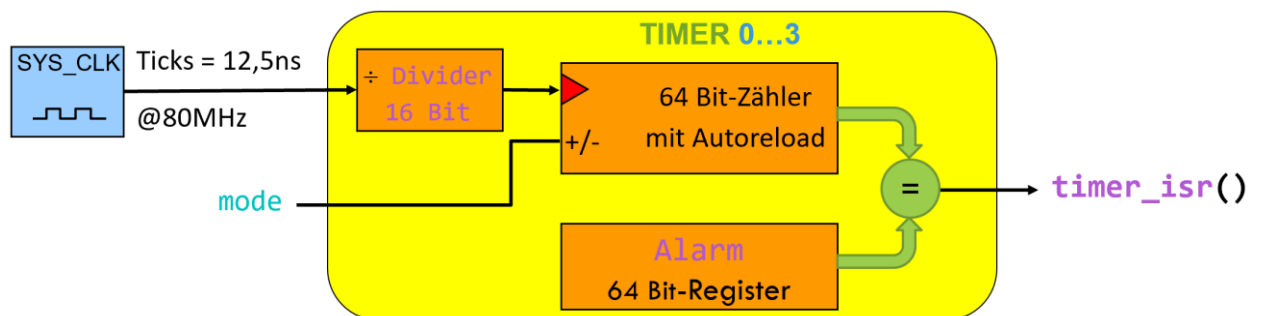


Timer sind Hardware-Zeitgeber, die unabhängig von der sonstigen Programmierung arbeiten. Sie bestehen im Wesentlichen aus einer Zählerereinheit mit (internem) Taktgeber. Wird ein externer Eingang für den Takt verwendet, spricht man von einem **Counter**, da hier dann die eintreffenden Impulse gezählt 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 64 Bit-Vergleichs-, bzw. Alarm-Register. Der Zähler wird vom heruntergeteilten Systemtakt hochgezählt. Die Zählfrequenz ergibt sich zu:

$$frequency = \frac{SYS_CLK}{Divider} = \frac{80\text{ MHz}}{Divider}$$

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:

$$t_{int} = \frac{Alarm}{frequency}$$

Damit ergeben sich bei 80 MHz Systemtakt unterschiedlichste Einstellmöglichkeiten für t_{int} :

Divider 16 Bit	Frequency	$t_{int} @ Alarm$				
		1	1000	10.000	1.000.000	2^{64}
2	40.000.000 Hz	25 ns	25 µs	0,250 ms	2,5 ms	14623 Jahre
3	26.666.666 Hz	37,5 ns	37,5 µs	0,375 ms	37,5 ms	21935 Jahre
8	10.000.000 Hz	100 ns	0,1 ms	1 ms	0,1 s	29247 Jahre
80	1.000.000 Hz	1 µs	1 ms	10 ms	1 s	584942 Jahre
800	100.000 Hz	10 µs	10 ms	100 ms	10 s	5,849 Mio J
8000	10.000 Hz	100 µs	100 ms	1000 ms = 1 s	100 s	58,49 Mio J
65520	1.221 Hz	819 µs	819 ms	8,19 s	819 s	ca. 480 Mio J

Jitter ca. 10-30ns

 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

Funktionen zur Timerprogrammierung in der Arduino-IDE

Erzeugung eines Handles und Start des Timers:

```
hw_timer_t *timer = timerBegin (frequency)
timer: Handle auf Objekt der Klasse hw_timer_t
frequency: 1.221 Hz ... 40.000.000 Hz
```

Tatsächliche Frequenz zurücklesen:

```
frequency = timerFrequency( timer )
```

Interrupt-Service-Routine mit Handle verbinden:

```
timerAttachInterrupt ( timer, &timer_isr)
&timer_isr: Interrupt-Service-Routine (Pointer)
```

Alarm-Wert schreiben und Autoreload aktivieren:

```
timerAlarm( timer, Alarm, AutoReload, reloadCount )
Alarm: 2 ... 264
AutoReload: true, false (= One Shot)
reloadCount: 0
```

Timer starten und stoppen. Zählwert bleibt erhalten:


```
timerStart( timer )
timerStop( timer )
```

Zähler auf 0 zurücksetzen:

```
timerRestart( timer )
```

Timer-Interrupt-Routine:

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

 Friedrich-Ebert-Schule Esslingen FES	IT: Hardwarenahes Programmieren	Name: Rahm Datum: 26.10.2022 1.7.3_Timer_Interrupt.docx
	Programmierung: Timer Interrupt	1.7.3.3

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

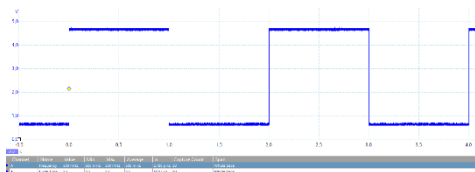
Für die Interruptzeit $t_{int} = 1s$ wird **frequency** und **Alarm** auf 1.000.000 gesetzt:

$$t_{int} = \frac{Alarm}{frequency} = \frac{1.000.000}{1.000.000} = 1s$$

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 unsigned long Alarm = 1'000'000;
const unsigned long frequency = 1'000'000;

hw_timer_t *timer1 = NULL;
volatile byte led = LOW;

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

void setup()
{
  pinMode(LEDpin,OUTPUT);

  timer1 = timerBegin(frequency);
  timerAttachInterrupt(timer1,&timer_isr);
  timerAlarm(timer1,Alarm,true,0);
}

void loop() {}
```

