Design for SummerSchool II (CanCAR for example)

XF

Pascal Sartoretti

Objectives

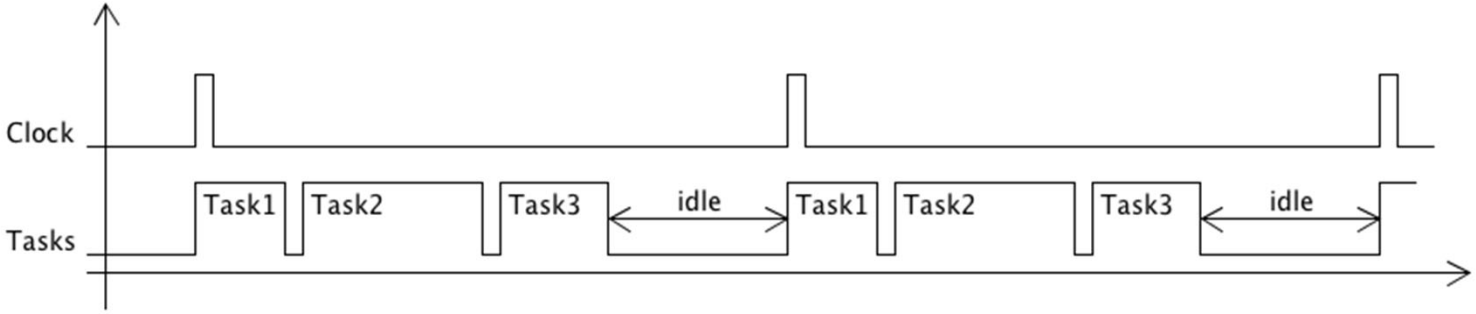
Working with XF (eXecution Framework)

- Goal, why
- How it works
- Usage example

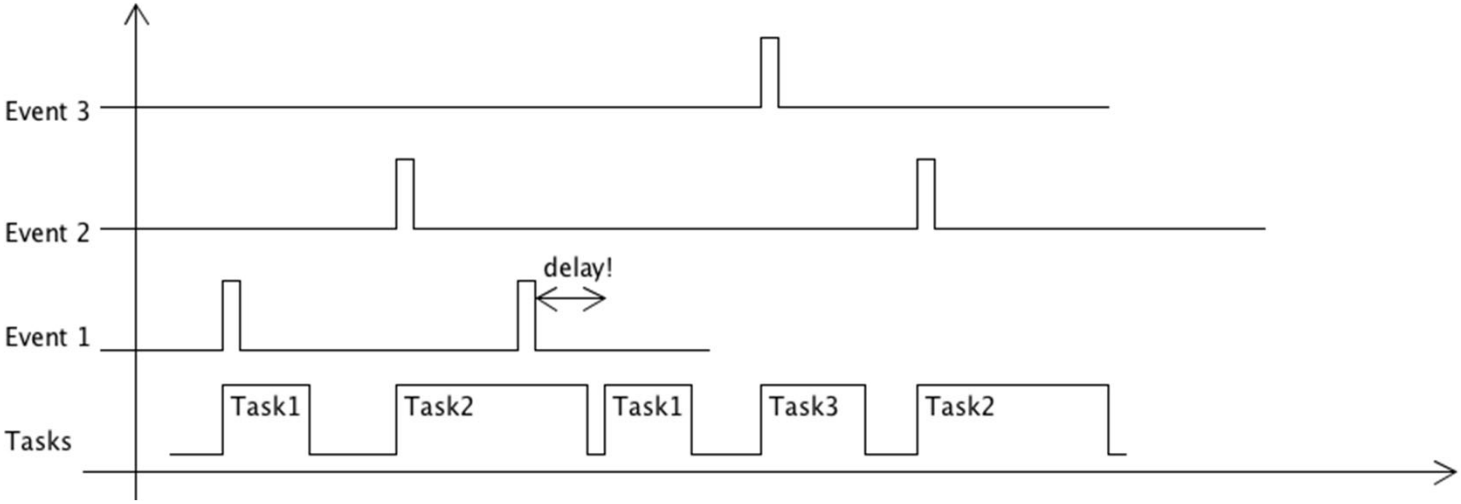
XF, why?

