



# Timer und Interrupt

Basic Timer TIM6 und TIM7



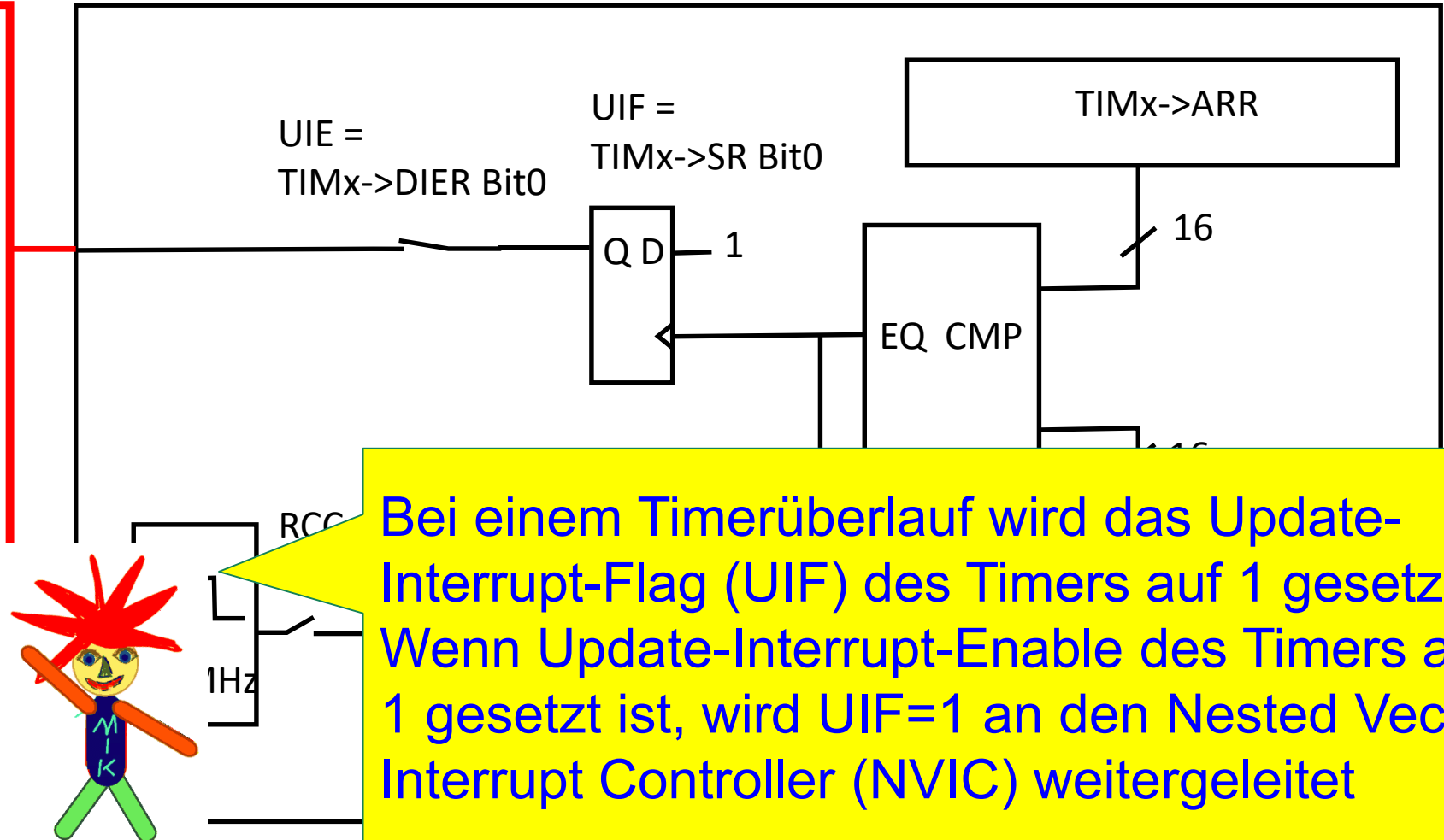
Ich bin Mik, Dein  
Mikrocontroller



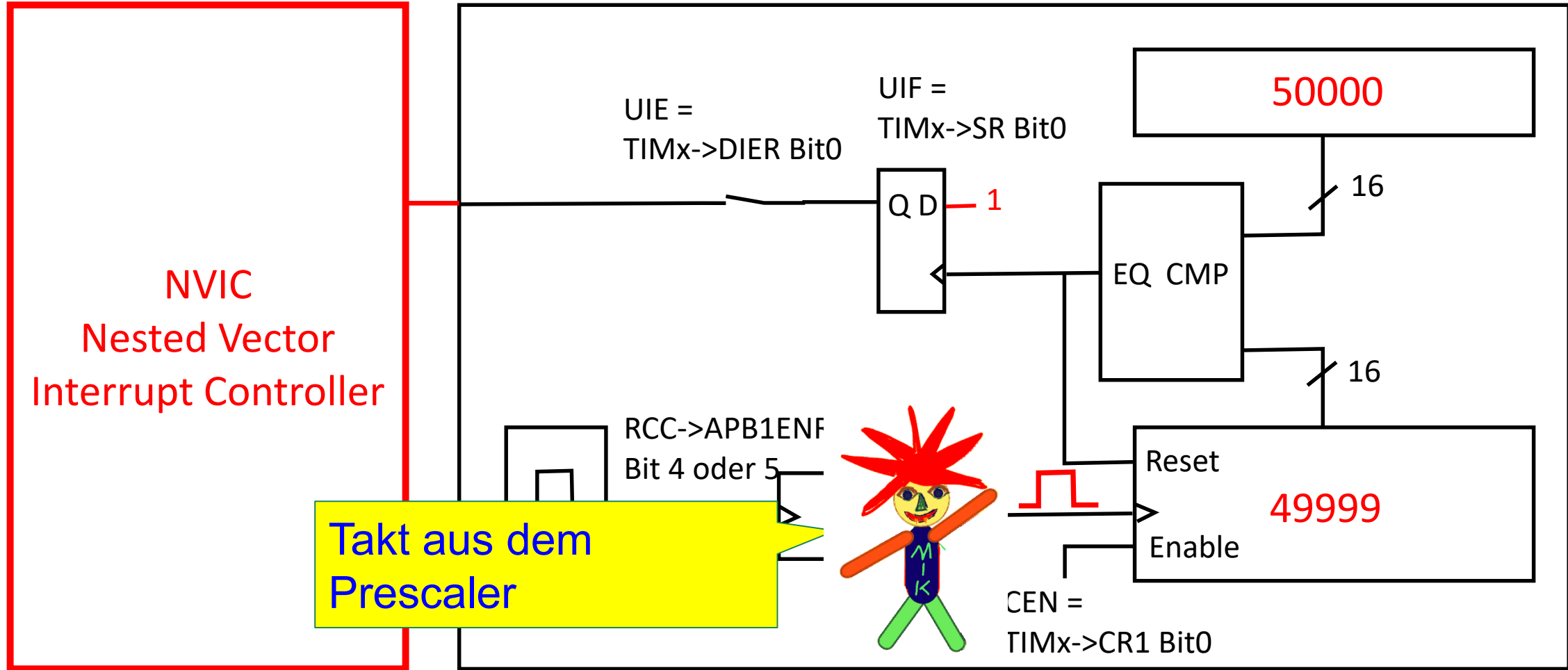
# Timer und Interrupt



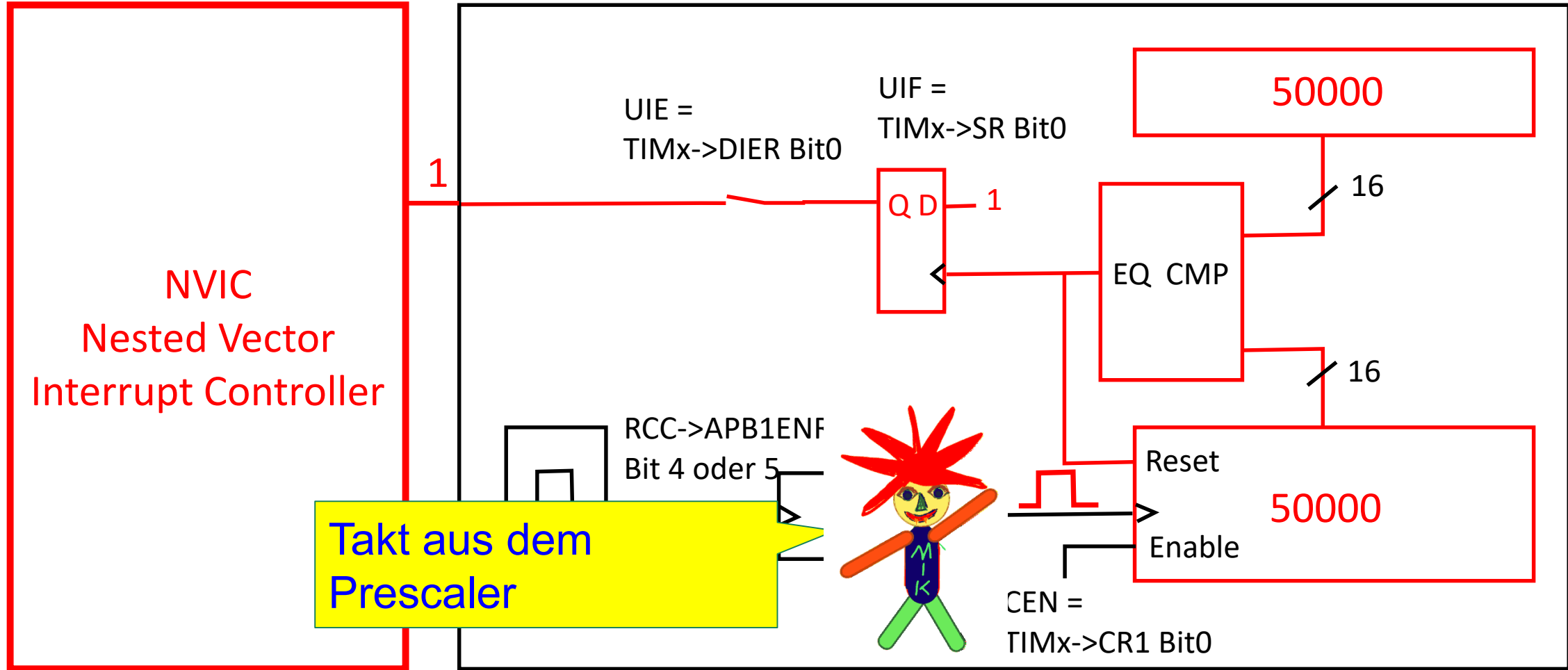
NVIC  
Nested Vector  
Interrupt Controller



