Warten mit Timer

Basic Timer TIM6 und TIM7

Ich bin Mik, Dein Mikrocontroller

Warten mit Timer

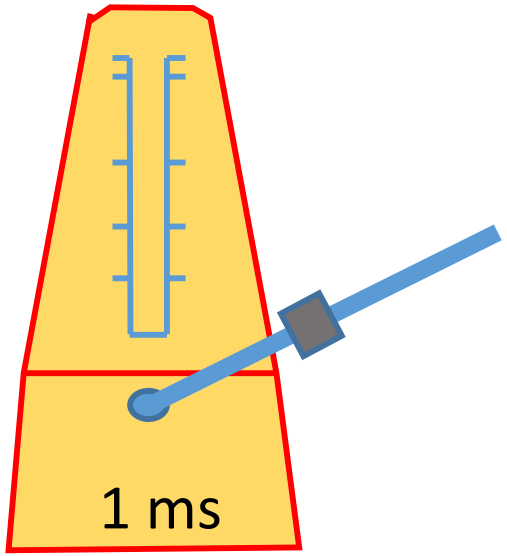
Warte 1 Sekunde

Der Auftrag

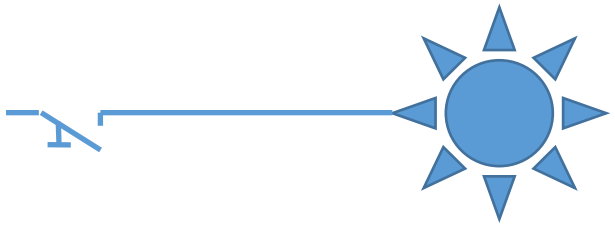
Warten mit Timer

1000 ms

0



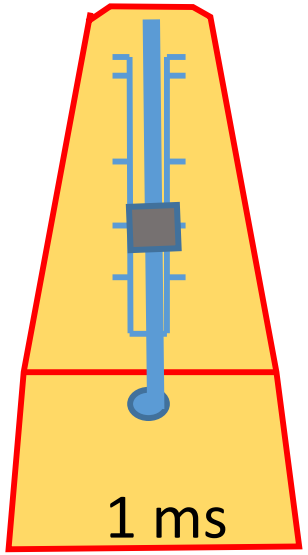
Licht



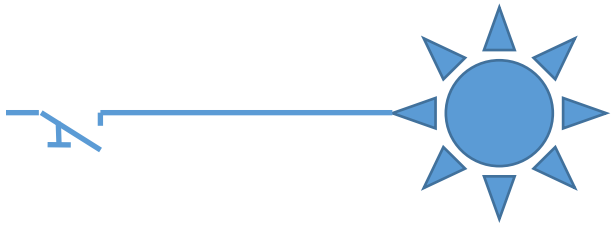
Warten mit Timer

1000 ms

1



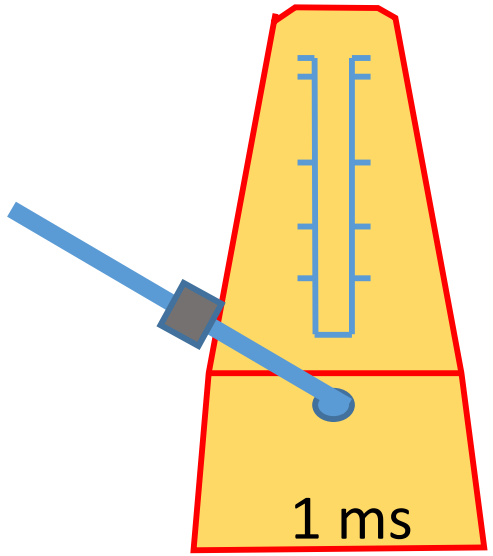
Licht



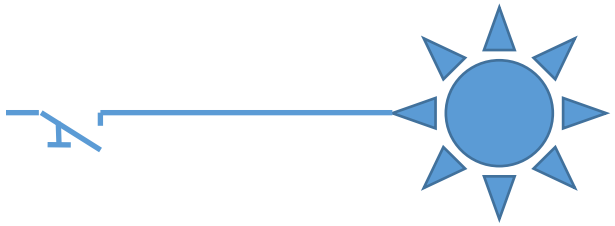
Warten mit Timer

1000 ms

1



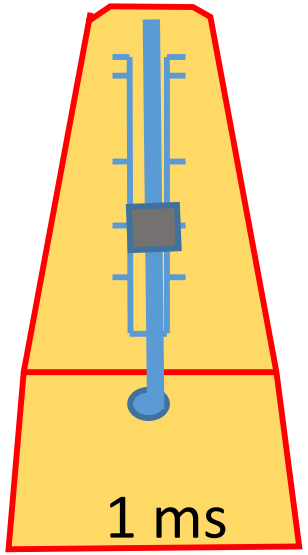
Licht



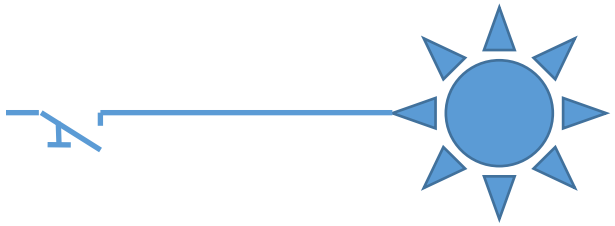
Warten mit Timer

1000 ms

1



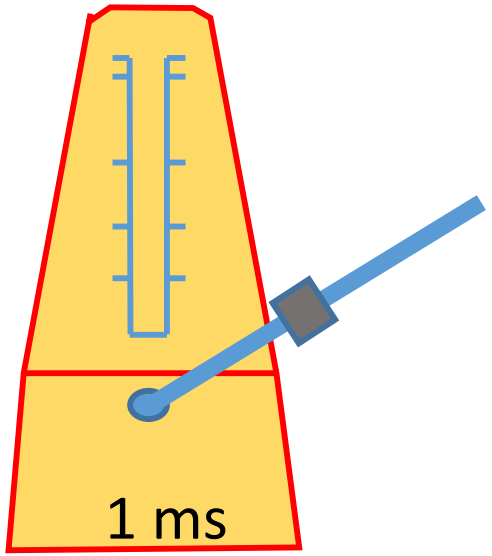
Licht



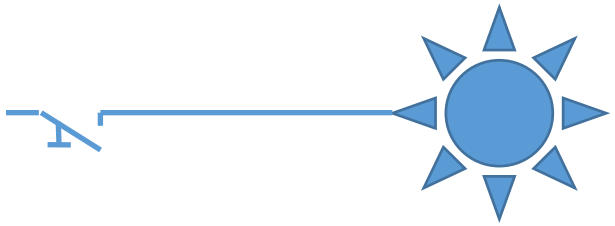
Warten mit Timer

1000 ms

1



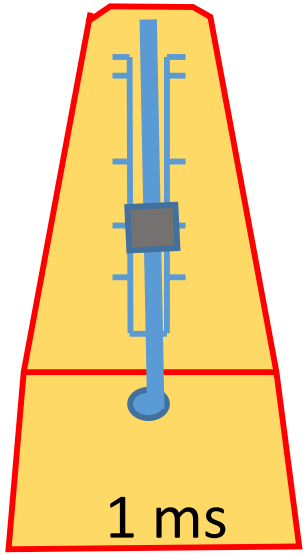
Licht



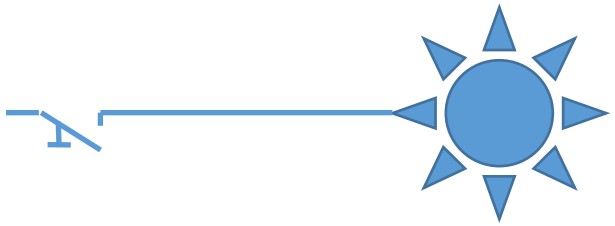
Warten mit Timer

1000 ms

2



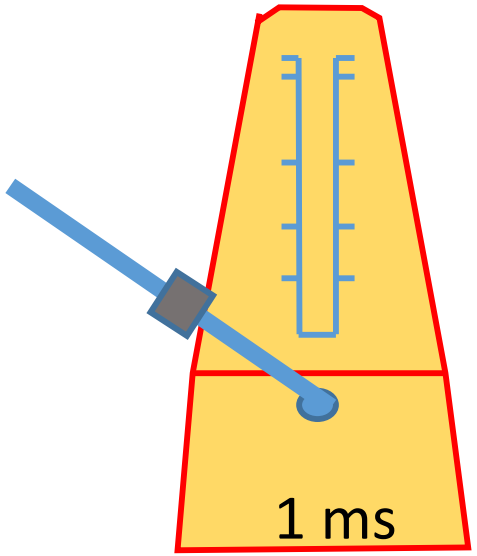
Licht



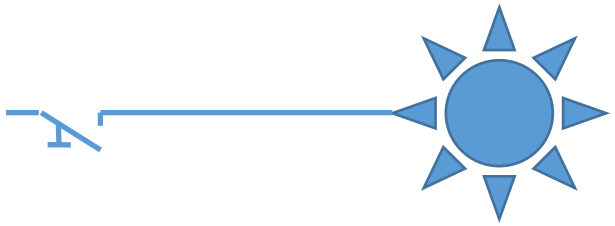
Warten mit Timer

1000 ms