- Repeating tasks

Synchronous



Asynchronous



XF characteristics

- Event queue
- Timer manager
- Event dispatcher
- Protection mechanisms

XF Event

- What is an Event

id	1 .. 255
processEvent	→
delay	⏰

```
typedef bool (*processEventT)(struct Event_* ev);

typedef struct Event_                // event structure
{
    uint8_t id;                      // id of event (>0)
    processEventT processEvent;      // function to call
    uint16_t delay;                  // delay if not 0
}Event;
```

XF Timer

- What is a Timer

tm	
id	1 .. 255
processEvent	→
delay	

```
typedef struct Timer_          // timer structure
{
    uint16_t tm;               // time to decount
    Event ev;                  // event to post
} Timer;
```

The XF

Timer[]	countTmr
Timer[]	maxTmr
...	
Event[]	countEvt
Event[]	maxEvt
...	

```
define MAXTIMER 8            // number of timers in system
#define MAXEVENT 20          // number of events in system
#define TICKINTERVAL 10      // this is the tick duration in ms

typedef struct XF_                  // the XF structure
{
    Timer timerList[MAXTIMER];      // the timers
    Event eventQueue[MAXEVENT];     // the events
    uint8_t countEvt;               // the events counter
    uint8_t maxEvt;                 // the max reached (debug)
    uint8_t countTmr;               // the timers counter
    uint8_t maxTmr;                 // the max reached (debug)
} XF;
```

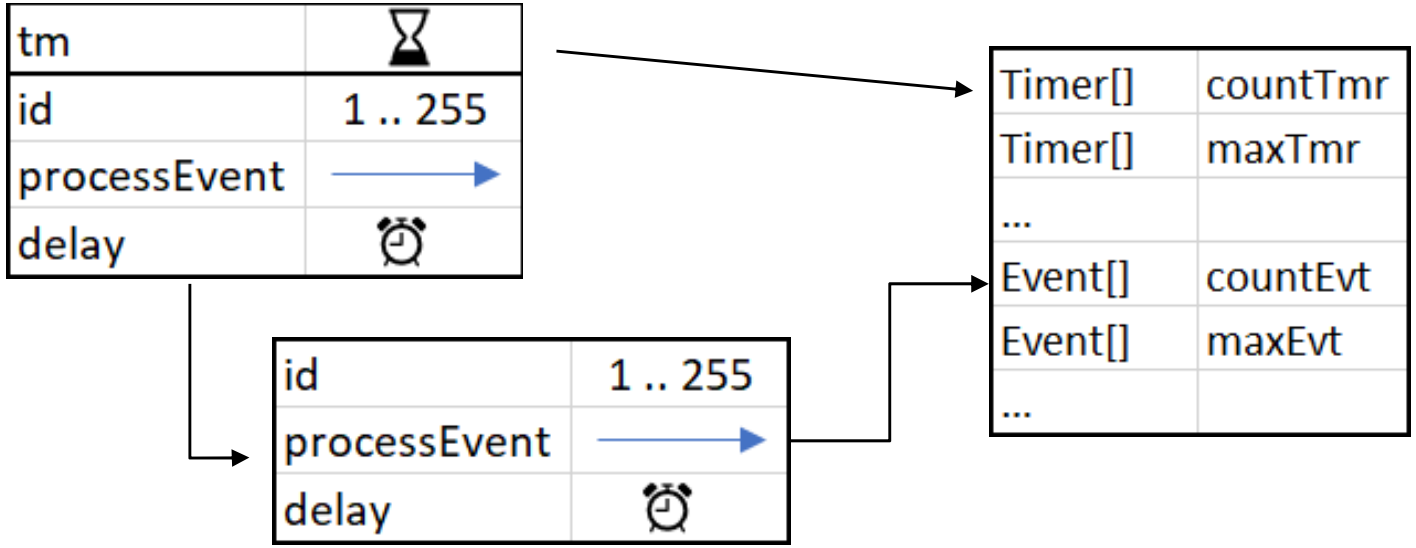
XF usage (main calls)

```
void main(void)
{

    SYSTEM_Initialize();        // MCC initialisation
    XF_init();                  // init the XF
    // set the callback timer interrupt routine
    TMR0_SetInterruptHandler(XF_decrementAndQueueTimers);
    INTERRUPT_GlobalInterruptEnable();  // enable timer ISR
    // post an event (example)
    XF_post(lightControl,LIGHT_INIT,0); // first event

    while(1)                    // forever loop
    {
        XF_executeOnce();       // execute the XF
    }
}
```

XF usage (XF_ `decrementAndQueueTimers`)

Decrement the timers of TICKINTERVAL

- If timer reachs 0, place in eventQueue



XF usage (XF_post) – never from ISR !

```
TimerID XF_post(processEventT processEvent,
                uint8_t id,
                Time delay);

tmr = XF_post(lightControl_Process,E_INIT,10);
```

Depending if delay is > 0

- goes in timerList
- else goes in eventQueue

Timer[]	countTmr
Timer[]	maxTmr
...	
Event[]	countEvt
Event[]	maxEvt
...	

