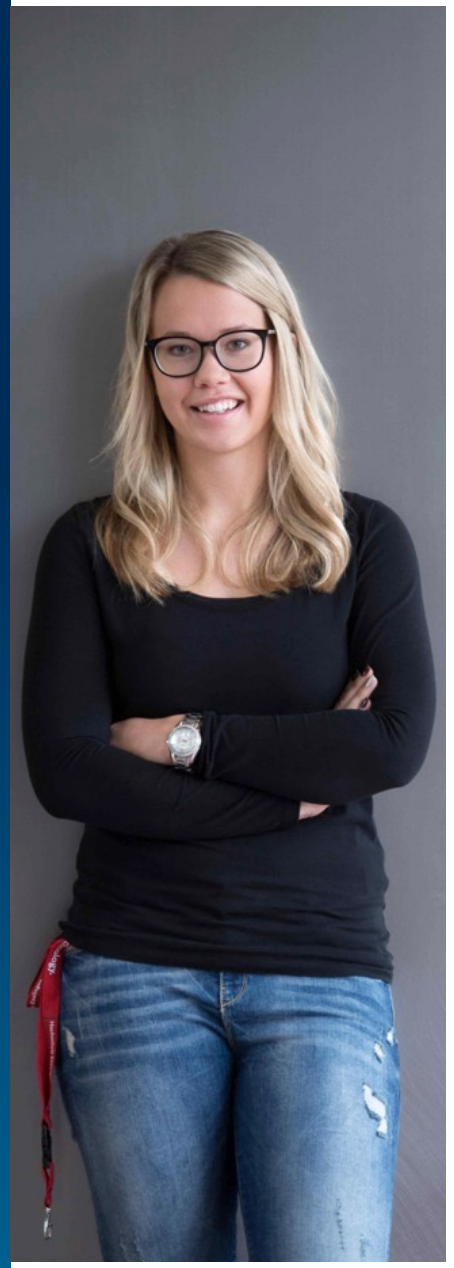
HOCHSCHULE
ESSLINGEN



LECTURE COMPUTER ARCHITECTURE

PERIPHERALS, DIGITAL/ ANALOG CONVERSION AND TIMED PERIPHERALS

RAINER KELLER

HOCHSCHULE
ESSLINGEN

CONTENT

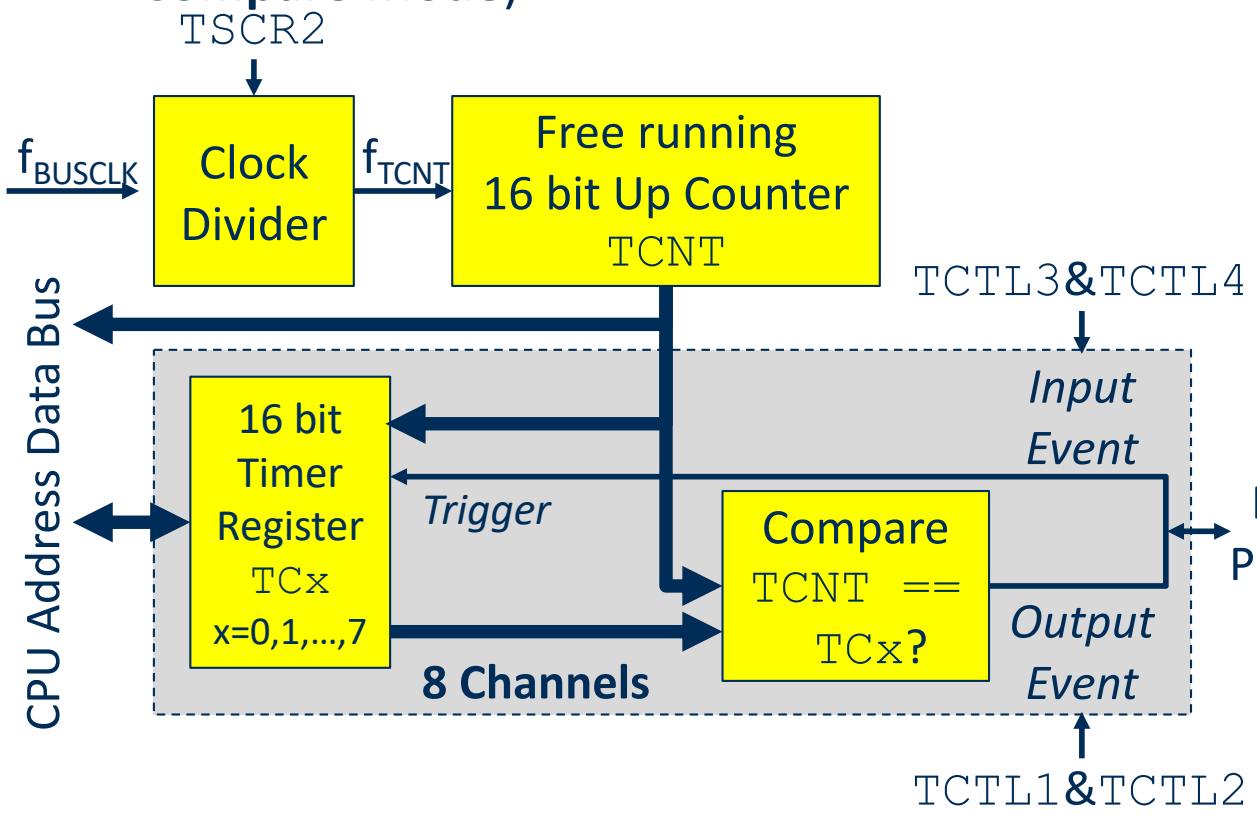
HOCHSCHULE
ESSLINGEN

Timer Unit

TIMER UNIT 1/4

Every modern microcontroller has a powerful timer unit for tasks like

- Measuring time differences/timeouts in programs
- Measuring time instants of external events, pulse periods and pulse lengths of input signals (**Input Capture Mode**)
- Generation of interrupts and output signals at programmable times (**Output Compare Mode**)



The HCS12 timer unit consists of:

- A free-running 16bit counter TCNT with programmable clock frequency f_{TCNT}
- 8 channels (Port T), which can be used as inputs in “Input Capture Mode” or as outputs in “Output Compare Mode.”

See `S12ECT_16B8CV1.pdf` for Enhanced Capture Timer Unit

TIMER UNIT 2/4

The Enhanced Capture Timer has many configurable options. Only the most important options will be described here. The description assumes that the CPU was reset and only differences to the default settings after reset will be described. For more Information see `S12ECT_16B8CV1.pdf [3.5]`

Configuration of the free-running 16 bit Counter TCNT

- Control Registers

`TSCR1` (8 bit register)	Enable the timer unit Bit 7 = 1 — Set to 1 to **enable** Bit 6…0=0 — Default after reset (don't touch)
`TSCR2` (8 bit register)	Set the timer clock frequency Bit 7…3 = 0 — Default after reset (don't touch) Bit 2…0 — **Set for Clock divider x**

Clock frequency*1: $f_{TCNT} = \frac{1}{T_{CNT}} = \frac{f_{BUSCLK}}{2^x}$

*1 Dragon12 operates at f_{BUSCLK}=24MHz

The counter can be configured to generate an interrupt on counter overflow (not shown here). In the associated ISR, overflow events may be counted by SW, so that the counter can be extended to more than 16 bit.

