sl_Start Function Stuck Issue Report

## 1. Test code：

```
int ik_wifi_start(void){

    if(ik_wifi.pow==0){
        ERR("power is off\r\n");
        return -1;
    }

    if(ik_wifi.start==1){
        WAR("The current status is enabled. Ignore this operation \r\n");
        return 0;
    }

    {
        static int flg = 0;
        if( flg == 0 ){
            flg = 1;
            if(tx_thread_create(&w_thread, "wifi_thread", w_thread_fun, 0,w_thread_stack, sizeof(w_thread_stack),20, 20, TX_NO_TIME_SLICE, TX_AUTO_START)!=TX_SUCCESS){
                ERR("tx_thread_create");
            }
        }
    }

    ik_msleep(100);

    int Status = sl_Start(NULL,NULL,NULL);
    if(ROLE_STA == Status){
        ik_wifi.wifi_mod = ROLE_STA;
        printf("wifi mod: STA");
    }else if(ROLE_AP == Status){
        ik_wifi.wifi_mod = ROLE_AP;
        printf("wifi mod: AP");
    }else if(ROLE_P2P == Status){
        ik_wifi.wifi_mod = ROLE_P2P;
        printf("wifi mod: P2P");
    }else{
        my_printf(__RED,"Status:%d\r\n",Status);
        ERR("Start Role error");
        return -1;
    }

    ik_wifi.start = 1;

    return 0;
}
```

```
TX_THREAD w_thread;
uint8_t w_thread_stack[1024];
void w_thread_fun(ULONG thread_input){
    sl_Task(NULL);
}
```

## 2. Description：

First, create a wifi_thread that calls the sl_task function for uninterrupted polling, and then executes t he sl_start function in blocking mode. But after calling sl_start, it got stuck in the function and could n't move forward.

## 3. Direct reason of the stuck：

```
        /* Mark that device is in progress! */
        SL_SET_DEVICE_START_IN_PROGRESS;

        sl_DeviceDisable();
        sl_IfRegIntHdlr((SL_P_EVENT_HANDLER)_SlDrvRxIrqHandler, NULL);
        g_pCB->pInitCallback = pInitCallBack;
        sl_DeviceEnable();

        if(NULL == pInitCallBack){

            TIPS("---");
            ret = _SlDrvWaitForInternalAsyncEvent(ObjIdx, INIT_COMPLETE_TIMEOUT, SL_OPCODE_DEVICE_INITCOMPLETE);
            TIPS("+++");

            SL_UNSET_DEVICE_START_IN_PROGRESS;
            SL_SET_DEVICE_STARTED;
```

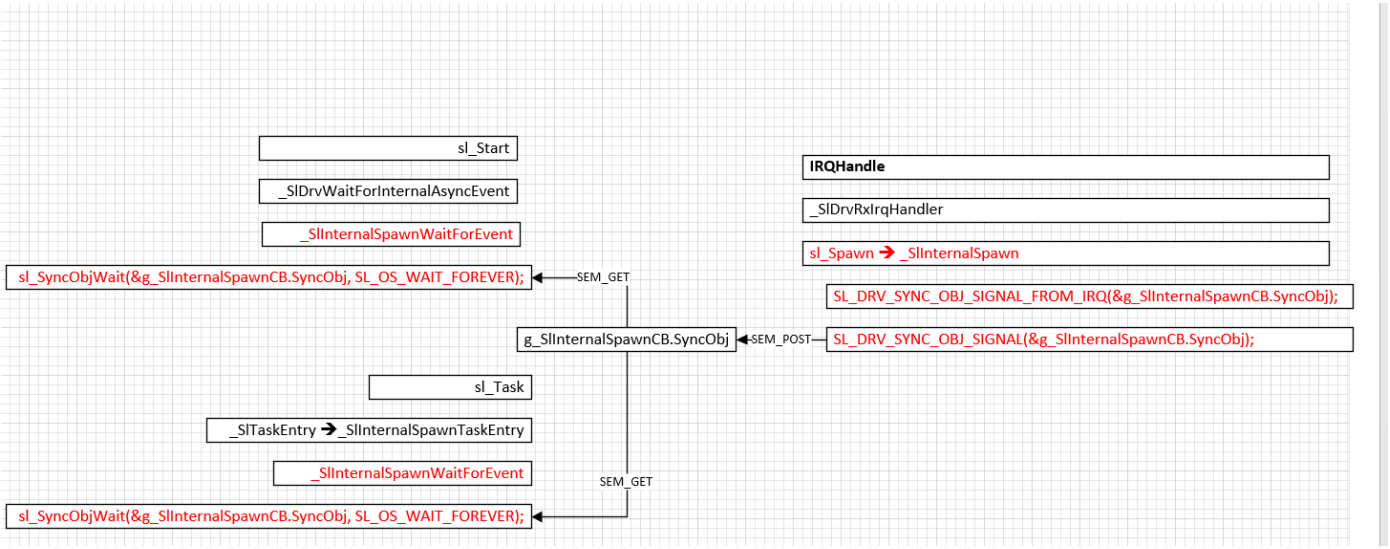
```
_SlReturnVal_t _SlDrvWaitForInternalAsyncEvent(_u8 ObjIdx,_u32 Timeout,_SlOpcode_t Opcode){

    _SlTimeoutParams_t SlTimeoutInfo = { 0 };
    if(_SlDrvIsSpawnOwnGlobalLock()){
        _SlDrvStartMeasureTimeout(&SlTimeoutInfo, Timeout);
        while(!Timeout || !_SlDrvIsTimeoutExpired(&SlTimeoutInfo)){
            /* If we are in spawn context, this is an API which was called from event handler,read any async event and check i
            ERR("---");
            _SlInternalSpawnWaitForEvent();
            ERR("+++");
            if(0 ==sl_SyncObjWait(&g_pCB->ObjPool[ObjIdx].SyncObj, SL_OS_NO_WAIT)){
                return(SL_OS_RET_CODE_OK);
            }
        }
        _SlDrvHandleFatalError(SL_DEVICE_EVENT_FATAL_CMD_TIMEOUT, Opcode,Timeout);
        return(SL_API_ABORTED);

    }else{
```

```
void _SlInternalSpawnWaitForEvent(void){

    sl_SyncObjWait(&g_SlInternalSpawnCB.SyncObj, SL_OS_WAIT_FOREVER);

    // call the processQ function will handle the pending async events alrea
    _SlSpawnMsgListProcess();

    // handle IRQ requests
    while(g_SlInternalSpawnCB.IrqWriteCnt != g_SlInternalSpawnCB.IrqReadCnt)
        // handle the ones that came from ISR context
        _SlDrvMsgReadSpawnCtx(g_SlInternalSpawnCB.pIrqFuncValue);
        g_SlInternalSpawnCB.IrqReadCnt++;
    }

}
```