2



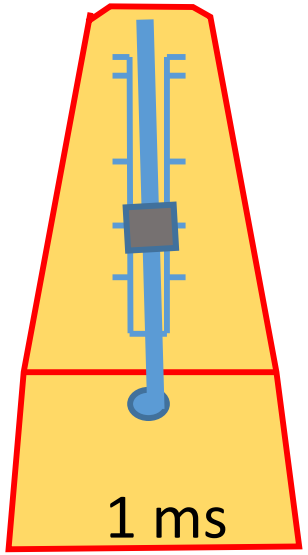
Licht



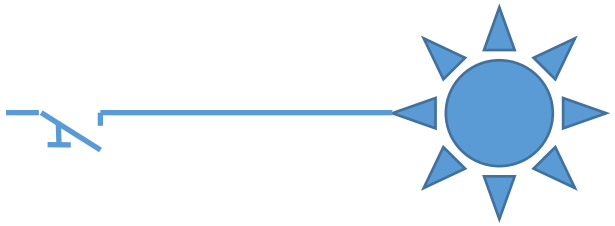
Warten mit Timer

1000 ms

2



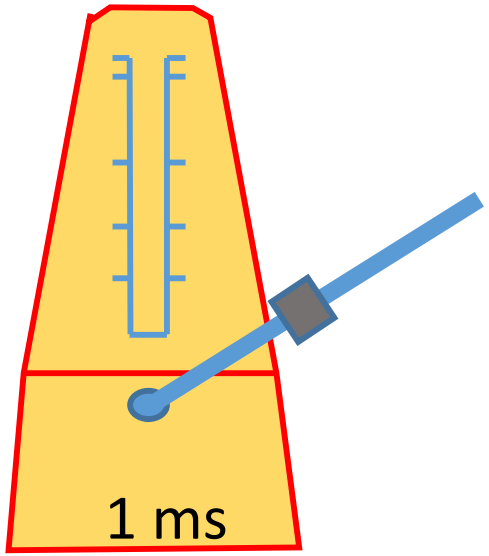
Licht



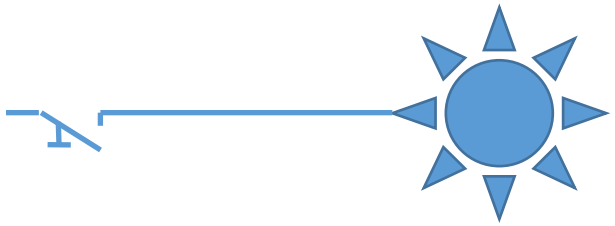
Warten mit Timer

1000 ms

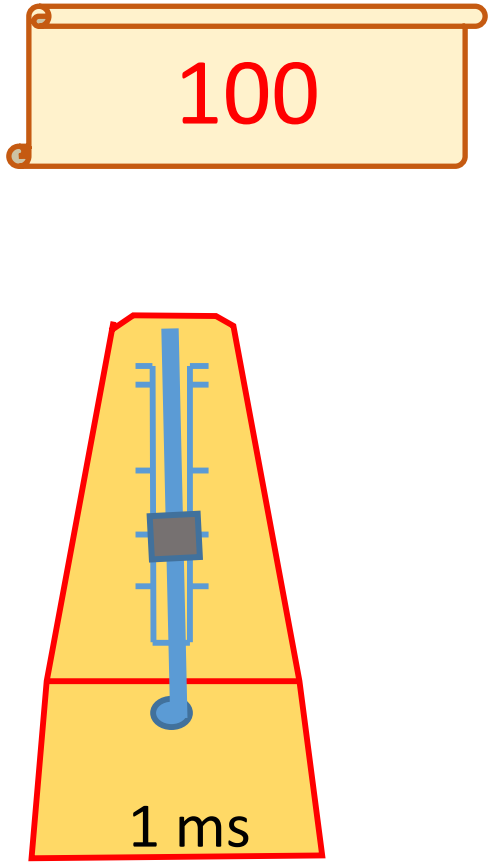
2

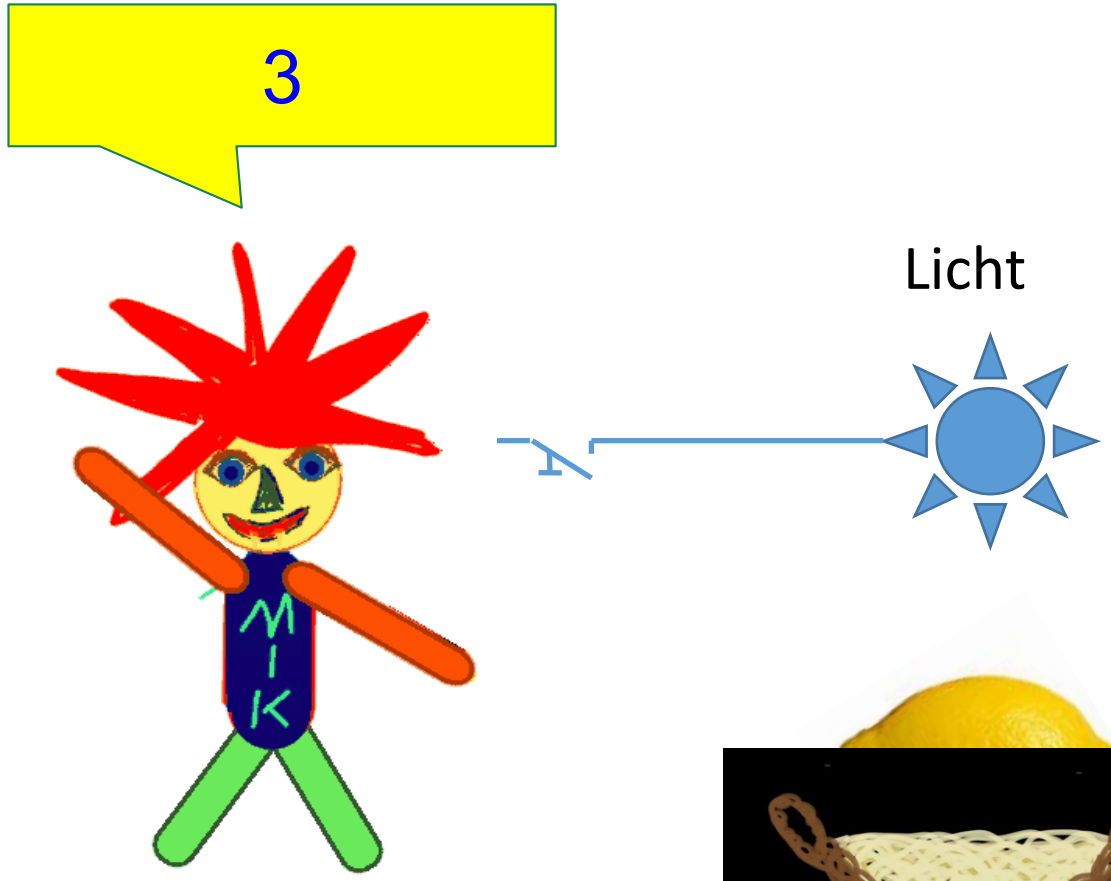


Licht



Warten mit Timer

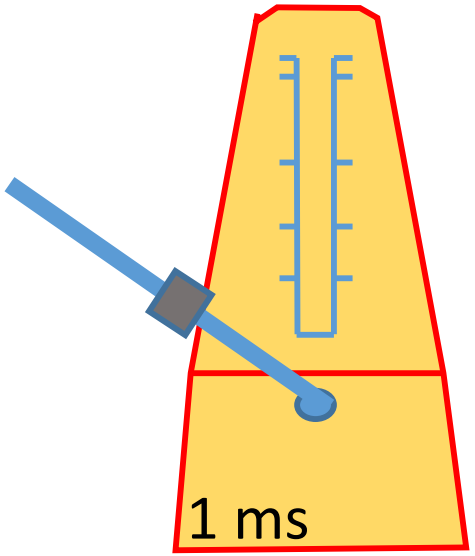




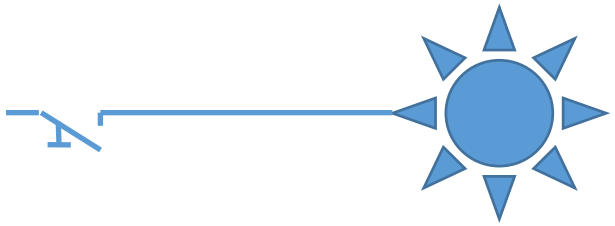
Warten mit Timer

1000 ms

3



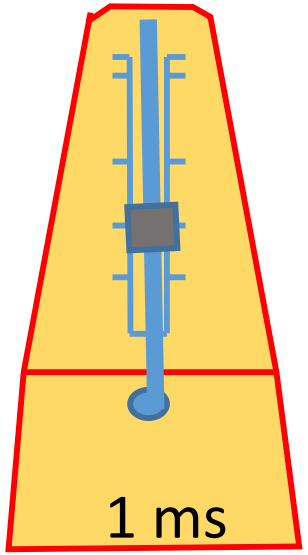
Licht



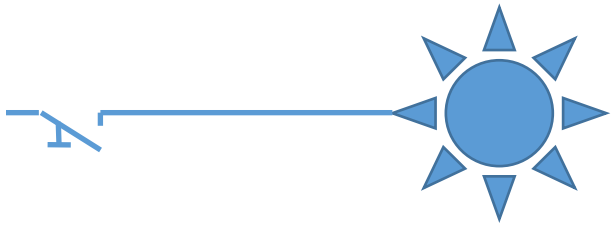
Warten mit Timer

1000 ms

3



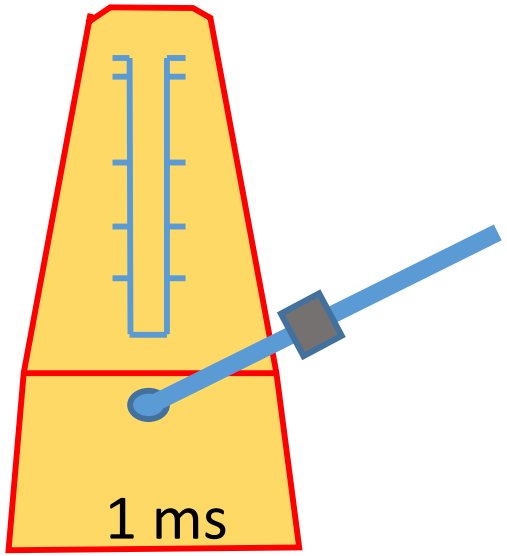
Licht



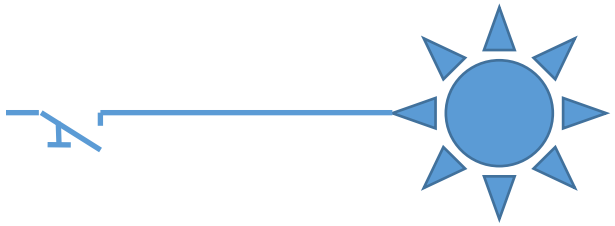
Warten mit Timer

1000 ms

3



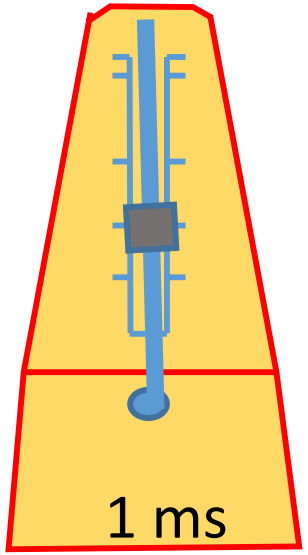
Licht



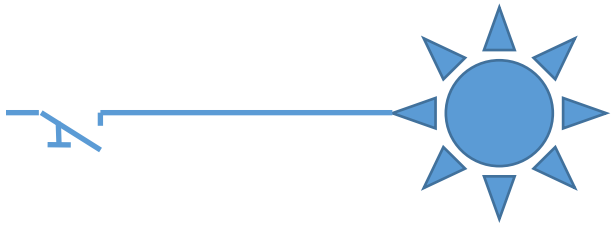
Warten mit Timer

1000ms

4



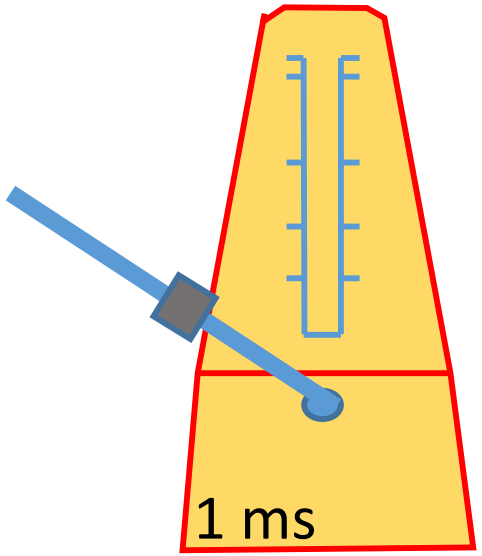
Licht



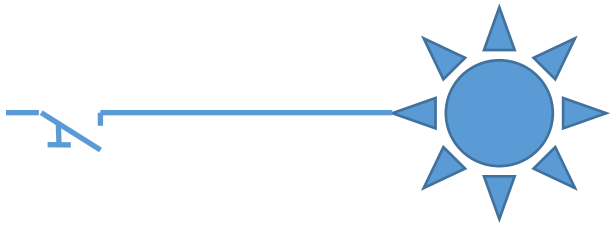
Warten mit Timer

1000 ms

4