TIMER UNIT 3/4

Configuration of the 8 Timer Channels

- Control Registers

`TIOS` (8 bit register)	Select **Input Capture** or **Output Compare** Mode Bit y = 1 (y=0,1…,7): Channel y in Output Compare Mode Default y=0, i.e. Input Capture Mode
`TIE` (8 bit register)	Interrupt Enable per Channel Bit y = 1 (y=0,1,…,7): Channel y does generate interrupts Default: y=0, i.e. no interrupt Each channel has its own ISR, which will be called by its associated input or output events.
`TFLG1` (8 bit register)	Interrupt Flag indicates an timer interrupts event Bit y = 1 (y=0,1,…,7): Channel y triggered an interrupt Must be reset in the ISR by writing a 1 to the channel's bit in `TFLG1`.

- Additional configuration registers for **Output Compare Mode**:

Bit	7	6	5	4	3	2	1	0
`TCTL1` (8 bit)	Channel 7		Channel 6		Channel 5		Channel 4	
`TCTL2` (8 bit)	Channel 3		Channel 2		Channel 1		Channel 0	

Set the type of output event:

00	Output pin not used (timer used for interrupt generation only)
01	Toggle output pin
10	Clear output to 0
11	Set output to 1

TIMER UNIT 4/4

HOCHSCHULE ESSLINGEN

- Additional configuration registers for **Input Capture Mode**:

Bit	7	6	5	4	3	2	1	0
TCTL3 (8 bit)	Channel 7				Channel 6			
					Channel 5			
					Channel 4			
TCTL4 (8 bit)	Channel 3				Channel 2			
					Channel 1			
					Channel 0			

Set the type of input trigger event:

00	Input not used
01	Positive edge
10	Negative edge
11	Positive & negative edge

- Time-stamp registers for the 8 channels:
- Channel 0 — TC0 (8 bit register)

…

- Channel 7 — TC7 (8 bit register)

Software **writes** to this register in Output Compare Mode to set "the time" (counter value), when the channel's next output event shall be triggered.

Software **reads** this register in Input Capture Mode to "the time" (counter value) of the channel's last input event.

