TIMER

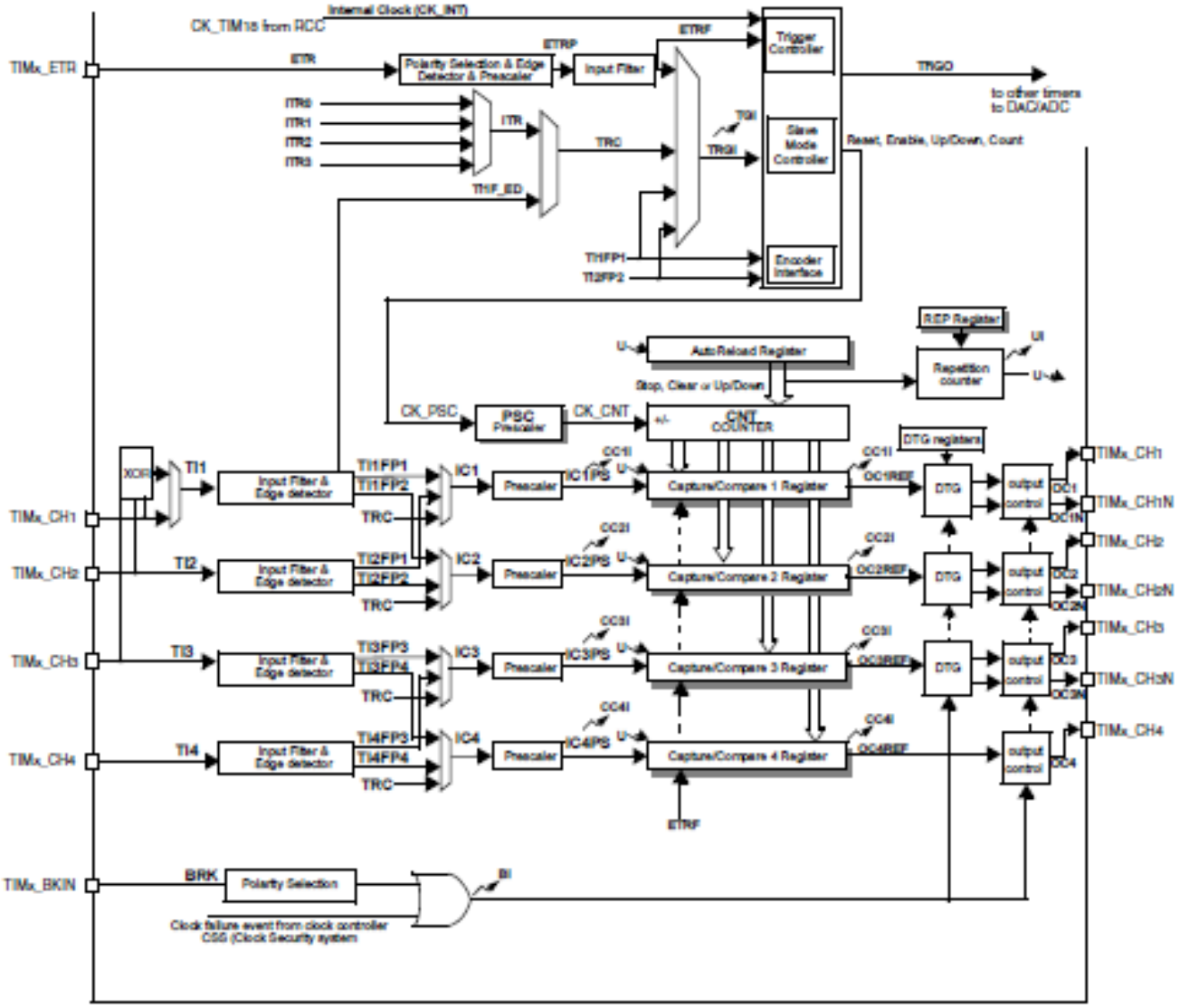
TIM feature

- 16 bit up, down, up/down, and auto-reload counter
- 16 bit programmable prescaler allowing dividing the counter clock by any number between 1 and 65535
- Up to 4 independent channel for:
  - Input, Output, PWM, and One-pulse mode output
- Synchronize circuit with other timers

Tim feature

- Repetition counter to update timer registers only after a given number of cycles of the counter
- Interrupt/DMA generation
- Support incremental encoder and hall sensor circuitry for positioning purpose
- Trigger input for external clock

Advanced control timer block diagram



Timer-base unit

- Counter register (TIMx_CNT)

  for counting purpose
- Prescaler register (TIMx_PSC)

  for clock division
- Auto-reload register (TIMx_ARR)

  setting the maximum/minimum count value to preload register
- Repetition counter register (TIMx_RCR)

  the number of repetition

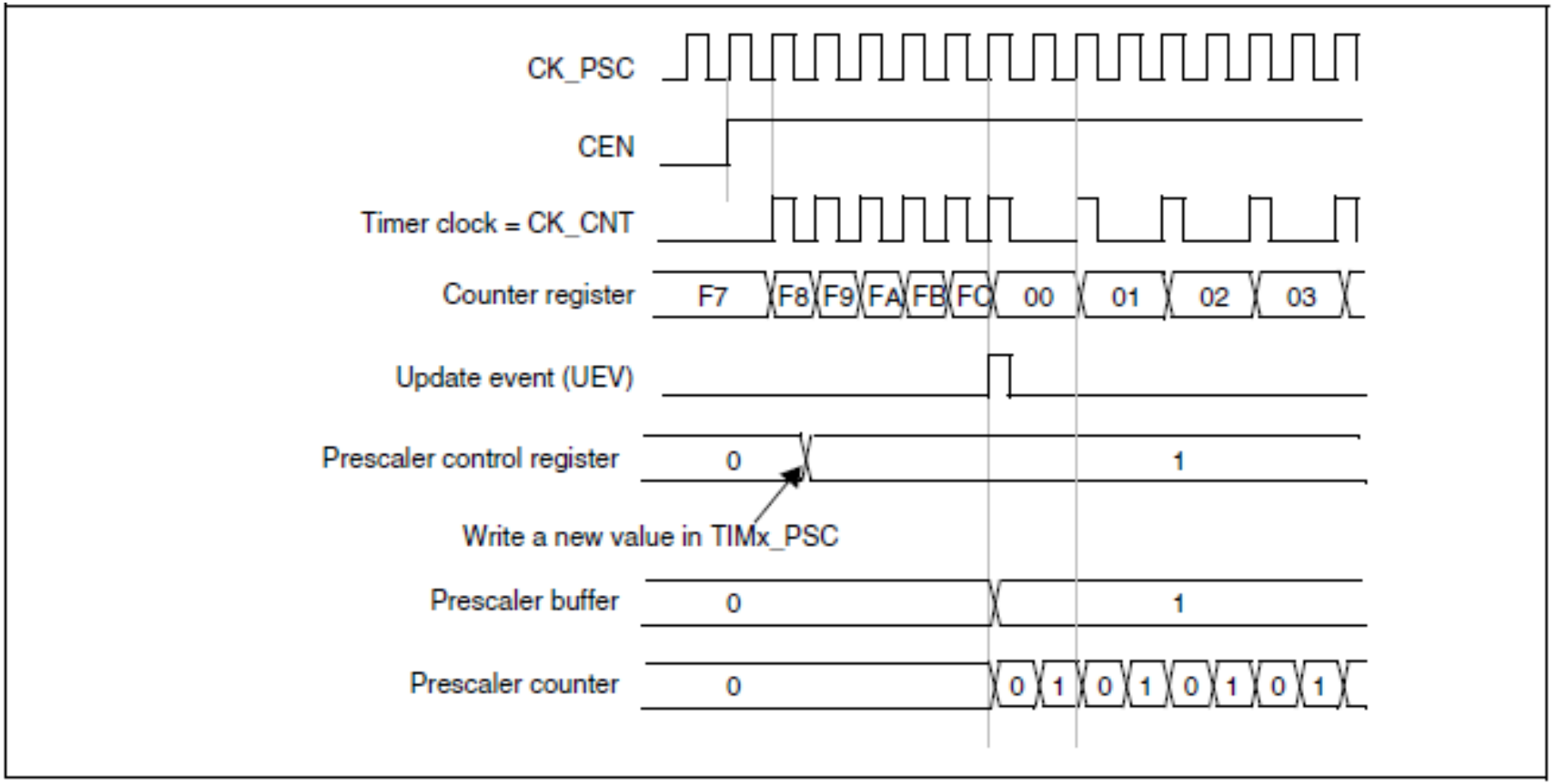
Auto-reload register

- Writing or reading from the auto-reload register will access to preload register
- The content of preload register are transferred to the shadow register at each update event (UEV) when ARPE register =1
- When ARPE register =0, the update will happen immediately
- The update of UEV is sent when the counter reach overflow

