南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Embedded System and Microcomputer Principle

LAB7 Timer Interrupt

2024 Fall
wangq9@mail.sustech.edu.cn

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

CONTENTS

01

General-purpose Timers Description

1. General-purpose Timers Description -- Timers of STM32F103RCT6

- There are three kinds of timer in STM32: advanced timers, general purpose timers and basic timers
  - General-purpose timers are the most common timers in STM32, which supports PWM generation, input capture, time-base generation(update interrupt) and output compare
  - Basic timers don’t have IOs channels for input capture/PWM generation so that they are only used in time-base generation purposes.
  - Advanced timers have more features than general purpose timers like complementary PWM generation.

1. General-purpose Timers Description

-- TIM2 to TIM5 main features

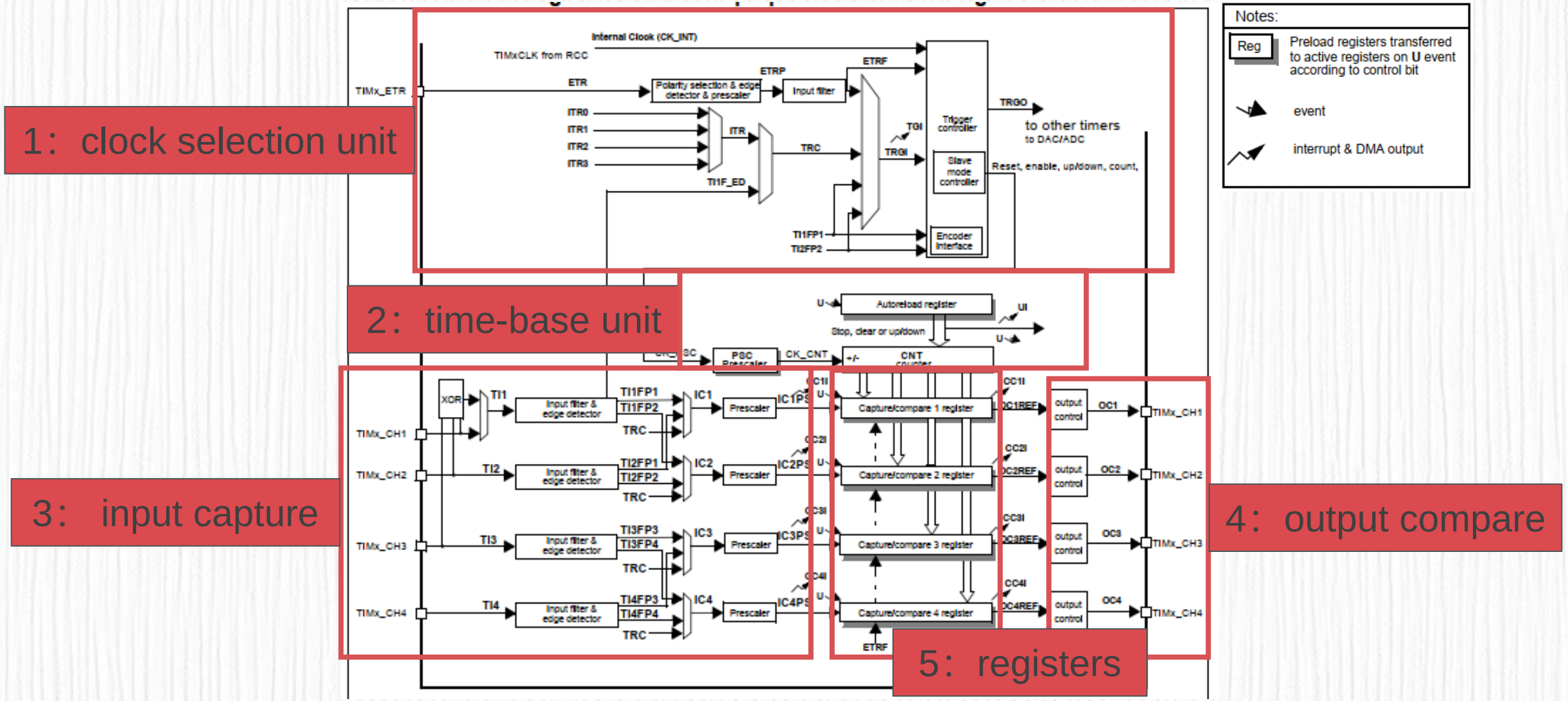
- 16-bit up, down, up/down auto-reload counter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65536.
- Up to 4 independent channels for
  - Input capture
  - Output compare
  - PWM generation (Edge- and Center-aligned modes)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers.

1. General-purpose Timers Description

-- TIM2 to TIM5 main features(continued)

- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

1. General-purpose Timers Description
-- General-purpose timer block diagram



1. General-purpose Timers Description

-- Clock selection

- The counter clock can be provided by the following clock sources
  - Internal clock (CK_INT)
  - External clock mode1: external input pin (TIx)
  - External clock mode2: external trigger input (ETR)
  - Internal trigger inputs (ITRx): using one timer as prescaler for another timer, for example, you can configure Timer1 to act as a prescaler for Timer 2

1. General-purpose Timers Description

-- Time-base unit

- The main block of the programmable timer is a 16-bit counter with its related auto-reload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.
- The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.
- The time-base unit includes:
  - Counter Register (TIMx_CNT)
  - Prescaler Register (TIMx_PSC)
  - Auto-Reload Register (TIMx_ARR)

1. General-purpose Timers Description

-- TIMx_ARR

- The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

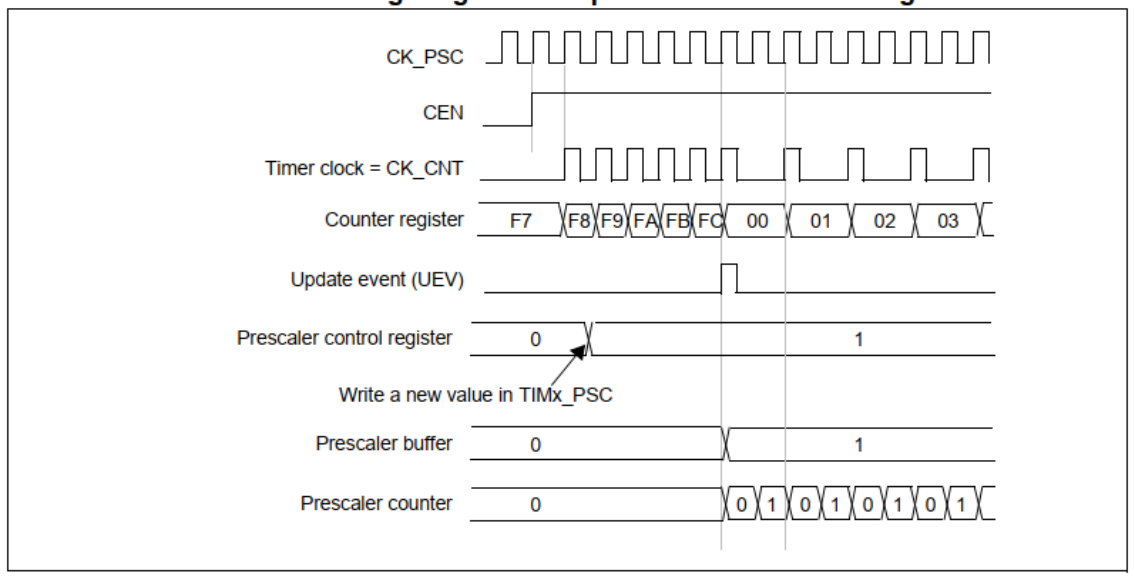
1. General-purpose Timers Description

-- Prescaler description

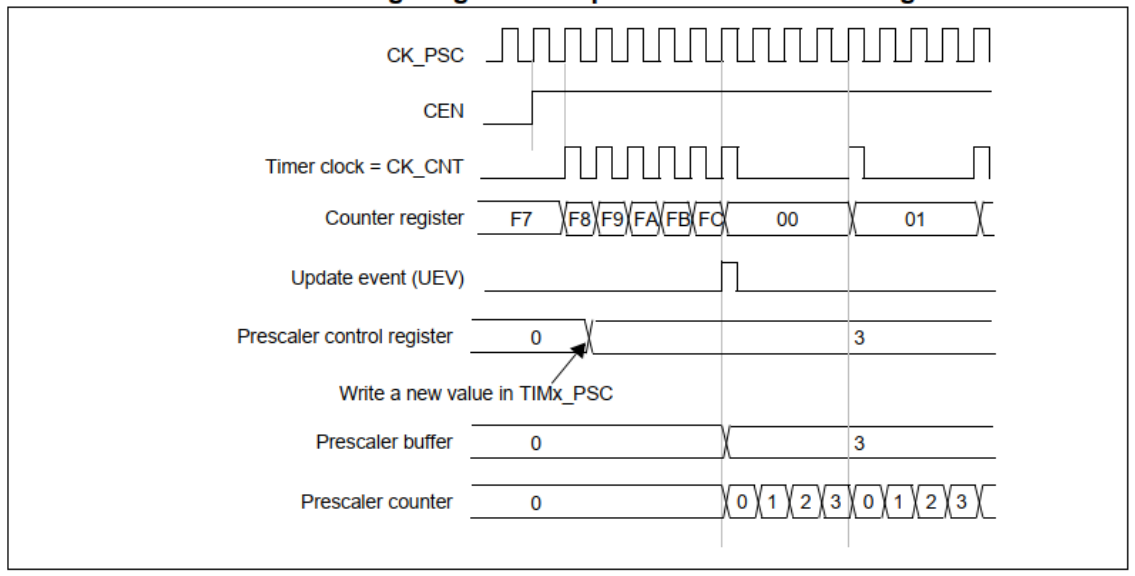
- The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register)
- It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

