# General Purpose Timer Module (GPTM)

Vorlesung Mikroprozessortechnik

HAW Hamburg

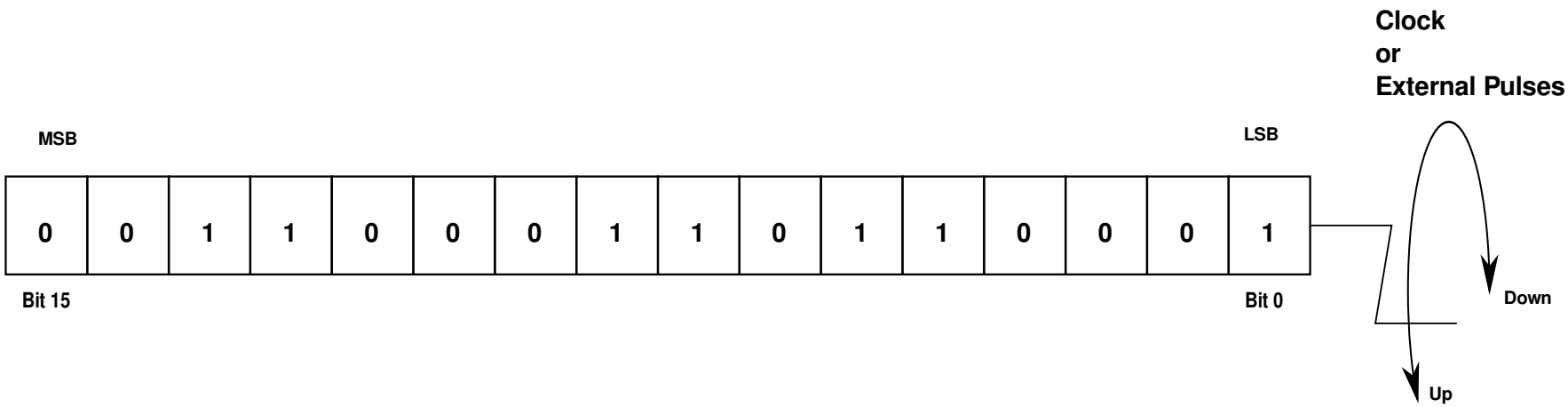
31. Dezember 2017

HAW HAMBURG

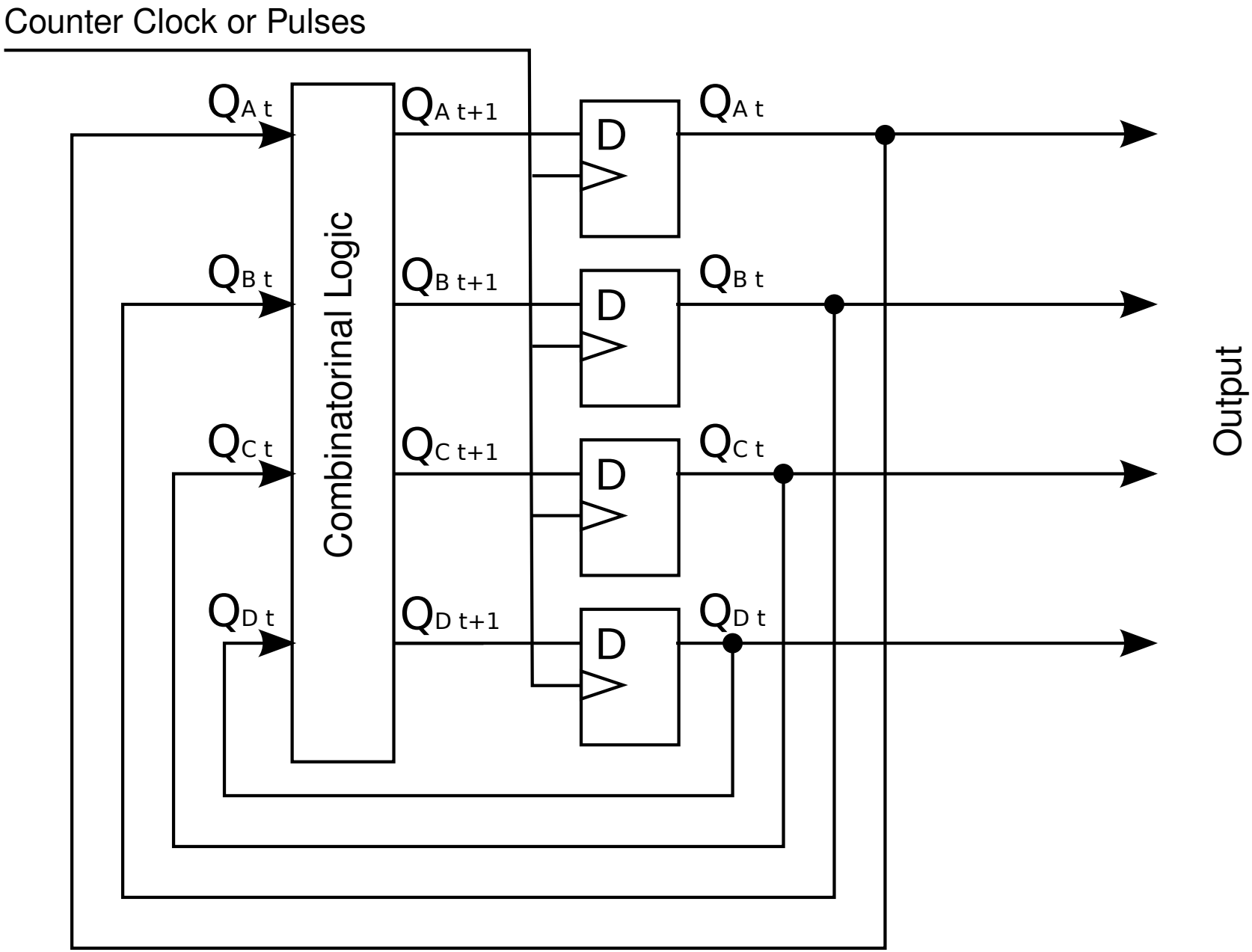
HAW HAMBURG

# Kernkomponente des Timers - der Zähler

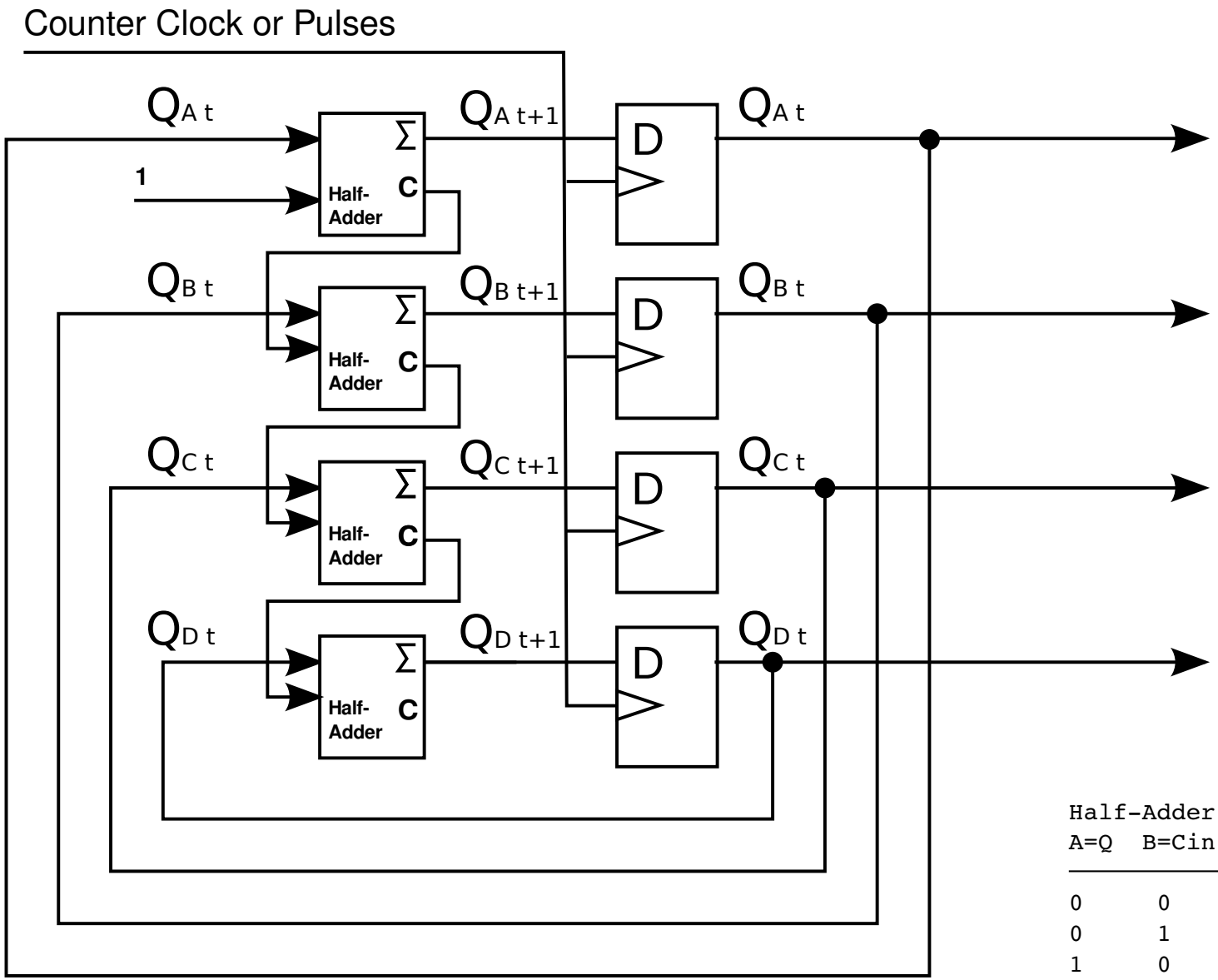
# Kernkomponente des Timers - der Zähler



# Synchronzähler Prinzip



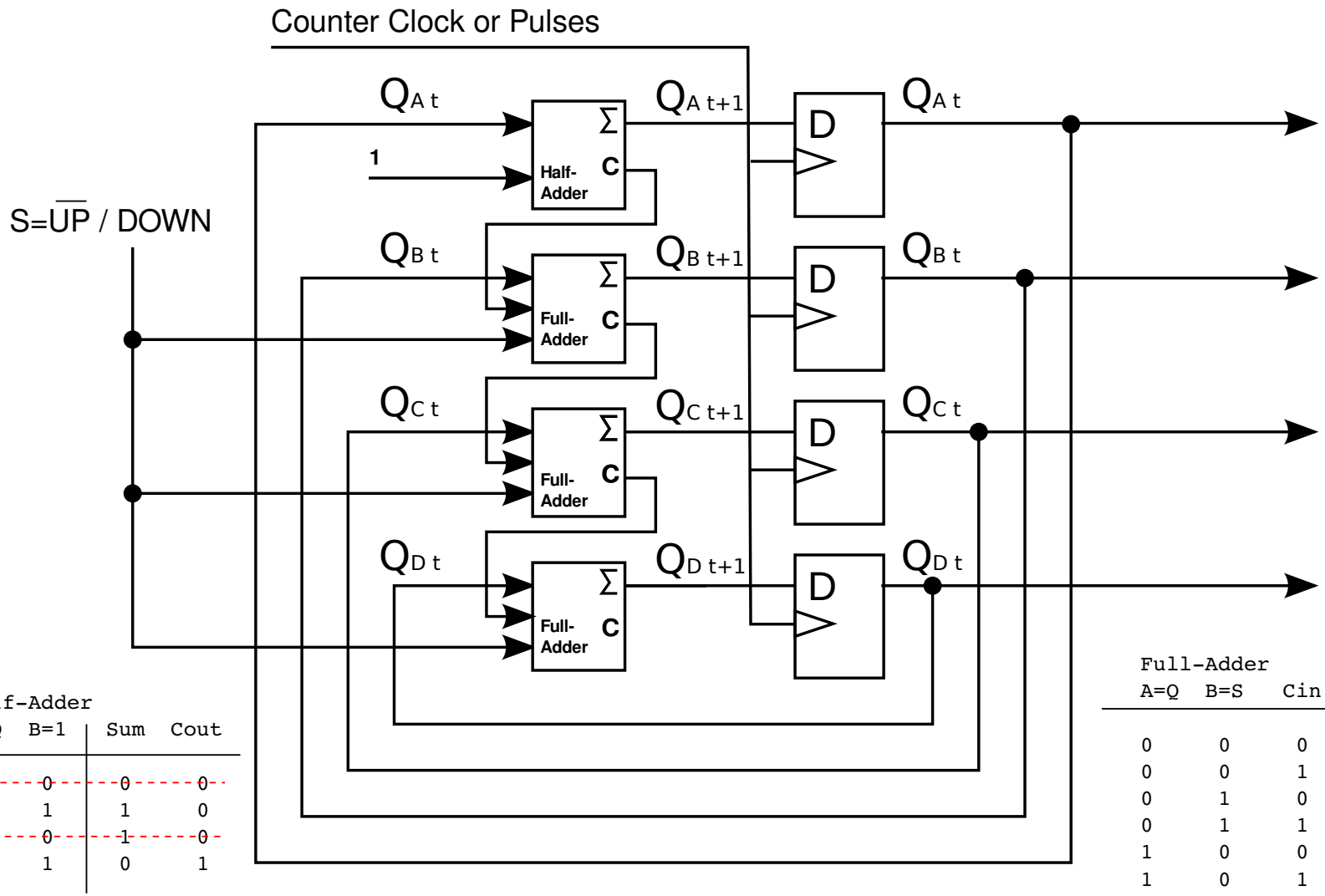
# Synchronous Up Counter




Half-Adder

| A=Q | B=Cin | Sum | Cout |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

# Synchronous Up Down Counter



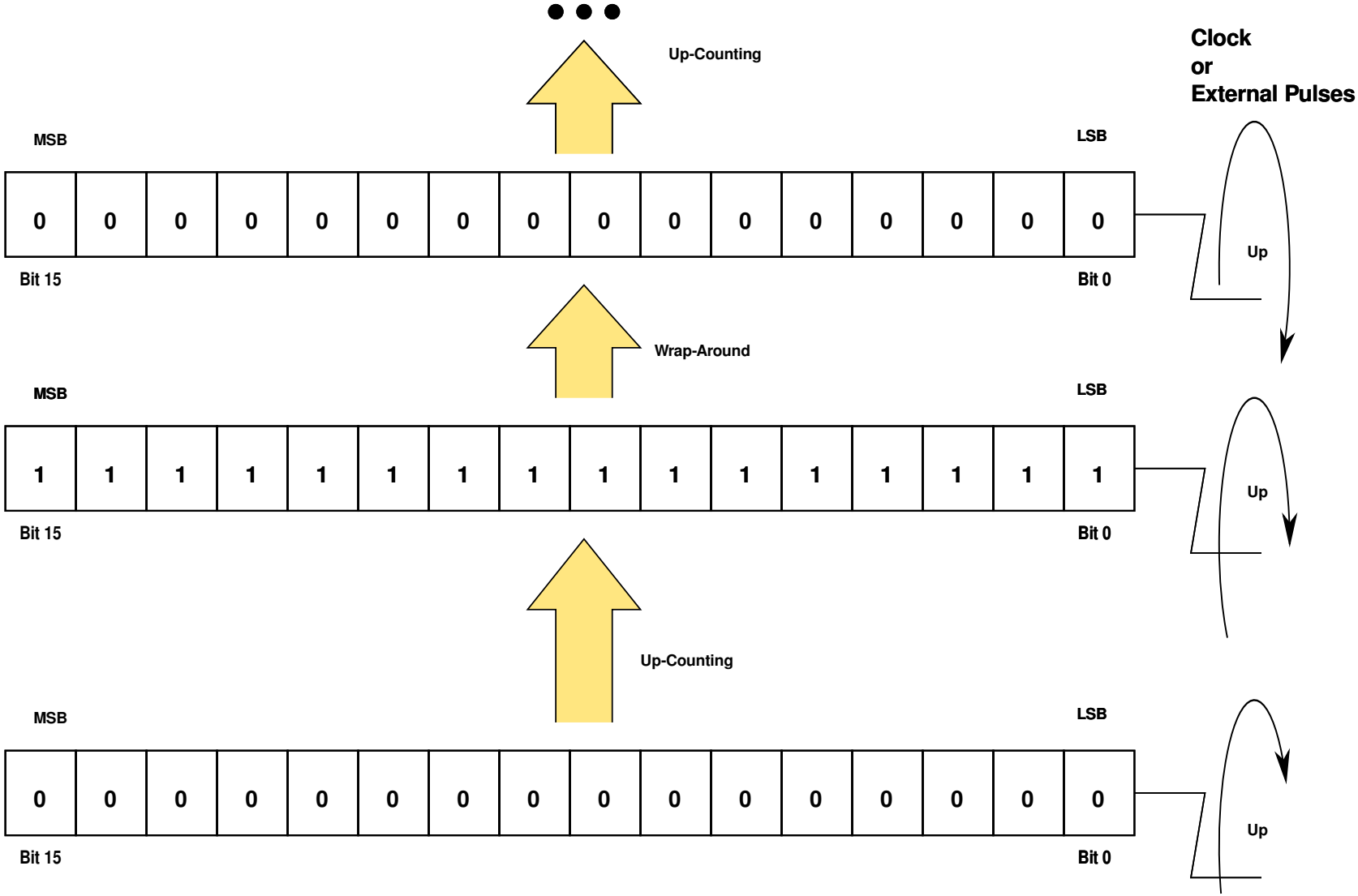

Half-Adder

| A=Q | B=1 | Sum | Cout |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

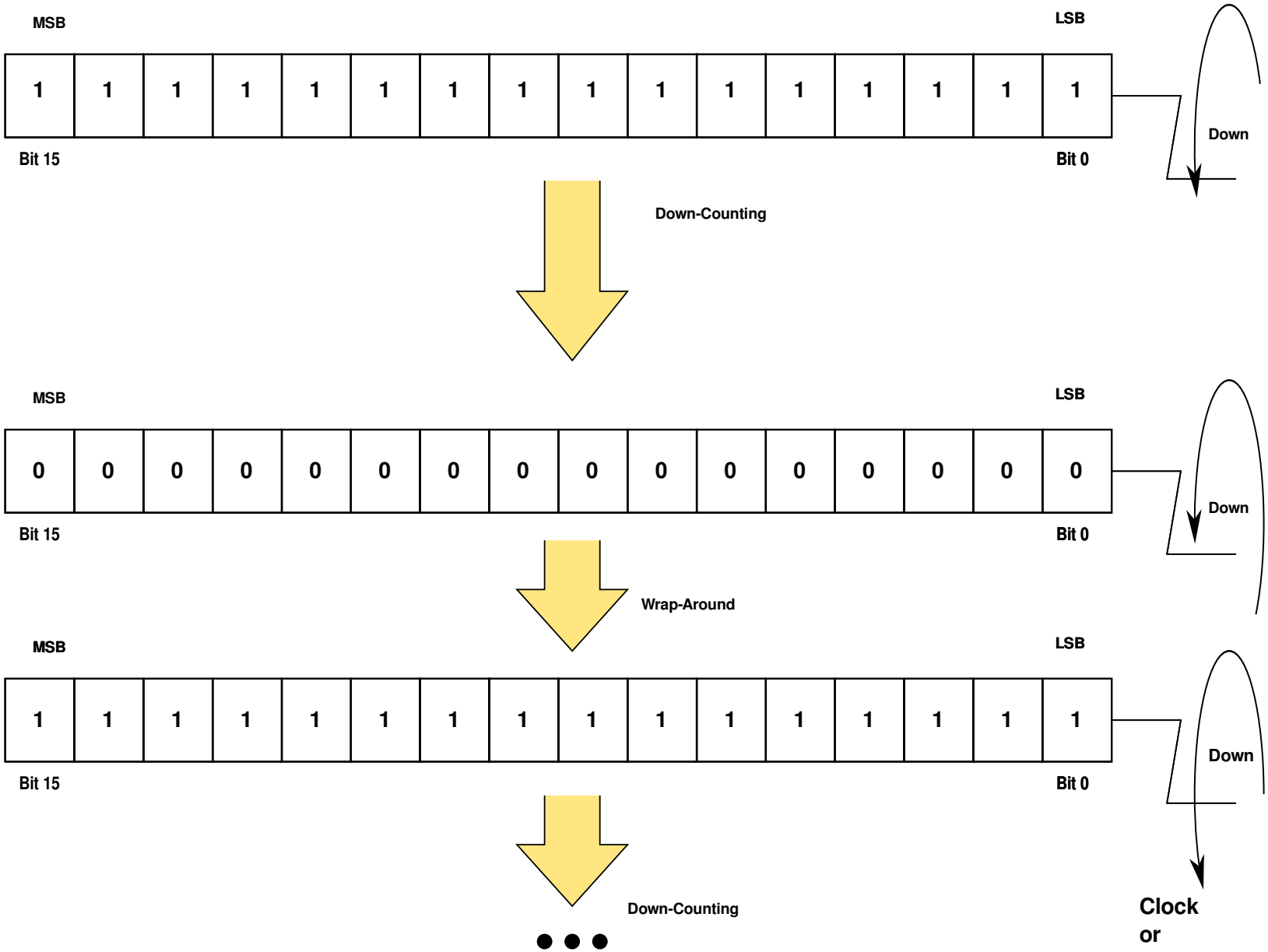
Full-Adder

| A=Q | B=S | Cin | Sum | Cout |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

# Umlauf Aufwärts - Wrap Around Up Counting



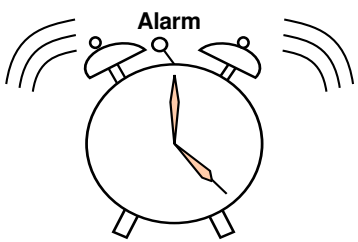
# Umlauf Abwärts - Wrap Around Down Counting



