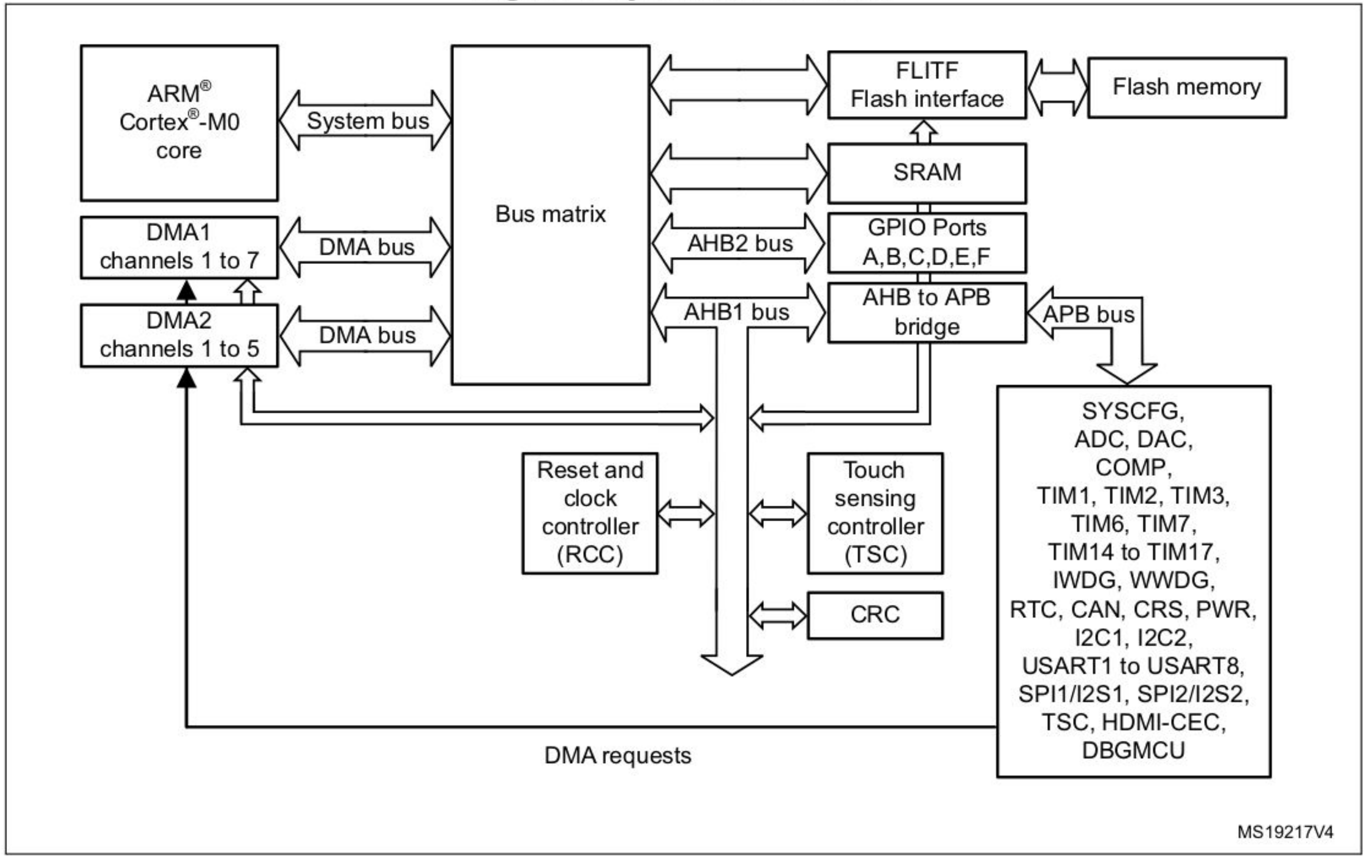
Introduction to embedded programming on STM32

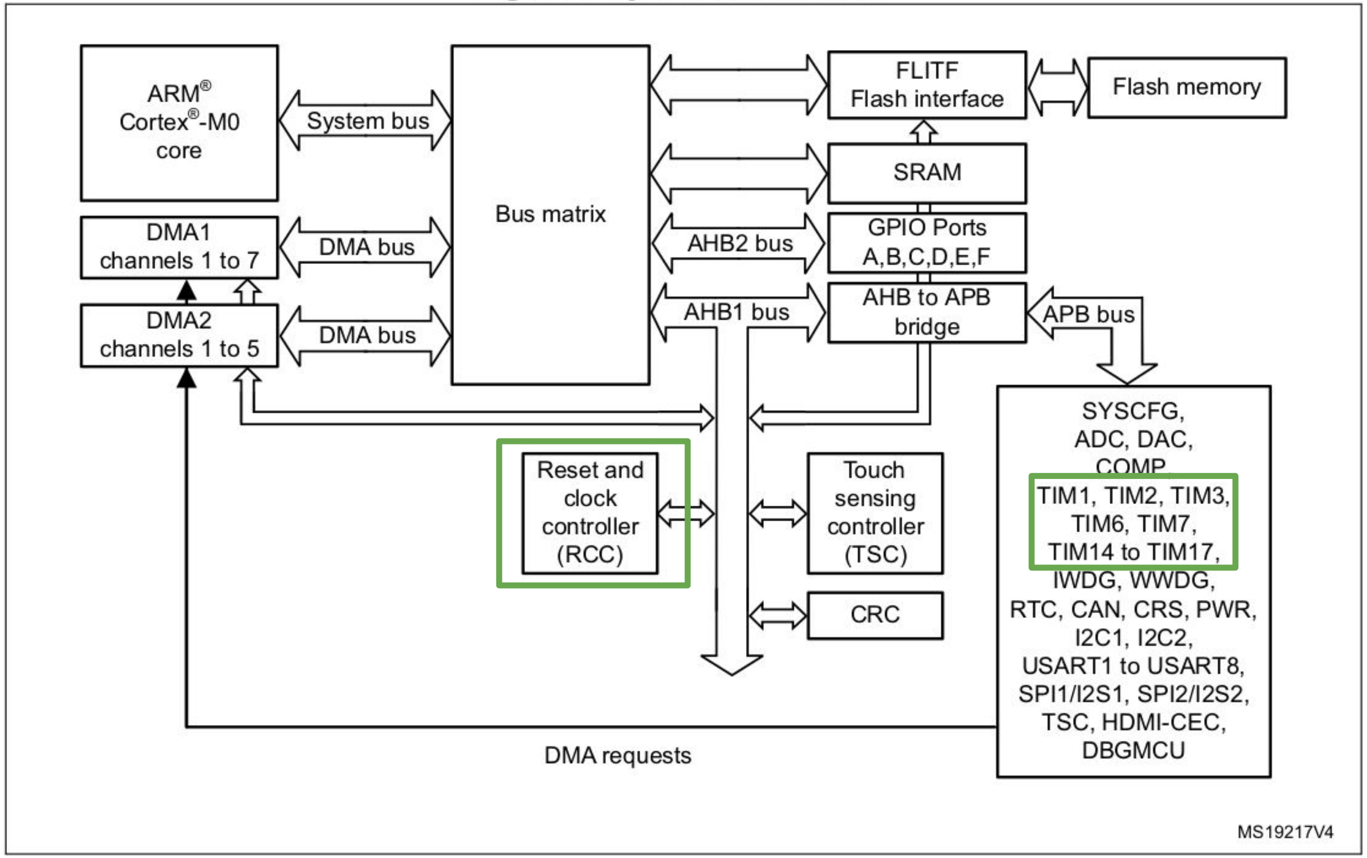
RCC, Timers

Skoltech, 2019

STM32F0 SoC overview


ARM® Cortex®-M0 core
System bus
Bus matrix
FLITF Flash interface
Flash memory
SRAM
DMA1 channels 1 to 7
DMA bus
AHB2 bus
GPIO Ports A,B,C,D,E,F
DMA2 channels 1 to 5
DMA bus
AHB1 bus
AHB to APB bridge
APB bus
SYSCFG, ADC, DAC, COMP, TIM1, TIM2, TIM3, TIM6, TIM7, TIM14 to TIM17, IWDG, WWDG, RTC, CAN, CRS, PWR, I2C1, I2C2, USART1 to USART8, SPI1/I2S1, SPI2/I2S2, TSC, HDMI-CEC, DBGMCU
Reset and clock controller (RCC)
Touch sensing controller (TSC)
CRC
DMA requests
MS19217V4

STM32F0 SoC overview



RCC (Reset and clock control). Reset

- Power reset
- System reset
- RTC domain reset

The RESET service routine vector is fixed at address 0x00000004 in the memory map