1. General-purpose Timers Description -- Prescaler description(continued)

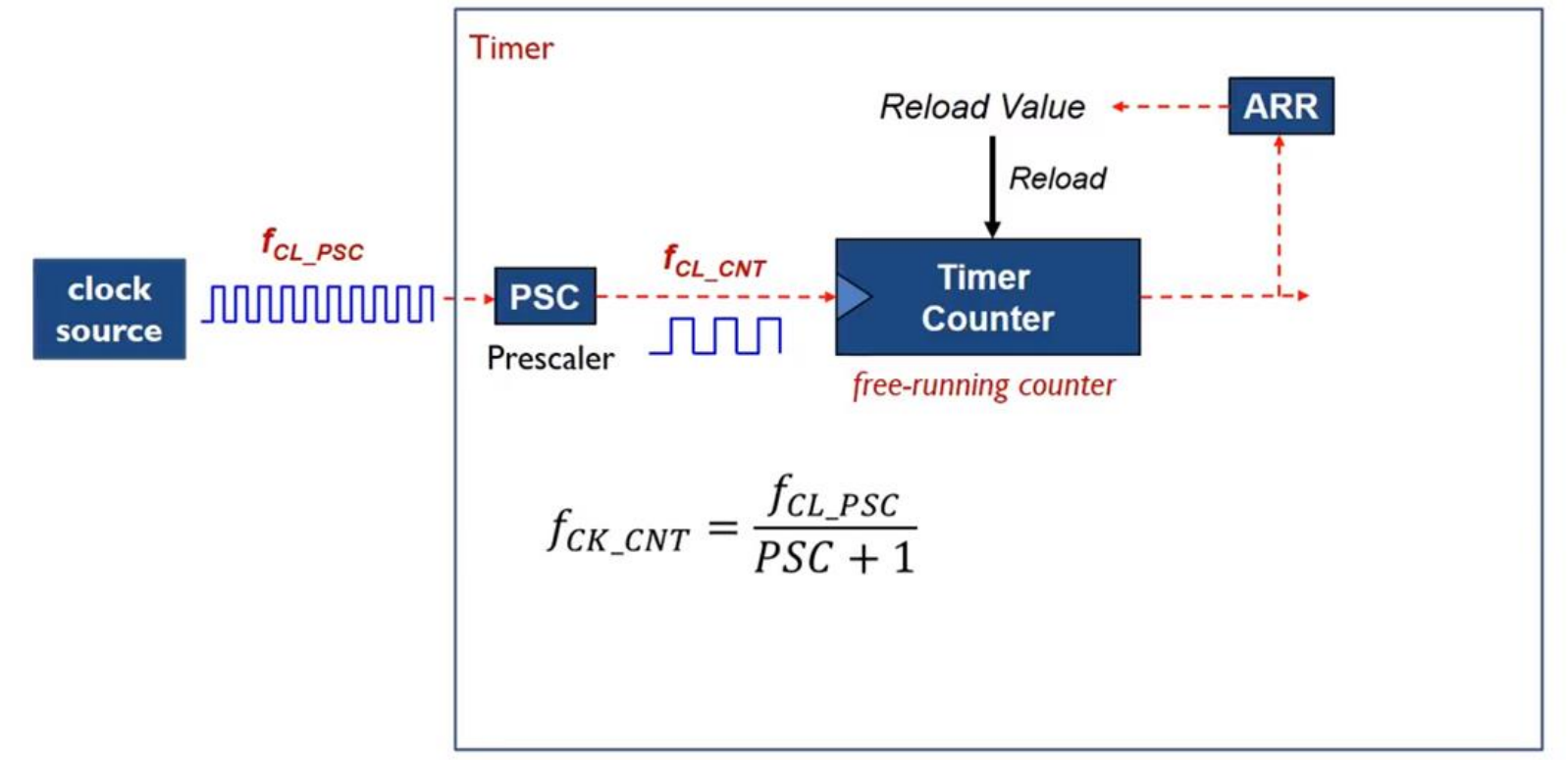
Counter timing diagram with prescaler division change from 1 to 2



Counter timing diagram with prescaler division change from 1 to 4



1. General-purpose Timers Description -- CK_CNT

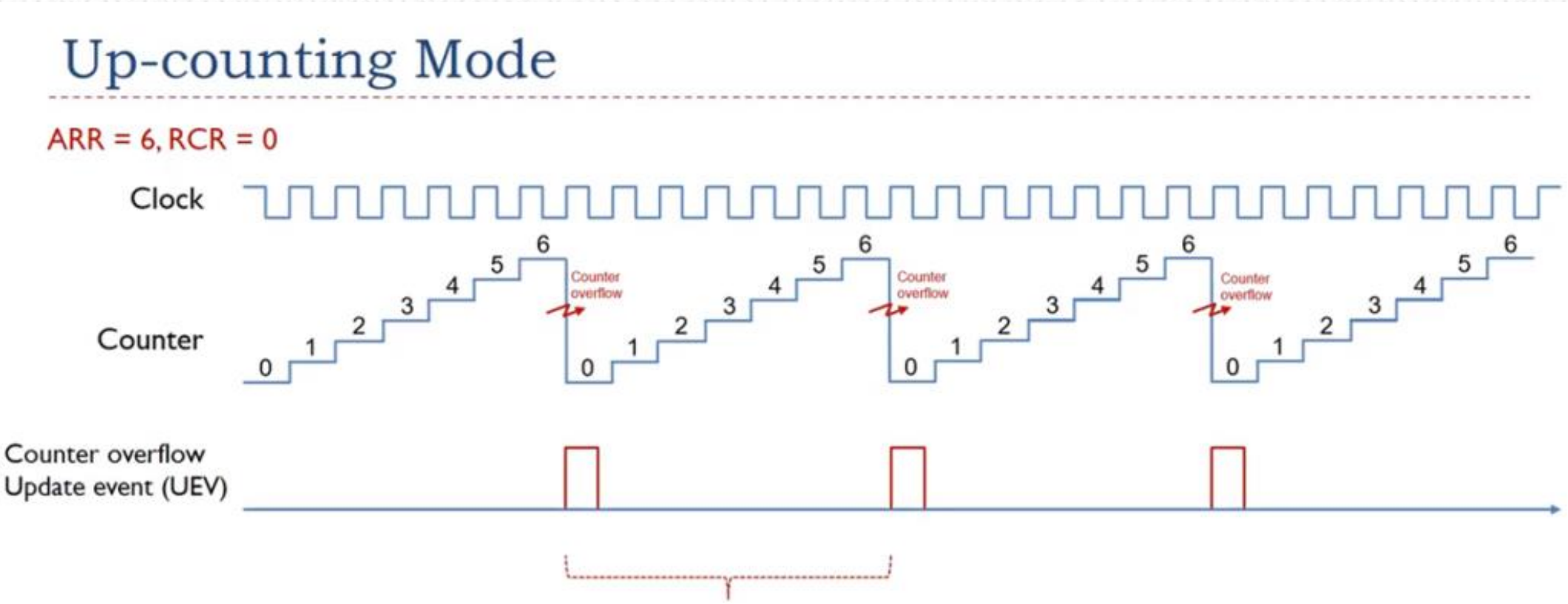


1. General-purpose Timers Description

-- Up-counting mode

- In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.
- An Update event can be generated at each counter overflow or by setting the UG bit in the TIMx_EGR register.
- When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set:
  - The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register)
  - The auto-reload shadow register is updated with the preload value (TIMx_ARR)