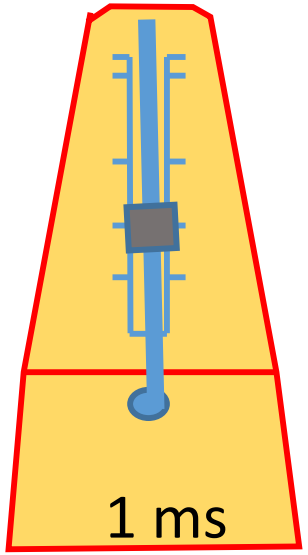
Licht



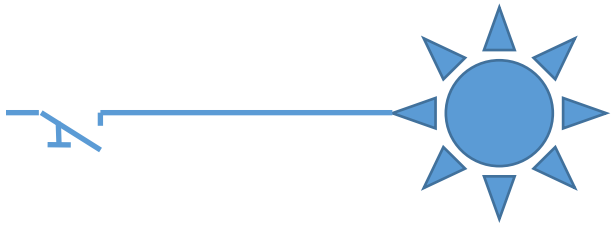
Warten mit Timer

1000 ms

4



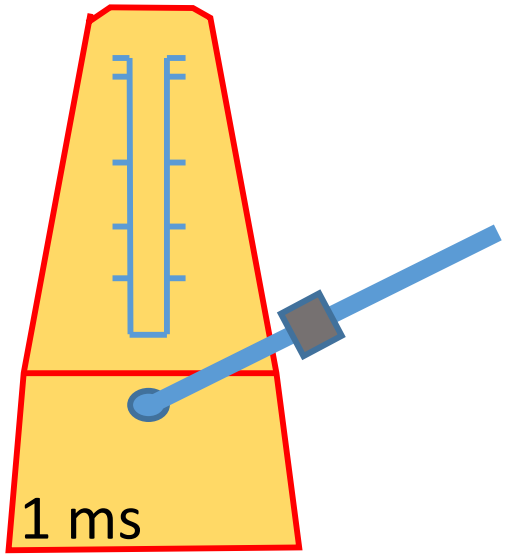
Licht



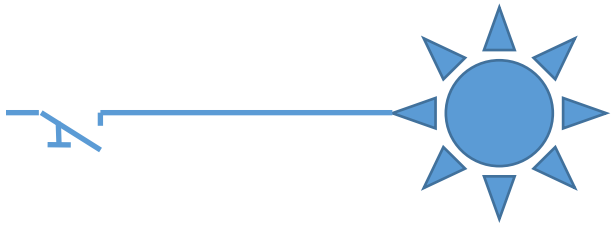
Warten mit Timer

1000 ms

4



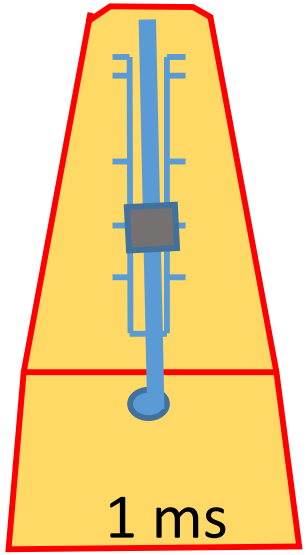
Licht



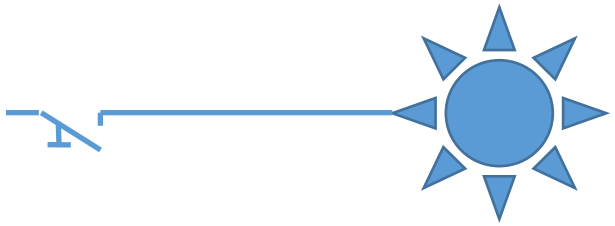
Warten mit Timer

1000 ms

5



Licht

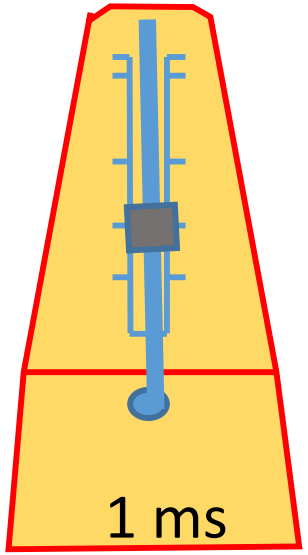


Warten mit Timer

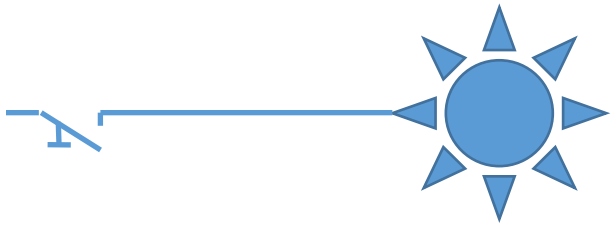
994 ms später

1000 ms

999



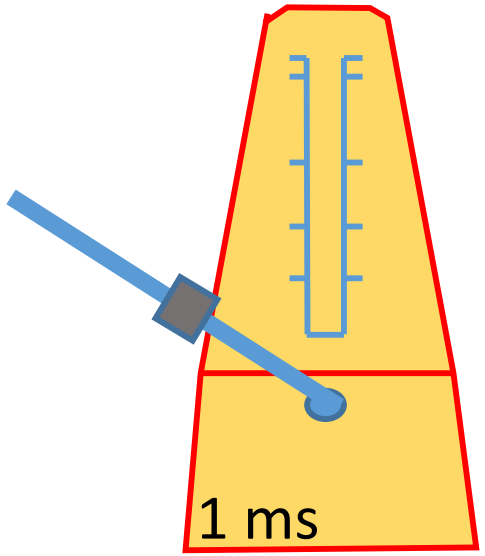
Licht



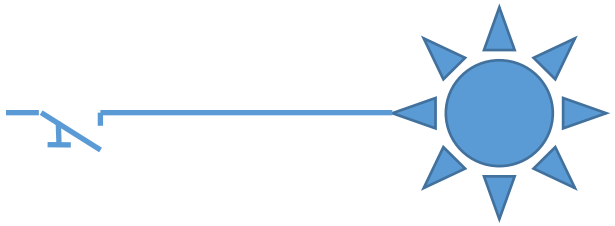
Warten mit Timer

1000 ms

999



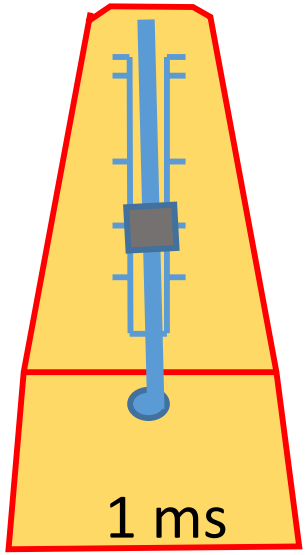
Licht



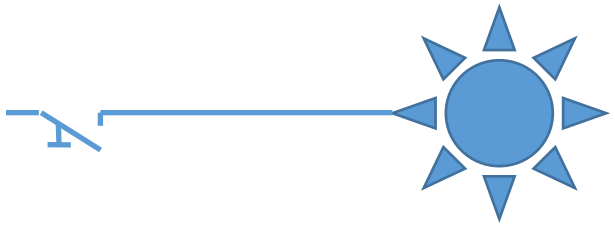
Warten mit Timer

1000 ms

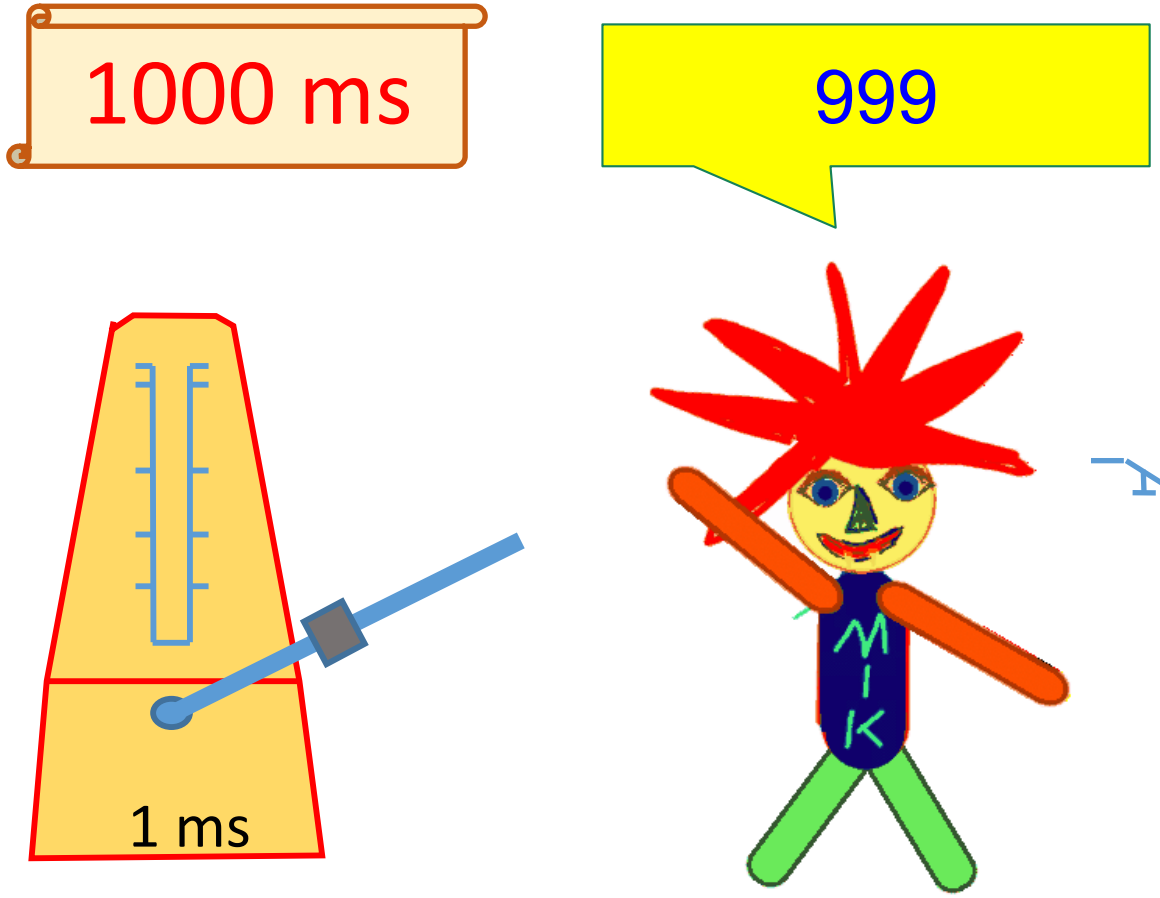
999

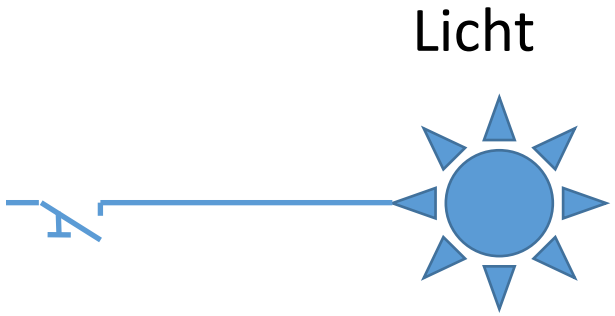


Licht



Warten mit Timer

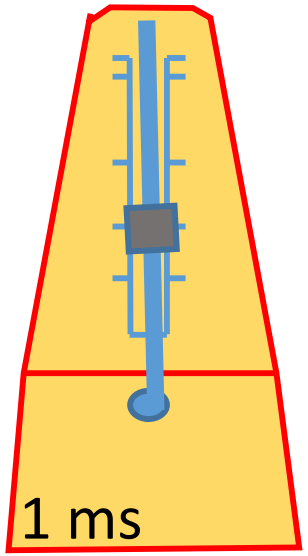




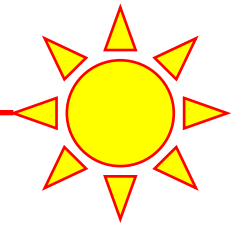
Warten mit Timer

1000 ms

