Triggers

Young Won Lim
4/11/16

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice.

Trigger.h and Trigger.c

Based on

Implementation of generic triggers.
Erich Styger, erich.styger@hslu.ch
21.03.2011

http://www.steinerberg.com/EmbeddedComponents/BookSrc/Trigger.c
http://www.steinerberg.com/EmbeddedComponents/BookSrc/Trigger.h

Overview of Trigger.h

```
typedef enum { ??? }        TRG_TriggerKind;

typedef uint16_t            TRG_TriggerTime;

typedef void *              TRG_CallBackDataPtr;

typedef void (*TRG_Callback)(TRG_CallBackDataPtr);


uint8_t   TRG_SetTrigger(??Kind, ??Time, ??Cback, ???CbDataPtr);

void      TRG_IncTick(void);

void      TRG_Init(void);
```

Overview of Trigger.c

typedef struct TRG_Trigger**Desc** { ??? } TRG_Trigger**Desc**;

static TRG_Trigger**Desc** TRG_**Triggers**[TRG_NOF_TRIGGERS];

uint8_t TRG_**Set**Trigger(??**Kind**, ??**Time**, ??**Cback**, ???Cb**DataPtr**) { *definition* }

static bool **Check**Callbacks(void) { *definition* }

void TRG_**Inc**Tick(void) { *definition* }

void TRG_**Init**(void) { *definition* }

Trigger.h

```
#define TRG_TICKS_MS  TMR_TICK_MS

typedef enum {
  TRG_LED_BLINK,
  TRG_BTNLED_OFF,
  TRG_BTNSND_OFF,
  TRG_KEYPRESS,
  TRG_NOF_TRIGGERS
} TRG_TriggerKind;

typedef void *TRG_CallBackDataPtr;

typedef void (*TRG_Callback)(TRG_CallBackDataPtr);

typedef uint16_t TRG_TriggerTime;

uint8_t TRG_SetTrigger(TRG_TriggerKind     trigger,
                       TRG_TriggerTime     ticks,
                       TRG_Callback        callback,
                       TRG_CallBackDataPtr data);

void TRG_IncTick(void);

void TRG_Init(void);
```

Trigger.c – TRG_SetTrigger( )

```
typedef enum {
  TRG_LED_BLINK,
  TRG_BTNLED_OFF,
  TRG_BTNSND_OFF,
  TRG_KEYPRESS,
  TRG_NOF_TRIGGERS
} TRG_TriggerKind;

typedef uint16_t  TRG_TriggerTime;
typedef void (*TRG_Callback)(TRG_CallBackDataPtr);
typedef void *    TRG_CallBackDataPtr;
```

```
typedef struct TRG_TriggerDesc {
  TRG_TriggerTime     ticks;
  TRG_Callback        callback;
  TRG_CallBackDataPtr data;
} TRG_TriggerDesc;

static TRG_TriggerDesc TRG_Triggers[TRG_NOF_TRIGGERS];
```

```
uint8_t TRG_SetTrigger(TRG_TriggerKind     trigger,
                       TRG_TriggerTime     ticks,
                       TRG_Callback        callback,
                       TRG_CallBackDataPtr data) {
  EnterCritical();

  TRG_Triggers[trigger].ticks = ticks;
  TRG_Triggers[trigger].callback = callback;
  TRG_Triggers[trigger].data = data;

  ExitCritical();

  return ERR_OK;
}
```

Trigger.c – CheckCallbacks( )

```
static bool CheckCallbacks(void) {
  TRG_TriggerKind     i;
  TRG_Callback        callback;
  TRG_CallBackDataPtr data;
  bool                calledCallBack = FALSE;

  for(i=(TRG_TriggerKind)0;i<TRG_NOF_TRIGGERS;i++) {
    EnterCritical();

    if (TRG_Triggers[i].ticks    == 0   &&
        TRG_Triggers[i].callback != NULL ) {

      callback = TRG_Triggers[i].callback;
      data     = TRG_Triggers[i].data;

      TRG_Triggers[i].callback = NULL;

      ExitCritical();

      callback(data);
      calledCallBack = TRUE;
    } else {
      ExitCritical();
    }
  }
  return calledCallBack;
}
```