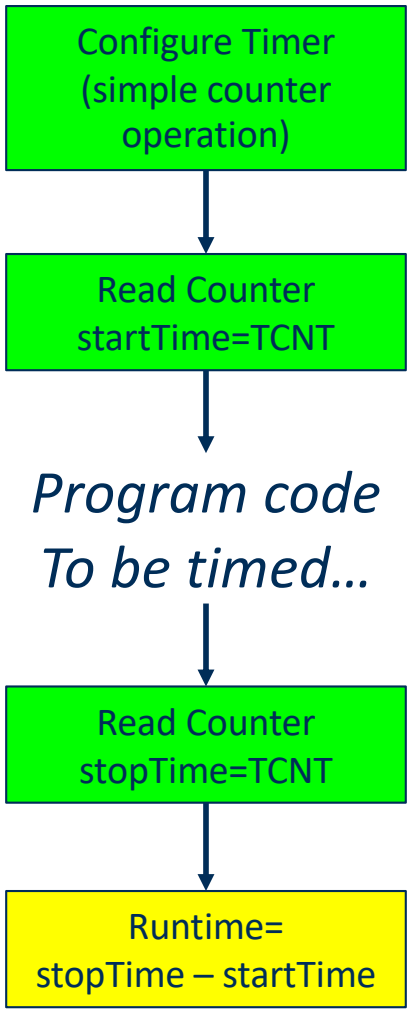
As the timer clock frequency is high, the timer value TCNT changes fast and overflows periodically. Software therefore

- Must read TCNT with a single 16 bit instruction, e.g. MOVW (never use two sequential 8 bit instructions MOVB, because the timer value may change in between reads.)
- When using TCNT to measure time, the time between events must never be greater than one counter period, i.e. $2^{16}/f_{TCNT}$. Then, the CPU's mod 2^{16}-arithmetic will handle timer overflows automatically, otherwise overflows must be counted & handled in SW.

TIMER UNIT – EXAMPLE 1

HOCHSCHULE ESSLINGEN

Example: Program for run-time measurement (see `timer1C.mcp`)

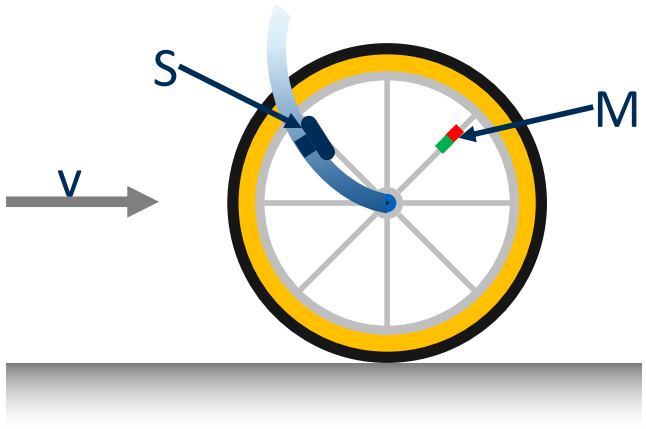



```
// --- Global variables --------------------------
unsigned int   startTime; // TCNT at start
unsigned int   stopTime;  // TCNT at end
unsigned long  deltaTime; // stopTime - startTime

void main(void) {
  EnableInterrupts;       // Needed for debugger
  // --- Initialize timer ------------------------


  // --- Program code to be measured -------------
     . . .
  // --- Compute run time ------------------------


}
```

Note: Resolution is 1 timer clock period, i.e. 5.3µs, measurement range: 1 counter period, i.e. max. 350 ms. A smaller clock period will increase measurement resolution, but will reduce measurement range.

HOCHSCHULE ESSLINGEN

Example: Measure the Period of a Pulse Signal in **Input Capture Mode.**



- The pulse signal is generated via a coil S and a rotating magnet M.
- When the magnet passes the coil a voltage impulse is induced.
- A Comparator converts the analog voltage impulse into a digital pulse signal
- Wheel speed $n \approx \frac{1}{impulse\ period\ T}$
- With the known wheel radius r the bicycle's speed $v = 2\pi * r * n$

A similar setup with a tooth wheel and a Hall sensor, which generate ~60 pulses per revolution is used to measure wheel speed of cars in ABS/ESP systems.