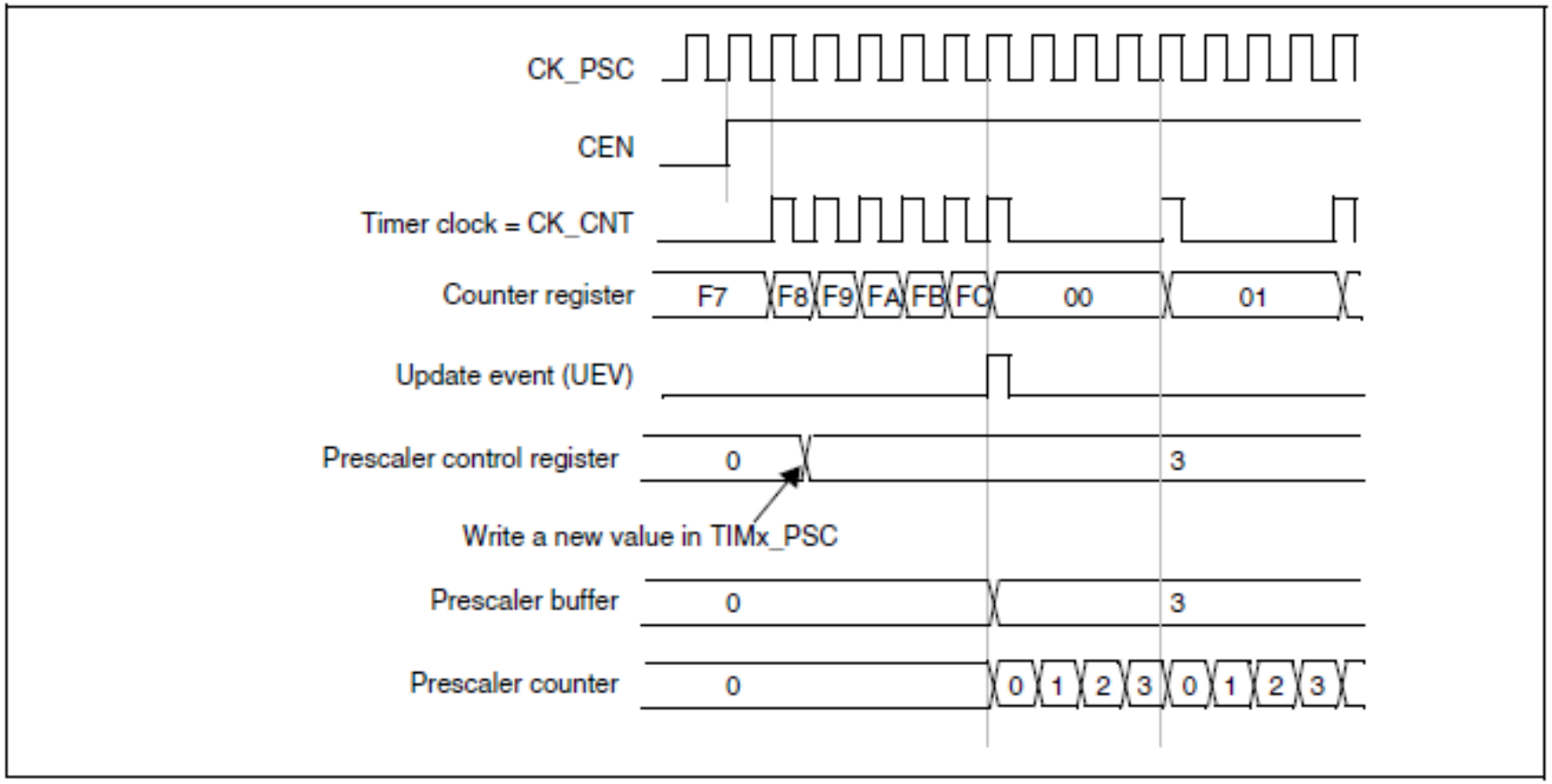
Prescaler description

- The prescaler can divide the counter clock frequency by any factor between 1 and 65536
- It can be updated on the fly, but the change will happen at the next update event
- The value of prescaler has to be added with 1

Counter timing diagram with prescale division changes from 1 to 2



Counter timing diagram with prescale division changes from 1 to 4



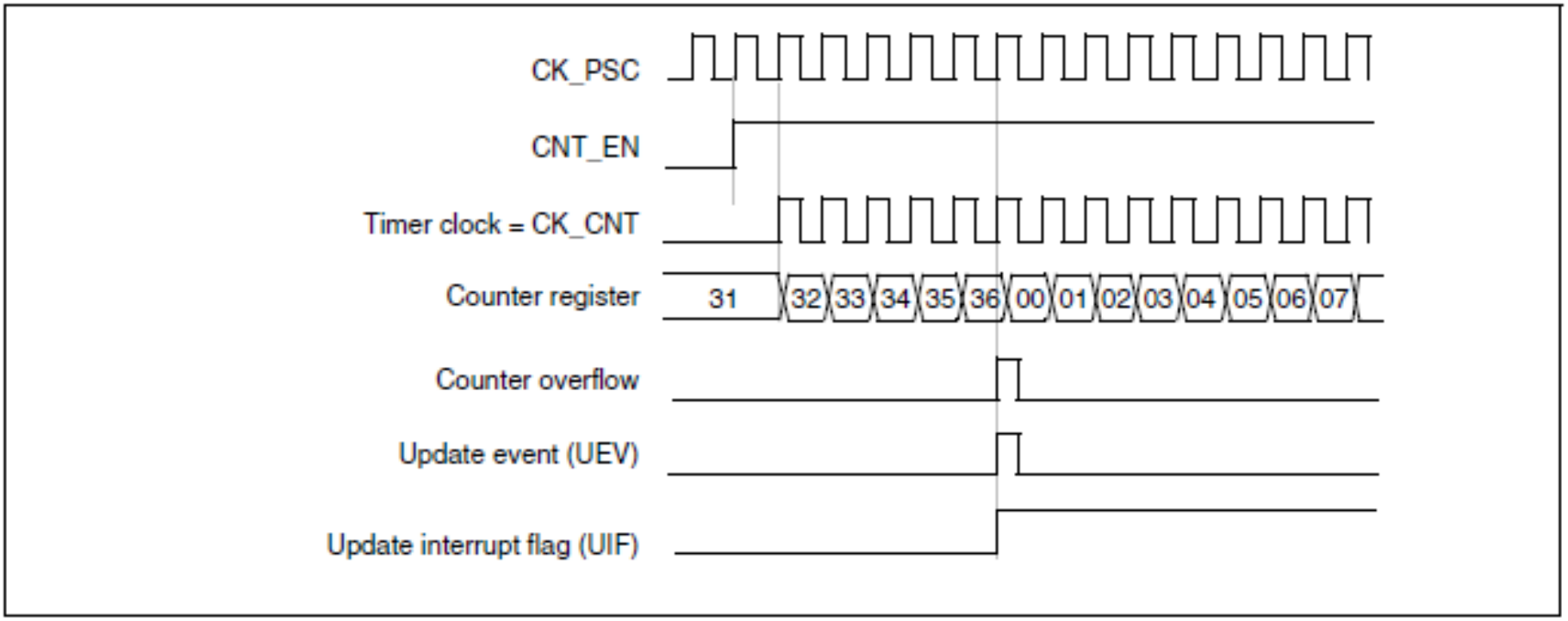
Counter modes

- Upcounting mode
- Downcounting mode
- Up/Down counting mode

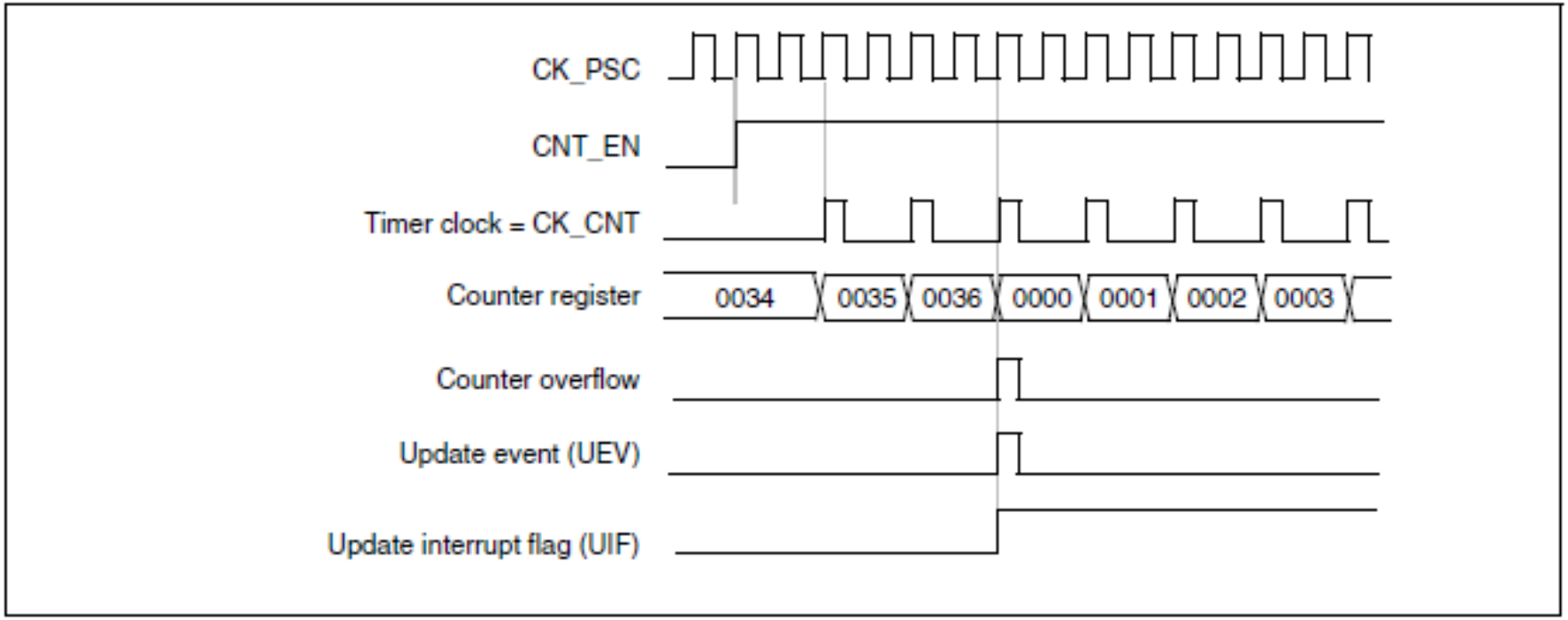
Upcounting

- The counter counts from 0 to auto-reloaded valued
- Then, it restart from 0 and generate a counter overflow event
- If the repetition counter is used, the update event (UEV) is generated and will repeat up to the number of times programmed in repetition counter register