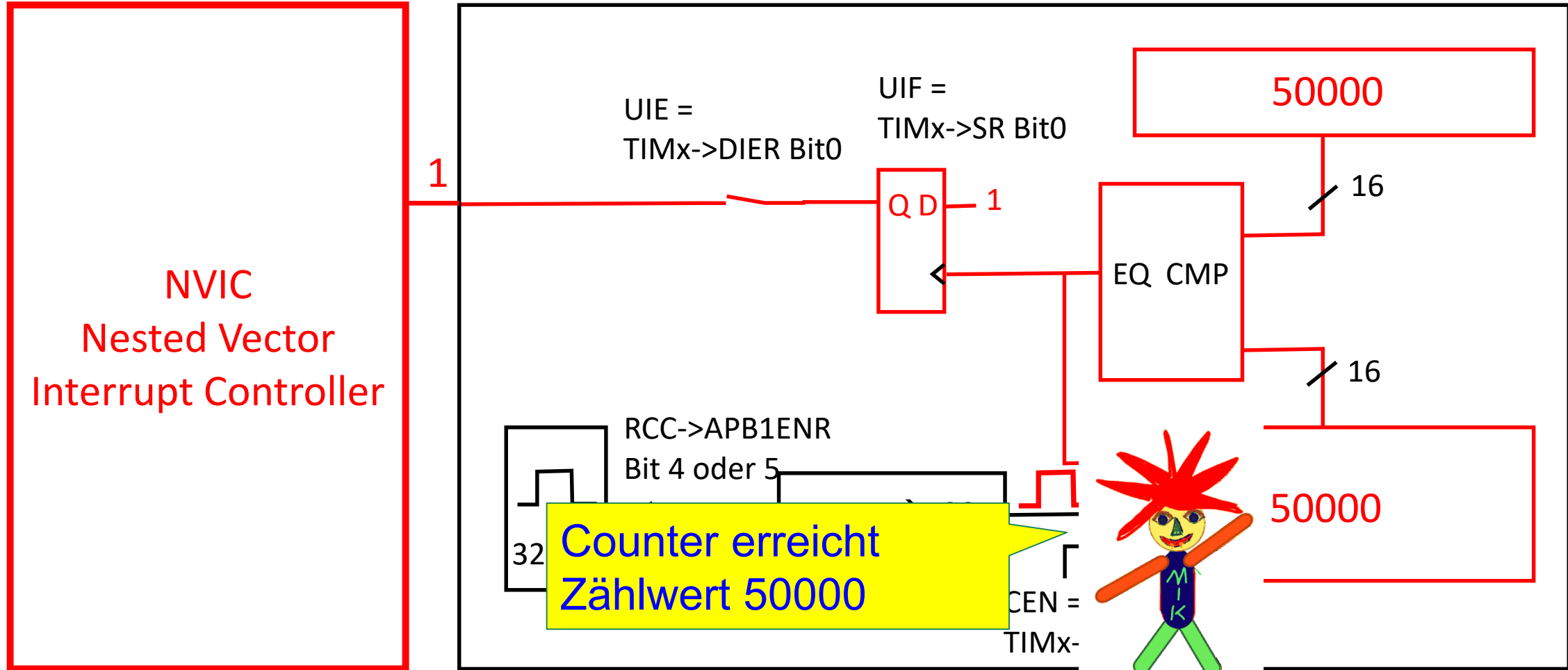
# Timer und Interrupt



# Timer und Interrupt



# Timer und Interrupt

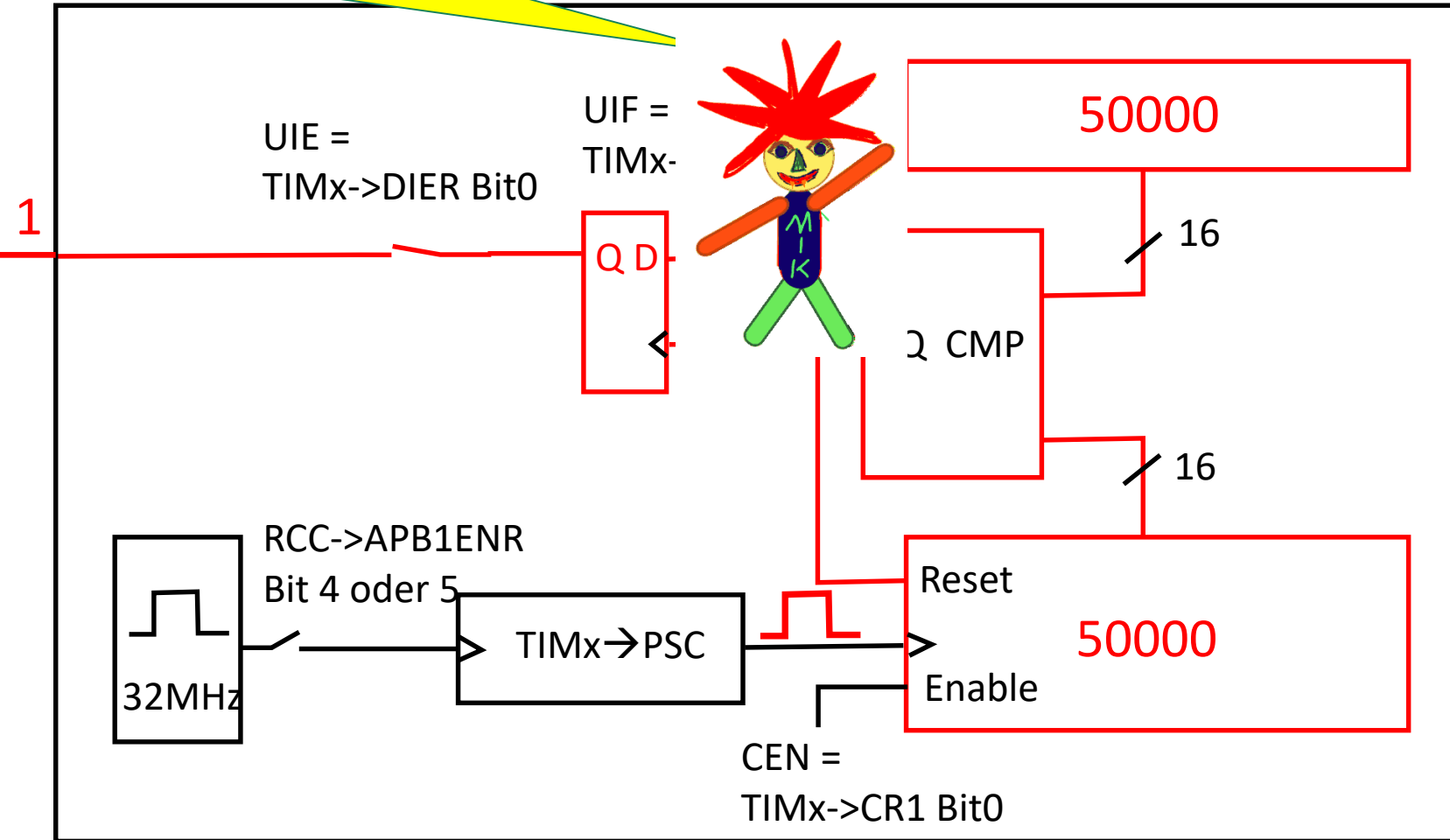


# Timer und Interrupt



Der gleiche Wert steht im  
Autoreloadregister: 50000

NVIC  
Nested Vector  
Interrupt Controller

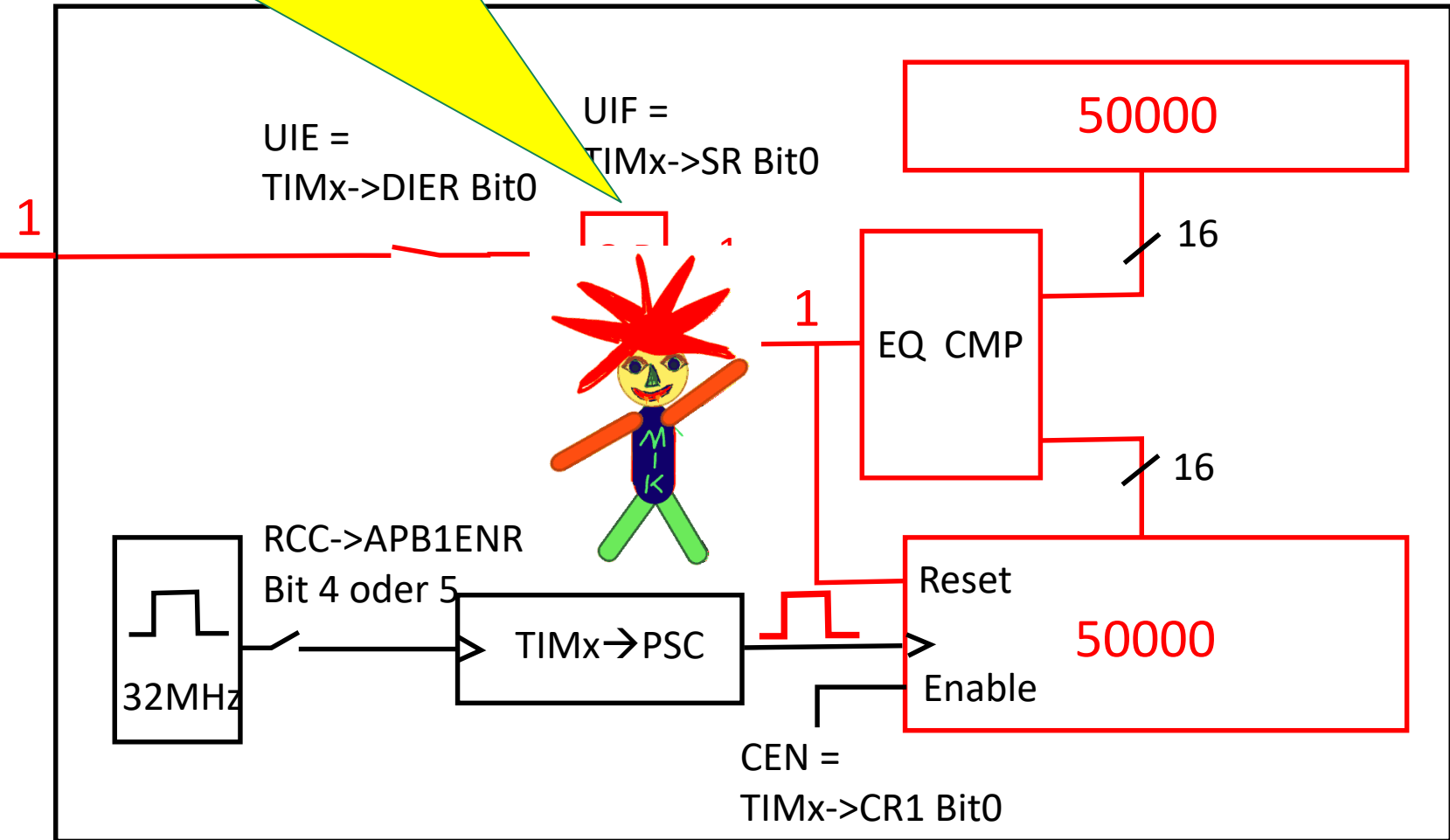


# Timer und Interrupt

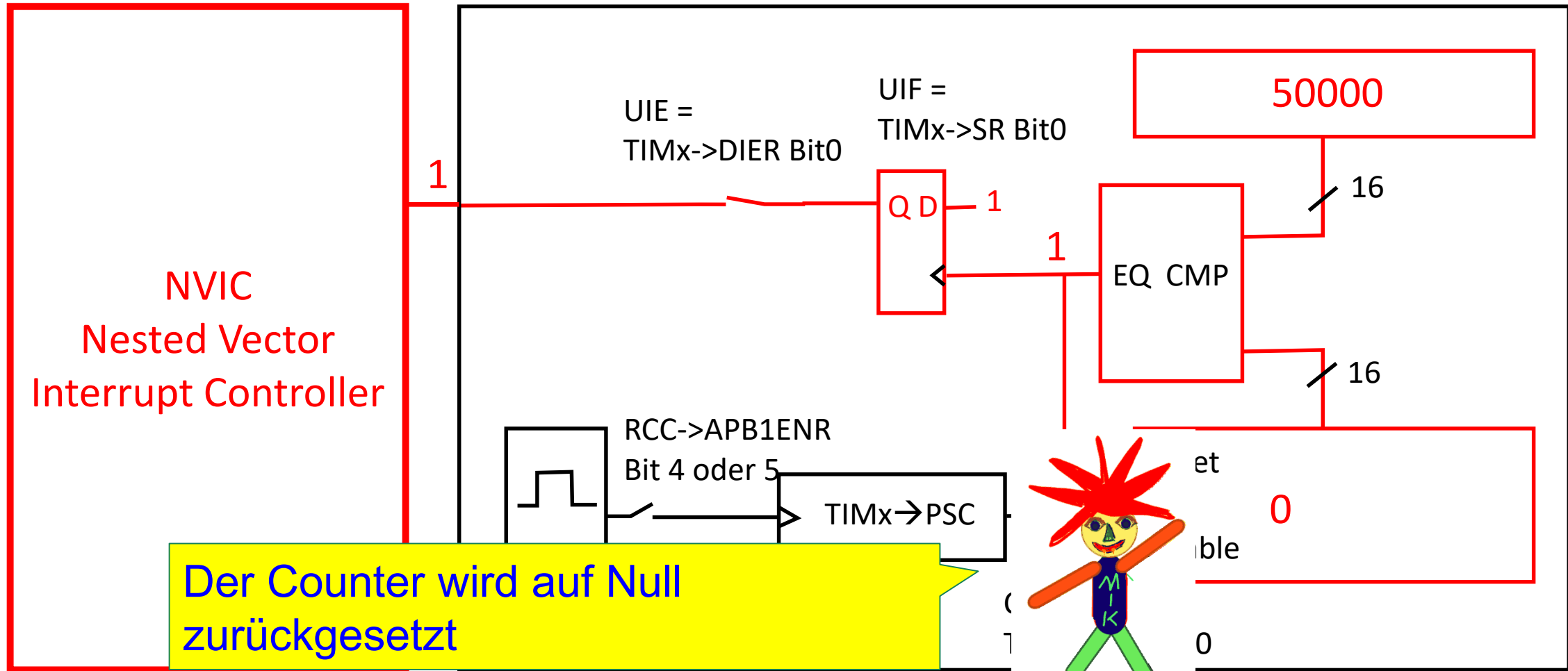


Der Vergleicherausgang wird 1

NVIC  
Nested Vector  