TIMER UNIT – EXAMPLE 2 2/3

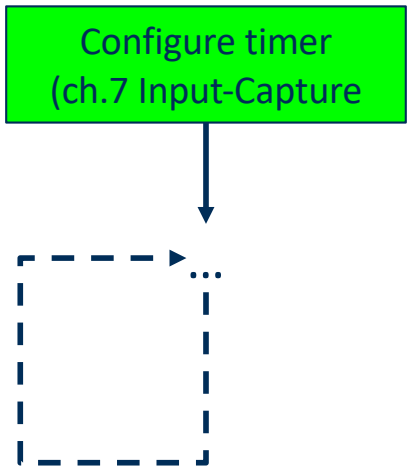
HOCHSCHULE ESSLINGEN

Core sample: (see `timer3C.mcp`)

Pulse signal connected to Port T.7 (input with frequency range 10Hz … 10kHz)

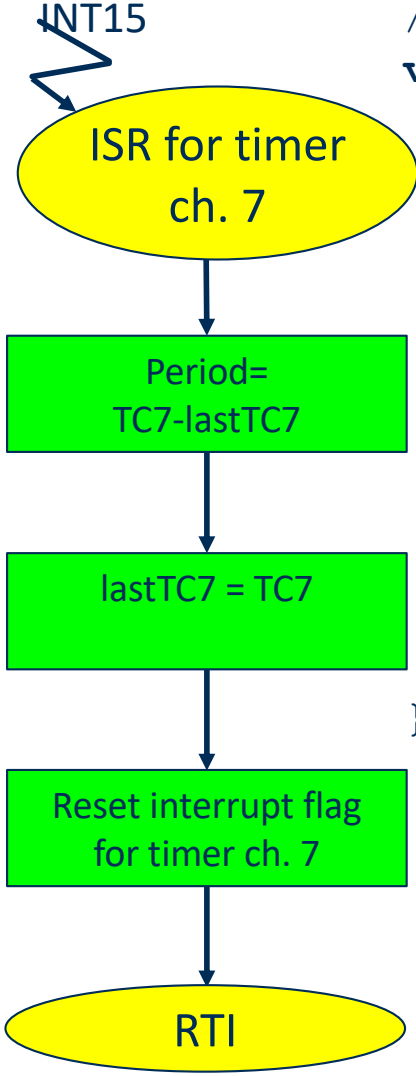
```
// --- Global variables ------------------------------
unsigned int signalPeriod=0;  // Signal period in TCNT
                              // clock periods
unsigned int lastTC7=0;       // TC7 at last inp.event

void main(void) {
  …
  //-- Initialize timer ----------------------------
```

Main program:

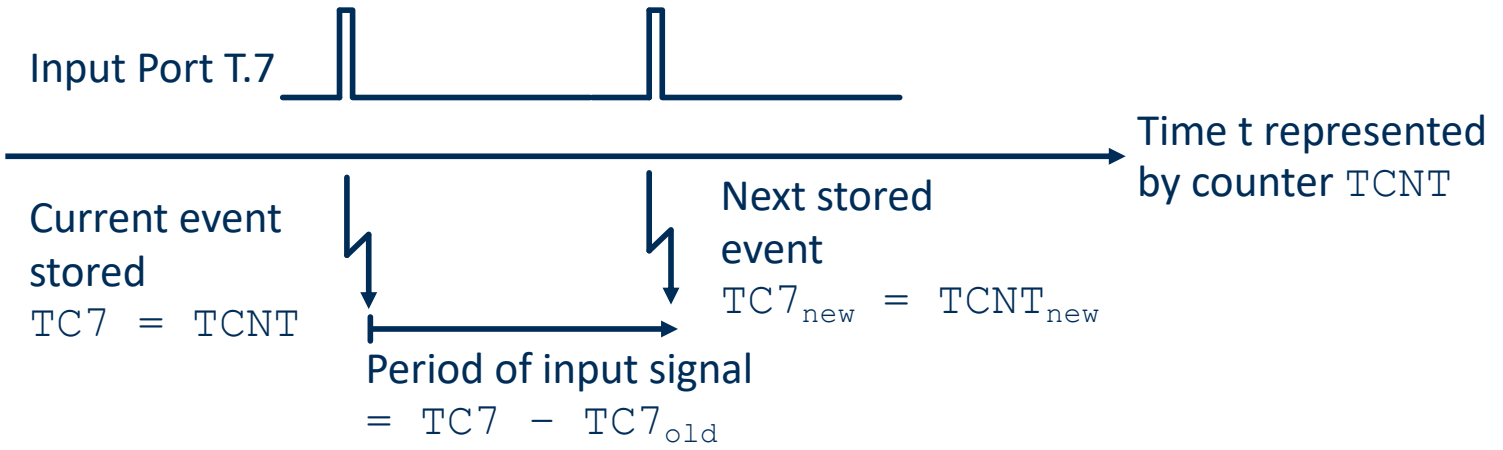



```
  for(;;) {}                  // Infinite loop
}
```