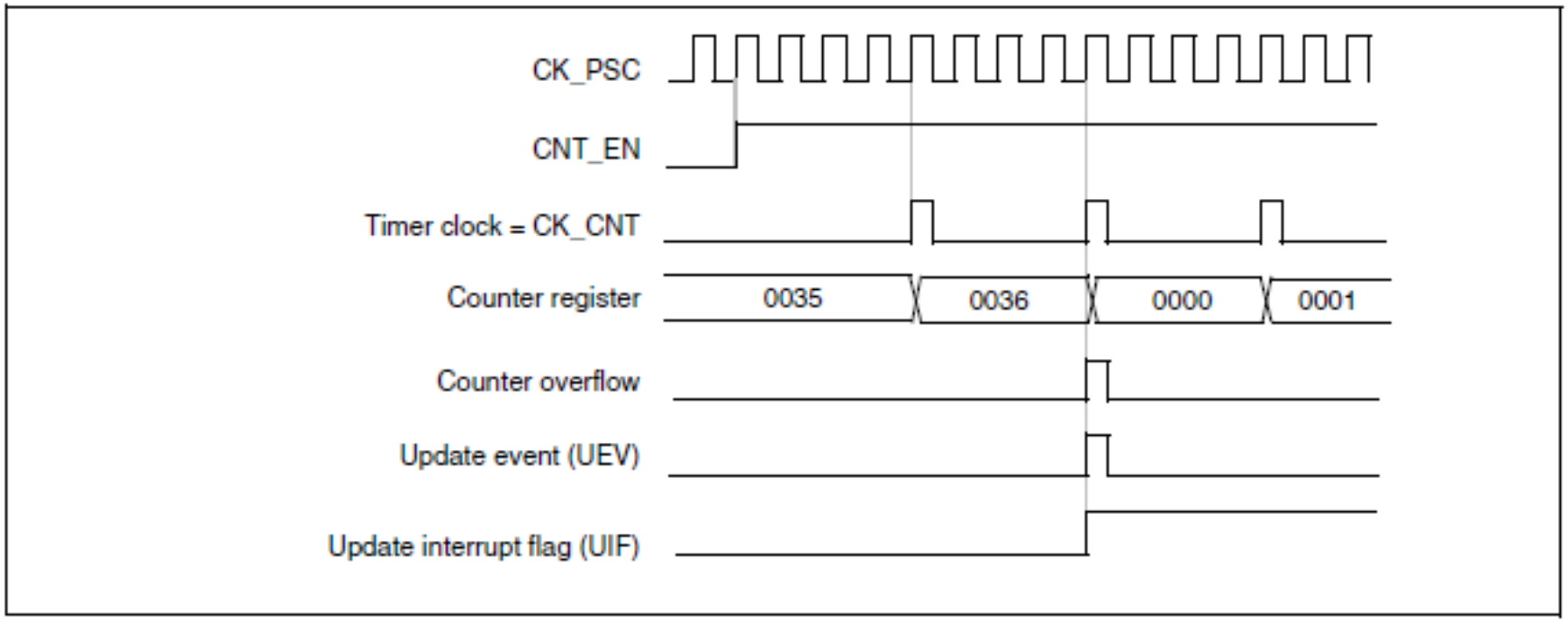
Counter timing diagram (divided by 1)



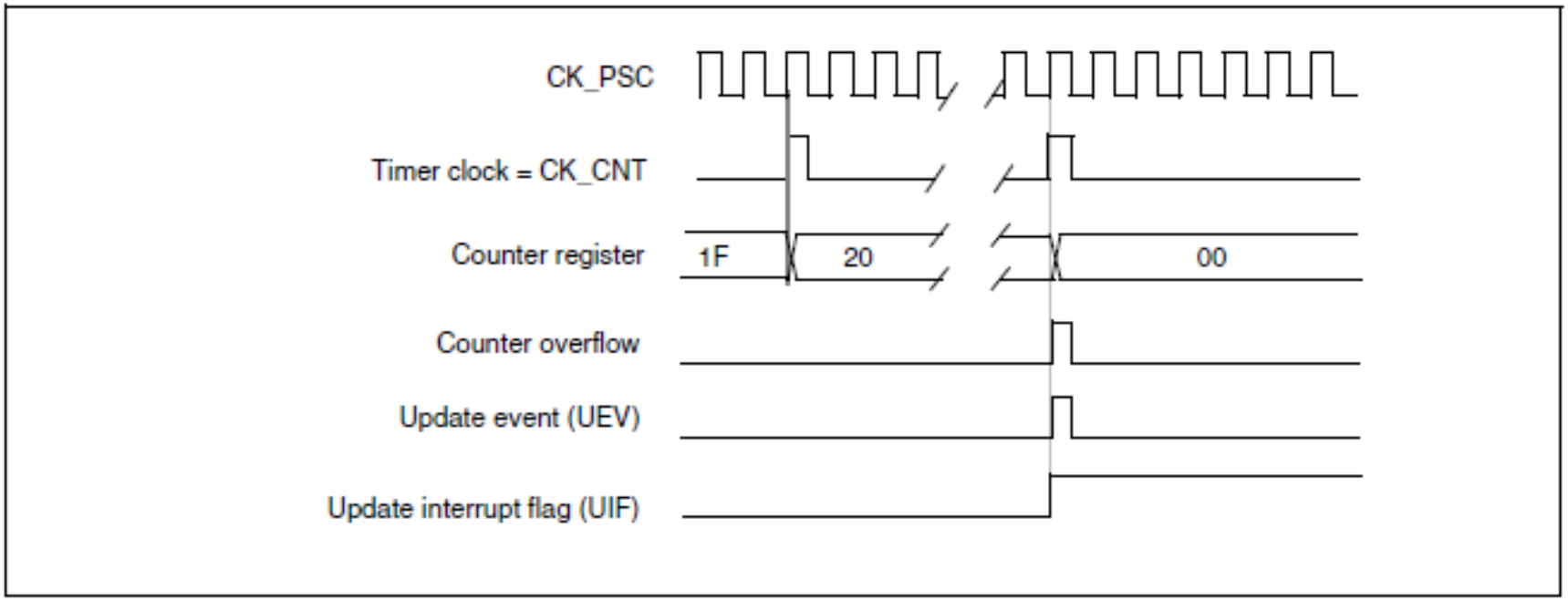
Counter timing diagram (divided by 2)



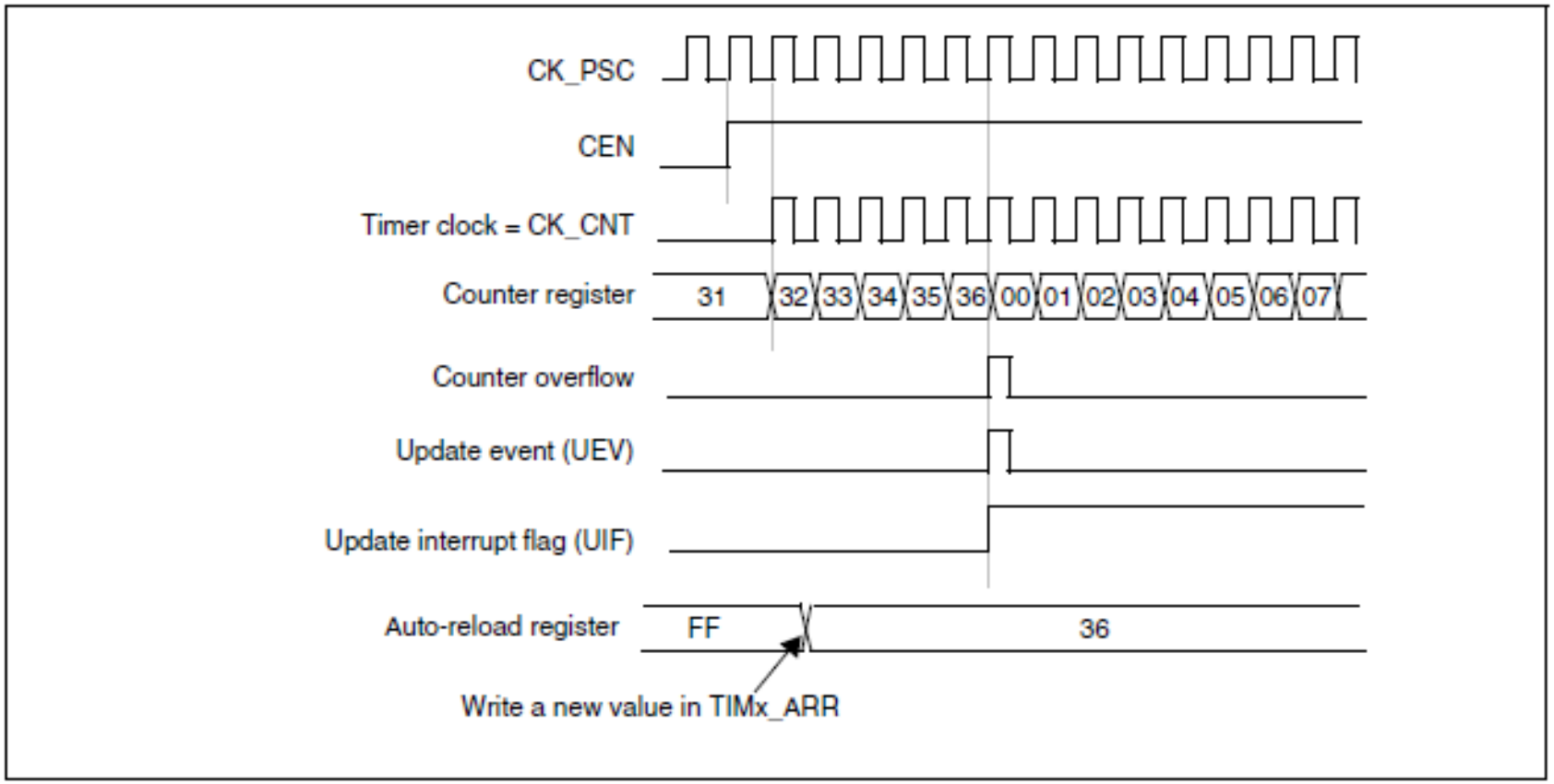
Counter timing diagram (divided by 4)



