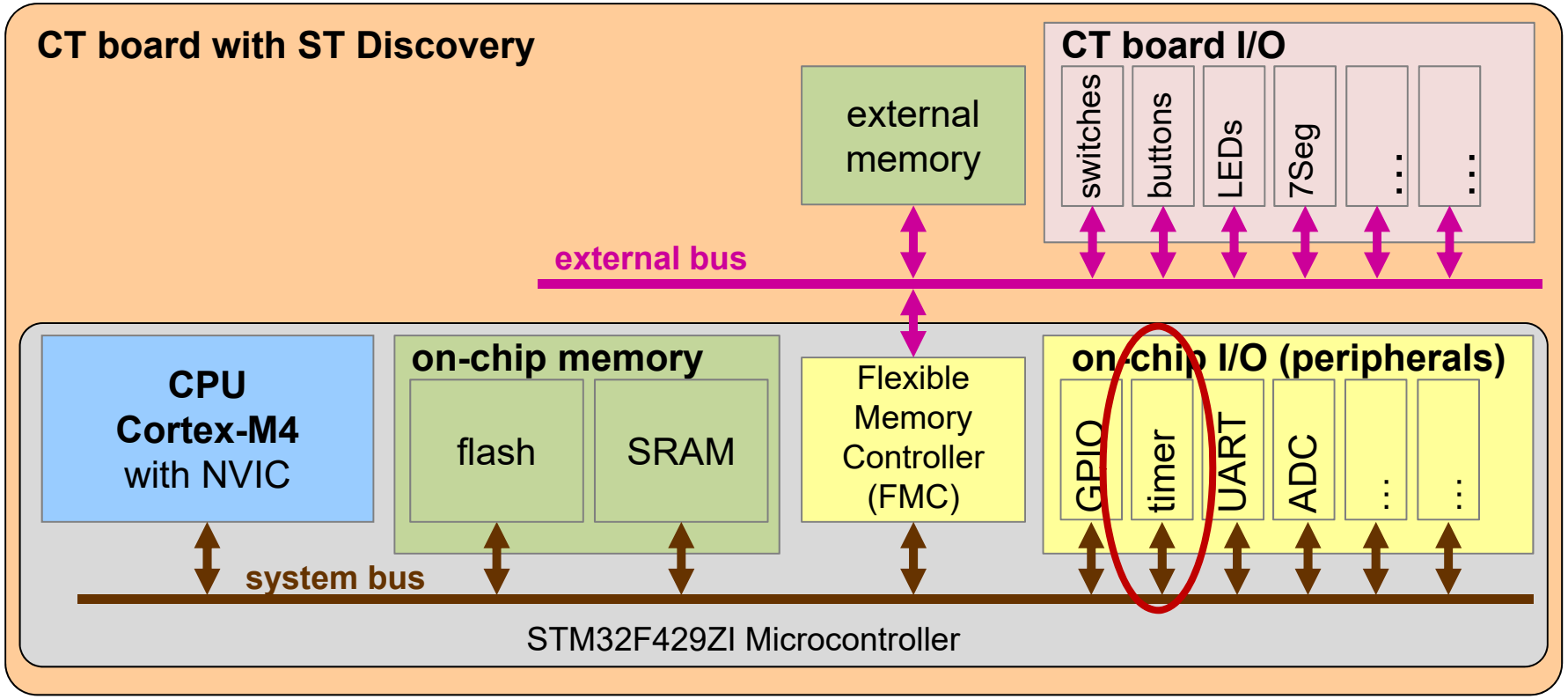
Timer / Counter

Computer Engineering 2

Overview

- **Timer / Counter**
  - Reference Manual pages 576-635



Agenda

School of Engineering
InES Institute of Embedded Systems

- **Timer / Counter – Basic Ideas**
- **Timers / Counters**
- **ST32F4xx Timers**
- **Timer Configuration**
- **Input Capture**
- **Pulse-Width-Modulation (PWM)**
- **Output Compare – Generating PWM Signals**
- **Capture / Compare Configuration**

Learning Objectives

At the end of this lesson you will be able

- to describe the functionality of timers
- to explain the realization of a timer
- to give an overview of timer functions
- to describe the timers of a real processor
- to interpret block diagrams of timers
- to explain the concepts of capture / compare
- to explain the idea of PWM
- to program timers using documents / data sheets

Timer / Counter – Basic Ideas

School of Engineering
InES Institute of Embedded Systems

- **Binary up- or down-counter**
  - Counts events / clock pulses or external signals
  - Output after a defined number of events (e.g. interrupt)

  - Timer: counting clock cycles or processor cycles (periodic)
  - Counter: counting events

- **Use**
  - Count of events
  - Measure of time, frequencies, phases, periods
  - Generate intervals, row of pulses, interrupts

Timer / Counter – Basic Ideas

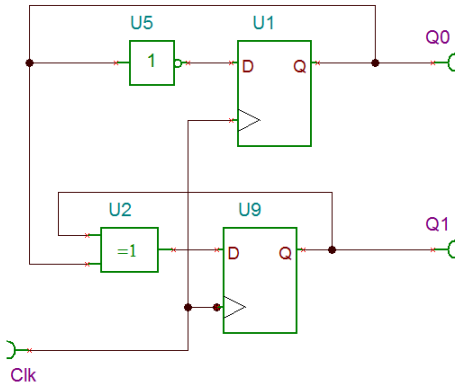
School of Engineering
InES Institute of Embedded Systems

- **Application examples**
  - Trigger for periodic software tasks
    - Display refresh
    - Sampling inputs e.g. buttons
  - Count number of pulses on input pin
  - Measure time between rising edges of an input pin
  - Generate defined sequence of pulses on an output pin

Timer / Counter – Basic Ideas

School of Engineering
InES Institute of Embedded Systems

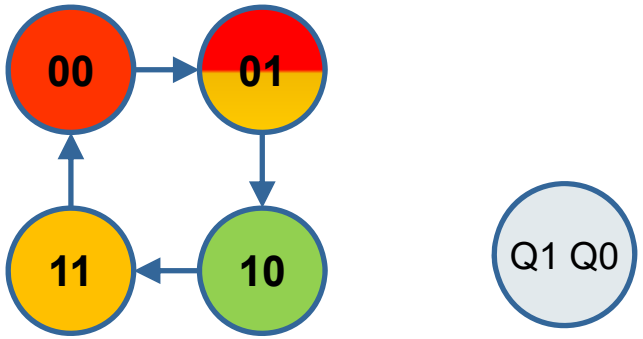
- **Repetition: 2-bit binary counter**



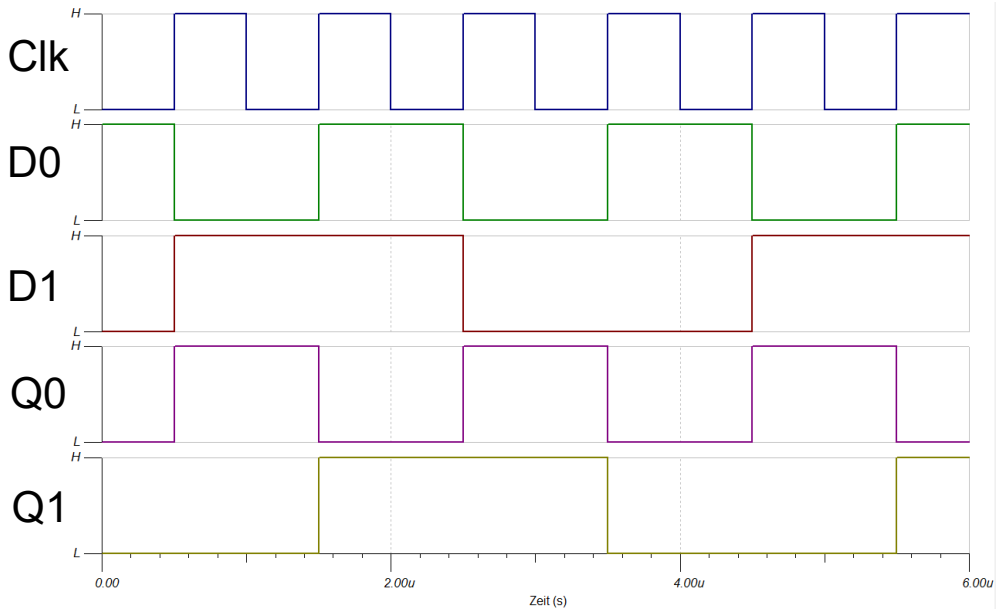
D1 = Q0 $ Q1
D0 = !Q0

Q1	Q0	D1	D0
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

State diagram



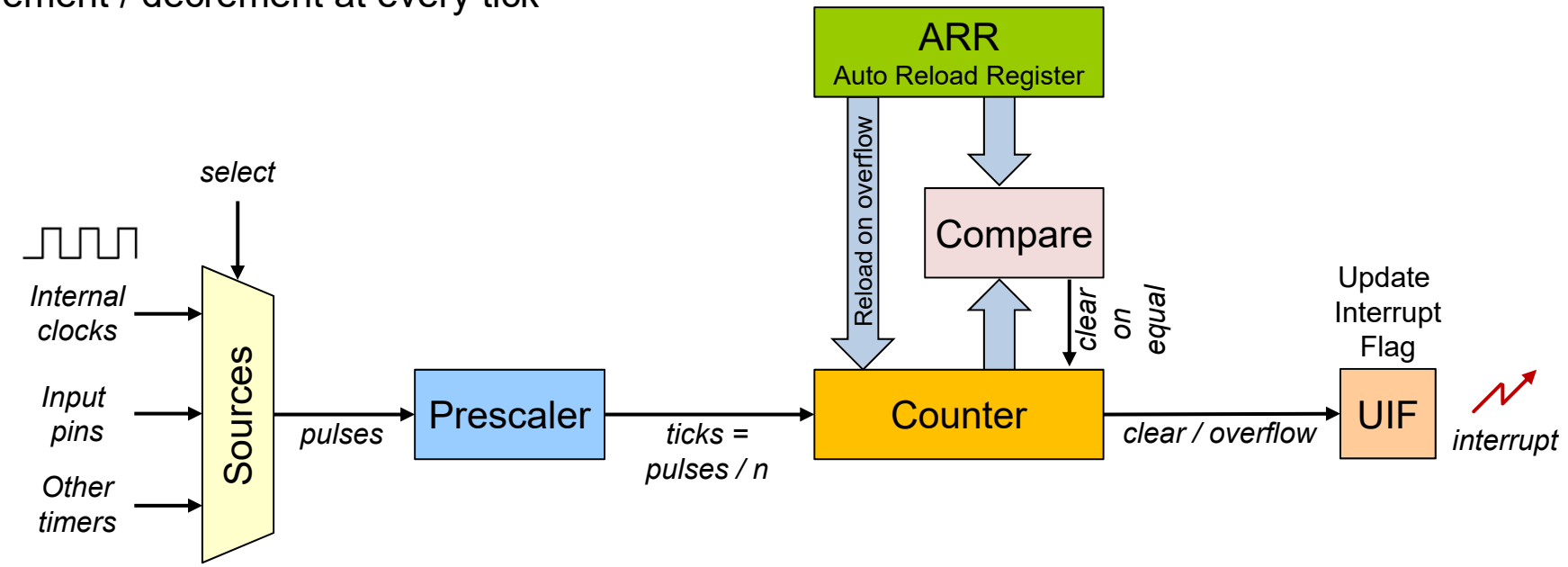
Timing diagram



Timers / Counters

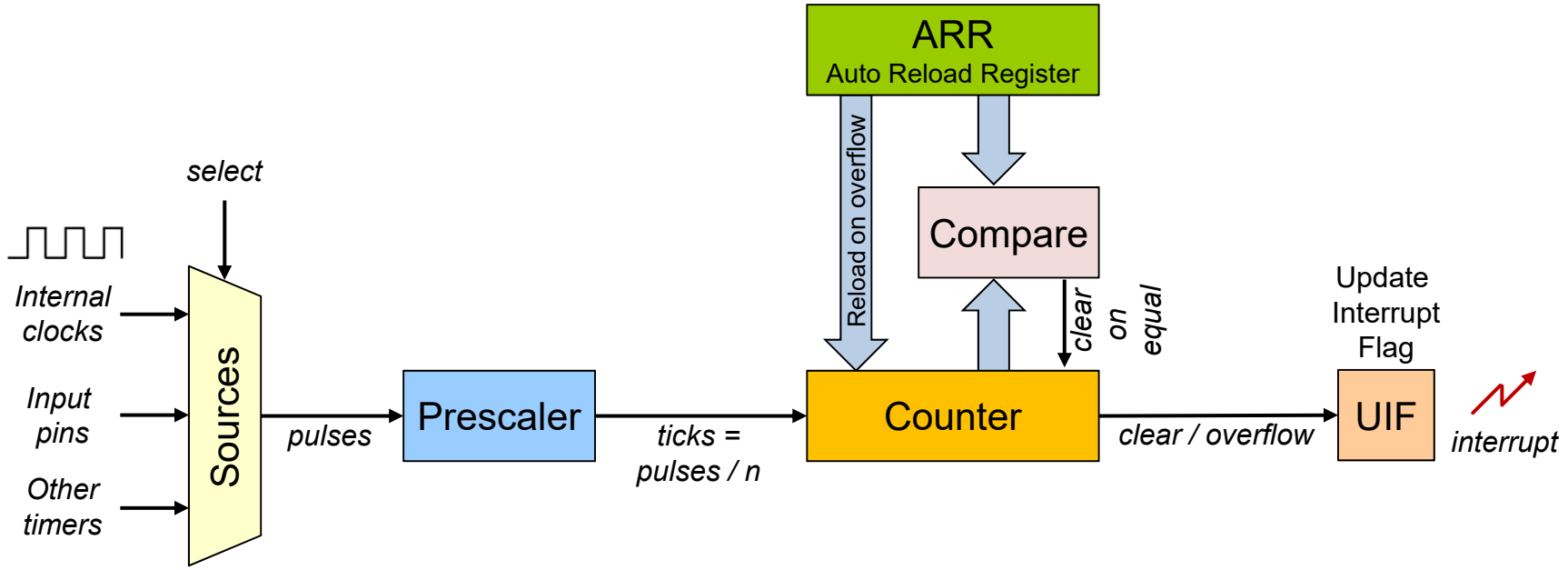
School of Engineering
InES Institute of Embedded Systems

- **Function**
  - Configure as up- or down-counter
  - Select source
  - 16-bit / 32-bit counter register
    - Increment / decrement at every tick