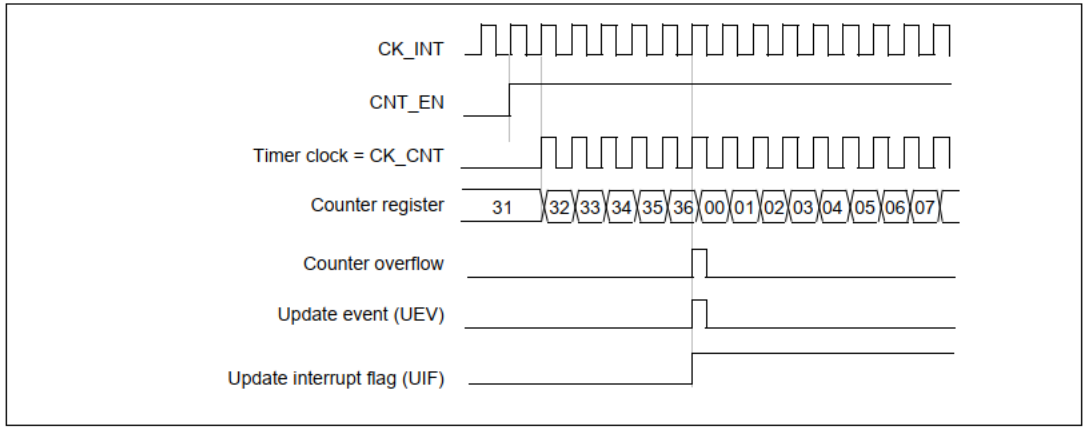
1. General-purpose Timers Description -- Up-counting mode(continued)



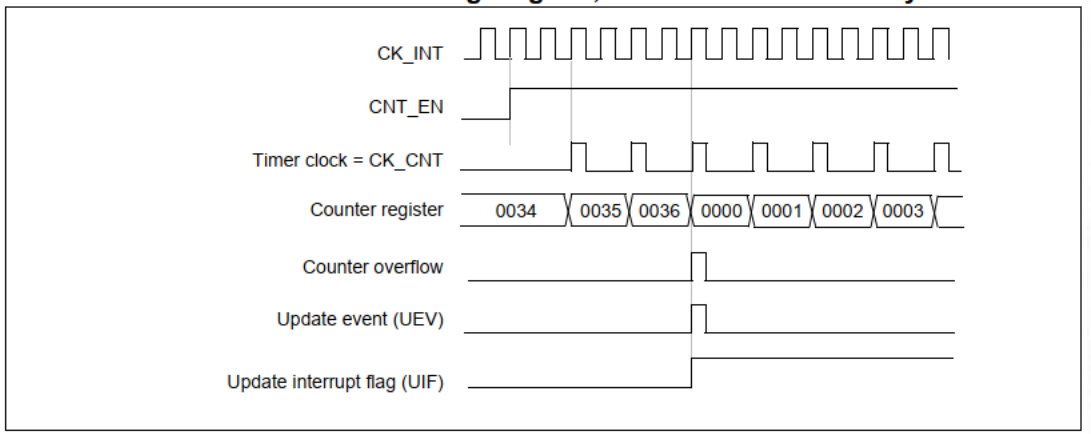
1. General-purpose Timers Description -- Up-counting mode(continued)

南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

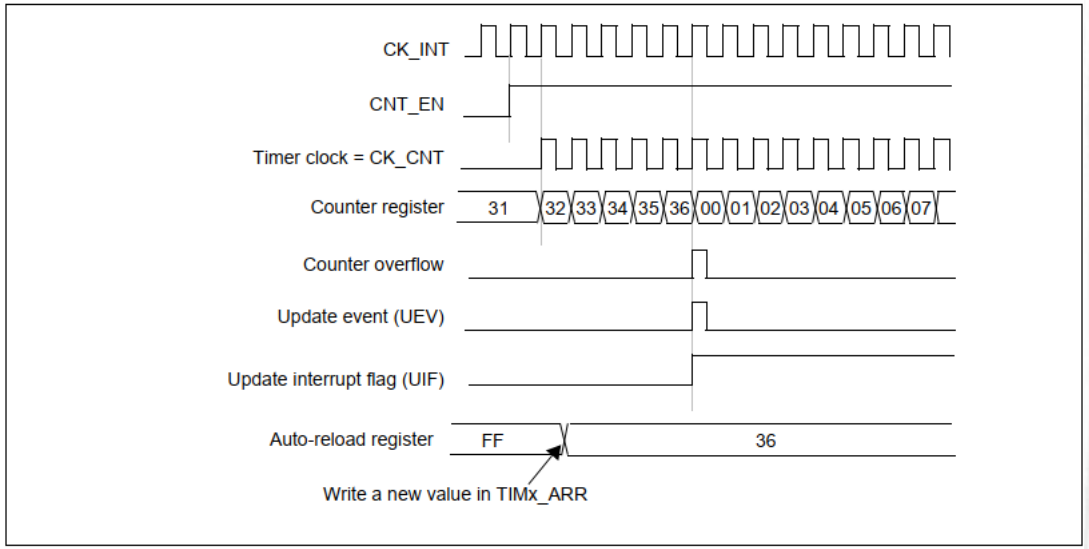
Counter timing diagram, internal clock divided by 1



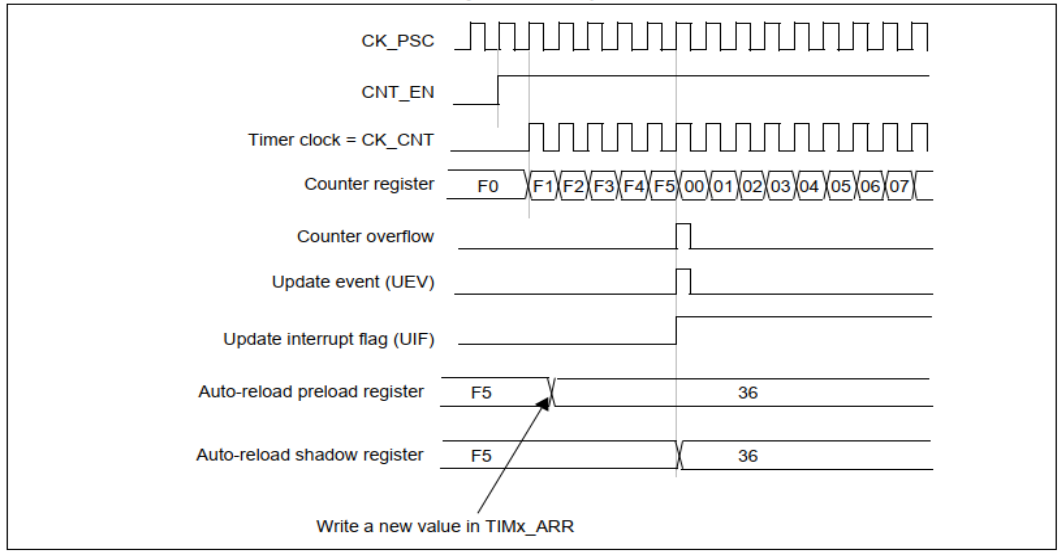
Counter timing diagram, internal clock divided by 2



Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded)



Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded)