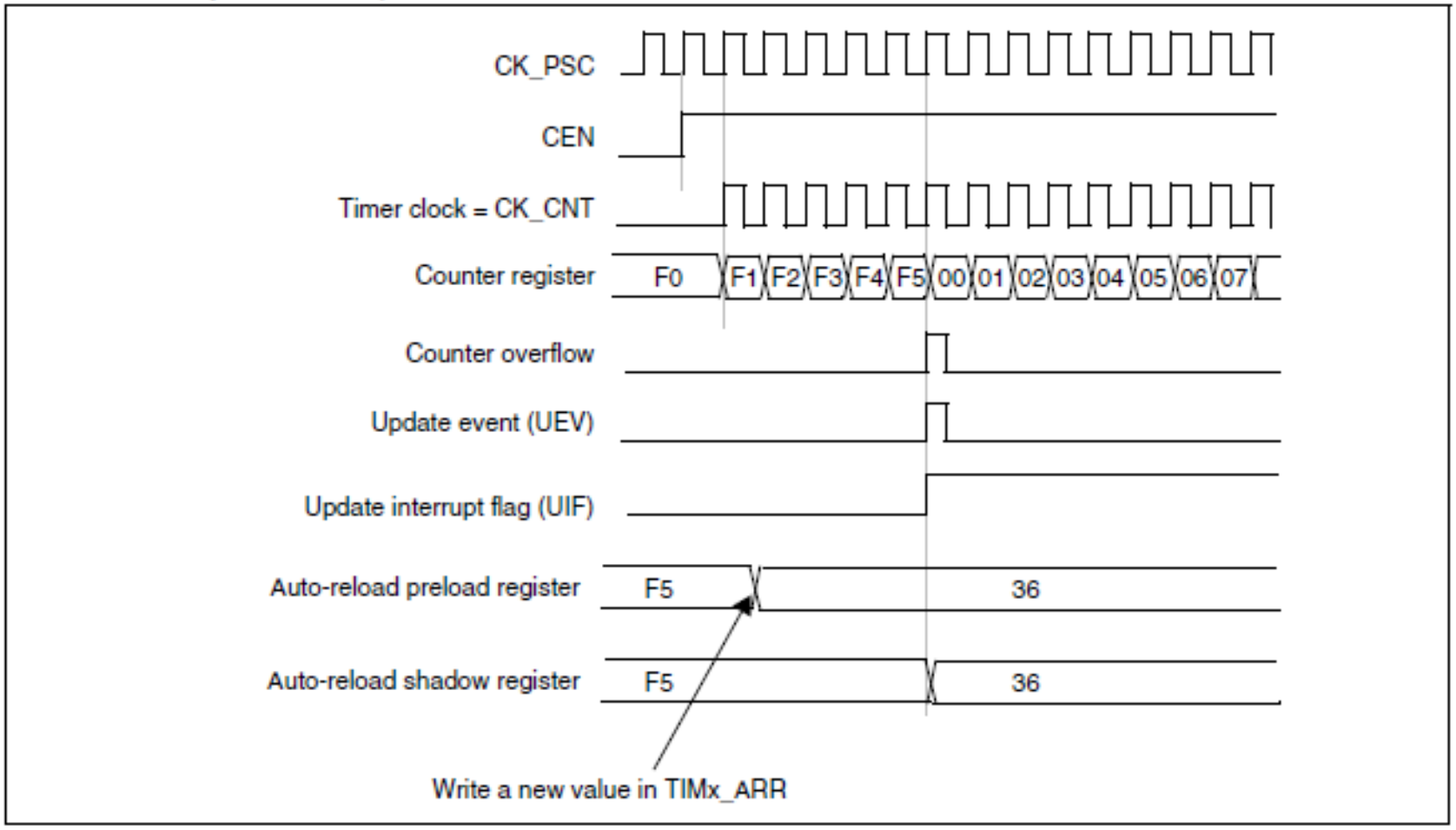
Counter timing diagram (divided by N)



Counter timing diagram when ARPE=0



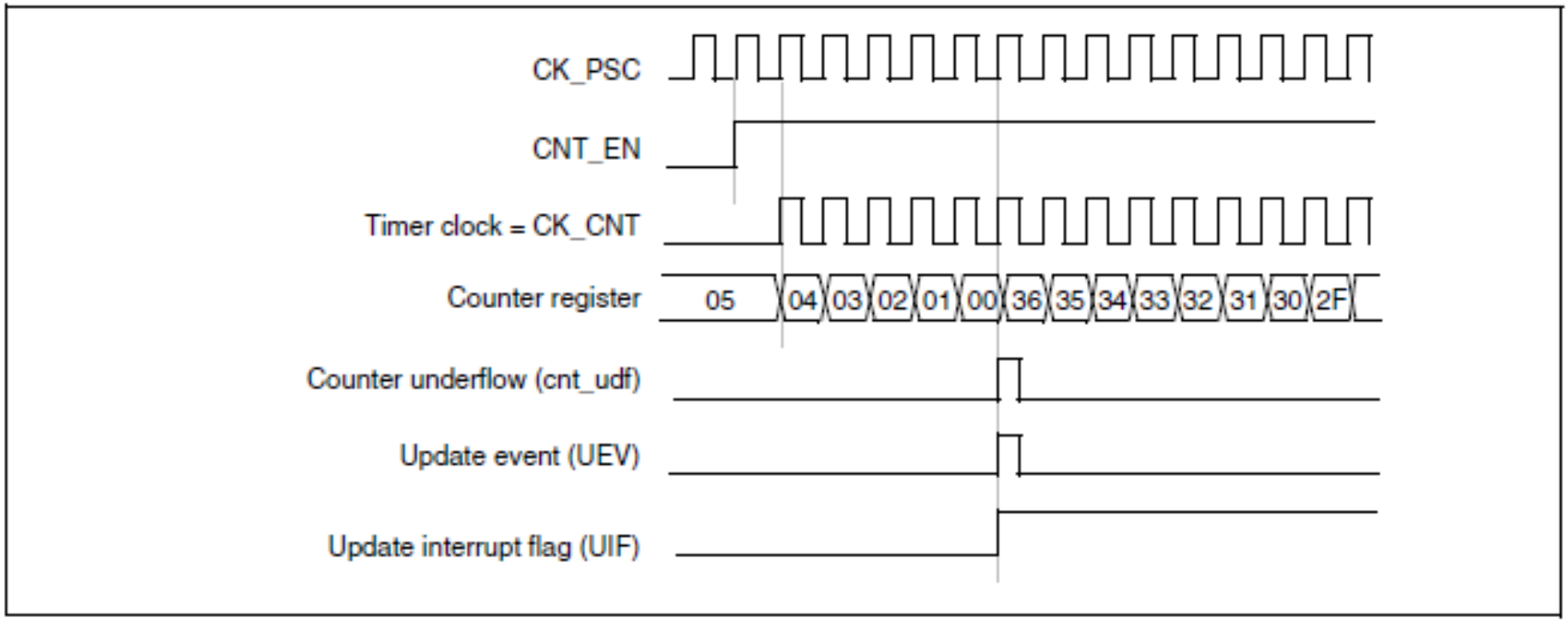
Counter timing diagram when ARPE=1



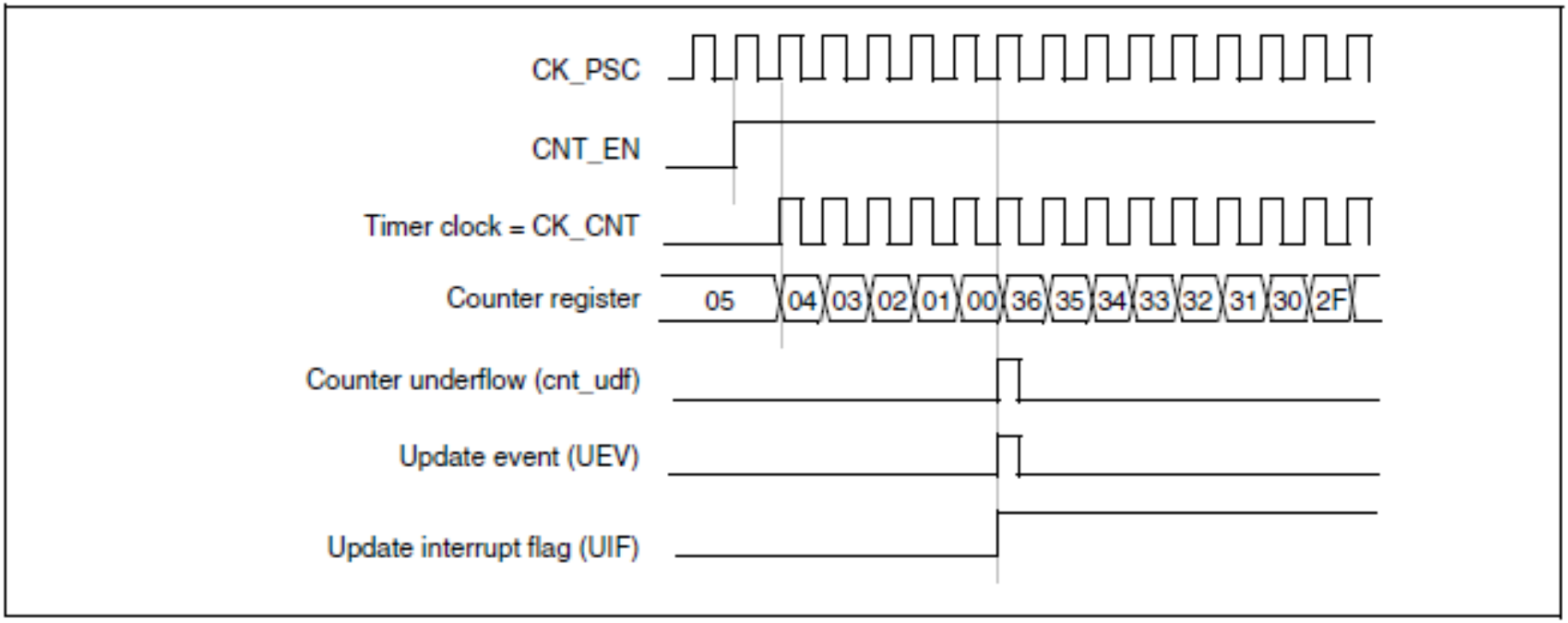
Down counting

- Counter count from auto-reload value down to 0
- After reaching 0, it will restart from auto-reload value and generate a count underflow event
- If the repetition counter is used, the update event (UEV) is generated, and will repeat for the number of times programmed in repetition register

