TIME-RELATED APIS IN MBED-OS

Handong university

Jong-won Lee

2022.04.13

Mbed-OS 시간 관련 클래스

- **Ticker**
  - 일정 시간 간격으로 반복하여 인터럽트를 발생시키는 목적으로 사용될 수 있음.
- **Timeout**
  - 일정 시간 후에 한번만 인터럽트를 발생시키는 목적으로 사용될 수 있음.
- **Timer**
  - 경과 시간을 측정할 목적으로 사용될 수 있음.
    - 내부 동작은 64 bit counter를 사용하고 있다.
  - 시간 해상도: µs

Ticker

Ticker class

생성자	**Ticker()**
동작	Ticker 객체를 생성한다.
함수	void **attach** (Callback<void()> func, std::chrono::microseconds t)
동작	t μs 간격으로 실행될 ISR func을 등록한다.
함수	void **detach**()
동작	등록을 해제한다. (interrupt가 해제된다.)

Ex.:

생성 (
- Ticker led_ticker; 티커 객체 생성

등록 (
- led_ticker.attach(&blink_led, 500000us); //every 0.5 sec.
- led_ticker.attach(&blink_led, 500ms); //every 0.5 sec.

같음

제거 (
- led_ticker.detach();

Timeout

Timeout class

생성자	**TimeOut()**
동작	TimeOut 객체를 생성한다.
함수	void **attach** (Callback<void()> func, std::chrono::microseconds t)
동작	t μs 후에 실행될 ISR func을 등록한다.
함수	void **detach**()
동작	등록을 해제한다. (interrupt가 해제된다.)
함수	std::chrono::microseconds **remaining_time** ()
동작	Timeout까지 남아있는 시간 값을 반환한다. 이미 timeout이 발생하였다면 음수 값이 반환된다.

Timeout

- **Timeout class**
  - **Ex.:**
    - Timeout tout;
    - tout.attach(&tout_cb, 1s); // execute tout_cb() after 1sec.

Timer

Timer class

생성자	**Timer()**
동작	Timer 객체를 생성한다.
함수	void **start** ()
동작	Timer 동작을 시작시킨다.
함수	void **stop** ()
동작	Timer 동작을 중단시킨다.
함수	void **reset** ()
동작	Timer의 값을 '0'으로 초기화시킨다. 현재 timer가 동작 상태이면 계속 동작한다.
함수	std::chrono::microseconds **elapse_time ()**
동작	경과한 시간 값을 반환한다. (64 bit integer)

unsigned long long int

Timer

- **Timer class**
  - **Ex.:**
    - Timer t;
    - t.start();
    - t.stop();
    - sprintf(buffer, "The time taken was %**llu** us\r\n", **(t.elapsed_time()).count()**);
    - sprintf(buffer, "The time taken was %**llu** ms\r\n", **duration_cast<milliseconds>(t.elapsed_time()).count()**);
    - sprintf(buffer, "The time taken was %**f** sec\r\n", **duration<float>(t.elapsed_time()).count()**);
      - Use standard printf library

Low Power Timer Related APIs

- **LowPowerTicker**
  - The LowPowerTicker class has the same methods as the Ticker class but **operates in deep sleep mode** and has less resolution.
    - when you only need millisecond accuracy
- **LowPowerTimeout**
- **LowPowerTimer**

TIME-RELATED APIS IN MBED-OS

Handong university

Jong-won Lee

2022.04.13

Lab. 6-1

- 실습 목적
  - Timer 객체를 생성하여 일정 시간 간격으로 반복하여 인터럽트를 발생시킬 수 있다.
- 실습 시나리오
  - Nucleo board 보드의 버튼을 누를 때마다 LED2의 점멸 시간의 주기가 바뀐다. 시간 변화는 다음과 같다. (500 ms on/off ⇨ 1 sec on/off ⇨ 2 sec on/off ⇨ 4 sec on/off ⇨ 125 ms on/off ⇨ 250 ms on/off ⇨ 500 ms on/off ⇨ …)
  - LED2 점멸을 Ticker 객체를 이용해서 구현한다.
- 실습 과정
  - LED2 점멸을 위한 Ticker 객체를, LED2 출력을 위한 DigitalOut 객체를, Interrupt 기능 구현을 위한 InterruptIn 객체를 생성한다.
  - 버튼이 눌릴 때 시간이 바뀌도록 구현하며, Ticker를 이용하여 바뀐 시간 간격에 따라 LED2가 점멸되도록 구현한다.