TIMER UNIT – EXAMPLE 2 3/3



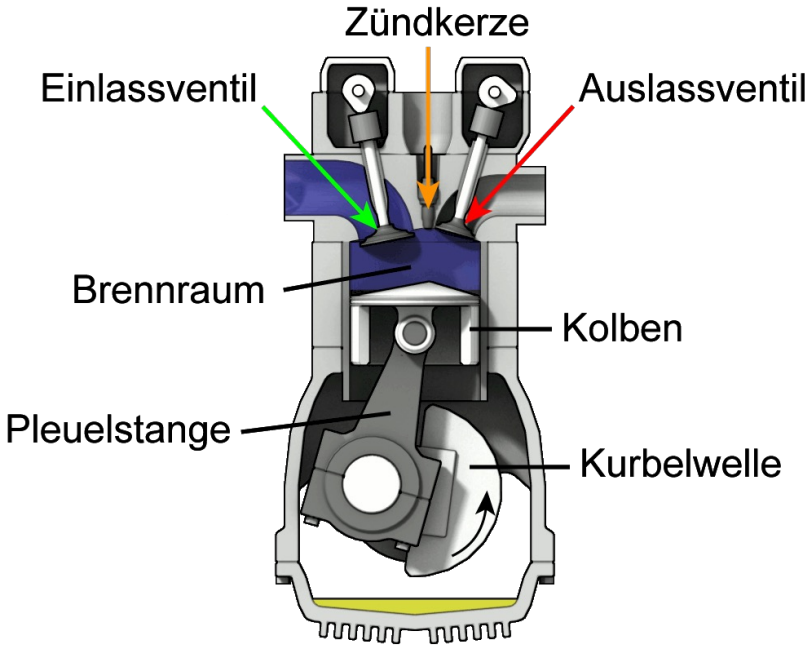

```
// --- Interrupt service routine for timer channel 7 -
void                    timer7Isr(void) {

  unsigned int temp = TC7;// Get current TC7
  // Compute signalPeriod in timer clock periods
  signalPeriod = TC7 - lastTC7;

  lastTC7 = temp;           // Save current TC7

  TFLG1 = TFLG1 | 0x80;     // Reset ch7 interrupt flag
}
```



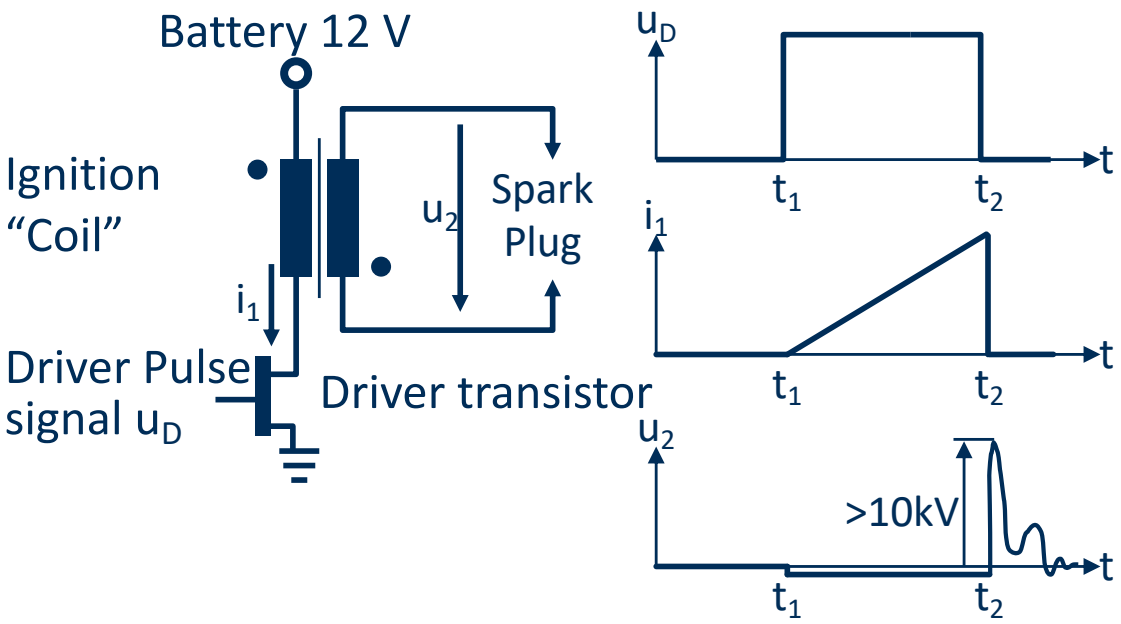
I.e. “time” is an integer multiple of counter clock period TCNT

TIMER UNIT – EXAMPLE 3 1/3

Example: Generate driving pulse for the spark plugs of a combustion engine in **Output Compare Mode.**



Source: https://www.leifiphysik.de



- The ignition spark is generated via the ignition "coil".
- The ignition timing t_2 (but also t_1) must be controlled **precisely**, because they influence efficiency and emissions → timing errors < 1µs required!