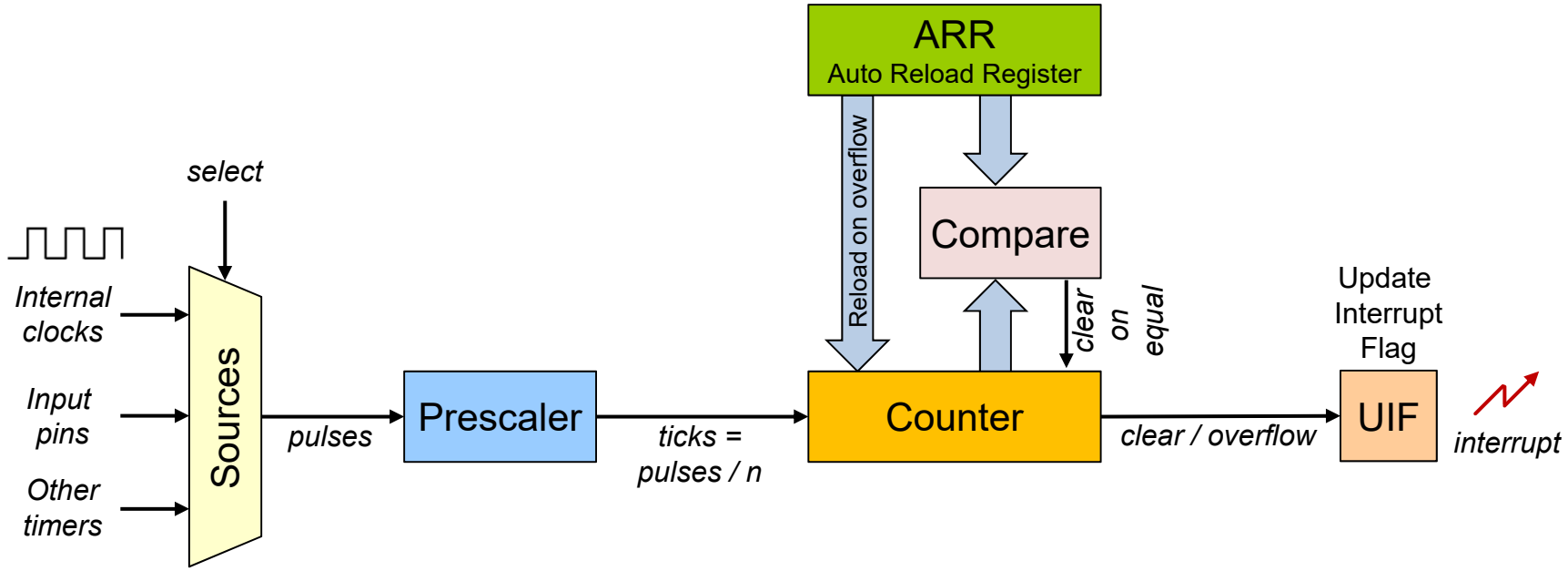


School of Engineering
InES Institute of Embedded Systems

- **Counter overflow**
  - Up Clear if counter reaches ARR
  - Down Reload with ARR if counter reaches zero
  - Set interrupt flag → trigger interrupt

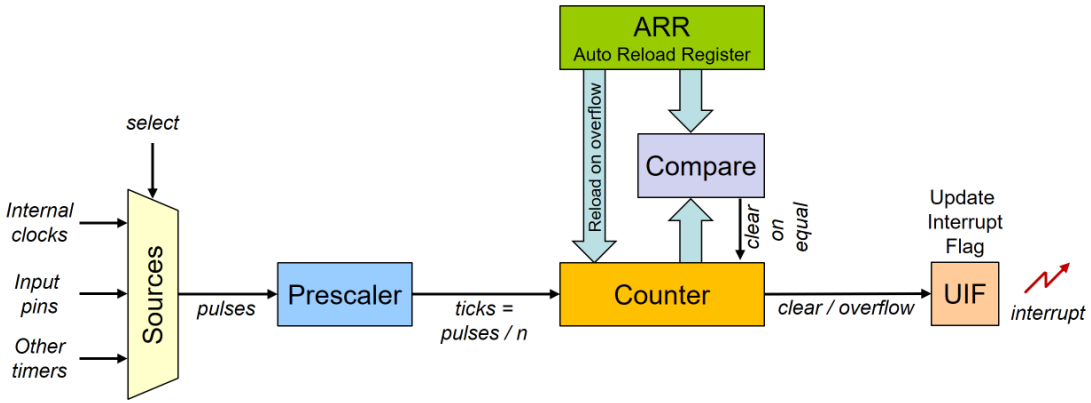


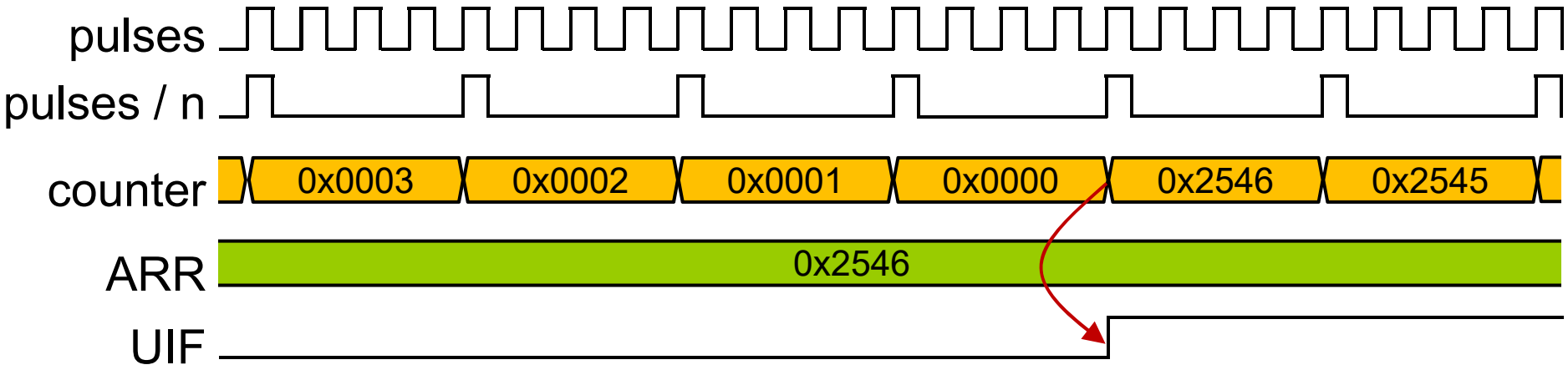
- **Prescaler**
  - Increase counting range
  - Count only every n-th event
    - e.g. n = {1, 2, 4, 8, 32, 64 , …}



School of Engineering
InES Institute of Embedded Systems

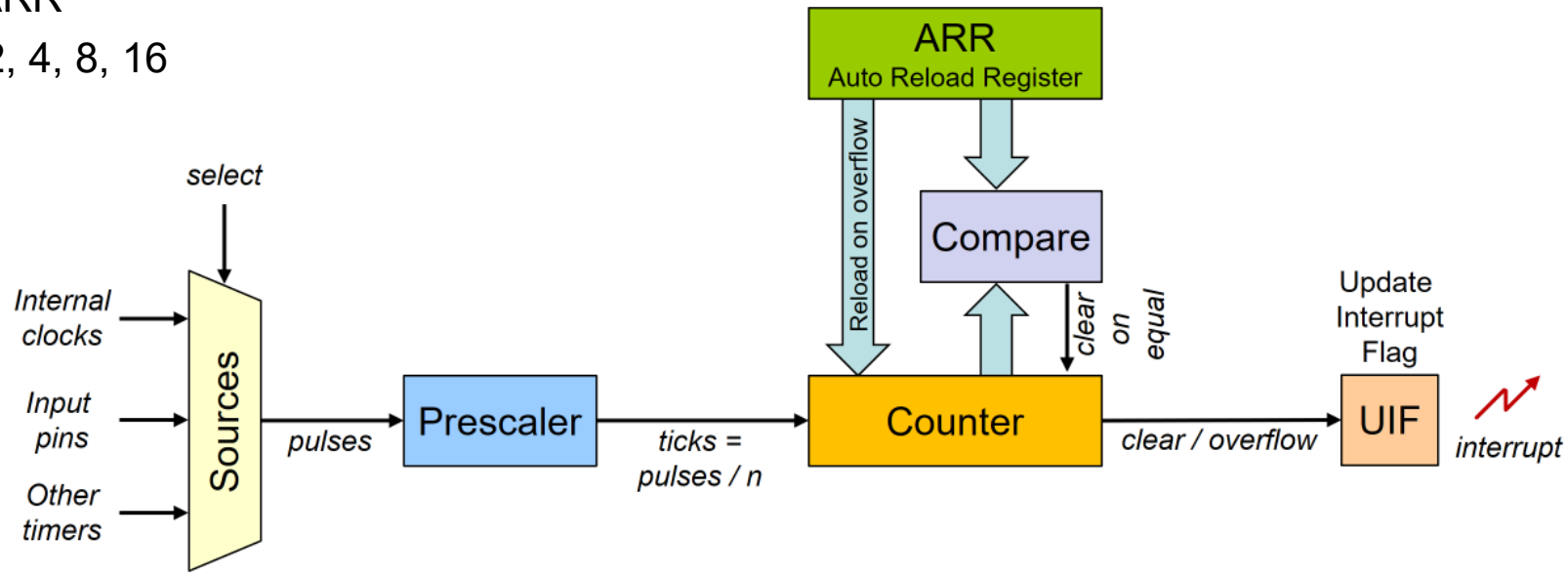
- **Down-counting example**
  - Prescaler → divide by 4
  - Count down to zero
    - Set interrupt request (UIF)
    - Restart from value in ARR





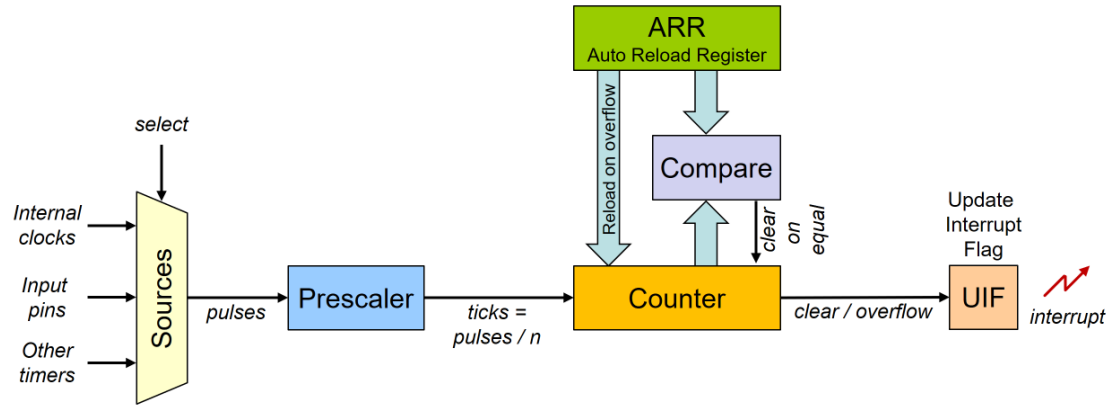
School of Engineering
InES Institute of Embedded Systems

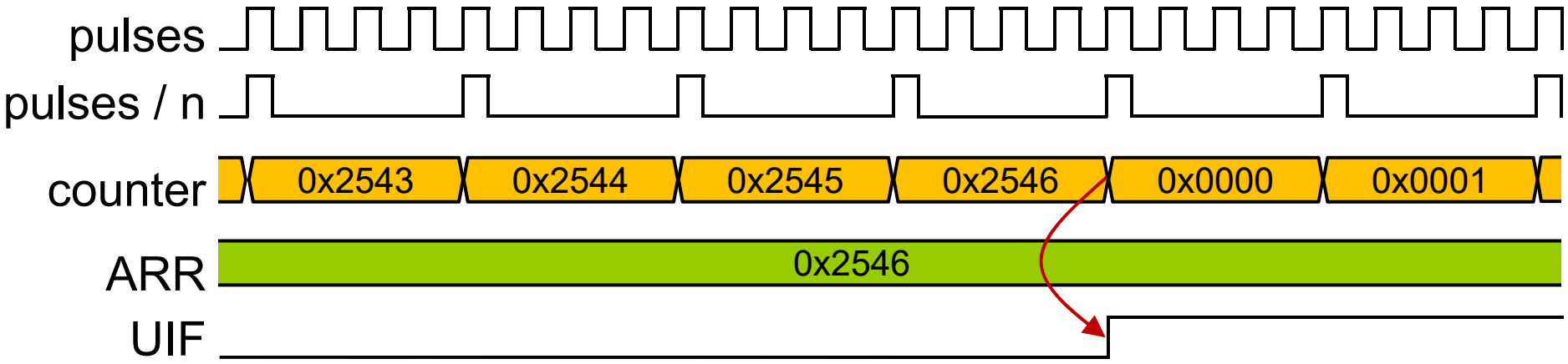
- **Exercise**
  - Source: 1 MHz
  - What needs to be set, when we want an interrupt every
    - 50 ms → 20 Hz
    - 1 s → 1 Hz
  - Assume
    - **16-bit** counter / ARR
    - Prescaler n = 1, 2, 4, 8, 16
    - Down-counter



Timers / Counters

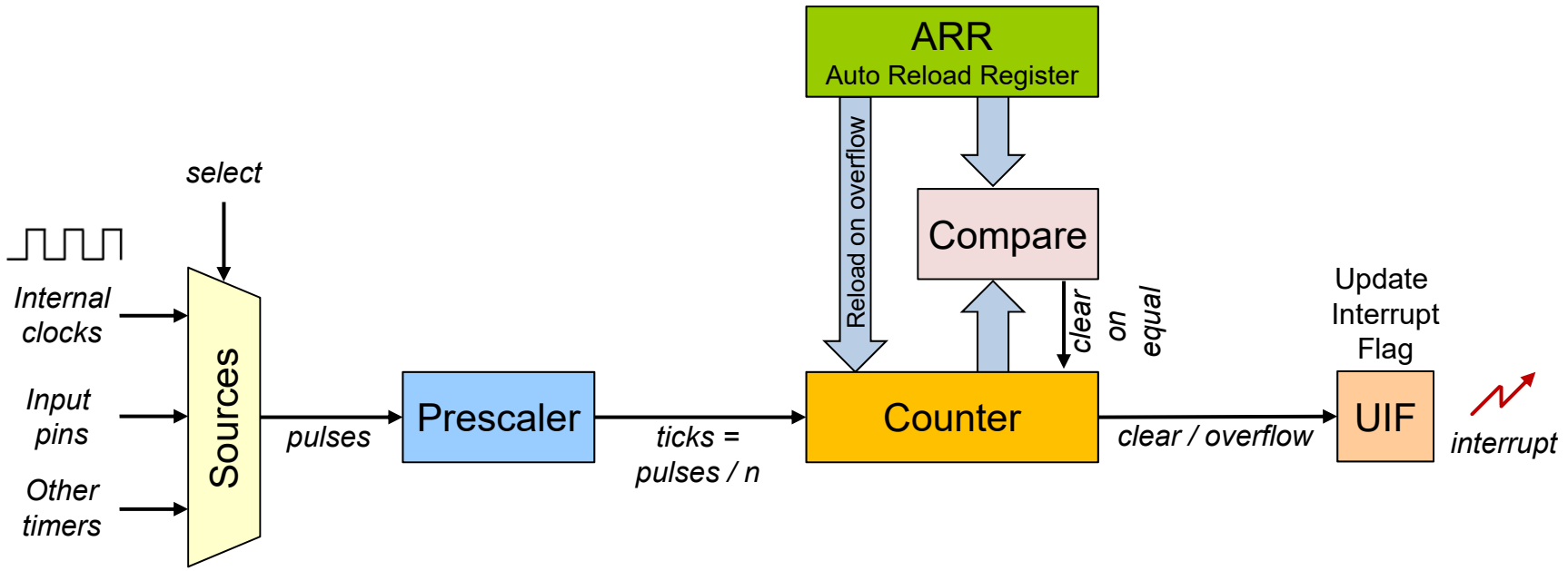
- **Up-counting example**
  - Prescaler → divide by 4
  - Count up to the value in reload register
    - Set interrupt request
    - Restart from 0





Timers / Counters

For a given problem there are often different ways to use the available hardware.
→ E.g. up-counter vs. down-counter.



