) {  
                RESET\_TEST1\_FSM();  
                return fsm\_rt\_cpl;  
            }  
    }  
    return fsm\_rt\_on\_going;  
}