# Grundfunktionen

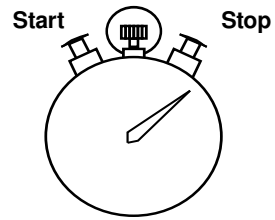
## Grundfunktionen eines Timers

### Compare



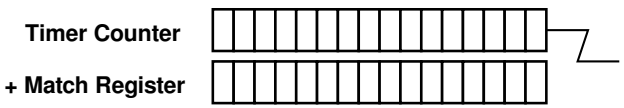

**Zeitgeber**

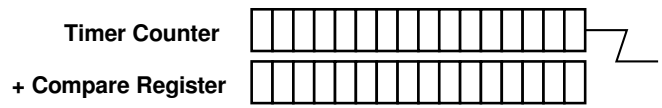
### Capture




**Zeitmesser**
**"Time Capture"**

**Ereigniszähler**
**"Imput Edge Count"**

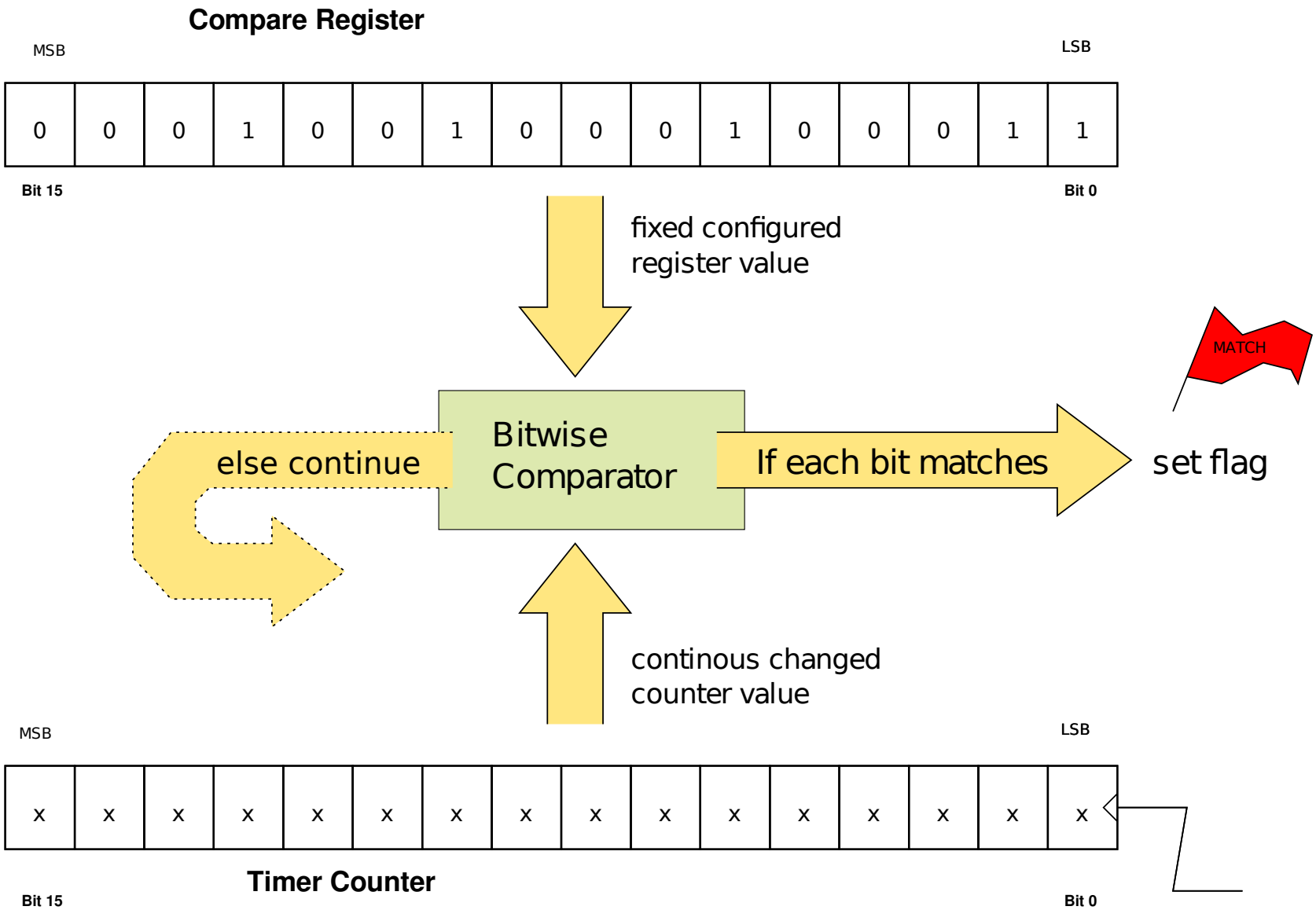


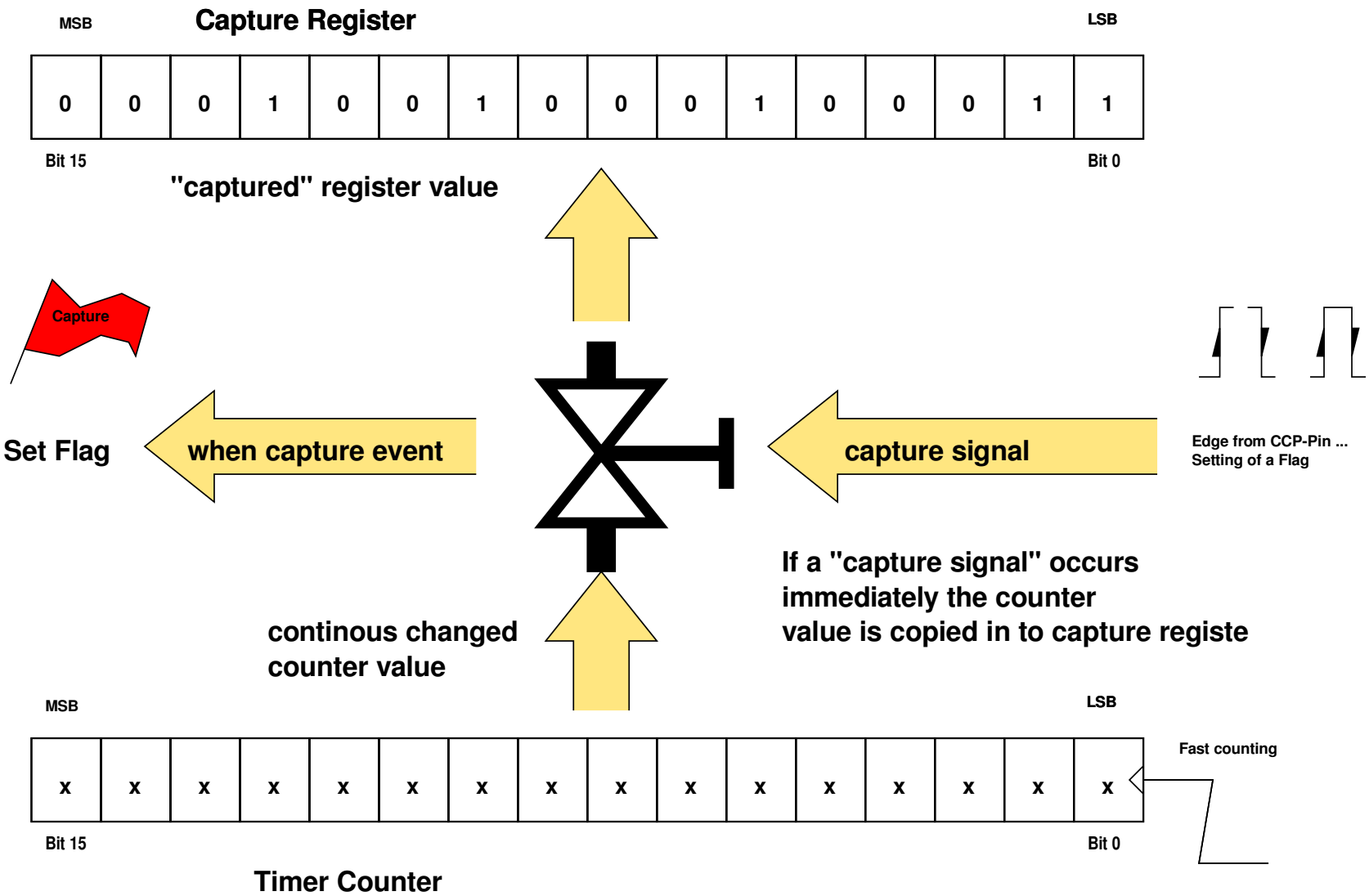

**Wiederholung ohne**
**Stopp bei Match/Capture Event**
**Periodic Mode**

**Einmalige Funktion**
**Stopp bei Match/Capture Event**
**One Shot Mode**

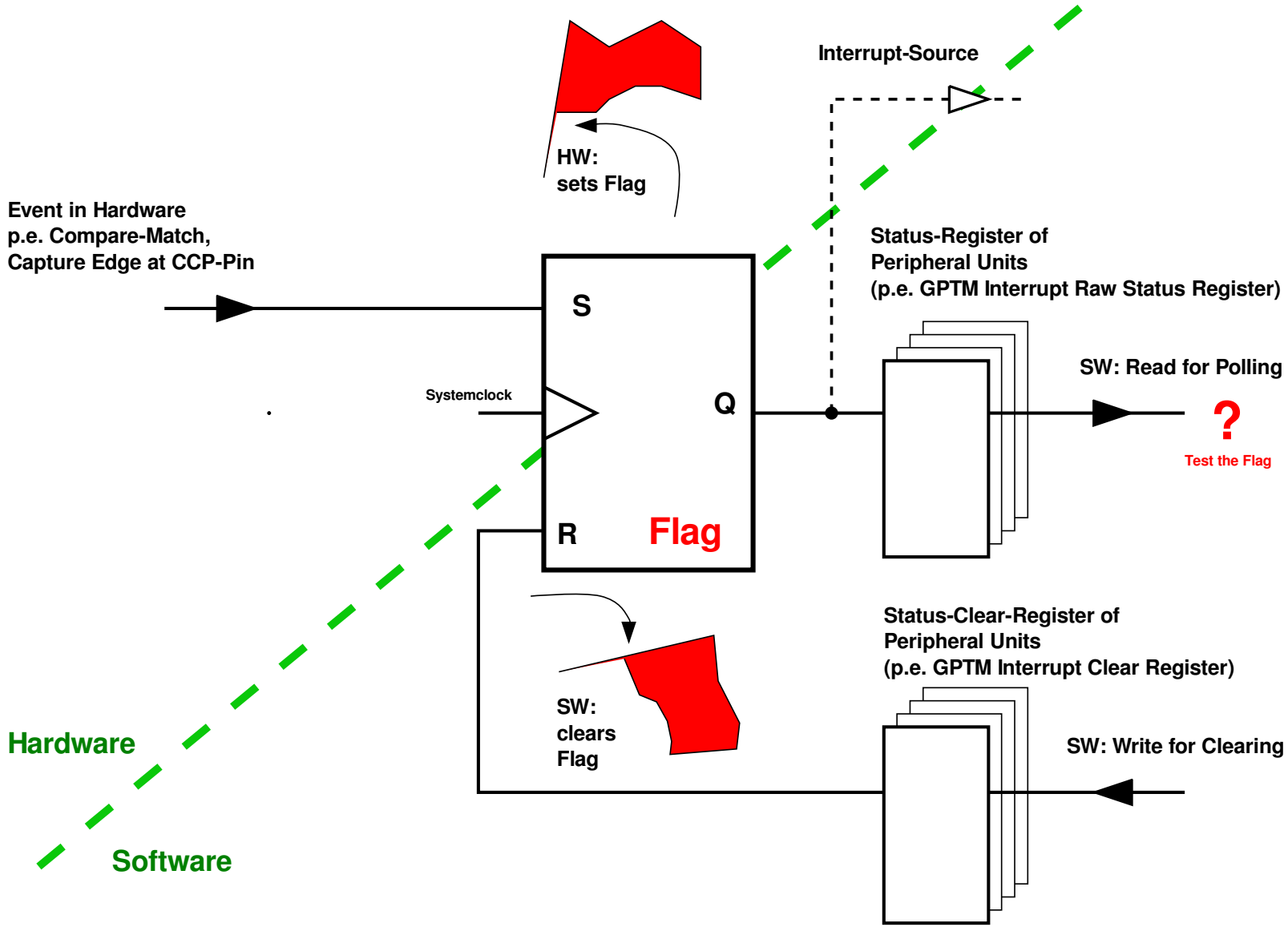
# Grundfunktion - Compare



# Grundfunktion - Capture



# Grundfunktion eines Flag: Nachricht von HW an SW



HAW HAMBURG

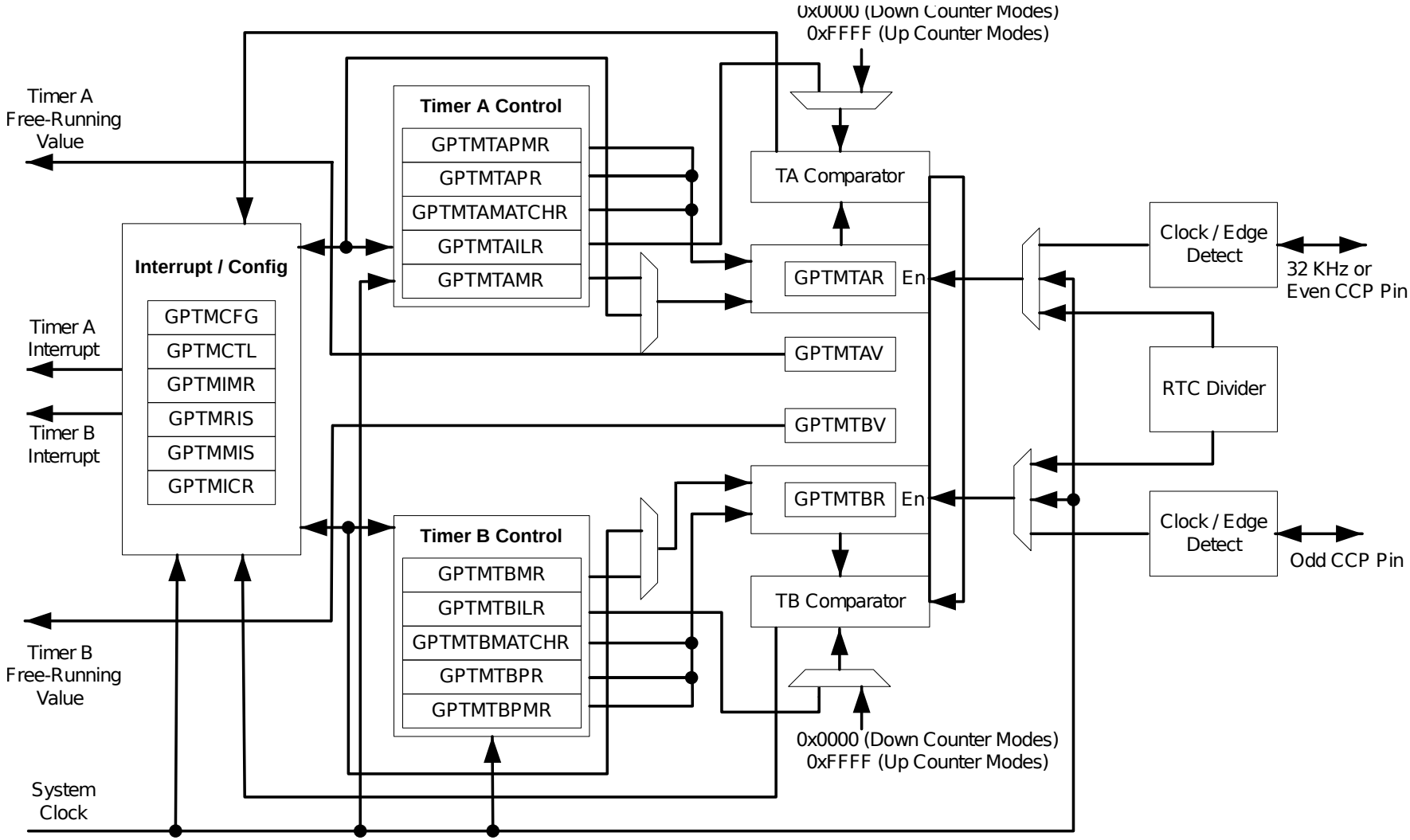
# General-Purpose Timer Modules (TM4C1294)

- Acht GPTM Kanäle (Blocks) Timer 0 ... Timer 7

  Operating modes:

- 16- or 32-bit programmable one-shot timer
- 16- or 32-bit programmable periodic timer
- 16-bit general-purpose timer with an 8-bit prescaler
- (32-bit Real-Time Clock (RTC) when using an external 32.768-KHz clock as the input) not in the Lab EKI
- 16-bit input-edge count- or time-capture modes
- 16-bit PWM mode with software-programmable output inversion of the PWM signal
- ...

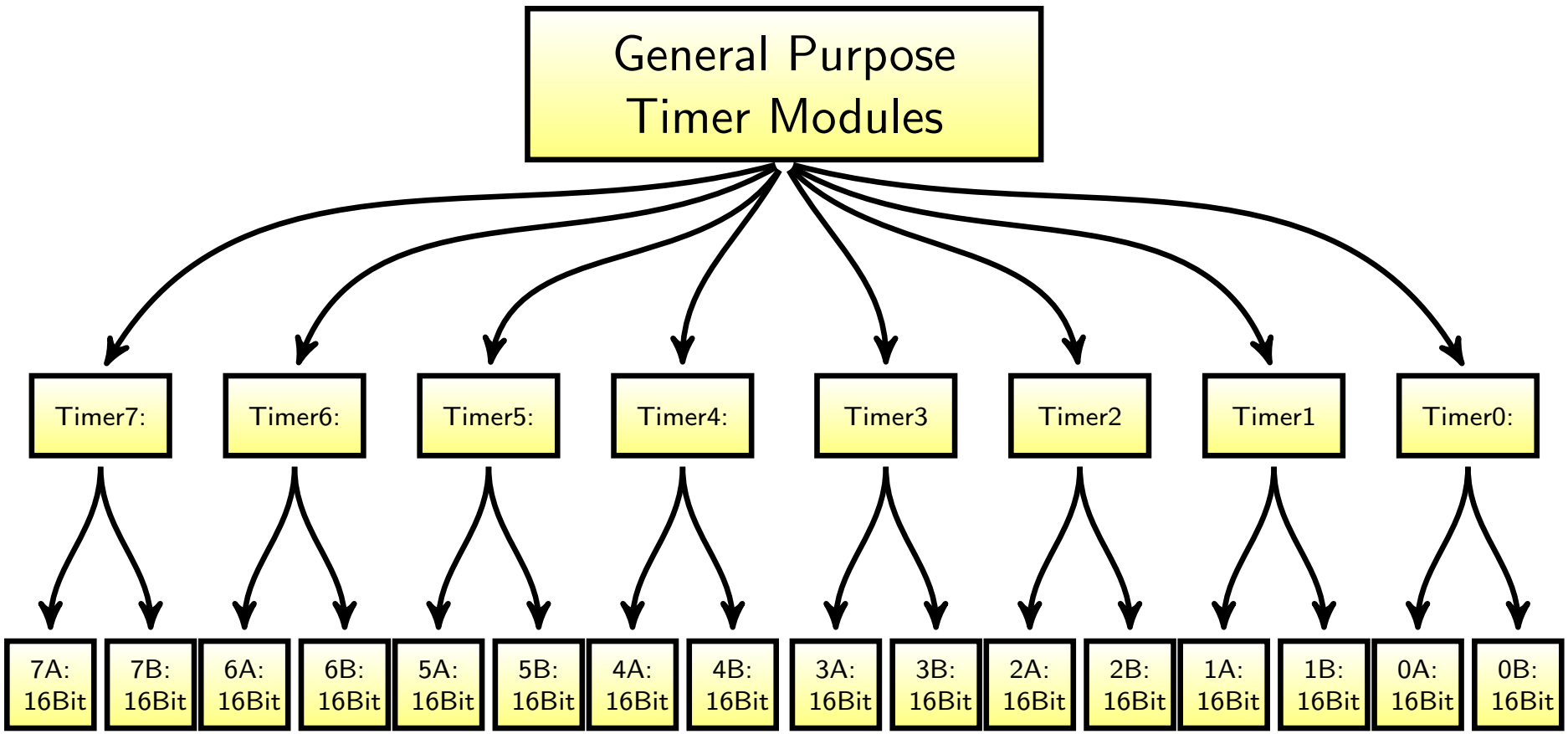
# GPTM Block-Diagramm



# General Purpose Timer im Controller TM4C1294

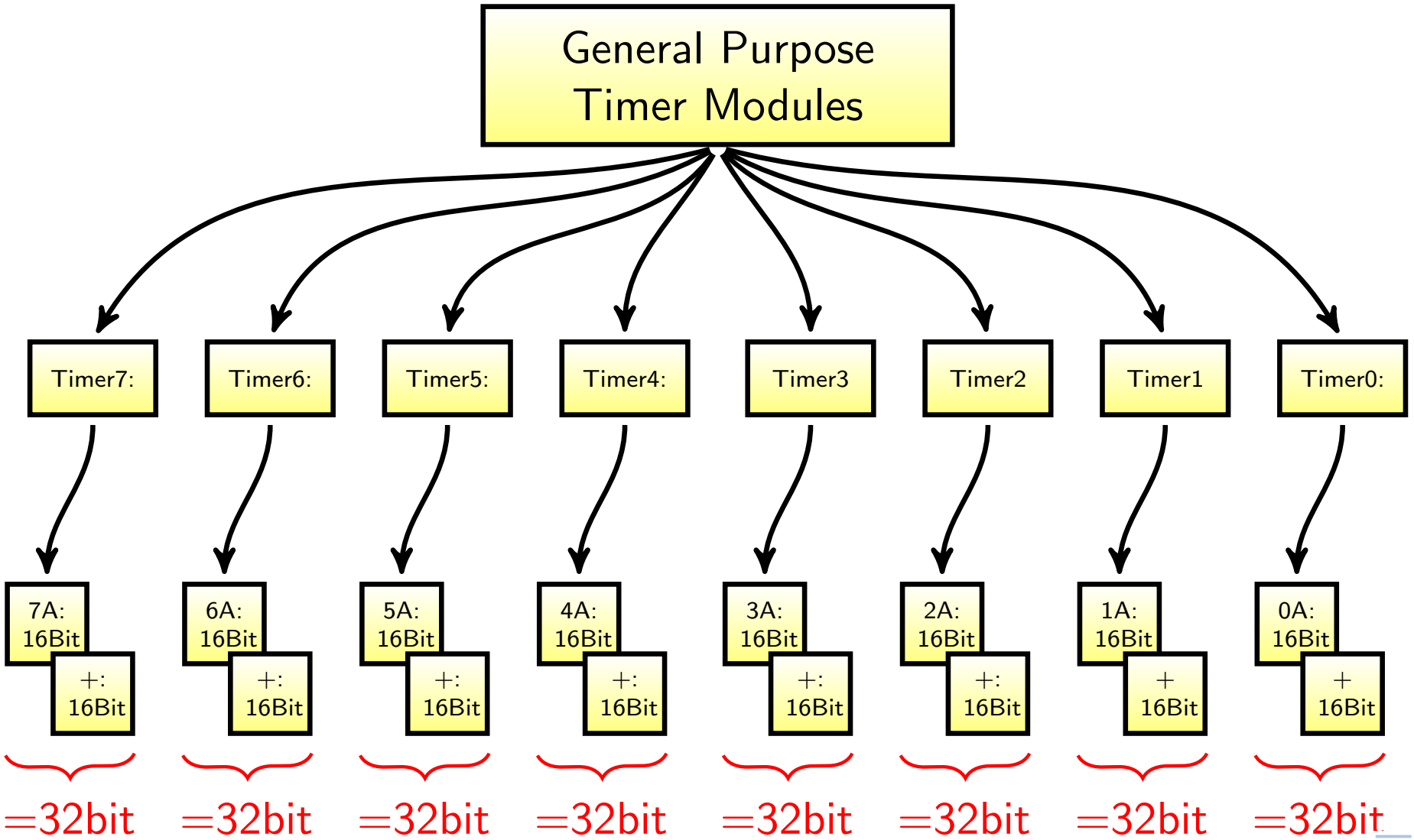
## Beispielkonfiguration als 8 Timer-Module je 2x16Bit