ST32F4xx Timers

School of Engineering
InES Institute of Embedded Systems

- **Full-featured general-purpose timers**
  - TIM2, TIM3, TIM4, TIM5
- **General-purpose timers**
  - TIM9, TIM10, TIM11, TIM12, TIM13, and TIM14
- **Advanced-control timers**
  - TIM1, TIM8
- **Basic timers**
  - TIM6, TIM7

Explained Here: TIM 2

Not explained in detail see reference manual

ST32F4xx Timers

School of Engineering
InES Institute of Embedded Systems

- **Timers TIM2 - TIM5**
  - 16-bit → TIM3 and TIM4 32-bit → TIM2 and TIM5
    - Up, down, up/down
    - Auto-reload
  - 16-bit programmable prescaler
    - Dividing counter clock frequency by factor between 1 and 65536
  - Up to 4 independent channels for:
    - Input capture
    - Output compare
    - PWM generation
    - One-pulse mode output
  - Synchronization circuit
    - Control timer with external signals
    - Interconnect several timers together
  - Interrupt/DMA generation based on several events

Timer Configuration

School of Engineering
InES Institute of Embedded Systems

- **Register address = Base address + Offset**
  - Offset address is given for each register in Reference Manual
  - Base address defined in memory map
    → Reference Manual

Boundary address	Peripheral
0x4000 0C00 - 0x4000 0FFF	TIM5
0x4000 0800 - 0x4000 0BFF	TIM4
0x4000 0400 - 0x4000 07FF	TIM3
0x4000 0000 - 0x4000 03FF	TIM2
0x4002 3800 - 0x4002 3BFF	RCC

RCC: Reset and Clock Configuration

Timer Configuration

School of Engineering
InES Institute of Embedded Systems

- **Enable timer block**
  - RCC APB1 peripheral clock enable register (RCC_APB1ENR)
  - RCC = Reset and Clock Control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8 EN	UART7 EN	DAC EN	PWR EN	Reser-ved	CAN2 EN	CAN1 EN	Reser-ved	I2C3 EN	I2C2 EN	I2C1 EN	UART5 EN	UART4 EN	USART 3 EN	USART 2 EN	Reser-ved
rw	rw	rw	rw		rw	rw		rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 EN	SPI2 EN	Reserved		WWDG EN	Reserved		TIM14 EN	TIM13 EN	TIM12 EN	TIM7 EN	TIM6 EN	TIM5 EN	TIM4 EN	TIM3 EN	TIM2 EN
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

Reference Manual

Timer Configuration → Selected Registers

School of Engineering
InES Institute of Embedded Systems

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Reserved																						CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x08	TIMx_SMCR	Reserved																ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			Reserved	SMS[2:0]		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x0C	TIMx_DIER	Reserved																	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Reserved	TIE	Reserved	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value																		0	0	0	0	0	0	0		0		0	0	0	0	0
0x10	TIMx_SR	Reserved																			CC4OF	CC3OF	CC2OF	CC1OF	Reserved		TIF	Reserved	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value																				0	0	0	0			0		0	0	0	0	0
0x14	TIMx_EGR	Reserved																									TG	Reserved	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																										0		0	0	0	0	0
0x24	TIMx_CNT	CNT[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CNT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Reserved																PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIMx_ARR	ARR[31:16] (TIM2 and TIM5 only, reserved on the other timers)																ARR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Enable Interrupts

Interrupt Flags
To be reset in Interrupt Service Routine

```
#define TIM2 ( (reg_tim_t *) 0x40000000 )
#define TIM3 ( (reg_tim_t *) 0x40000400 )
#define TIM4 ( (reg_tim_t *) 0x40000800 )
#define TIM5 ( (reg_tim_t *) 0x40000c00 )
```

```
typedef struct {
    volatile uint32_t CR1;
    volatile uint32_t CR2;
    volatile uint32_t SMCR;
    volatile uint32_t DIER;
    volatile uint32_t SR;
    volatile uint32_t EGR;
    volatile uint32_t CCMR1;
    volatile uint32_t CCMR2;
    volatile uint32_t CCER;
    volatile uint32_t CNT;
    volatile uint32_t PSC;
    volatile uint32_t ARR;
    volatile uint32_t RCR;
    volatile uint32_t CCR1;
    volatile uint32_t CCR2;
    volatile uint32_t CCR3;
    volatile uint32_t CCR4;
    volatile uint32_t BDTR;
    volatile uint32_t DCR;
    volatile uint32_t DMAR;
    volatile uint32_t OR;
} reg_tim_t;
```

Timer Configuration

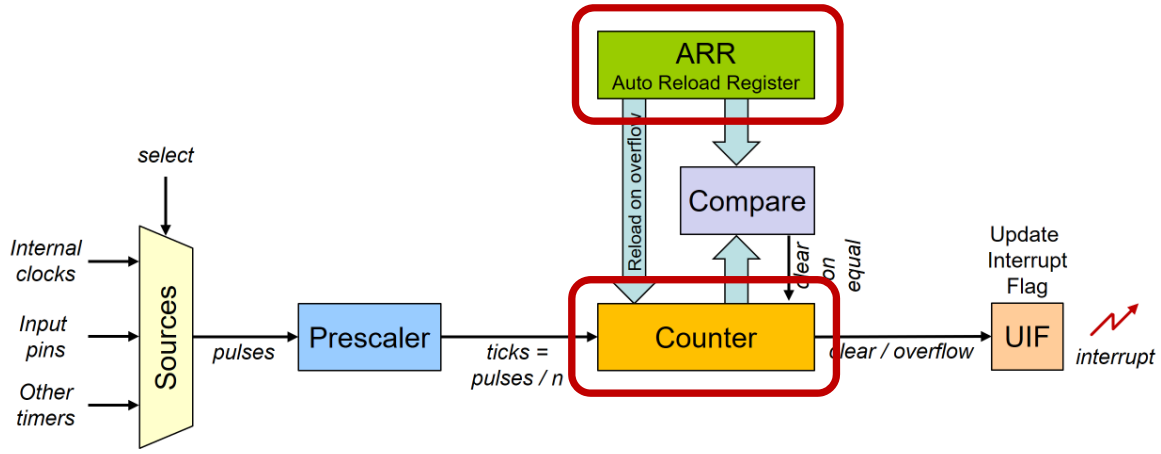
- **TIMx counter (TIMx_CNT)**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

- **TIMx auto-reload register (TIMx_ARR)**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

  - ARR is the value to be loaded in the actual auto-reload register

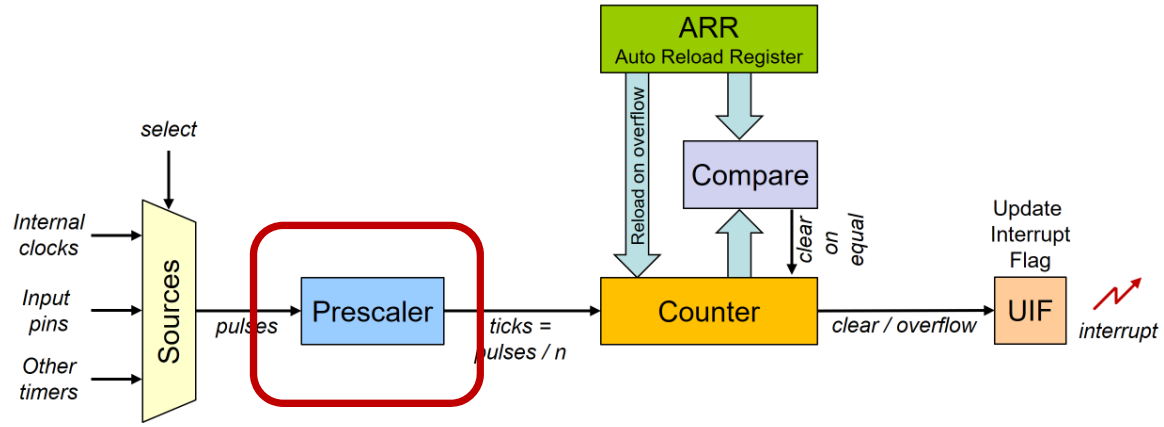


Timer Configuration

- **TIMx prescaler (TIMx_PSC)**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

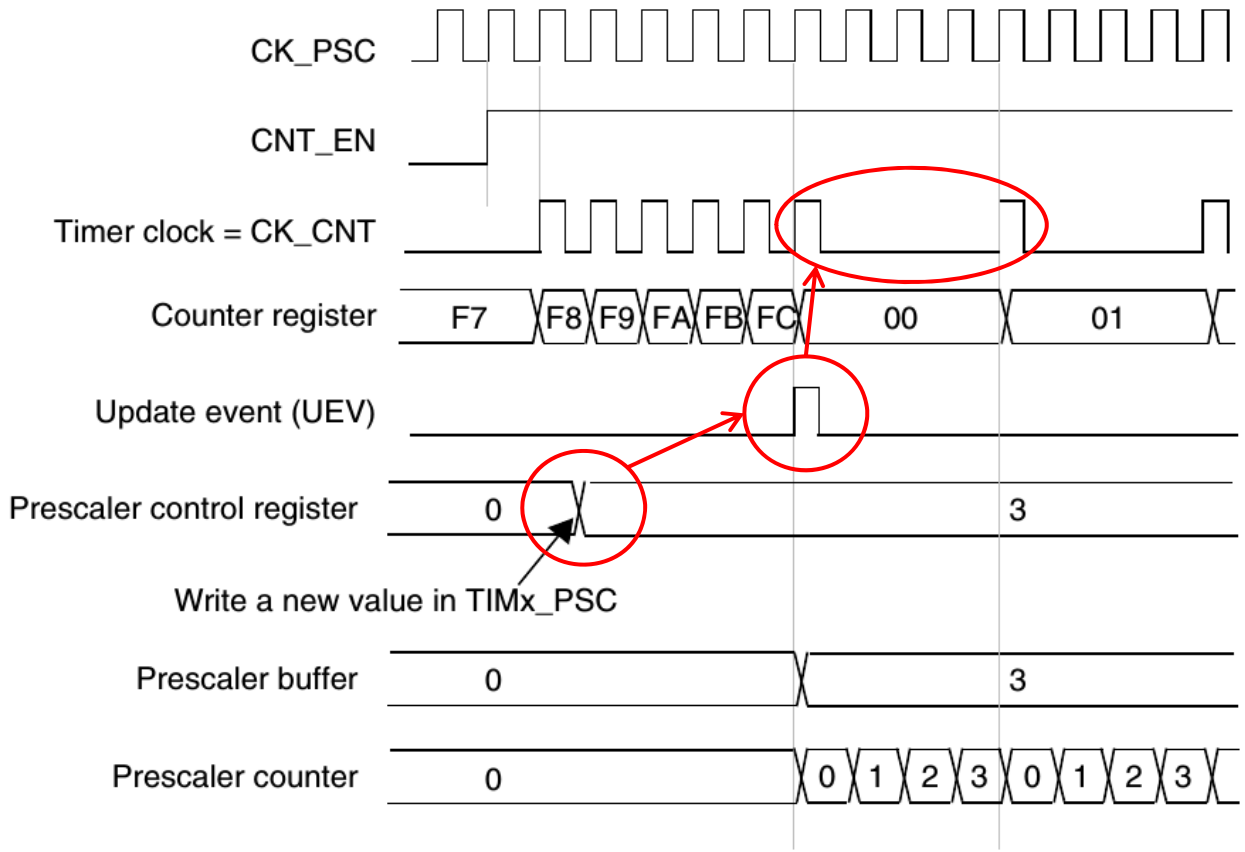
  - Divides counter clock frequency by a factor between 1 and 65536
  - Clock frequency CK_CNT equal to f_{CK_PSC} / (PSC[15:0] + 1)
  - TIMx_PSC can be changed on the fly
    - Reason: TIMx_PSC is buffered → see next slide



Timer Configuration

- **Prescaler**
  - Example: Changing prescaler division from 1 to 4

Example
TIMx_ARR=0x00FC



Timer Configuration

School of Engineering
InES Institute of Embedded Systems

- **TIMx Control Register 1 (TIMx_CR1)**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	CMS		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

  - CMS Center-aligned mode selection
    - 00 count up or down depending on DIR
    - others center-aligned
  - DIR Direction
    - 0 up-counter
    - 1 down-counter
  - CEN Counter enable

  - *Other settings for advanced use -> keep at default values*

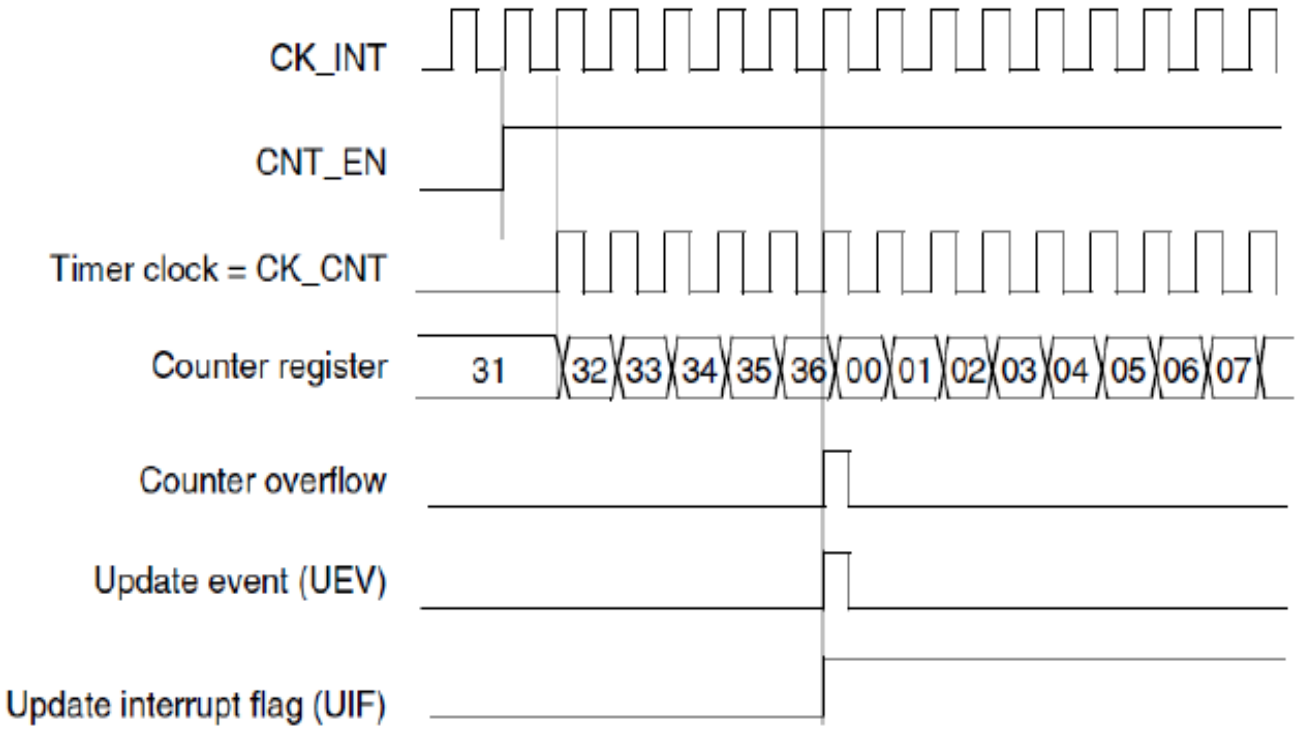
Reference Manual

Timer Configuration

- **Up-counting mode**
  - Counting from 0 to auto-reload value (TIMx_ARR)
  - Generates a counter overflow event
  - Restarts from 0