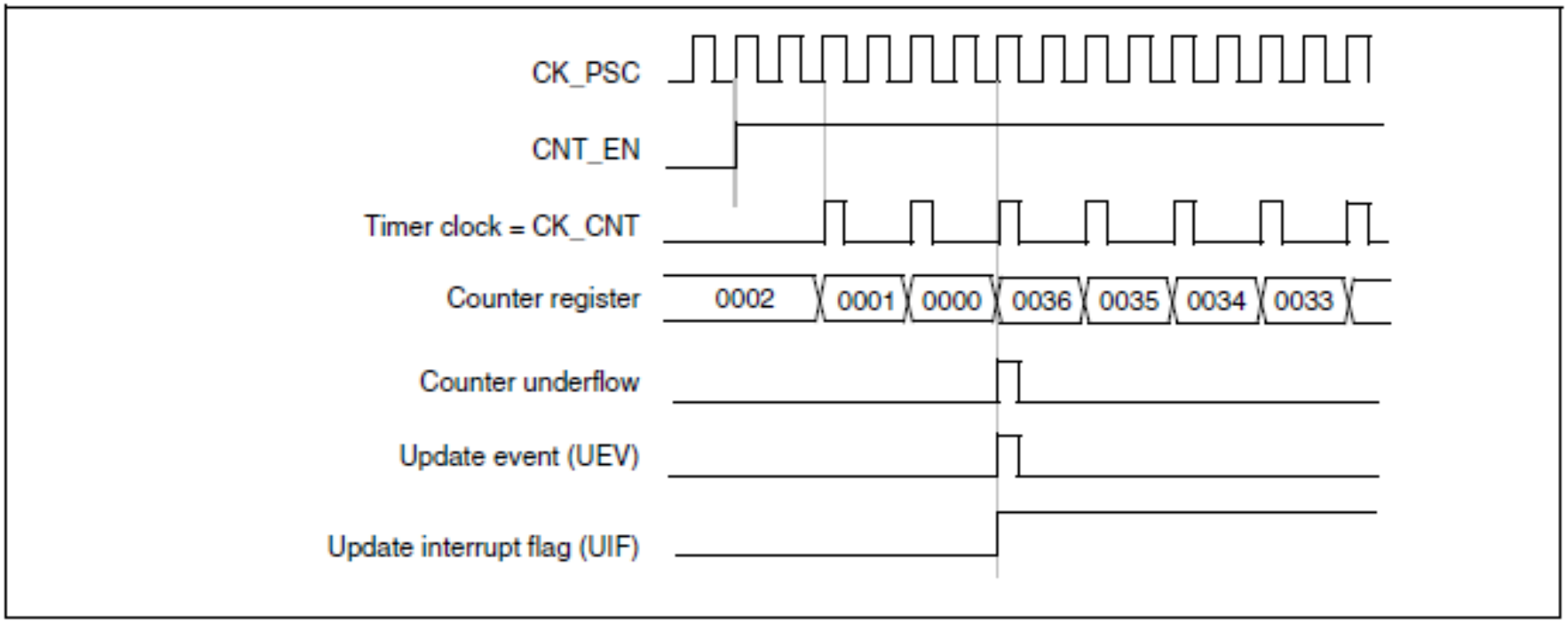
Down counter timing diagram (divided by 1)



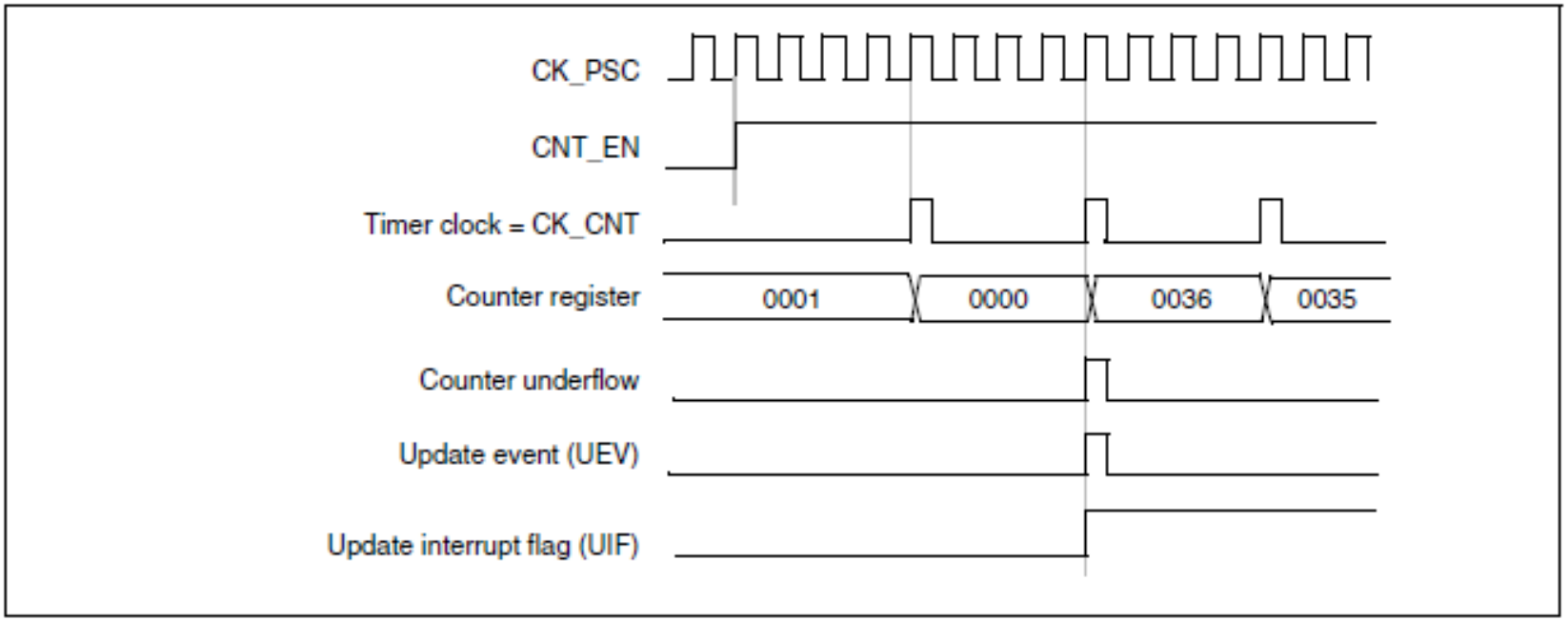
Down counter timing diagram (divided by 1)



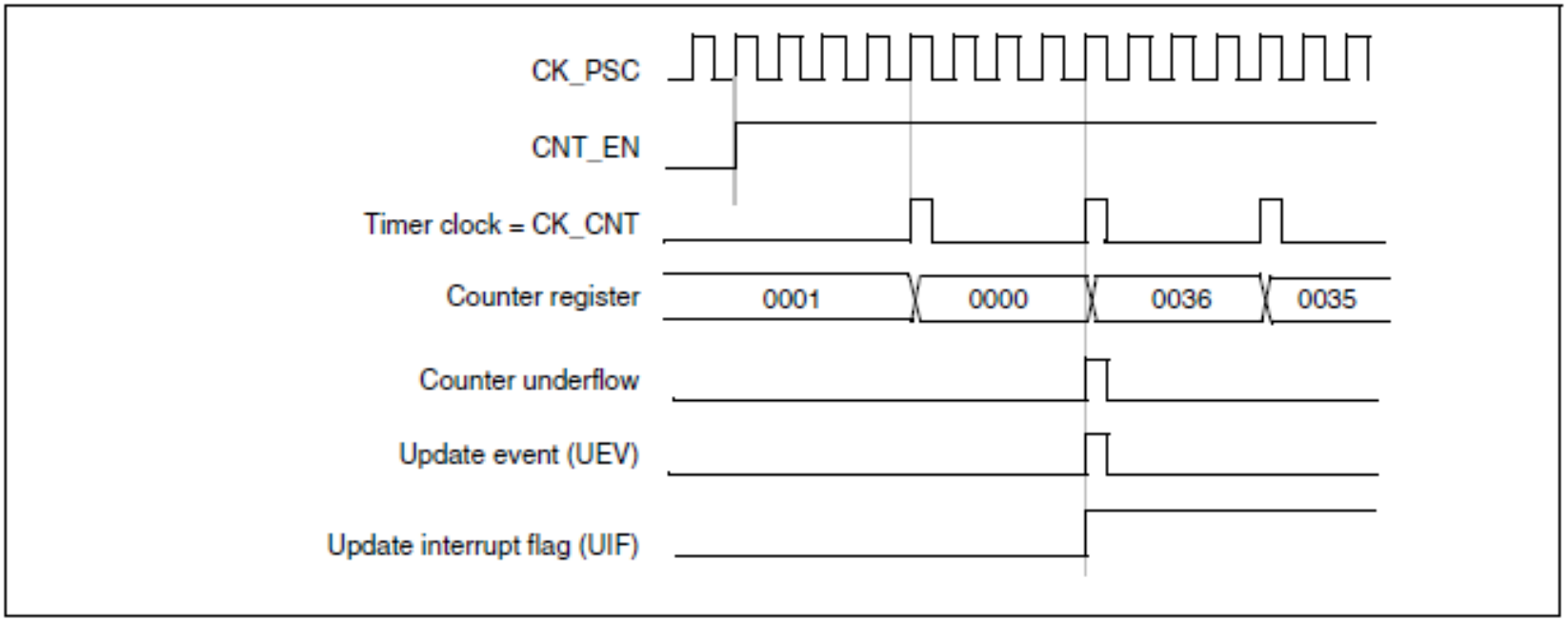
Down counter timing diagram (divided by 2)



