

# Recap - Trigger & Debouncing

Simon Durrer und Manuel Felber

October 21, 2016

# Table of contents

## Trigger

- Gründe für Triggers?

- Idee

- Descriptor

- Setzen & Genauigkeit

- Keep in Mind!

- Triggers != Events

## Debouncing

- Grund für Endprellen

- Idee

- Debounce FSM

- Timing

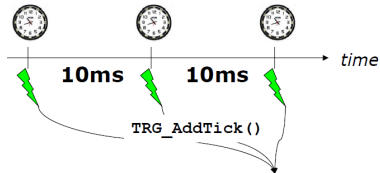
## Quiz

- Quiz

# Gründe für Triggers?

- ▶ Timer sind Mangelware → Trigger Modul braucht nur 1 Timer
- ▶ Aufwändige Implementation bei vielen verschiedenen Anwendungen → Trigger bieten gemeinsames IF
- ▶ Schwieriges Handling in der ISR → Laufzeit/Speicher optimiert

# Idee



```
void TRG_AddTick(void) {  
    Increment Tick Counter;  
    if HasTriggerForThisTickCount then  
        removeTrigger;  
        callback();  
    end if  
}
```

# Descriptor

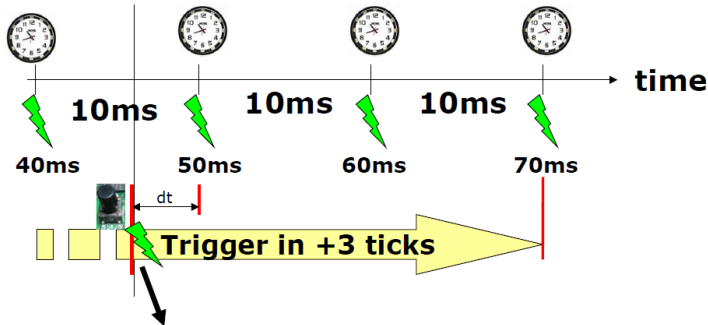
```
typedef void (*TRG_Callback) (void*);  
  
typedef struct {  
    uint16_t triggerTick;  
    TRG_Callback callback;  
    void *data;  
} TriggerDesc;
```

Wann

Wo

Wie

## Setzen & Genauigkeit



```
TRG_SetTrigger(TRG_BTNLED_OFF, 3, BlinkLED, NULL);
```

## Keep in Mind!

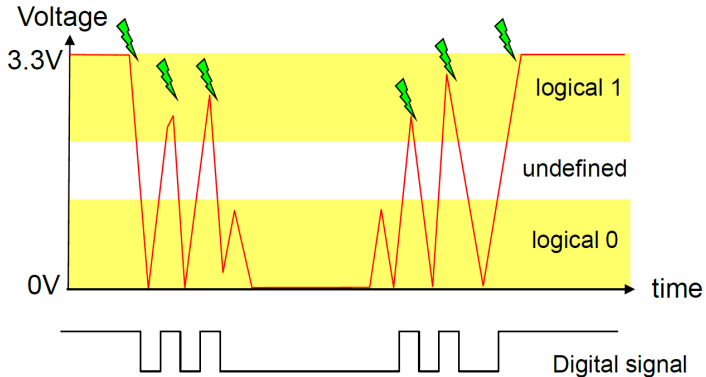
- ▶ Callback wird im ISR Kontext ausgeführt
- ▶ Callback so kurz wie möglich
- ▶ Reentrancy!
- ▶ Tick "Dauer" sind System abhängig

# Triggers != Events

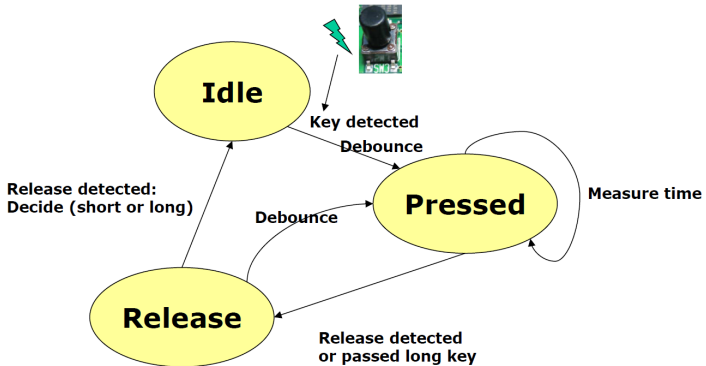
- ▶ Trigger → Reagieren in  $x[s]$  (periodisch)
- ▶ Events → Reagieren auf Ereignisse (asynchron)