Example
TIMx_ARR=0x0036

Division by 1

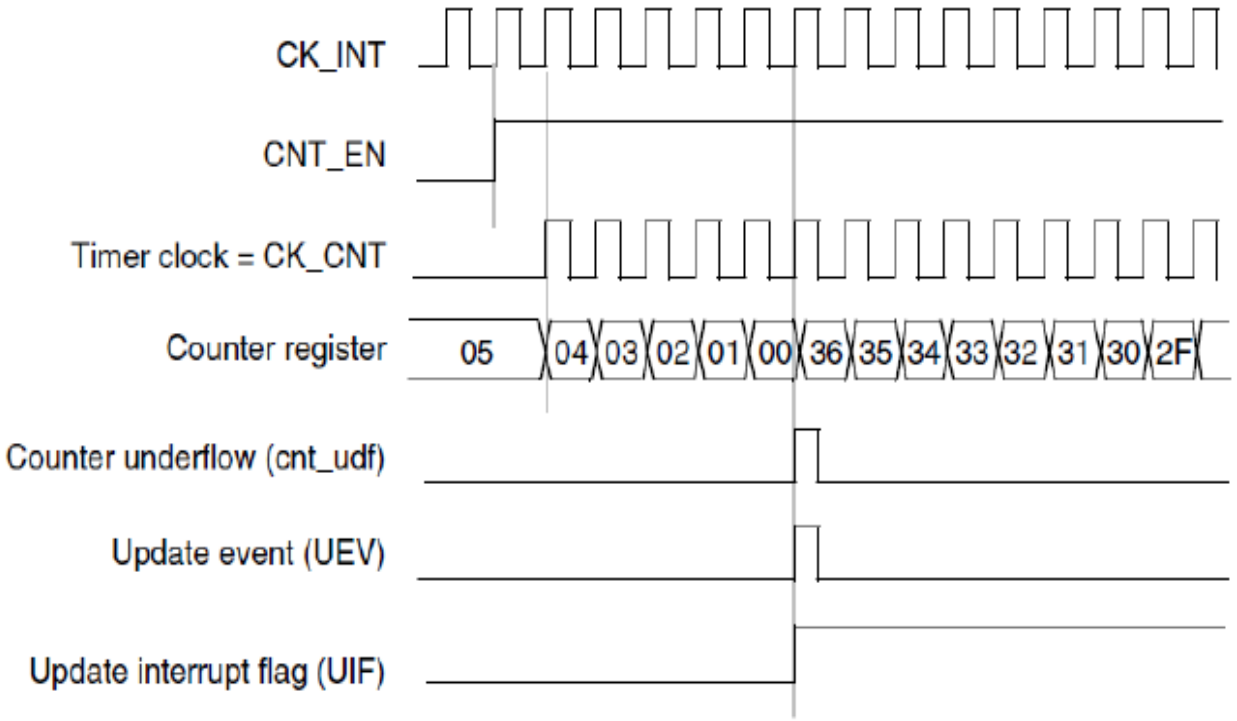


Timer Configuration

- **Down-counting mode**
  - Counting from auto-reload value (TIMx_ARR) down to 0
  - Generates a counter underflow event
  - Restarts from auto-reload value

Example
TIMx_ARR=0x0036

Division by 1

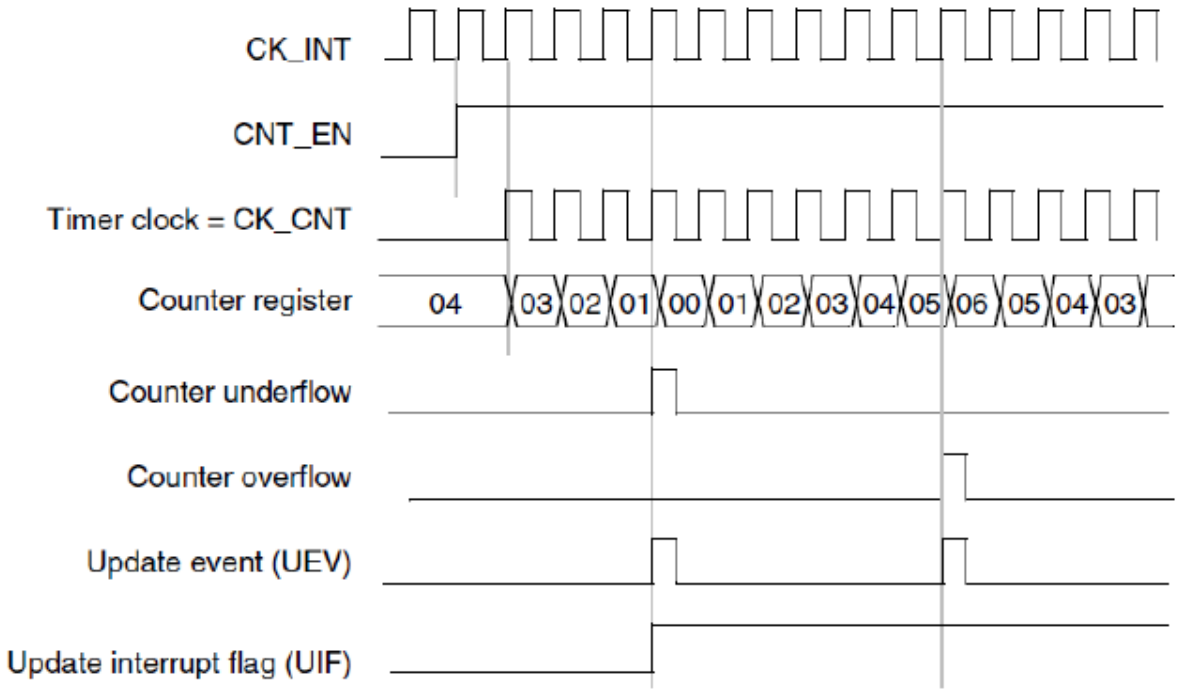


Timer Configuration

- **Center-aligned mode (up/down counting)**
  - Counts from 0 to auto-reload value (TIMx_ARR) – 1
  - Generates a counter overflow event
  - Counts from auto-reload value down to 1
  - Generates a counter underflow event
  - Restarts counting from 0

Example
TIMx_ARR=0x0006

Division by 1



Timer Configuration

School of Engineering
InES Institute of Embedded Systems

- **TIMx slave mode control register (TIMx_SMCR)**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

  - SMS: Master/Slave mode
    - 000: Slave Mode off -> using internal clock (CK_INT)

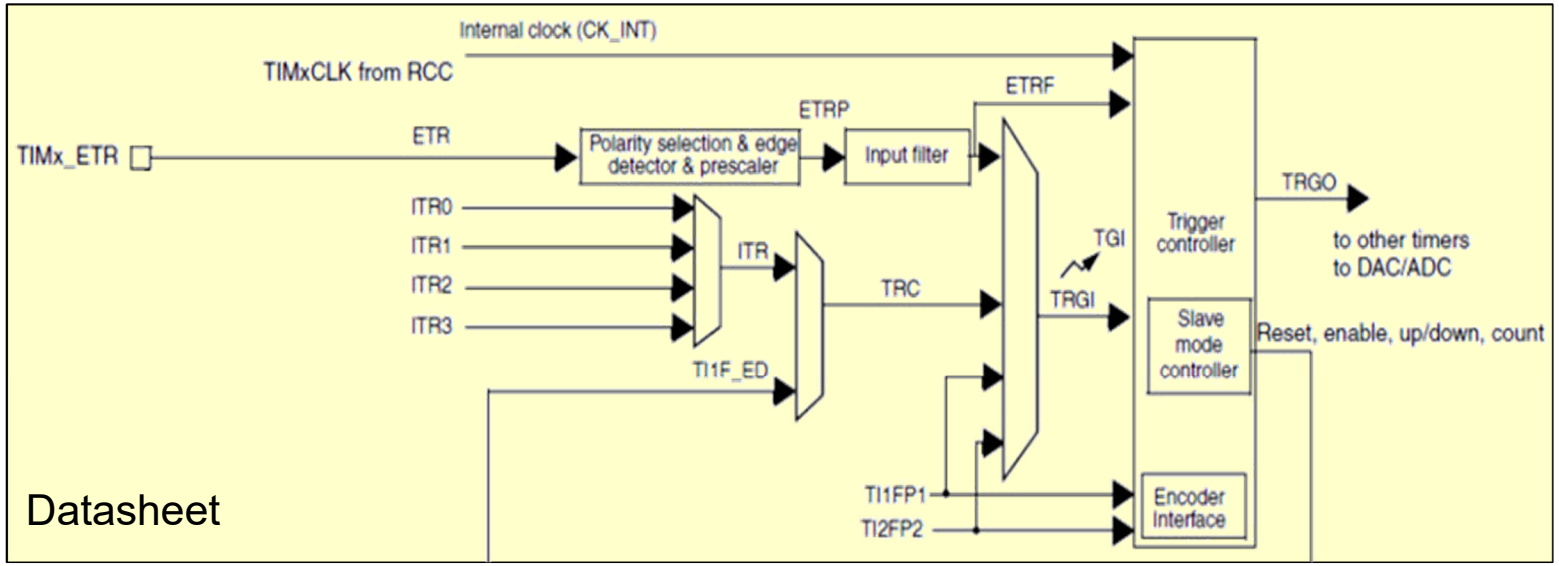
  - *Other settings for advanced use -> keep at default value*

Reference Manual

Timer Configuration

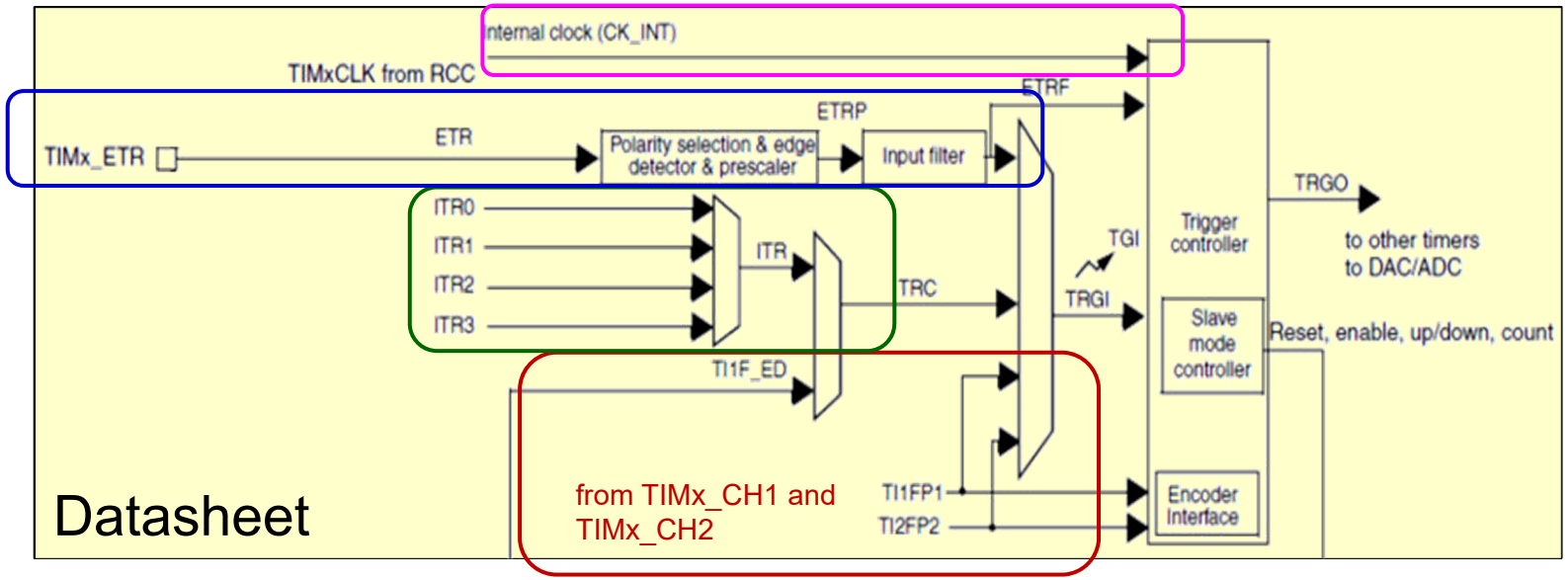
School of Engineering
InES Institute of Embedded Systems

- **Clock sources for TIM2-TIM5**
  - Internal clock (**CK_INT**)
  - External input pins (**TIMx_CH1** and **TIMx_CH2**)
  - External trigger input (**TIMx_ETR**)
  - Internal trigger inputs (**ITRx**)
    - Using one timer as prescaler for another timer



Timer Configuration

- **Clock sources for TIM2-TIM5**
  - Internal clock (**CK_INT**)
  - External input pins (**TIMx_CH1** and **TIMx_CH2**)
  - External trigger input (**TIMx_ETR**)
  - Internal trigger inputs (**ITRx**)
    - Using one timer as prescaler for another timer



Timer Configuration

School of Engineering
InES Institute of Embedded Systems

- **TIMx DMA/Interrupt Enable Register (TIMx_DIER)**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	Res	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

- UIE Update Interrupt enable
- CCxIE Capture Compare Interrupt Enable

} **Covered later**

- *Other settings for advanced use -> keep at default value*

Reference Manual

Timer Status

School of Engineering
InES Institute of Embedded Systems

TIMx Status Register (TIMx_SR)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CC4OF	CC3OF	CC2OF	CC1OF	Reserved		TIF	Res	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

- UIF – Update Interrupt Flag
  - Set by hardware on update event
  - Cleared by software