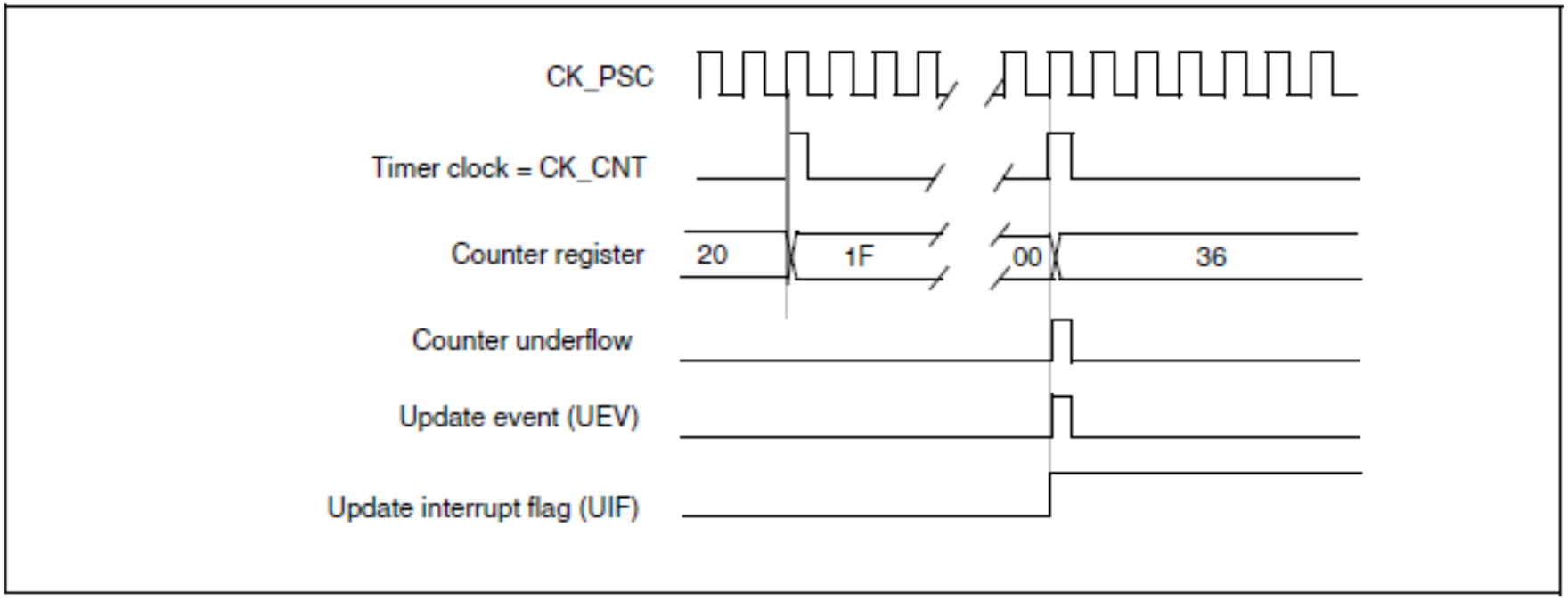
Down counter timing diagram (divided by 4)



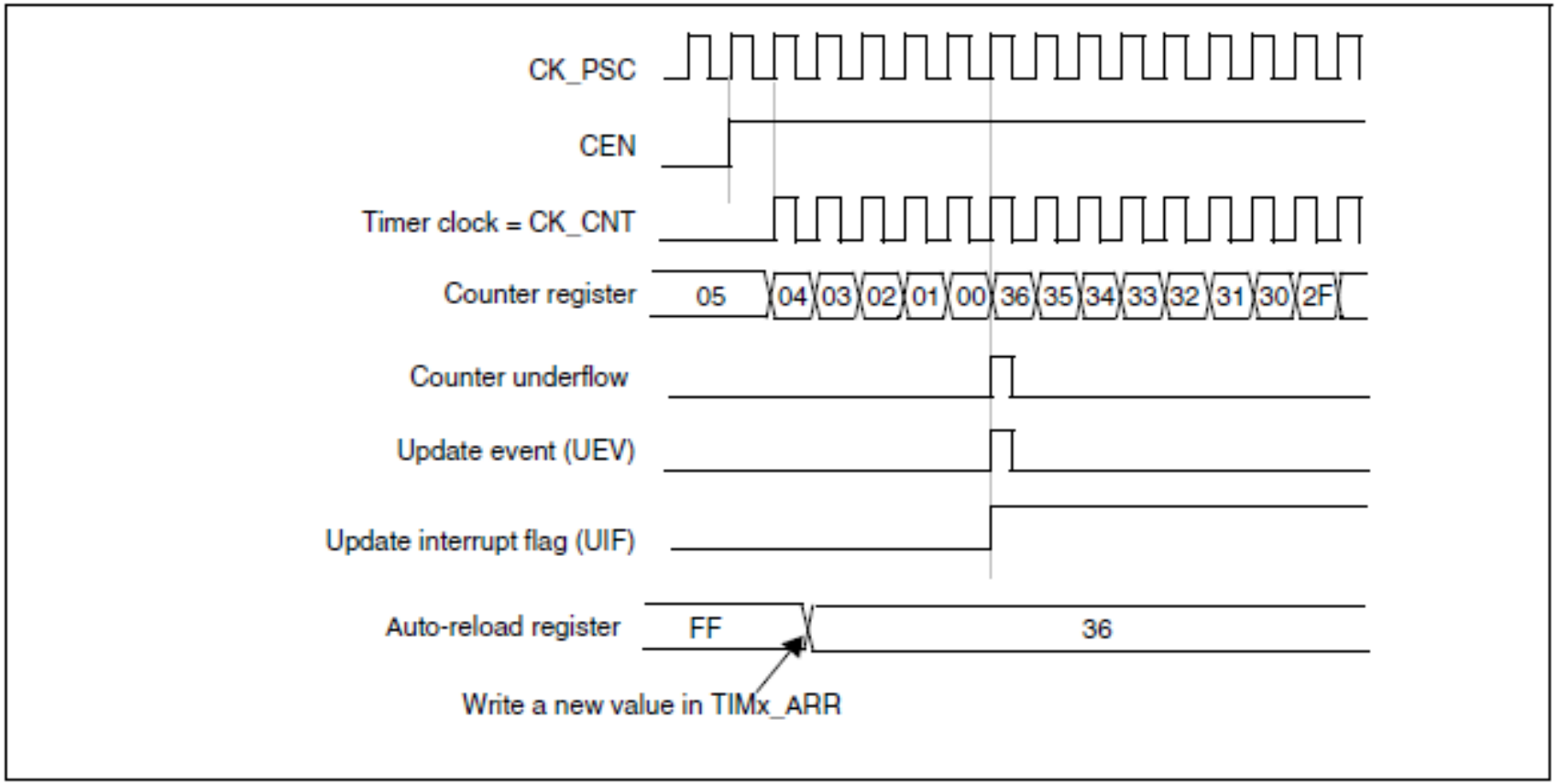
Down counter timing diagram (divided by 4)



Down counter timing diagram (divided by N)