Interrupt Controller



# Timer und Interrupt



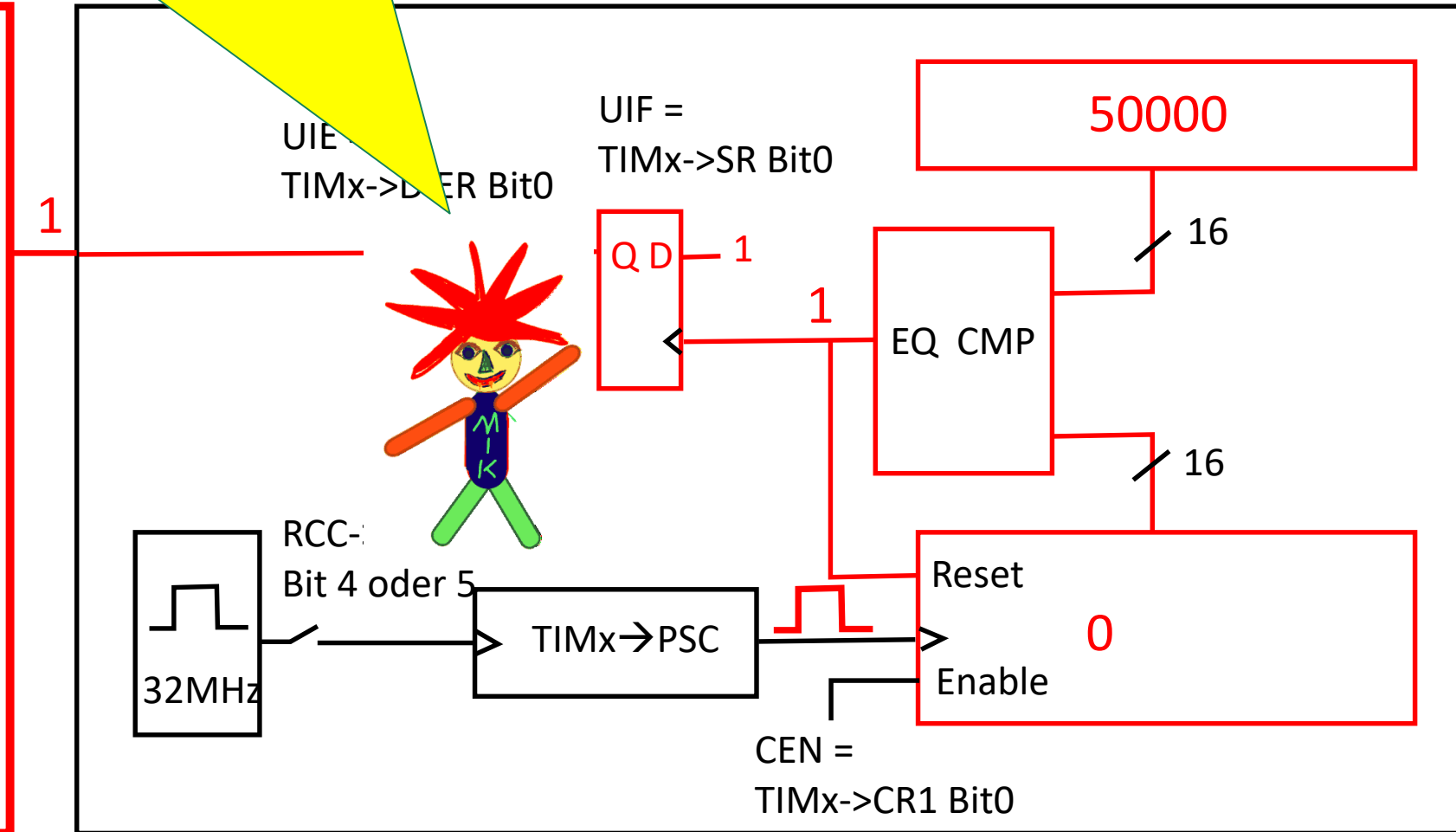


# Timer und Interrupt



Gleichzeitig wird das UIF Flipflop (Update-Interrupt-Flag) auf 1 gesetzt

NVIC  
Nested Vector  
Interrupt Controller



## A vintage-style alarm clock with a white face and Roman numerals, set against a light blue background. The clock has a dark metal frame and two bells on top. The time shown is approximately 10:10.