RCC (Reset and clock control). Reset

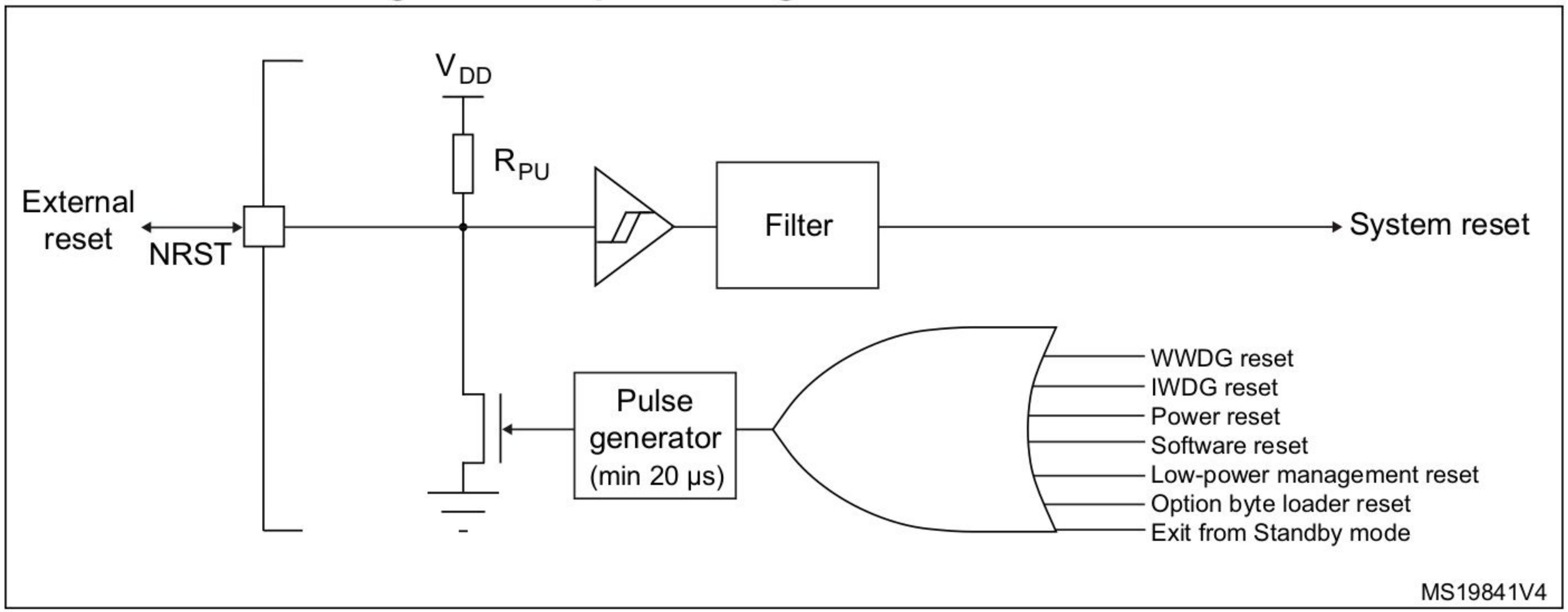
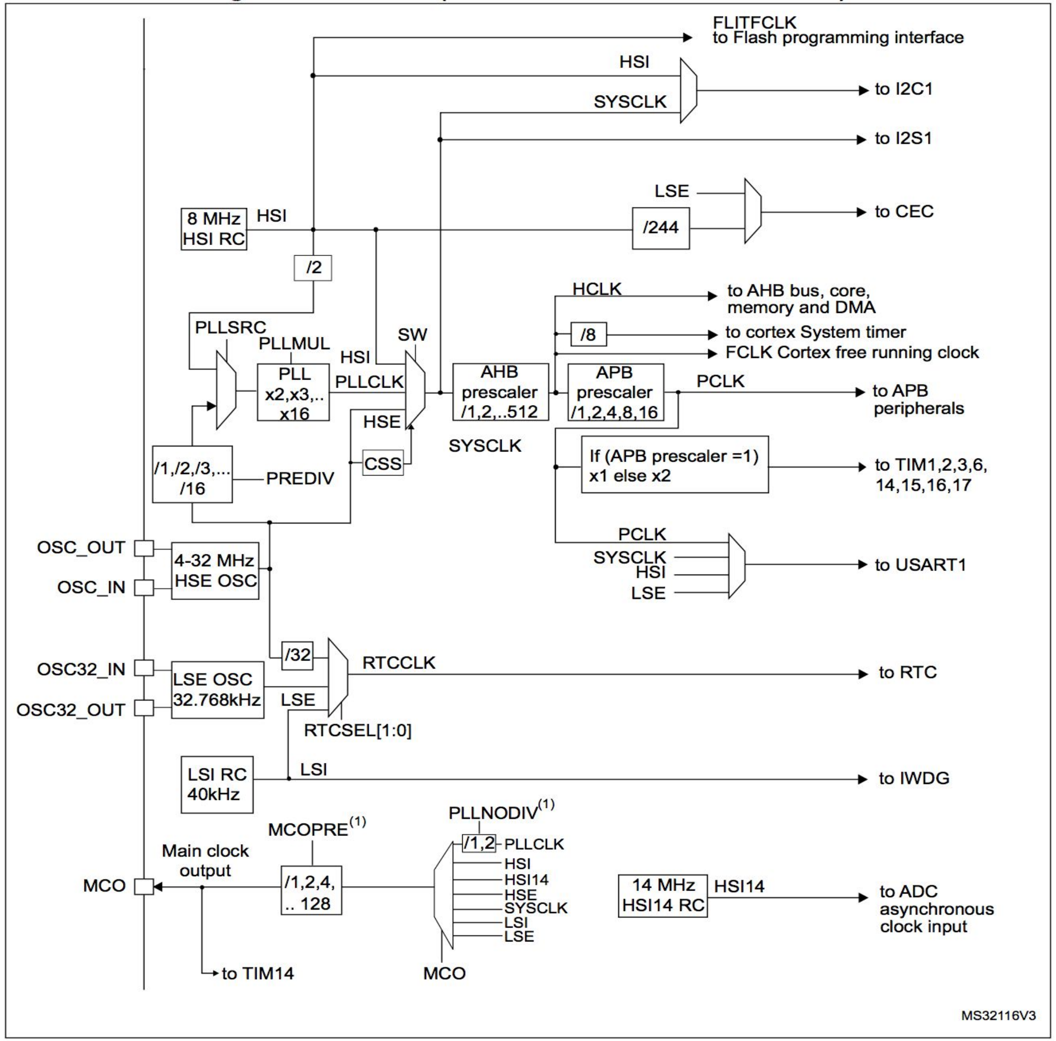


Figure 10. Clock tree (STM32F03x and STM32F05x devices)



RCC (Reset and clock control). Clock Control

Various main clock sources:

- HSI 8 MHz RC oscillator clock
- HSE oscillator clock
- PLL clock

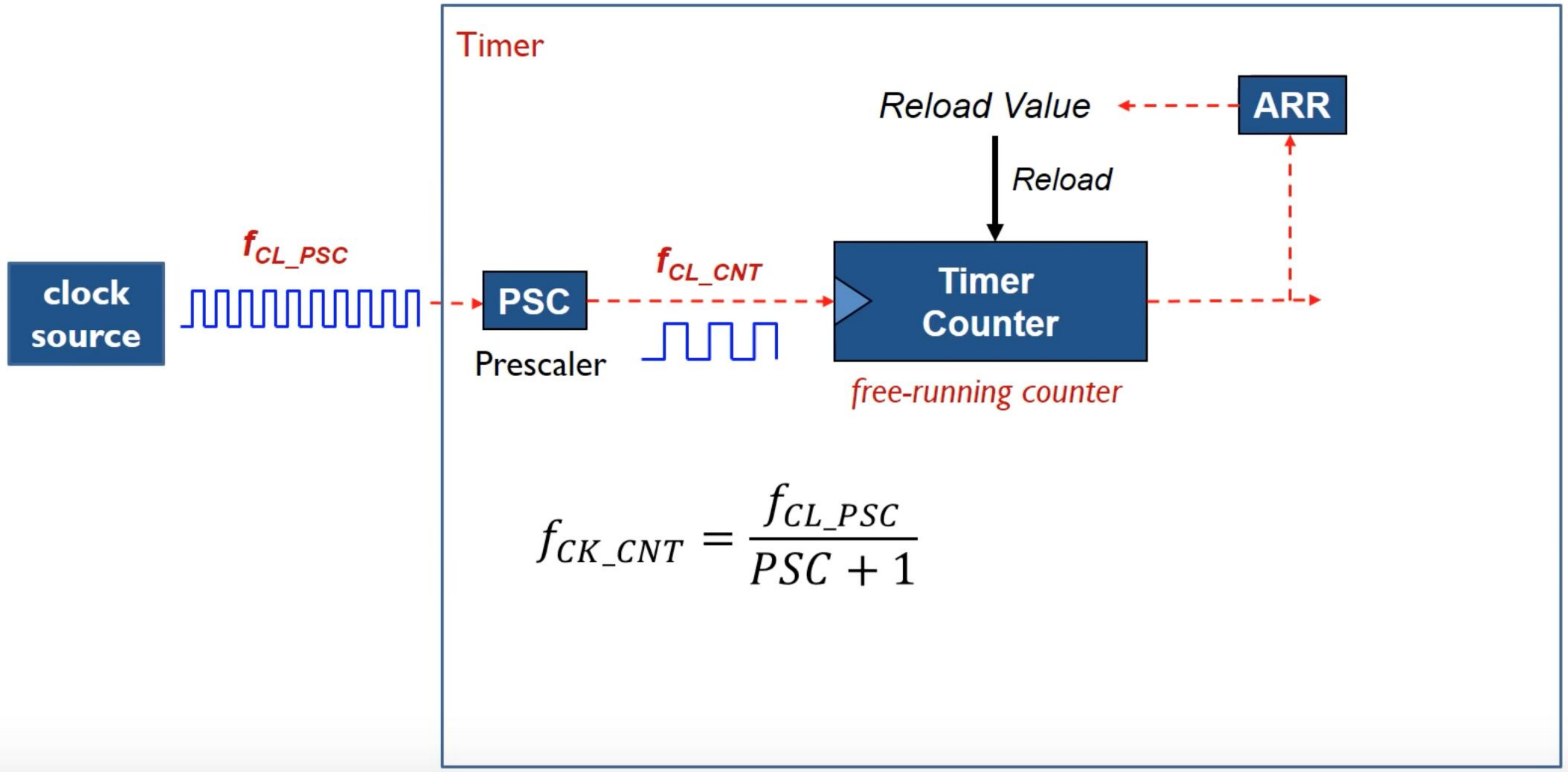
Additional clock sources:

- 40 kHz low speed internal RC (LSI RC)
- 32.768 kHz low speed external crystal (LSE crystal)
- 14 MHz high speed internal RC (HSI14) dedicated for ADC

Timers

- 16-bit up, down, up/down auto-reload counter
- 16-bit programmable prescaler
- Up to 4 independent channels for
  - Input Capture
  - Output Compare
  - PWM (Edge- and Center-aligned modes)
  - One-pulse mode output
- Interrupts
  - Counter overflow/underflow
  - Trigger event (counter start, stop)
  - Input capture
  - Output compare

