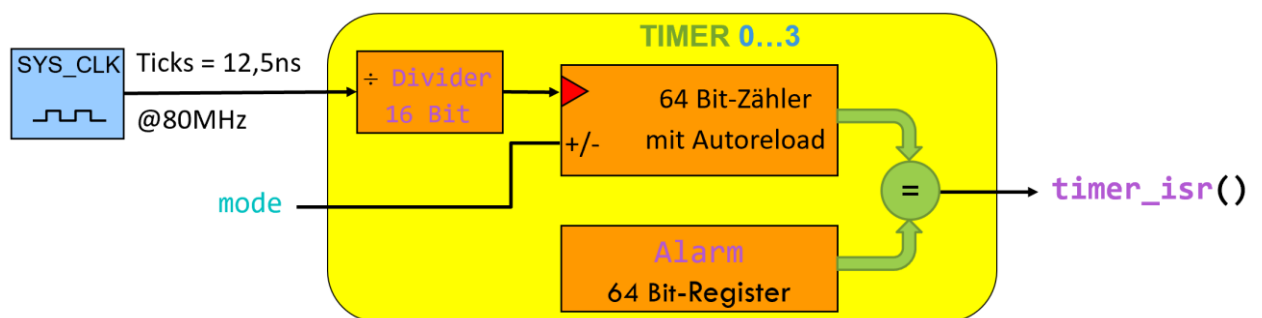
 Friedrich-Ebert-Schule Esslingen FES	IT: Hardwarenahes Programmieren	Name: Rahm Datum: 26.10.2022 1_7_3_Timer_Interrupt.docx
	Programmierung: Timer Interrupt	1.7.3.1

**Timer** sind Hardware-Zeitgeber, die unabhängig von der sonstigen Programmierung arbeiten. Sie bestehen im wesentlichen aus einer Zählerereinheit mit (internem) Taktgeber. Wird ein externer Eingang für den Takt verwendet, spricht man von einem **Counter**, da hier dann gewissermaßen die eintreffenden Impulse gezählt werden können.

Mit Timern können quatzgenaue Verzögerungszeiten (z.B. für eine Uhr) realisiert werden. In der Regel wird ein Timer zusammen mit einem Interrupt verwendet. Der Timer löst dann nach der konfigurierten Zeit einen Interrupt aus.

### Aufbau der Timereinheiten beim ESP32



Eine Timer-Einheit des ESP32 besteht aus einem einstellbaren Vorteiler (Divider, Prescaler), einem 64 Bit-Zähler und einem Vergleichs-, bzw. Alarm-Register. Der Zähler wird vom heruntergeteilten Systemtakt hochgezählt. Erreicht der Zähler den voreingestellten Wert im Alarmregister, wird der Timer-Interrupt ausgelöst. Gleichzeitig wird der Zähler wieder auf 0 gesetzt und der Zählvorgang beginnt von neuem.

Die Zeit bis zum Auslösen des Interrupts kann folgendermaßen berechnet werden:

$$t_{int} = Ticks \cdot Divider \cdot Alarm$$

Damit ergibt sich bei 80 MHz Systemtakt eine maximal mögliche Interrupt-Zeit von:

$$t_{int} = 12,5ns \cdot 2^{16} \cdot 2^{64} = 15.111.572.745.182.864 \text{ s} \approx \mathbf{480 \text{ Mio Jahre !}}$$

### Funktionen zur Timerprogrammierung in der Arduino-IDE

```

hw_timer_t *timer = timerBegin (timerNr, Divider, mode)
timer:      Pointer auf Objekt der Klasse hw_timer_t
timerNr:    0,1,2,3
Divider:    1 ... 65535
mode:       true (UP), false (DOWN)

timerAttachInterrupt ( timer, &timer_isr, Trigger )
&timer_isr: Interrupt-Service-Routine (Adresse)
Trigger:     true (EDGE), false (LEVEL)


timerAlarmWrite( timer, Alarm, AutoReload )
AutoReload:  true, false

timerAlarmEnable( timer )
timerAlarmDisable( timer )
  
```

### Definition der ISR

```

void IRAM_ATTR timer_isr( void ) { }
IRAM_ATTR: Verschiebt ISR ins interne RAM
  
```

 Friedrich-Ebert-Schule Esslingen FES	IT: Hardwarenahes Programmieren	Name: Rahm Datum: 26.10.2022 1.7.3_Timer_Interrupt.docx
	Programmierung: Timer Interrupt	1.7.3.2

Mit dem Timer0 des ESP32 soll eine LED im Sekundentakt getoggled werden. Die Timer-ISR muss daher im Sekundentakt aufgerufen werden und die LED togglen.

Für die Berechnung der Interruptzeit ist bei einem Systemtakt von 80MHz:

$$t_{int} = Ticks \cdot Divider \cdot Alarm$$

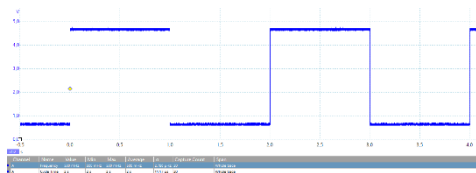
$$1s = 12,5ns \cdot 80 \cdot 1000000$$

Einer der beiden Werte Divider (16 Bit) oder Alarm (64 Bit) muss zunächst gewählt werden, dann lässt sich der fehlende Wert berechnen.

## Arbeitsauftrag

1. Programmieren und testen Sie den abgebildeten Programmcode. Analysieren Sie das Programm.

2. Messen Sie mit dem Oszilloskop die Ein- und Ausschaltzeit der LED nach.



3. Dokumentieren Sie Ihre Ergebnisse.

```
const int LEDpin = 32;
const int timerNr = 0;
const int Divider = 80;
const unsigned long Alarm = 1000000;

hw_timer_t *timer = NULL;
volatile byte led = LOW;

void IRAM_ATTR timer_isr(void)
{
  led = !led;
  digitalWrite(LEDpin, led);
}

void setup()
{
  pinMode(LEDpin, OUTPUT);
  timer = timerBegin(timerNr, Divider, true);
  timerAttachInterrupt(timer, &timer_isr, true);
  timerAlarmWrite(timer, Alarm, true);
  timerAlarmEnable(timer);
}

void loop() {}
```