sl_start function finally got stuck at sl_SyncObjWait(&g_SlInternalSpawnCB.SyncObj, SL_OS_WAIT_FOREVER);

g_SlInternalSpawnCB.SyncObj signal has not been obtained by sl_start, causing the thread pending.

## 4. Analysis of correlated semaphores



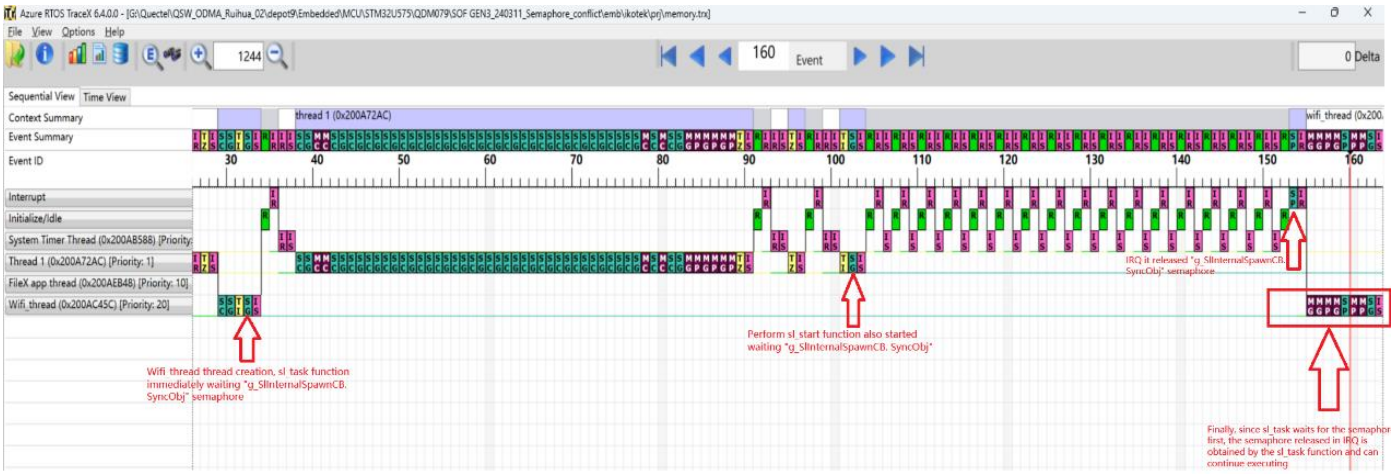

As shown above, both sl_start and sl_task will wait for g_SlInternalSpawnCB.SyncObj semaphore. And this semaphore is released in the IRQ function.

## 5. Further inference

We used TraceX to get the sequence diagram of system events.

Thread 1 is the main thread, and calls sl_start. wifi thread is sl_task thread. Pay attention to the gre en SG/SP symbol pointed by the arrow, representing the waiting and releasing of g_SlInternalSpawnCB.S yncObj (200AF2AC) semaphore.

It can be seen that sl_task enters waiting mode before sl_start. Therefore, when IRQ release semap hore, it is obtained by sl_task, and sl_start got stuck in waiting mode without obtaining a semaphore.




The captured data is as follows. Please use Azure RTOS TraceX to view the event timing chart.

memory.trx