Time Base Unit



$$f_{CK\_CNT} = \frac{f_{CL\_PSC}}{PSC + 1}$$

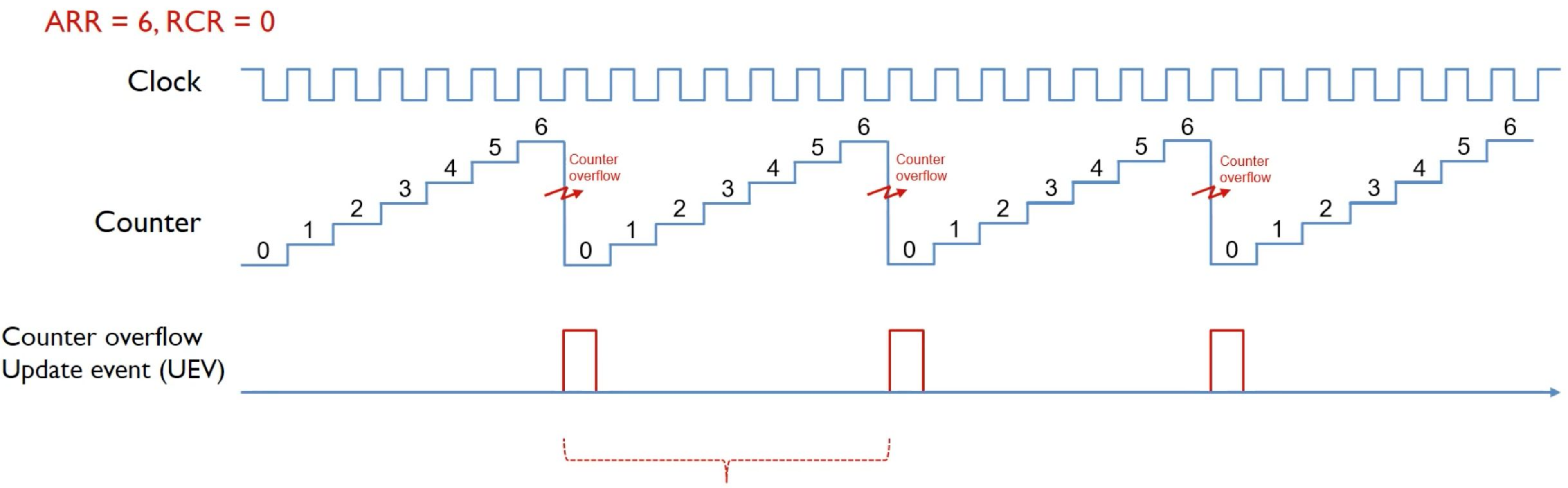
Main Timer Registers

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)
- Counter mode register (UP/DOWN/CENTER)

Clock sources

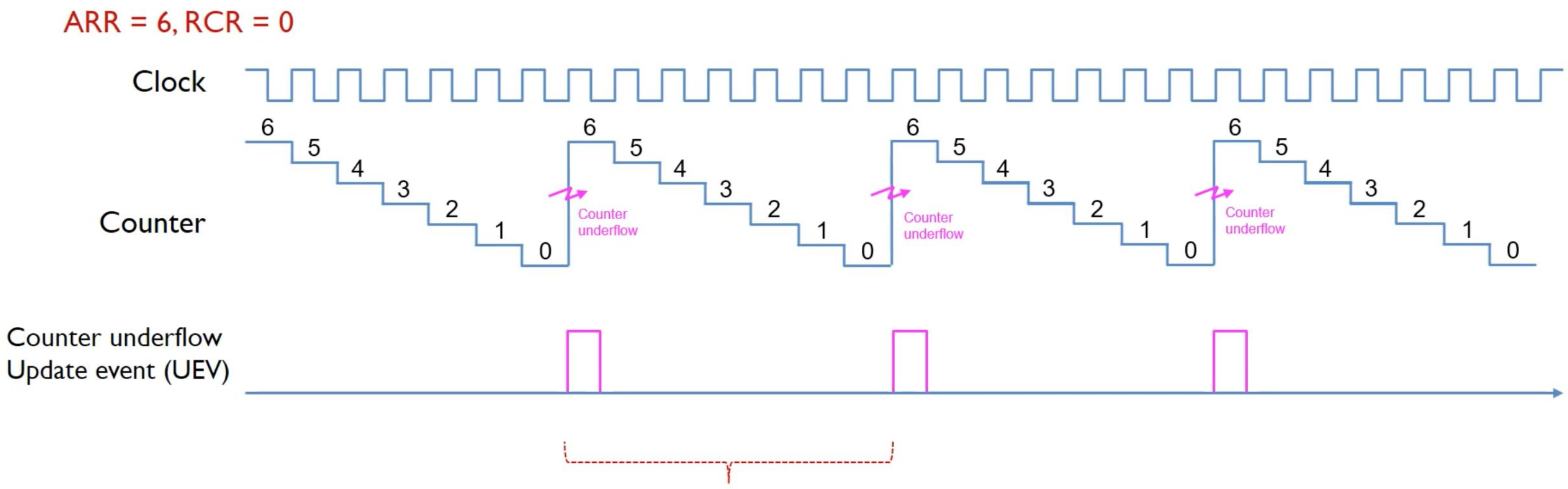
- Internal
  - Time counting
  - PWM generation
- External
  - Counts at each rising or falling edge on a selected pin
  - Measure time between two consecutive impulses
  - Encoder mode
  - Measure PWM duty cycle

Up-counting mode



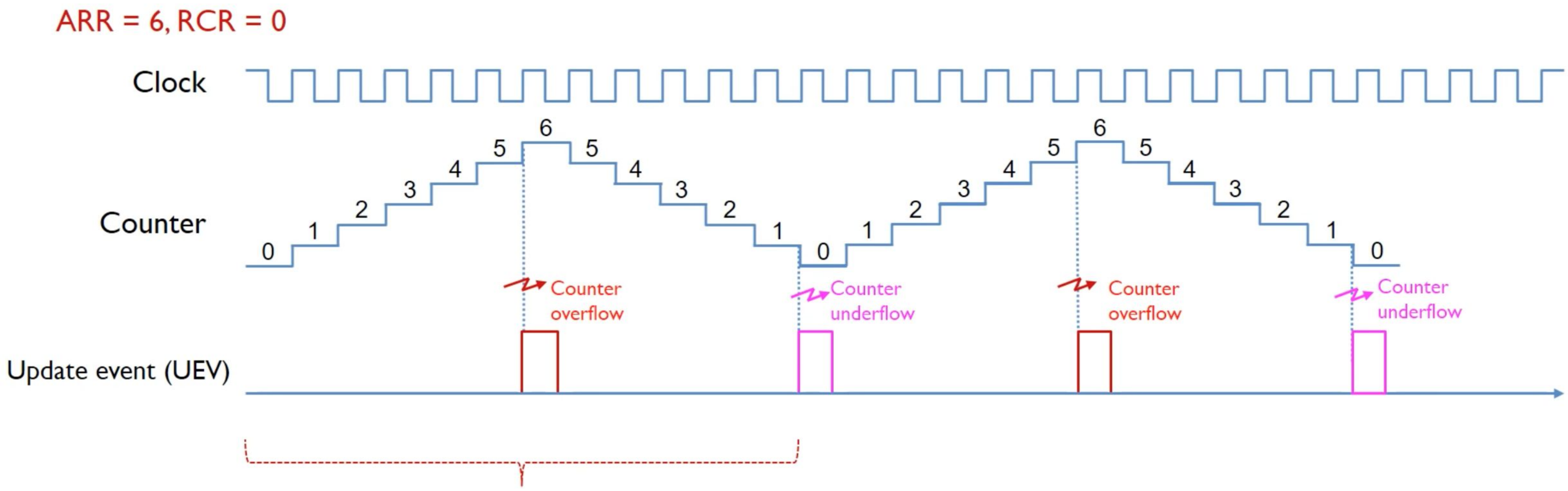
```
Period = (1 + ARR) * Clock Period
       = 7 * Clock Period
```

Down-counting mode



```
Period = (1 + ARR) * Clock Period
```

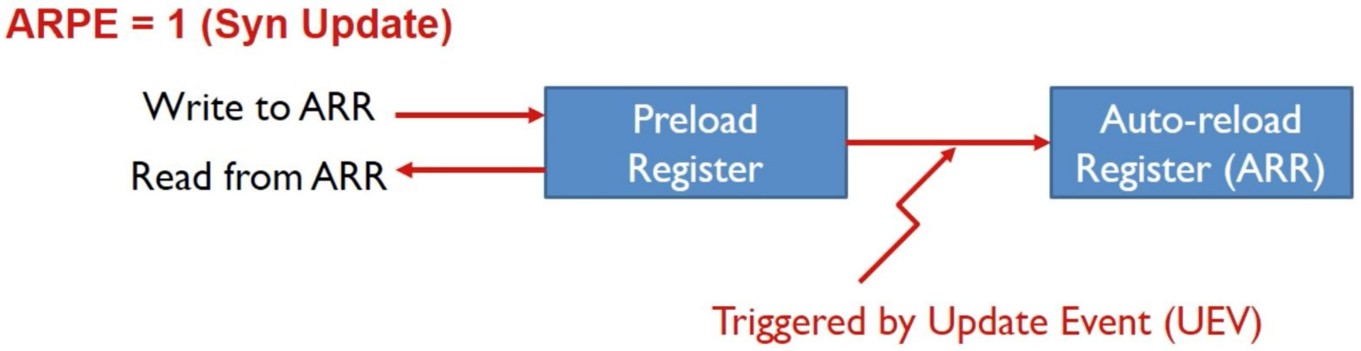
Center-aligned mode

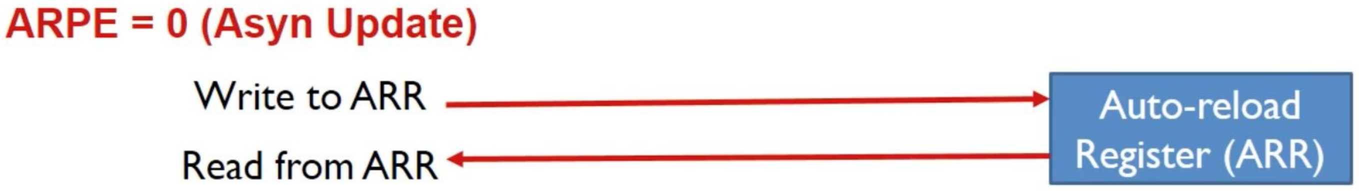


```
Period = 2 * ARR * Clock Period
       = 12 * Clock Period
```