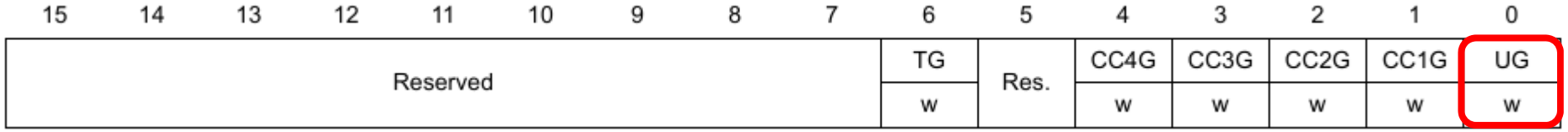
- CCxIF
  - Output: Set by hardware if CNT == CCR
  - Input: Set by hardware on capture
  - Cleared by software or by reading CCR register

Covered later

Timer Configuration

School of Engineering
InES Institute of Embedded Systems

- **TIMx Event Generation Register (TIMx_EGR)**



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

- UG: Update generation
  - Re-initializes the counter and generates an update of the registers

    0: No action / 1: action is taken
    Bits set by software in order to generate an event
    Bits automatically cleared by hardware

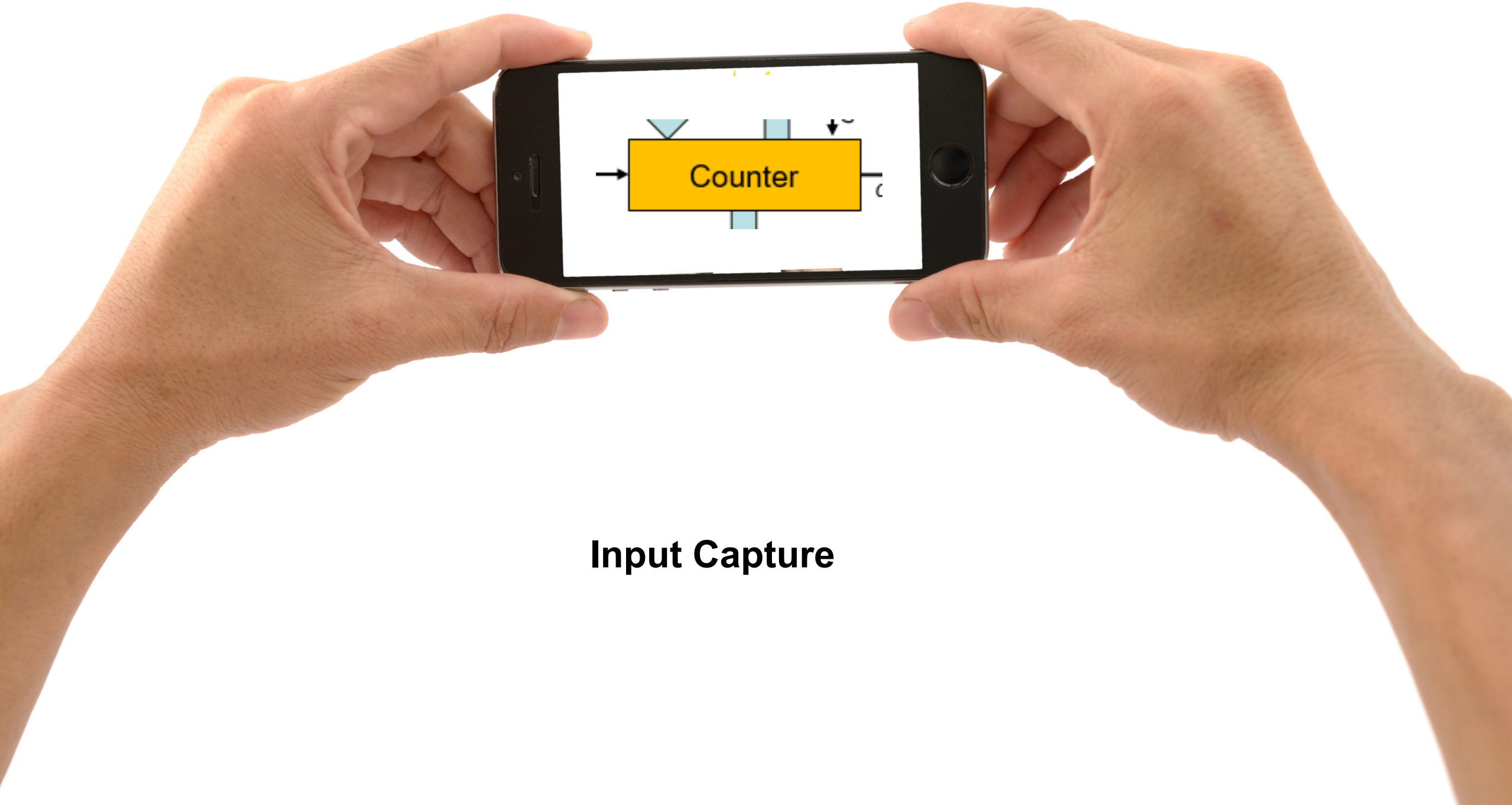
- *Other settings for advanced use -> keep at default value*

Reference Manual

Exercise

School of Engineering
InES Institute of Embedded Systems

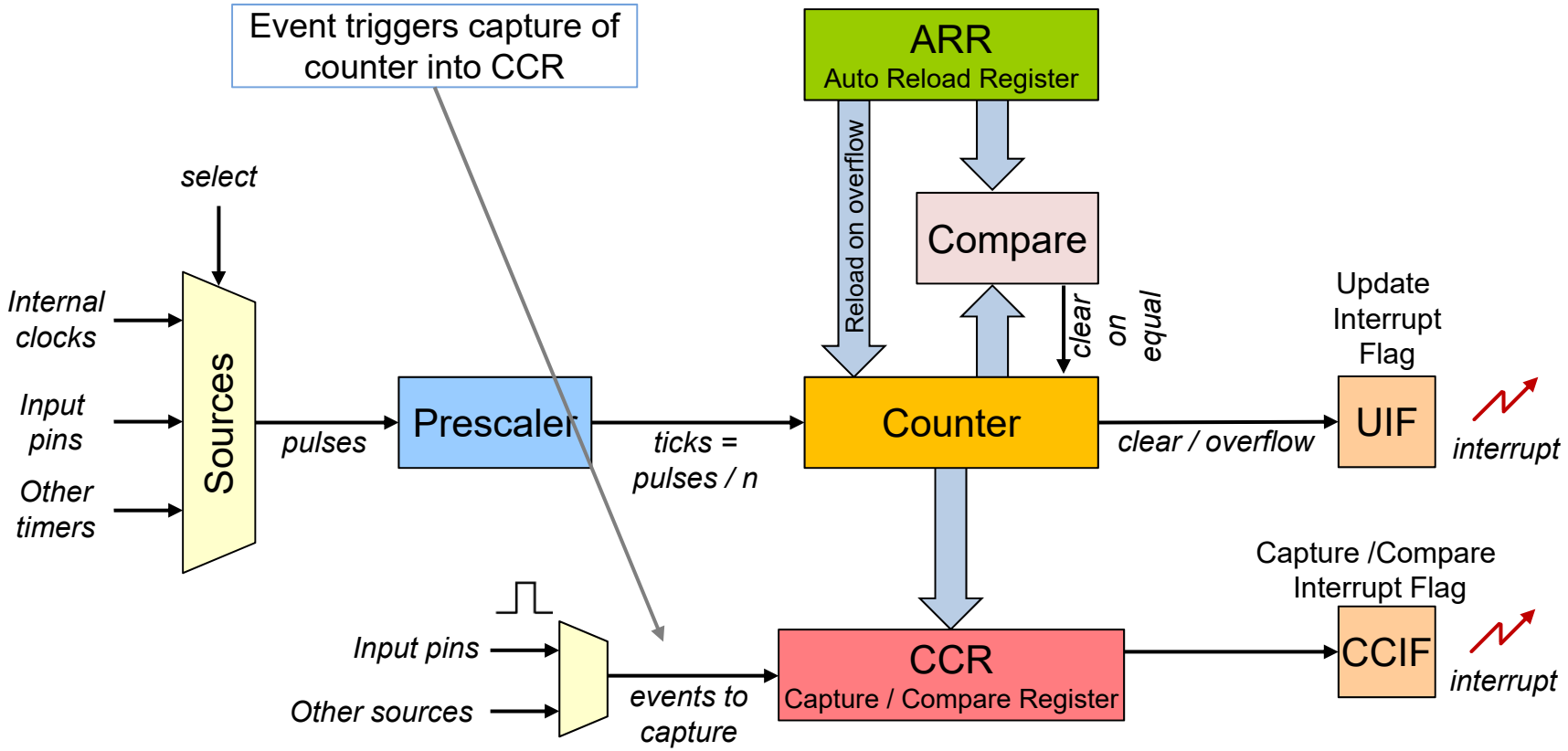
- **Given**
  - CK_INT is already configured to 84 MHz
- **Task**
  - Generate an interrupt every 1 s
  - Use Timer 3 (16-bit) in up-counting mode
- **Wanted**
  - Names and addresses of configuration registers
  - Settings for configuration registers



Input Capture

Input Capture

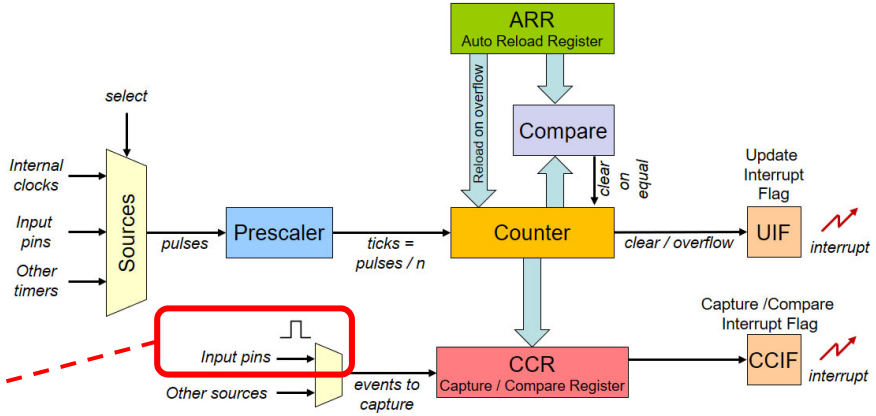
- **Measuring intervals → pulse lengths and periods**
  - Count ticks between timer start and an event

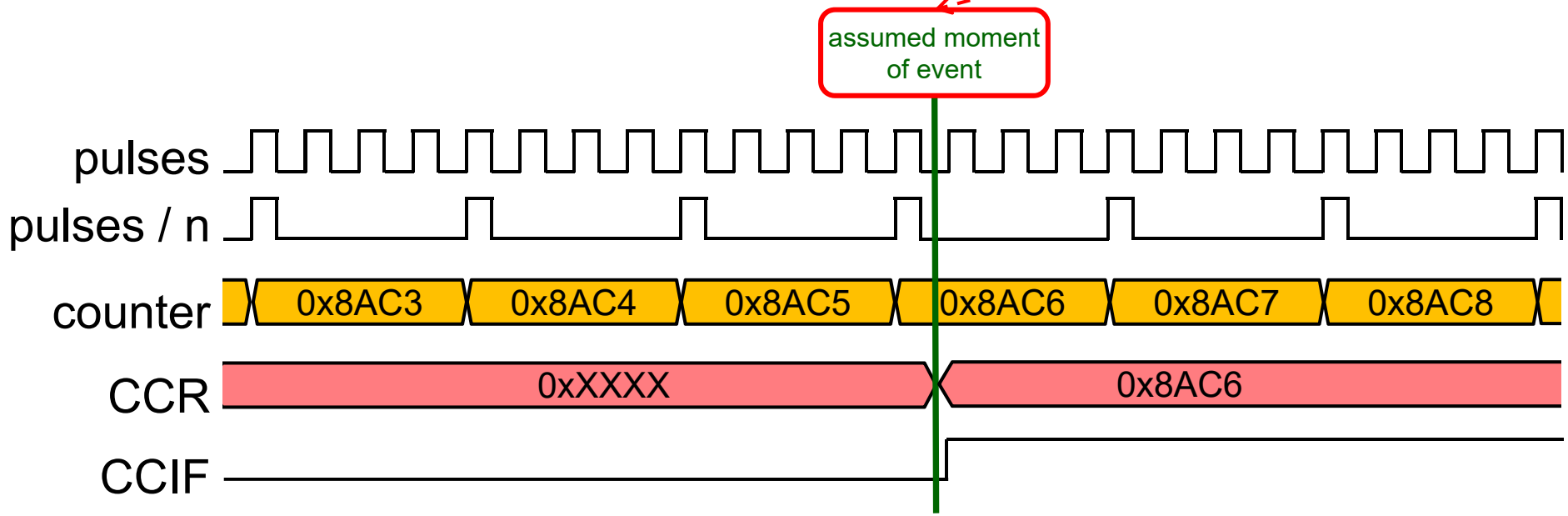


Input Capture

School of Engineering
InES Institute of Embedded Systems

- **Capture example**
  - Stop watch
  - At which moment in time does the user push the button?
    - Event = rising edge on input pin
    - Time of event is captured
    - Count continues

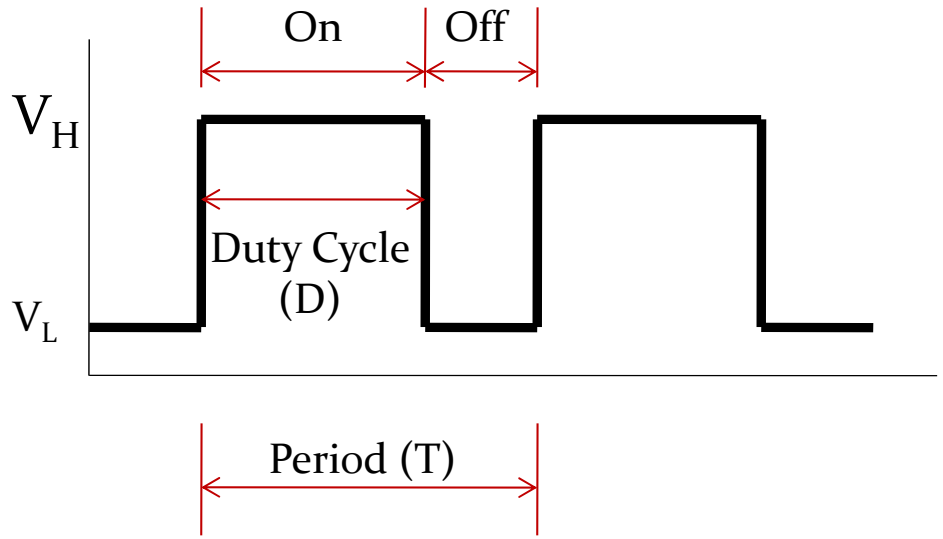




Pulse Width Modulation (PWM)

Pulse-Width-Modulation (PWM)