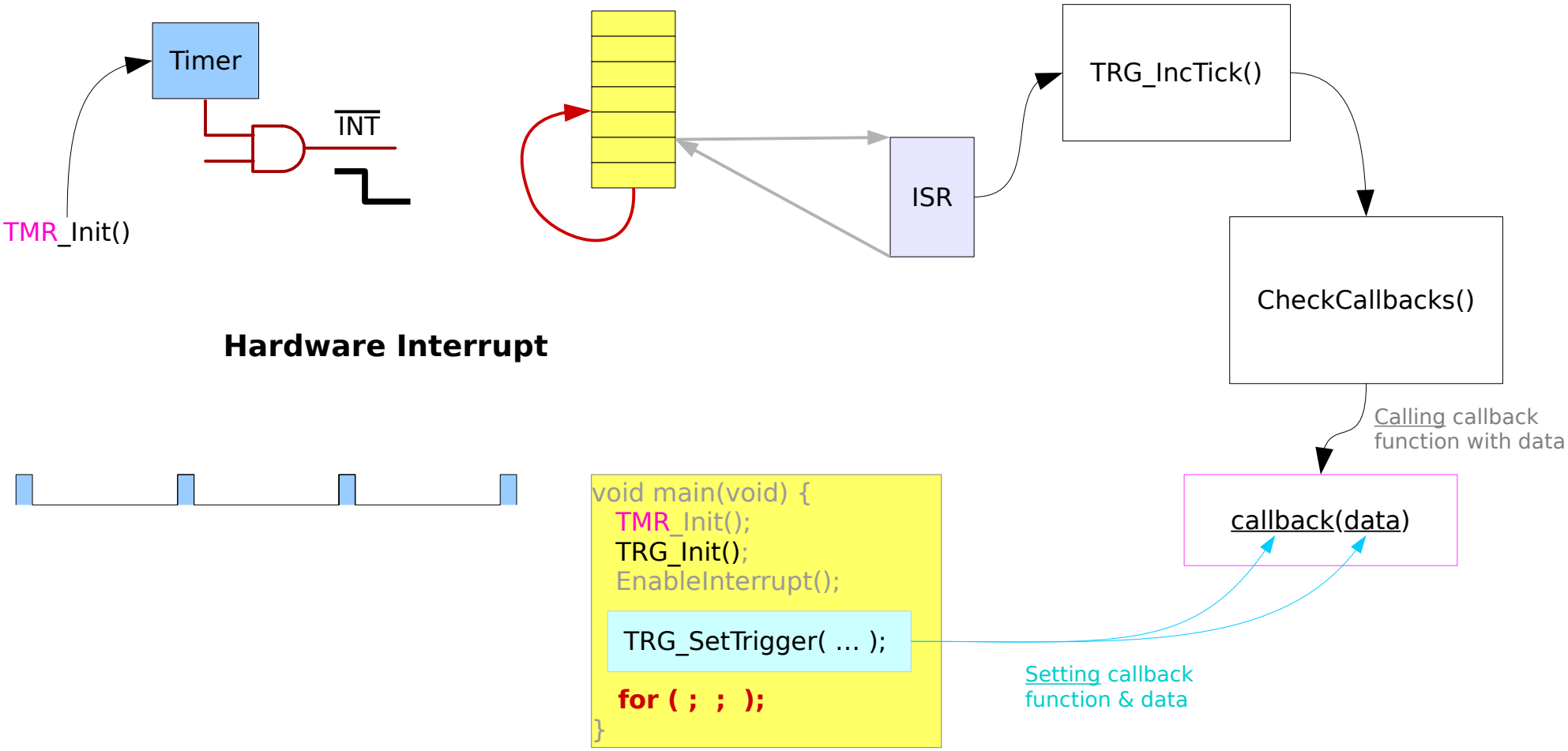
```
typedef enum {
  TRG_LED_BLINK,
  TRG_BTNLED_OFF,
  TRG_BTNSND_OFF,
  TRG_KEYPRESS,
  TRG_NOF_TRIGGERS
} TRG_TriggerKind;

typedef uint16_t  TRG_TriggerTime;
typedef void (*TRG_Callback)(TRG_CallBackDataPtr);
typedef void *    TRG_CallBackDataPtr;
```