XF usage (XF_unscheduleTimer)

```
void XF_unscheduleTimer(TimerID id,
                        bool inISR);

XF_unscheduleTimer(tmr,false);
```

Remove timer[id] in the XF timers list

- Will never arrive

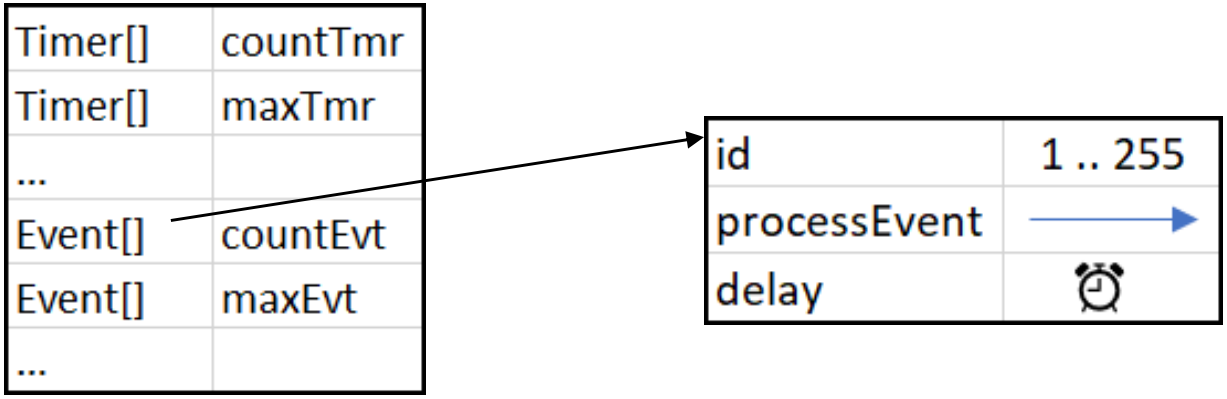
Timer[]	countTmr
Timer[]	maxTmr
...	
Event[]	countEvt
Event[]	maxEvt
...	

XF usage (XF_executeOnce)

Called infinitely from main - while(1)