- **Duty Cycle – Definition**



$$Duty\ Cycle = \frac{On\ Time}{Period} \times 100\%$$

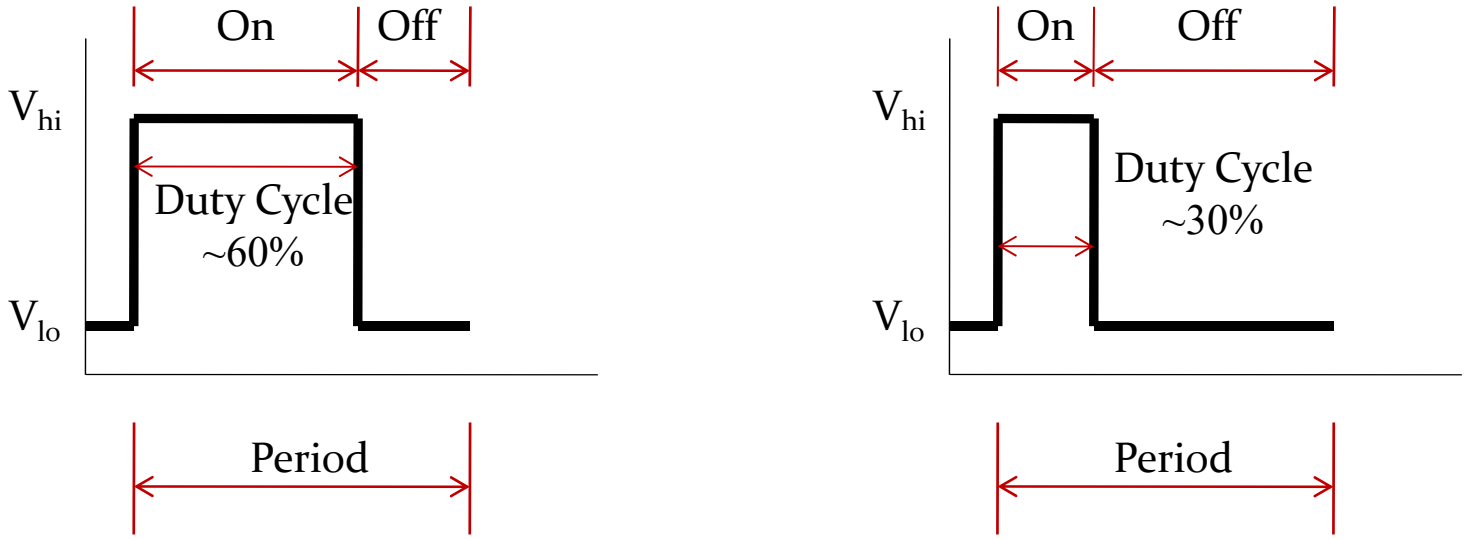
Average signal

$$V_{avg} = D \cdot V_H + (1 - D) \cdot V_L$$

Usually, V_L is taken as zero volts for simplicity.

Pulse-Width-Modulation (PWM)

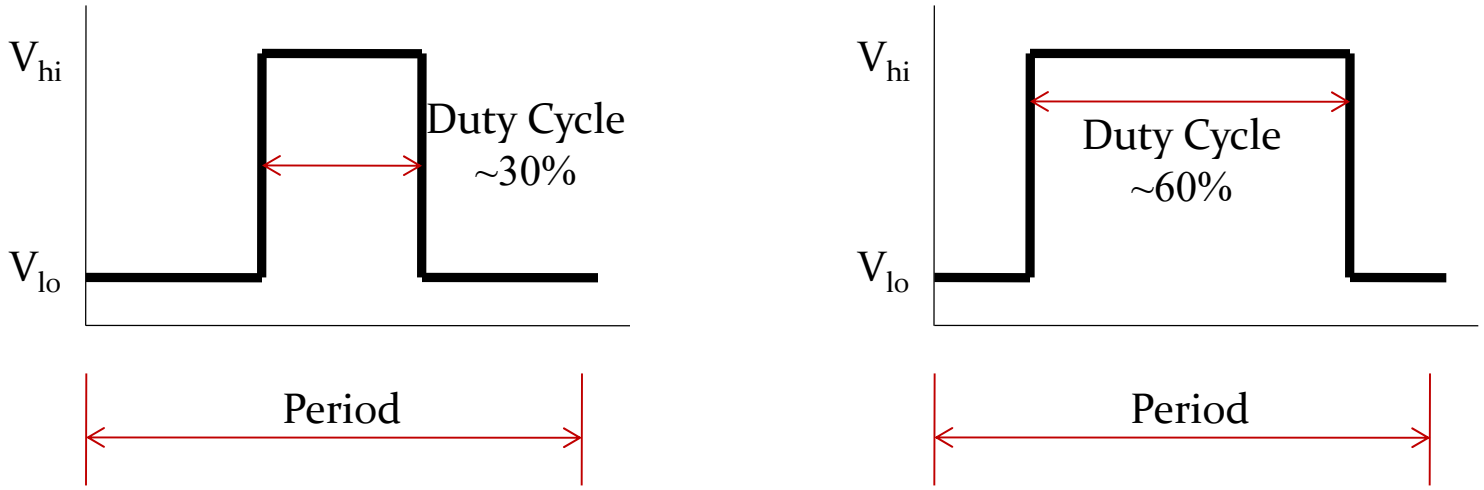
- **Types of PWM – Left Aligned**
  - Left edge is fixed, the trailing edge is modulated.



source: Zak Ahmad

Pulse-Width-Modulation (PWM)

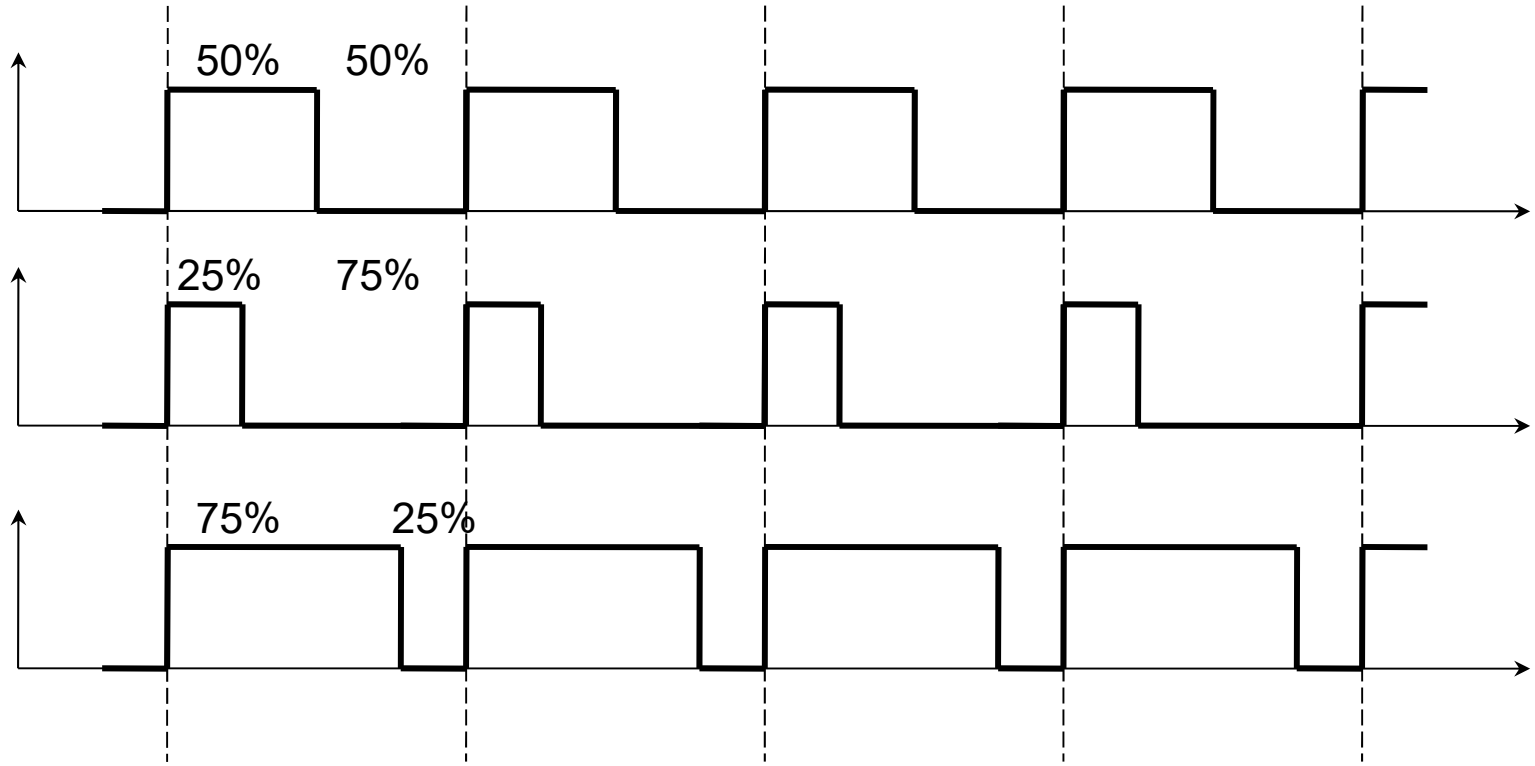
- **Types of PWM – Center Aligned**
  - Center of signal is fixed, both edges are modulated



source: Zak Ahmad

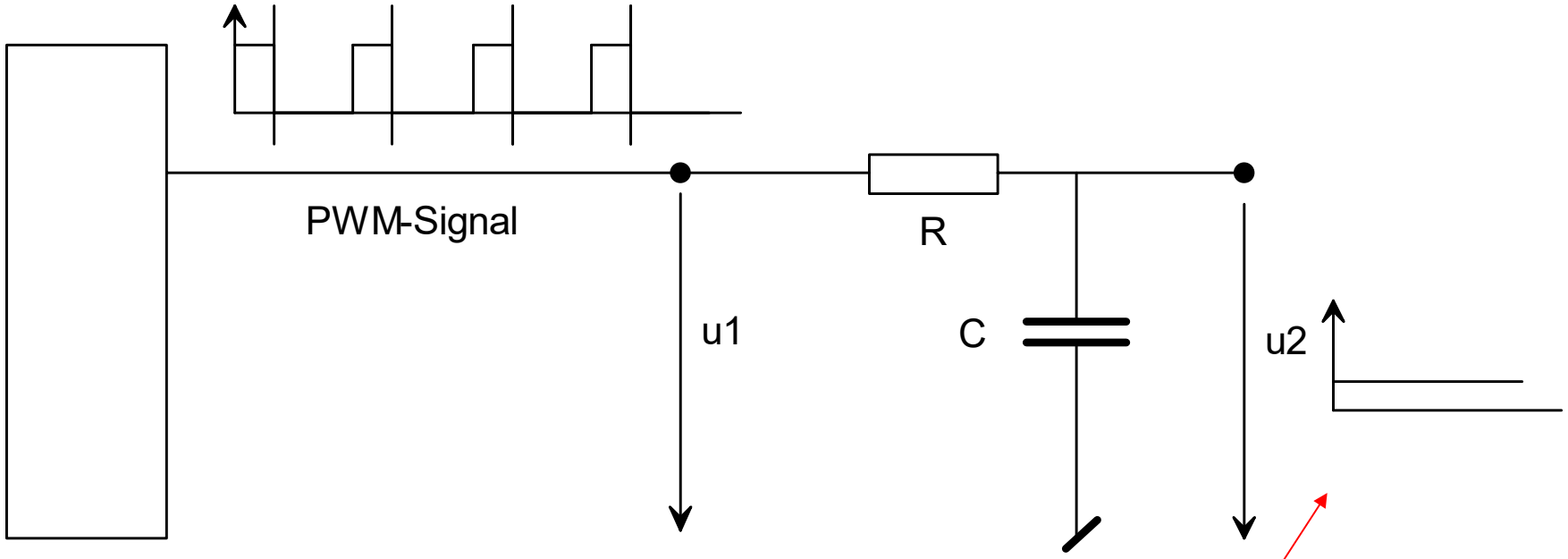
Pulse-Width-Modulation (PWM)

- **PWM-Signals are digital signals (0/1) with a defined frequency and variable pulse width**



Pulse-Width-Modulation (PWM)

- **Application**
  - Dimming LED with variable on / off
  - Digital/Analog-Converters (DAC)



Voltage is proportional to duty cycle of PWM-Signal

Pulse-Width-Modulation (PWM)