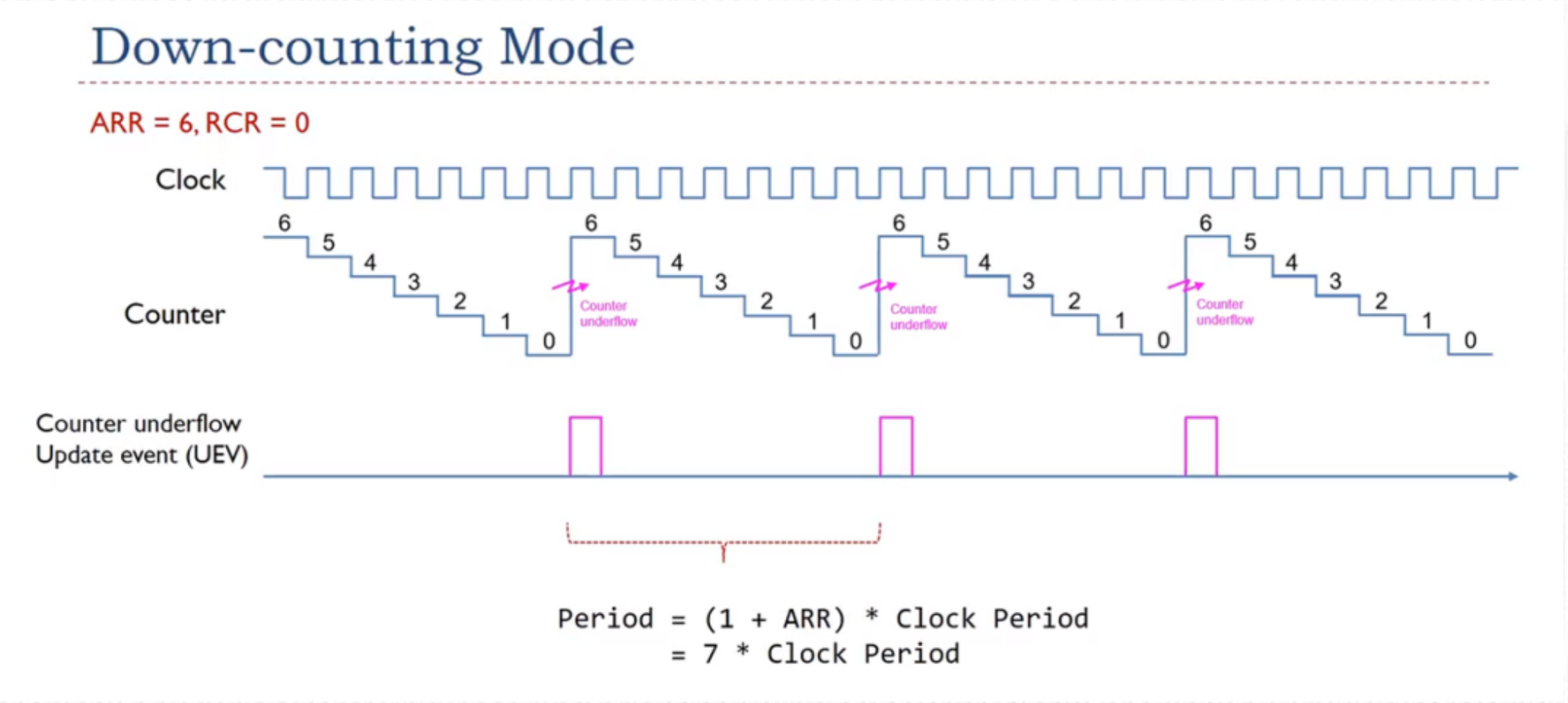


1. General-purpose Timers Description

-- Down-counting mode

- In downcounting mode, the counter counts from the auto-reload value down to 0, then restarts from the auto-reload value and generates a counter underflow event.
- An Update event can be generate at each counter underflow or by setting the UG bit in the TIMx_EGR register.
- When an update event occurs, all the registers are updated and the update flag is set :
  - The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
  - The auto-reload active register is updated with the preload value (content of the TIMx_ARR register).

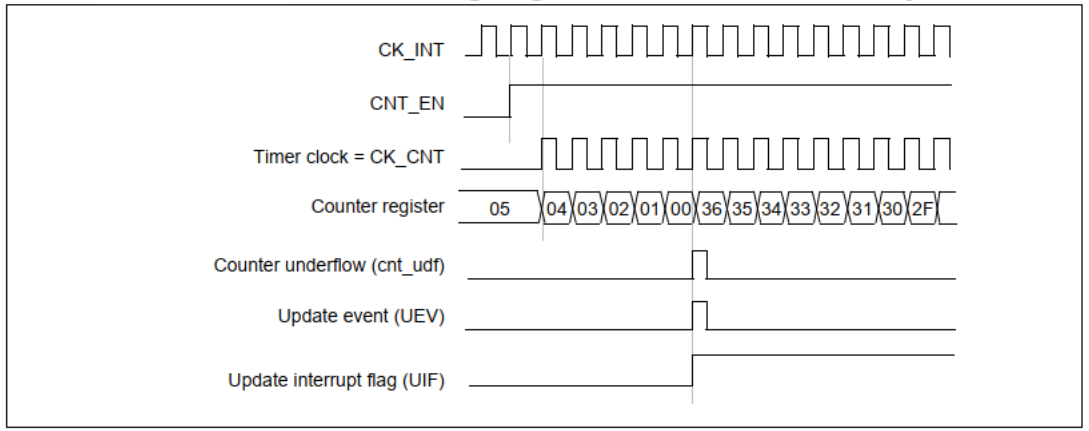
1. General-purpose Timers Description -- Down-counting mode



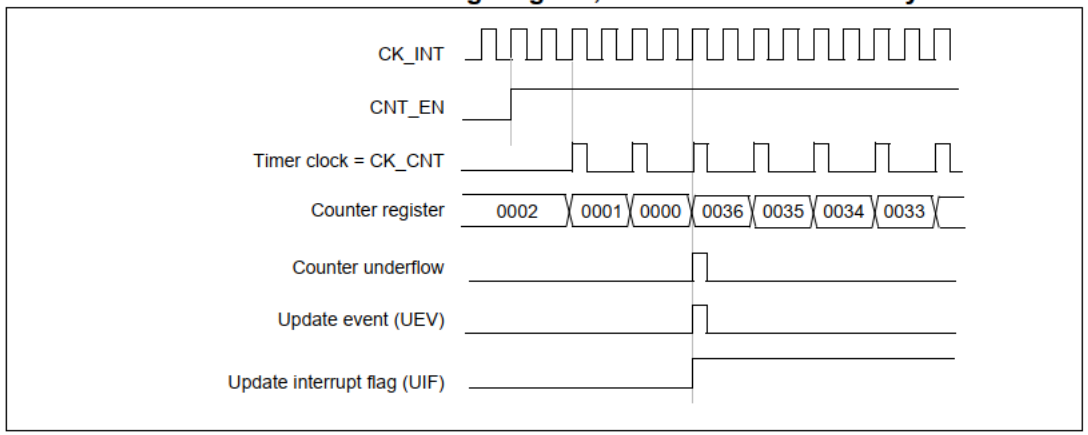
1. General-purpose Timers Description -- Down-counting mode

南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

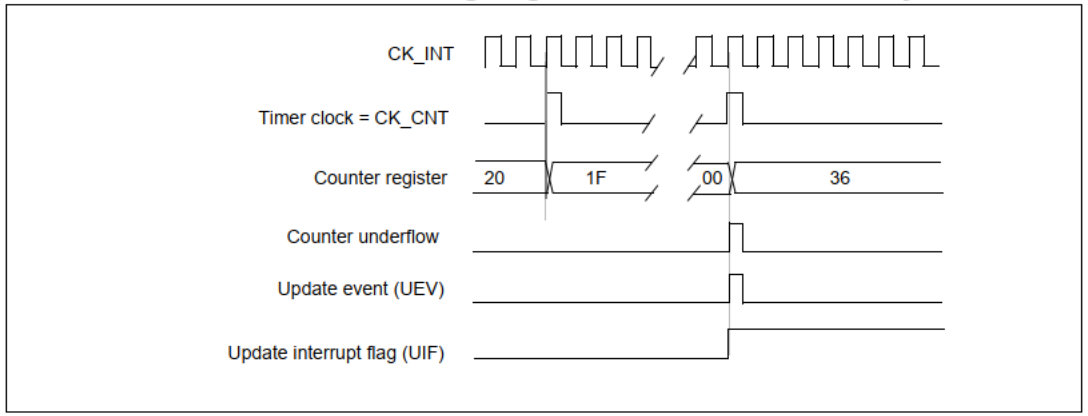
Counter timing diagram, internal clock divided by 1



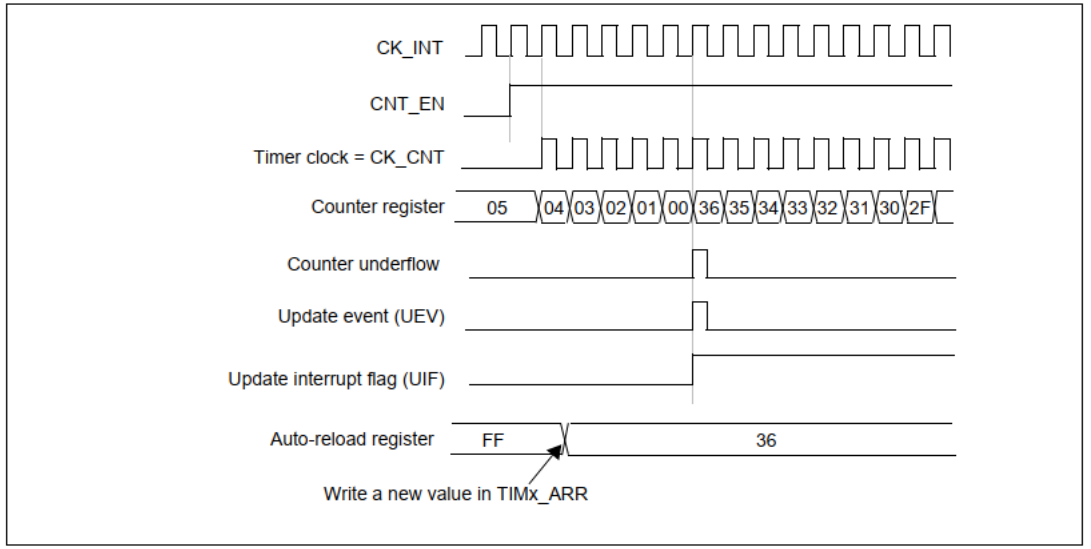
Counter timing diagram, internal clock divided by 2