1000



Licht

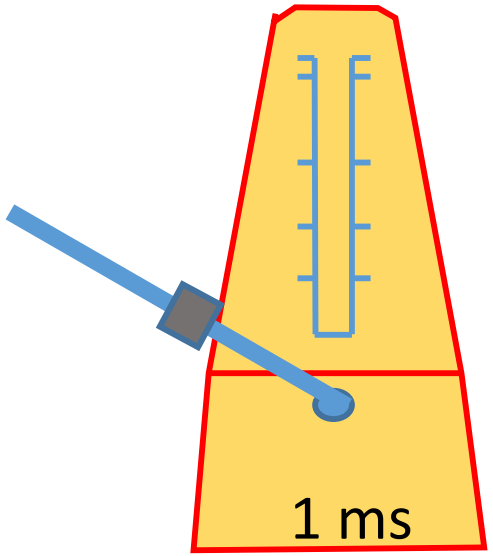


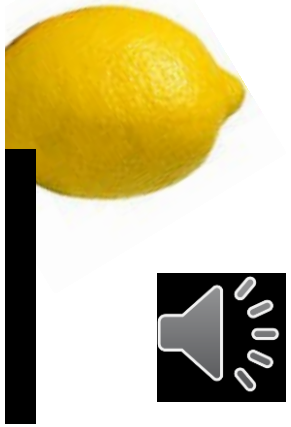
Warten mit Timer

1000 ms

Stopp

Licht

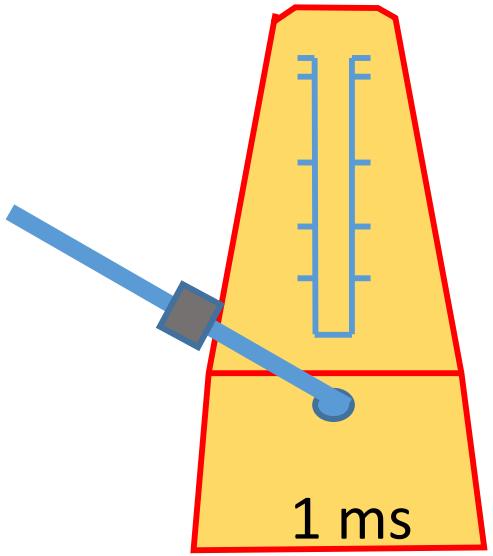




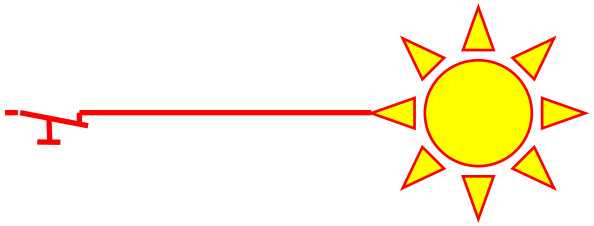
Warten mit Timer

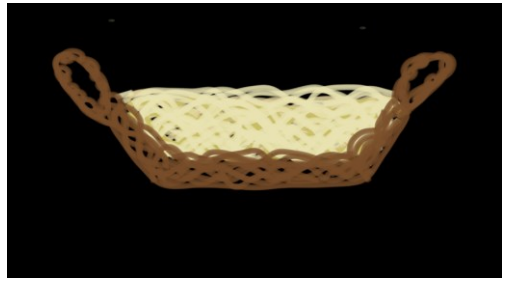
1000 ms

Stopp



Licht

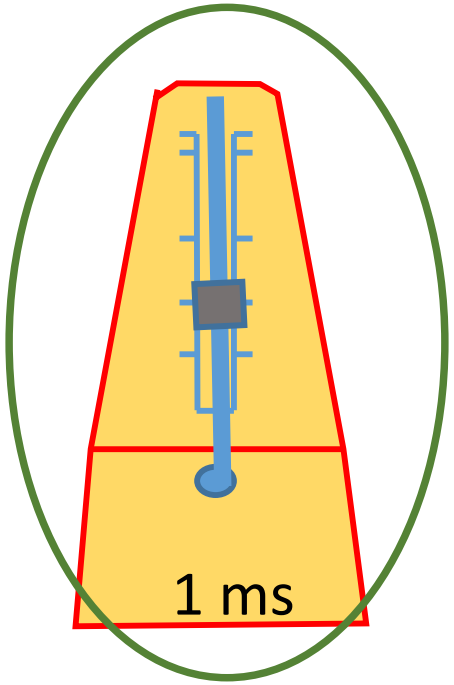




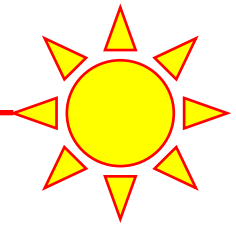
Warten mit Timer

1000 ms

5



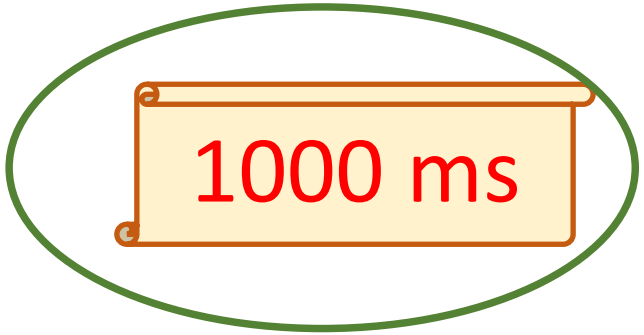
Licht

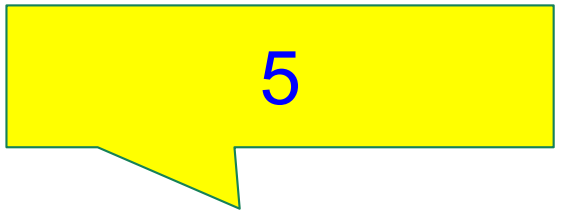


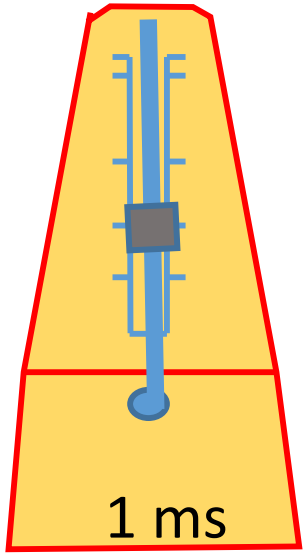
Prescaler PSC => Zählgeschwindigkeit

Warten mit Timer

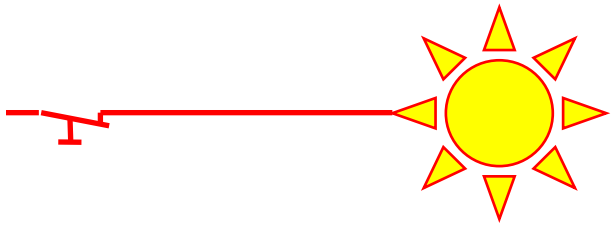
Autoreload Register ARR: So weit zählen

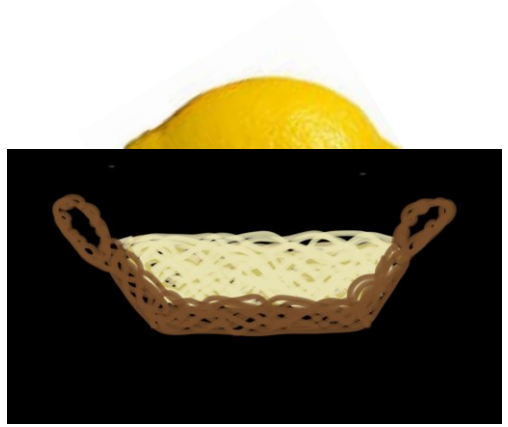






Prescaler PSC



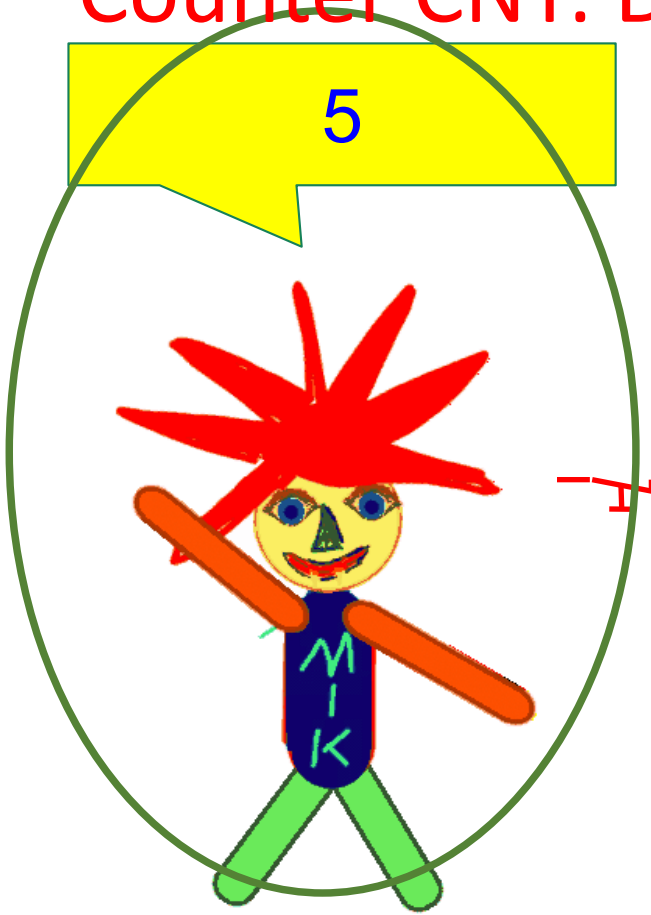


Warten mit Timer

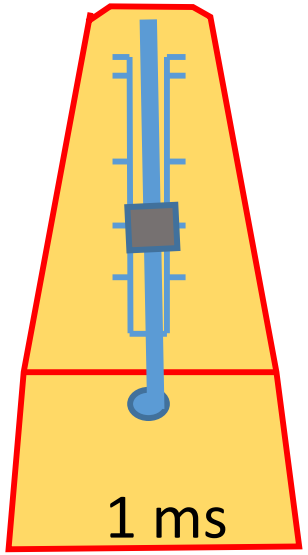
Autoreload Register ARR

Counter CNT: Der Zähler

1000 ms



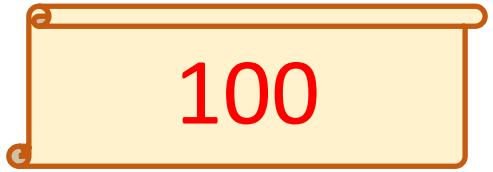
Licht