School of Engineering
InES Institute of Embedded Systems

- **Sine wave approximation**

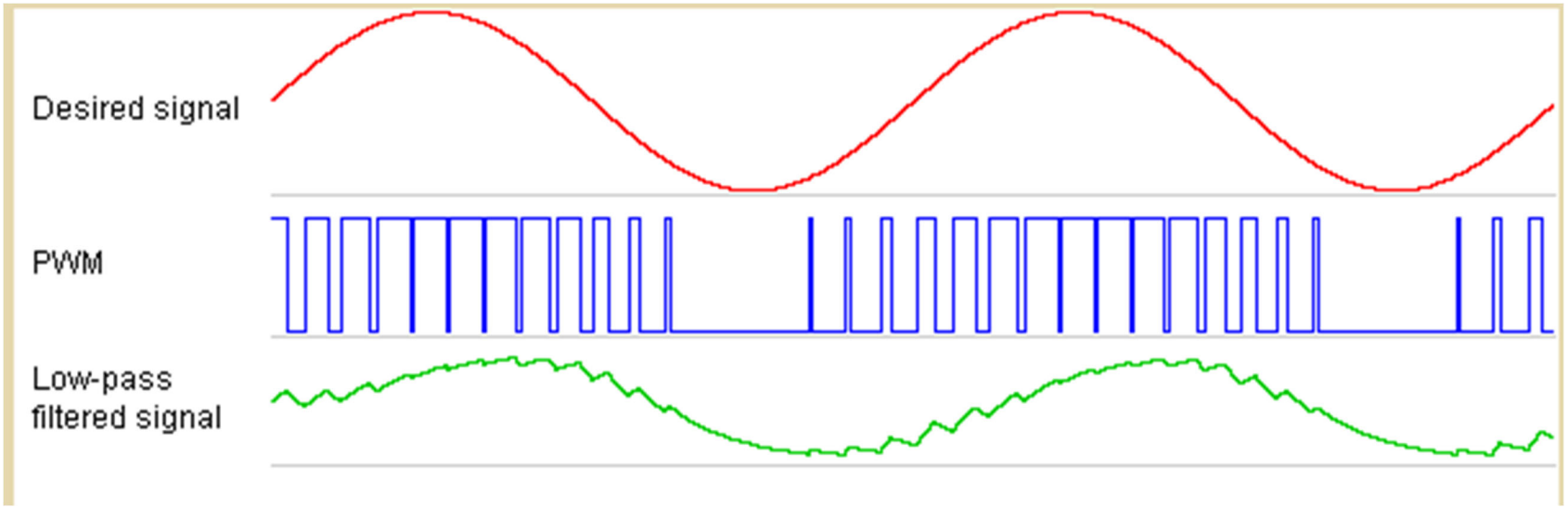
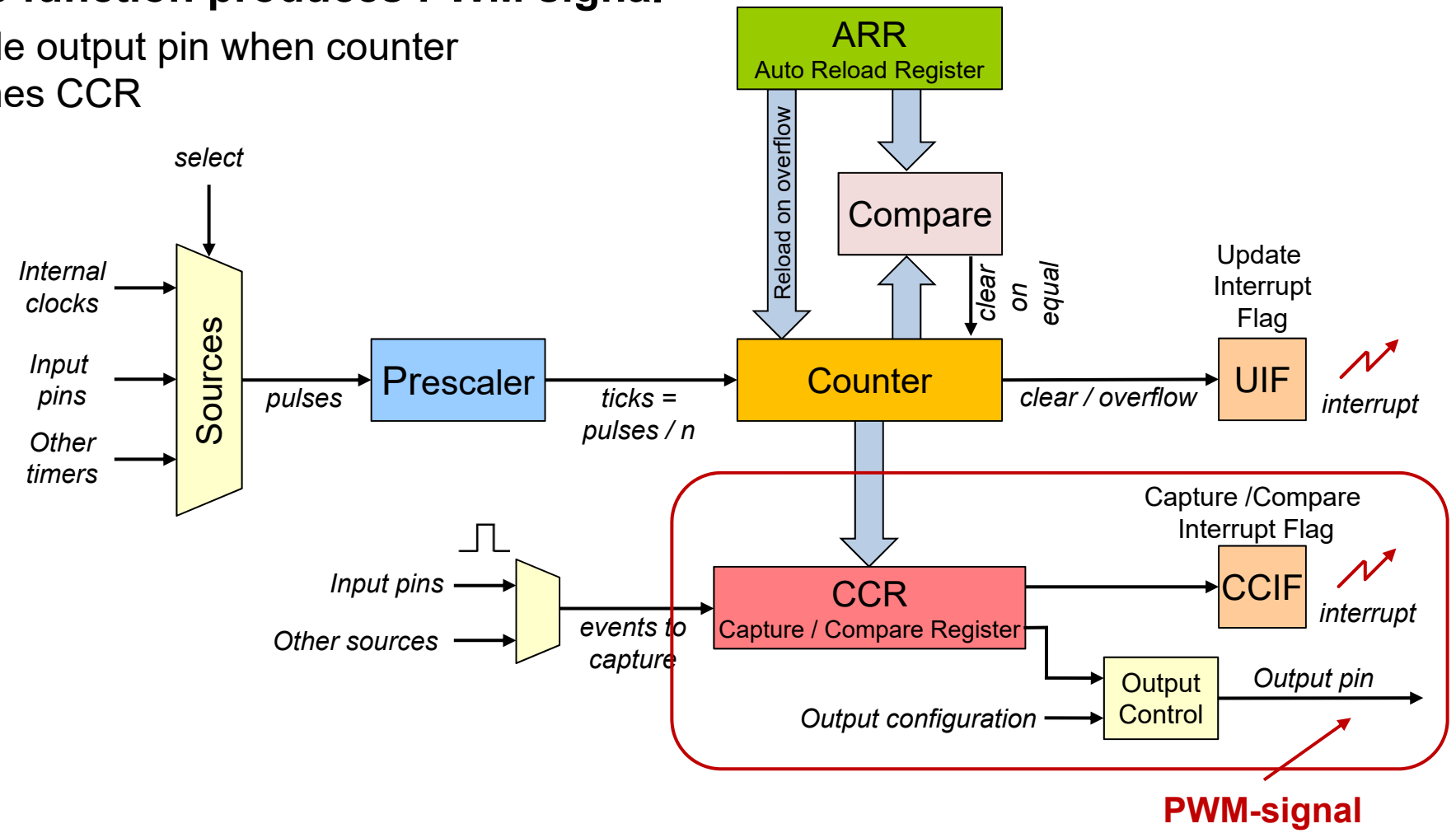


Image: Zak Ahmad

Output Compare – Generating PWM Signals

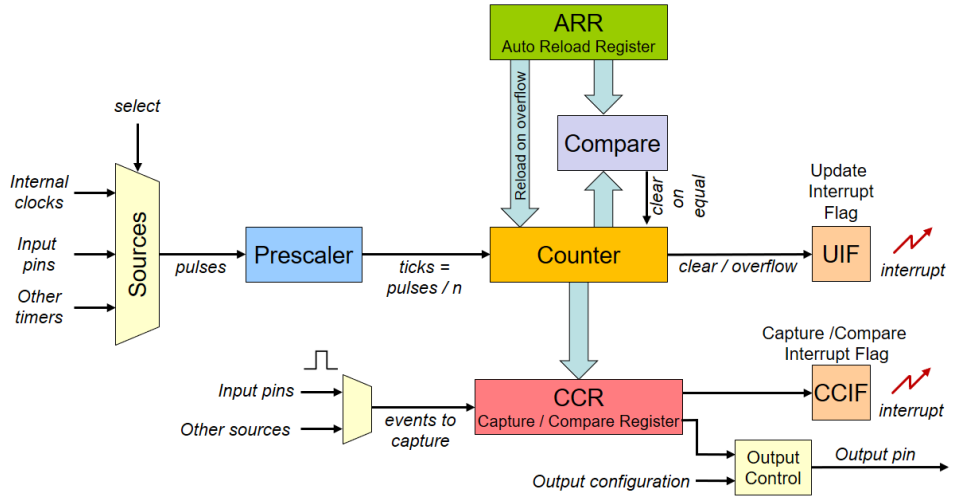
- **Compare function produces PWM signal**
  - Toggle output pin when counter reaches CCR

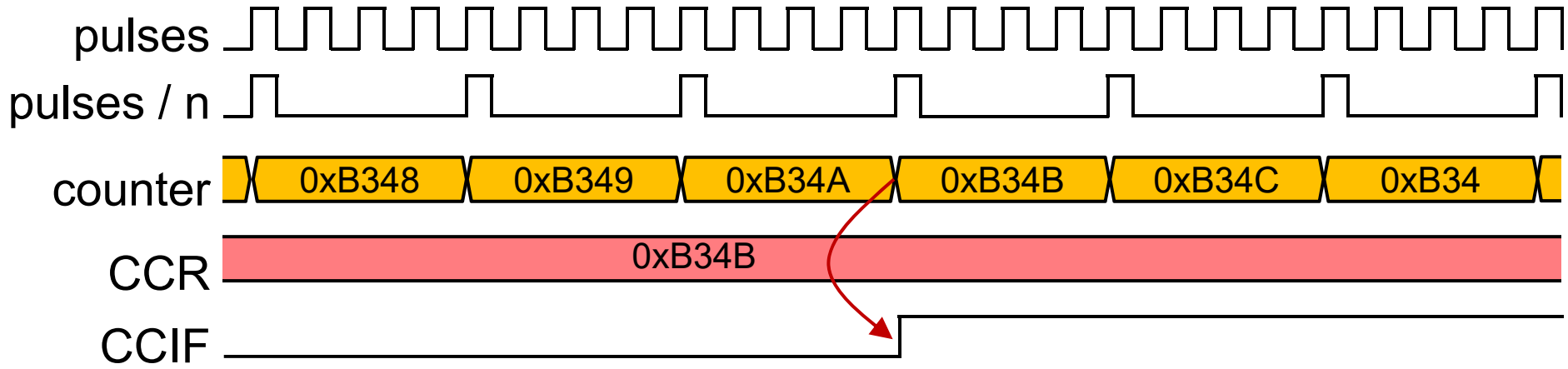


Output Compare – Generating PWM Signals

School of Engineering
InES Institute of Embedded Systems

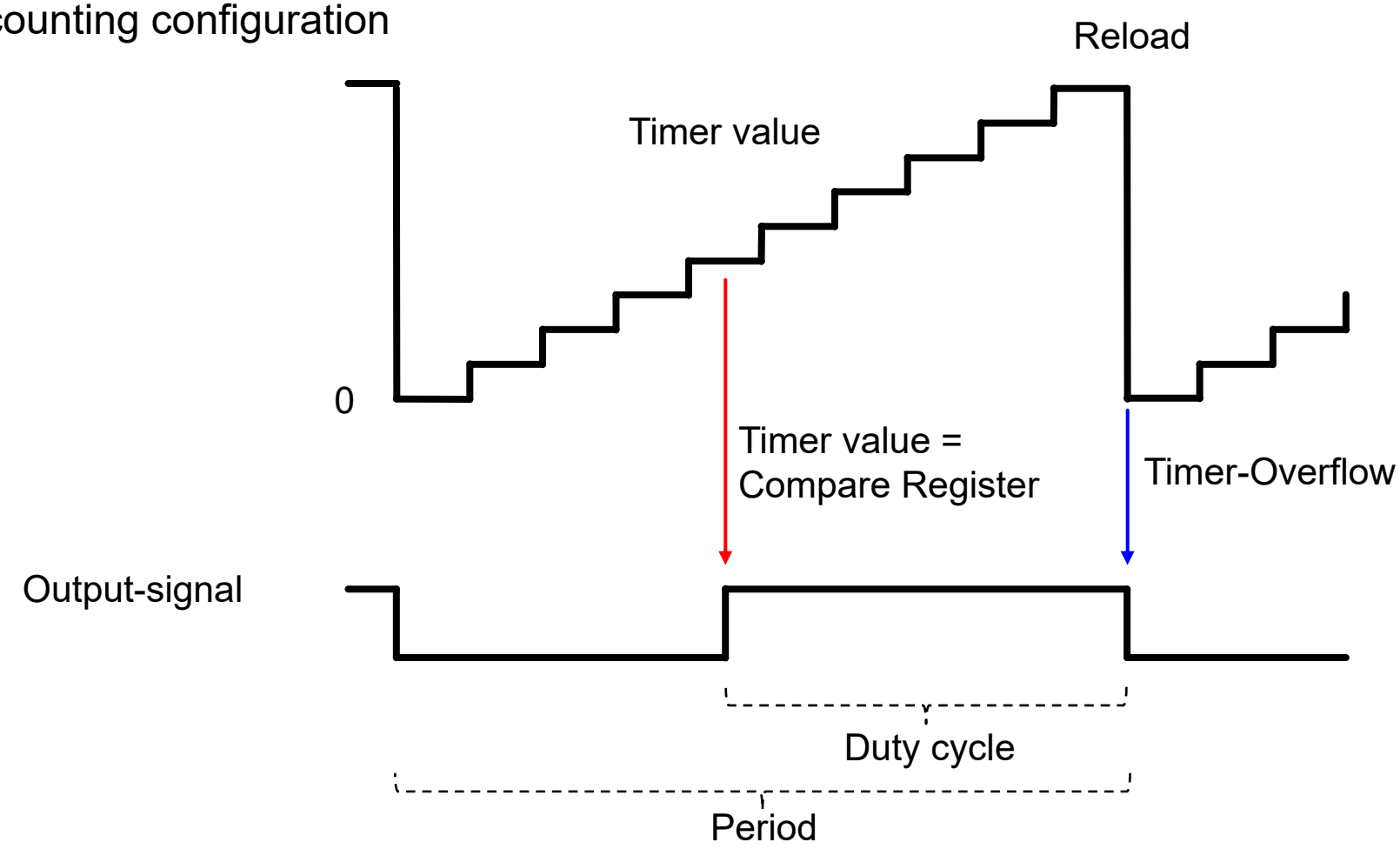
- **Compare example**
  - Raise an alarm when specified count is reached or exceeded
  - Continuously compare counter value to a reference value





Output Compare – Generating PWM Signals

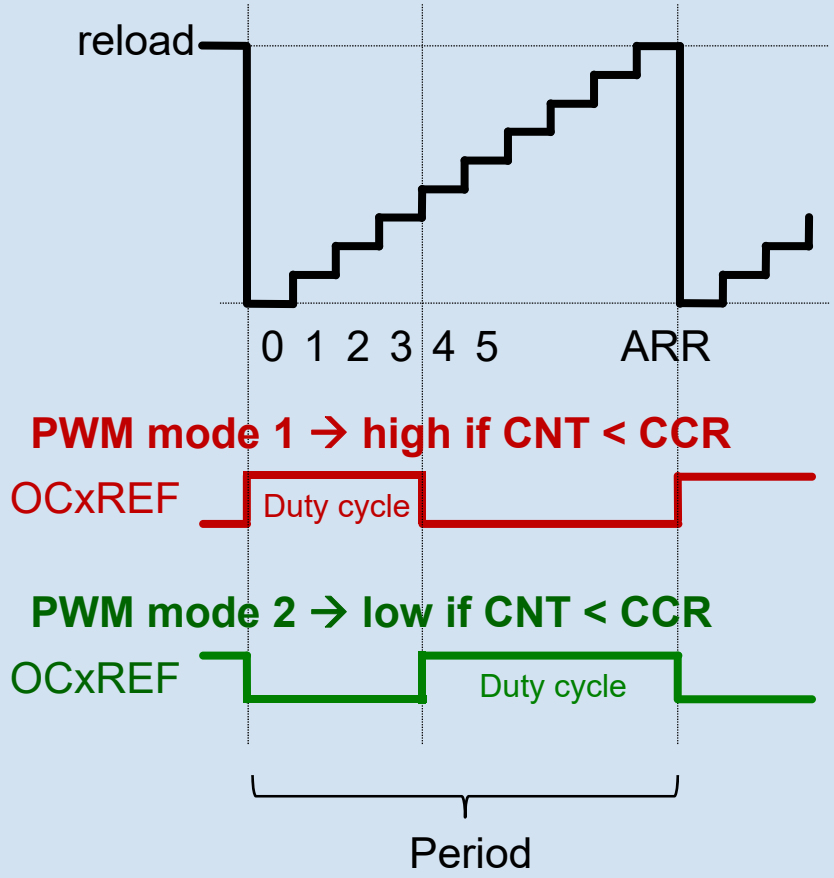
- **Edge-aligned mode**
  - Up-counting configuration