Trigger.c – IncTick( ), Init( )

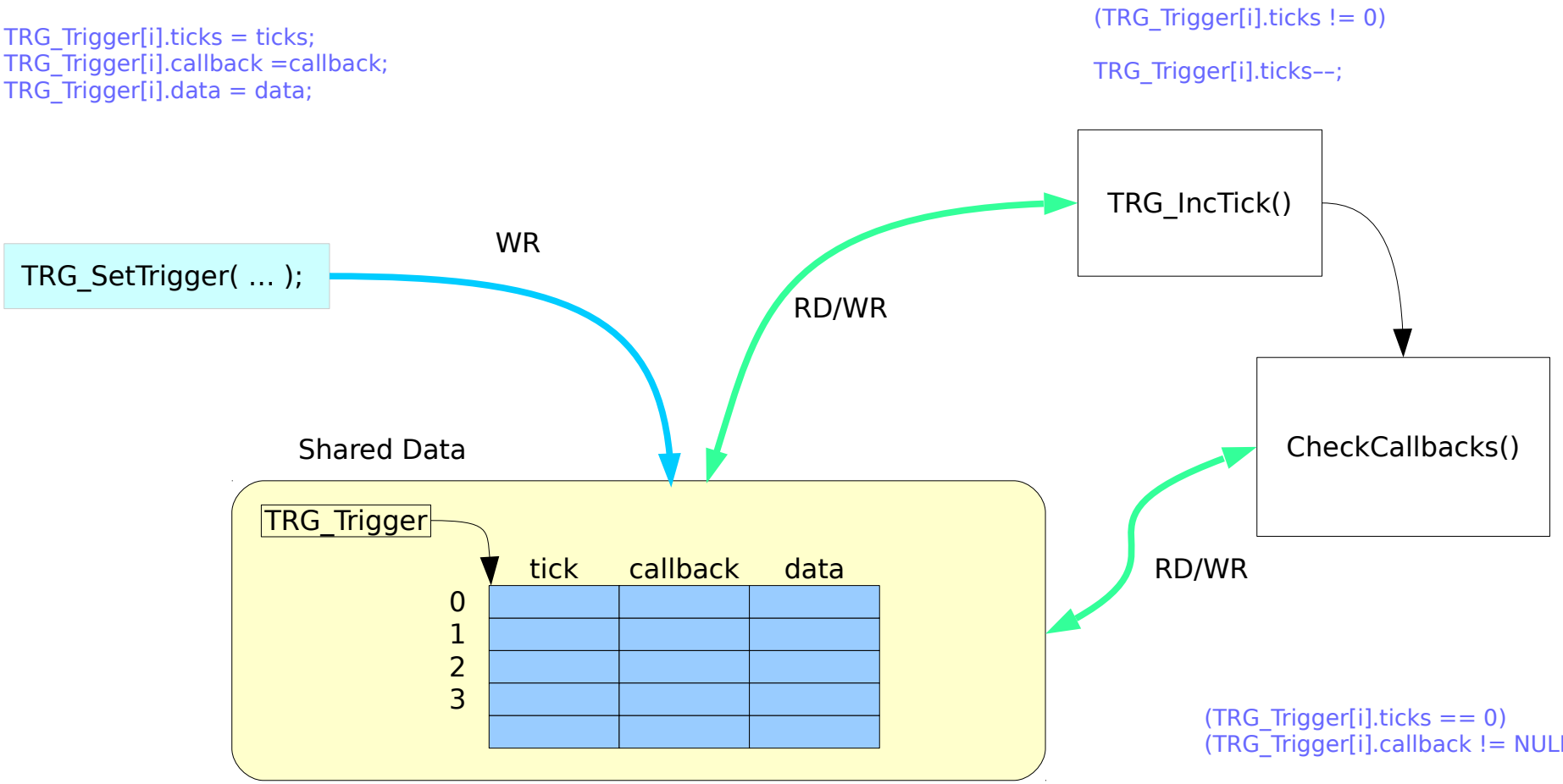
```
void TRG_IncTick(void) {
  TRG_TriggerKind i;

  EnterCritical();

  for(i=(TRG_TriggerKind)0; i < TRG_NOF_TRIGGERS; i++) {
    if (TRG_Triggers[i].ticks!=0) {
      TRG_Triggers[i].ticks--;
    }
  }

  ExitCritical();

  while(CheckCallbacks()) {}
}
```

```
void TRG_Init(void) {
  TRG_TriggerKind i;

  for(i=(TRG_TriggerKind)0;i<TRG_NOF_TRIGGERS;i++) {
    TRG_Triggers[i].ticks = 0;
    TRG_Triggers[i].callback = NULL;
    TRG_Triggers[i].data = NULL;
  }
}
```