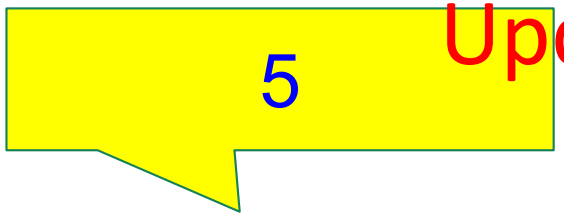
Prescaler PSC

Warten mit Timer

Autoreload Register ARR

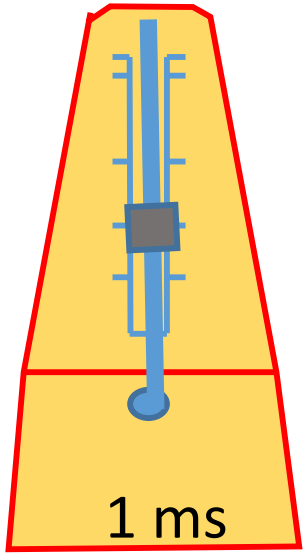


Counter CNT

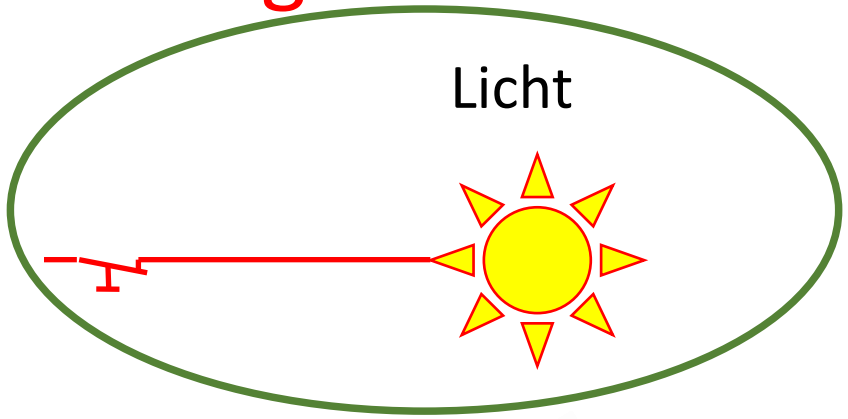


Update Interrupt Flag UIF

Zeigt Überlauf an



Prescaler PSC



Warten mit Timer

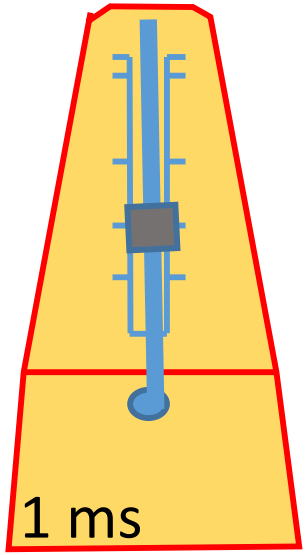
Autoreload Register ARR

1000 ms

Counter CNT

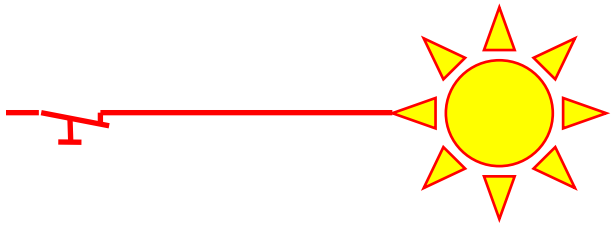
5

Update Interrupt Flag UIF



Prescaler PSC

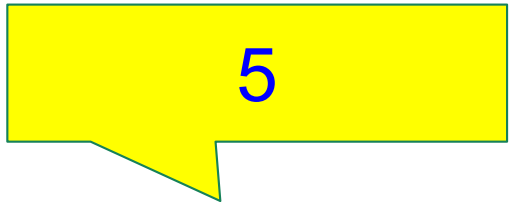
Licht

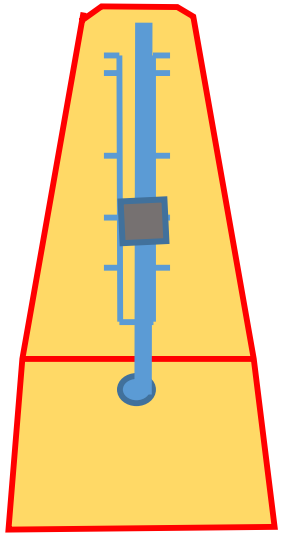


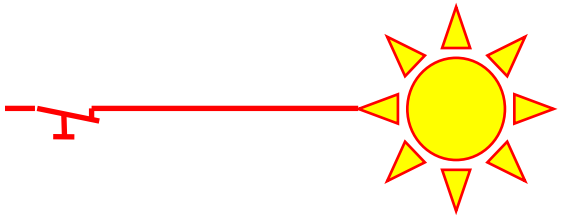
Counter Enable CEN:

Schaltet zählen ein

Warten mit Timer

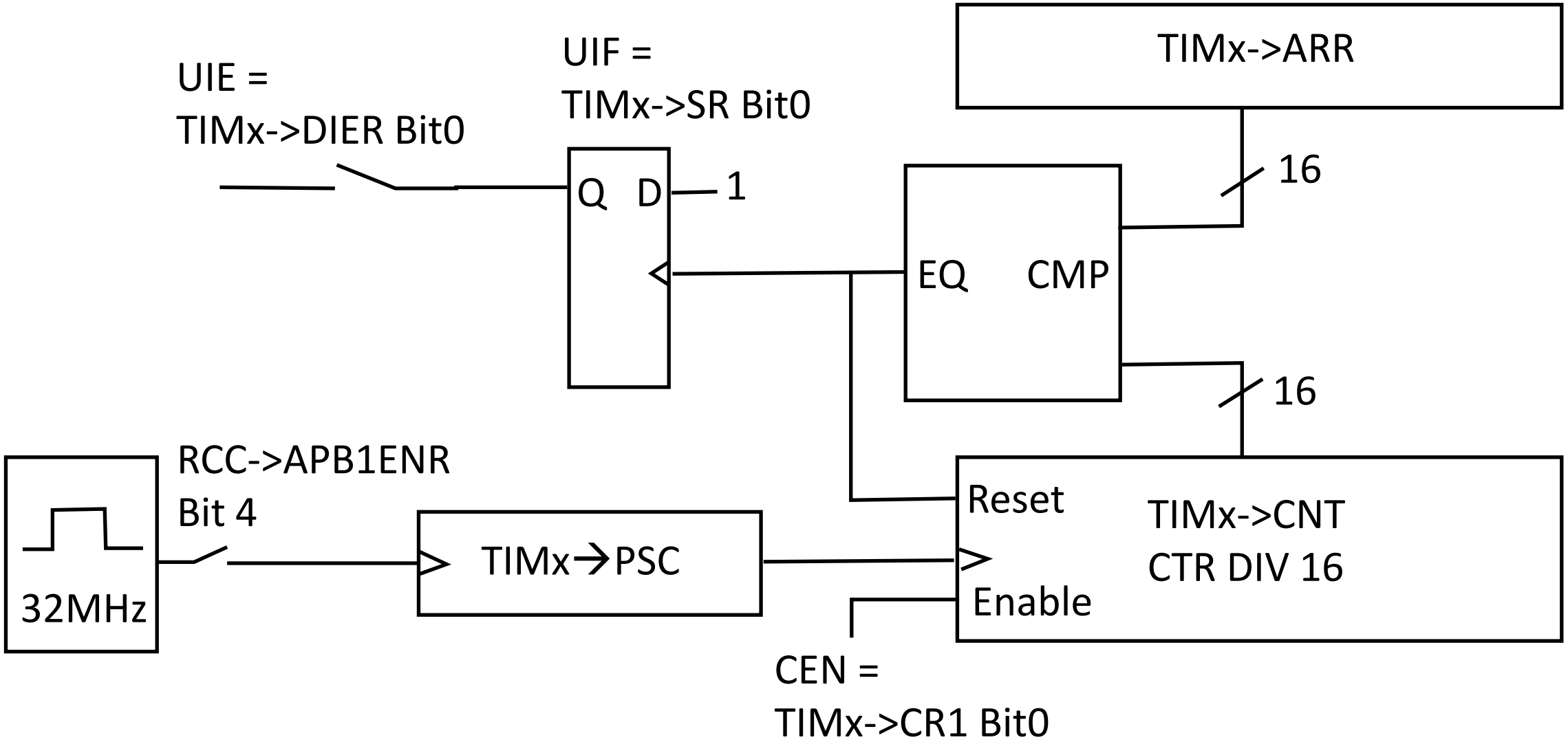




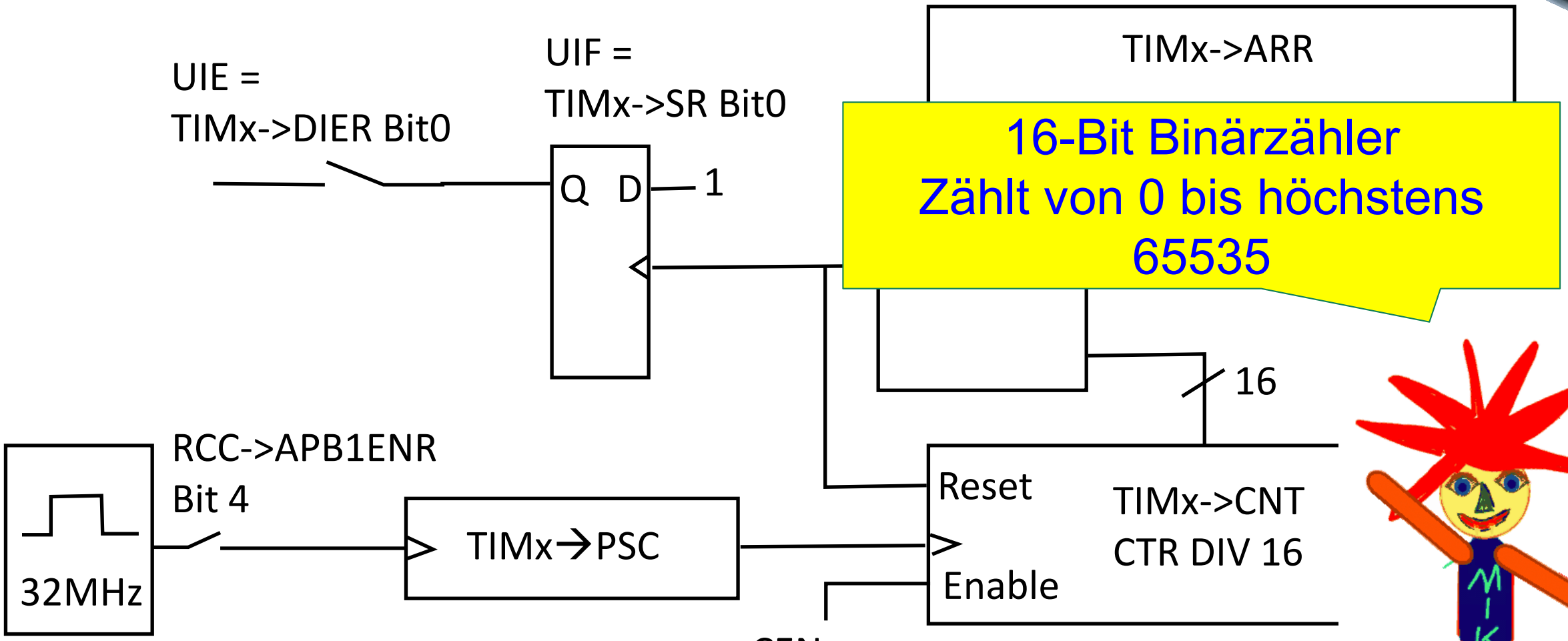


Basic Timer TIM6 oder TIM7

Timer

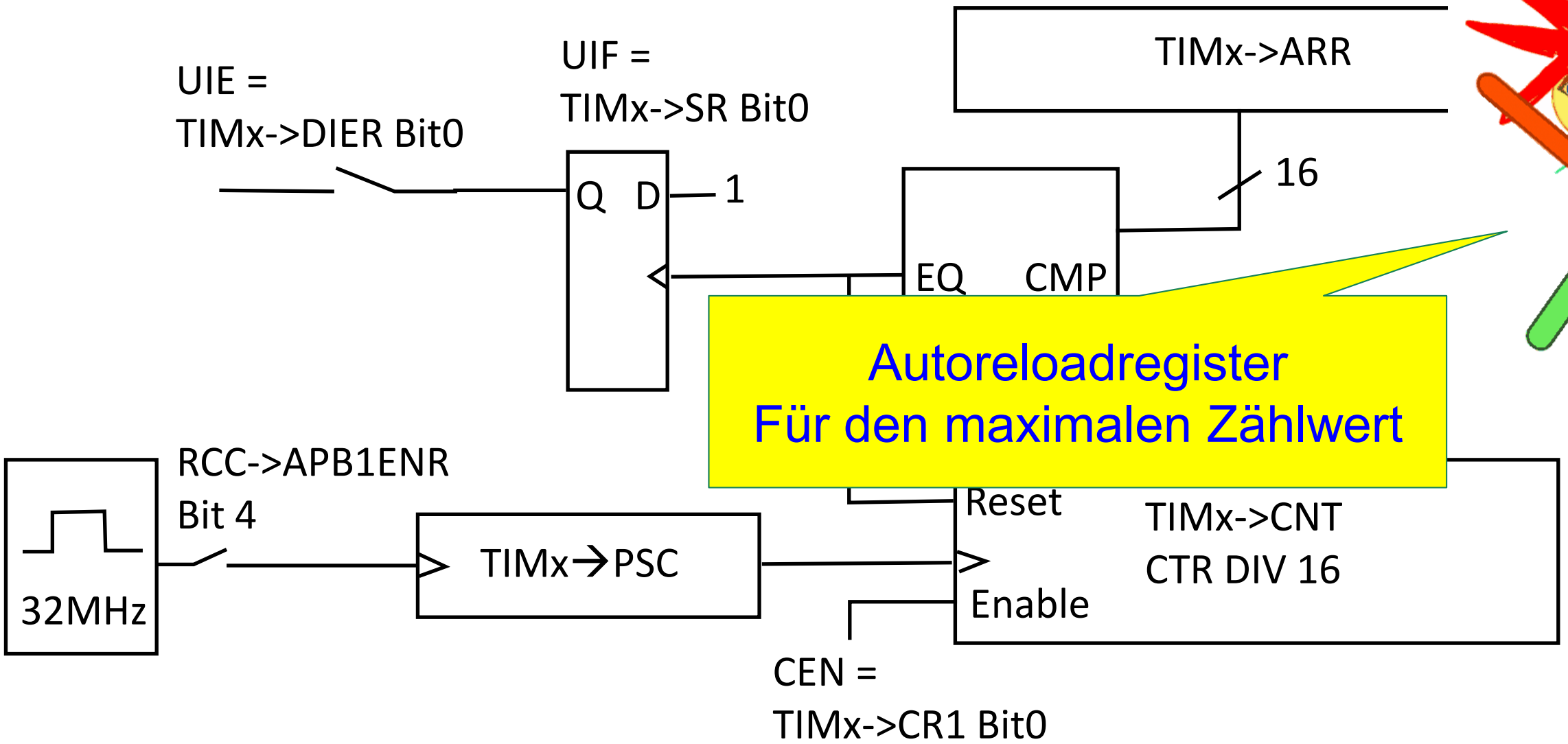

UIE =
TIMx->DIER Bit0
UIF =
TIMx->SR Bit0
Q
D
1
TIMx->ARR
16
EQ
CMP
16
RCC->APB1ENR
Bit 4
32MHz
TIMx→PSC
Reset
Enable
TIMx->CNT
CTR DIV 16
CEN =
TIMx->CR1 Bit0