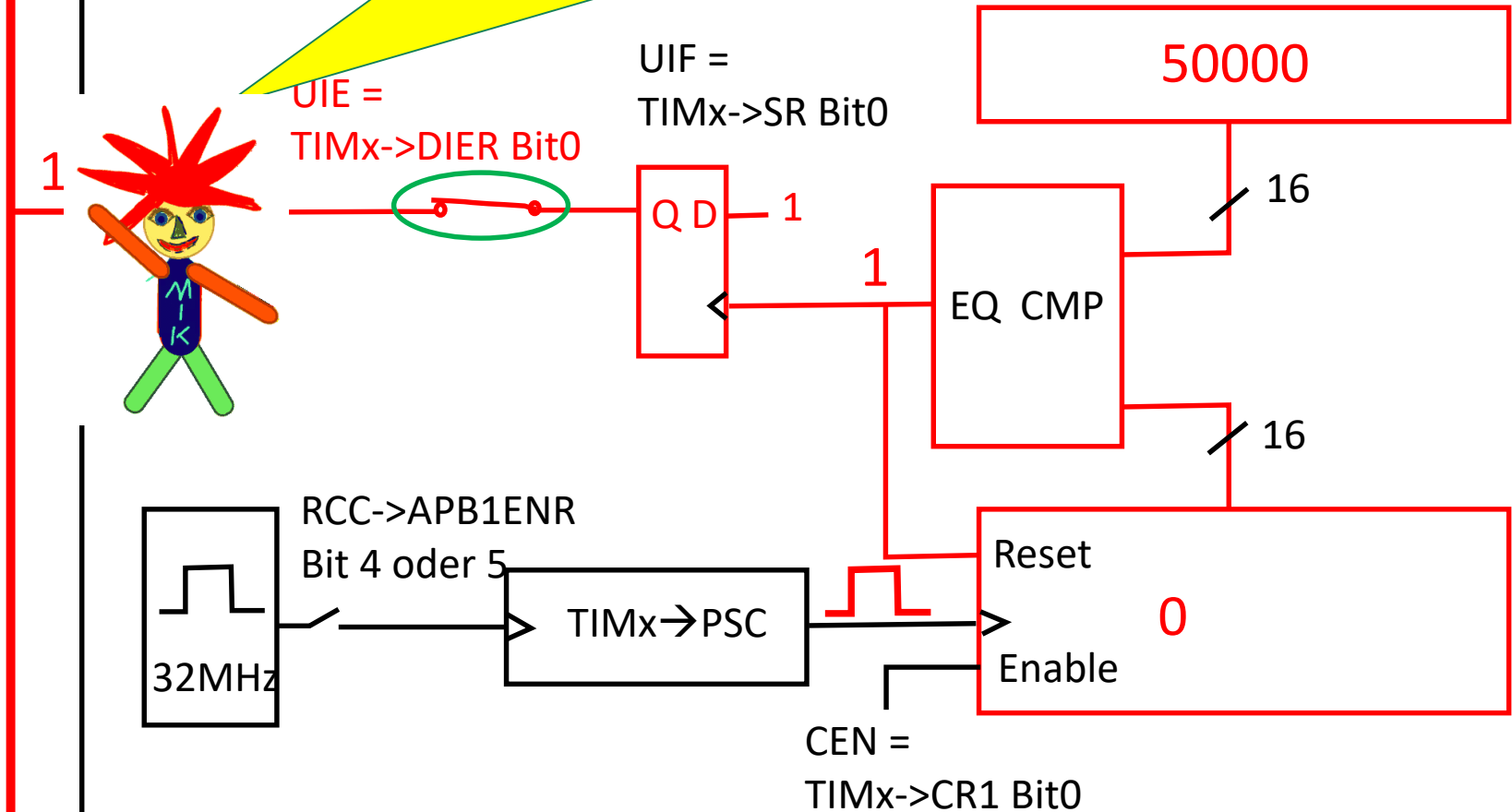


# Timer und Interrupt



Wird die 1 an den Nested Vector Interrupt Controller (NVIC) zur Verarbeitung weitergeleitet

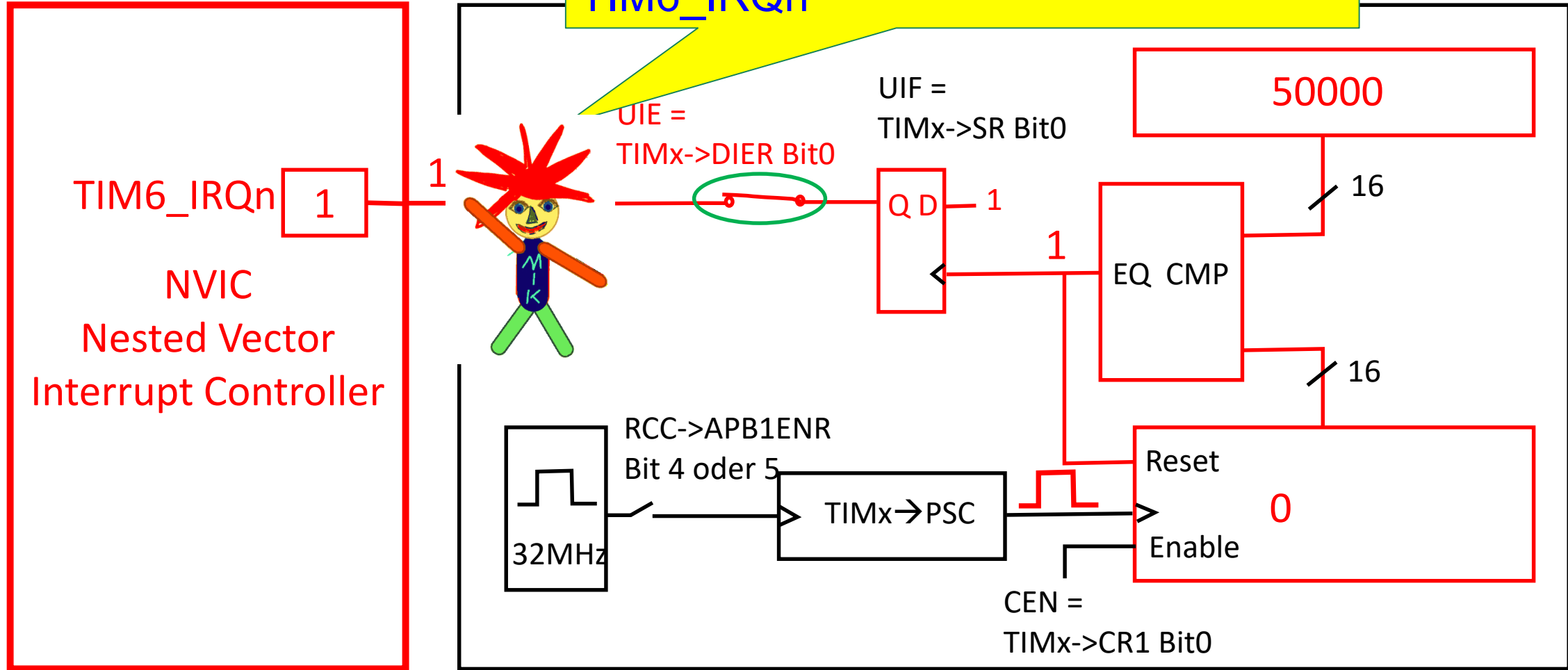
NVIC  
Nested Vector  
Interrupt Controller