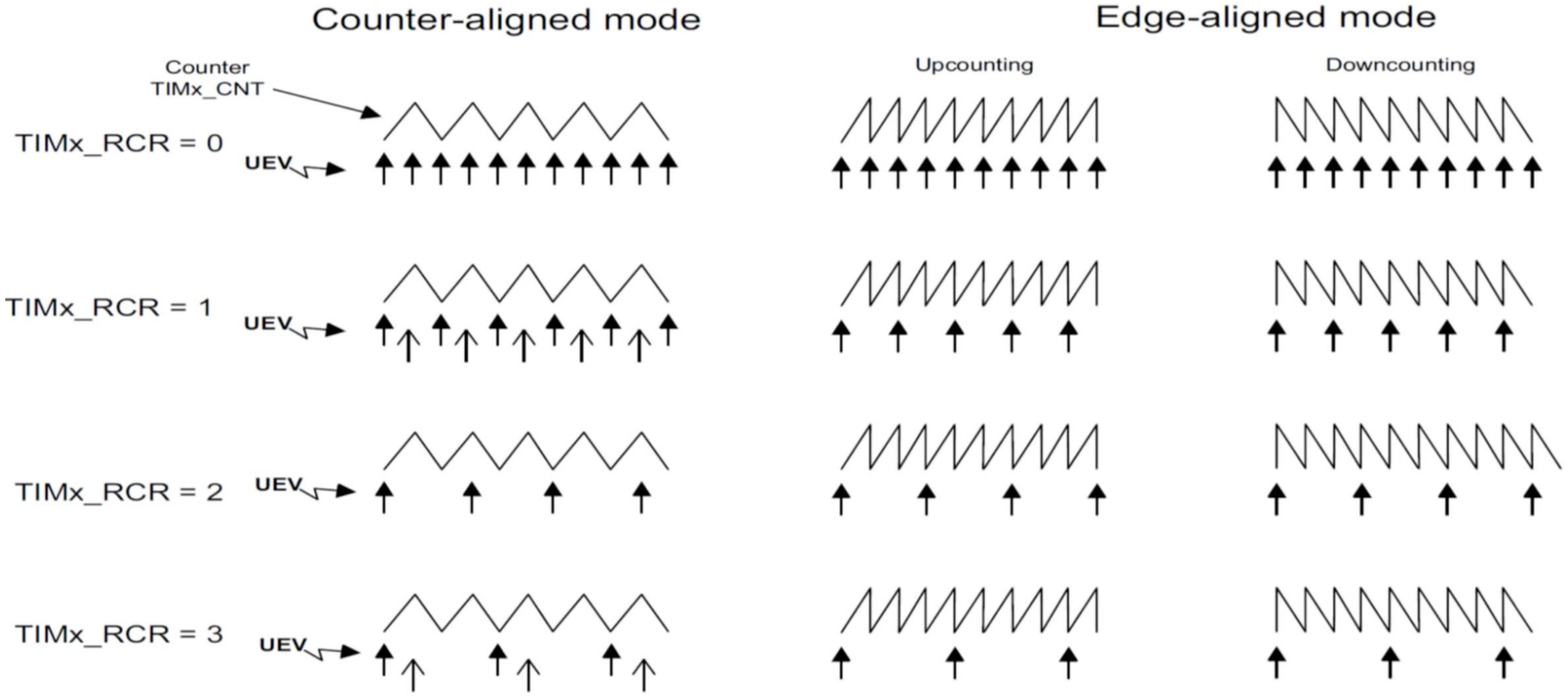
Auto-Reload Register

- Auto-Reload Preload Enable (ARPE) bit in TIMx_CR1

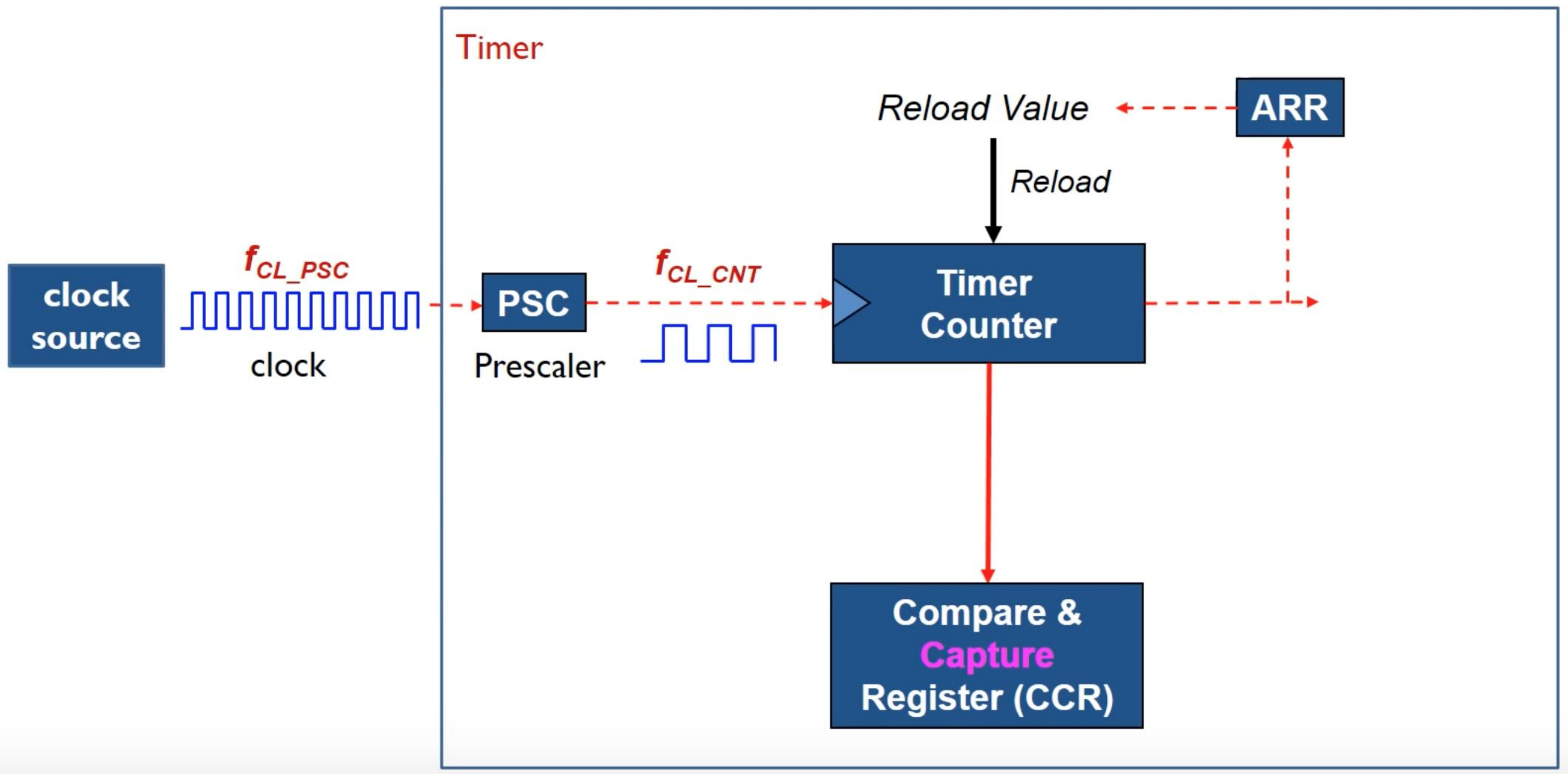




Repetition Counter Register

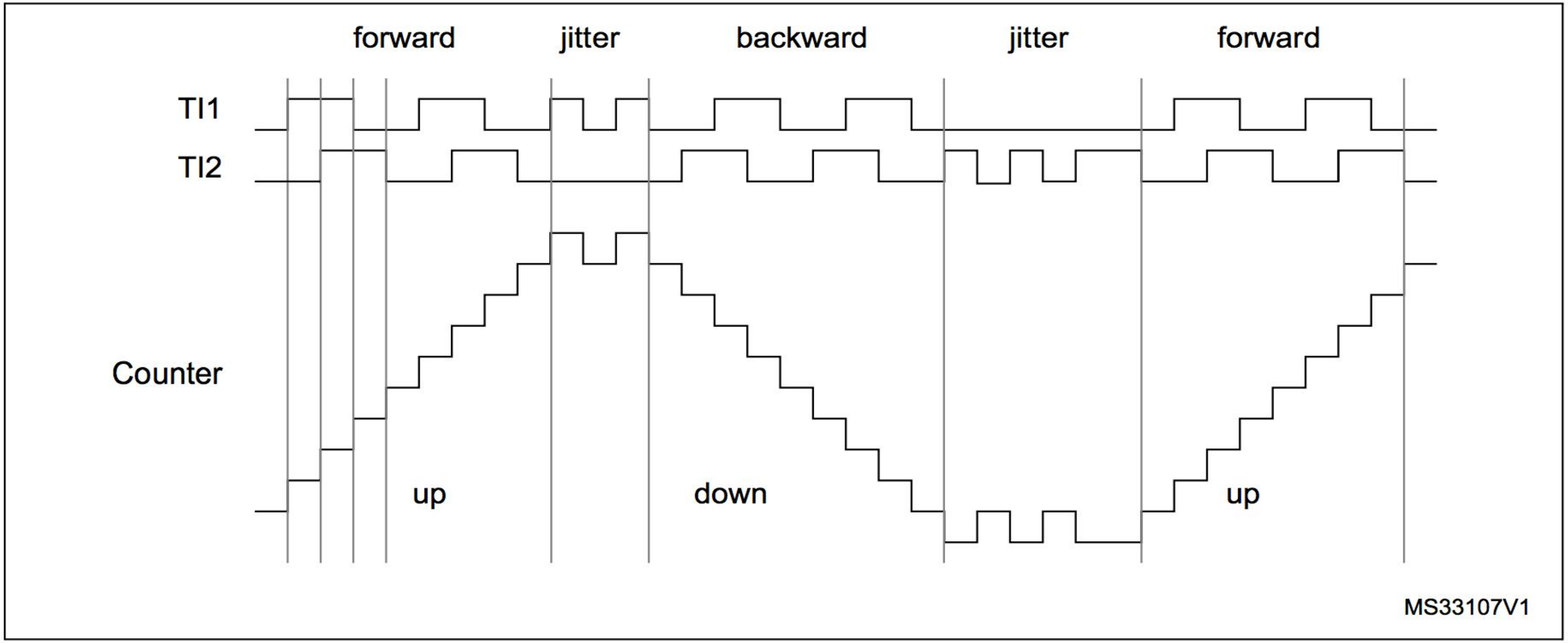


Input capture mode



Encoder mode

Figure 100. Example of counter operation in encoder interface mode.