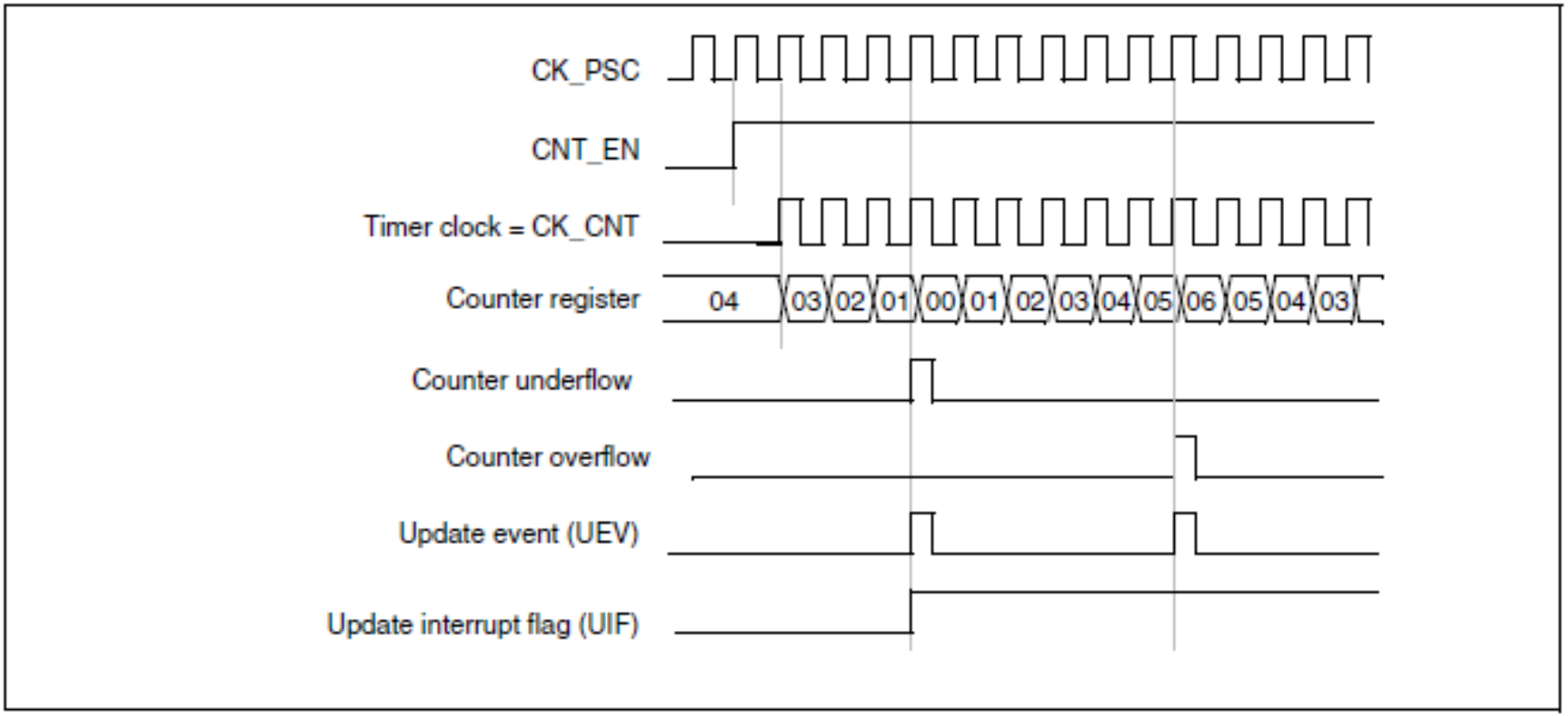
Counter timing diagram with auto-reload register update



Up/Down counting mode

- The counter counts from 0 to the auto-reload value -1 then generate a counter overflow event
- Then, it counts from auto reload value to 0, then generate a counter underflow event
- Then, it restarts from 0

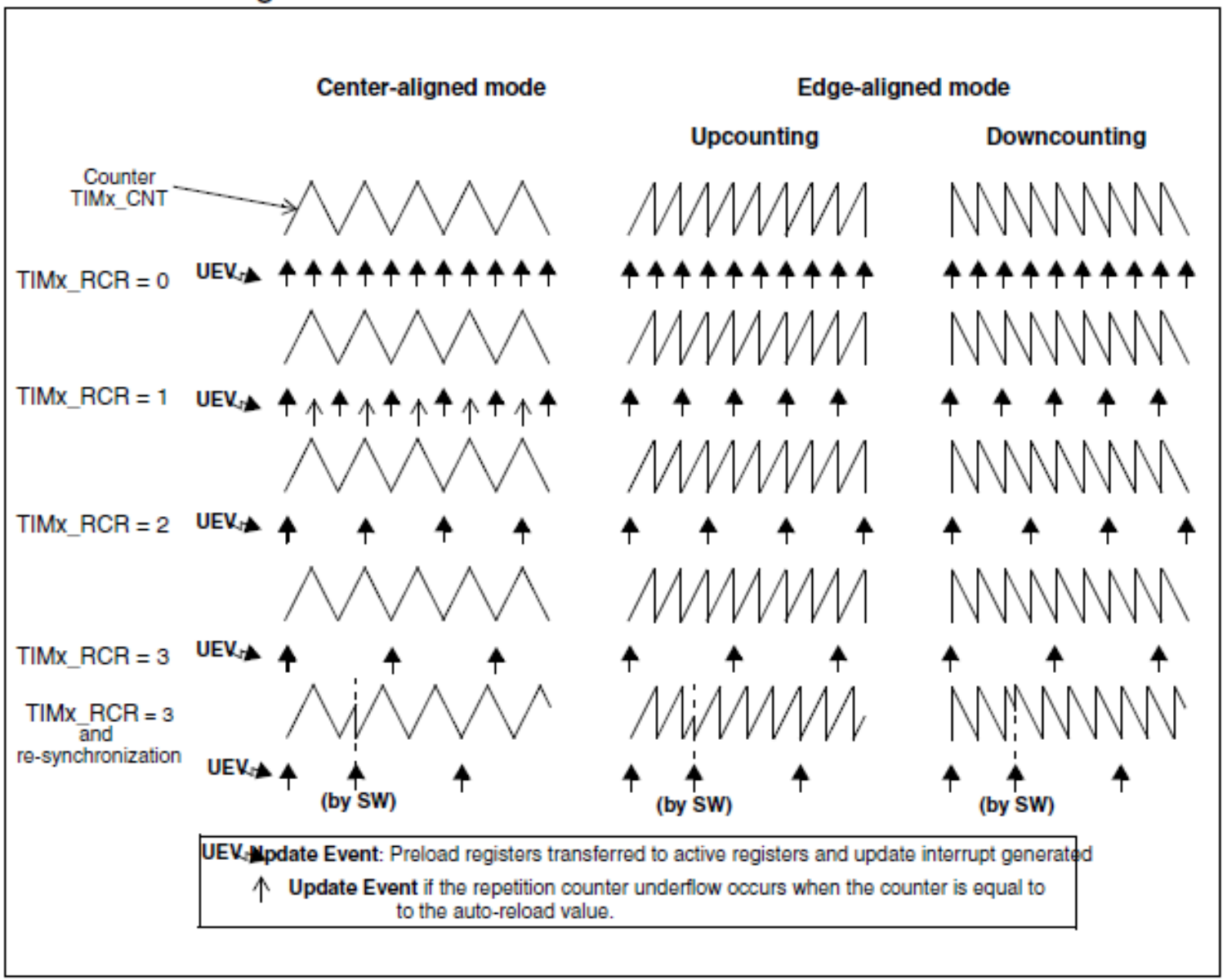
Counter timing diagram with clock divided by 1



Repetition counter

- It mainly uses in PWM
- It controls how the update event (UEV) or interrupt is generated
- It will actually generated with TIM1_RCR (repetition counter) is counted to 0
- TIM1_RCE will decrement when
  - At each counter overflow in upcounting mode
  - At each counter underflow in downcounting mode
  - At each counter overflow and then underflow in up-down counting mode

TIM1 RCR update



PWM mode

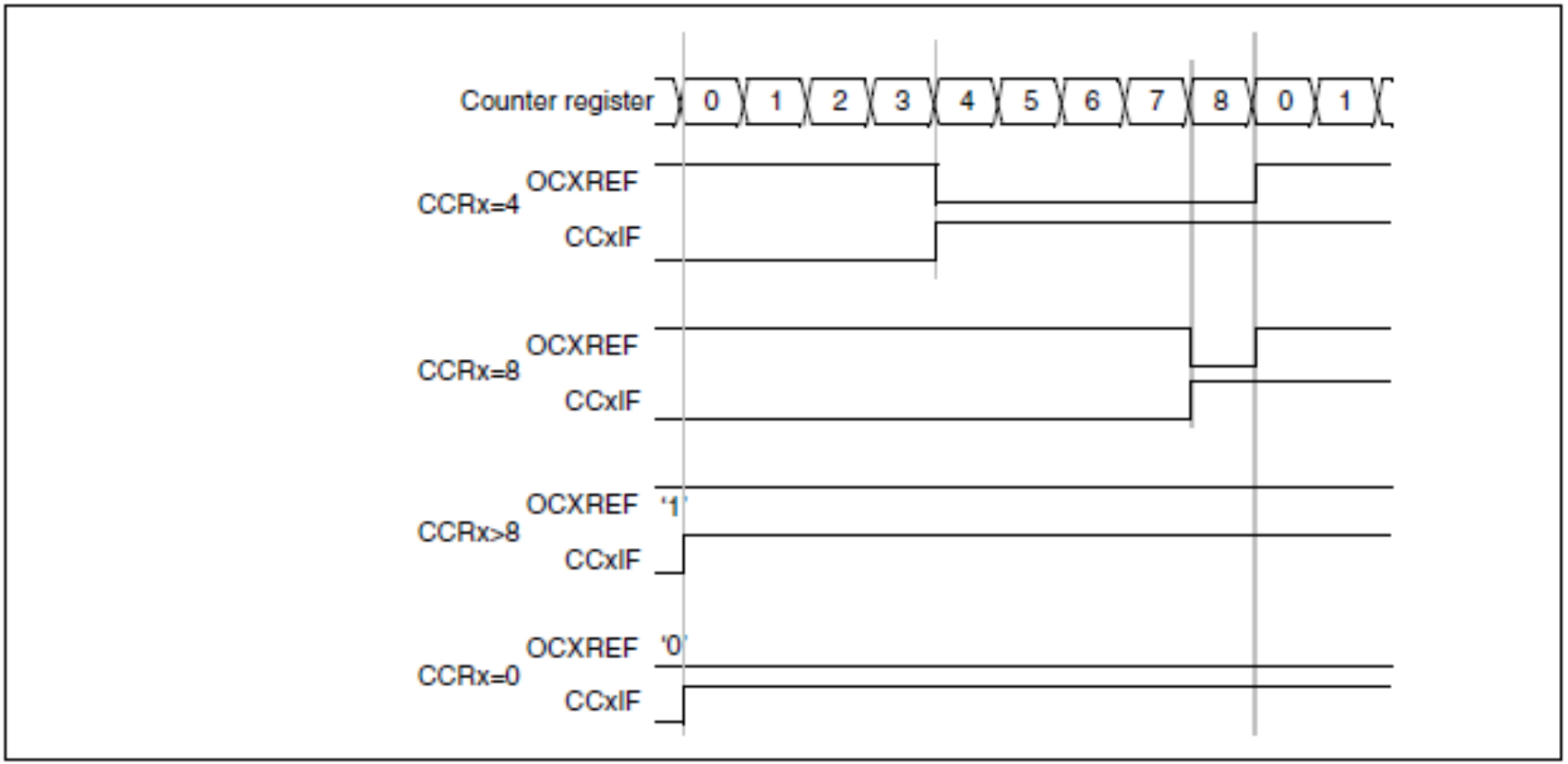
- It can generate a signal with a frequency determined by the value of TIMx_ARR and a duty cycle determined by TIMx_CCRx register
- PWM mode can be selected independently for each channel
- Timer is generated PWM in edge-aligned mode or center aligned mode