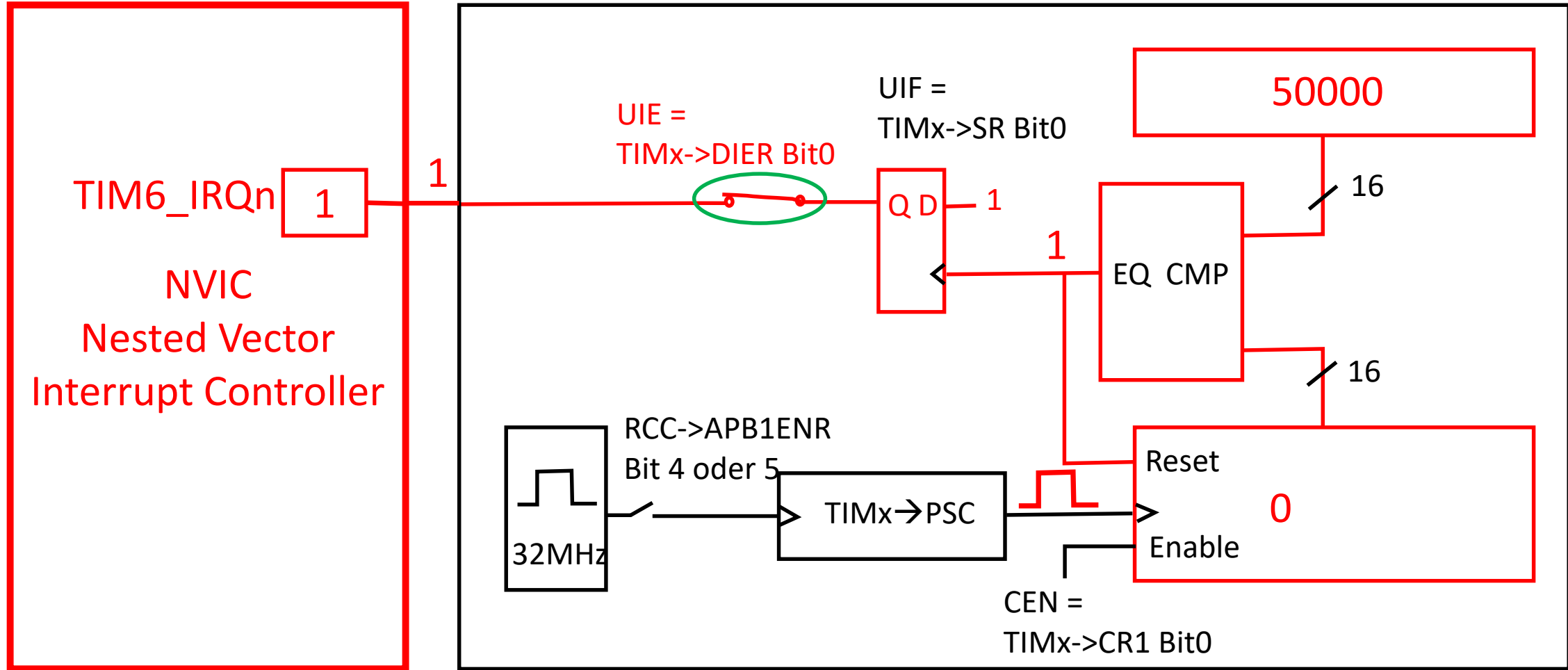
# Timer und Interrupt



NVIC speichert die  
Interruptanforderung im Pendingbit  
TIM6\_IRQn

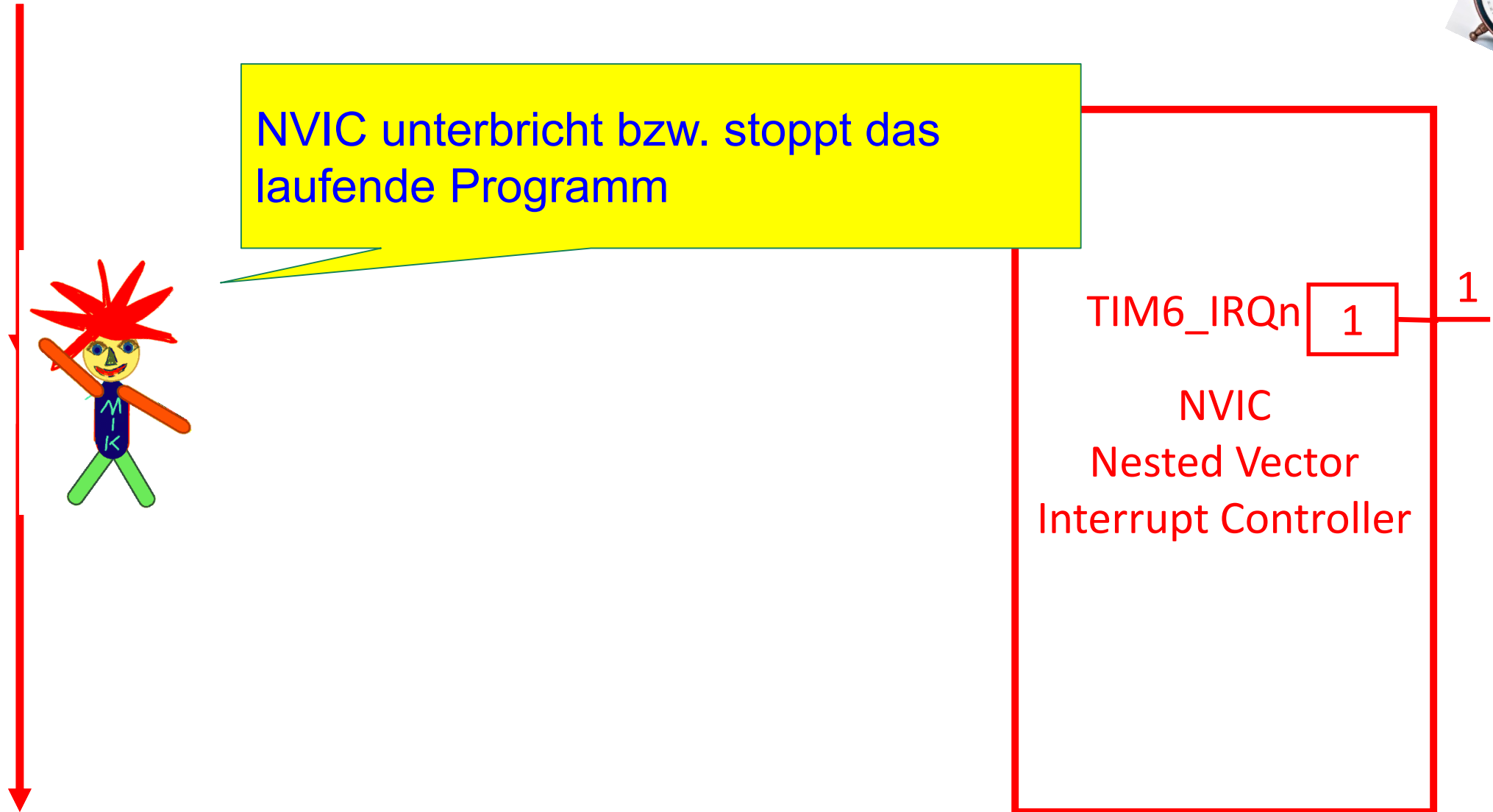


# Timer und Interrupt



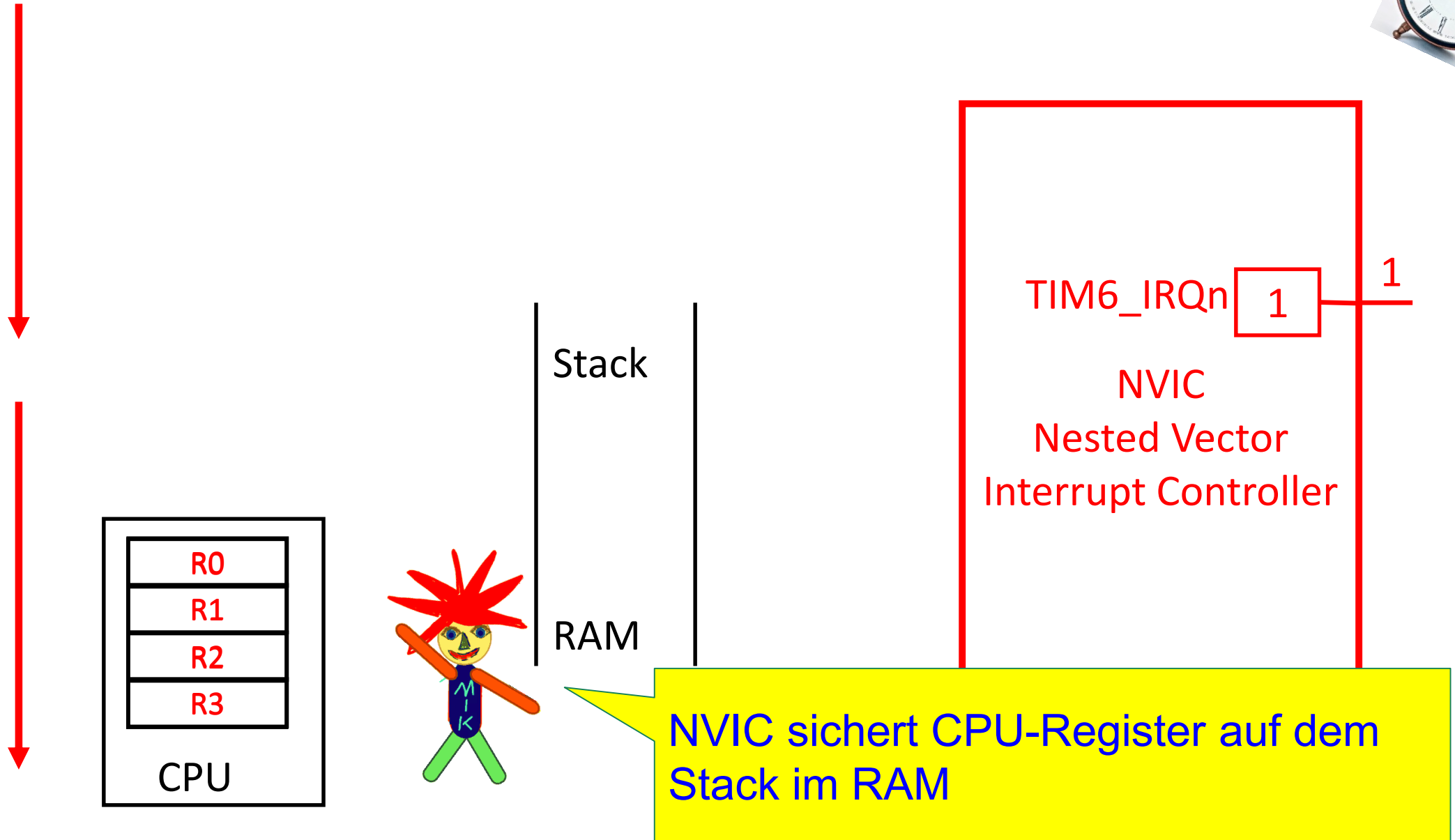
# Timer und Interrupt

Hauptprogramm, Unterprogramm oder ISR mit niedrigerer Priorität



