怎么添加应用代码

1 在协议栈内注册一个user_event内核定时器，定时调度事件

- 调度间隔时间

```
#USER_EVENT_PERIOD   20  // 20ms (min_value >=10ms and min_unit=1ms)
```

- 向协议栈注册一个user_event的定时器

```
#include "builtin_timer.h"

static void ls_user_event_timer_init(void);
static void ls_user_event_timer_cb(void *param);
static struct builtin_timer *user_event_timer_inst = NULL;

static void ls_user_event_timer_init(void)
{
    user_event_timer_inst =builtin_timer_create(ls_user_event_timer_cb);
    builtin_timer_start(user_event_timer_inst, USER_EVENT_PERIOD, NULL);
}
```

- 在定时器的回调函数内处理相关应用层代码

```
static void ls_user_event_timer_cb(void *param)
{
   /**
     user_code
   */

   builtin_timer_start(user_event_timer_inst, USER_EVENT_PERIOD, NULL);
}
```

- user_event_timer初始函数添加位置

user_event_timer是向协议栈注册，故需要协议栈初始化完成，否则无效。导致该函数有一定的位置制约。最早设置位置如下：

```
static void dev_manager_callback(enum dev_evt_type type,union dev_evt_u *evt)
{
    switch(type)
    {
        case STACK_READY:
        {
           ls_user_event_timer_init();
        }
    }
}
```

2 向协议栈内核注册一个user_event事件，由内核调度器调度事件

```
void func_post(void (*func)(void *),void *param);
```

- 事件单次触发注册user_event

```
void user_event_1(void *arg)
{
    /**
       user_code
    */
}

void func_event(void)
{
    void *param;
    func_post(user_event_1, param);  /* !<Register user_event_1 to schedule in
the kernel*/
}
```

- 事件递归调度

```
void user_event_2(void *arg)
{
    /**
       user_code
    */
    func_post(user_event_2, Null);  /* !<Register user_event_2 to schedule in
the kernel*/
}
```

- user_event 函数添加位置

user_event是向协议栈注册，故需要协议栈初始化完成，否则无效。导致该函数有一定的位置制约。最早设置位置如下：

```
static void dev_manager_callback(enum dev_evt_type type,union dev_evt_u *evt)
{
    switch(type)
    {
        case STACK_READY:
        {
           func_event();
           user_event_2(NULL);
        }
    }
}
```