- Read the eventQueue
- If an event exist

  Call the function: event->processEvent

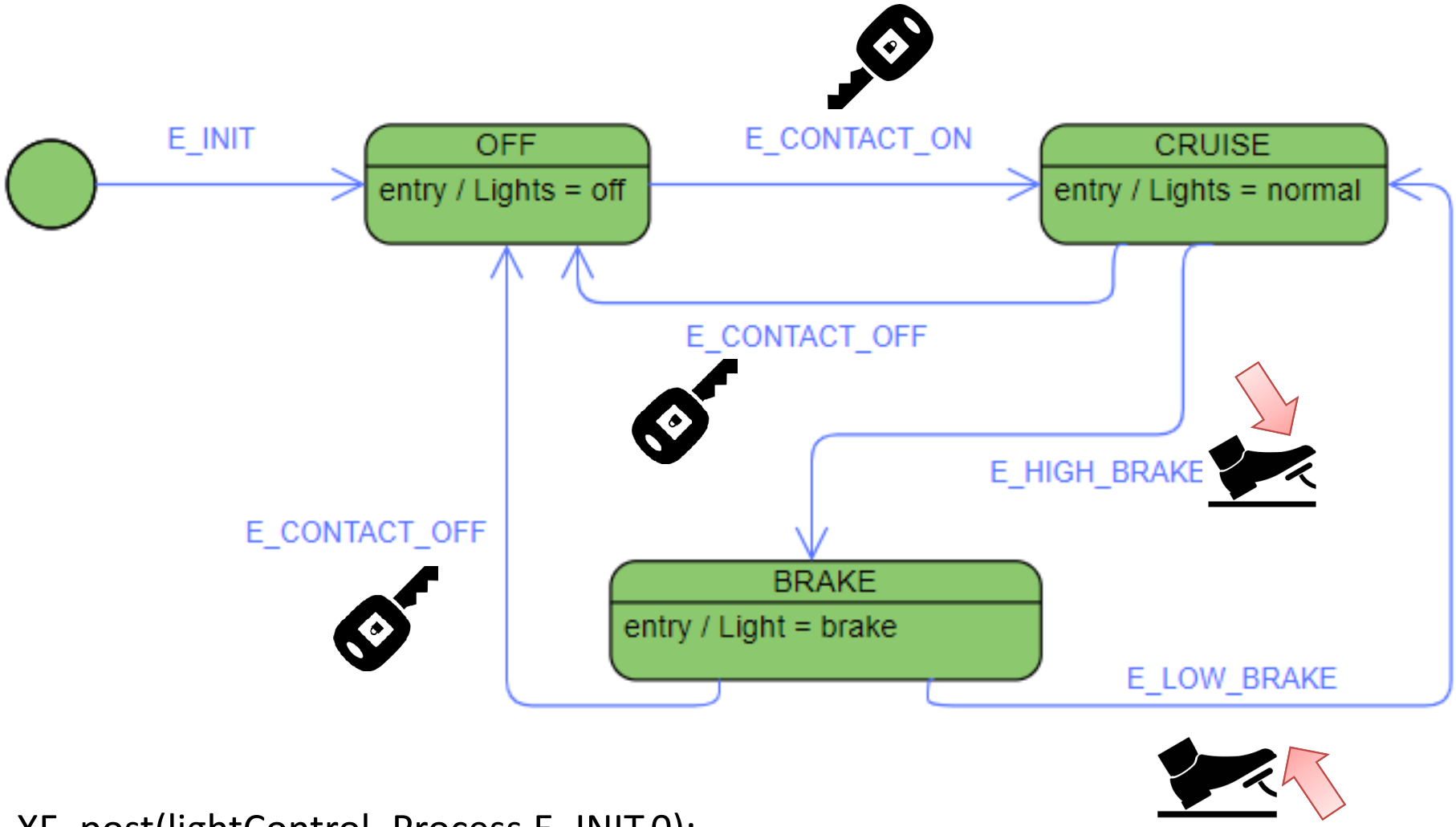


Read CAN messages in main()

```
...
 while(1)
 {
   updateCarState(); // high priority CAN read function
   XF_executeOnce(); // XF execution
 }
...
```

```
 void updateCarState(void)
 ...
  case ID_CONTACT_KEY:
     carState.contactKey = rxd[0];
     if(rxd[0] == 0)
     {
       XF_post(lightControl_Process,E_CONTACT_OFF,0);
     }
     else
     {
       XF_post(lightControl_Process,E_CONTACT_ON,0);
     }
     break;
 ...
```

XF usage (lightControl_Process)



XF_post(lightControl_Process,E_INIT,0);

XF usage (lightControl_Process) 1/4

```
bool lightControl_Process(Event* ev)
{
 typedef enum
    {
        INIT,
        OFF,
        CRUISE,
        BRAKE
    }LightState;
    static LightState state = INIT;
    static LightState oldState = INIT;

    ...
```

XF usage (lightControl_Process) 2/4

```
...
switch(state)        // transition state machine
    {
        case INIT:
            if(ev->id == E_INIT)
            {
                state = OFF;
            }
        break;
        case OFF:
            if(ev->id == E_CONTACT_ON)
            {
                state = CRUISE;
            }
        break;
        case CRUISE:
            if(ev->id == E_CONTACT_OFF)
            {
                state = OFF;
            }
            else if (ev->id == E_HIGH_BRAKE)
            {
                state = BRAKE;
            }
        break;
        ...
```