Lab 6-1

- A sample code

```
#include "mbed.h"

DigitalOut led2(LED2);
InterruptIn button(USER_BUTTON);
// Used for blinking the LED2
Ticker led2_ticker;

float interval = 0.5;   // 500ms
```

```
// ISR to handle button pressed event
void button_onpressed_cb(void)
{
    led2_ticker.detach();  // disable the timer
    interval = interval * 2;
    if (interval > 4.0) {
        interval = 0.125;
    }
    led2 = 1;   // turn on the LED2
    led2_ticker.attach(&timeout_cb, interval);
}
```

Lab 6-1

- A sample code (con't)

```
// ISR for time out
void timeout_cb(void)
{
    led2 = !led2;
}
```

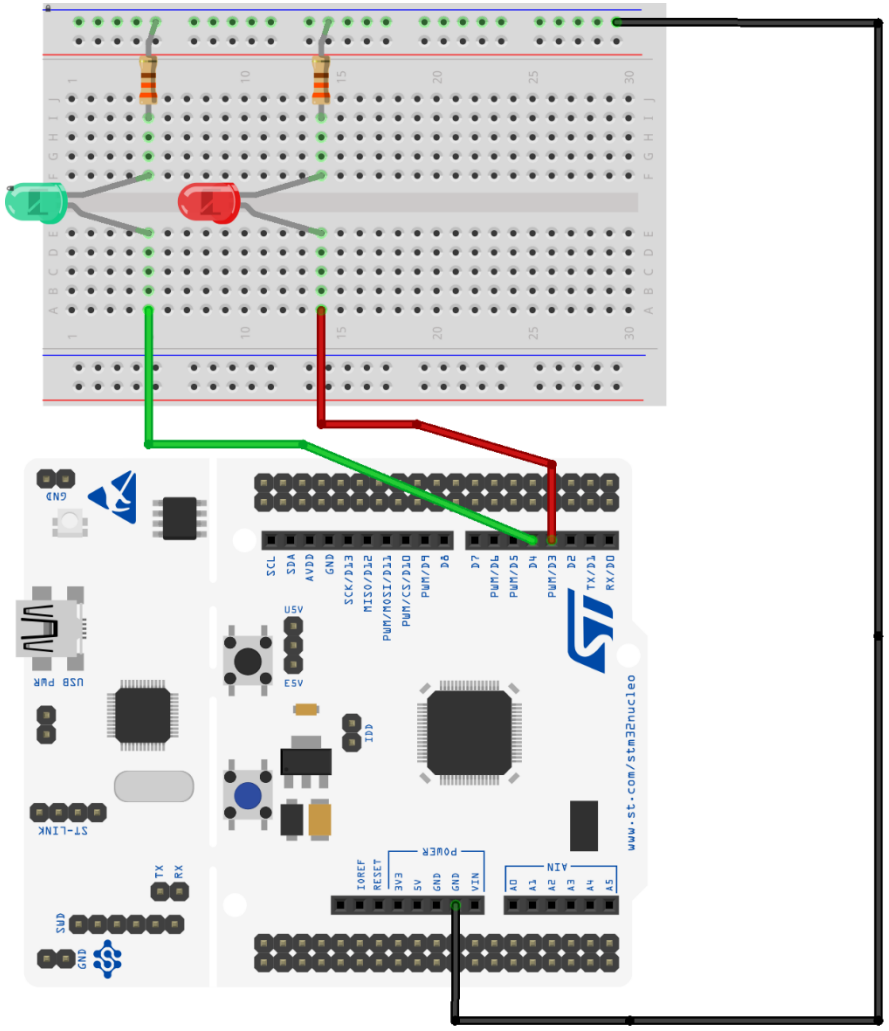
```
int main()
{
    button.mode(PullUp); // Activate pull-up
    // Attach ISR to handle button-pressed event
    button.fall(&button_onpressed_cb));

   // timer ISR for blinking the LED2
   led2_ticker.attach(&timeout_cb, interval);
    while(1) {
    }
}
```

Lab 6-2

- 실습 목적
  - Timeout 객체를 이용하여 일정 시간 후에 인터럽트를 발생시킬 수 있다.
- 실습 시나리오
  - Ticker를 이용하여 Red led를 1초 주기로 깜빡이고, Green led가 10초 후에 꺼지도록 한다.
- 실습 과정
  - Red led 점멸을 위한 Ticker 객체를, Green led를 끄기 위한 Timeout 객체를 생성한다.
  - Red led는 1초 주기로 깜빡이고, Green led는 10초 후에 꺼지도록 기능을 구현한다.

Lab 6-2

- 회로 구성.
  - LED 제어를 위한 회로 구성
  - D3, D4에 전선을 연결 후 저항(220~330Ω)과 LED 연결
  - D3 (Red LED), D4(Green LED)



Lab 6-3

- 실습 목적
  - Timer 객체를 이용해서 경과 시간을 측정할 수 있다.
- 실습 시나리오
  - 1부터 1,000,000까지 더하는데 소요되는 시간을 반복해서 측정하며 출력한다.
- 실습 과정
  - Timer 객체를 생성한다.
  - 여러 함수들을 호출하여 해당 기능을 수행할 수 있도록 한다.