Counter timing diagram, internal clock divided by N



Counter timing diagram, Update event

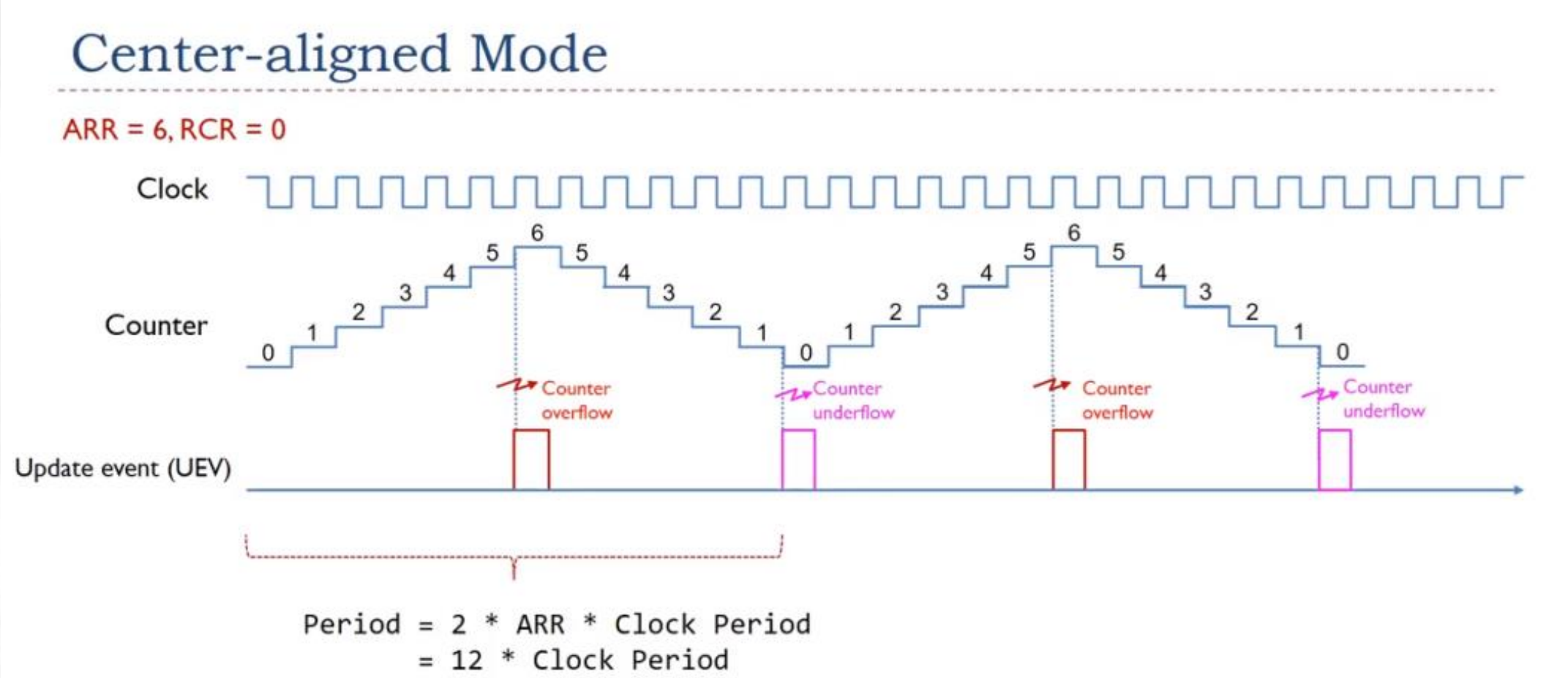


1. General-purpose Timers Description -- Center-aligned mode

南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

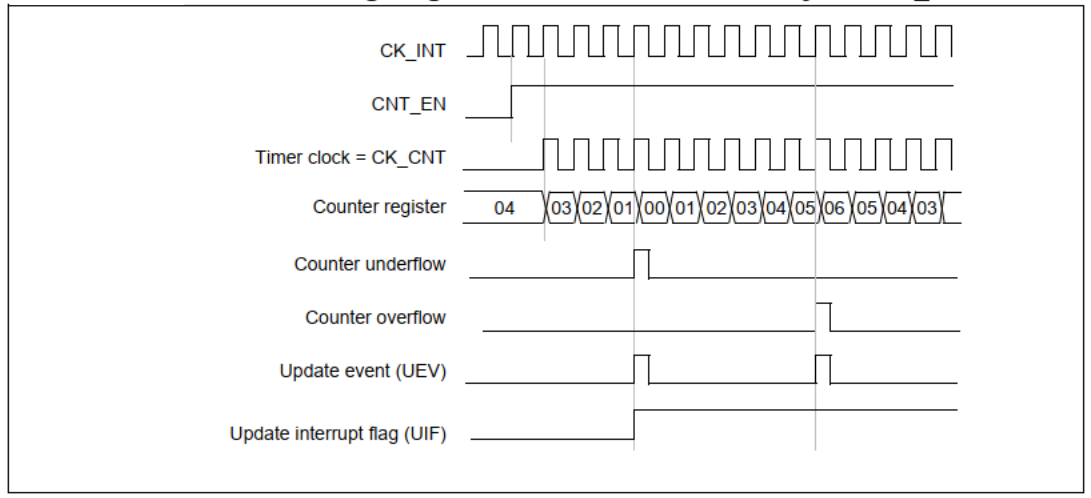
- In center-aligned mode, the counter counts from 0 to the auto-reload value -1, generates a counter overflow event, then counts from the autoreload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.
- In this mode, the direction bit (DIR from TIMx_CR1 register) cannot be written. It is updated by hardware and gives the current direction of the counter.
- The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

1. General-purpose Timers Description -- Center-aligned mode

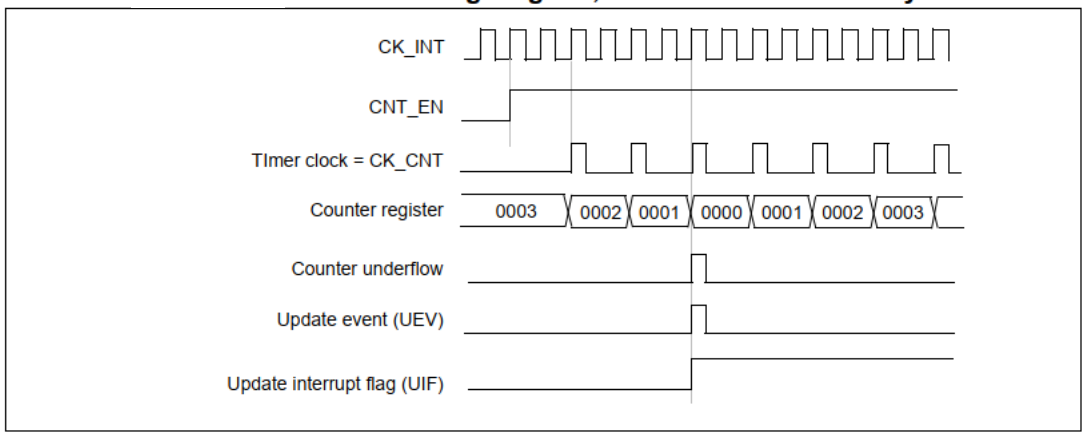


1. General-purpose Timers Description -- Center-aligned mode

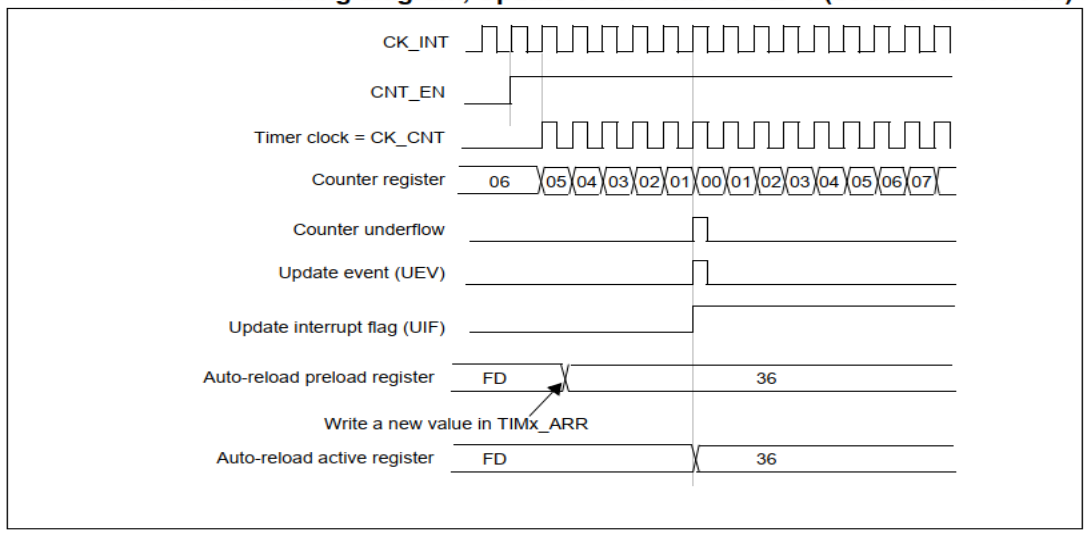
Counter timing diagram, internal clock divided by 1, TIMx_ARR=0x6