## Getrennt nutzbar als Timer{x}A und Timer{x}B, x=0,...,7



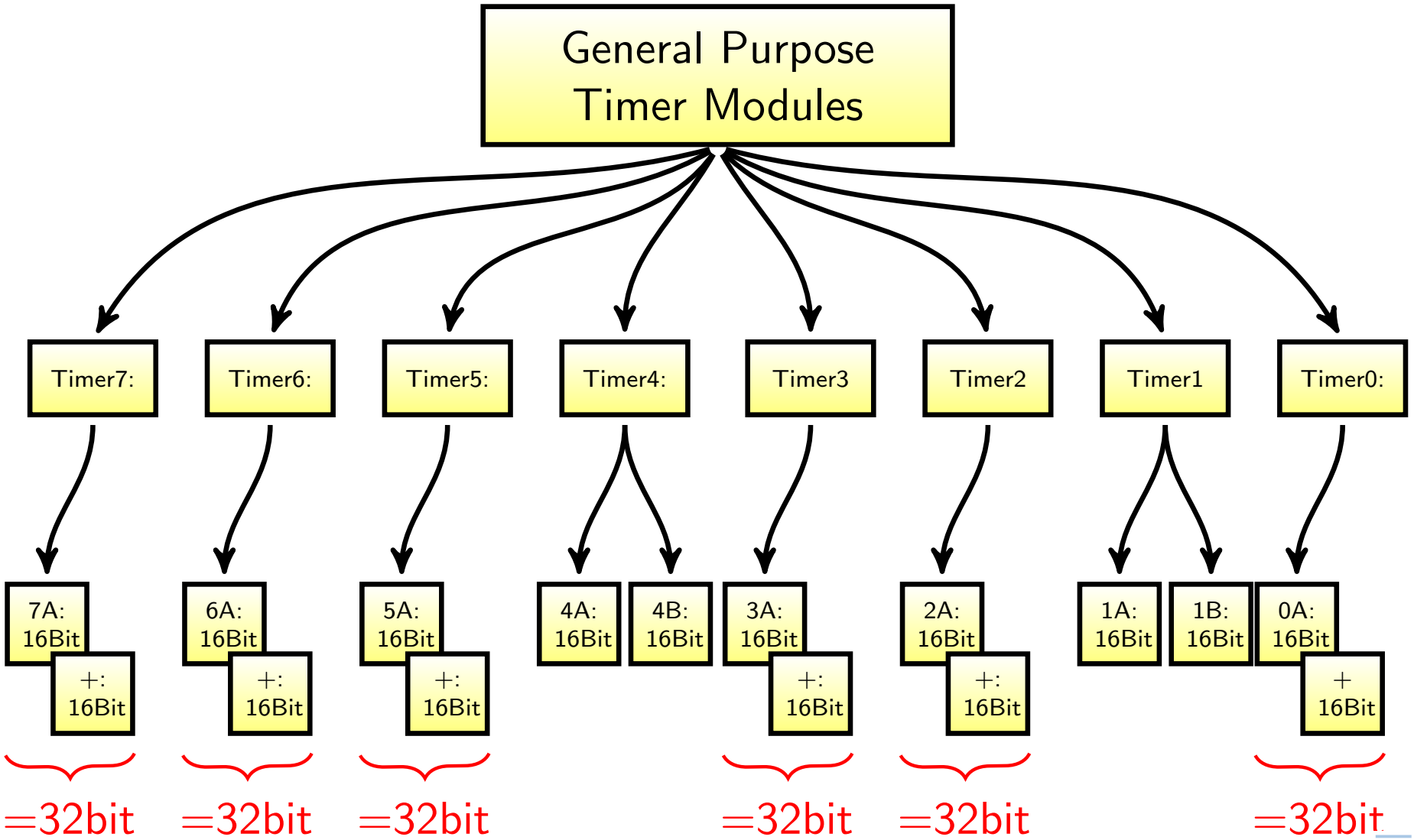
# General Purpose Timer im Controller TM4C1294

**Beispielkonfiguration als 8 Timer je 32Bit, Zugriff nur als Timer{x}A, wird Timer{x}B 'angehängt', ist aber nicht separat nutzbar, x=0,...,7**



# General Purpose Timer im Controller TM4C1294

## gemischte Beispielkonfiguration als 6 Timer je 32Bit (Timer{x}A) und 2 Timer je 2x16 Bit (Timer{x}A und Timer{x}B),x=0,...,7



# General-Purpose Timer TM4C1294 - Konfiguration der Arbeitsmodi

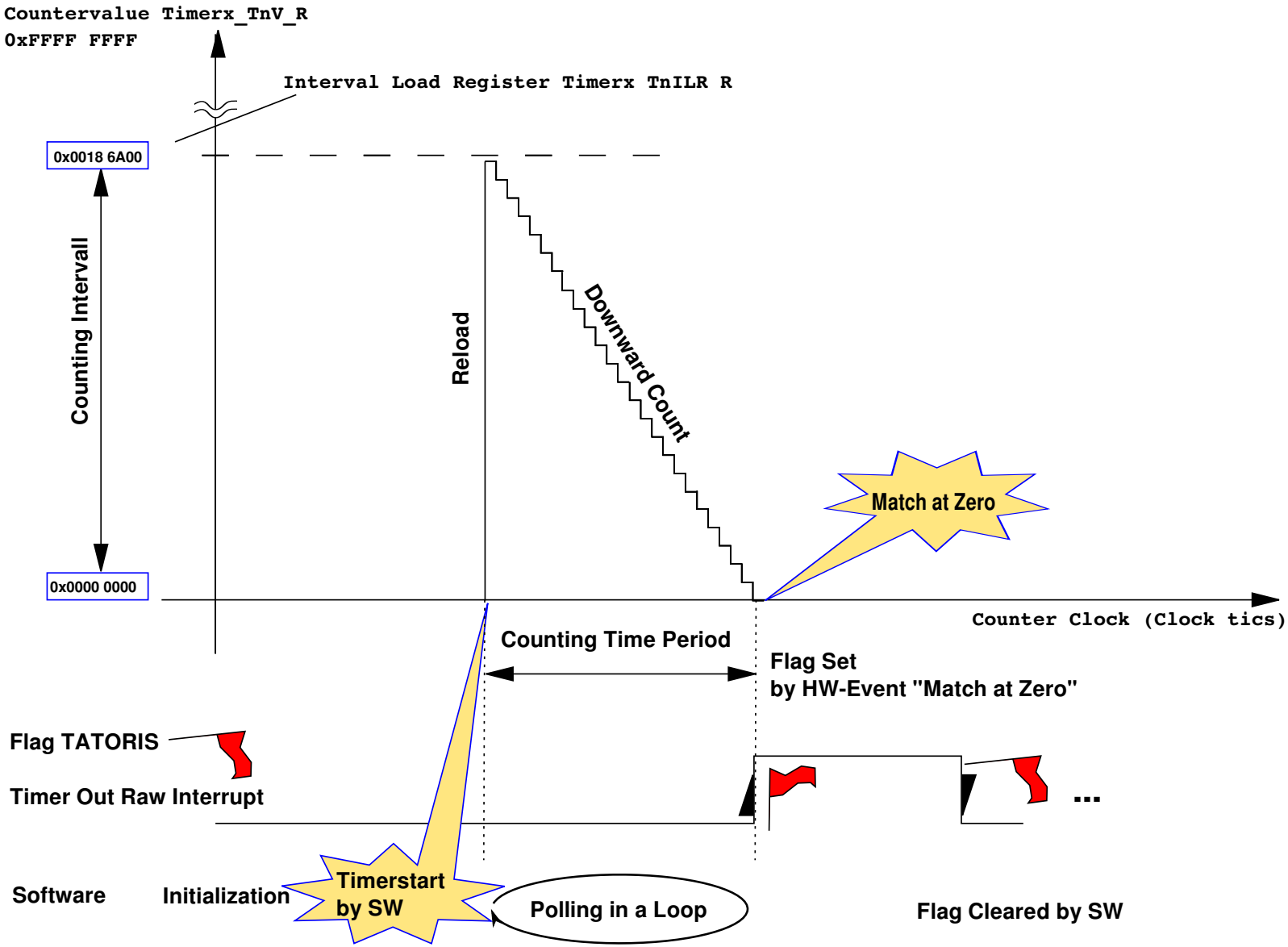
| Principle | Mode | TimerA/B | Counter | Prescaler | Count Dir. |
|---|---|---|---|---|---|
| Compare | One-shot | A,B Individual | 2x16-bit | 8-bit (opt.) | Up or Down |
| | | Concatenated | 1x32-bit | - | Up or Down |
| | Periodic | A,B Individual | 2x16-bit | 8-bit (opt.) | Up or Down |
| | | Concatenated | 1x32-bit | - | Up or Down |
| | Real Time Clock (RTC)*) | Concatenated | 1x32-bit | - | Up Only |
| | Pulse Width Modulation (PWM) | Individual | 16-bit | - | Down Only |
| Capture | Edge Count | A,B Individual | 2x16-bit | 8-bit (opt.) | Down Only |
| | Edge Time | A,B Individual | 2x16-bit | 8-bit (opt.) | Down Only |

*) erfordert 32,768 kHz Uhrenquarz

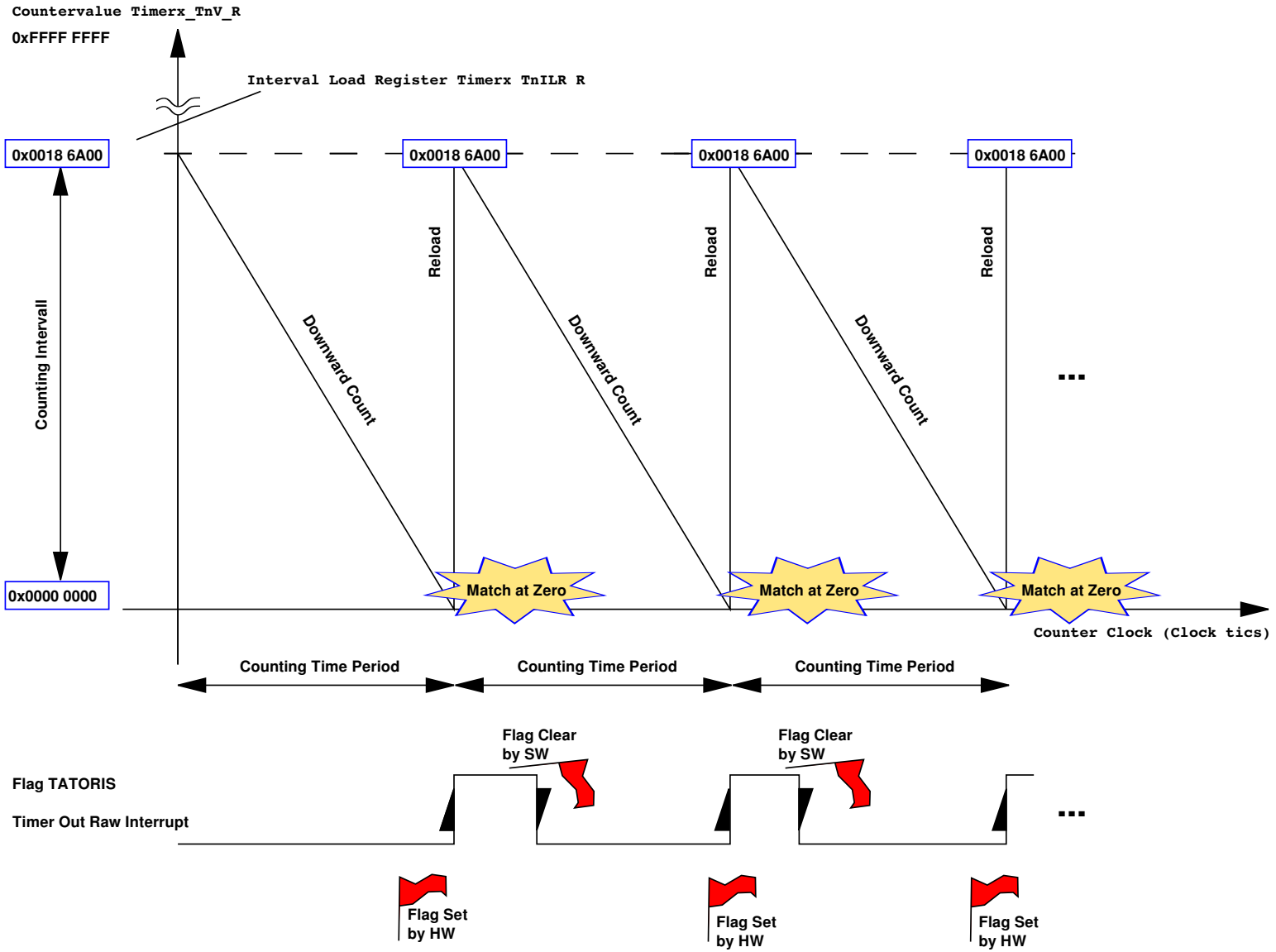
Erweiterte Beschreibung auch in Chapter 13.3, S.958 Texas Instruments Tiva TM4C1294NCPDT Microcontroller Datasheet, June 18, 2014

HAW HAMBURG

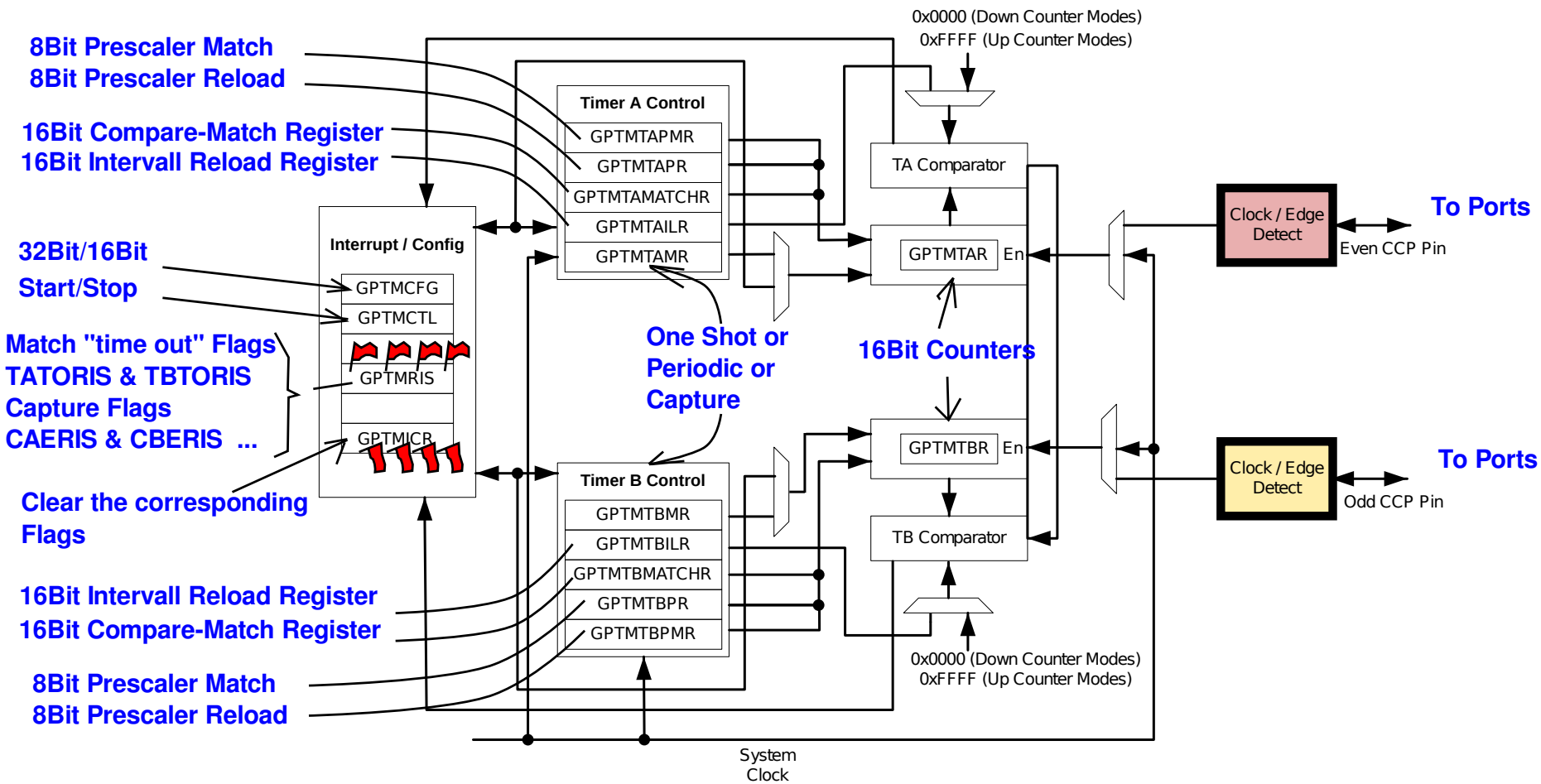
# One Shot Compare Match at Zero



# Peridioc Counting Compare Match at Zero

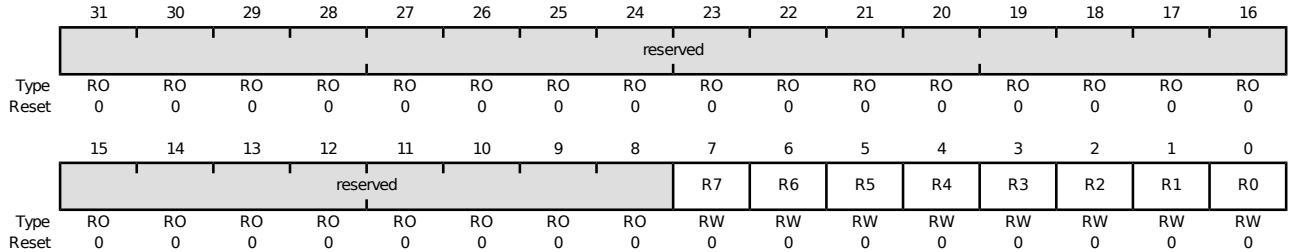


# Block Diagramm vereinfacht und kommentiert



# Timer x - Takt zuschalten im Run Mode Clock Gating Control Register 1 SYSCTL_RCGCTIMER_R

16/32-Bit General-Purpose Timer Run Mode Clock Gating Control (RCGCTIMER)




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0 | ... |
| | | | | 16/32-Bit General-Purpose Timer 7 Run Mode Clock Gating Control |
| 7 | R7 | RW | 0 | Value Description<br>0 16/32-bit general-purpose timer module 7 is disabled.<br>1 enable and provide a clock to 16/32-bit general-purpose timer |
| 0 | R0 | RW | 0 | ... |

Tiva TM4C1294 Mikrocontroller Data Sheet p. 380ff

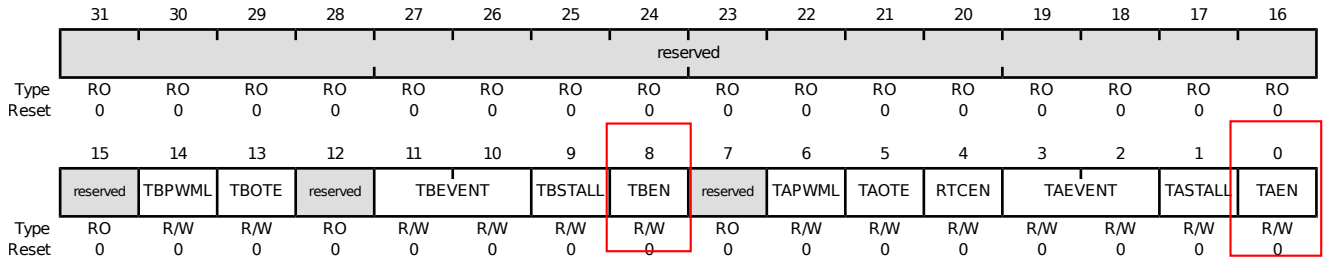
```
SYSCTL_RCGCTIMER_R=0x00000001;// clock enable timer 0
SYSCTL_RCGCTIMER_R=0x00000002;// clock enable timer 1
SYSCTL_RCGCTIMER_R=0x00000003;// clock enable timer 0+1
```

usw.

# Am Anfang jeder Konfiguration: Stop Timer x

## im GPTM Control Register `TIMERx_CTL_R x=0,...,7`

GPTM Control (GPTMCTL)




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| ... | | | | |
| 8 | TBEN | R/W | 0 | **TBEN** values are defined as follows:<br>Value Description<br>0 Timer B is disabled.<br>Timer B is enabled and begins counting or the capture logic is enabled based on the **GPTMCFG** register.1 |
| ... | | | | |
| 0 | TAEN | R/W | 0 | TAEN values are defined as follows:<br>Value Description<br>0 Timer A is disabled.<br>1 Timer A is enabled and begins counting or the capture logic is enabled based on the **GPTMCFG** register. |

**In 32 Bit mode is here to start / stop**

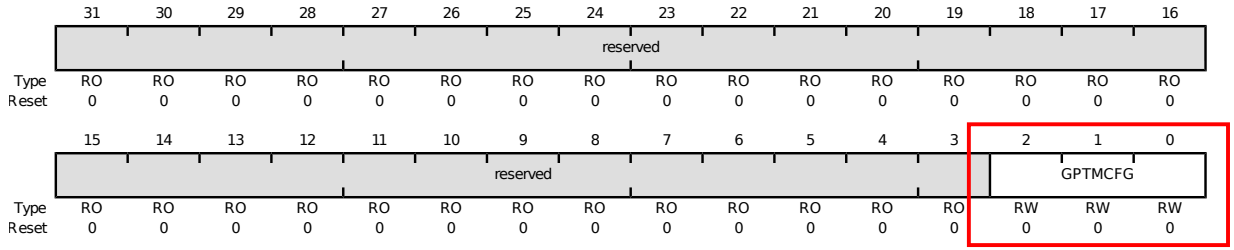
Tiva TM4C1294 Mikrocontroller Data Sheet ...

```
TIMER0_CTL_R &= 0xFFFFFFFE; // stop timer 0A
TIMER3_CTL_R &= ~0x00000100; // stop timer 3B
```

# Entscheidung der Konfiguration 32 Bit o. 2x16 Bit

## im GPTM Configuration Register `TIMERx_CFG_R x=0,...,7`

GPTM Configuration (GPTMCFG)




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2:0 | GPTMCFG | RW | 0x0 | GPTM Configuration<br>The `GPTMCFG` values are defined as follows: |

| Value | Description |
|---|---|
| 0x0 | For a 16/32-bit timer, this value selects the 32-bit timer configuration. |
| 0x1 | For a 16/32-bit timer, this value selects the 32-bit real-time clock (RTC) counter configuration. |
| 0x2-0x3 | Reserved |
| 0x4 | For a 16/32-bit timer, this value selects the 16-bit timer configuration.<br>The function is controlled by bits 1:0 of **GPTMTAMR** and **GPTMTBMR**. |
| 0x5-0x7 | Reserved |

Tiva TM4C1294 Mikrocontroller Data Sheet p. 376ff

```
TIMER0_CFG_R = 0x00000000; // set 32 bit mode timer 0
TIMER3_CFG_R = 0x00000004; // set 16 bit mode timer 3
```

# Entscheidung für One-Shot oder Periodic Mode

## im GPTM Mode Register `TIMERx_TnMR_R x=0,...,7 n=A,B`

GPTM Timer A Mode (GPTMTAMR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TCACT | | | TACINTD | TAPLO | TAMRSU | TAPWMIE | TAILD | TASNAPS | TAWOT | TAMIE | TACDIR | TAAMS | TACMR | TAMR | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| ... | | | | |
| 4 | TACDIR | RW | 0 | GPTM Timer A Count Direction<br>Value Description<br>0 The timer counts down.<br>The timer counts up. When counting up, the timer starts from a value of 0x0.1<br>When in PWM or RTC mode, the status of this bit is ignored. PWM mode always counts down and RTC mode always counts up. |
| ... | | | | |
| 1:0 | TAMR | RW | 0x0 | GPTM Timer A Mode<br>The TAMR values are defined as follows:<br>Value Description<br>0x0 Reserved<br>0x1 One-Shot Timer mode<br>0x2 Periodic Timer mode<br>0x3 Capture mode<br>The Timer mode is based on the timer configuration defined by bits 2:0 in the **GPTMCFG** register. |