TIMER UNIT – EXAMPLE 3 2/3

HOCHSCHULE ESSLINGEN

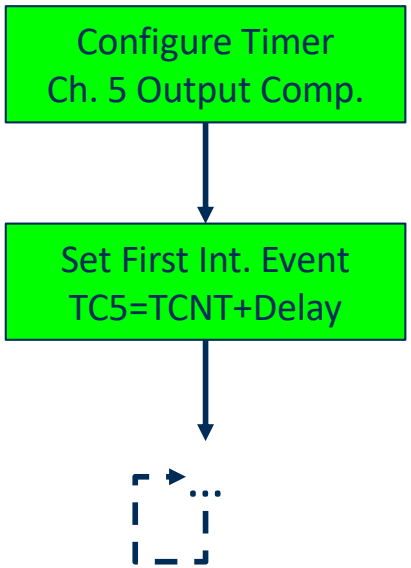
Dragon12 Beeper in Output-Compare Mode (see `timer2C.mcp`)

A beeper (buzzer) on port T.5 (used as output) shall be driven by a 500 Hz pulse signal with a 1:1 duty cycle.

```
#define DELAY (24000/128) // Delay 1ms*24MHz / 2^7
```

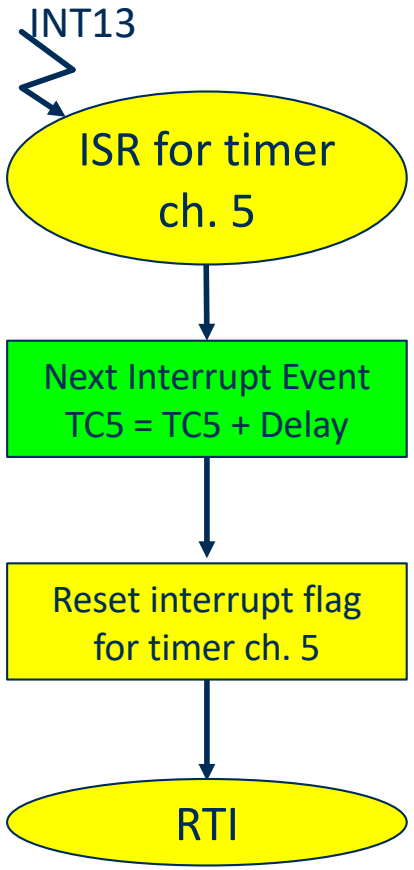
Main program:

```
void main(void) {
  EnableInterrupts;          // Global interrupt enable
  …
  // --- Initialize Timer ----------------------------
  TSCR1 = TSCR1 | 0x80;      // Enable timer unit
  TSCR2 = 0x07;              // Clock period 2^7/24 MHz
```

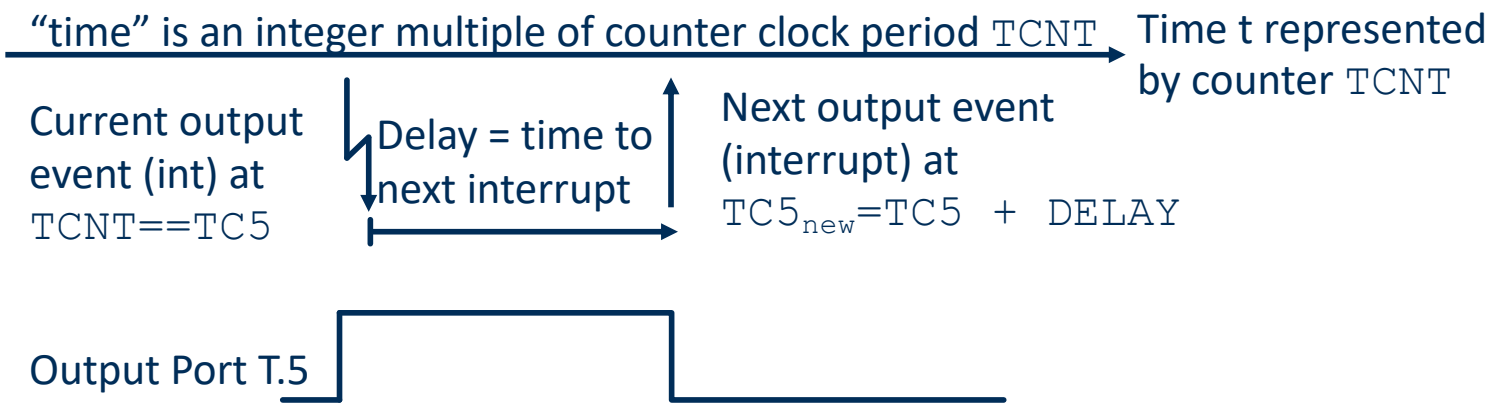



```
  for (;;) {}                // Infinite loop
}
```