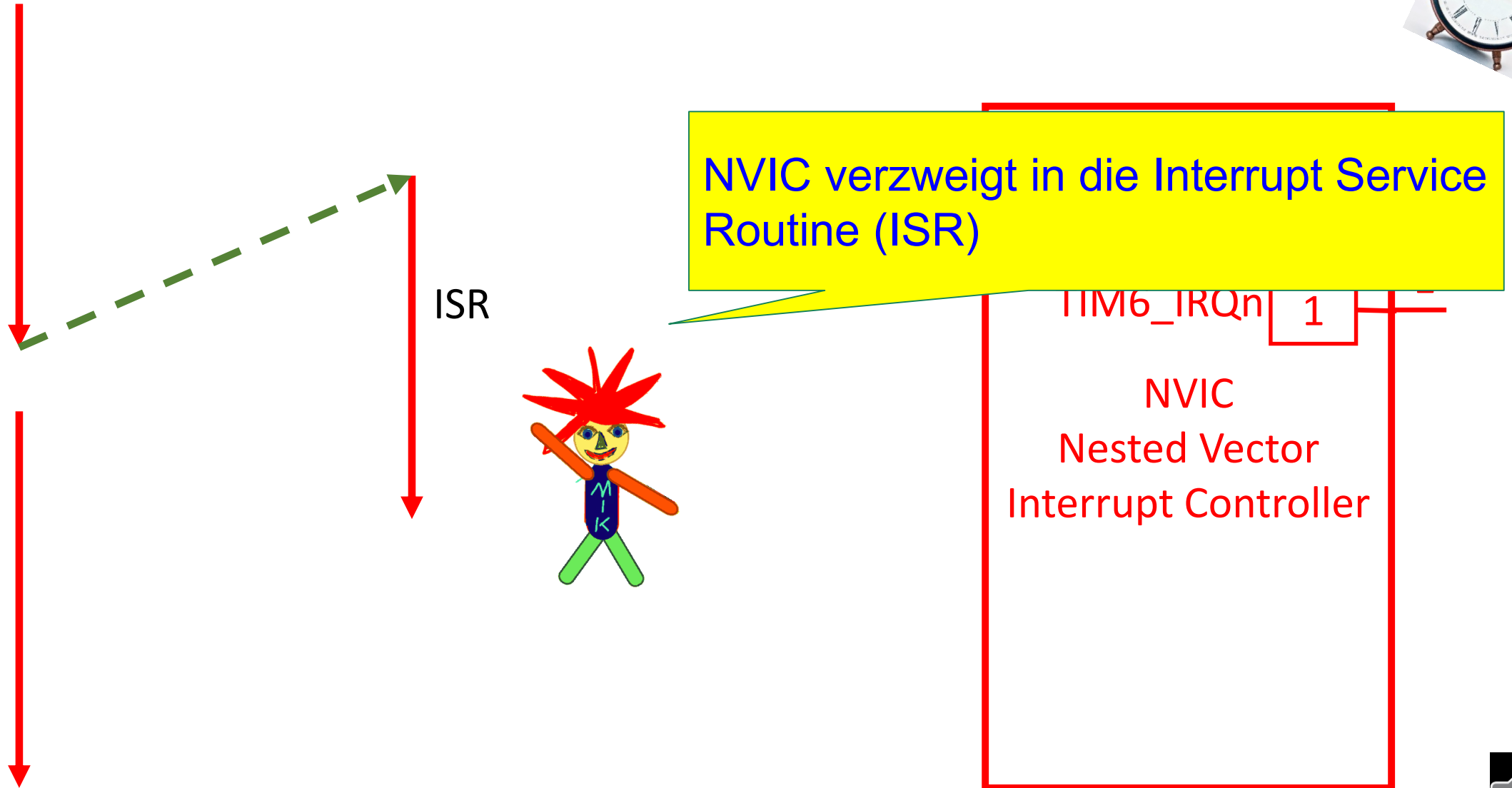
# Timer und Interrupt

Hauptprogramm, Unterprogramm oder ISR mit niedrigerer Priorität



# Timer und Interrupt

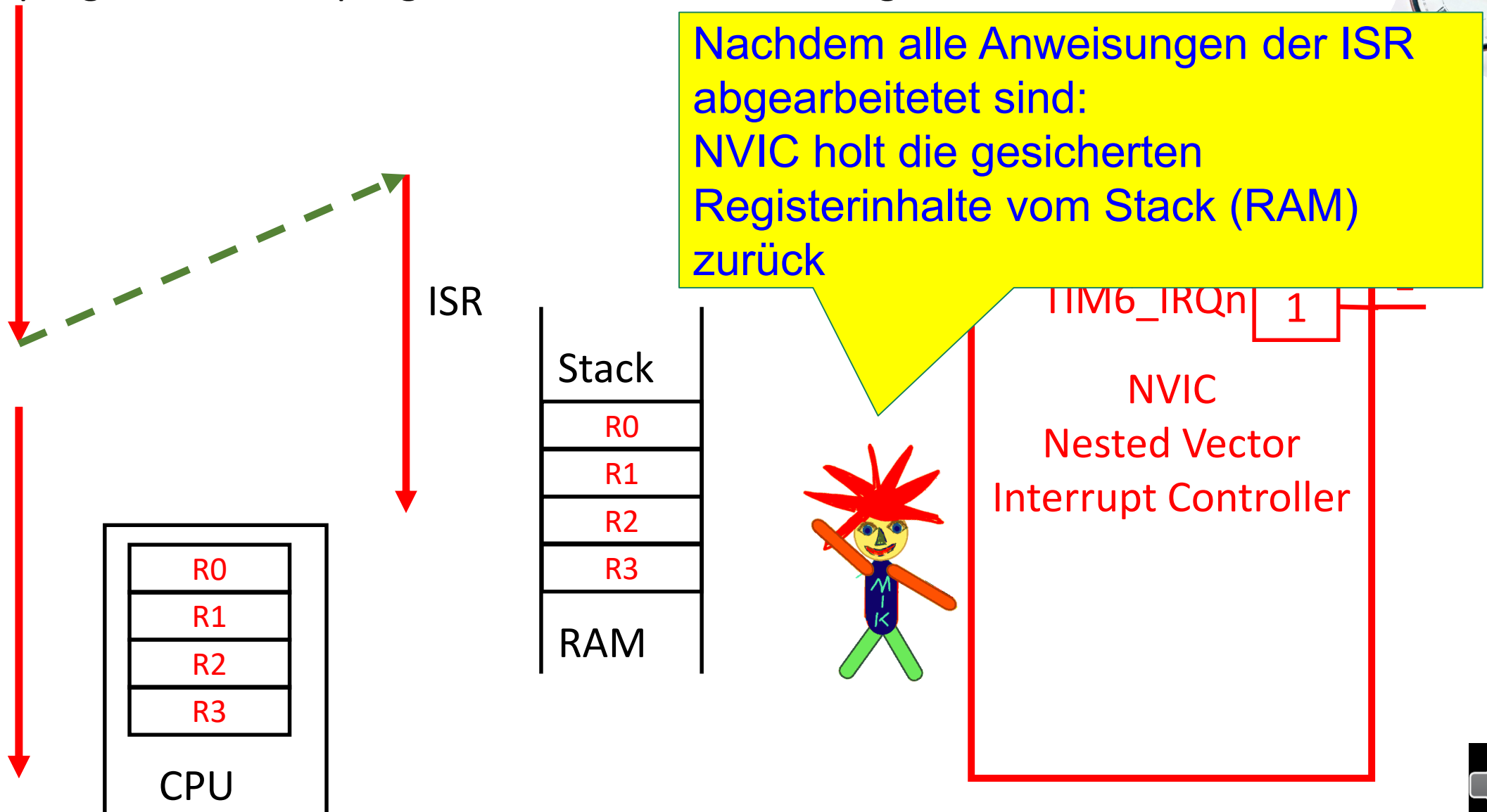
Hauptprogramm, Unterprogramm oder ISR mit niedrigerer Priorität