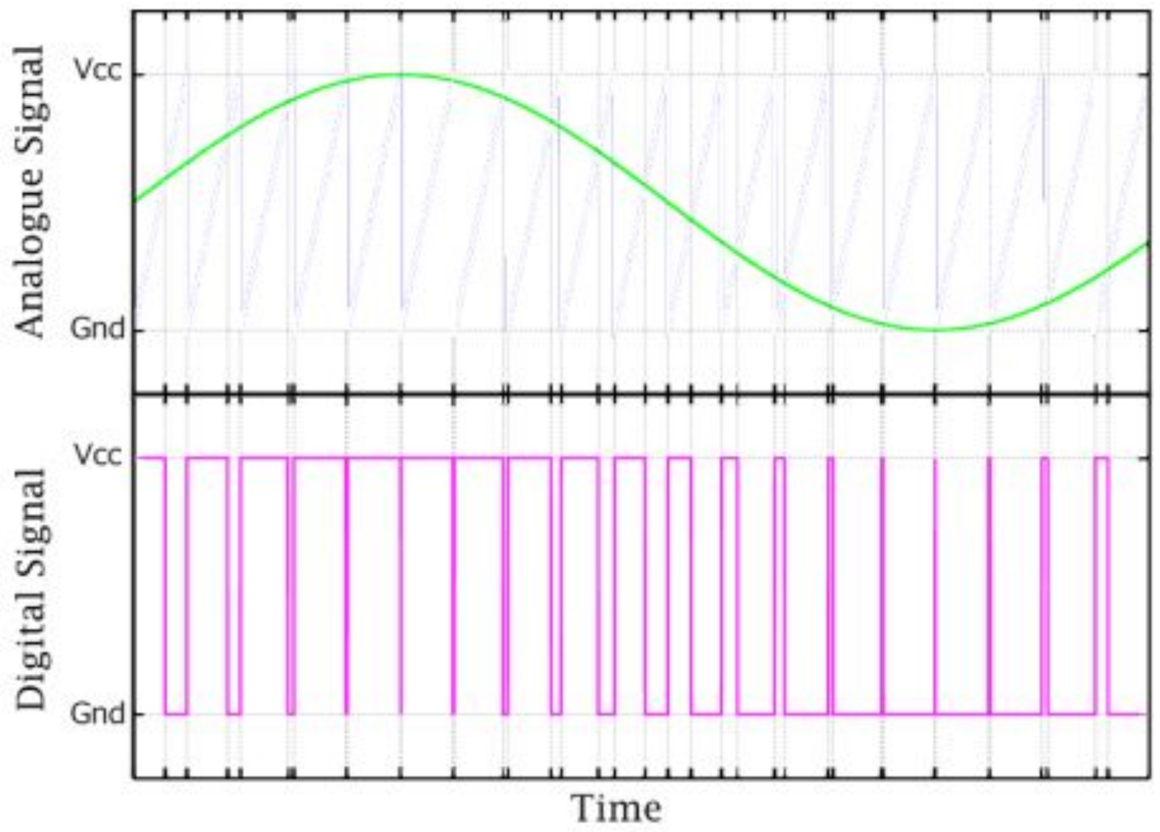


Introduction to embedded programming on STM32

Timers, UART

Skoltech, 2019

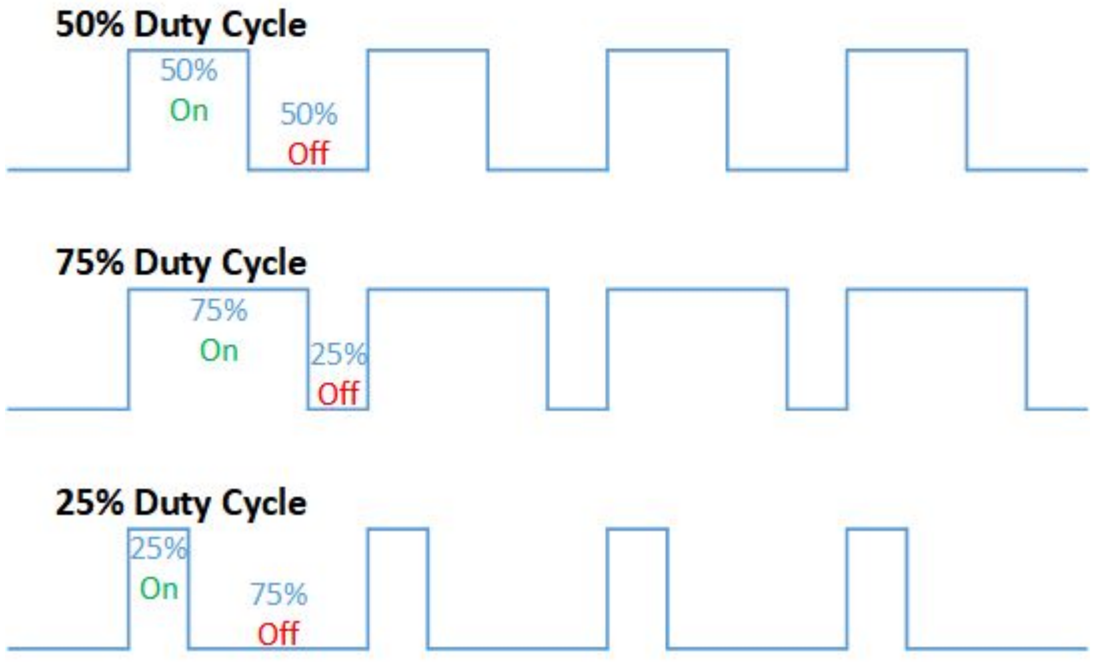
PWM Signal



PWM Signal

Key parameters:

1. Amplitude
2. Frequency
3. Duty Cycle

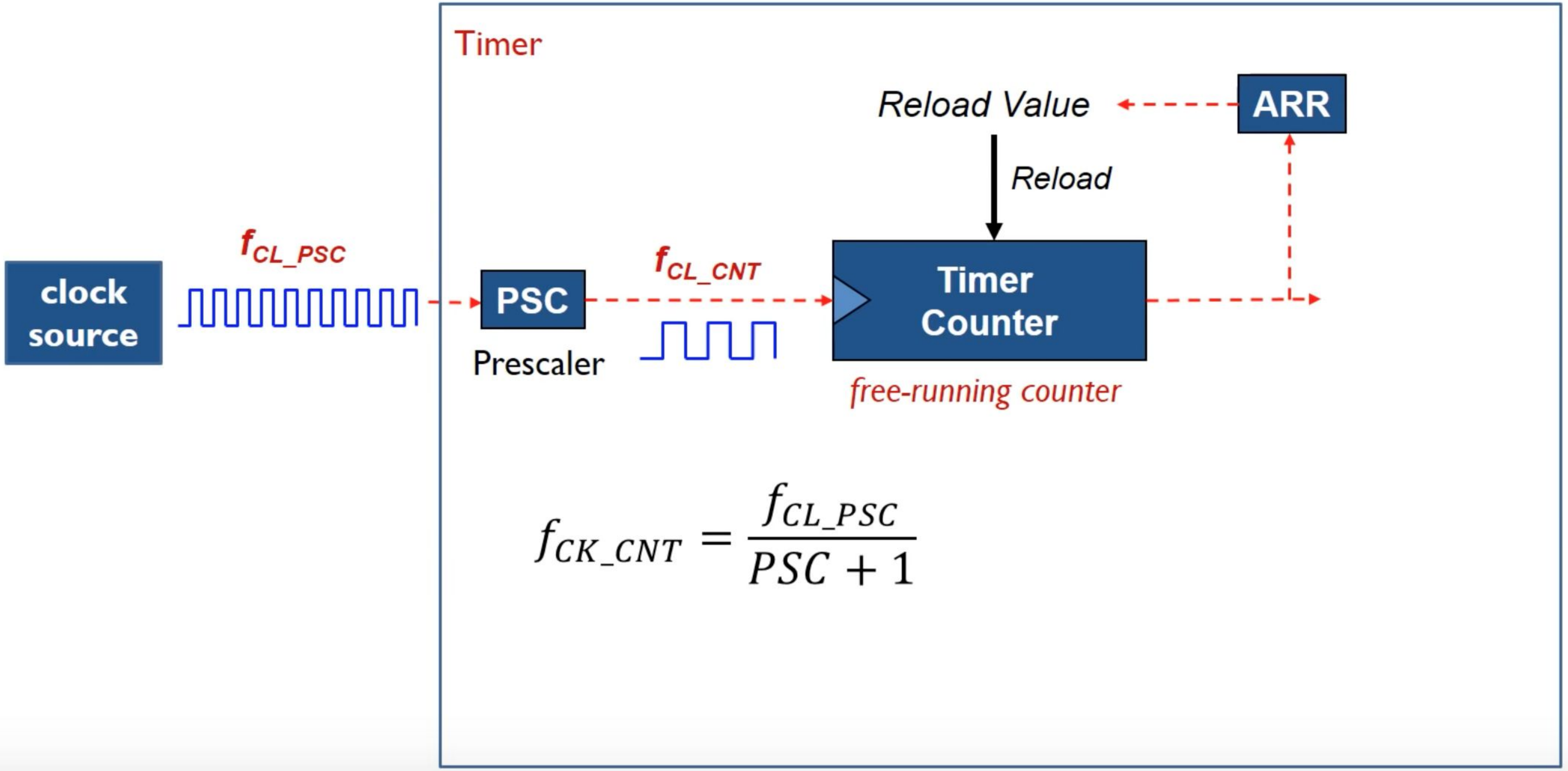


Main Timer Registers

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)