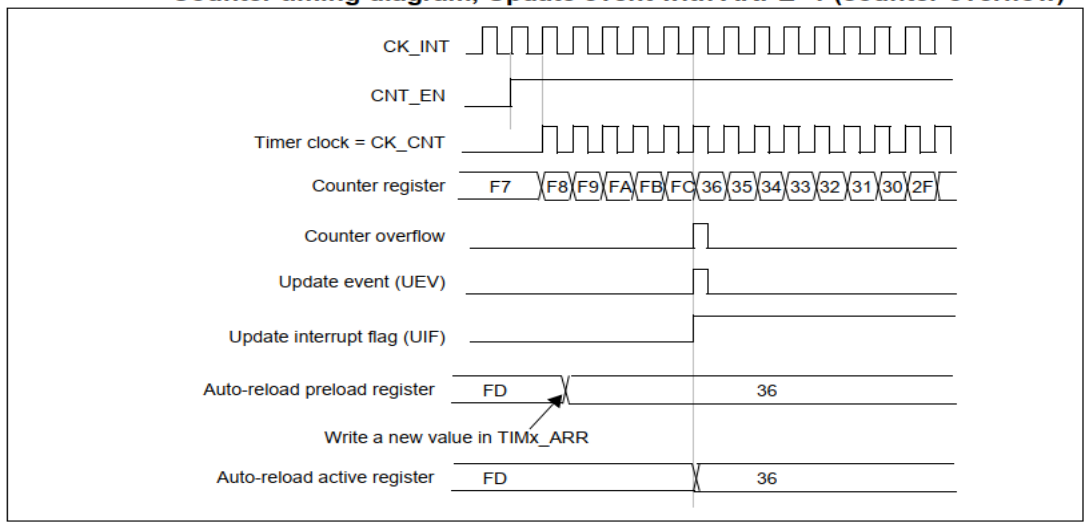
Counter timing diagram, internal clock divided by 2



Counter timing diagram, Update event with ARPE=1 (counter underflow)



Counter timing diagram, Update event with ARPE=1 (counter overflow)



02

TIM2 to TIM5 Registers

2. TIM2 to TIM5 Registers -- TIMx_CNT

- TIMx counter

Address offset: 0x24

Reset value: 0x0000

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

Bits 15:0 **CNT[15:0]**: Counter value

2. TIM2 to TIM5 Registers -- TIMx_PSC

- TIMx prescaler

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency CK_CNT is equal to f_{CK_PSC} / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event.

2. TIM2 to TIM5 Registers -- TIMx_ARR

- TIMx auto-reload register

Address offset: 0x2C

Reset value: 0x0000

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

Bits 15:0 **ARR[15:0]**: Prescaler value

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

2. TIM2 to TIM5 Registers -- TIMx_CR1

- TIMx control register 1

Address offset: 0x00

Reset value: 0x0000

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

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **CKD**: Clock division

This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and sampling clock used by the digital filters (ETR, TIx),

00: $t_{DTS} = t_{CK\_INT}$

01: $t_{DTS} = 2 \times t_{CK\_INT}$

10: $t_{DTS} = 4 \times t_{CK\_INT}$

11: Reserved

2. TIM2 to TIM5 Registers -- TIMx_CR1(continued)

南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:5 **CMS**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)

Bit 4 **DIR**: Direction

0: Counter used as upcounter
1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.
0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.
1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled
1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

CEN is cleared automatically in one-pulse mode, when an update event occurs.

2. TIM2 to TIM5 Registers -- TIMx_DIER

- TIMx DMA/Interrupt enable register

Address offset: 0x0C

Reset value: 0x0000

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

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled.
1: Trigger DMA request enabled.

Bit 13 Reserved, always read as 0

Bit 12 **CC4DE**: Capture/Compare 4 DMA request enable
0: CC4 DMA request disabled.
1: CC4 DMA request enabled.

2. TIM2 to TIM5 Registers -- TIMx_DIER(continued)

Bit 11 **CC3DE**: Capture/Compare 3 DMA request enable
0: CC3 DMA request disabled.
1: CC3 DMA request enabled.

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable
0: CC2 DMA request disabled.
1: CC2 DMA request enabled.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled.
1: CC1 DMA request enabled.

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled.
1: Update DMA request enabled.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled.
1: Trigger interrupt enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IE**: Capture/Compare 4 interrupt enable
0: CC4 interrupt disabled.
1: CC4 interrupt enabled.

Bit 3 **CC3IE**: Capture/Compare 3 interrupt enable
0: CC3 interrupt disabled.
1: CC3 interrupt enabled.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled.
1: CC2 interrupt enabled.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled.
1: CC1 interrupt enabled.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

2. TIM2 to TIM5 Registers
-- Timer Interrupt Configuration Steps

- Enable timer clock
- Initialize timer, configure arr and PSC
- Enable timer interrupt, configure NVIC
- Enable timer
- Implement interrupt service function

03

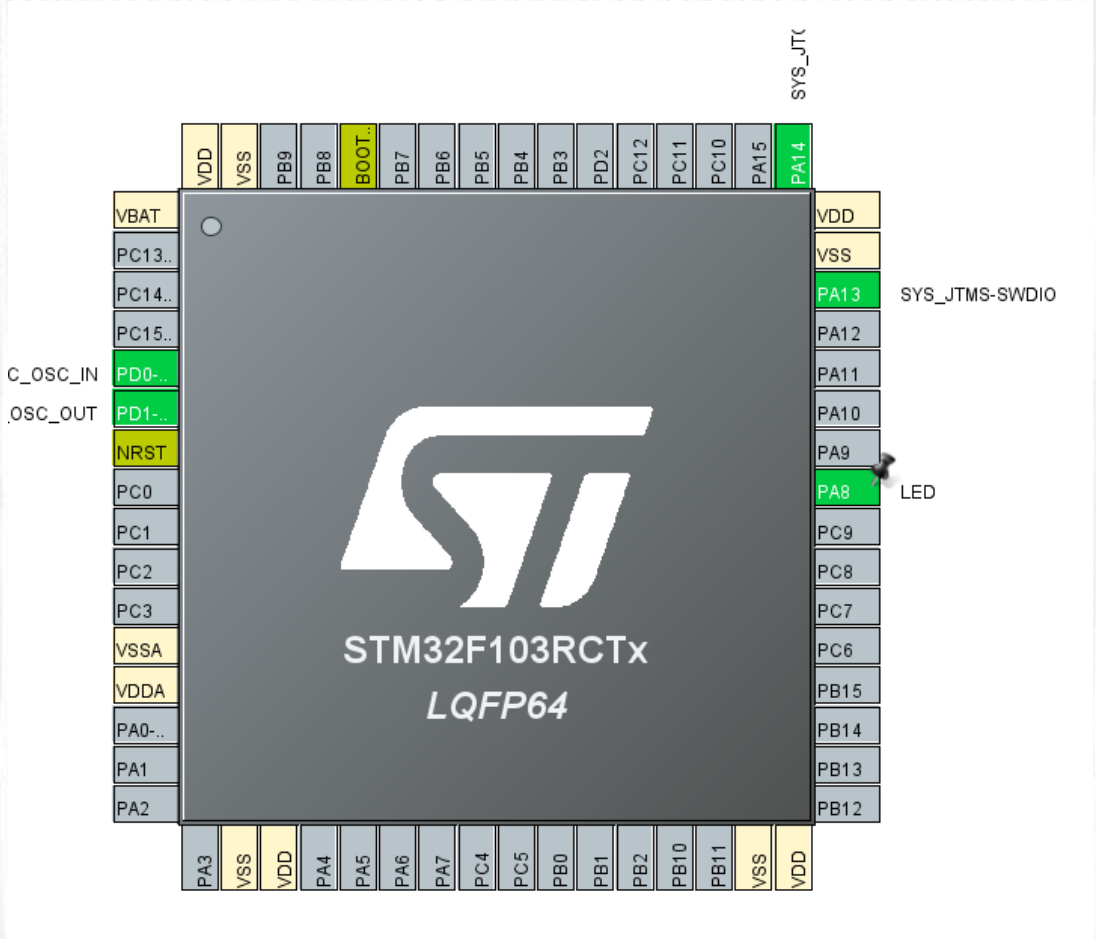
How to Program

3. How to Program

- Our Goal
  - Use timer interrupt to blink the LED every 1 second
  - Do not use any delay function