Timer


UIE =
TIMx->DIER Bit0
UIF =
TIMx->SR Bit0
Q D
1
TIMx->ARR
16-Bit Binärzähler
Zählt von 0 bis höchstens
65535
16
RCC->APB1ENR
Bit 4
32MHz
TIMx→PSC
Reset
Enable
TIMx->CNT
CTR DIV 16
CEN =
TIMx->CR1 Bit0

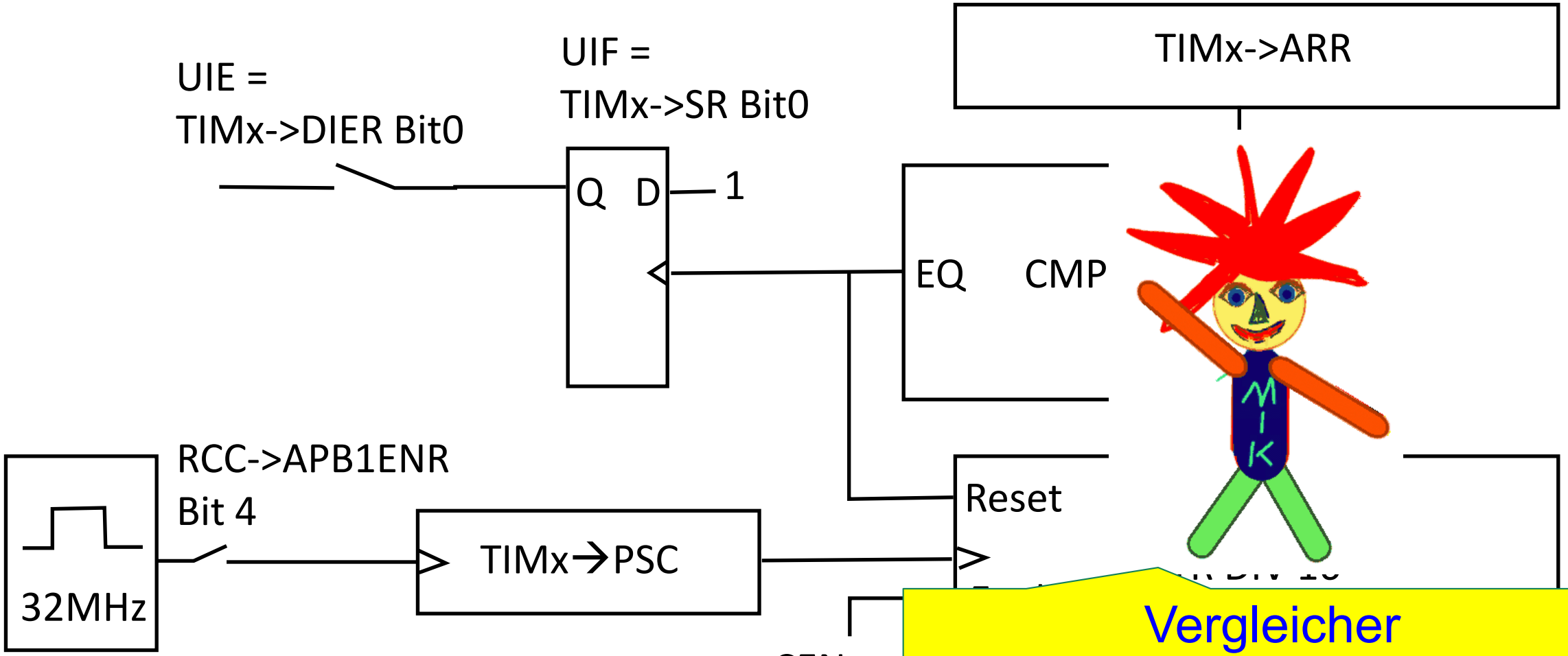
MIK

Timer


UIE =
TIMx->DIER Bit0
UIF =
TIMx->SR Bit0
Q
D
1
TIMx->ARR
16
EQ
CMP
Autoreloadregister
Für den maximalen Zählwert
RCC->APB1ENR
Bit 4
32MHz
TIMx→PSC
Reset
Enable
TIMx->CNT
CTR DIV 16
CEN =
TIMx->CR1 Bit0

MIK

Timer



Vergleicher
Wenn ARR=CNT dann von vorne

Timer

UIE =
TIMx->DIER Bit0

UIF =
TIMx-

Q D

TIMx->ARR

16

Der Überlauf wird im UIF
Update Interrupt Flag
angezeigt

RCC->APB1ENR
Bit 4

32MHz

TIMx→PSC

Reset

Enable

TIMx->CNT
CTR DIV 16

CEN =
TIMx->CR1 Bit0

Timer

UIE =
TIMx->DIER Bit0

UIF =
TIMx->SR Bit0

Q D

1

TIMx->ARR

16

EQ CMP

16

Mit Counter Enable
CEN=1 wird der Counter
freigegeben

RCC->APB1ENR
Bit 4

32MHz

TIMx→

Reset

Enable

TIMx->CNT
CTR DIV 16

EN =
IMx->CR1 Bit0

Timer

UIE =
TIMx->DIER Bit0

UIF =
TIMx->SR Bit0

TIMx->ARR

16

EQ CMP

16

Der Prescaler stellt die Zählgeschwindigkeit ein:
31 = 1µs
31999 = 1ms

Bit

32MHz

TIMx→PSC

Reset

Enable

TIMx->CNT
CTR DIV 16

CEN =
TIMx->CR1 Bit0

Timer

UIE =
TIMx->D

Der Timer muss mit 32MHz Takt versorgt werden um arbeiten zu können

TIMx->ARR

16

MP

16

RCC->APB1ENR
Bit

32MHz

TIMx→PSC

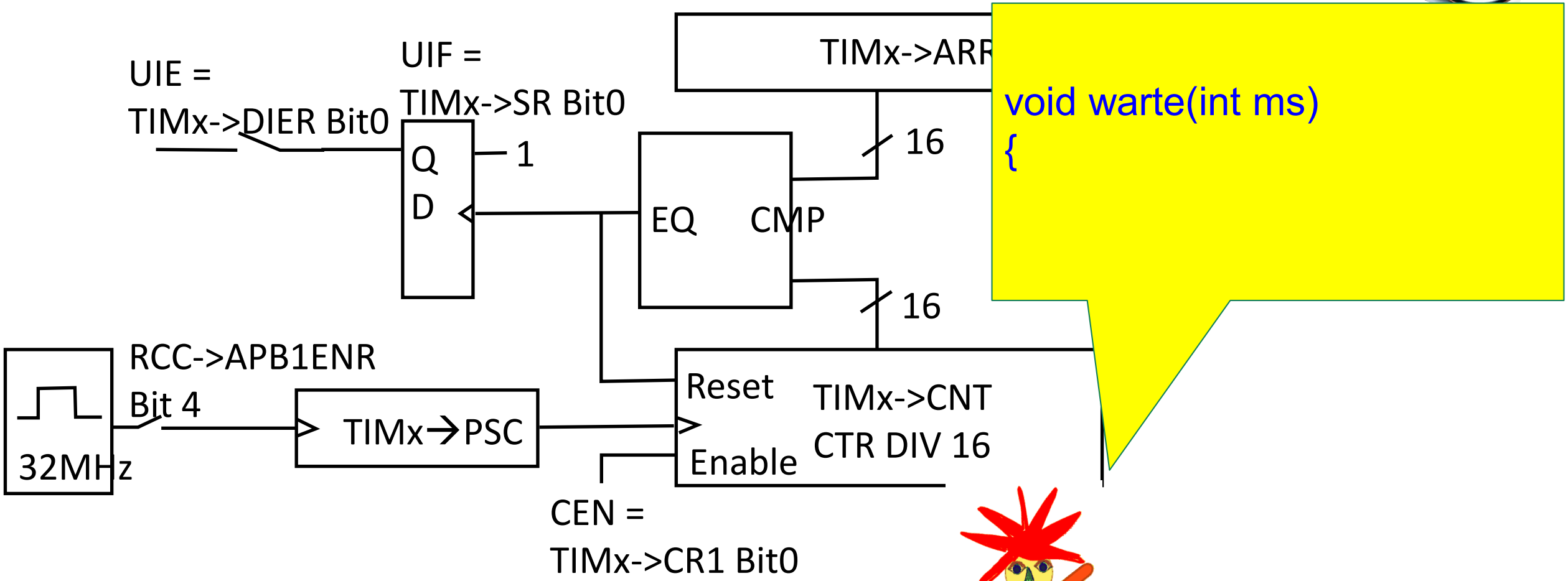
Reset

Enable

TIMx->CNT
CTR DIV 16

CEN =
TIMx->CR1 Bit0

Timer




```
void warte(int ms)
{
```

Timer

UIE =
TIMx->DIER Bit0

UIF =
TIMx->SR Bit0

Q
D

1

TIMx->ARR

16

EQ CMP

16

RCC->APB1ENR
Bit 4

32MHz

TIMx→PSC

Reset

Enable

TIMx->CNT
CTR DIV 16

CEN =
TIMx->CR1 Bit0

```
void warte(int ms)
{
RCC->APB1ENR |=0b10000;
```

Timer

UIE =
TIMx->DIER Bit0

UIF =
TIMx->SR Bit0

Q
D

1

TIMx->ARR

16

EQ CMP

16

RCC->APB1ENR
Bit 4

32MHz

TIMx→PSC

Reset

Enable

TIMx->CNT
CTR DIV 16

CEN =
TIMx->CR1 Bit0

```
void warte(int ms)
{
RCC->APB1ENR |=0b10000;
TIM6->PSC=31999; //Takt 1ms
```