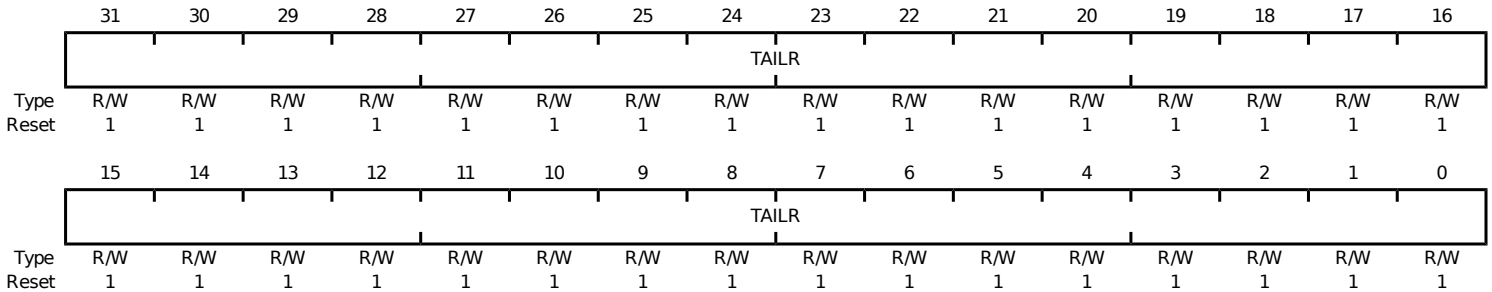
Tiva TM4C1294 Mikrocontroller Data Sheet p. 977ff

```
TIMER0_TAMR_R |= 0x00000002 ; // timer 0 periodic mode
TIMER0_TAMR_R &= 0xFFFFFFFE ; // -"-
```

# Load Value für das Zahlintervall

## in GPTM Interval Load Register `TIMERx_TnILR_R x=0,...,7 n=A,B`

GPTM Timer A Interval Load (GPTMTAILR)




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | TAILR | R/W | 0xFFFF.FFFF | GPTM Timer A Interval Load Register<br>Writing this field loads the counter for Timer A. A read returns the current value of **GPTMTAILR**. |

Tiva TM4C1294 Mikrocontroller Data Sheet p. 1004

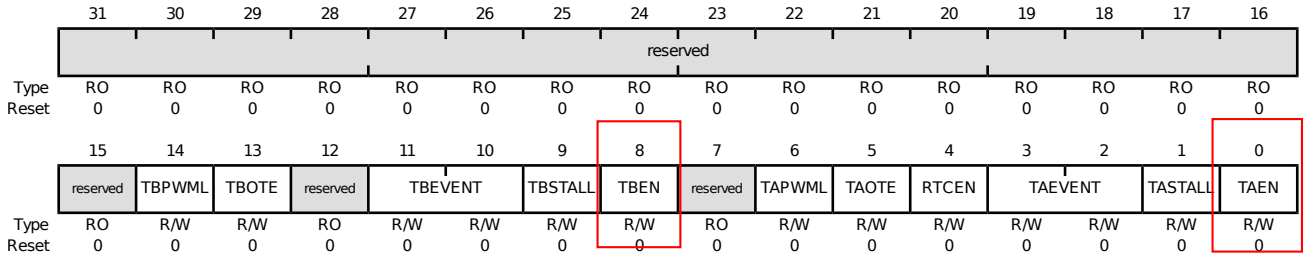
```
TIMER0_TAIL_R=16000000-1;//16 Mio = 1sec at 16MHz clk
TIMER5_TAIL_R=50000000-1;//50 Mio = 2sec at 25MHz clk
TIMER3_TBIL_R=   10000-1;//10000 = 10ms at 1MHz clk
```

usw.

# Start Timer x im GPTM Control Register

## TIMERx_CTL_R x=0,...,7

GPTM Control (GPTMCTL)




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| ... | | | | |
| 8 | TBEN | R/W | 0 | **TBEN** values are defined as follows:<br>Value Description<br>0 Timer B is disabled.<br>Timer B is enabled and begins counting or the capture logic is enabled based on the **GPTMCFG** register.1 |
| ... | | | | |
| 0 | TAEN | R/W | 0 | TAEN values are defined as follows:<br>Value Description<br>0 Timer A is disabled.<br>1 Timer A is enabled and begins counting or the capture logic is enabled based on the **GPTMCFG** register. |

**In 32 Bit mode is here to start / stop**

Tiva TM4C1294 Mikrocontroller Data Sheet p. ...

```
TIMER0_CTL_R |= 0x00000001; // start timer 0A
TIMER3_CTL_R |= 0x00000100; // start timer3B
```

# Read-Only Polling der Flags 'Timer Time-Out'

## im GPTM Interrupt Raw Register `TIMERx_RIS_R x=0,...,7`

GPTM Raw Interrupt Status (GPTMRIS)

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | DMABRIS | reserved | TBMRIS | CBERIS | CBMRIS | TBTORIS | reserved | | DMAARIS | TAMRIS | RTCRIS | CAERIS | CAMRIS | TATORIS |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

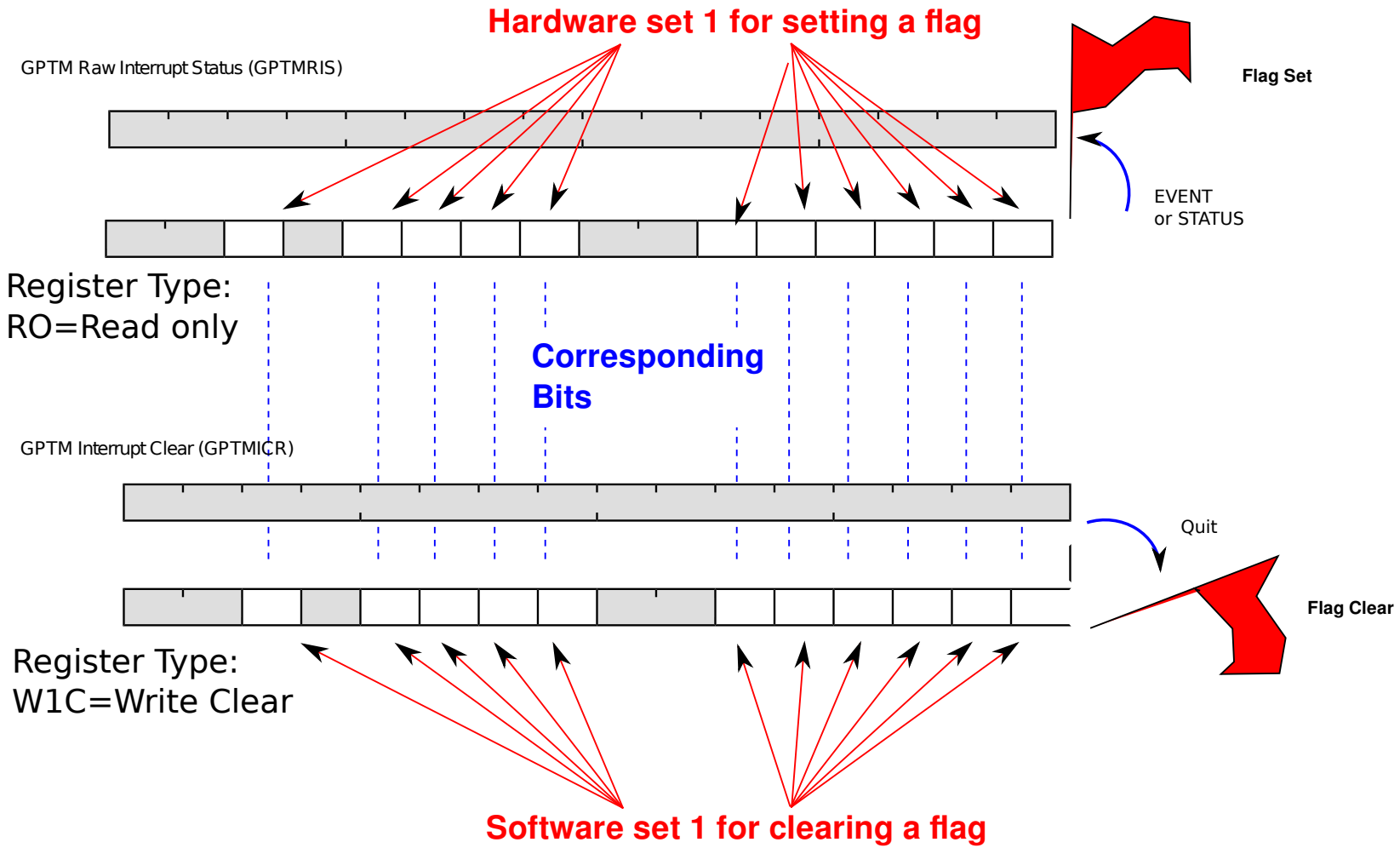
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 8 | TBTORIS | RO | 0 | GPTM Timer B Time-Out Raw Interrupt<br>Value Description<br>1 Timer B has timed out.<br>0 Timer B has not timed out.<br>This bit is cleared by writing a 1 to the TBTOCINT bit in the **GPTMICR** register. |
| 0 | TATORIS | RO | 0 | GPTM Timer A Time-Out Raw Interrupt<br>Value Description<br>1 Timer A has timed out.<br>0 Timer A has not timed out.<br>This bit is cleared by writing a 1 to the TATOCINT bit in the **GPTMICR** register. |

Tiva TM4C1294 Mikrocontroller Data Sheet p. 996ff

```
while (TIMER0_RIS & 0x1) ==0x0);// polling flag TATORIS timer0A
while (!(TIMER3_RIS & 0x400) );// polling flag CBERIS timer3B
```

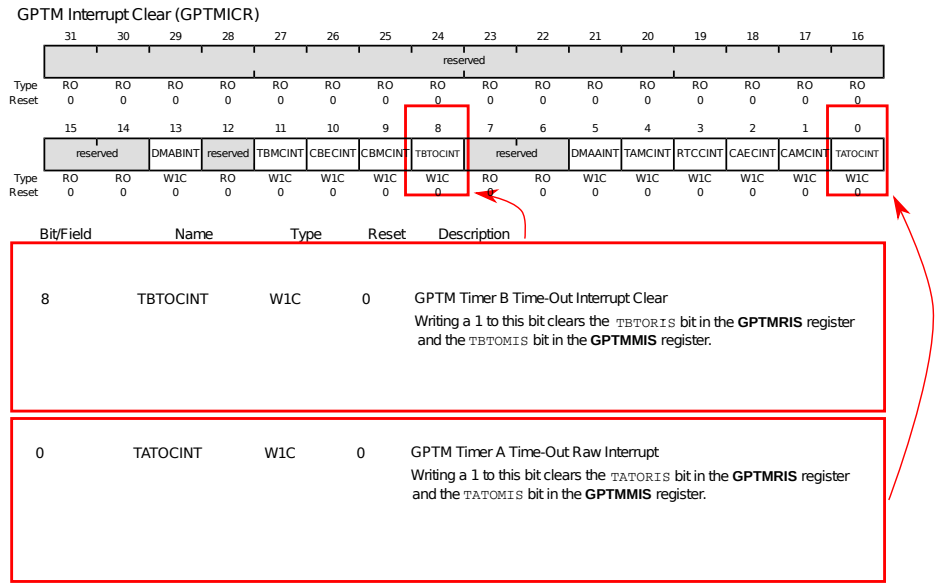
# Arbeitsweise Timerflags: Korrespondierende Bits

## in getrennten Status- und Clear-Register




Diese Trennung erfolgt bei anderen Controllern/ Peripherie-Modulen oft nicht, dann wird im Statusregister durch die Software nach dem Auslesen mit 0 überschrieben.

# Löschen der Flags im separaten Clear-Register: Timer x `Timerx_ICR_R x=0,...,7`




Tiva TM4C1294 Mikrocontroller Data Sheet p. 1002

```
TIMER0_ICR_R|= 0x00000001;// clear flag TATORIS timer 0A
TIMER3_ICR_R|= 0x0000400;// clear flag CBERIS timer3B
```

usw.

HAW HAMBURG

# Beispiel: 32 Bit + Compare + Periodic Mode (I)

Schritt 0: Port-Takt zuschalten, Ports für die Ausgaben vorbereiten

GPTM in simple 32Bit compare and periodic mode to control port output 1 of 3

```
1  // Simple compare example 32bit for LED binary counting
2  // with approx. 100ms counting period
3  // (= 100ms counter match for "counter time out")
4  // internal RC-oscillator (16 MHz, not precise(!)) is used
5  #include <stdint.h>
6  #include "inc/tm4c1294ncpdt.h"
7
8  void main(void)
9  {
10     int wt=0;            // aux. variable for very short time wait
11     // GPIO Port initialize
12     SYSCTL_RCGCGPIO_R = 0x00000800;  wt++;// clock enable port M, wait for stable clock
13     GPIO_PORTM_DEN_R = 0xFF;      // digital enable I /O all pins PPJ7..PJ0
14     GPIO_PORTM_DIR_R = 0xFF;      // PJ7..PJ0 is output


....
```

# Beispiel: 32 Bit + Compare + Periodic Mode (II)

Schritt 1 Timer 0 initialisieren:
Timer-Takt zuschalten, Timer stoppen
32 Bit konfigurieren, Periodic Mode starten, Downward counting
Startvalue errechnen und setzen
Timer starten
....

GPTM in simple 32Bit compare and periodic mode to control port output 2 of 3

```
15
16      // GPTM timer 0 initialize
17      SYSCTL_RCGCTIMER_R = 0x00000001; wt++; // clock enable timer
18      TIMER0_CTL_R    &= 0xFFFFFFFE;     // stop timer 0
19      TIMER0_CFG_R    =0x00000000;      // timer 0 in 32 bit mode
20      TIMER0_TAMR_R   |= 0x02;          // timer 0 in periodic mode
21      TIMER0_TAMR_R   &= 0xFFFFFFEF;   // timer in downward counting mode
22      //TIMER0_TAMR_R |= 0x10;         // timer in upward counting mode (alternative)
23      TIMER0_TAILR_R = 1600000-1 ;     // start value 0.1 sec =16MHz/1.6Mio
24      TIMER0_ICR_R   |= 1;              // clear timeout flag of timer 0A
25      TIMER0_CTL_R   |= 0x0000001;     // start timer 0
```

....

# Beispiel: 32 Bit + Compare + Periodic Mode (III)

Schritt 3: Mit dem Timer 'arbeiten'
Anfängliche Portausgabe
Endlosschleife:
Warteschleife und Polling bis Compare Flag TATORIS gesetzt ist
Portausgabe, einen Dualwert erhöhen
Flag clear, zurück zum Warten und Polling
....

GPTM in simple 32Bit compare and periodic mode to control port output 3 of 3

```
26
27      GPIO_PORTM_DATA_R=0x00;         // output  zero to PJ7..PJ0
28      while(1)
29      {
30        while(!(TIMER0_RIS_R & 0x00000001)); // wait and poll flag for timer 0 timeout
31        GPIO_PORTM_DATA_R = (GPIO_PORTM_DATA_R + 0x01) & 0xFF ;  // "count 8 output LED"
32        TIMER0_ICR_R |= 0x00000001;           // clear flag of timer 0
33      }
34  }
```

## now complete: GPTM in simple 32Bit compare and periodic mode to control port output

```
1  // Simple compare example 32bit for LED binary counting
2  // with approx. 100ms counting period
3  // (= 100ms counter match for "counter time out")
4  // internal RC-oscillator (16 MHz, not precise(!)) is used
5  #include <stdint.h>
6  #include "inc/tm4c1294ncpdt.h"
7
8  void main(void)
9  {
10     int wt=0;            // aux. variable for very short time wait
11     // GPIO Port initialize
12     SYSCTL_RCGCGPIO_R = 0x00000800;  wt++;// clock enable port M, wait for stable clock
13     GPIO_PORTM_DEN_R = 0xFF;       // digital enable I /O all pins PPJ7..PJ0
14     GPIO_PORTM_DIR_R = 0xFF;       // PJ7..PJ0 is output
15
16     // GPTM timer 0 initialize
17     SYSCTL_RCGCTIMER_R = 0x00000001; wt++; // clock enable timer
18     TIMER0_CTL_R   &= 0xFFFFFFFE;     // stop timer 0
19     TIMER0_CFG_R   =0x00000000;      // timer 0 in 32 bit mode
20     TIMER0_TAMR_R  |= 0x02;          // timer 0 in periodic mode
21     TIMER0_TAMR_R  &= 0xFFFFFFEF;   // timer in downward counting mode
22     //TIMER0_TAMR_R |= 0x10;         // timer in upward counting mode (alternative)
23     TIMER0_TAILR_R = 1600000-1 ;    // start value 0.1 sec =16MHz/1.6Mio
24     TIMER0_ICR_R  |= 1;              // clear timeout flag of timer 0A
25     TIMER0_CTL_R  |= 0x0000001;     // start timer 0
26
27     GPIO_PORTM_DATA_R=0x00;         // output  zero to PJ7..PJ0
28     while(1)
29     {
30       while(!(TIMER0_RIS_R & 0x00000001)); // wait and poll flag for timer 0 timeout
31       GPIO_PORTM_DATA_R = (GPIO_PORTM_DATA_R + 0x01) & 0xFF ;  // "count 8 output LED"
32       TIMER0_ICR_R |= 0x00000001;          // clear flag of timer 0
33     }
34 }
```

## slightly varied : GPTM: 32Bit compare one shot mode upward counting controls port output

```
1  // Simple compare example 32bit for LED binary counting
2  // Remark for Differences <===== ONE SHOT MODE, UPWARD COUNTING
3  // Approx. 500ms counting period internal RC-oscillator (16 MHz, not precise(!))
4  #include <stdint.h>
5  #include "inc/tm4c1294ncpdt.h"
6
7  void main(void)
8  {
9      int wt=0;            // aux. variable for very short time wait
10     // GPIO Port initialize
11     SYSCTL_RCGCGPIO_R = 0x00000800;  wt++;// clock enable port J, wait for stable clock
12     GPIO_PORTM_DEN_R = 0xFF;       // digital enable I /O all pins PPJ7..PJ0
13     GPIO_PORTM_DIR_R = 0xFF;       // PJ7..PJ0 is output
14
15     // GPTM timer 0 initialize
16     SYSCTL_RCGCTIMER_R = 0x00000001; wt++; // clock enable timer
17     TIMER0_CTL_R   &= 0xFFFFFFFE;     // stop timer 0
18     TIMER0_CFG_R   =0x00000000;      // timer 0 in 32 bit mode
19     TIMER0_TAMR_R  |= 0x01;          // timer 0 in one shot mode <========
20     //TIMER0_TAMR_R  &= 0xFFFFFFEF;  // timer in downward counting mode
21     TIMER0_TAMR_R |= 0x10;        // timer in upward counting mode  <========
22     TIMER0_TAILR_R = 8000000-1 ;    // start value 0.5 sec =16MHz/8Mio <========
23     TIMER0_ICR_R  |= 1;              // clear timeout flag of timer 0A
24     TIMER0_CTL_R  |= 0x0000001;     // start timer 0
25
26     GPIO_PORTM_DATA_R=0x00;         // output  zero to PJ7..PJ0
27     while(1)
28     {
29         while(!(TIMER0_RIS_R & 0x00000001)); // wait and poll flag for timer 0 timeout
30         GPIO_PORTM_DATA_R = (GPIO_PORTM_DATA_R + 0x01) & 0xFF ;  // "count 8 output LED"
31         TIMER0_ICR_R |= 0x00000001;           // clear flag of timer 0
32         TIMER0_CTL_R  |= 0x0000001;     // restart timer 0  <========
33     }
34 }
```

HAW HAMBURG

# 16 Bit Mode im GPTM Configuration Register

## Timerx_CFG_R x=0,1,2,3

GPTM Configuration (GPTMCFG)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | | GPTMCFG | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2:0 | GPTMCFG | RW | 0x0 | GPTM Configuration<br>The GPTMCFG values are defined as follows: |

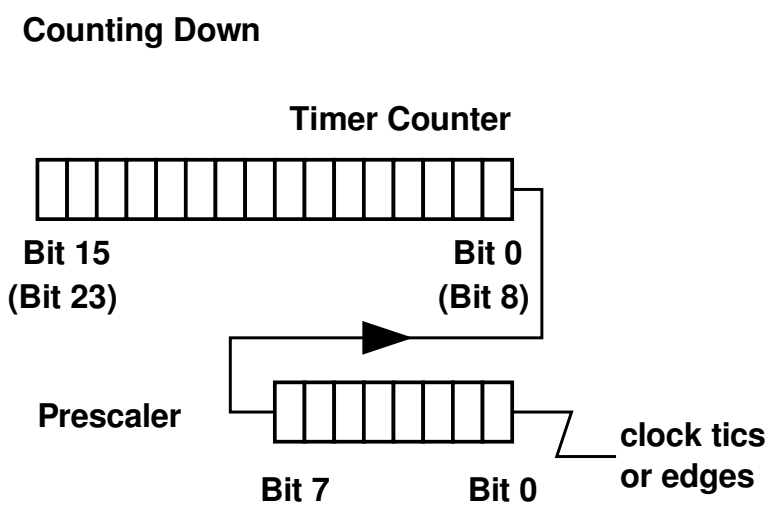
| Value | Description |
|---|---|
| 0x0 | For a 16/32-bit timer, this value selects the 32-bit timer configuration. |
| 0x1 | For a 16/32-bit timer, this value selects the 32-bit real-time clock (RTC) counter configuration. |
| 0x2-0x3 | Reserved |
| 0x4 | For a 16/32-bit timer, this value selects the 16-bit timer configuration.<br>The function is controlled by bits 1:0 of **GPTMTAMR** and **GPTMTBMR**. |
| 0x5-0x7 | Reserved |

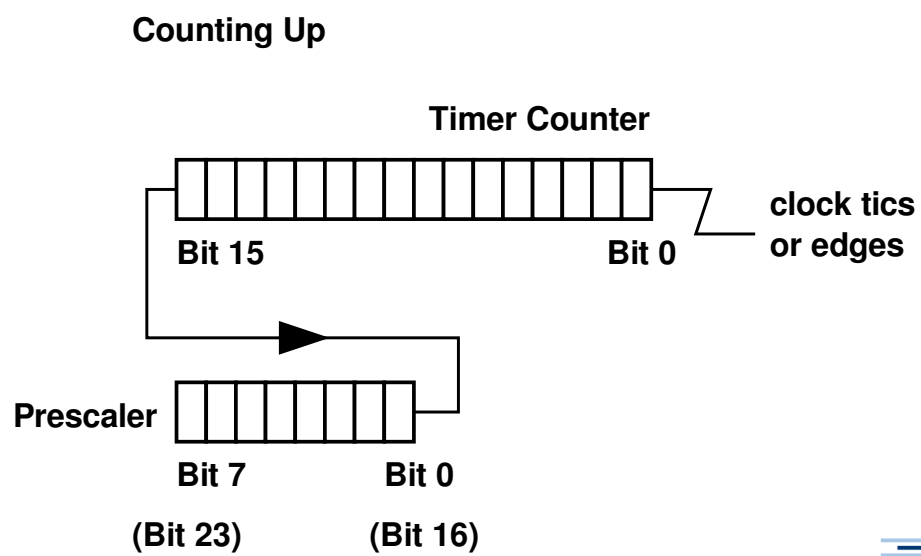
LTiva TM4C1294 Mikrocontroller Data Sheet p. 376ff

```
TIMER3_CFG_R = 0x00000004; // set 16 bit mode timer 3
```