TIMER UNIT – EXAMPLE 3 3/3




```
// --- Interrupt service routine for timer channel 5 -
// Output port T.5 is toggled automatically, whenever
// this ISR is called
void                    timer5Isr(void) {
}
```

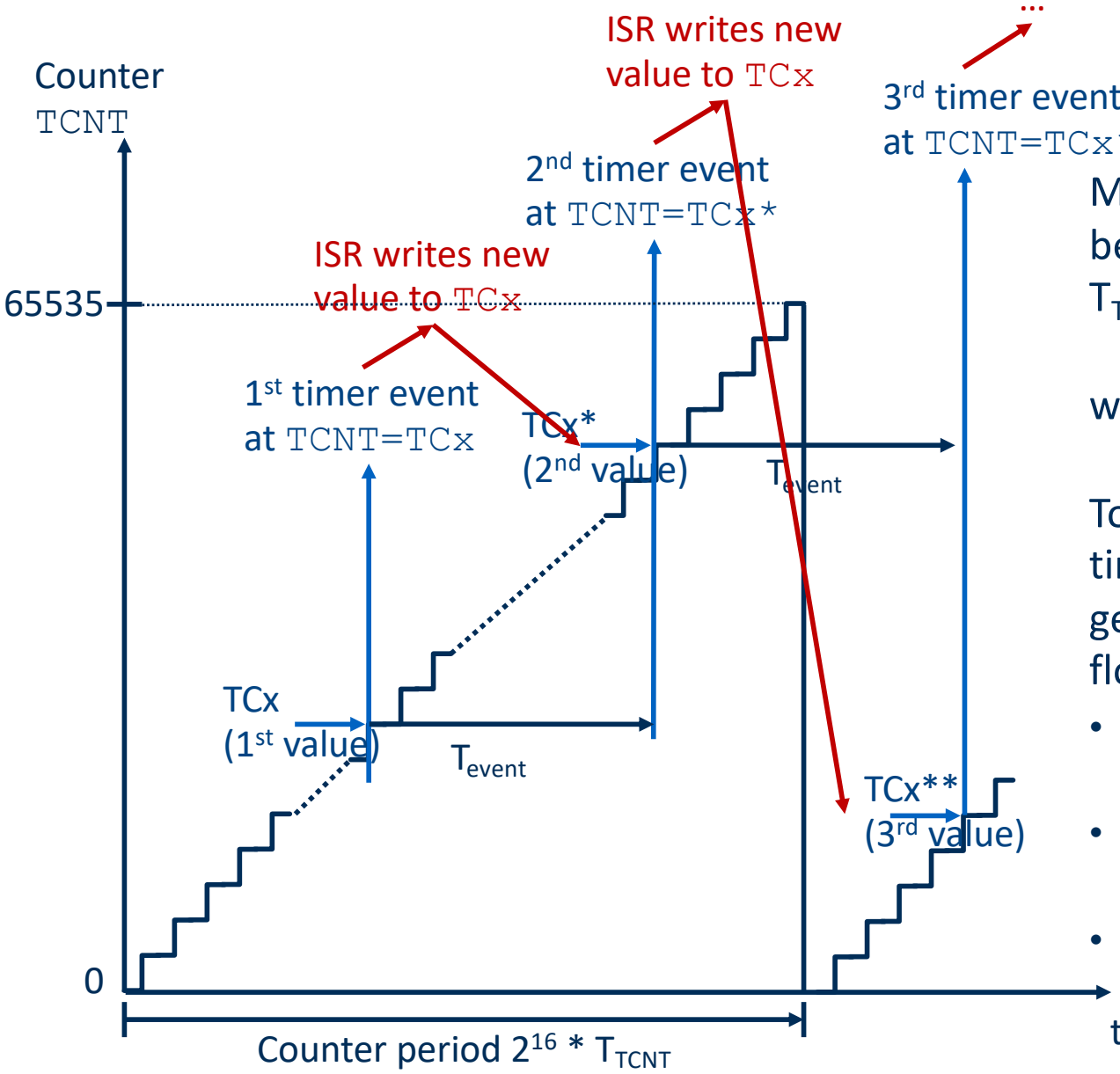


By setting the bits in TCTL1 to 0, the output compare mode will trigger an ISR only at predefined time, but does not change the port pins output signal.

TIMER UNIT – HW/SW INTERACTION

HOCHSCHULE ESSLINGEN

Interaction between Hardware and Software in Output Compare Mode:



Minimum & Maximum time between timer events:

$T_{TCNT} << T_{ISR} < T_{Event} < 2^{16}*T_{TCNT}$

with T_{ISR} = execution time of ISR.