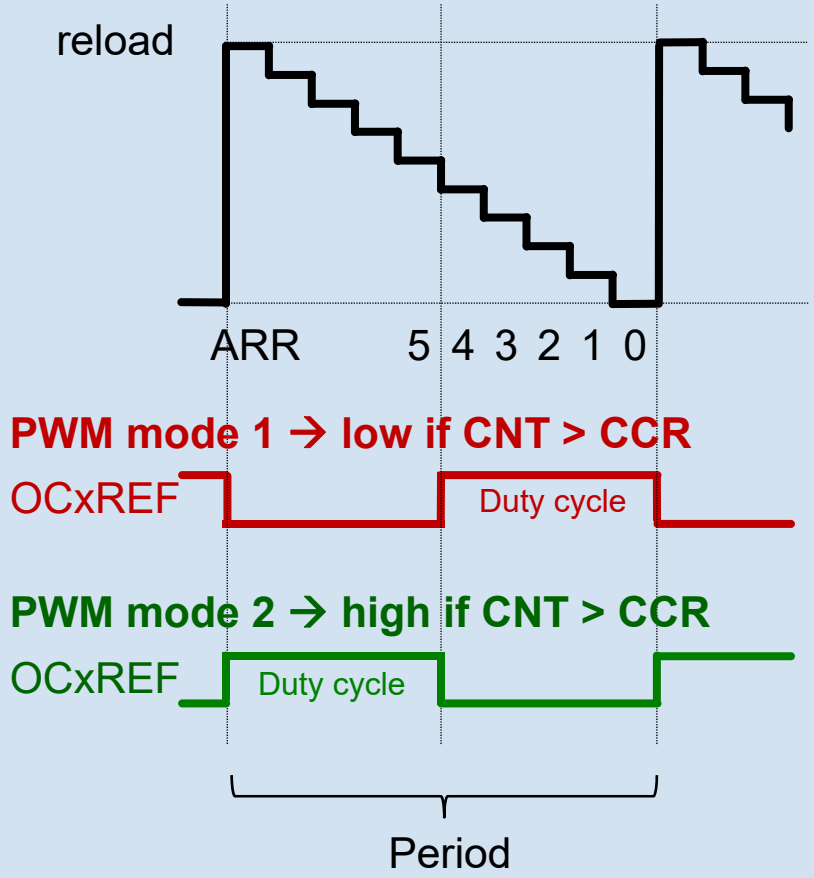
Output Compare – Generating PWM Signals

assuming Capture Compare Register (CCR) = 4

Up-counting

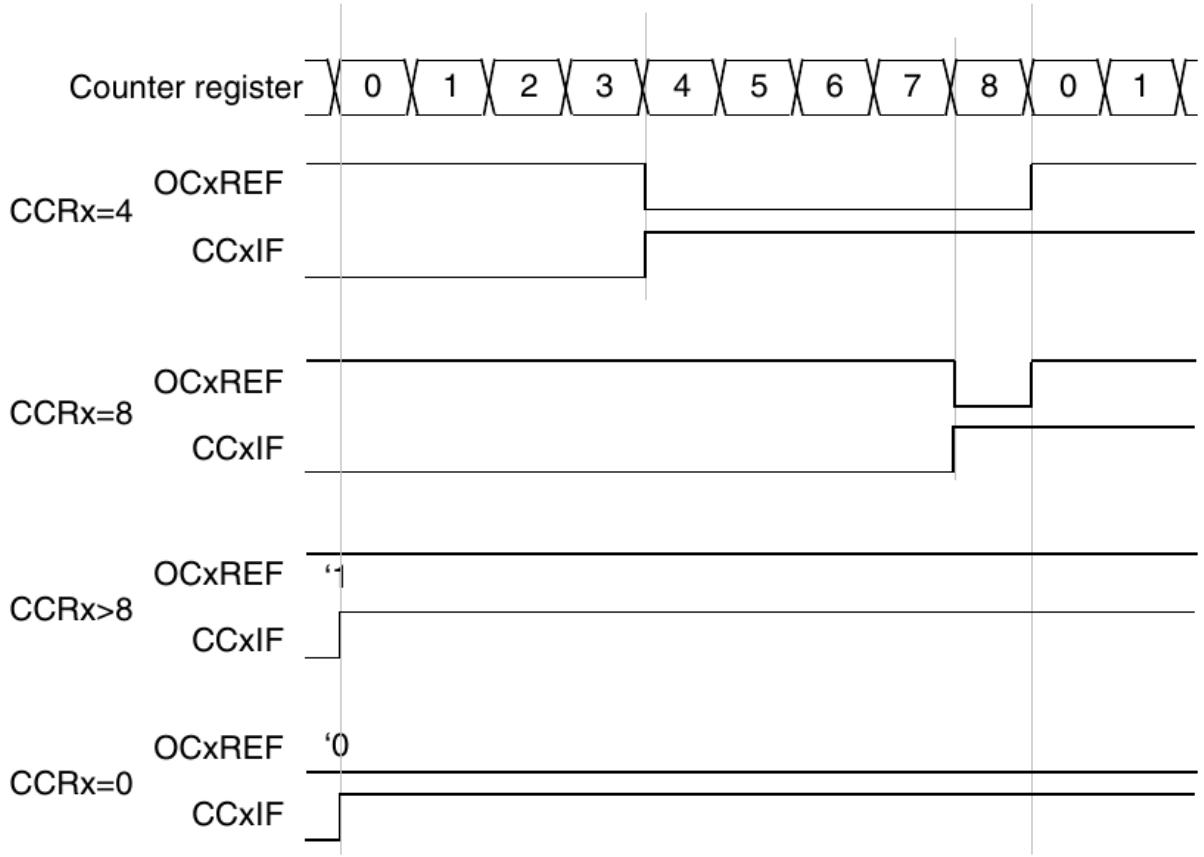


Down-counting



Output Compare – Generating PWM Signals

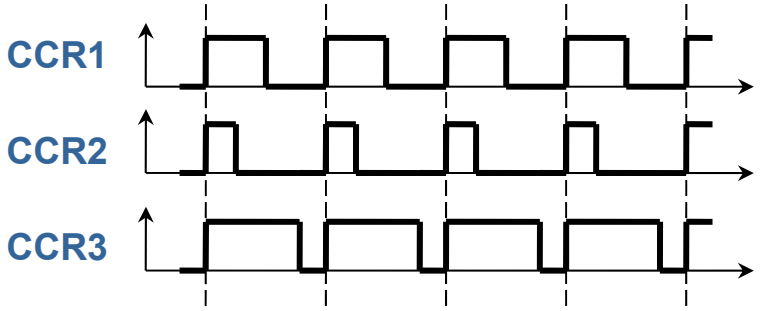
- **Edge-aligned mode**
  - Up-counting configuration → 4 examples for different CCR values
  - TIMx_ARR = 8
  - PWM mode 1



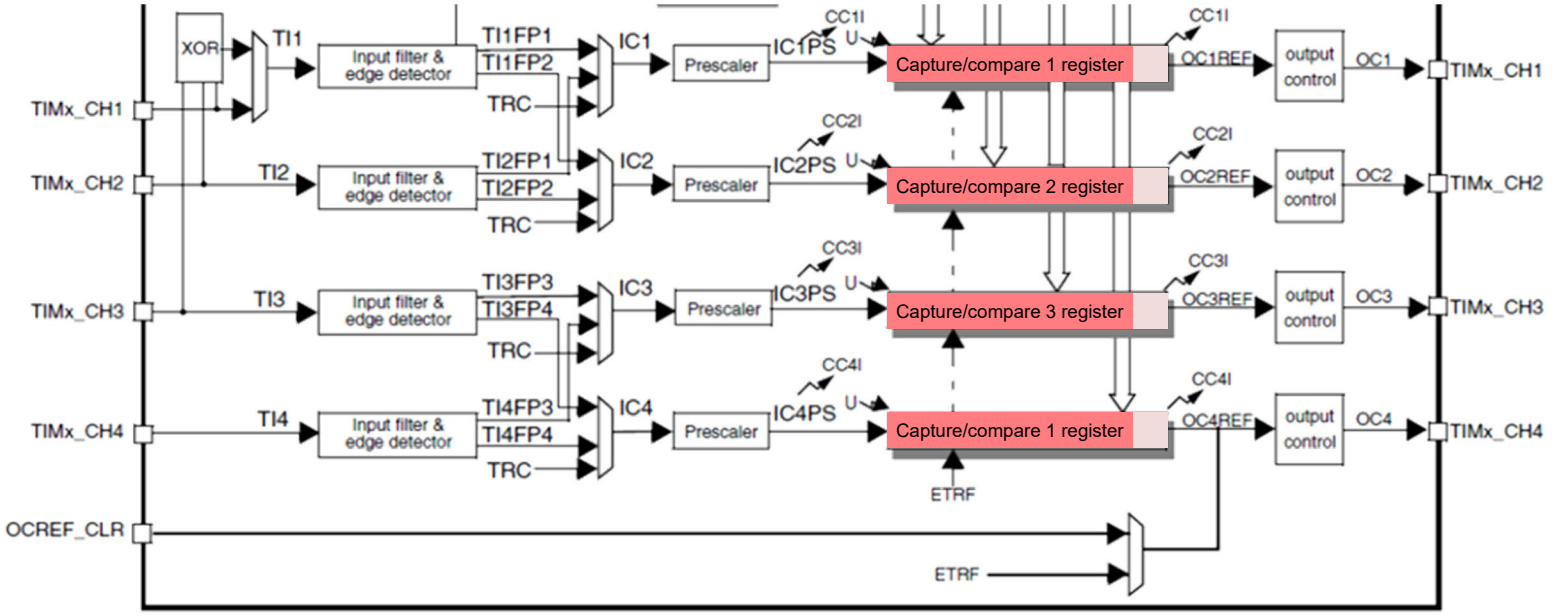
Input Capture / Output Compare

- **4 independent channels for**
  - Input capture
  - Output compare
  - PWM generation
  - One-pulse mode output

Aligned PWM signals with same period but different duty cycles



Single timer defines common period



Capture / Compare Configuration

School of Engineering
InES Institute of Embedded Systems

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x18	TIMx_CCMR1 Output Compare mode	Reserved																OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1 Input Capture mode	Reserved																IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	TIMx_CCMR2 Output Compare mode	Reserved																O24CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2 Input Capture mode																	IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	TIMx_CCER	**Output enable of CC ⟶**																CC4NP	Reserved	CC4P	CC4E	CC3NP	Reserved	CC3P	CC3E	CC2NP	Reserved	CC2P	CC2E	CC1NP	Reserved	CC1P	CC1E
	Reset value																	0		0	0	0		0	0	0		0	0	0		0	0
0x34	TIMx_CCR1	CCR1[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR1[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	TIMx_CCR2	CCR2[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR2[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	TIMx_CCR3	CCR3[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR3[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	TIMx_CCR4	CCR4[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR4[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

```
#define TIM2 ( (reg_tim_t *) 0x40000000 )
#define TIM3 ( (reg_tim_t *) 0x40000400 )
#define TIM4 ( (reg_tim_t *) 0x40000800 )
#define TIM5 ( (reg_tim_t *) 0x40000c00 )
```

```
typedef struct {
    volatile uint32_t CR1;
    volatile uint32_t CR2;
    volatile uint32_t SMCR;
    volatile uint32_t DIER;
    volatile uint32_t SR;
    volatile uint32_t EGR;
    volatile uint32_t CCMR1;
    volatile uint32_t CCMR2;
    volatile uint32_t CCER;
    volatile uint32_t CNT;
    volatile uint32_t PSC;
    volatile uint32_t ARR;
    volatile uint32_t RCR;
    volatile uint32_t CCR1;
    volatile uint32_t CCR2;
    volatile uint32_t CCR3;
    volatile uint32_t CCR4;
    volatile uint32_t BDTR;
    volatile uint32_t DCR;
    volatile uint32_t DMAR;
    volatile uint32_t OR;
} reg_tim_t;
```

Capture / Compare Configuration

School of Engineering
InES Institute of Embedded Systems

- **TIMx capture/compare mode register 1 (TIMx_CCMR1)**
- **TIMx capture/compare mode register 2 (TIMx CCMR2)**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

- OCxM: Output compare mode
  - 110: PWM mode 1
  - 111: PWM mode 2

- *Other settings for advanced use -> keep at default value*

Reference Manual

Capture / Compare Configuration

School of Engineering
InES Institute of Embedded Systems

- **TIMx capture/compare enable register (TIMx_CCER)**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
rw		rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw

  - CCxE: Capture/Compare x output enable

  - *Other settings for advanced use -> keep at default value*

Reference Manual

Capture / Compare Configuration

School of Engineering
InES Institute of Embedded Systems

- **PWM output cookbook**
  - Select counter clock (internal, external, prescaler)
  - Write desired data to TIMx_ARR register
    → defines common period of PWM signals
  - Write desired data to TIMx_CCRx registers
    → defines duty cycles of PWM signals
  - Set **CCxIE** bits if **interrupts** are to be generated (in TIMx_DIER register)
  - Select the output mode (registers CCMRx / CCER)
  - Enable counter by setting the CEN bit in the TIMx_CR1 register

Capture / Compare Configuration

School of Engineering
InES Institute of Embedded Systems

- **Use macros and structs from “reg_stm32f4xx.h”**

Example: `TIM3->CCMR2 = 0x0000;`

```
/**
 *  \struct reg_tim_t
 *  \brief  Representation of Timer register.
 *
 *  Described in reference manual p.507ff.
 */
typedef struct {
    volatile uint32_t CR1;      /**< Configuration register 1. */
    volatile uint32_t CR2;      /**< Configuration register 2. */
    volatile uint32_t SMCR;     /**< Slave mode control register. */
    volatile uint32_t DIER;     /**< DMA/interrupt enable register. */
    volatile uint32_t SR;       /**< Status register. */
    volatile uint32_t EGR;      /**< Event generation register. */
    volatile uint32_t CCMR1;    /**< Capture/compare mode register 1. */
    volatile uint32_t CCMR2;    /**< Capture/compare mode register 2. */
    volatile uint32_t CCER;     /**< Capture/compare enable register. */
    volatile uint32_t CNT;      /**< Count register. */
    volatile uint32_t PSC;      /**< Prescaler register. */
    volatile uint32_t ARR;      /**< Auto reload register. */
    volatile uint32_t RCR;      /**< Repetition counter register. */
    volatile uint32_t CCR1;     /**< Capture/compare register 1. */
    volatile uint32_t CCR2;     /**< Capture/compare register 2. */
    volatile uint32_t CCR3;     /**< Capture/compare register 3. */
    volatile uint32_t CCR4;     /**< Capture/compare register 4. */
    volatile uint32_t BDTR;     /**< Break and dead-time register. */
    volatile uint32_t DCR;      /**< DMA control register. */
    volatile uint32_t DMAR;     /**< DMA address for full transfer. */
    volatile uint32_t OR;       /**< Option register. */
} reg_tim_t;
```

```
#define TIM2  ( (reg_tim_t *) 0x40000000 )
#define TIM3  ( (reg_tim_t *) 0x40000400 )
#define TIM4  ( (reg_tim_t *) 0x40000800 )
#define TIM5  ( (reg_tim_t *) 0x40000c00 )
```

Exercise: Capture / Compare Configuration

School of Engineering
InES Institute of Embedded Systems

- **Timer 2 already configured**
  - CK_INT is configured to 84 MHz
  - Timer 2 (see exercise “Timer”)
    - Up-counting, Period = 1s
    - TIM2_ARR = (10000 – 1)

- **Configure PWM with Capture/Compare 1**
  - Duty cycle 25%
  - PWM mode 1

ST32F4xx Timers

School of Engineering
InES Institute of Embedded Systems

- **Feature comparison**

Timer type	Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/ compare channels	Complementary output	Max interface clock (MHz)	Max timer clock (MHz) (1)
Advanced -control	TIM1, TIM8	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	Yes	90	180
General purpose	TIM2, TIM5	32-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	45	90/180
	TIM3, TIM4	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	45	90/180
	TIM9	16-bit	Up	Any integer between 1 and 65536	No	2	No	90	180
	TIM10 , TIM11	16-bit	Up	Any integer between 1 and 65536	No	1	No	90	180
	TIM12	16-bit	Up	Any integer between 1 and 65536	No	2	No	45	90/180
	TIM13 , TIM14	16-bit	Up	Any integer between 1 and 65536	No	1	No	45	90/180
Basic	TIM6, TIM7	16-bit	Up	Any integer between 1 and 65536	Yes	0	No	45	90/180

Conclusion

School of Engineering
InES Institute of Embedded Systems

- **Timer / counter functionality**
- **Realization in hardware**
- **Detailed view on Timers TIM2 – TIM5 ST32F4xx**
- **Capture / compare Unit**
- **PWM signals**
- **Programming example**

- ***Literature***
  - *STM32F4xx Reference Manual*