Timer

UIE =
TIMx->DIER Bit0

UIF =
TIMx->SR Bit0

Q
D

1

TIMx->ARR

16

EQ CMP

16

RCC->APB1ENR
Bit 4

32MHz

TIMx→PSC

Reset

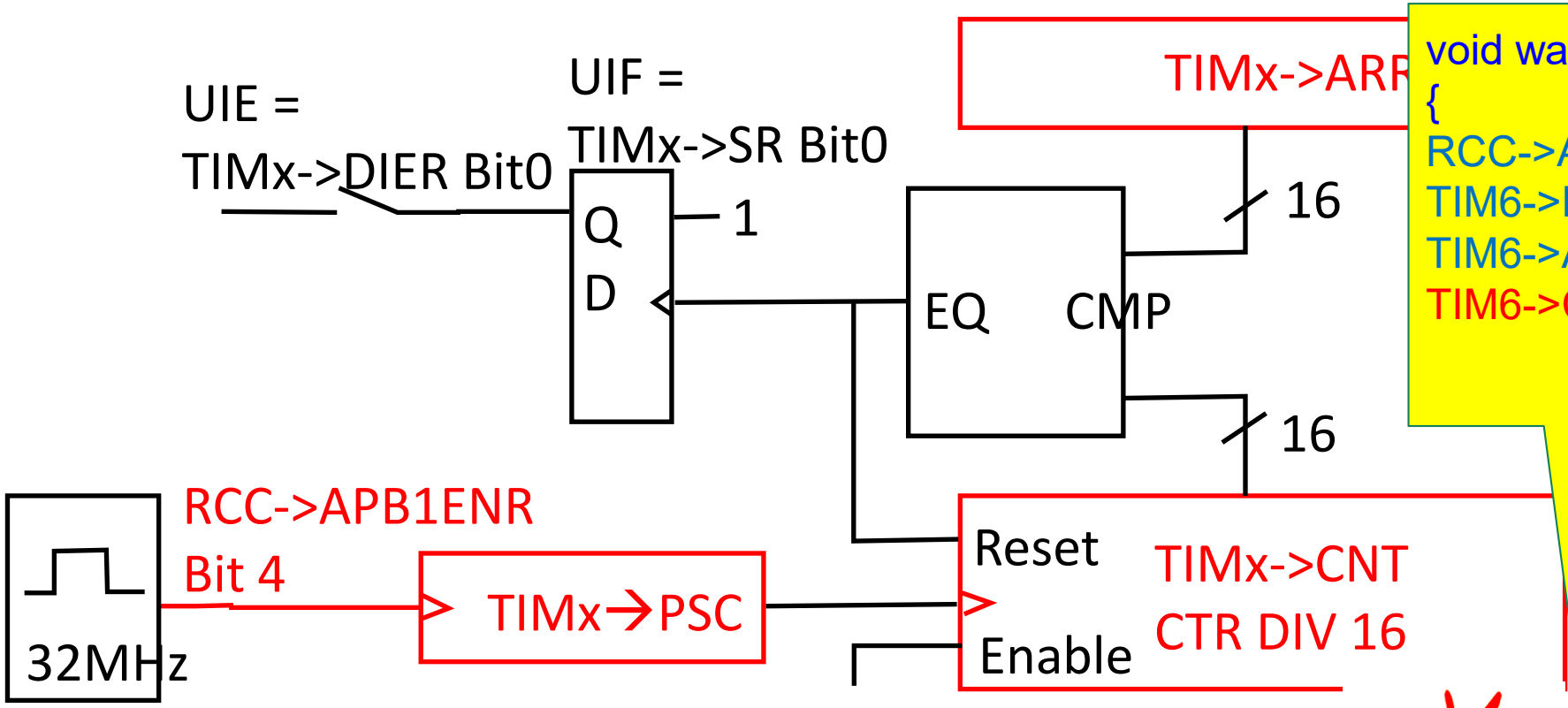
Enable

TIMx->CNT
CTR DIV 16

CEN =
TIMx->CR1 Bit0

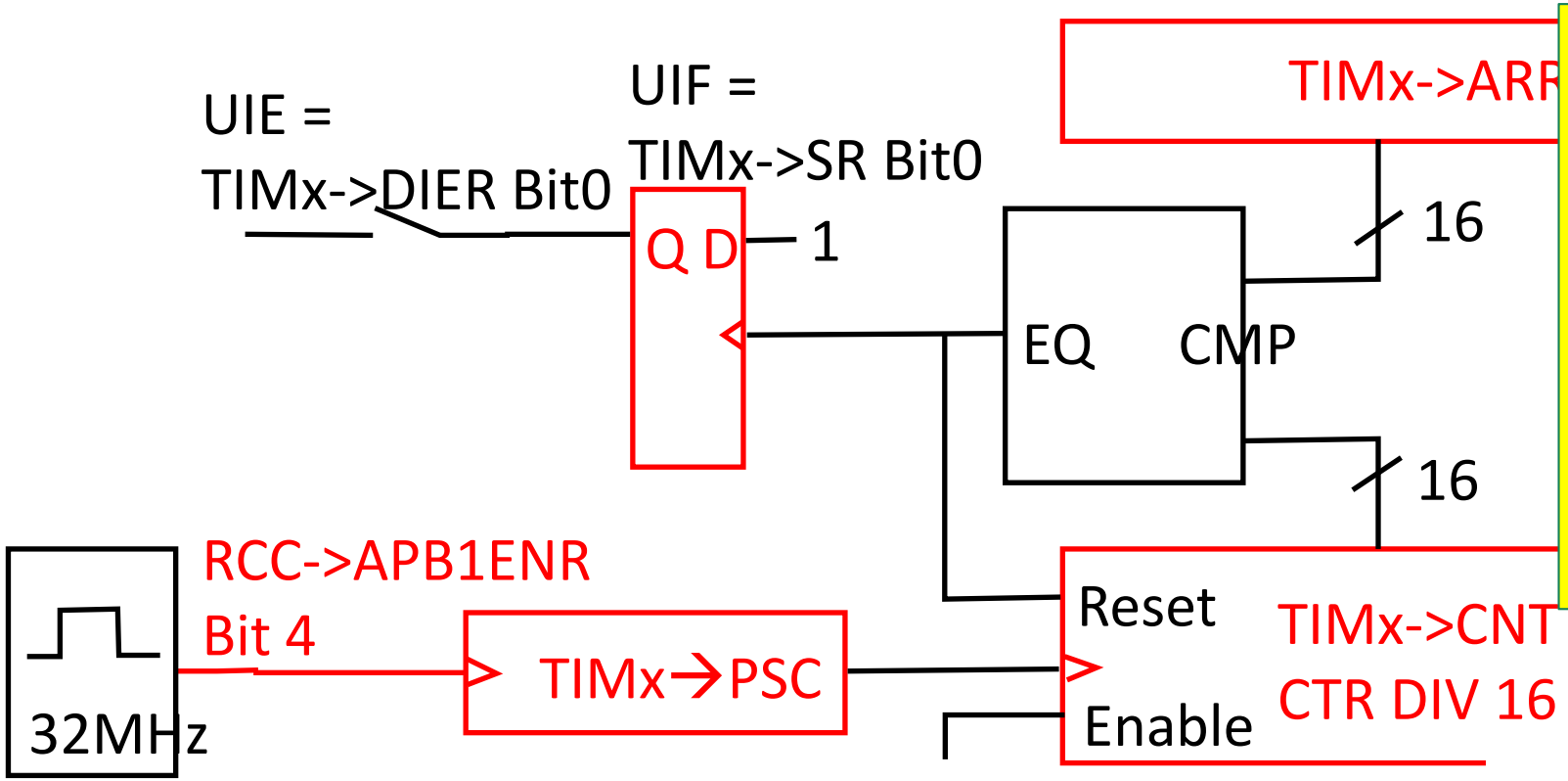
```
void warte(int ms)
{
RCC->APB1ENR |=0b10000;
TIM6->PSC=31999; //Takt 1ms
TIM6->ARR=ms;  //ms warten
```

Timer



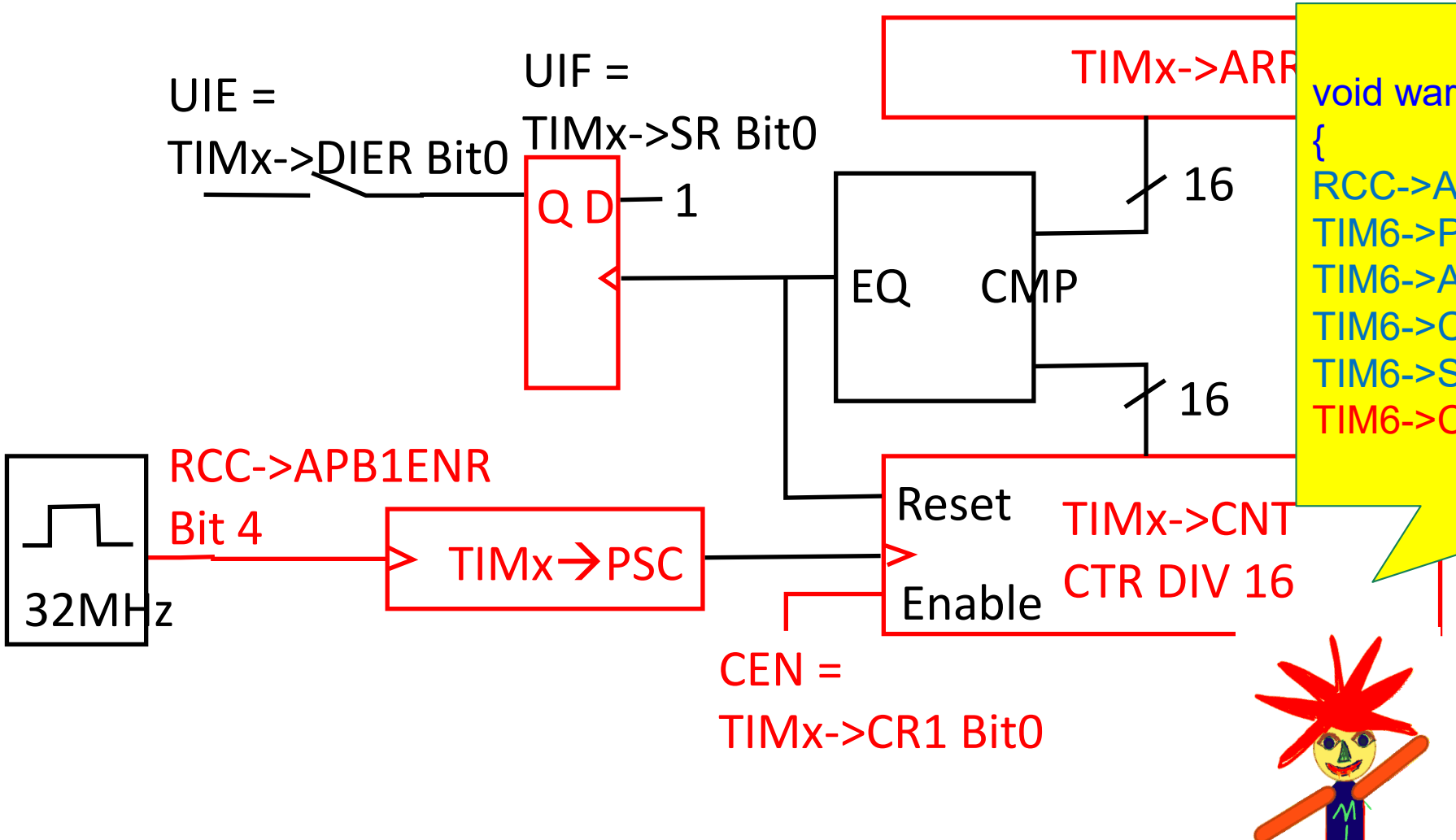

```
void warte(int ms)
{
RCC->APB1ENR |=0b10000;
TIM6->PSC=31999; //Takt 1ms
TIM6->ARR=ms;  //ms warten
TIM6->CNT=0; //bei 0 beginnen
```