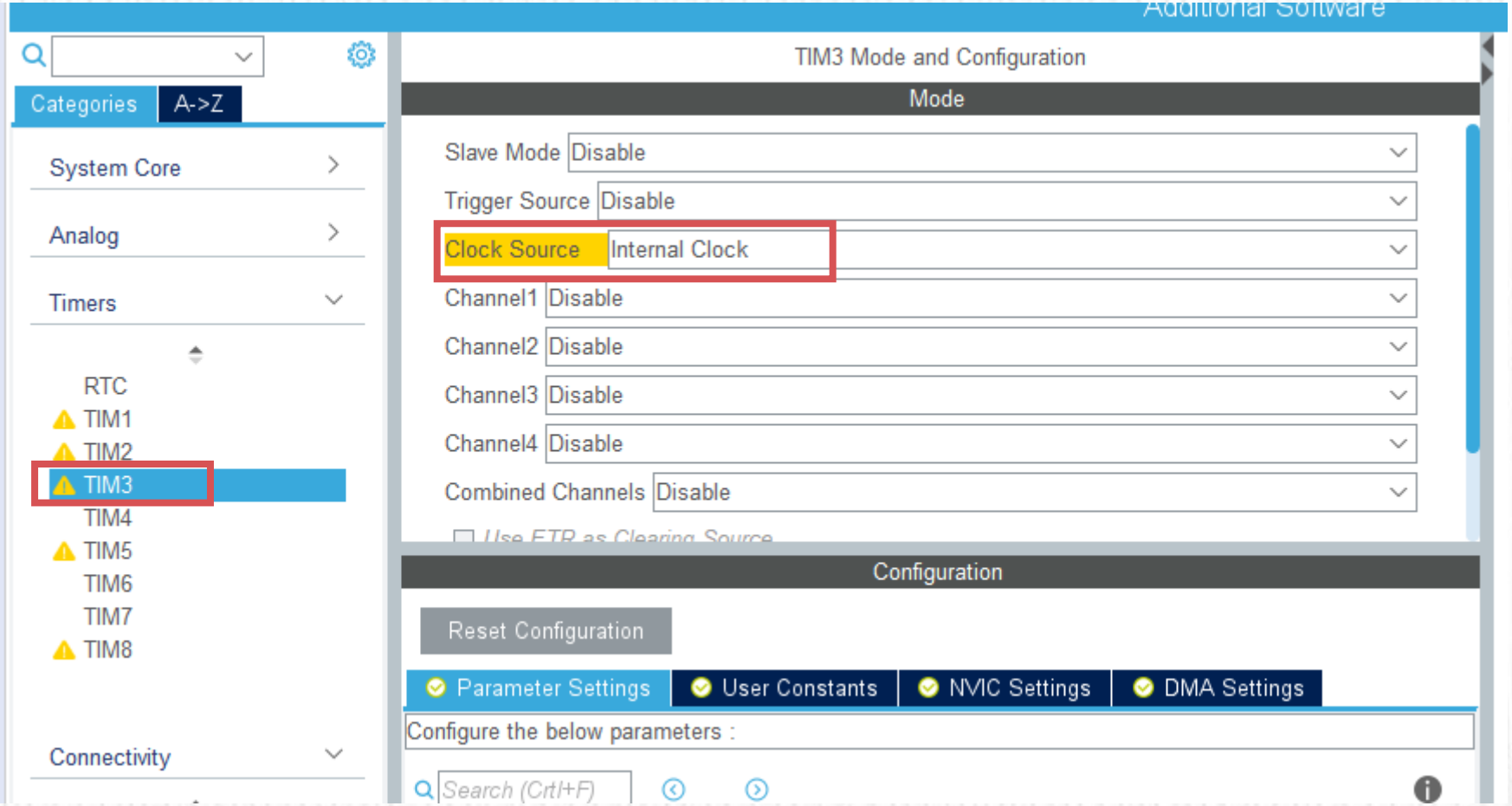
3. How to Program

- Configure GPIO: either led is OK



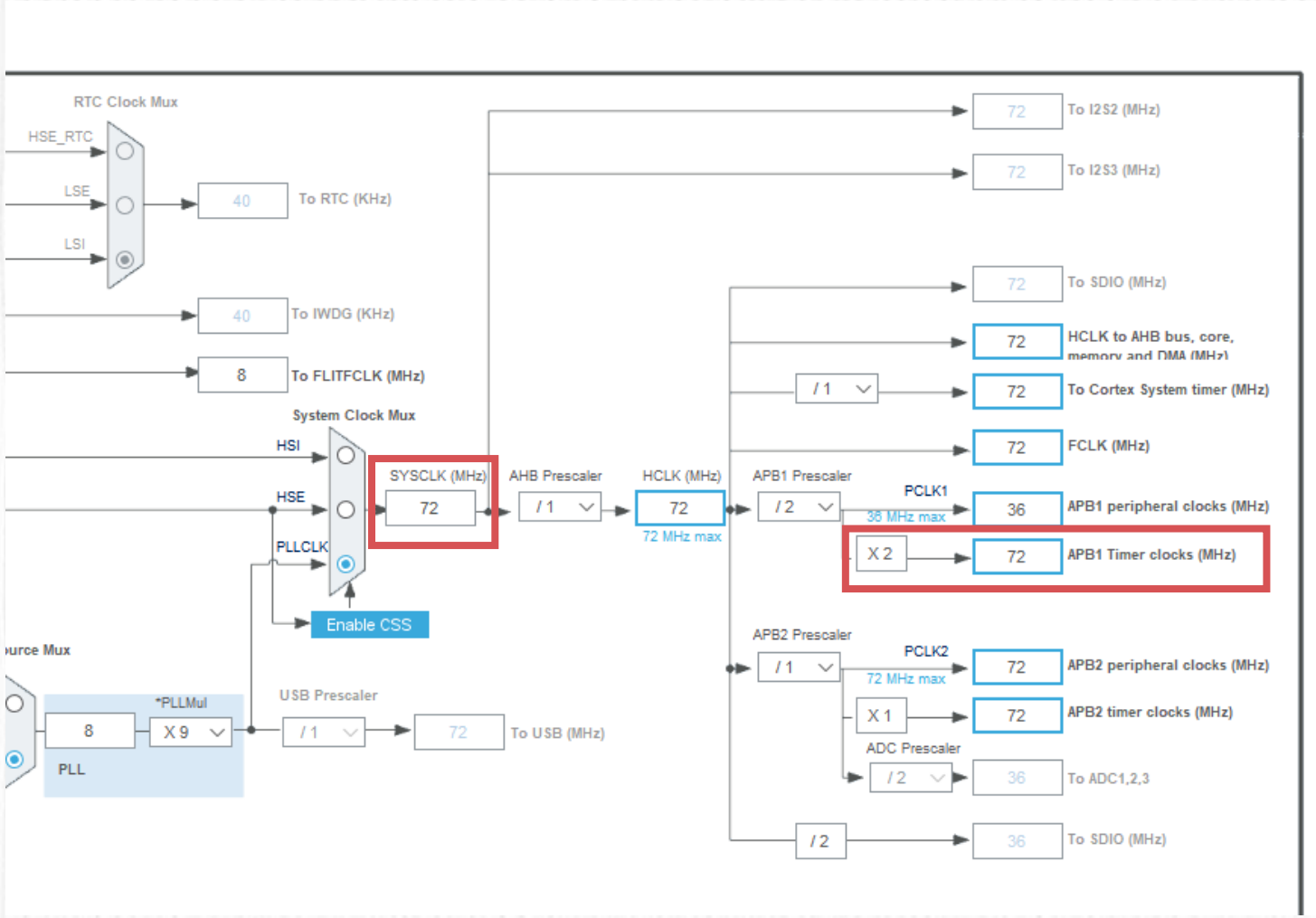
3. How to Program

- Configure TIM3 in STM32CubeIDE
  - Select TIM3 and set the Clock Source as **Internal Clock**