PWM edge-aligned mode

- Mode 1: set OCxM bit to “110”. During the up count, PWM will be 1 when TIMx_CNT < TIMx_CCRx

- Mode 2: set OCxM bit to “111”. During the up count, PWM will be 0 when TIMx_CNT < TIMx_CCRx

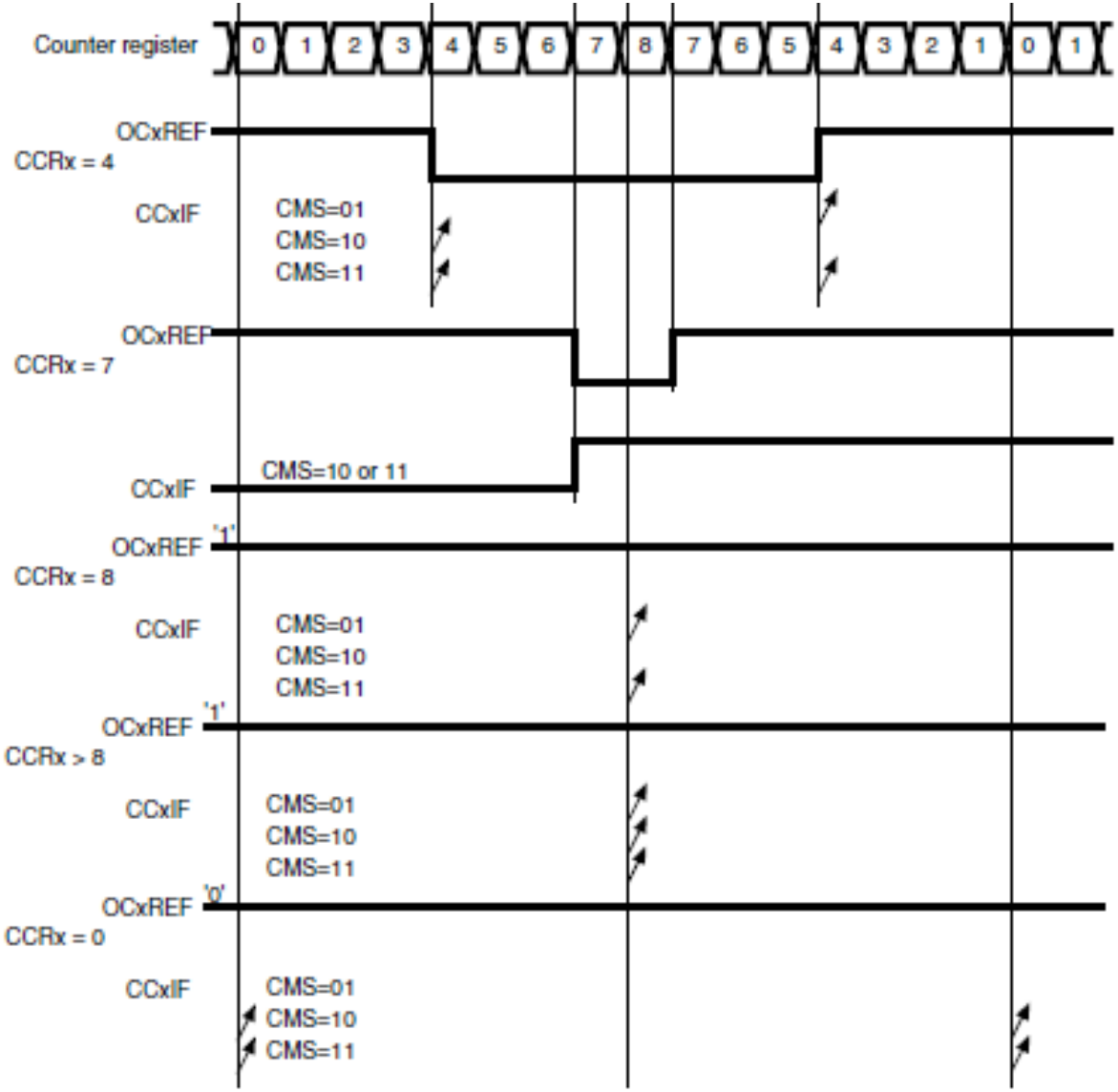
Edge aligned PWM waveform



PWM center aligned mode

- Mainly used in Up-Down counting mode
- To enable this mode, CMS bits are set to any value except “00”

Center align PWM waveform



Example of ARM code

```
int main(){
   int pulse1 = 1000/2;
   int pulse1_change_flag=0;
   flag = 0;
   RCC_setup();      // RCC Configuration
   GPIO_setup();     // GPIO Configuration
   TIMER_setup();    // TIMER Configuration
   while(1) {
        TIM_OCInitStructure_c1.TIM_Pulse = pulse1; // set duty cycle
        TIM_OCInit(TIM4, &TIM_OCInitStructure_c1);
}
}
```

```
void GPIO_setup(){
  GPIO_InitTypeDef GPIO_InitStructure;   // Enable GPIOA GPIOB clock
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA|RCC_APB2Periph_GPIOB
    |RCC_APB2Periph_AFIO,ENABLE);
  GPIO_InitStructure.GPIO_Pin   =  GPIO_Pin_10 ;
  GPIO_InitStructure.GPIO_Speed =  GPIO_Speed_50MHz;
  GPIO_InitStructure.GPIO_Mode  =  GPIO_Mode_IN_FLOATING;
  GPIO_Init(GPIOA, &GPIO_InitStructure);
  GPIO_InitStructure.GPIO_Pin   =  GPIO_Pin_0 | GPIO_Pin_1 | GPIO_Pin_9 ;
  GPIO_InitStructure.GPIO_Mode  =  GPIO_Mode_AF_PP;
  GPIO_Init(GPIOA, &GPIO_InitStructure);
  GPIO_InitStructure.GPIO_Pin   =  GPIO_Pin_10 ;
  GPIO_InitStructure.GPIO_Speed =  GPIO_Speed_50MHz;
  GPIO_InitStructure.GPIO_Mode  =  GPIO_Mode_IN_FLOATING;
  GPIO_Init(GPIOB, &GPIO_InitStructure);
  GPIO_InitStructure.GPIO_Pin   =  GPIO_Pin_6 | GPIO_Pin_7 | GPIO_Pin_8|
    GPIO_Pin_9 ;
  GPIO_InitStructure.GPIO_Mode  =  GPIO_Mode_AF_PP;
  GPIO_Init(GPIOB, &GPIO_InitStructure);
GPIO_PinRemapConfig(GPIO_Remap_SWJ_Disable, ENABLE);
 /* Disable the Serial Wire Jtag Debug Port SWJ-DP */
}
```

```
void TIMER_setup(){
   TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;
   RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4,ENABLE); // Enable TIM4 clock

// Time base configuration
   TIM_TimeBaseStructure.TIM_Period = 1999;//19999;
   TIM_TimeBaseStructure.TIM_Prescaler = 719;//179;
   TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;
   TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
   TIM_TimeBaseInit(TIM4, &TIM_TimeBaseStructure);

   /* PWM1 Mode configuration: Channel1 */
   TIM_OCInitStructure_c1.TIM_OCMode = TIM_OCMode_PWM1;
   TIM_OCInitStructure_c1.TIM_Channel = TIM_Channel_1;
   TIM_OCInitStructure_c1.TIM_Pulse = (int)(PWM_Period / 2);
   TIM_OCInitStructure_c1.TIM_OCPolarity = TIM_OCPolarity_High;
   TIM_OCInit(TIM4, &TIM_OCInitStructure_c1);
```

```
/* PWM1 Mode configuration: Channel2 */
  TIM_OCInitStructure_c2.TIM_OCMode = TIM_OCMode_PWM1;
  TIM_OCInitStructure_c2.TIM_Channel = TIM_Channel_2;
  TIM_OCInitStructure_c2.TIM_Pulse = (int)(PWM_Period / 2);
  TIM_OCInitStructure_c2.TIM_OCPolarity = TIM_OCPolarity_High;
  TIM_OCInit(TIM4, &TIM_OCInitStructure_c2);

  /* PWM1 Mode configuration: Channel3 */
  TIM_OCInitStructure_c3.TIM_OCMode = TIM_OCMode_PWM1;
  TIM_OCInitStructure_c3.TIM_Channel = TIM_Channel_3;
  TIM_OCInitStructure_c3.TIM_Pulse = (int)(PWM_Period / 2);
  TIM_OCInitStructure_c3.TIM_OCPolarity = TIM_OCPolarity_High;
  TIM_OCInit(TIM4, &TIM_OCInitStructure_c2);

  TIM_OC1PreloadConfig(TIM4, TIM_OCPreload_Enable);
  TIM_ARRPreloadConfig(TIM4, ENABLE);
  TIM_Cmd(TIM4, ENABLE);

}
```

Questions?