Counter mode register (UP/DOWN/CENTER)

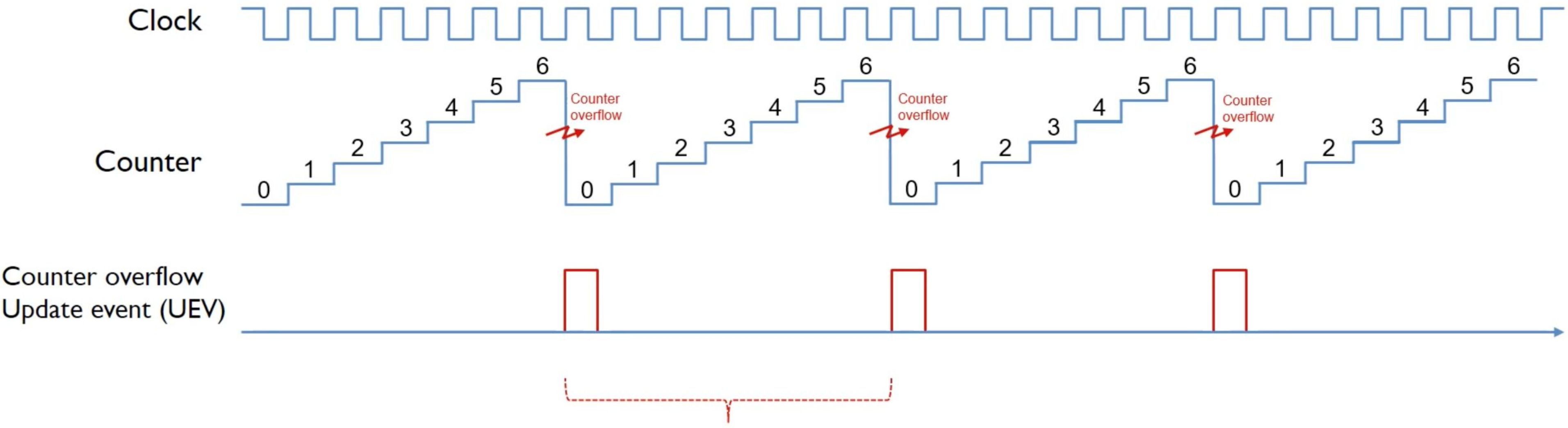
Time Base Unit



$$f_{CK\_CNT} = \frac{f_{CL\_PSC}}{PSC + 1}$$

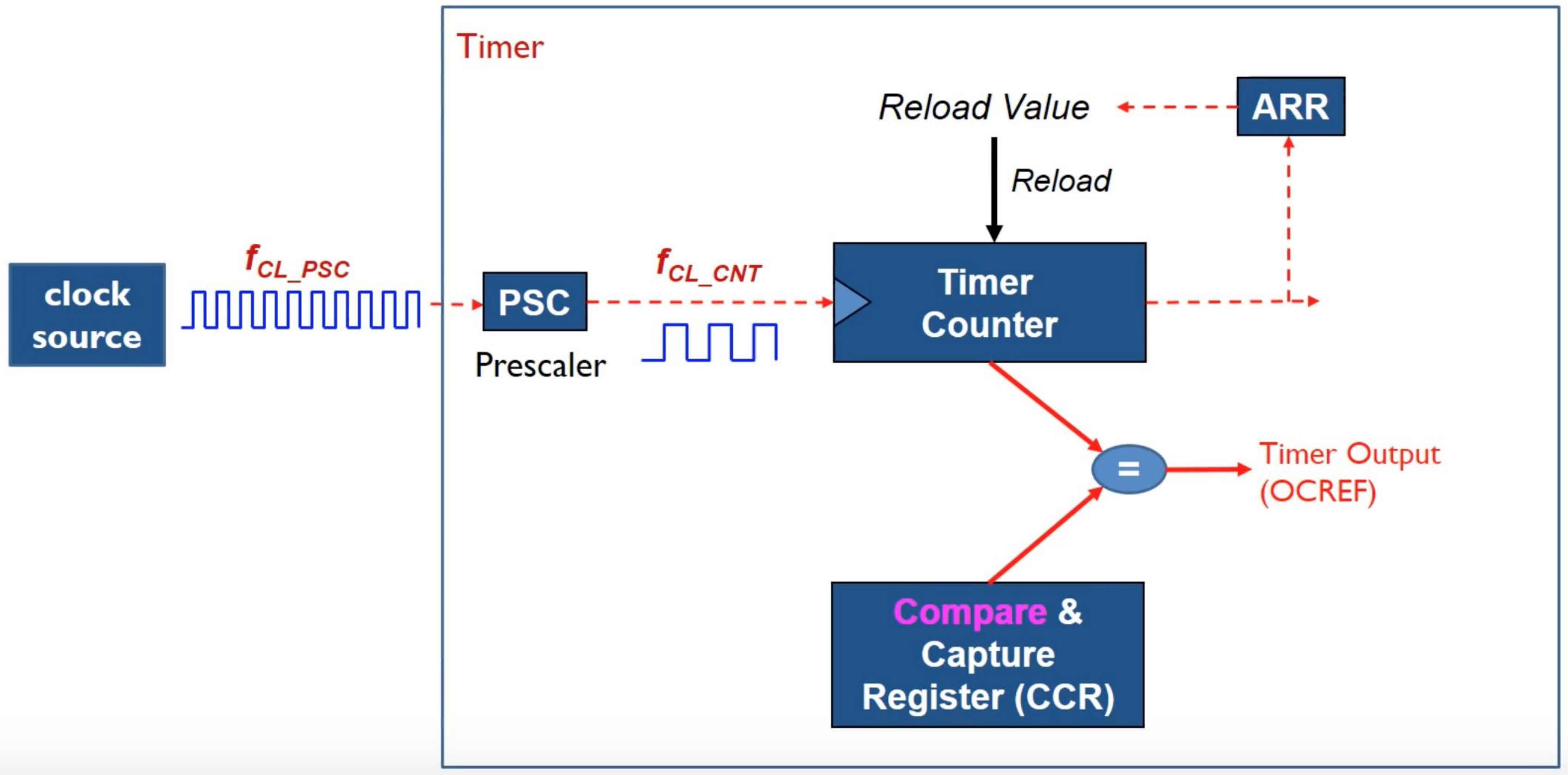
Up-counting mode

ARR = 6, RCR = 0



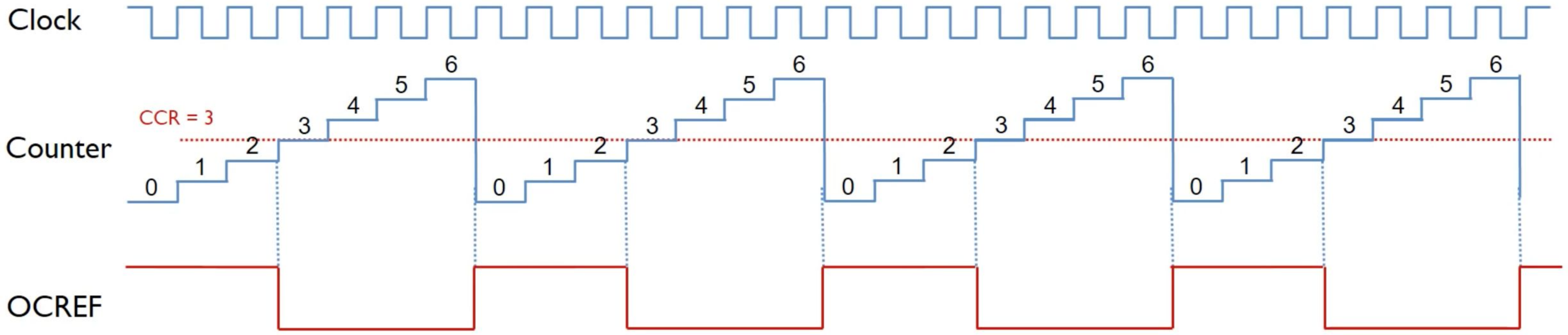
```
Period = (1 + ARR) * Clock Period
       = 7 * Clock Period
```

Output compare mode



PWM Mode 1 (Low-True)