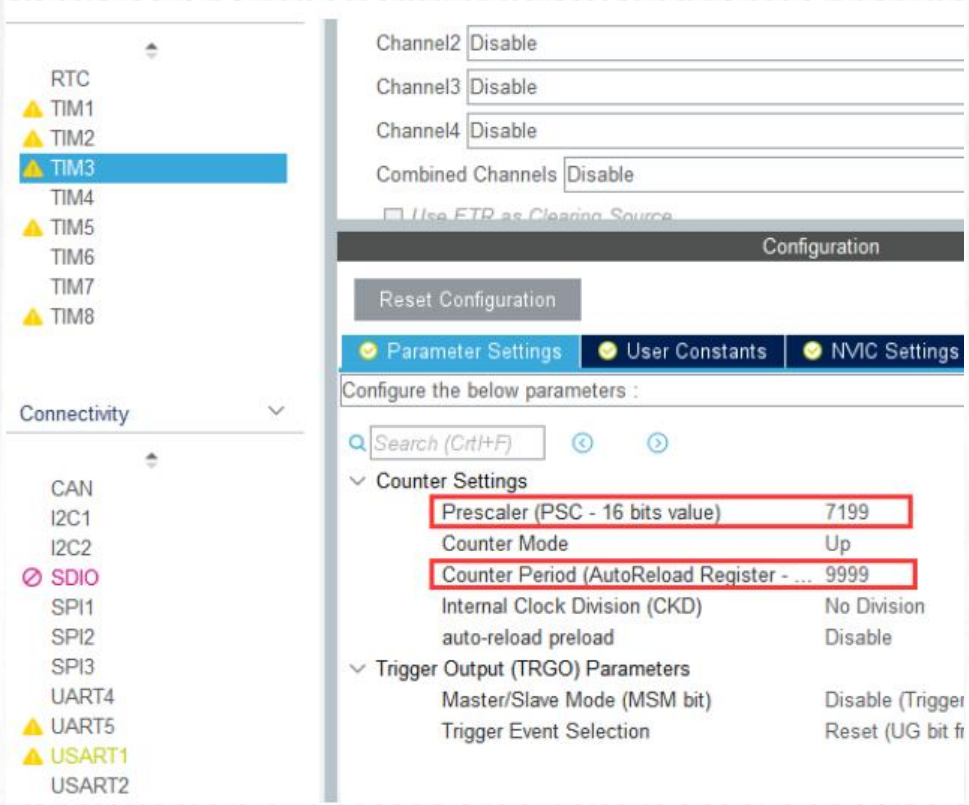
3. How to Program

- What is the frequency of the Internal Clock?
  - SYCLK is 72MHz
  - f_{AHB} = 72MHz
  - f_{APB1} = 36MHz
  - APB1 Prescaler = f_{AHB} / f_{APB1} = 2
  - TIM3 is connected on the APB1 bus
  - f_{ck_psc} = f_{APB1} * 2 = 72MHz



3. How to Program

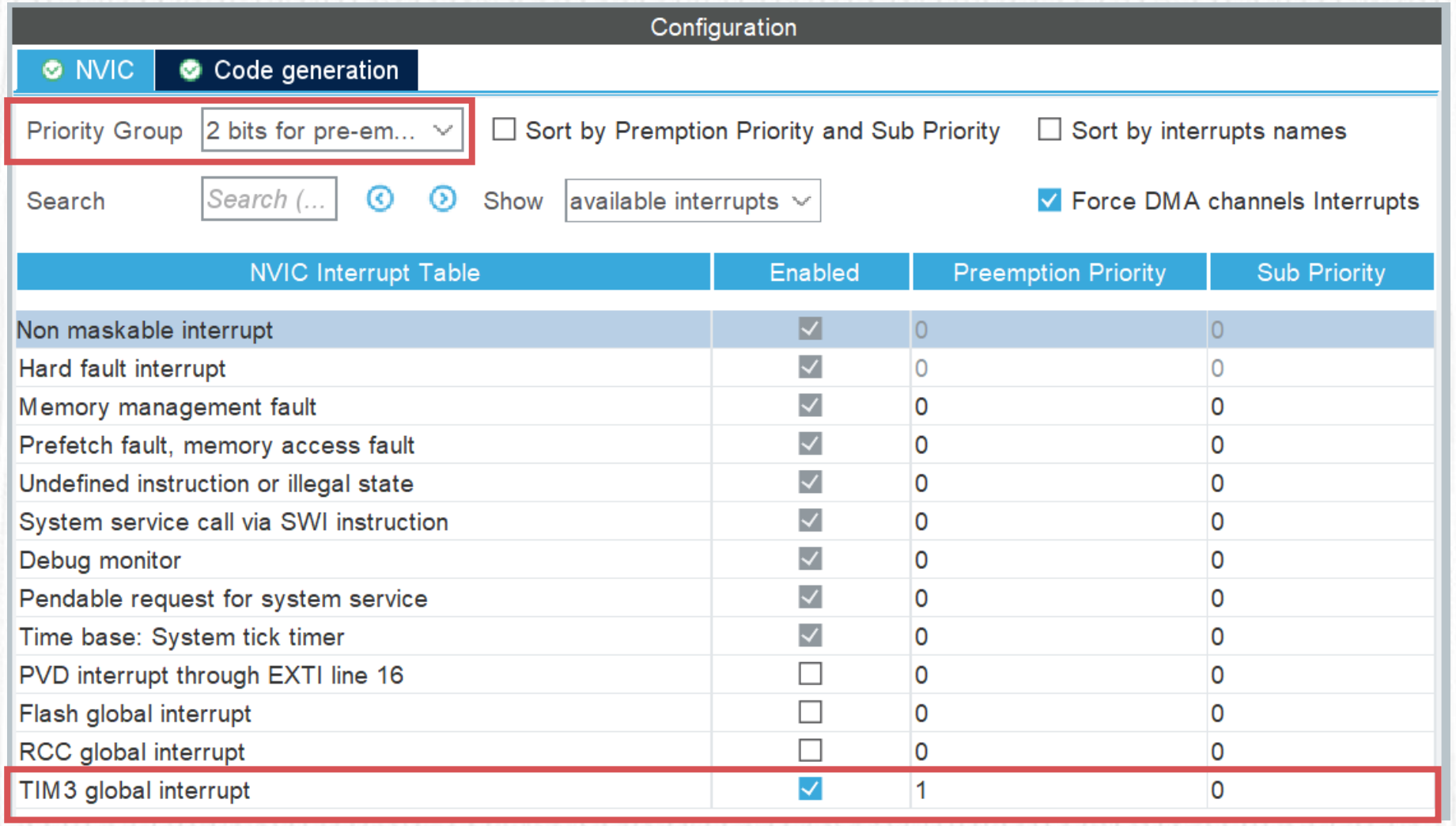
- Counter setting
- $f_{ck\_cnt} = f_{ck\_psc} / (psc+1)$
- $T = (arr+1)(psc+1)/ f_{ck\_psc}$
- where:
  - T is the update interrupt period(us)
  - f_{ck_psc} is the frequency of the internal clock(MHz)
  - arr is the auto-reload value
  - psc is the prescaler



- As we know, f_{ck_psc} = 72Mhz, if we want to set T as 1 second, we can set arr = 9999 and psc = 7199. Other settings will be all right as long as both ranges from 1 to 65535.

3. How to Program

- Configure NVIC in STM32CubeIDE



3. How to Program

- Some functions we used
  - Call the following function to start/stop timer in interrupt mode(for example in the main routine)

```
HAL_StatusTypeDef HAL_TIM_Base_Start_IT(TIM_HandleTypeDef *htim)
HAL_StatusTypeDef HAL_TIM_Base_Stop_IT(TIM_HandleTypeDef *htim)
```

```
/* USER CODE BEGIN PV */
extern TIM_HandleTypeDef htim3;
/* USER CODE END PV */
```

```
int main(void)
{
  //...
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_TIM3_Init();
  MX_USART1_UART_Init();
  /* USER CODE BEGIN 2 */
  HAL_TIM_Base_Start_IT(&htim3);
  /* USER CODE END 2 */
  //...
}
```

3. How to Program

- Some functions we used
  - All the interrupts of TIM3 are handled by TIM3_IRQHandler in stm32f1xx_it.c, it calls the public TIM interrupt handler HAL_TIM_IRQHandler, which will call the corresponding callback function according to the interrupt type and clear the corresponding interrupt pending bits.

```c
/**
  * @brief This function handles TIM3 global interrupt.
  */
void TIM3_IRQHandler(void)
{
  /* USER CODE BEGIN TIM3_IRQn 0 */

  /* USER CODE END TIM3_IRQn 0 */
  HAL_TIM_IRQHandler(&htim3);
  /* USER CODE BEGIN TIM3_IRQn 1 */

  /* USER CODE END TIM3_IRQn 1 */
}
```

```c
void HAL_TIM_IRQHandler(TIM_HandleTypeDef *htim)
{
  //...
  /* TIM Update event */
  if (__HAL_TIM_GET_FLAG(htim, TIM_FLAG_UPDATE) != RESET)
  {
    if (__HAL_TIM_GET_IT_SOURCE(htim, TIM_IT_UPDATE) != RESET)
    {
      __HAL_TIM_CLEAR_IT(htim, TIM_IT_UPDATE);
#if (USE_HAL_TIM_REGISTER_CALLBACKS == 1)
      htim->PeriodElapsedCallback(htim);
#else
      HAL_TIM_PeriodElapsedCallback(htim);
#endif /* USE_HAL_TIM_REGISTER_CALLBACKS */
    }
  }
  //..
}
```

```c
/**
  * @brief  Period elapsed callback in non-blocking mode
  * @param  htim TIM handle
  * @retval None
  */
__weak void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  /* Prevent unused argument(s) compilation warning */
  UNUSED(htim);

  /* NOTE : This function should not be modified, when the callback
            the HAL_TIM_PeriodElapsedCallback could be implemented i
file
   */
}
```

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

04

Practice

4. Practice

- Re-implement the **HAL_TIM_PeriodElapsedCallback( )**, and run the demo on MiniSTM32 board.
- Design a simple digital counter that starts counting from 0 and displays a digit number increasing of 1 every 0.5 seconds. When KEY_WK is pressed, the system starts counting again from 0. (Any delay function is not permitted)