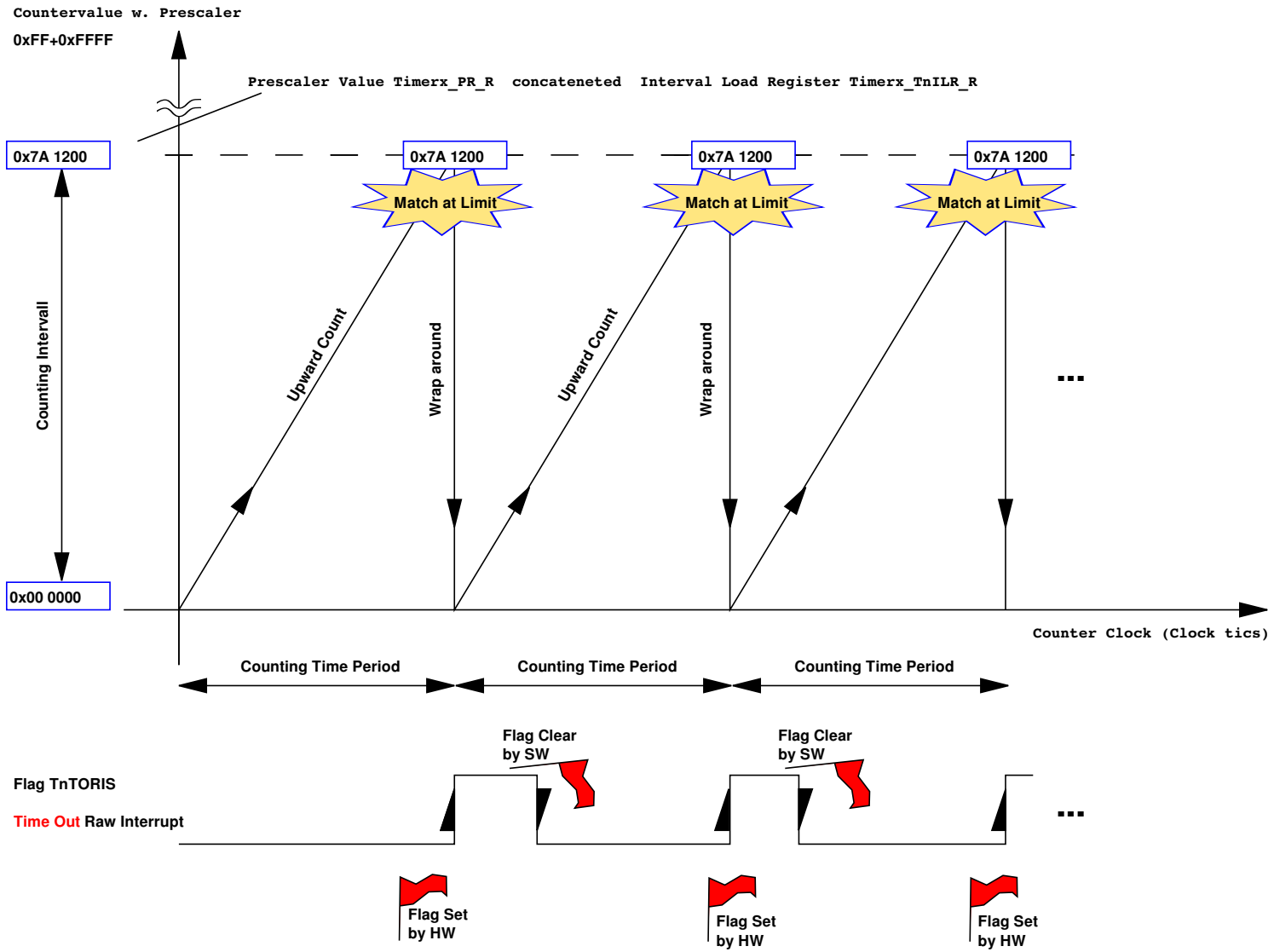
# Prescaler erweitert 16-Bit auf 24-Bit in der Compare- und Edge-Count-Capture-Funktion

- In der Zählrichtung counting down:
  - arbeitet der Prescaler als vorgeschalteter Vorteiler 'true prescaler'
  - und enthält die niedrigsten Bits des Zählwertes
- In der Zählrichtung counting up:
  - arbeitet der Prescaler als Zähler-Verlängerung nach 'oben'
  - und enthält die höchsten Bits des Zählwertes



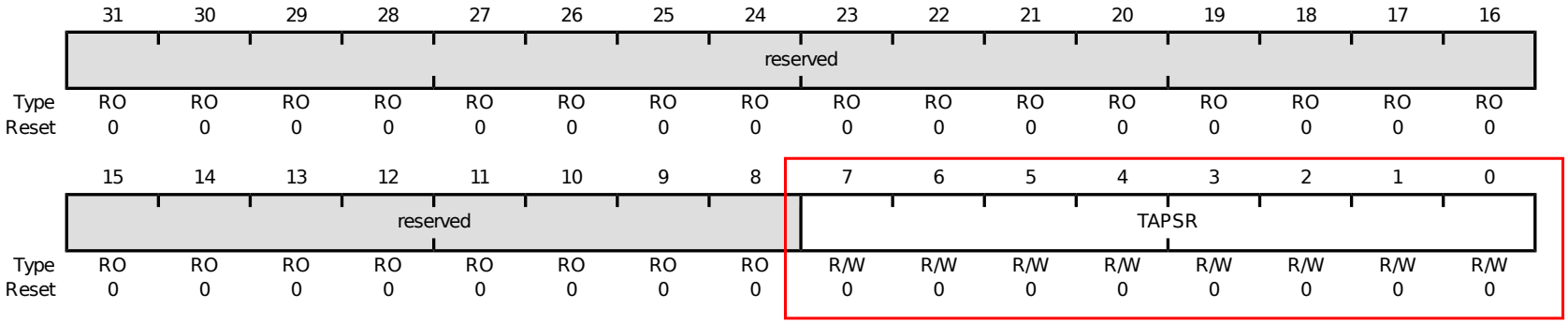
# Peridioc Counting Upward w. Prescaler Compare - Match at Limit



# Setzen des Vorteilers im GPTM Prescale Register

## TIMERx_TnPR_R x=0,..,7 n=A,B

GPTM Timer A Prescale (GPTMTAPR)




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7:0 | TAPSR | R/W | 0x00 | GPTM Timer A Prescale<br>The register loads this value on a write. A read returns the current value of the register.<br>Refer to Table 10-5 on page 532 for more details and an example. |

```
TIMER3_TBPR_R = 100-1; // Set Prescaler 100
```

# Periodic Mode + Aufwärtszählen GPTM Mode Register `Timerx_TnMR_R x=0,1,2,3 n=A,B`

GPTM Timer A Mode (GPTMTAMR)

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TCACT | | | TACINTD | TAPLO | TAMRSU | TAPWMIE | TAILD | TASNAPS | TAWOT | TAMIE | TACDIR | TAAMS | TACMR | TAMR | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| ... | | | | |
| 4 | TACDIR | RW | 0 | GPTM Timer A Count Direction<br>Value Description<br>0 The timer counts down.<br>The timer counts up. When counting up, the timer starts from a value of 0x0.1<br>When in PWM or RTC mode, the status of this bit is ignored. PWM mode always counts down and RTC mode always counts up. |
| ... | | | | |
| 1:0 | TAMR | RW | 0x0 | GPTM Timer A Mode<br>The `TAMR` values are defined as follows:<br>Value Description<br>0x0 Reserved<br>0x1 One-Shot Timer mode<br>0x2 Periodic Timer mode<br>0x3 Capture mode<br>The Timer mode is based on the timer configuration defined by bits 2:0 in the **GPTMCFG** register. |

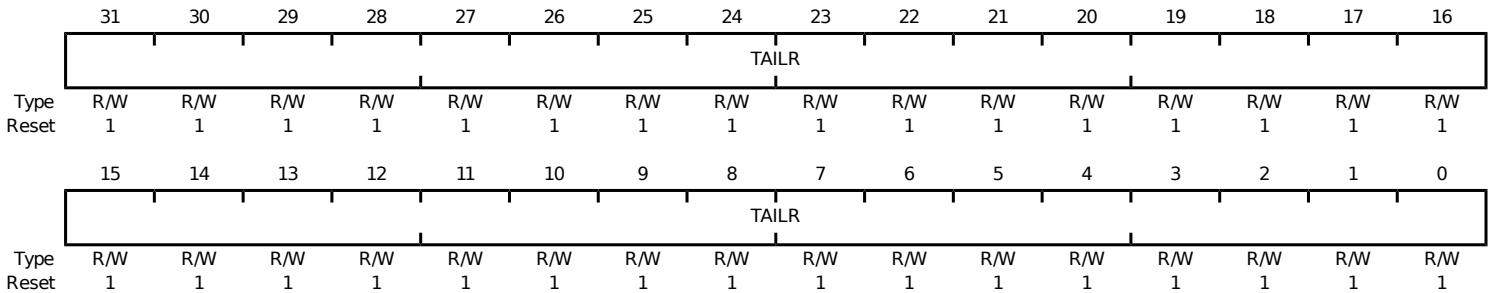
LM3S9B92 Microcontroller Data Sheet p. 543

```
TIMER3_TBMR_R |= 0x00000002;// timer 3B periodic mode
TIMER3_TBMR_R |= 0x00000010;// timer 3B upward counting
```

# Limit Value in GPTM Interval Load Register

## Timerx_TnILR_R x=0,1,2,3 n=A,B

GPTM Timer A Interval Load (GPTMTAILR)



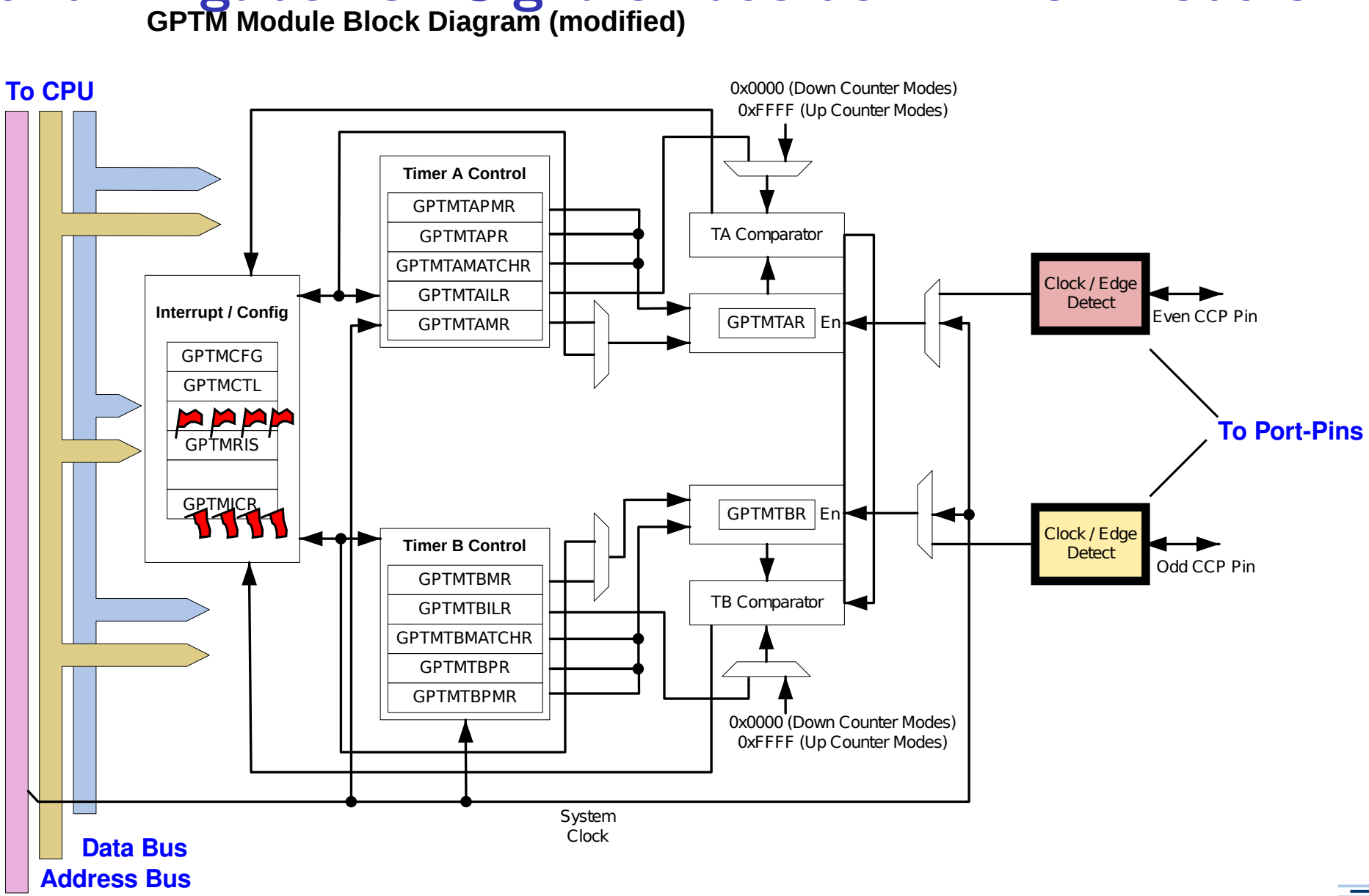

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | TAILR | R/W | 0xFFFF.FFFF | GPTM Timer A Interval Load Register<br>Writing this field loads the counter for Timer A. A read returns the current value of **GPTMTAILR**. |

```
TIMER3_TBIL_R = 50000-1;
// 50000 times 1/(16MHz clk / Prescaler)
```

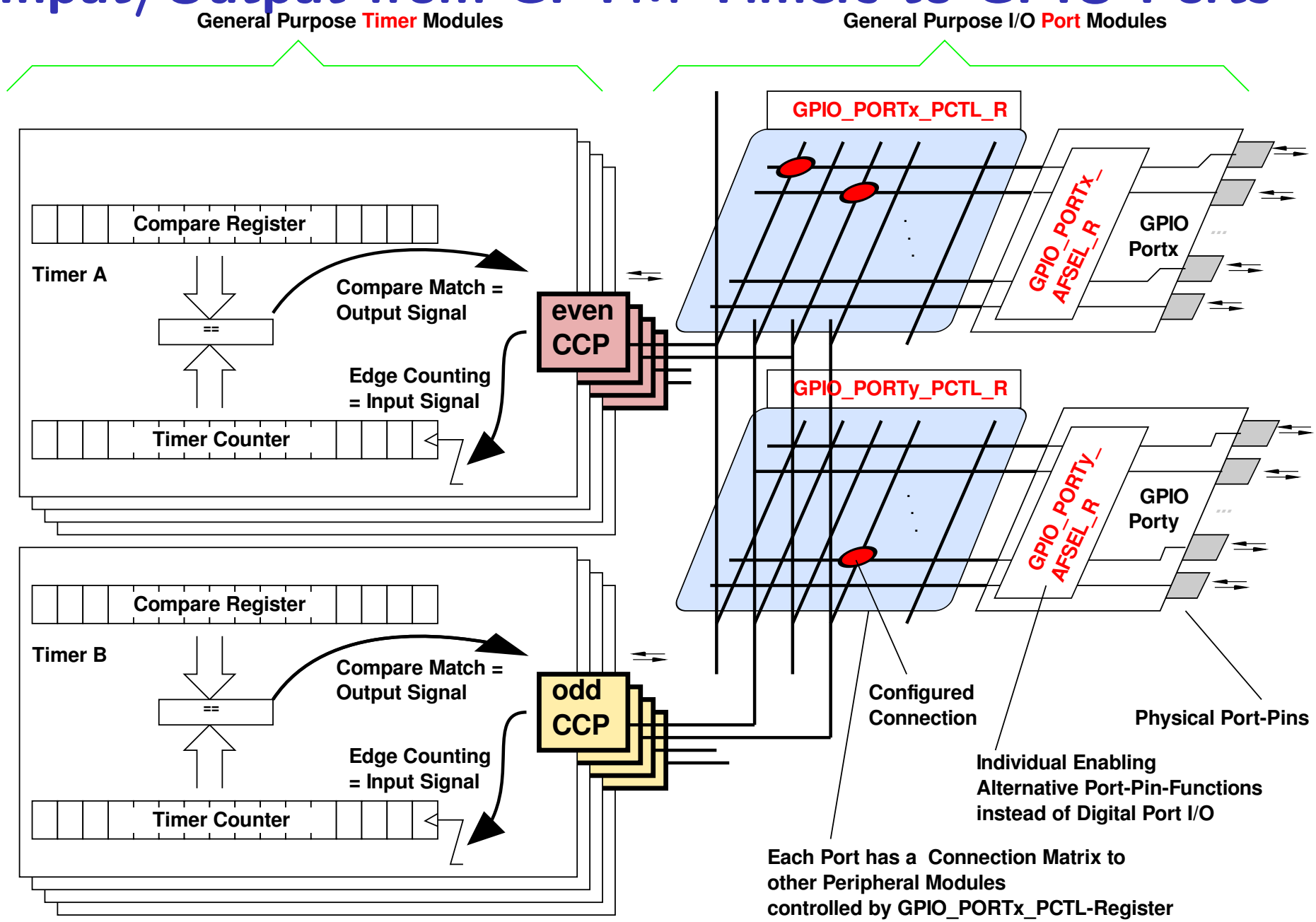
# GPTM in Simple 16Bit Downward Counting Compare Mode with Port Output (Blinking) 1 of 1

```
1  // Simple compare example 16bit w. prescaler for on board LED blinking
2  // with 400ms period (= 200ms on 200ms off)
3  // internal RC-oscillator approx. 16MHz
4  #include <stdint.h>
5  #include "inc/tm4c1294ncpdt.h"
6  void main(void)
7  {
8  int wt=0;
9  SYSCTL_RCGCGPIO_R = 0x00001000;  wt++;// clock enable port N + wait for stable clock
10 GPIO_PORTN_DEN_R |= 0x02;     // digital enable I /O pin PN1
11 GPIO_PORTN_DIR_R |= 0x02;     // PN1 set to output
12 // GPTM timer 3B
13 SYSCTL_RCGCTIMER_R= 0x00000008; wt++; // clock enable timer 3
14 TIMER3_CTL_R     &= ~0x0000100;    // stop timer 3B
15 TIMER3_CFG_R      = 0x00000004;    // set timer 3 in 16 bit mode
16 TIMER3_TBPR_R     = 160-1;         // set prescaler intervall timer3B for scaling with 160
17 TIMER3_TBMR_R    |= 0x02;          // set timer 3B in periodic mode
18 TIMER3_TBMR_R    &= ~0x11;         // set downward counting
19 TIMER3_TBILR_R    = 20000-1;       // set intervall reload value
20 TIMER3_CTL_R     |= 0x0000100;     // start timer3B
21 //Calculate: (160 * 20000 ) /16.000.000 =  0.2 s = 200ms
22
23 GPIO_PORTN_DATA_R &=~0x02;        // inital output PN1 = 0
24 while(1)
25   {
26     while((TIMER3_RIS_R & 0x100)==0x000); // wait and poll timer 3B timeout flag
27     GPIO_PORTN_DATA_R  ^=0x02;     // toggle PN1 output with bitwise XOR
28     TIMER3_ICR_R |= 0x00100;       // clear flag of timer 3B
29   }
30 }
```

# Capture Compare PWM Pins (CCP) - Direkte Aus- und Eingabe von Signalen aus den Timer Modulen



# Input/Output from GPTM-Timers to GPIO-Ports



# Welche GPIO-Port-Pins sind CCP-Pins ?

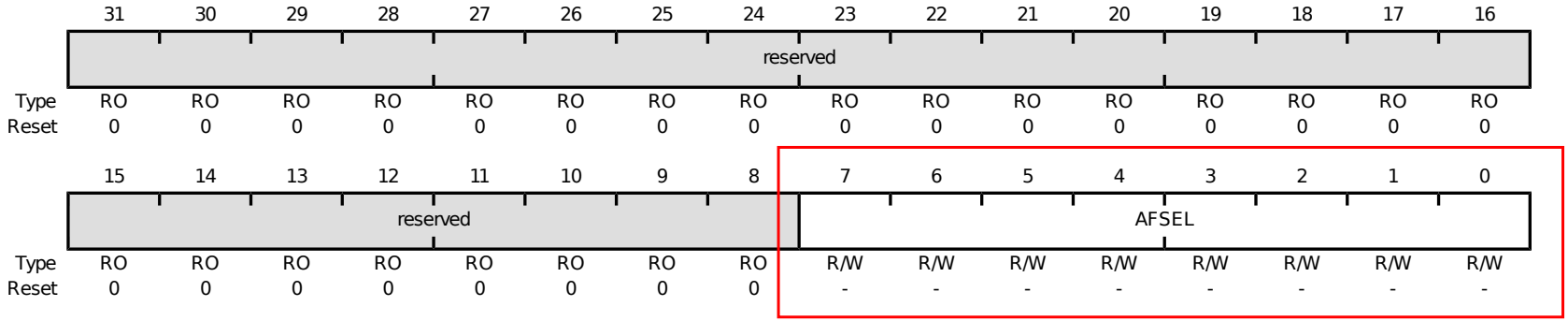
| Timer Module | Name | Gehäuse-Pins 128TQFP | PortPins | Portcontrol Bitfield PCTL |
|---|---|---|---|---|
| Timer 0A | T0CCP0 | 1 | PD0 | 3 |
| | | 33 | PA0 | 3 |
| | | 85 | PL4 | 3 |
| Timer 0B | T0CCP1 | 2 | PD1 | 3 |
| | | 34 | PA1 | 3 |
| | | 86 | PL5 | 3 |
| Timer 0A | T1CCP0 | 3 | PD2 | 3 |
| | | 35 | PA2 | 3 |
| | | 94 | PL6 | 3 |
| Timer 1B | T1CCP1 | 4 | PD3 | 3 |
| | | 36 | PA3 | 3 |
| | | 93 | PL7 | 3 |
| Timer 2A | T2CCP0 | 37 | PA4 | 3 |
| | | 78 | PM0 | 3 |
| Timer 2B | T2CCP1 | 38 | PA5 | 3 |
| | | 77 | PM1 | 3 |
| Timer 3A | T3CCP0 | 40 | PA6 | 3 |
| | | 76 | PM2 | 3 |
| | | 125 | PD4 | 3 |
| Timer 3B | T3CCP1 | 41 | PA7 | 3 |
| | | 75 | PM3 | 3 |
| | | 126 | PD5 | 3 |
| Timer 4A | T4CCP0 | 74 | PM4 | 3 |
| | | 95 | PB0 | 3 |
| | | 127 | PD6 | 3 |
| Timer 4B | T4CCP1 | 73 | PM5 | 3 |
| | | 96 | PB1 | 3 |
| | | 128 | PD7 | 3 |
| Timer 5A | T5CCP0 | 72 | PM6 | 3 |
| | | 91 | PB2 | 3 |
| Timer 5B | T5CCP1 | 71 | PM7 | 3 |
| | | 92 | PB3 | 3 |
| Timer 6A,6B,7A,7B | - | nicht verfügbar | - | - |

Zusammengefasst nach Tiva TM4C1294 Microcontroller Manual S.956/958 ohne Gewähr

# Alternative Function Select für GPIO-Ports

## GPIO_Portx_AFSEL_R x=A,..,H,J,..,P,Q

GPIO Alternate Function Select (GPIOAFSEL)