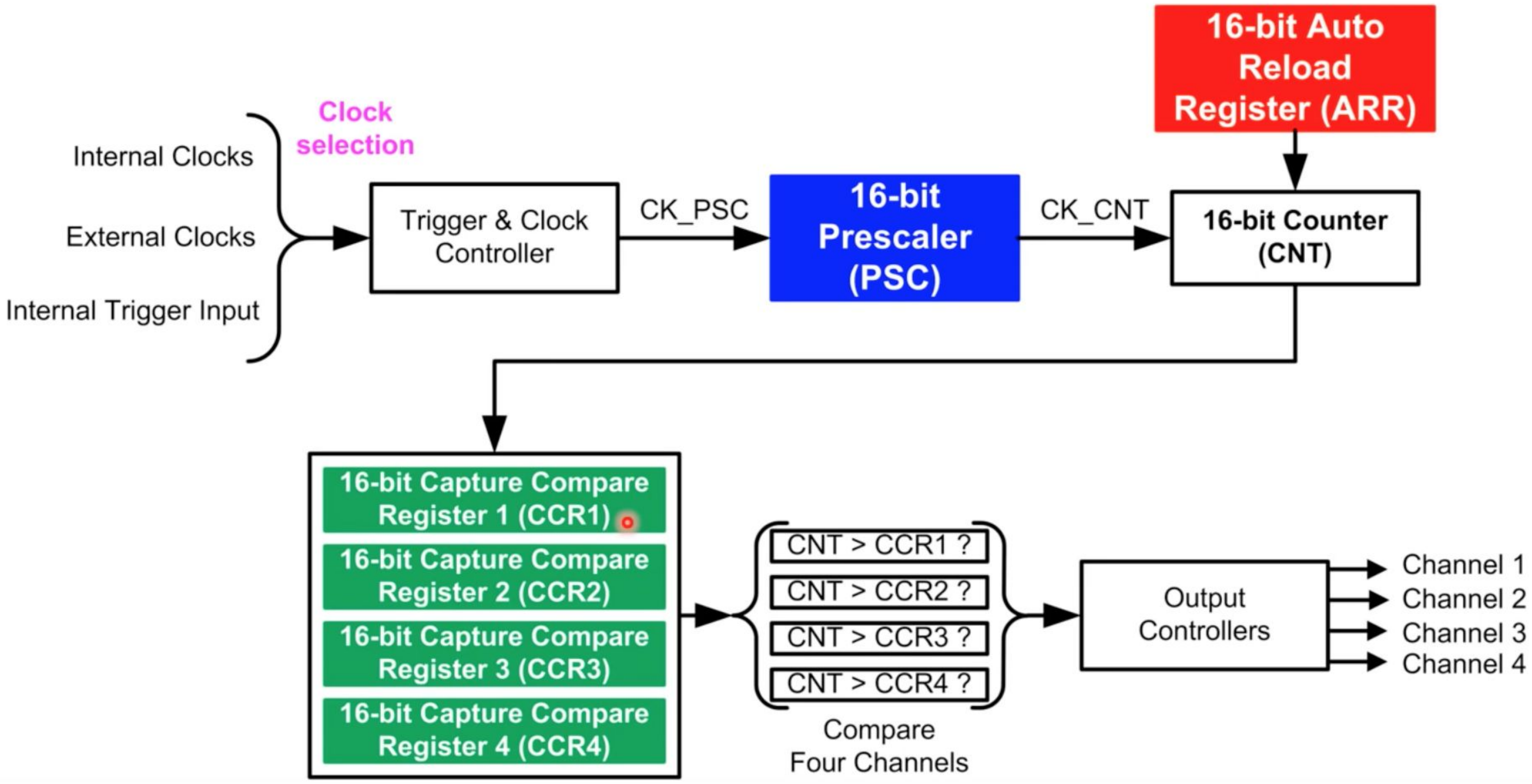
Upcounting mode, ARR = 6, CCR = 3, RCR = 0



Duty Cycle = $\dfrac{\text{CCR}}{\text{ARR + 1}}$

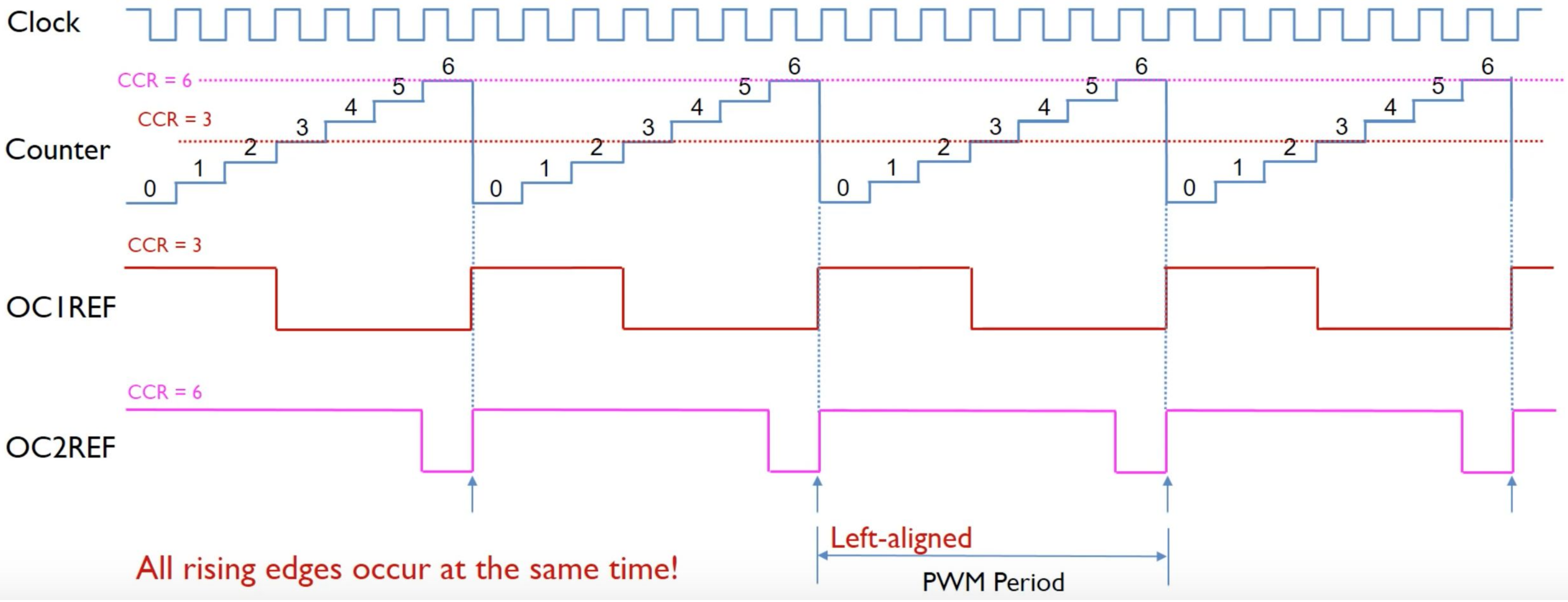
$= \dfrac{3}{7}$

Period = (1 + ARR) * Clock Period
= 7 * Clock Period


Internal Clocks
External Clocks
Internal Trigger Input
Clock selection
Trigger & Clock Controller
CK_PSC
16-bit Prescaler (PSC)
CK_CNT
16-bit Auto Reload Register (ARR)
16-bit Counter (CNT)
16-bit Capture Compare Register 1 (CCR1)
16-bit Capture Compare Register 2 (CCR2)
16-bit Capture Compare Register 3 (CCR3)
16-bit Capture Compare Register 4 (CCR4)
CNT > CCR1 ?
CNT > CCR2 ?
CNT > CCR3 ?
CNT > CCR4 ?
Compare Four Channels
Output Controllers
Channel 1
Channel 2
Channel 3
Channel 4

Edge-aligned PWM

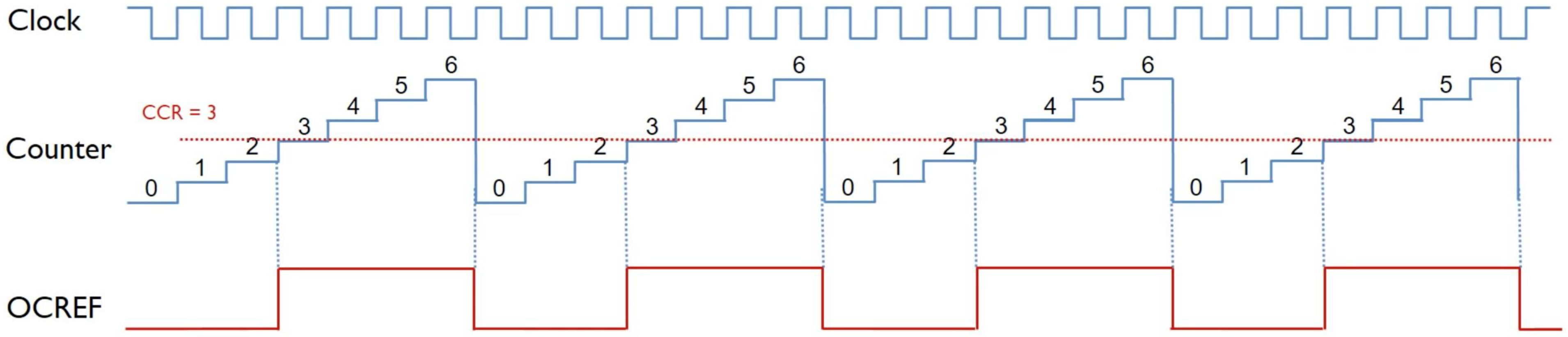
Upcounting mode, ARR = 6, CCR = 3, RCR = 0



All rising edges occur at the same time!

PWM Mode 2 (High-True)