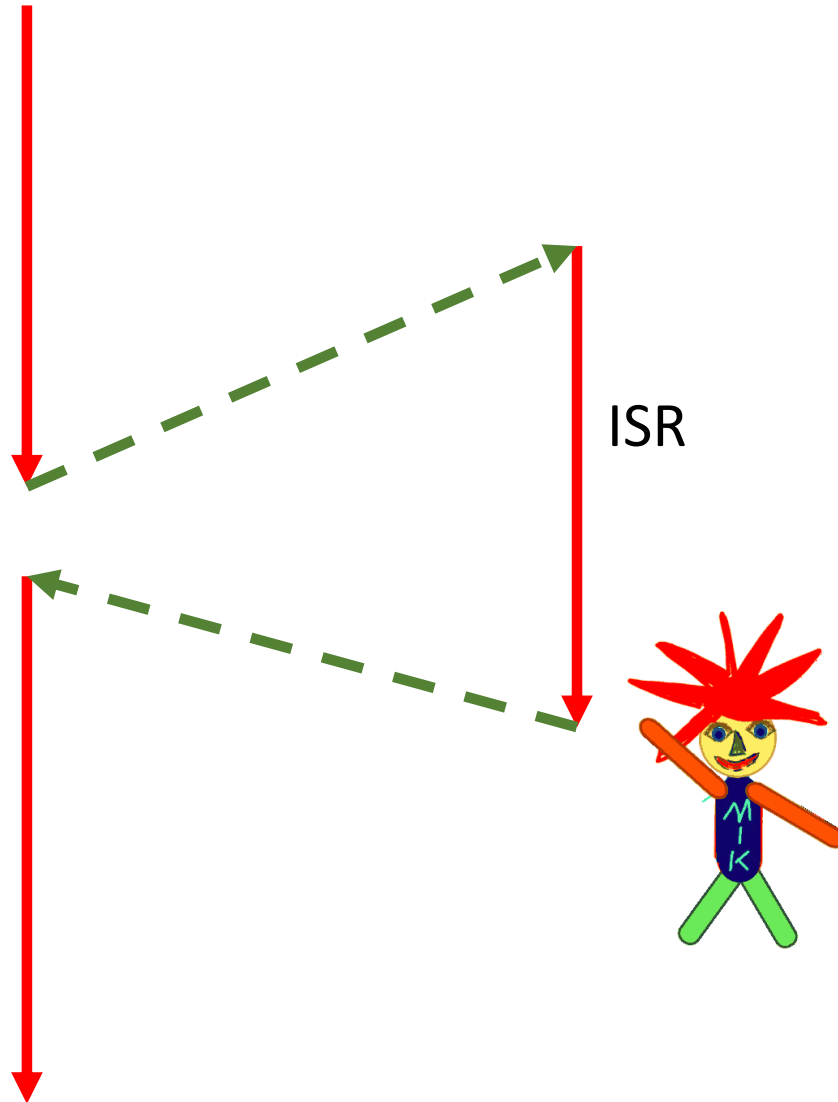
# Timer und Interrupt

Hauptprogramm, Unterprogramm oder ISR mit niedrigerer Priorität



# Timer und Interrupt

Hauptprogramm, Unterprogramm oder ISR mit niedrigerer Priorität

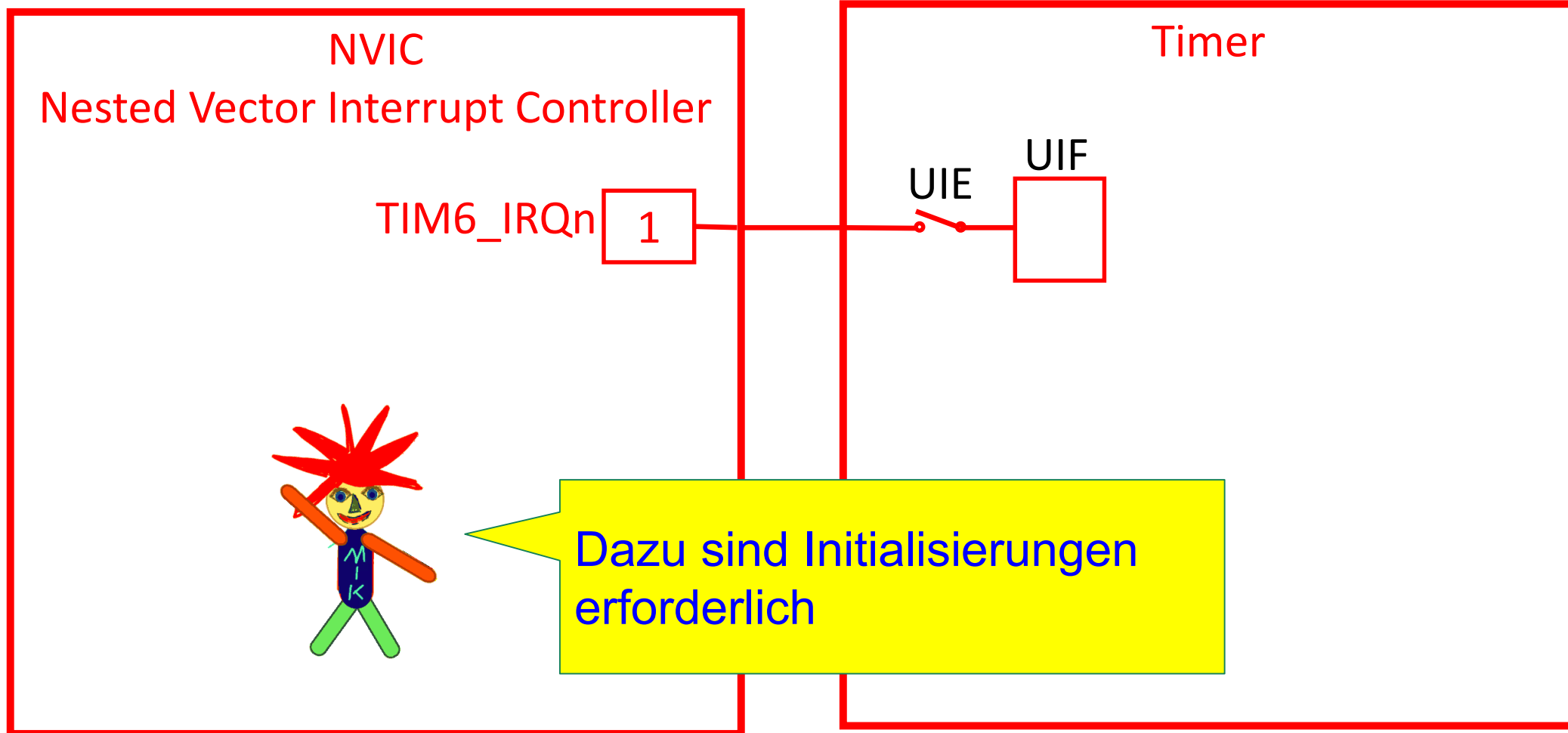


NVIC verzweigt zurück zum laufenden Programm

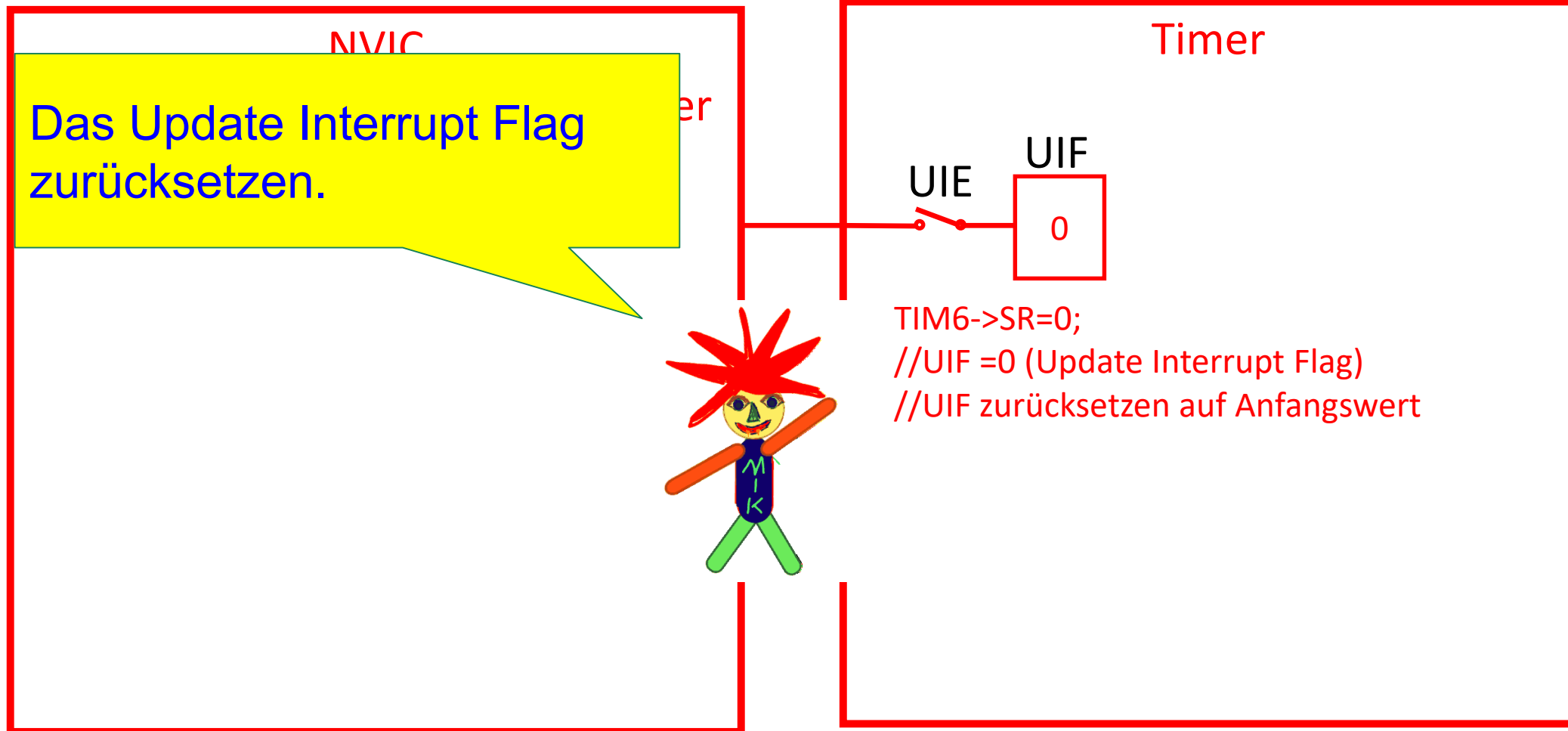
NVIC  
Nested Vector  
Interrupt Controller



# Timer und Interrupt



# Timer und Interrupt



# Timer und Interrupt

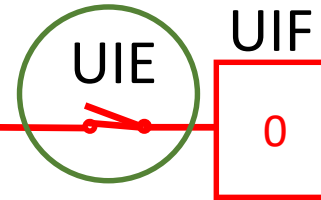


NVIC

Nested Vector Interrupt Controller

Timerinterrupt freigeben

Timer



```
TIM6->SR=0;  
//UIF =0 (Update Interrupt Flag)  
//UIF zurücksetzen auf Anfangswert
```

```
TIM6->DIER=1;  
//UIE = 1 (Update Interrupt Enable)
```



# Timer und Interrupt



## NVIC

### Nested Vector Interrupt Controller

```
NVIC_SetVector(TIM6_IRQn, (uint32_t)&isr);
```



Die Anfangsadresse der ISR in der Vektortabelle eintragen, an der Stelle (TIM6\_IRQn) die für z.B. TIM6 reserviert ist

## Timer TIM6

UIE    UIF  
         0

```
TIM6->SR=0;  
//UIF =0 (Update Interrupt Flag)
```

Vektor-  
tabelle

isr

TIM6\_IRQn

TIM7\_IRQn

RAM



# Timer und Interrupt



## NVIC

### Nested Vector Interrupt Controller

```
NVIC_SetVector(TIM6_IRQn, (uint32_t)&isr);
```



Über die Tabelle weiß NVIC,  
welche ISR bei diesem  
speziellen Interrupt gestartet  
werden soll.