To generate times > $2^{16}*T_{TCNT}$, the timer can be configured to generate an interrupt upon over-flow of the 16 bit TCNT:

- Timer Overflow Int. Enable: Register TSCR2 Bit 7 = 1
- Timer Overflow Int. Flag: Register TFLG2 Bit 7 = 1
- Interrupt Vector Table: Entry 16 @ address $FFDE

APPENDIX B: CLOCK GENERATOR AND CLOCK DIVIDER

HOCHSCHULE ESSLINGEN

RTI Interrupt:

Interrupt period $T_{RTI} = 2^{9+X} * (Y + 1) / f_{OSCCLK}$. All values in ms @ f_{OSCCLK} = 4 MHz

	Y=0	Y=1	Y=2	Y=3	Y=4	Y=5	Y=6	Y=7	Y=8	Y=9	Y=10	Y=11	Y=12	Y=13	Y=14	Y=15
X=1:	0.256	0.512	0.768	1.024	1.280	1.536	1.792	2.048	2.304	2.560	2.816	3.072	3.324	3.584	3.840	4.096
X=2:	0.512	1.024	1.536	2.048	2.560	3.072	3.584	4.096	4.608	5.120	5.632	6.144	6.656	7.168	7.680	8.192
X=3:	1.024	2.048	3.072	4.096	5.120	6.144	7.168	8.192	9.216	10.240	11.264	12.288	13.312	14.336	15.360	16.384
X=4:	2.048	4.096	6.144	8.192	10.240	12.288	14.336	16.384	18.432	20.480	22.528	24.576	26.624	28.672	30.720	32.768
X=5:	4.096	8.192	12.288	16.384	20.480	24.576	28.672	32.768	36.864	40.960	45.056	49.152	53.248	57.344	51.440	65.536
X=6:	8.192	16.384	24.576	32.768	40.960	49.152	57.344	65.536	73.728	81.920	90.112	98.304	106.496	114.688	122.880	131.072
X=7:	16.384	32.768	49.152	65.356	81.920	98.304	114.68	131.072	147.456	163.840	180.224	196.608	212.992	229.376	245.760	262.144

ECT Timer: @ f_{BUSCLK} = 24 MHz

Clock period $T_{TCNT} = 1/f_{TCNT} = 2^X/f_{BUSCLK}$. In µs

X=0	X=1	X=2	X=3	X=4	X=5	X=6	X=7
0.042	0.083	0.167	0.333	0.667	1.333	2.667	5.333

Clock period $T_P = 2^{16} * TTC_{NT}$ Values in ms

X=0	X=1	X=2	X=3	X=4	X=5	X=6	X=7
2.731	5.461	10.293	21.845	43.691	87.381	174.763	349.525

PWM: @ f_{BUSCLK} = 24 MHz

Clock period of fast clock $T_{A/B} = 2^X/f_{BUSCLK}$. In µs

X=0	X=1	X=2	X=3	X=4	X=5	X=6	X=7
0.042	0.083	0.167	0.333	0.667	1.333	2.667	5.333

Clock period of slow clock $T_{SA/B} = 2*y*T_{A/B}$. In µs

	X=0	X=1	X=2	X=3	X=4	X=5	X=6	X=7
Y=1	0.083	0.167	0.333	0.667	1.333	2.667	5.333	10.667
…	…	…	…					
Y=255	21.250	42.500	85.000	170000	340000	680000	1360000	2720000

Serial Interface: @ f_{BUSCLK} = 24 MHz

Clock Divider Register SCIxBD = $f_{BUSCLK}/(16*f_{BIT})$

f_{BIT}=300 bit/s	600 bit/s	1,2 kbit/s	2,4kbit/s	4,8kbit/s	9,6kbit/s	19,2kbits	38,4kbit/s	57,6kbit/s	115,2kbit/s
5000	2500	1250	625	313	156	78	39	26	13

LECTURE: LITERATURE

HOCHSCHULE ESSLINGEN

According to list of literature:

- Patterson, D., Hennessy, J.: *Computer Organization and Design*, Kaufmann, 2011
  (Deutsche Übersetzung: Rechnerorganisation und –entwurf, Spektrum)
- Hennessy, J., Patterson, D.: *Computer Architecture: A quantitative Approach, **5th edition***, Morgan Kaufmann, 2017
- Tanenbaum, A., Austin, T.: *Structured Computer Organization*, Pearson, 6th edition, 2013
- Tanenbaum, A., Austin, T.: *Rechnerarchitektur: von der digitalen Logik zum Parallelrechner*, Pearson, 6th ed., 2014
- Huang, H.W.: *The HCS12/9S12. An Introduction to the HW and SW interface*, Thomson Learning, 2009
- Beierlein, T.: *Taschenbuch Mikroprozessortechnik*, Carl-Hanser Verl., 2011