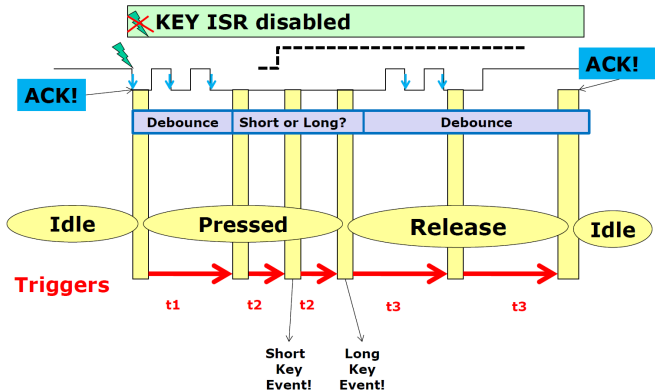
# Grund für Endprellen



# Idee



# Debounce FSM



# Timing

- ▶ Interrupt → Event → Scan
- ▶ Zeitdauer bis FSM den Key Value überprüft kann gross sein.
- ▶ Ideal: Scan im Interrupt

# Quiz

- ▶ Welches sind die drei Parameter der Trigger Struktur?

# Quiz

- ▶ Welches sind die drei Parameter der der Trigger Struktur?

```
typedef void (*TRG_Callback) (void*);  
  
typedef struct {  
    uint16_t triggerTick;  
    TRG_Callback callback;  
    void *data;  
} TriggerDesc;
```

Wann

Wo

Wie

# Quiz

- ▶ Welches sind die drei Parameter der der Trigger Struktur?

```
typedef void (*TRG_Callback) (void*);  
  
typedef struct {  
    uint16_t triggerTick;  
    TRG_Callback callback;  
    void *data;  
} TriggerDesc;
```

Wann

Wo

Wie

- ▶
- ▶ Du möchtest mit einem Trigger eine blockierende ADC Messung machen. Die Messung dauert 4.5[ms]. Wird das funktionieren?

# Quiz

- ▶ Welches sind die drei Parameter der der Trigger Struktur?

```
typedef void (*TRG_Callback) (void*);  
  
typedef struct {  
    uint16_t triggerTick;  
    TRG_Callback callback;  
    void *data;  
} TriggerDesc;
```

Wann

Wo

Wie

- ▶
- ▶ Du möchtest mit einem Trigger eine blockierende ADC Messung machen. Die Messung dauert 4.5[ms]. Wird das funktionieren?
- ▶ **Nein. Dies würde das System blockieren, da der Callback im ISR Kontext ausgeführt wird.**



# Quiz

- ▶ Du möchtest mit einem Taster eine LED toggeln. Beim testen stellst du fest, dass das LED zufällig ein oder ausgeschaltet ist. Was könnte das Problem sein?

# Quiz

- ▶ Du möchtest mit einem Taster eine LED toggeln. Beim testen stellst du fest, dass das LED zufällig ein oder ausgeschaltet ist. Was könnte das Problem sein?
- ▶ **Taster muss entprellt werden. (SW oder HW)**

# Quiz

- ▶ Du möchtest mit einem Taster eine LED toggeln. Beim testen stellst du fest, dass das LED zufällig ein oder ausgeschaltet ist. Was könnte das Problem sein?
- ▶ **Taster muss entprellt werden. (SW oder HW)**
- ▶ Du hast nun eine FSM Entprellung implementiert. Nun hast aber ein anderes Problem. Das LED lässt sich genau einmal toggeln. Was könnte da das Problem sein?

# Quiz

- ▶ Du möchtest mit einem Taster eine LED toggeln. Beim testen stellst du fest, dass das LED zufällig ein oder ausgeschaltet ist. Was könnte das Problem sein?
- ▶ **Taster muss entprellt werden. (SW oder HW)**
- ▶ Du hast nun eine FSM Entprellung implementiert. Nun hast aber ein anderes Problem. Das LED lässt sich genau einmal toggeln. Was könnte da das Problem sein?
- ▶ **Key ISR wurde nicht wieder aktiviert.**