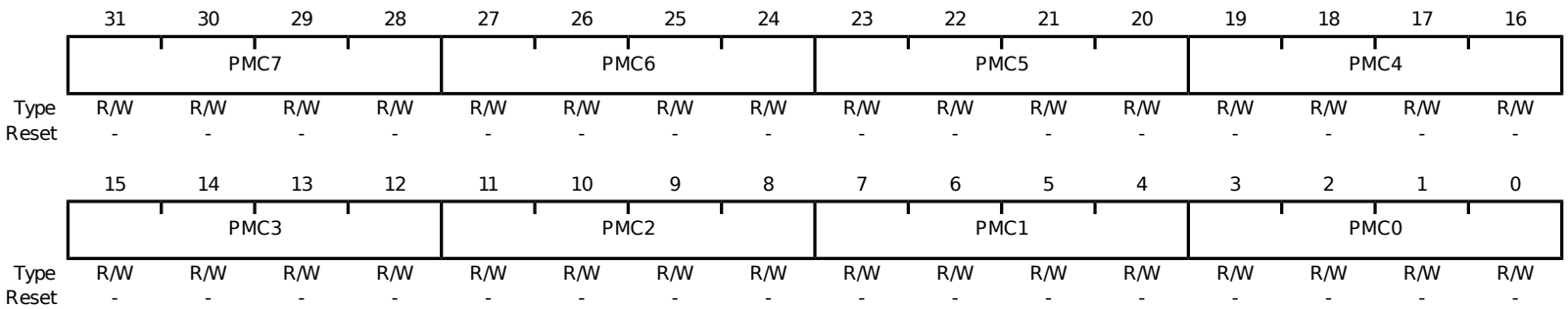

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7:0 | AFSEL | R/W | - | GPIO Alternate Function Select |

| Value | Description |
|---|---|
| | The associated pin functions as a GPIO and is controlled by the GPIO registers.0 |
| | The associated pin functions as a peripheral signal and is controlled by the alternate hardware function. The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 8-1 on page 398.1 |

# Port Control Register steuert die Auswahlmatrix der alternativen Funktion der Pins

`GPIO_Portx_CTL_R x=x=A,..,H,J,..,P,Q`

GPIO Port Control (GPIOPCTL)

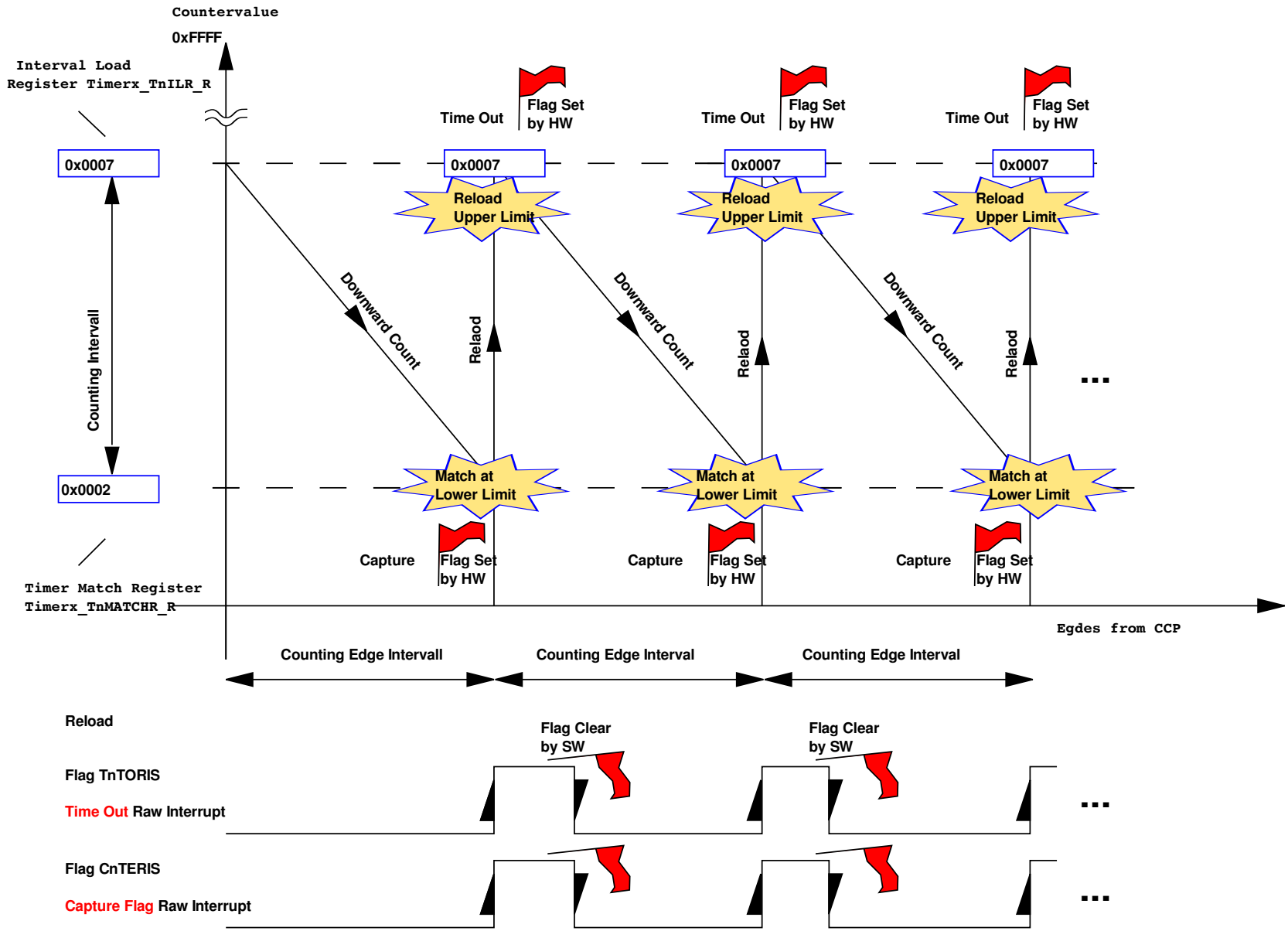



| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:28 | PMC7 | R/W | - | Port Mux Control 7<br>This field controls the configuration for GPIO pin 7. |
| ... | | | | |
| 3:0 | PMC0 | R/W | - | Port Mux Control 0<br>This field controls the configuration for GPIO pin 0. |

Siehe S. 1808ff in Texas Instruments Tiva TM4C1294NCPDT Microcontroller Datasheet, June 18, 2014

# Lower Limit Match in Edge Counting

## Timerx_TnTAMTCHR_R x=A,..,H,J,..,P,Q



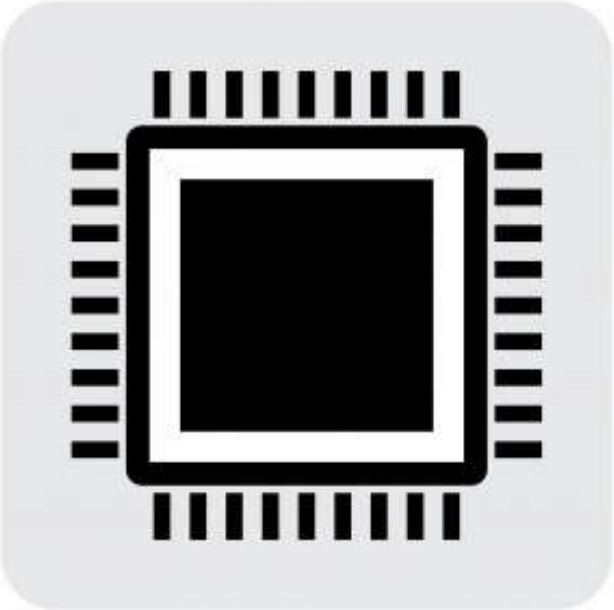
# GPTM Input Edge Count Mode 1 of 2

```
1  // ARM Cortex M4 GTPM Input Capture Edge Counting Mode
2  // Count the egde made with a manual switch and toogle LED's after a number
3  // of edges defined by (upperstart-value - lower match value) <====
4  // remark: use a bounce-free switch on Port M Pin 4 ... For simplification
5  // in this example LEDs toogles the output after 5 pushes of the switch
6  #include <stdint.h>
7  #include "inc/tm4c1294ncpdt.h"
8
9  void main(void)
10 {
11     int wt=0;            // aux. variable for very short time wait
12     // GPIO Port initialize
13     SYSCTL_RCGCGPIO_R = 0x00000A00;  wt++;// clock enable Port M + Port K  wait for stable
         clock
14     SYSCTL_RCGCTIMER_R = 0x00000010; wt++; //clock enable timer 4
15
16     // Port K complete for LED Output - Port M4 for switch in put on CCP-Pin
17     GPIO_PORTK_DEN_R  = 0xFF;        // digital I/O pins PK7 to PK0 for LED
18     GPIO_PORTK_DIR_R  = 0xFF;        // define output direction I/O pins PK7 to PK0
19     GPIO_PORTK_DATA_R = 0x0F;        // inital output PortK: 4 LED on , 4 LED off
20     GPIO_PORTM_DEN_R  = 0x10;        // digital I/O pins PM4
21     GPIO_PORTM_DIR_R  = 0x00;        // define input direction I/O pins PM4
22     GPIO_PORTM_AFSEL_R= 0x10;        // CCP enable alternative function on PM4
23     GPIO_PORTM_PCTL_R = 0x00030000; // CCP => connect T4CCP0 to PM4 !
24                                      // refer to Table 26-5 on page 1809
25                                      // TM4C1294 Microcontroller Datasheet
26     // GPTM timer 4 initialization
27     TIMER4_CTL_R &=  ~0x00000001;    // disable timer4A during setup
```

# GPTM Input Edge Count Mode 2 of 2

```
27      TIMER4_CTL_R &=  ~0x00000001;   // disable timer4A during setup
28      TIMER4_CFG_R  =   0x00000004 ;  // configure for 16-bit timer mode
29      TIMER4_TAMR_R =   0x00000003;   // for edge counting mode
30      TIMER4_CTL_R &=  ~(0xC);        // configure for rising edge event
31      TIMER4_TAILR_R=   0x00000007;   // upper start value <====
32      TIMER4_TAMATCHR_R=0x00000002;   // lower match value <====
33      TIMER4_ICR_R |=   0x00000002;   // clear flag of timer 4A
34      TIMER4_CTL_R |=   0x00000001;   // enable timer 4A
35
36      while(1) {
37              while((TIMER4_RIS_R & 0x02)==0x00); // wait and poll timer 4A capture flag
38              GPIO_PORTK_DATA_R  ^=0xFF;          // toggle Port K all Pin's output
39              TIMER4_ICR_R |= 0x02;               // clear flag of timer 4A
40              TIMER4_CTL_R    |=  0x00000001;     // re-enable timer 4A
41      }
42  }
```

Hochschule für Angewandte
Wissenschaften Hamburg
*Hamburg University of Applied Sciences*



# General Purpose Timers – further examples

**Prof. Dr. Paweł Buczek**

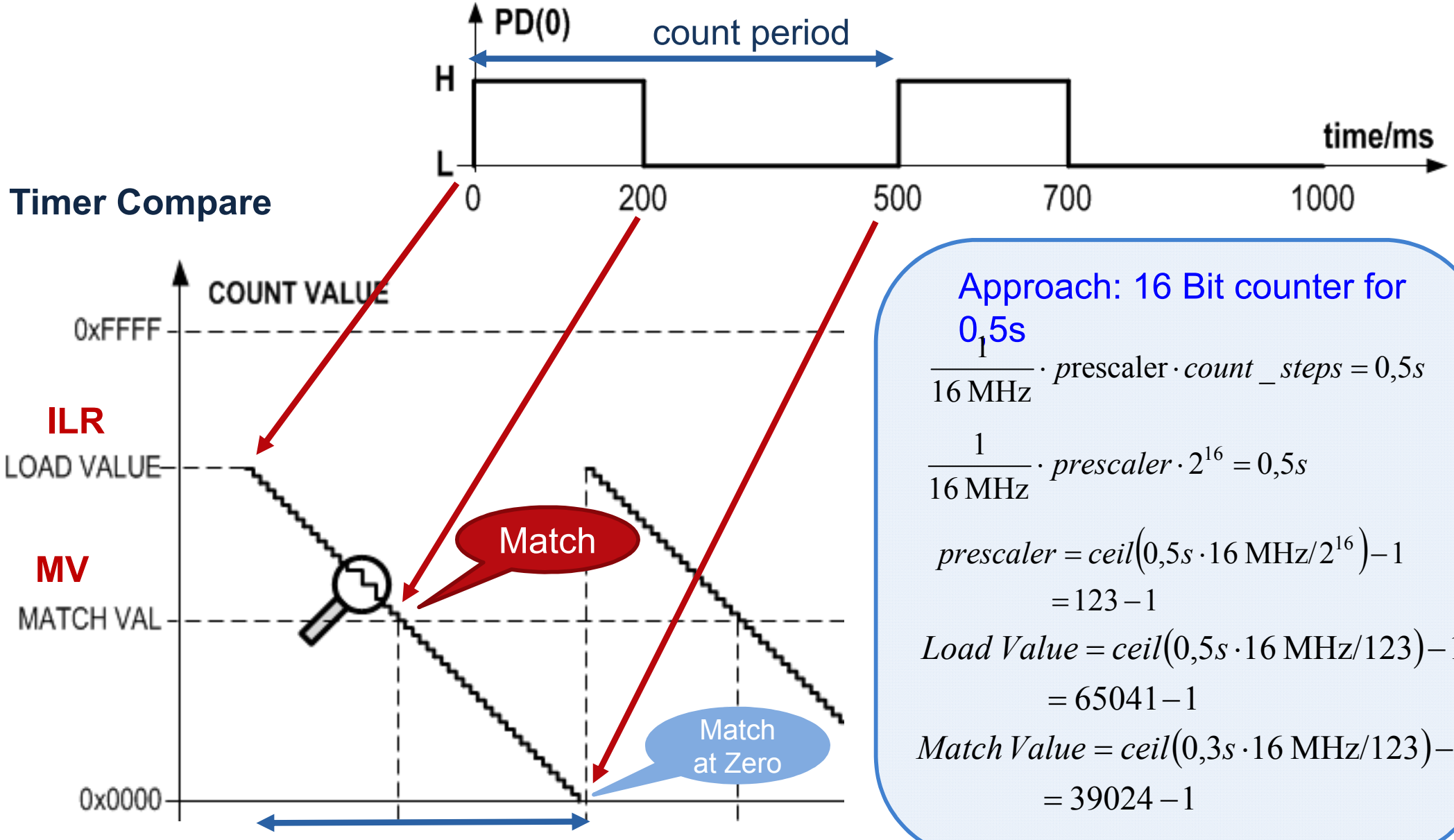
Based on the lecture slides of Prof. Dr.-Ing. Lutz Leutelt
Logo courtesy of Wikipedia

# Cortex-M4

- Introduction to timers
- Drawbacks of SW timers
- General Timer Compare Principle
- 16bit periodic timer compare mode (TM4C1294)

# Programmieraufgabe - Approach

- Generate a periodic signal with Timer 0A




Approach: 16 Bit counter for 0,5s

$$\frac{1}{16\,\text{MHz}} \cdot \text{prescaler} \cdot count\_steps = 0{,}5s$$

$$\frac{1}{16\,\text{MHz}} \cdot prescaler \cdot 2^{16} = 0{,}5s$$

$$prescaler = ceil\left(0{,}5s \cdot 16\,\text{MHz}/2^{16}\right) - 1$$
$$= 123 - 1$$

$$Load\ Value = ceil\left(0{,}5s \cdot 16\,\text{MHz}/123\right) - 1$$
$$= 65041 - 1$$

$$Match\ Value = ceil\left(0{,}3s \cdot 16\,\text{MHz}/123\right) - 1$$
$$= 39024 - 1$$

# Programming task – C code (configuration)

```
#include "inc/tm4c1294ncpt.h"
void main(void) {

    // configure port D
    SYSCTL_RCGCGPIO_R = 0x00000008; // clock port D
    while(!(SYSCTL_PRGPIO_R & 0x00000008)); // wait for port D activation
    GPIO_PORTD_AHB_DEN_R |= 0x01;    // PD(0) enable
    GPIO_PORTD_AHB_DIR_R |= 0x01;    // PD(0) output

    // configure Timer 0
    SYSCTL_RCGCTIMER_R |= (1<<0);   // system clock  auf Timer 0
    while(!(SYSCTL_PRTIMER_R & 0x01)); // wait for Timer 0 activation
    TIMER0_CTL_R &= ~0x0001;        // disable Timer 0
    TIMER0_CFG_R  = 0x04;           // 2 x 16-bit mode
    TIMER0_TAMR_R   = 0x22;         // periodic mode + match enable
    TIMER0_TAPR_R = 123-1;          // prescaler PR= ceil(16M/2^16*0.5)-1
    TIMER0_TAILR_R= 65041-1;        // ILR= ceil(16M/123*0.5)-1
    TIMER0_TAMATCHR_R = 39024-1;    // MV= ceil(16M/123*0.3)-1
    TIMER0_CTL_R |= 0x0001;         // enable Timer 0
```

# Programming task – C Code (Pulse generation)

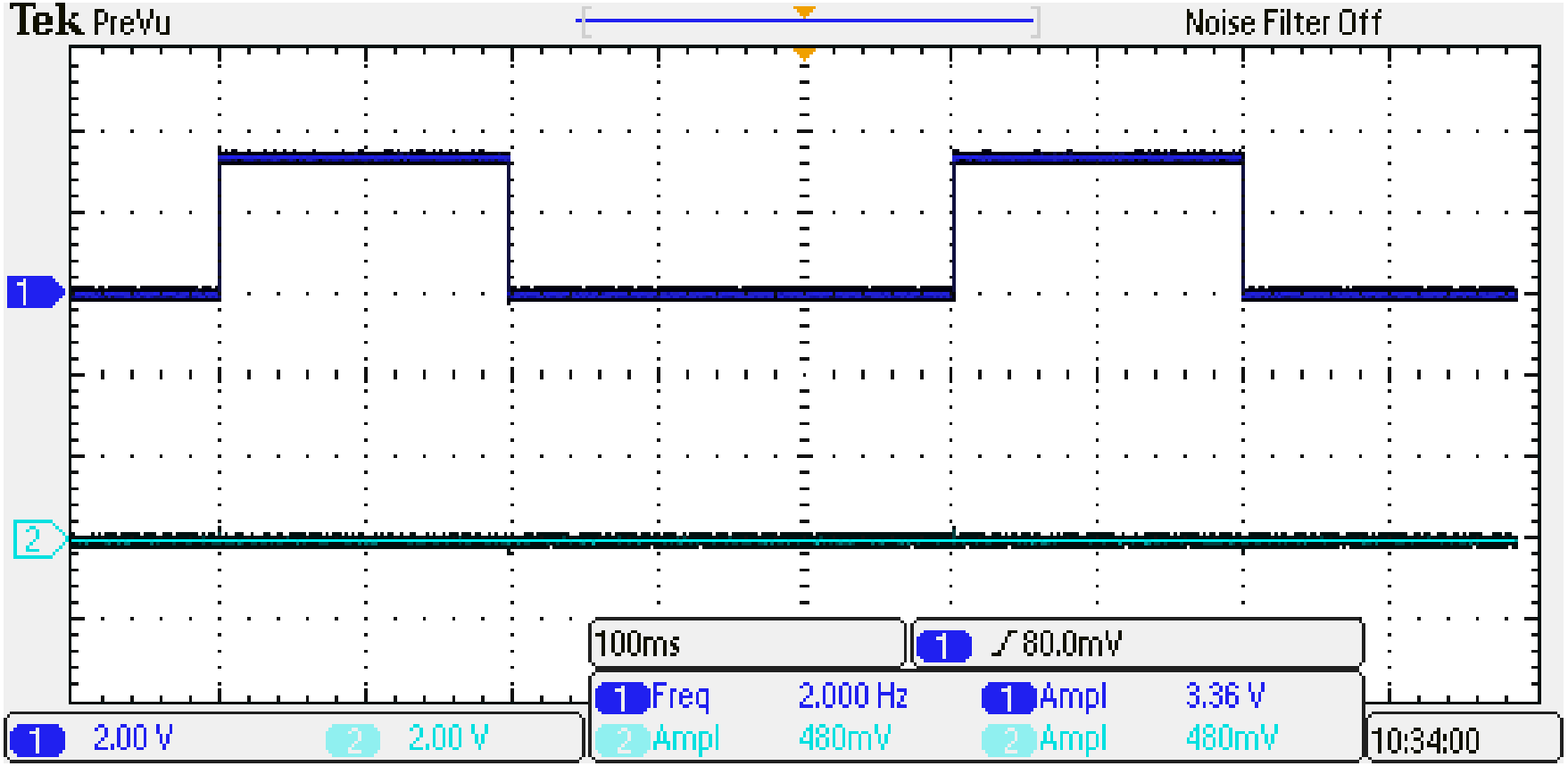
- Add comments to the C code:

```
while(1){                                          // add comments here:
        GPIO_PORTD_AHB_DATA_R |= 0x01;             PD(0) high, initially or when "time-out" reached
        while((TIMER0_RIS_R & (1<<4))==0)          wait for "match value reached" flag
            ;
        TIMER0_ICR_R |= (1<<4);                    clear "match value reached" flag
        GPIO_PORTD_AHB_DATA_R &= ~0x01;            PD(0) low, when match value reached
        while((TIMER0_RIS_R & (1<<0))==0)          wait for "time-out" flag
            ;
        TIMER0_ICR_R |= (1<<0);                    clear "time-out" flag
    }
}
```

- Disadvantages of this approach? Improvement?

Microcontroller is busy with generating a simple signal. non-blocking behavior would be better, even better would be a generation by an autonomous HW block that needs no SW interaction after configuration (-> PWM mode).