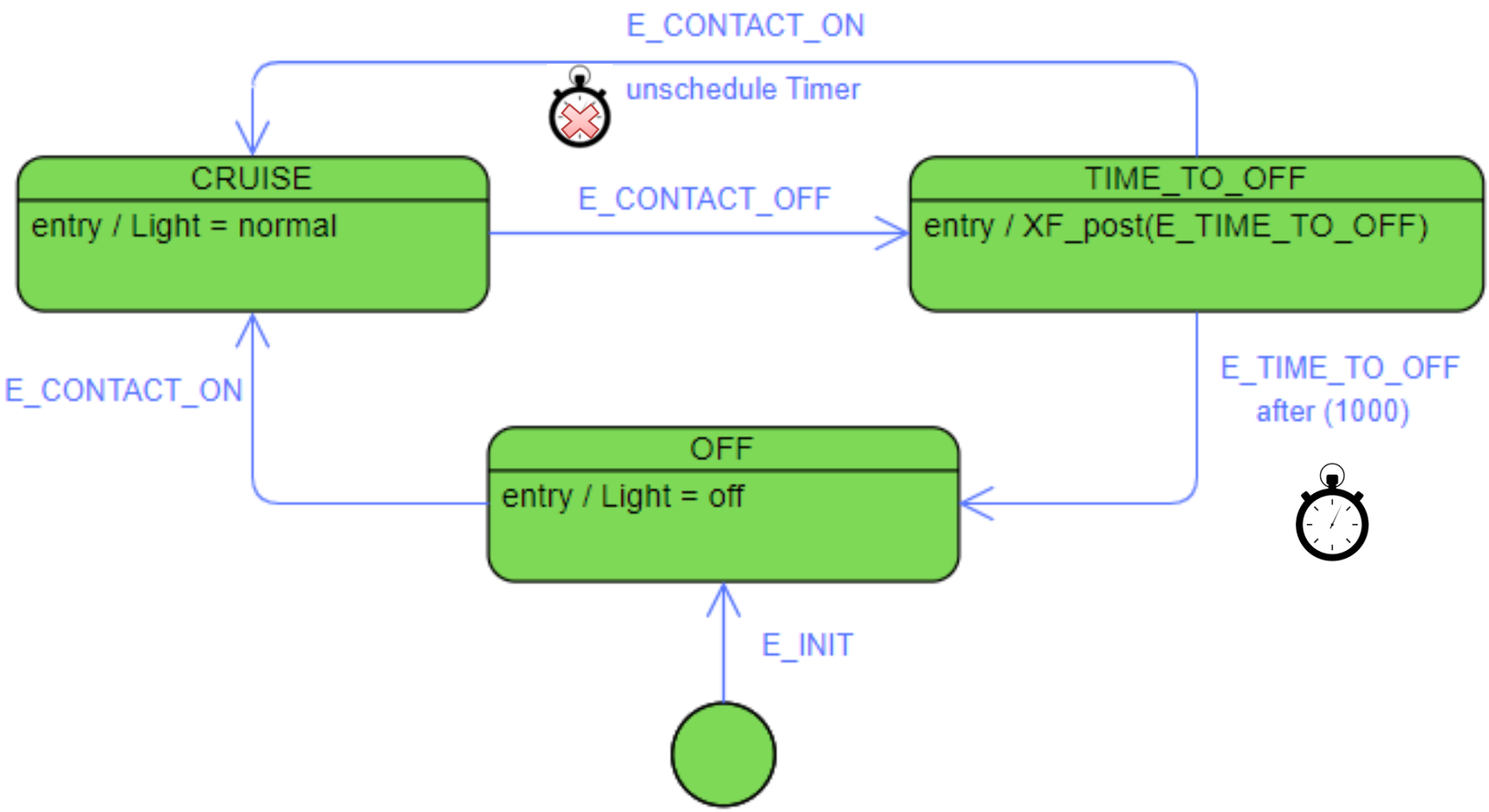
XF usage (lightControl_Process) 3/4

```
...
if(oldState == state)        // check state change
  {
      return 0;              // no change, no entry action
  }
...
```

XF usage (lightControl_Process) 4/4

```
...
switch(state)       // entry execution state
{
 case INIT:
 break;
 case OFF:
    lightContol_FrontLight(0);
    lightContol_BackLight(0);
 break;
 case CRUISE:
    lightContol_FrontLight(50);
    lightContol_BackLight(50);
 break;
 ...
}
```

XF usage (lightControl_Process with timer)



XF usage (XF_pushEvent) – in ISR (no timer)

```
bool XF_pushEvent(Event ev,
                  bool inISR,
                  TimerId* tmid);

void __interrupt() anyISR(void)
{
    uint8_t tmid;              // not used but mandatory
    Event evt;                 // the Event
    evt.delay = 0;             // must be 0 in ISR
    evt.id = E_ISR_ANY;        // for example
    evt.processEvent = functionToCall;  // for example

    XF_pushEvent(evt,true,&tmid); // true because ISR
}
```

XF : protection mechanisms

Called from ISR

- `XF_decrementAndQueueTimers()`
- `XF_pushEvent() -> without timer`
- `XF_unscheduleTimer()`

Called from main

- `XF_executeOnce()`
- `XF_post(), XF_pushEvent()`
- `XF_unscheduleTimer()`

```
static void ENTERCRITICAL(bool inISR)
{
    if (inISR == false)
    {
      //GIE = 0;
      INTERRUPT_GlobalInterruptDisable();
    }
}
```