Timer & Interrupt



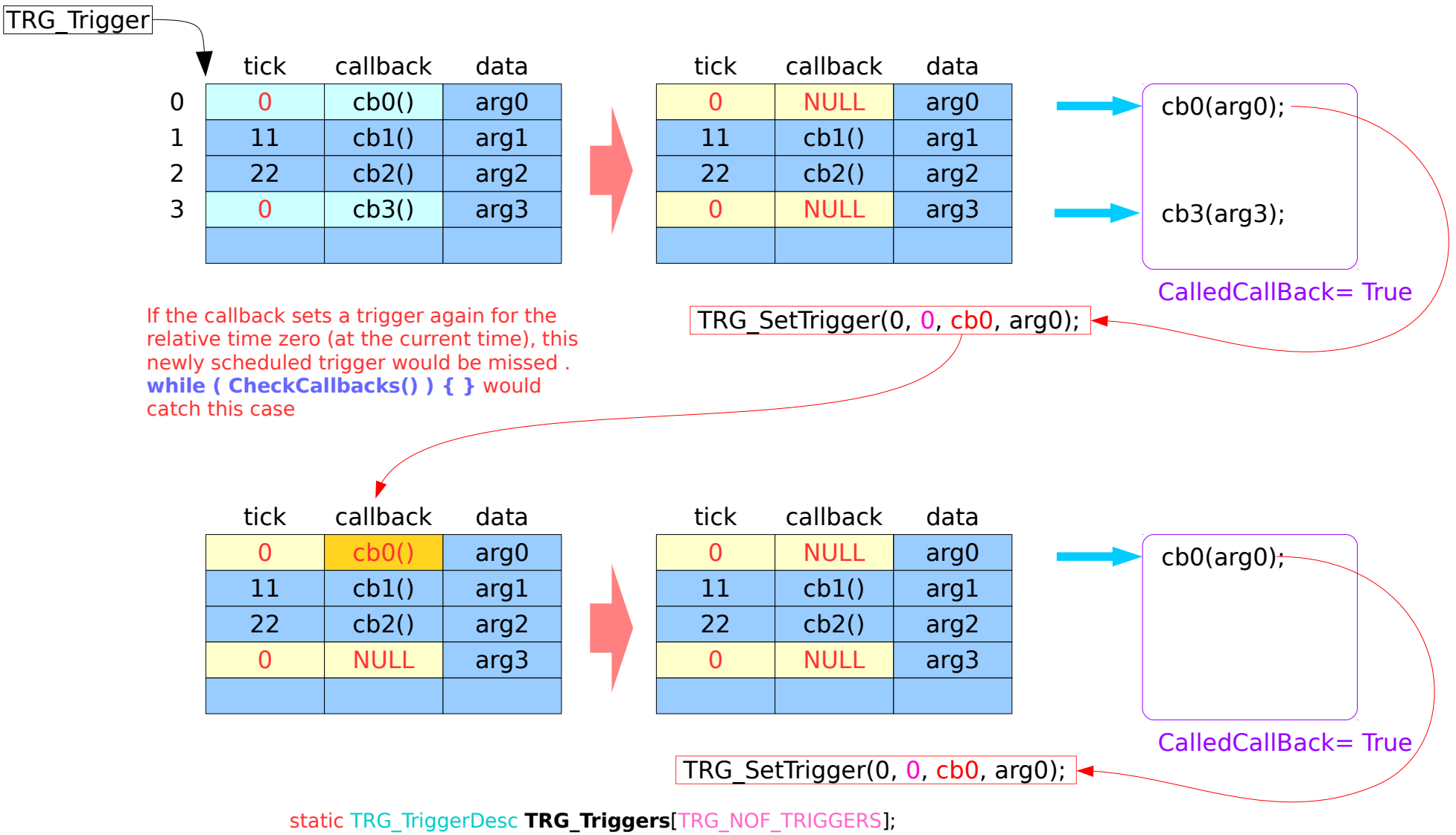
Critical Section Access




```
static TRG_TriggerDesc TRG_Triggers[TRG_NOF_TRIGGERS];
```