# Progamming task – Test




- works…

# Assignment

discuss with neighbor, 12 min. total time

Change the source code to generate the periodic signal 10ms high-5ms low
@ $f_{CPU}$ = 80MHz

```
TIMER0_TAPR_R =
TIMER0_TAILR_R =
TIMER0_TAMATCHR_R =
```

PR = ceil(80M/$2^{16}$ * 0.015)-1 = 19 - 1

ILR = ceil(80M/19 * 0.015)-1 = 63158-1

MV = ceil(80M/19 * 0.005)-1 = 21053-1

# Register to be configured for compare mode (summary):

- configure **port(s)**
- configure **Timer**
  - `SYSCTL_RCGCTIMER_R // activate timer clock`
  - `while(!(SYSCTL_PRTIMER_R & (1<< x))); // ready?`
  - `TIMERx_CTL_R // stop timer`
  - `TIMERx_CFG_R // 2 x 16-bit or 32-bit`
  - `TIMERx_TnMR_R // periodic, one-shot, capture mode`
  - `TIMERx_TnPR_R // prescaler`
  - `TIMERx_TnILR_R // set Interval Load Value`
  - `TIMERx_TnMATCHR_R // set Match Value`
  - `TIMERx_CTL_R // start Timer`
- check **flags** **Match** **Time out**

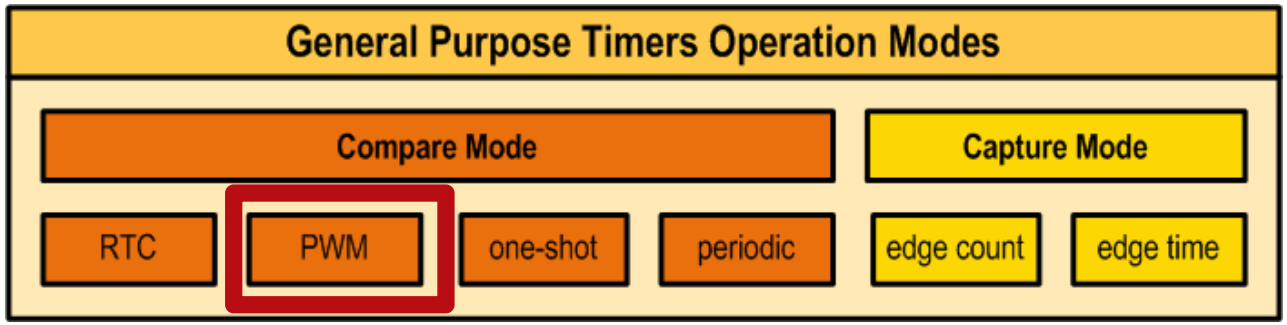
```
while((TIMER0_RIS_R & (1<<4))==0)  or (1<<0))==0)
```

# General Purpose Timers

## Part 2

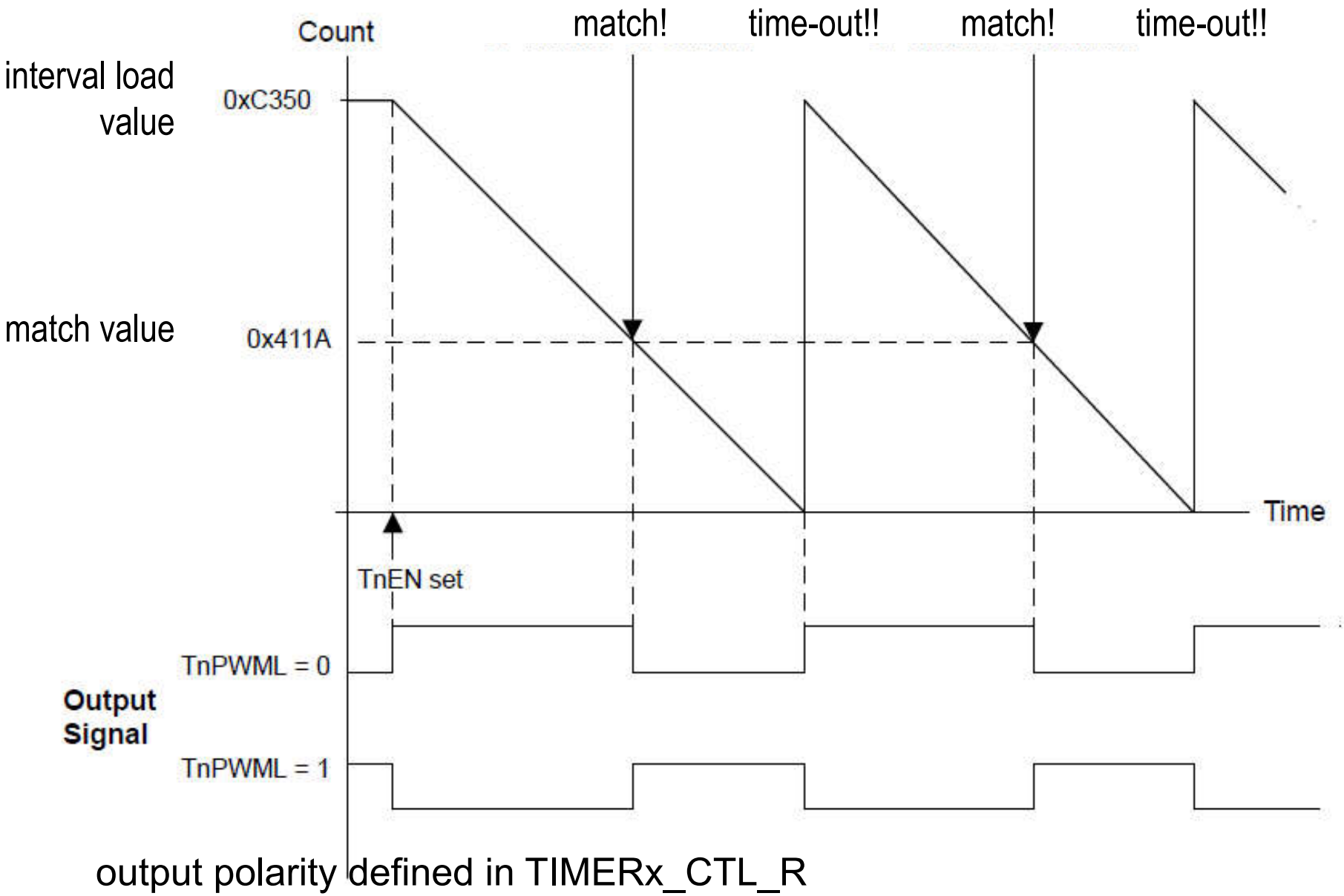
- Timer modes
- Connecting timers to pins
- Timer Capture Mode
- Pulse Width Modulation

# Timer module in PWM mode




- **PWM mode** is the same as a periodic 16 bit, downwards timer compare mode, but with a GPIO pin connected to it
- can be used to create **periodic pulse signals with varying duty cycle**

# Timer module in PWM mode



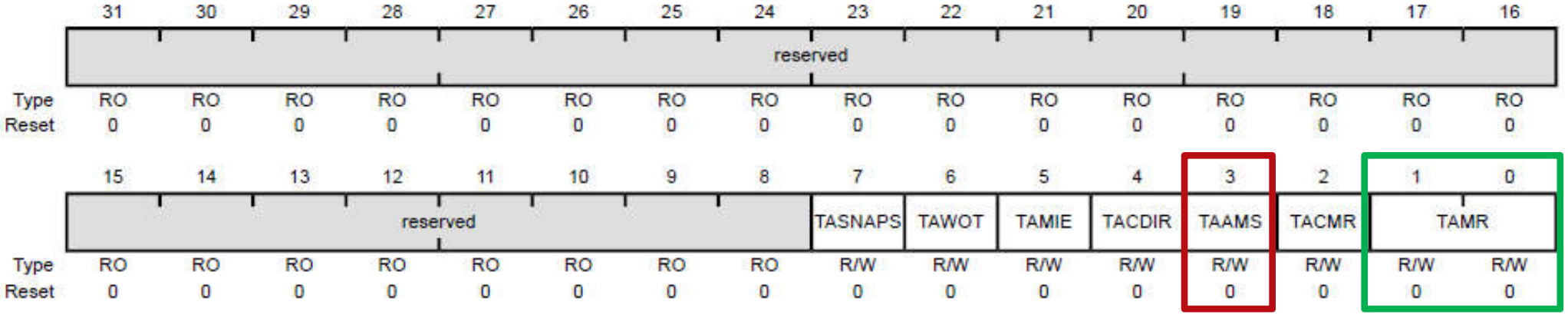
# Timer module in PWM mode

- in PWM mode the counter works only in following configuration: 16bit, periodic, count down mode
- in register TIMERx_TAMR_R/TIMERx_TBMR_R the TAAMS/TBAMS bit must be set to enable PWM mode
- in register TIMERx_CTL_R the TAPWML/TBPWML bit determines whether toggling of output starts with H- or L-level (see figure on previous page)
- at match and time-out the output value toggles
- prescaler is not supported

# GPTM Timer A/B Mode

*(PWM mode)*

**TIMERx_TAMR_R**
**TIMERx_TBMR_R**




TAAMS R/W 0 GPTM Timer A Alternate Mode Select

The TAAMS values are defined as follows:

| Value | Description |
|---|---|
| 0 | Capture mode is enabled. |
| 1 | PWM mode is enabled. |

**Note:** To enable PWM mode, you must also clear the TACMR bit and configure the TAMR field to 0x1 or 0x2. !!!

Compare mode

# GPTM Timer A/B Mode

*(PWM mode)*

**TIMERx_TAMR_R**
**TIMERx_TBMR_R**

| | | | |
|---|---|---|---|
| TACMR | R/W | 0 | GPTM Timer A Capture Mode |

The TACMR values are defined as follows:

Compare mode

| Value | Description |
|---|---|
| 0 | Edge-Count mode |
| 1 | Edge-Time mode |

!!!

| | | | |
|---|---|---|---|
| TAMR | R/W | 0x0 | GPTM Timer A Mode |

The TAMR values are defined as follows:

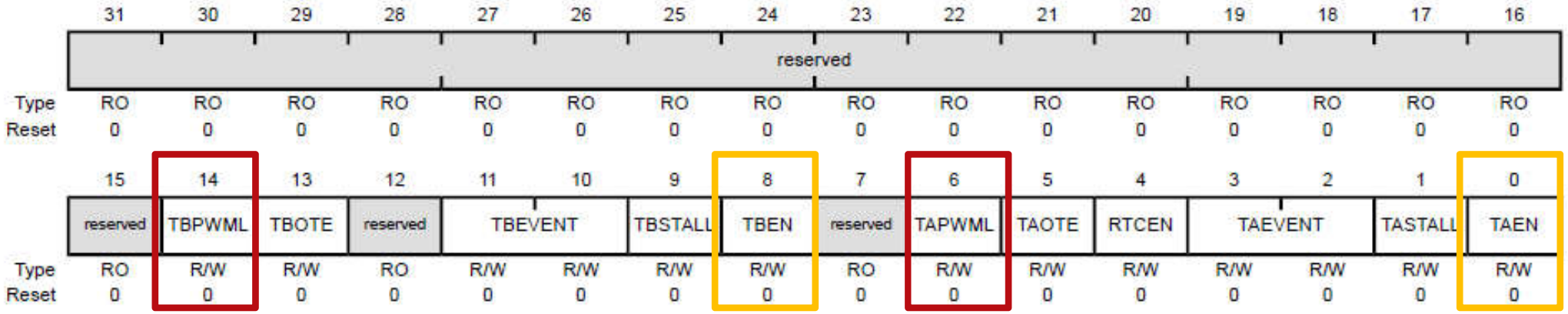
Compare mode

| Value | Description |
|---|---|
| 0x0 | Reserved |
| 0x1 | One-Shot Timer mode |
| 0x2 | Periodic Timer mode |
| 0x3 | Capture mode |

The Timer mode is based on the timer configuration defined by bits 2:0 in the **GPTMCFG** register.

# GPTM Control Register

**TIMERx_CTL_R**




| | | | | |
|---|---|---|---|---|
| 8 | TBEN | R/W | 0 | GPTM Timer B Enable |

The TBEN values are defined as follows:

**TIMER START** **Timer B**

| Value | Description |
|---|---|
| 0 | Timer B is disabled. |
| 1 | Timer B is enabled and begins counting or the capture logic is enabled based on the **GPTMCFG** register. |

| | | | | |
|---|---|---|---|---|
| 0 | TAEN | R/W | 0 | GPTM Timer A Enable |

The TAEN values are defined as follows:

**TIMER START** **Timer A**

| Value | Description |
|---|---|
| 0 | Timer A is disabled. |
| 1 | Timer A is enabled and begins counting or the capture logic is enabled based on the **GPTMCFG** register. |

# GPTM Control Register

**TIMERx_CTL_R**

| | | | | |
|---|---|---|---|---|
| 14 | TBPWML | R/W | 0 | GPTM Timer B PWM Output Level<br>The TBPWML values are defined as follows: |

PWM mode only

**Timer B**

| Value | Description |
|---|---|
| 0 | Output is unaffected. |
| 1 | Output is inverted. |

| | | | | |
|---|---|---|---|---|
| 6 | TAPWML | R/W | 0 | GPTM Timer A PWM Output Level<br>The TAPWML values are defined as follows: |

PWM mode only

**Timer A**

| Value | Description |
|---|---|
| 0 | Output is unaffected. |
| 1 | Output is inverted. |

# Programming example

```
#include "inc/tm4c1294ncpt.h"
#include <stdio.h>

void main(void) {

    // configure port D
    SYSCTL_RCGCGPIO_R |= (1<<3);          // clock port D
    while(!(SYSCTL_PRGPIO_R & (1<<3))); // wait for port D
activation

    GPIO_PORTD_AHB_DEN_R   |= (1<<1)|(1<<0);    // PD(1:0) enable
    GPIO_PORTD_AHB_DIR_R   |= (1<<1)|(1<<0);    // PD(1:0) output
    GPIO_PORTD_AHB_DATA_R  &= ~(1<<0);          // clear PD(0)
    GPIO_PORTD_AHB_AFSEL_R |= (1<<0);       // PD(0) alternate
function
    GPIO_PORTD_AHB_PCTL_R   = 0x00000003;  // PD(0) connected to
Timer0A
```

# Programming example

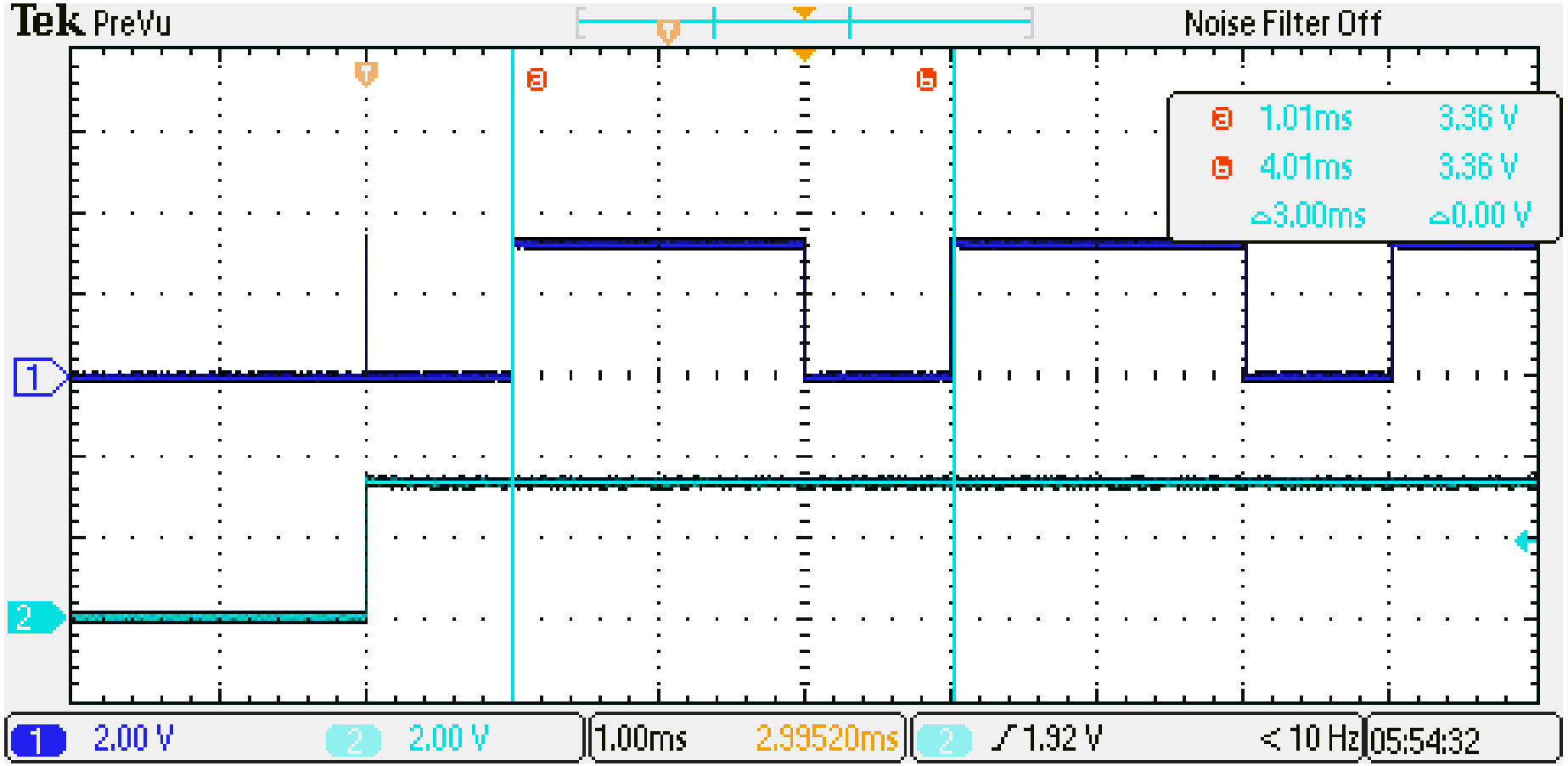
```
    // configure Timer 0
   SYSCTL_RCGCTIMER_R |= (1<<0);     // timer 0
   while(!(SYSCTL_PRTIMER_R & (1<<0))); // wait for timer 0 activation
   TIMER0_CTL_R &= ~0x0001;          // disable Timer 0
   TIMER0_CFG_R  = 0x04;             // 2 x 16-bit mode

   // compare mode, down, pwm: TAAMS=1, periodic: TAMR=0x2
   TIMER0_TAMR_R   = 0x000A;
   TIMER0_CTL_R     |= (1<<6);          // TAPWML=1 (inverting)
   TIMER0_TAILR_R    = 48000-1;         // ILR = 0.003*16e6-1;
   TIMER0_TAMATCHR_R = 32000-1;         // MATCH = 0.001*16e6-1;
   GPIO_PORTD_AHB_DATA_R |=(1<<1);      // set PD(1) - Startsignal
                                        //start on 2nd channel
   TIMER0_CTL_R    |= 0x0001;           // enable Timer 0A

   while(1)                             // empty while loop
      ;
}
```

# Programming example

- generated PWM output (and timer start signal on channel 2):



# Questions (Self-Test) on Part 2

(approx. 30 min for all 5 questions)

# Questions

- Why is it necessary to configure GPIO port registers when using timers in capture and PWM mode?

Both modes require that the timer is directly connected to GPIO pins. The GPIO pin has to be configured for the correct alternate function.

- Timer 2B is to be connected to Port G. Configure port G (give C code).

Timer 2 B can be connected to PG(7) with multiplex value: 0x8.

```
SYSCTL_RCGC2_R |= (1<<6);
GPIO_PORTG_DEN_R |= 0x80;
GPIO_PORTG_AFSEL_R |= 0x80;
GPIO_PORTF_PCTL_R  |= 0x80000000;
```

# Questions

- A PWM signal with 12 kHz with 30% duty cycle is generated @ f_CPU=20 MHz by a 16 bit timer. Give the interval load value and the match value. The timer is in non-inverting mode. What is the actual frequency of the PWM signal?

high phase = 0.3 *(1/12kHz) = 2.5e-5, period = 1/12 kHz = 8.333e-5

ILV = 1/12kHz*20MHz-1 = 1667-1, MATCH = 0.7*(1/12kHz)*20MHz-1=1167-1

f'_PWM= 20MHz/1667= 11.9976 kHz

- Let a 16 bit timer be in edge-time mode. The timer is started (interval load value=0xFFFF) with a rising edge of an input signal. It stops with a falling edge of the input signal (captured timer value is 0xC020). What is the time between falling and rising edge (f_CPU=50 MHz)?

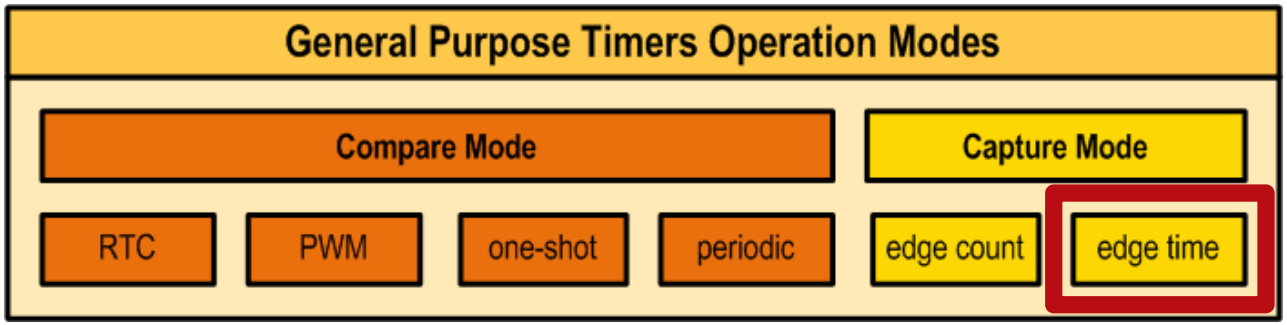
(0xFFFF-0xC020) / 50MHz = 327.02µs

# General Purpose Timers

## Part 2

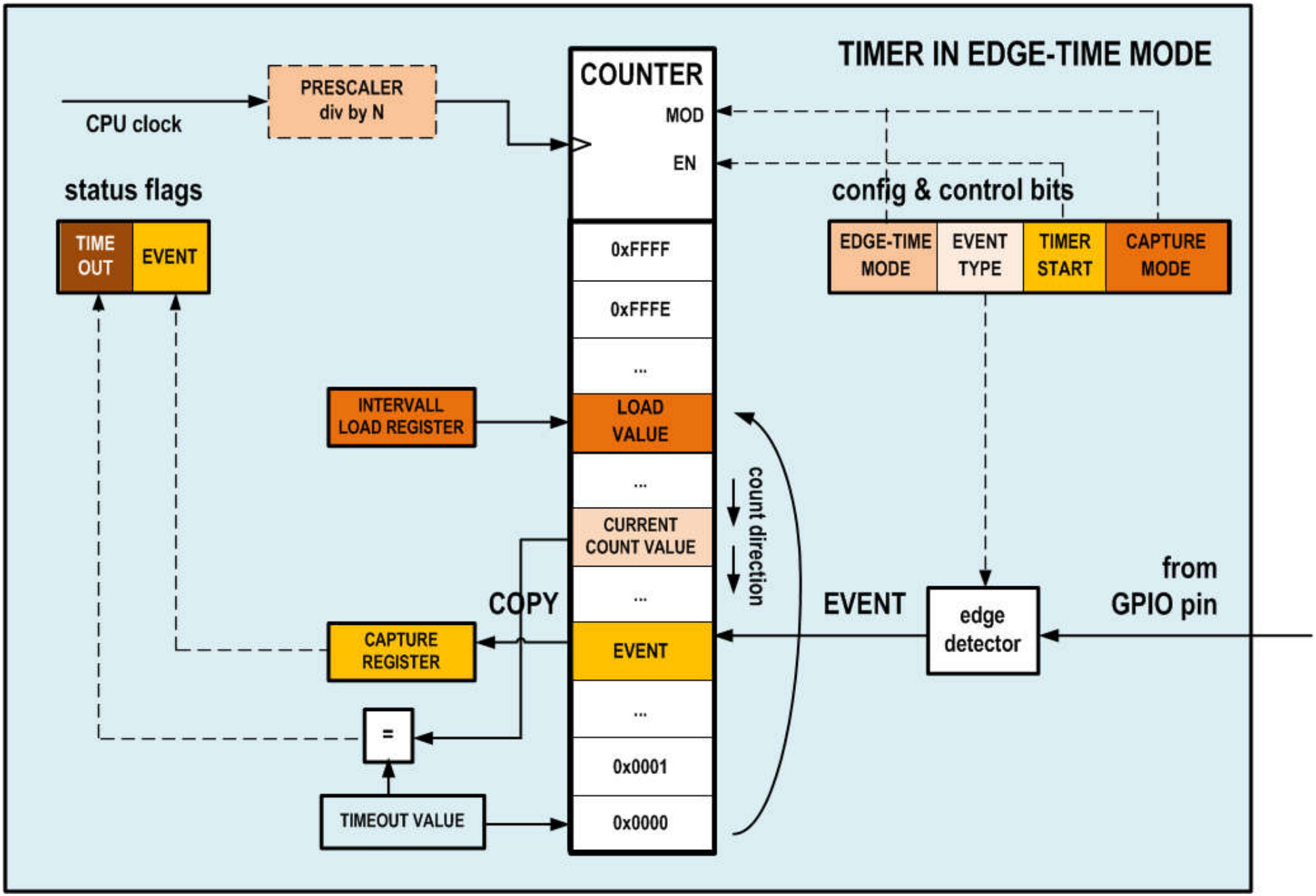
- Timer modes
- Connecting timers to pins
- Timer Capture Mode
- Pulse Width Modulation

# Timer module in edge-time mode

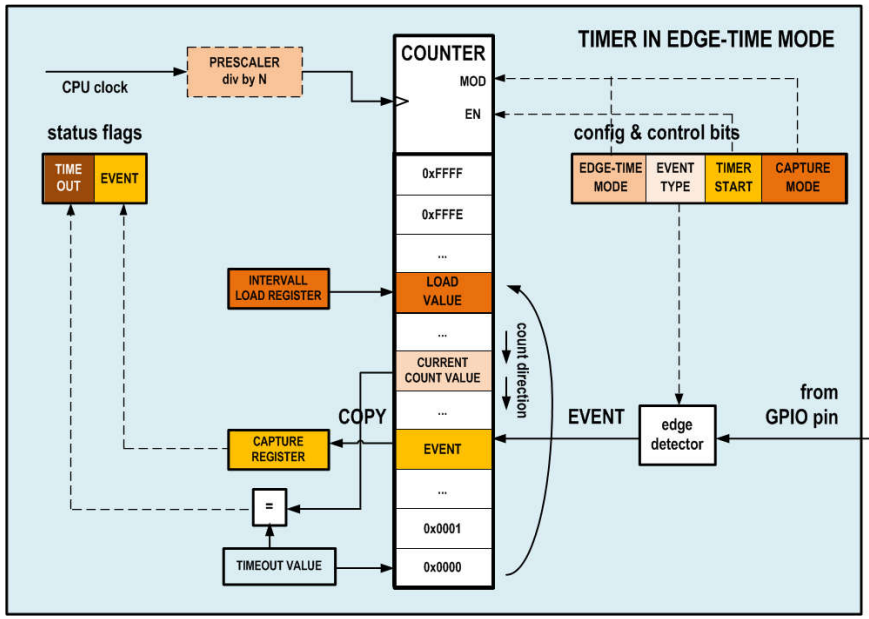



- **Edge-time mode** measures the time between events at a port pin connected to timer module (event=rising/falling/both edge(s) of an input signal)
- requires 16 bit, downwards timer mode
- configuration similar to one-shot/periodic modes, most differences in TIMERx_TAMR_R/ TIMERx_TBMR_R registers
- prescaler not supported in edge-time mode!