## Timer TIM6

UIE    UIF  
         0

```
TIM6->SR=0;  
//UIF =0 (Update Interrupt Flag)
```

Vektor-  
tabelle

isr

TIM6\_IRQn

TIM7\_IRQn

RAM



# Timer und Interrupt



## NVIC

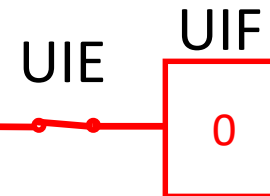
### Nested Vector Interrupt Controller

```
NVIC_SetVector(TIM6_IRQn, (uint32_t)&isr);  
HAL_NVIC_EnableIRQ(TIM6_IRQn);
```



Interruptfreigabe im NVIC

## Timer TIM6

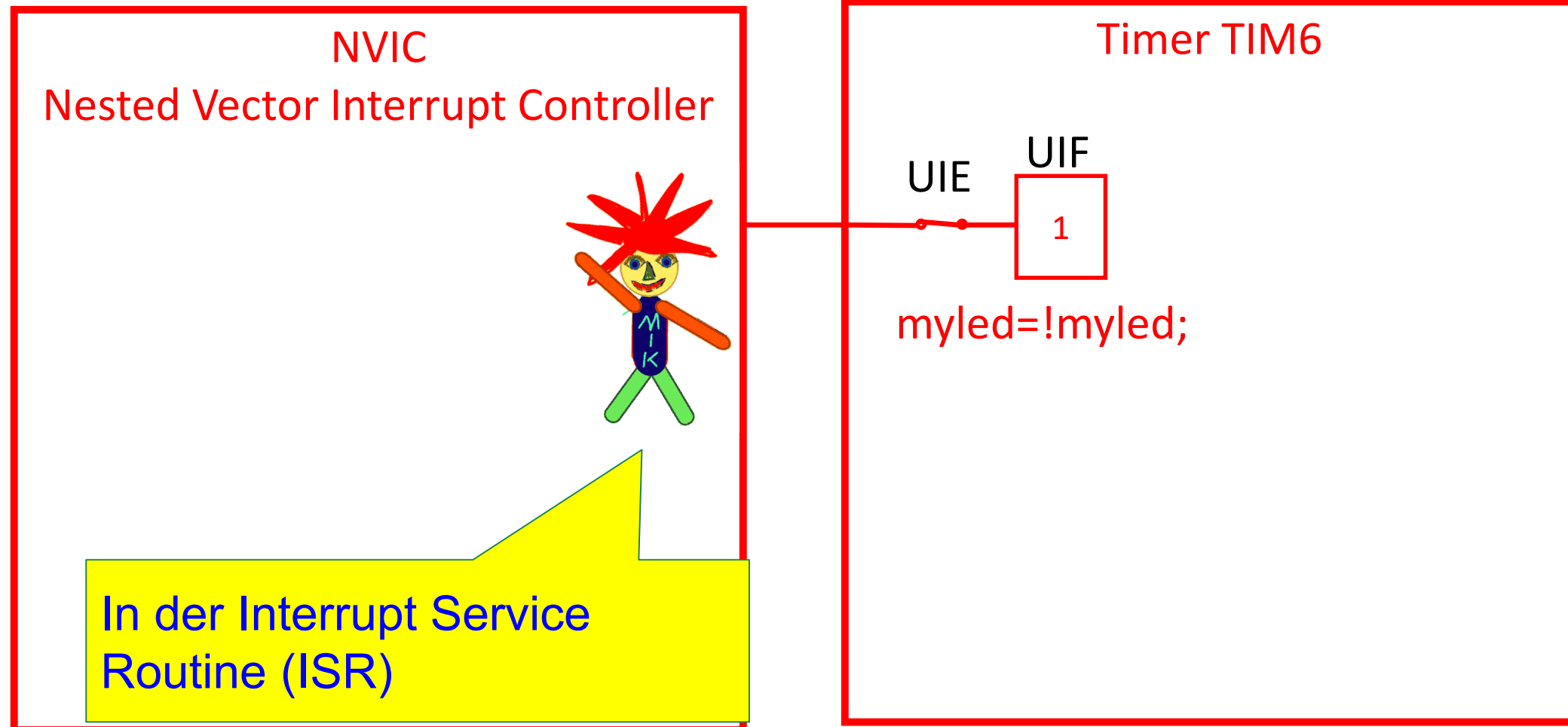


```
TIM6->SR=0;  
//UIF =0 (Update Interrupt Flag)  
//UIF zurücksetzen auf Anfangswert  
  
TIM6->DIER=1;  
//UIE = 1 (Update Interrupt Enable)
```

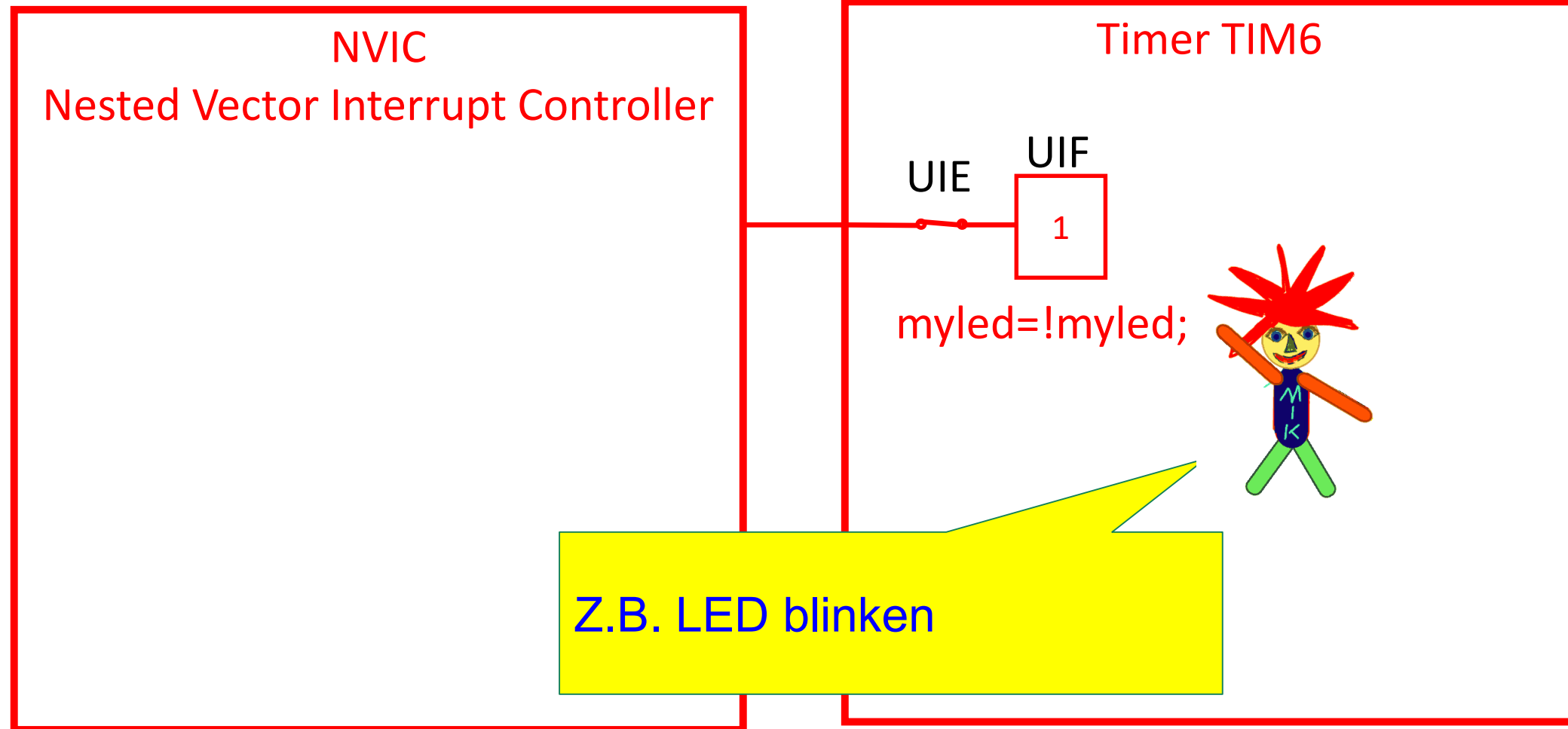




# Timer und Interrupt



# Timer und Interrupt



# Timer und Interrupt



NVIC

Nested Vector Interrupt Controller

TIM6\_IRQn 1

`HAL_NVIC_ClearPendingIRQ(TIM6_IRQn);`

Pendingbit auf 0  
zurücksetzen

Timer TIM6

UIE UIF  
0

`myled=!myled;`  
`TIM6->SR=0; //UIF=0`



# Timer und Interrupt



NVIC

Nested Vector Interrupt Controller

TIM6\_IRQn 0

`HAL_NVIC_ClearPendingIRQ(TIM6_IRQn);`

Pendingbit auf 0  
zurücksetzen

Timer TIM6

UIE 0 UIF

`myled=!myled;`

`TIM6->SR=0; //UIF=0`



# Timer und Interrupt



NVIC

Nested Vector Interrupt Controller

TIM6\_IRQn 0

```
HAL_NVIC_ClearPendingIRQ(TIM6_IRQn);
```

Jetzt kann der nächste  
Interrupt kommen

Timer TIM6

UIE 0 UIF

```
myled=!myled;
```

```
TIM6->SR=0; //UIF=0
```