File Scope : static global variable

Invoking Callbacks



static TRG_TriggerDesc **TRG_Triggers**[TRG_NOF_TRIGGERS];

Trigger.c

```
main() {

        TRG_SetTrigger(TRG_LED, 0, LED_HeartBeat, NULL);

}
```

```
TRG_TriggerKind :       TRG_LED,
TRG_TriggerTime :       0,
TRG_Callback :          LED_HeartBeat,
TRG_CallBackDataPt :    NULL
```

Callback function

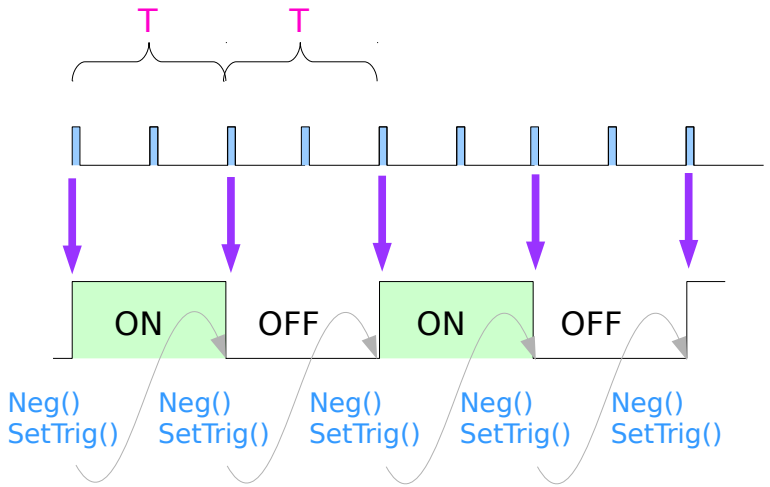
```
LED_HeartBeat() {

        LED1_Neg();

        TRG_SetTrigger(TRG_LED, T, LED_HeartBeat, NULL);

}
```

```
Main() {
  foo();
  foo();
  foo();
}
```

Call foo() multiple times
with the same led location

```
foo() {

  static uint8_t led = 1;

  TRG_SetTrigger(TRG_LED, 0, LED_HeartBeat, &led);

}
```

0 : immediate trigger



References

[1] http://wiki.osdev.org/ARM_RaspberryPi_Tutorial_C
[2] http://blog.bobuhiro11.net/2014/01-13-baremetal.html
[3] http://www.valvers.com/open-software/raspberry-pi/
[4] https://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/os/downloads.html