# Timer module in edge-time mode

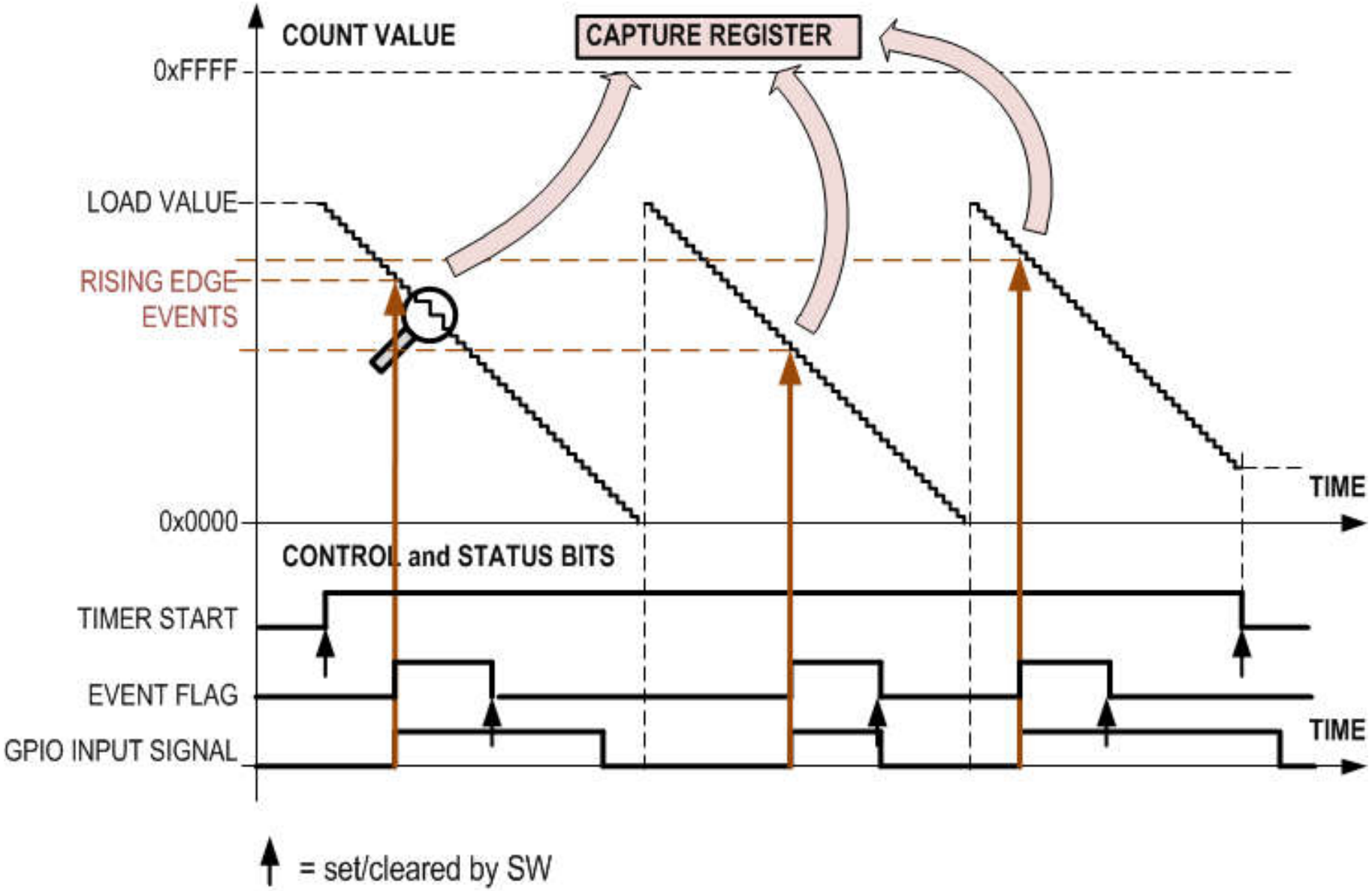


# Timer module in edge-time mode



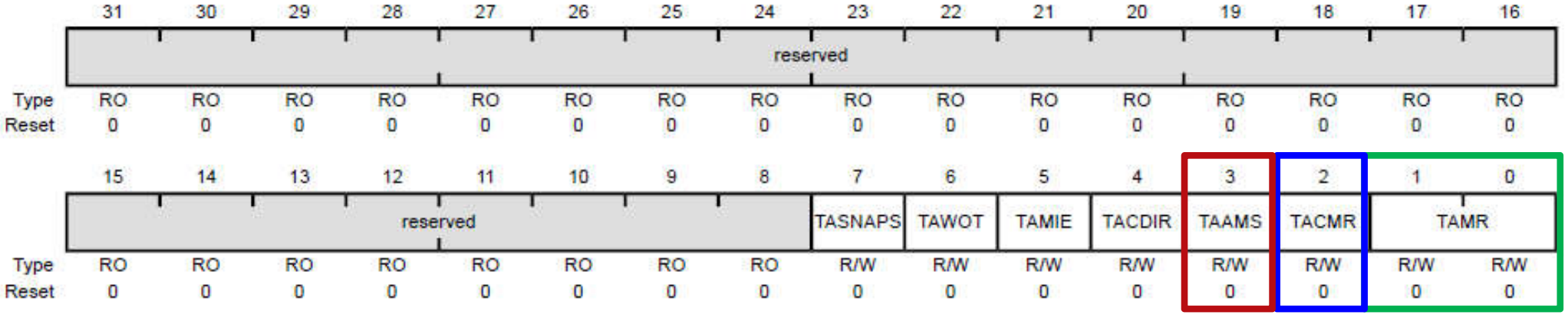

- Timer starts at *interval load value* and counts down. When an event (falling/rising/both edges) at the GPIO input pin occurs, the *current count value* is copied into *capture register* and *event status flag* is set.
- Timer continues counting, rolls over at 0x0000 and restarts with *interval load value*. The next event overwrites the previous value in capture register
- prescaler and 32bit mode are not supported in TM4C1294

# Timer module in edge-time mode (2)



# GPTM Timer A/B Mode

**TIMERx_TAMR_R**
**TIMERx_TBMR_R**




TAAMS R/W 0 GPTM Timer A Alternate Mode Select

The TAAMS values are defined as follows:

| Value | Description |
|---|---|
| 0 | Capture mode is enabled. |
| 1 | PWM mode is enabled. |

**Note:** To enable PWM mode, you must also clear the TACMR bit and configure the TAMR field to 0x1 or 0x2.

**CAPTURE MODE**

**UP DOWN**

# GPTM Timer A/B Mode

**TIMERx_TAMR_R**
**TIMERx_TBMR_R**

| | | | |
|---|---|---|---|
| TACMR | R/W | 0 | GPTM Timer A Capture Mode |

The TACMR values are defined as follows:

**EDGE-TIME MODE**

| Value | Description |
|---|---|
| 0 | Edge-Count mode |
| 1 | Edge-Time mode |

| | | | |
|---|---|---|---|
| TAMR | R/W | 0x0 | GPTM Timer A Mode |

The TAMR values are defined as follows:

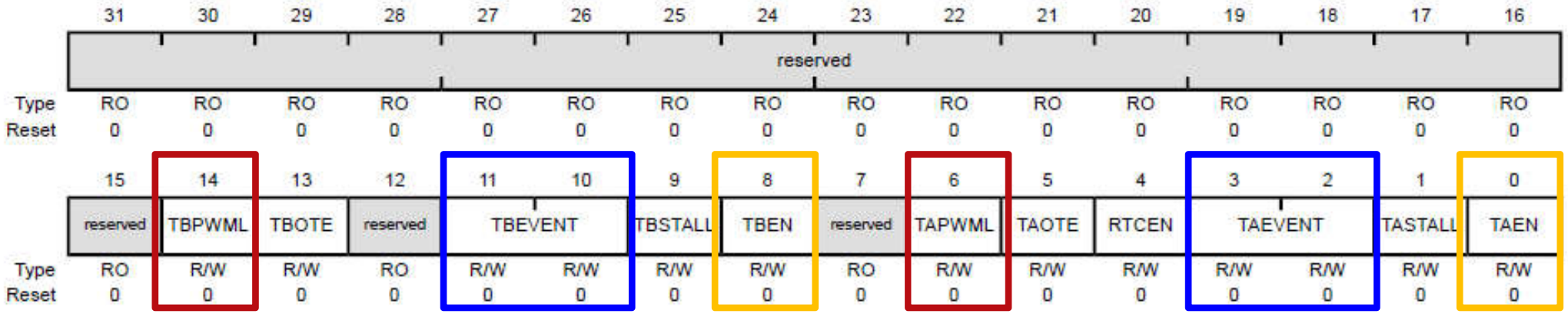
**CAPTURE MODE**

| Value | Description |
|---|---|
| 0x0 | Reserved |
| 0x1 | One-Shot Timer mode |
| 0x2 | Periodic Timer mode |
| 0x3 | Capture mode |

The Timer mode is based on the timer configuration defined by bits 2:0 in the **GPTMCFG** register.

# GPTM Control Register

**TIMERx_CTL_R**




| | | | | |
|---|---|---|---|---|
| 8 | TBEN | R/W | 0 | GPTM Timer B Enable |

The TBEN values are defined as follows:

**TIMER START**

| Value | Description |
|---|---|
| 0 | Timer B is disabled. |
| 1 | Timer B is enabled and begins counting or the capture logic is enabled based on the **GPTMCFG** register. |

| | | | | |
|---|---|---|---|---|
| 0 | TAEN | R/W | 0 | GPTM Timer A Enable |

The TAEN values are defined as follows:

**TIMER START**

| Value | Description |
|---|---|
| 0 | Timer A is disabled. |
| 1 | Timer A is enabled and begins counting or the capture logic is enabled based on the **GPTMCFG** register. |

# GPTM Control Register

**TIMERx_CTL_R**

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 11:10 | TBEVENT | R/W | 0x0 | GPTM Timer B Event Mode<br>The TBEVENT values are defined as follows: |

EVENT TYPE — **Timer B**

| Value | Description |
|---|---|
| 0x0 | Positive edge |
| 0x1 | Negative edge |
| 0x2 | Reserved |
| 0x3 | Both edges |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3:2 | TAEVENT | R/W | 0x0 | GPTM Timer A Event Mode<br>The TAEVENT values are defined as follows: |

EVENT TYPE — **Timer A**

| Value | Description |
|---|---|
| 0x0 | Positive edge |
| 0x1 | Negative edge |
| 0x2 | Reserved |
| 0x3 | Both edges |

# GPTM Control Register

**TIMERx_CTL_R**

| | | | | |
|---|---|---|---|---|
| 14 | TBPWML | R/W | 0 | GPTM Timer B PWM Output Level<br>The TBPWML values are defined as follows: |

PWM mode only

| Value | Description |
|---|---|
| 0 | Output is unaffected. |
| 1 | Output is inverted. |

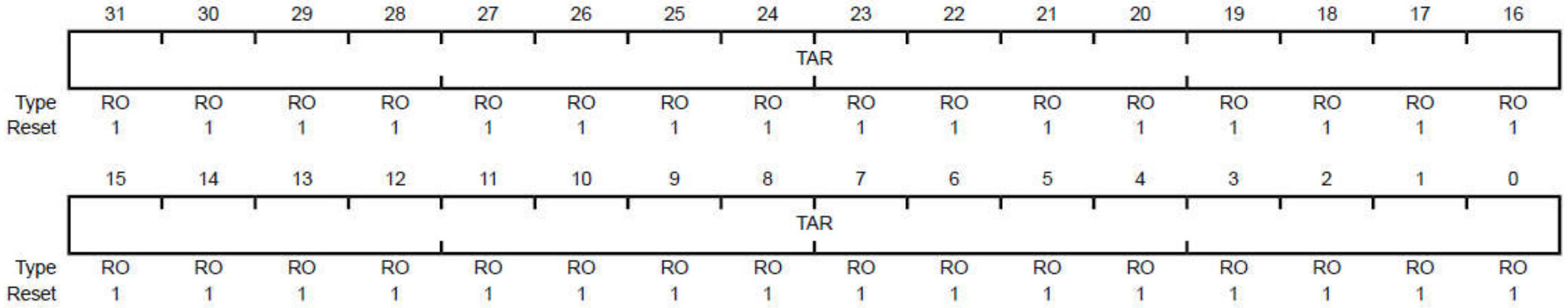
| | | | | |
|---|---|---|---|---|
| 6 | TAPWML | R/W | 0 | GPTM Timer A PWM Output Level<br>The TAPWML values are defined as follows: |

PWM mode only

| Value | Description |
|---|---|
| 0 | Output is unaffected. |
| 1 | Output is inverted. |

# GPTM Timer A/B

**TIMERx_TAR_R**
**TIMERx_TBR_R**



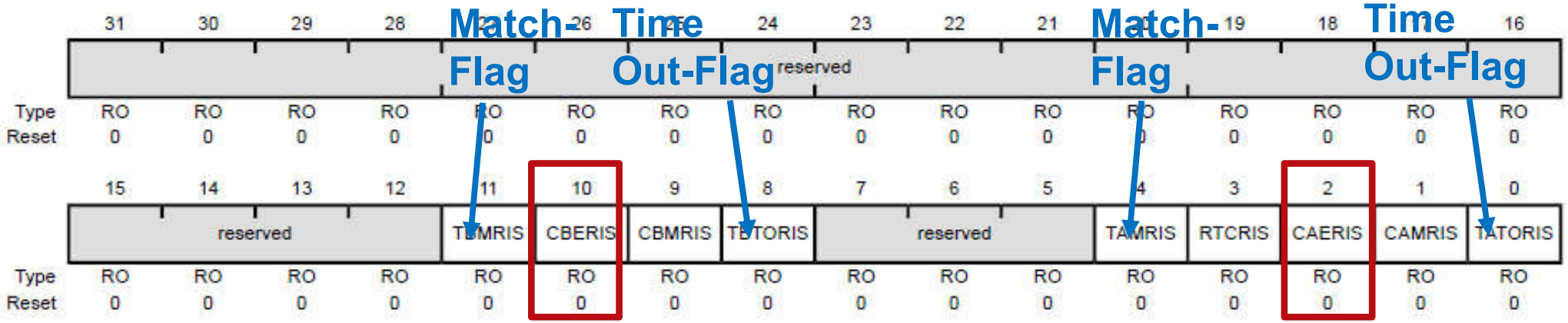

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | TAR | RO | 0xFFFF.FFFF | GPTM Timer A Register |

A read returns the current value of the **GPTM Timer A Count Register**, in all cases except for Input Edge Count and Time modes. In the Input Edge Count mode, this register contains the number of edges that have occurred. In the Input Edge Time mode, this register contains the time at which the last edge event took place.

**CAPTURE REGISTER**

# GPTM Raw Interrupt Status

**TIMERx_RIS_R**




| | | | | |
|---|---|---|---|---|
| 10 | CBERIS | RO | 0 | GPTM Timer B Capture Mode Event Raw Interrupt |

**Timer B**

| Value | Description |
|---|---|
| 1 | A capture mode event has occurred for Timer B. This interrupt asserts when the subtimer is configured in Input Edge-Time mode. |
| 0 | The capture mode event for Timer B has not occurred. |

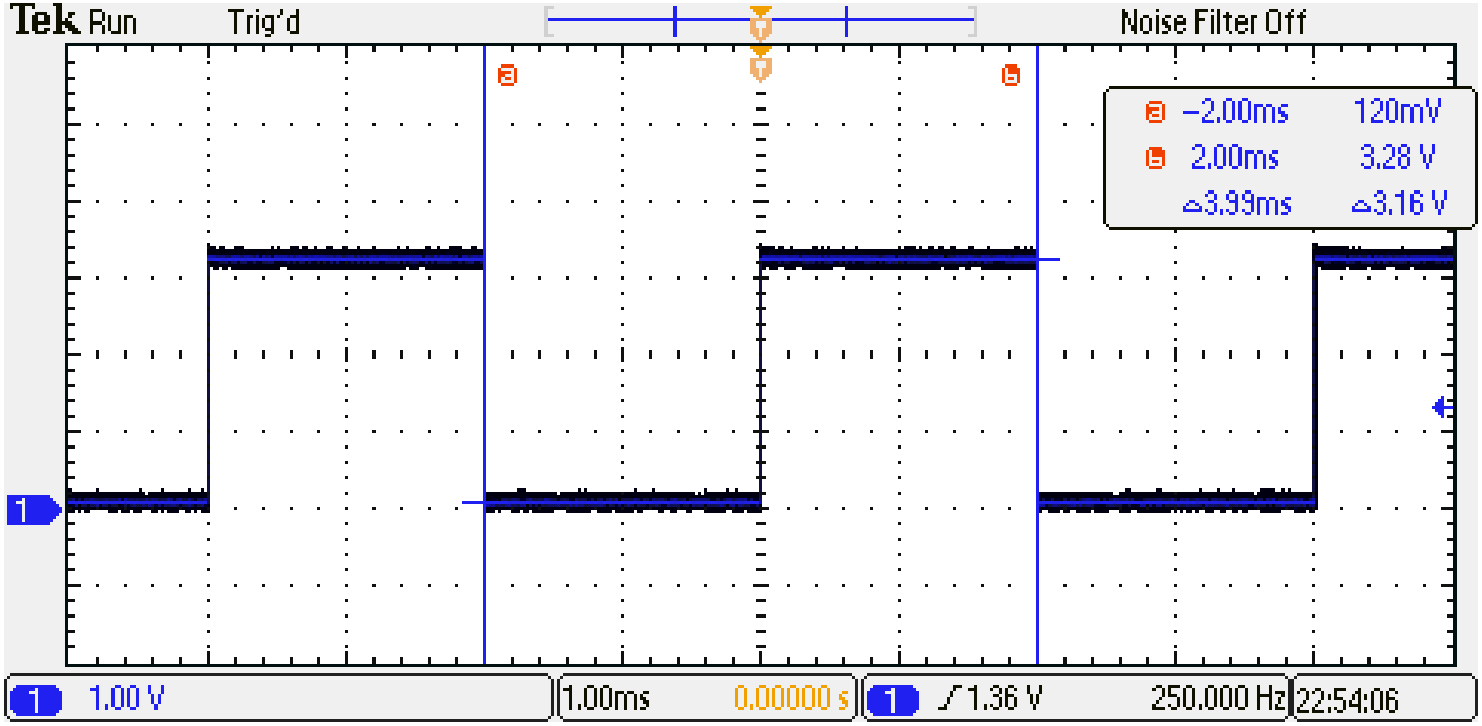
This bit is cleared by writing a 1 to the CBECINT bit in the **GPTMICR** register.

| | | | | |
|---|---|---|---|---|
| 2 | CAERIS | RO | 0 | GPTM Timer A Capture Mode Event Raw Interrupt |

**Timer A**

# Example program (input capture, edge-time)

- measure time between **falling edges** of input signal at **port PL(4)** with **Timer0A**
- calculate the time difference in µs between falling edges



# Example program (input capture, edge-time)

```
#include "inc/tm4c1294ncpt.h"
#include <stdint.h>

#define ASIZE 10

void main(void) {
    unsigned long ulVal;

    // configure port L
    SYSCTL_RCGCGPIO_R |= (1<<10);  // clock port L
    while(!(SYSCTL_PRGPIO_R & (1<<10))); // wait for port L
clock
    GPIO_PORTL_DEN_R   |= (1<<4);      // PL(4) enable

    GPIO_PORTL_AFSEL_R |= (1<<4);      // PL(4) alternate
function
    GPIO_PORTL_PCTL_R  |= 0x00030000; // PL(4) connected
toTimer0A
```

# Example program (input capture, edge-time)

```
// configure Timer 0
SYSCTL_RCGCTIMER_R |= (1<<0);      // TIMER0 = 1
while(!(SYSCTL_PRTIMER_R & 0x01)); // wait for timer 0 clock
TIMER0_CTL_R &= ~0x0001;           // disable Timer 0 for config
TIMER0_CFG_R  = 0x04;              // 2 x 16-bit mode

TIMER0_TAMR_R   |= 0x0007;         // capture, down, match disable
TIMER0_TAILR_R= 0xFFFF;            // ILR= 65535 (count interval)
TIMER0_CTL_R = TIMER0_CTL_R  | 0x0004; // falling edge

TIMER0_ICR_R |= 0x001F;            // clear all flags Timer0A
TIMER0_CTL_R |= 0x0001;            // enable Timer 0A
```

# Example program (input capture, edge-time)

```
while(1){
    //synchronize to next falling edge
    while((TIMER0_RIS_R & (1<<2))==0) // wait for capture event
        ;
    TIMER0_ICR_R |= (1<<2);      // clear Timer0A capture event flag
    TIMER0_CTL_R &= ~0x0001;      // re-enable Timer 0A
    TIMER0_CTL_R |= 0x0001;

    while((TIMER0_RIS_R & (1<<2))==0) // wait for capture event
        ;
    ulVal = TIMER0_TAR_R;        // save timer value at capture event

    printf("%d µs ",(unsigned short)(0xFFFF-ulVal)/16);
    TIMER0_ICR_R |= (1<<2);           // clear capture event flag
    }
}
```

# Example program (input capture, edge-time)

- Measurement result

  `4000 µs 4001 µs 4000 µs 4000 µs 3999 µs …`