Timer



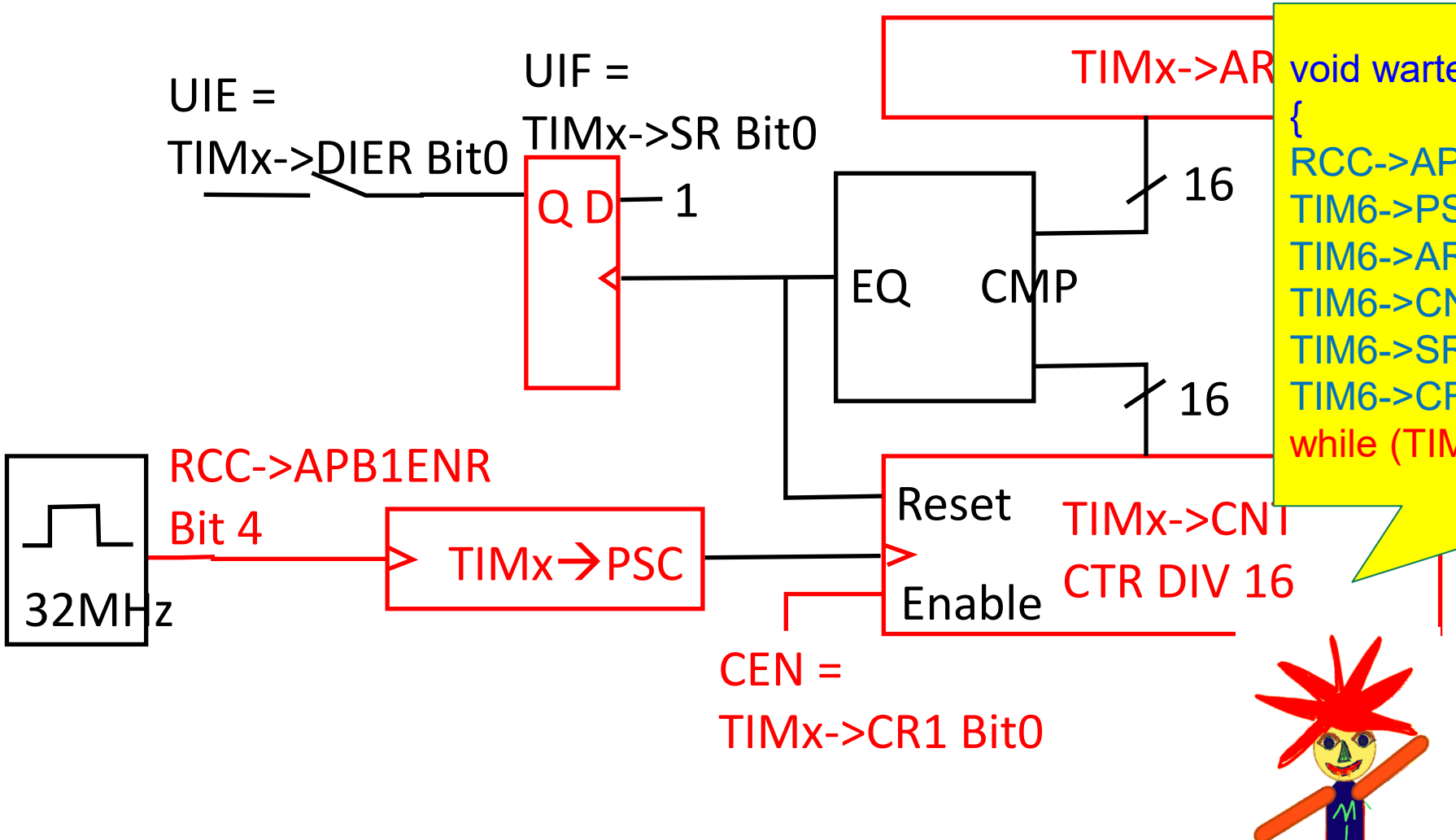

```
void warte(int ms)
{
RCC->APB1ENR |=0b10000;
TIM6->PSC=31999; //Takt 1ms
TIM6->ARR=ms;  //ms warten
TIM6->CNT=0; //bei 0 beginnen
TIM6->SR=0;   //UIF = 0
```

Timer



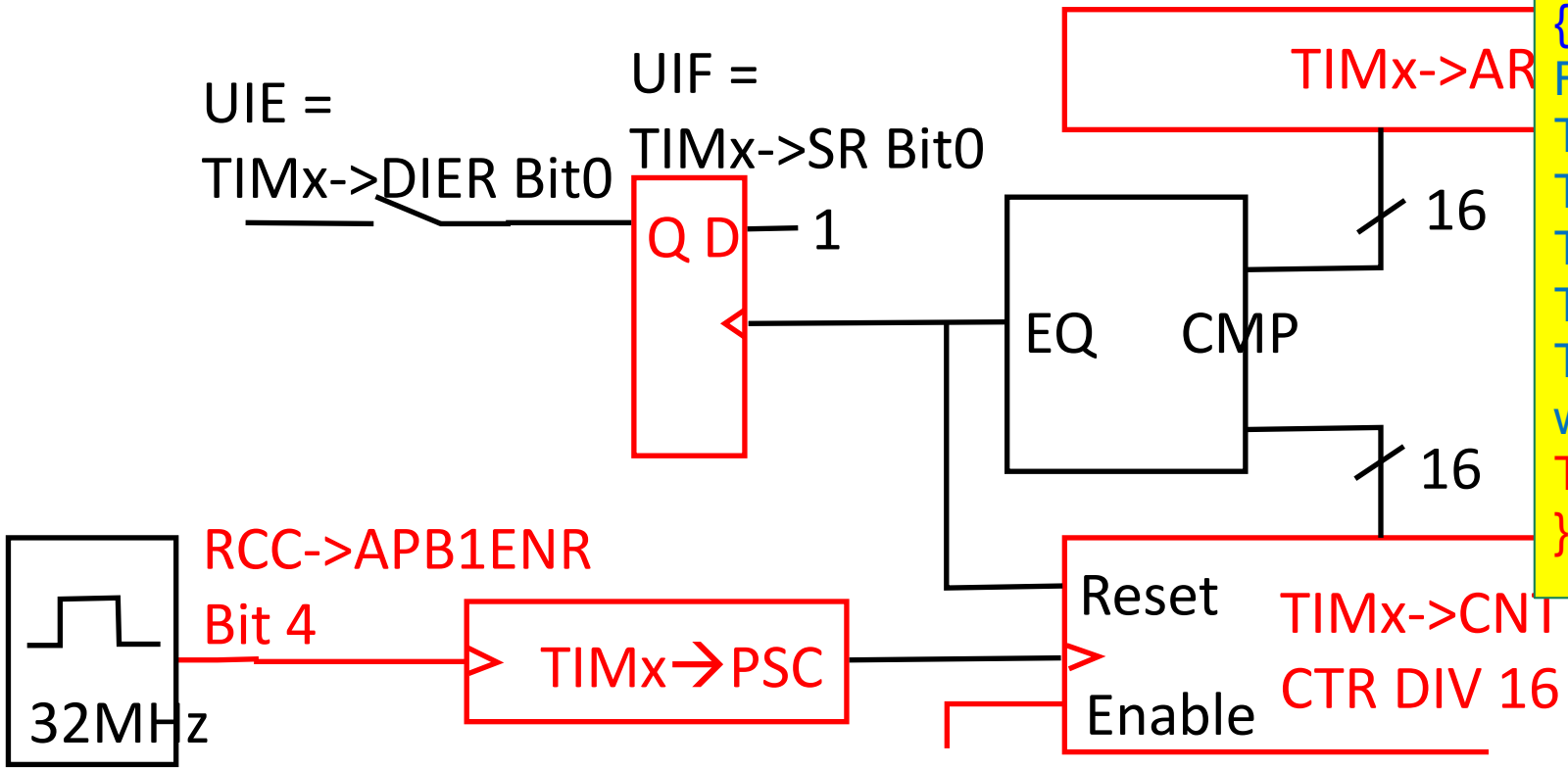

```
void warte(int ms)
{
RCC->APB1ENR |=0b10000;
TIM6->PSC=31999; //Takt 1ms
TIM6->ARR=ms;  //ms warten
TIM6->CNT=0; //bei 0 beginnen
TIM6->SR=0;   //UIF = 0
TIM6->CR1=1; //CEN=1 start
```

Timer



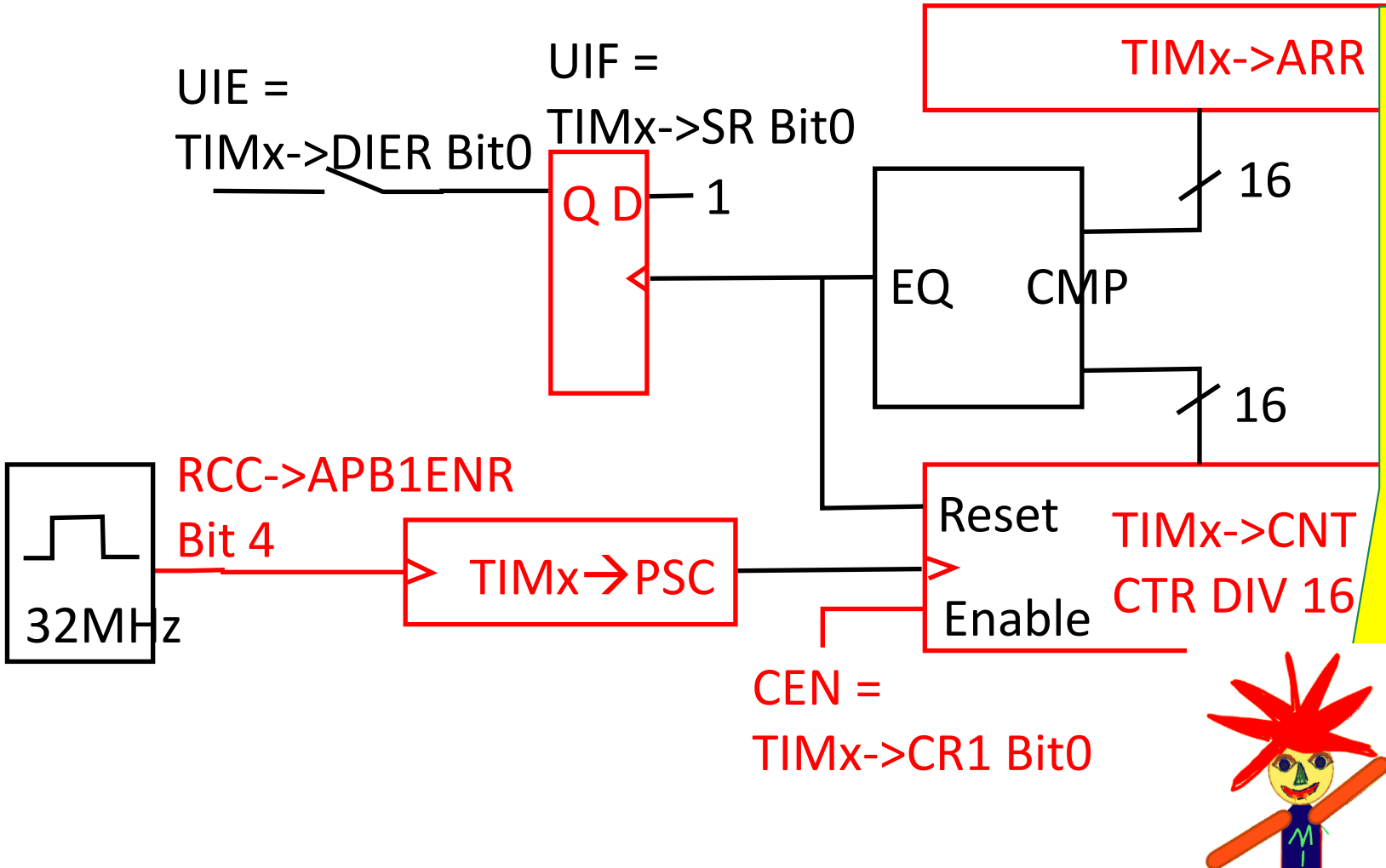

```
void warte(int ms)
{
RCC->APB1ENR |=0b10000;
TIM6->PSC=31999; //Takt 1ms
TIM6->ARR=ms;  //ms warten
TIM6->CNT=0; //bei 0 beginnen
TIM6->SR=0;   //UIF = 0
TIM6->CR1=1; //CEN=1 start
while (TIM6->SR==0); //Warten auf UIF=1
```

Timer




```
void warte(int ms)
{
RCC->APB1ENR |=0b10000;
TIM6->PSC=31999; //Takt 1ms
TIM6->ARR=ms;  //ms warten
TIM6->CNT=0; //bei 0 beginnen
TIM6->SR=0;   //UIF = 0
TIM6->CR1=1; //CEN=1 start
while (TIM6->SR==0); //Warten auf UIF=1
TIM6->CR1=0; //CEN=0 stop
}
```

Timer




```
Hauptprogramm:
int main()
{
  init(); //Timer intialisieren
  DigitalOut LED(PC_0);
  while(true)
  {
     LED=!LED; //LED umschalten
     warte(500);
  }
}
```