Upcounting mode, ARR = 6, CCR = 3, RCR = 0

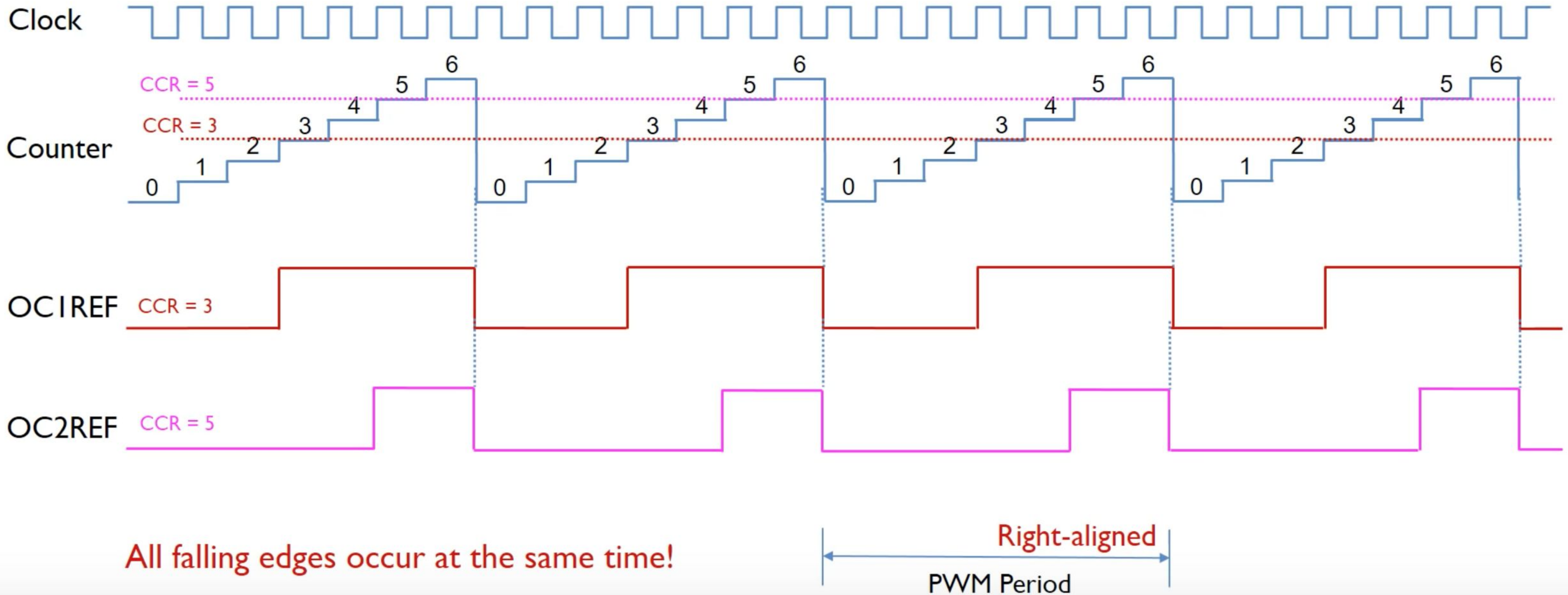


$$\text{Duty Cycle} = 1 - \frac{\text{CCR}}{\text{ARR} + 1} = \frac{4}{7}$$

$$\text{Period} = (1 + \text{ARR}) * \text{Clock Period} = 7 * \text{Clock Period}$$

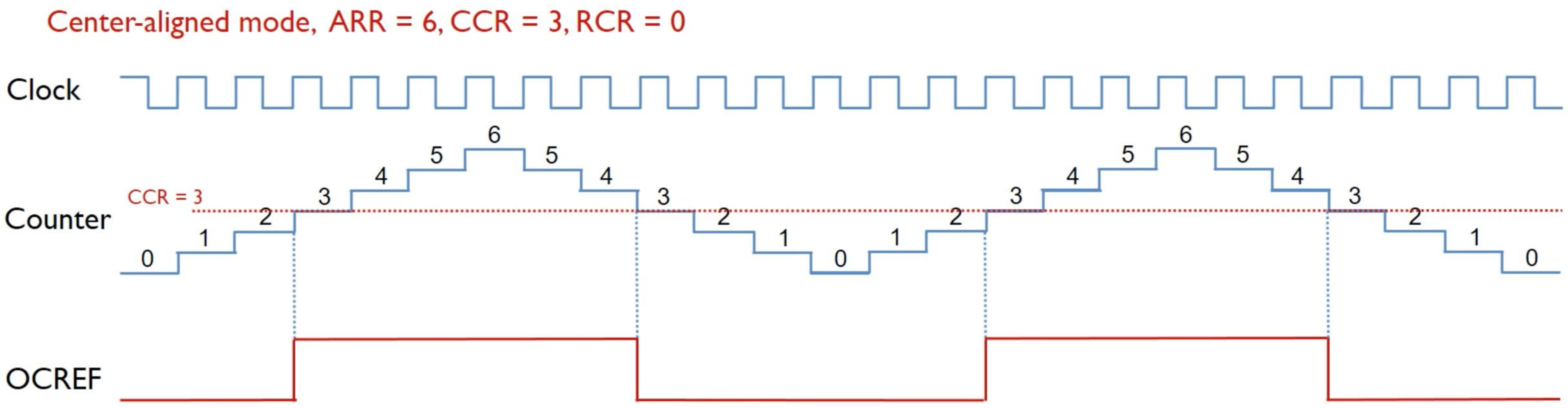
Edge-aligned PWM

Upcounting mode, ARR = 6, CCR = 3, RCR = 0



All falling edges occur at the same time!

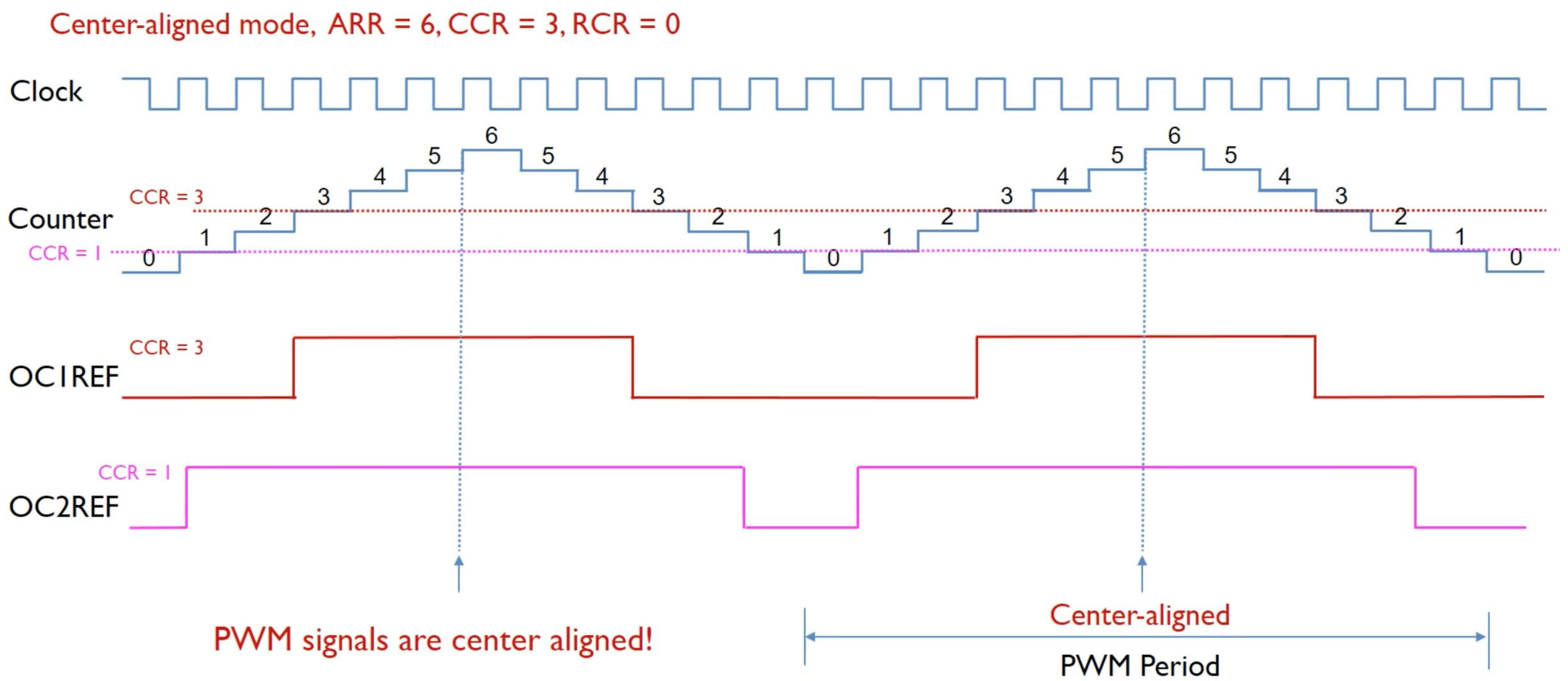
Center-aligned PWM Mode 2 (High-True)



$$\text{Duty Cycle} = 1 - \frac{\text{CCR}}{\text{ARR}} = \frac{1}{2}$$

$$\text{Period} = 2 * \text{ARR} * \text{Clock Period} = 12 * \text{Clock Period}$$

Center-aligned PWM Mode 2 (High-True)



PWM Output Polarity

Mode	Counter < CCR	Counter ≥ CCR
PWM mode 1 (Low True)	Active	Inactive
PWM mode 2 (High True)	Inactive	Active

Output Polarity:

- Software can program the CCxP bit in the TIMx_CCER register

	Active	Inactive
Active High	High Voltage	Low Voltage
Active Low	Low Voltage	High Voltage

Code exercise. PWM configuration

1. Configure output pin
2. Set alternate mode
3. Set up Time Base Unit
4. Set up Output Compare Unit
5. Enable Counter

See examples/pwm.c

GPIO configuration

Set clocking for Port C

```
LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOC);
```

Set pin to alternate mode

```
LL_GPIO_SetPinMode(GPIOC, LL_GPIO_PIN_8, LL_GPIO_MODE_ALTERNATE);
```

Set alternate function (number from documentation)

```
LL_GPIO_SetAFPin_8_15(GPIOC, LL_GPIO_PIN_8, LL_GPIO_AF_1);
```

Set push/pull output type

```
LL_GPIO_SetPinOutputType(GPIOC, LL_GPIO_PIN_8, LL_GPIO_OUTPUT_PUSHPULL);
```

Timer configuration

LL_APB1_GRP1_EnableClock(LL_APB1_GRP1_PERIPH_TIM3);	**Set clocking for timer**
LL_TIM_CC_EnableChannel(TIM3, LL_TIM_CHANNEL_CH3);	**Enable capture/compare module 3rd channel**
LL_TIM_SetCounterMode(TIM3, LL_TIM_COUNTERMODE_UP);	**Set counting mode**
LL_TIM_SetAutoReload(TIM3, 48000);	**Set ARR register**
LL_TIM_SetPrescaler(TIM3, 1);	**Set PCS register**
LL_TIM_OC_SetMode(TIM3, LL_TIM_CHANNEL_CH3, LL_TIM_OCMODE_PWM1);	**Set PWM mode**
LL_TIM_OC_EnablePreload(TIM3, LL_TIM_CHANNEL_CH3);	**Enable preload for synchronous mode**
LL_TIM_OC_SetCompareCH3(TIM3, 4800);	**Set CCR register**
LL_TIM_GenerateEvent_UPDATE(TIM3);	**Generate update event to update registers**
LL_TIM_EnableCounter(TIM3);	**Start counting**