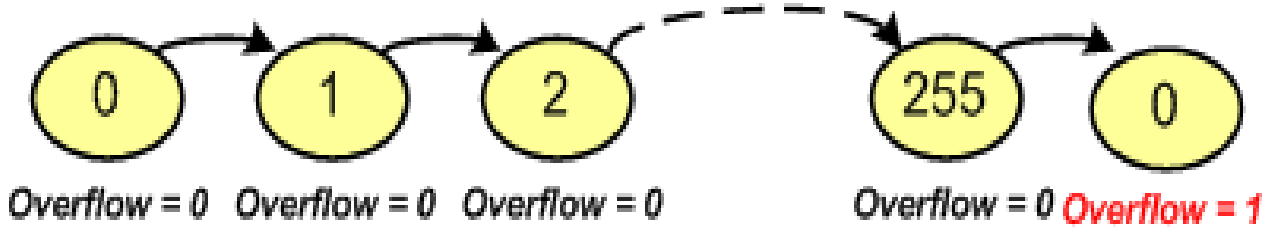
Timer

Topics

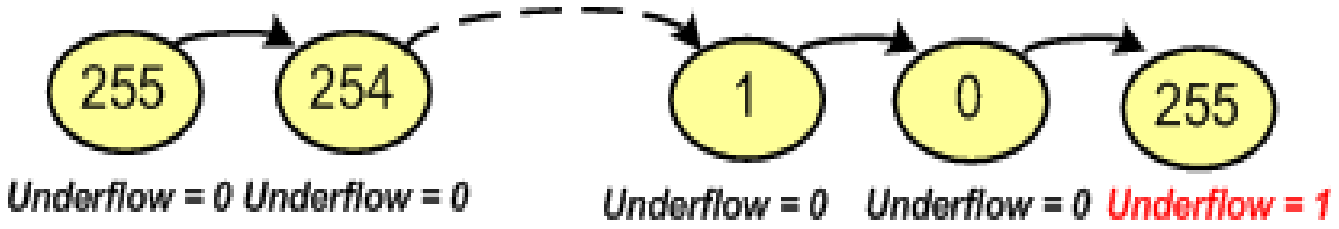
- SysTick Timer
- STM32 Timers

8-bit counter Stages

- Up-counter

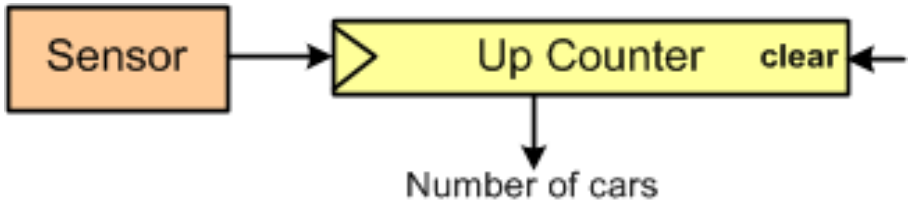


- Down counter

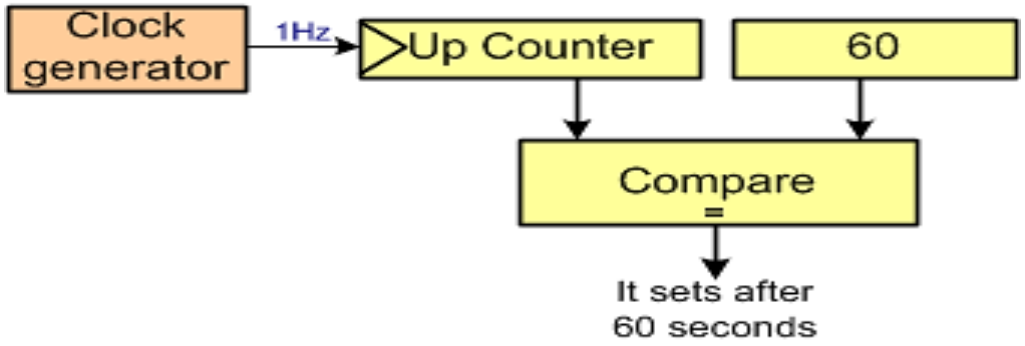


Some Counter Usages

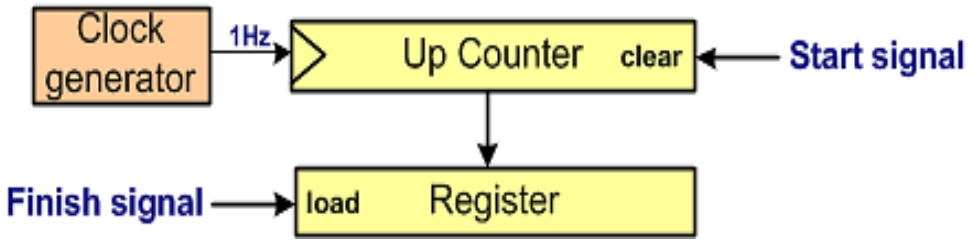
- Event Counter



- Timer

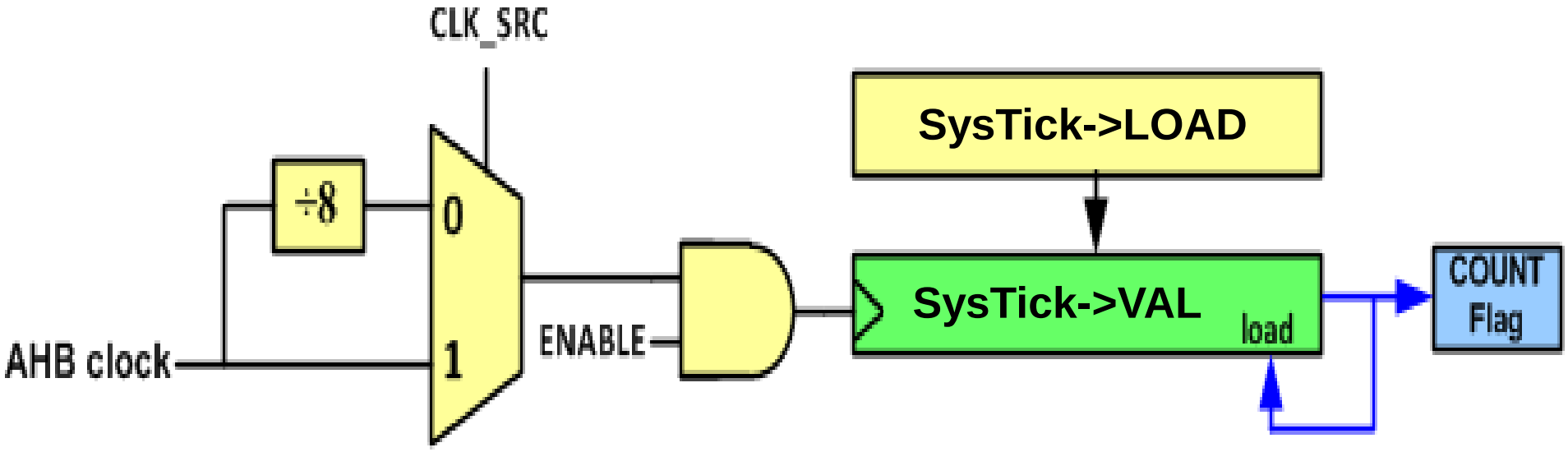


- Measuring the time between 2 events

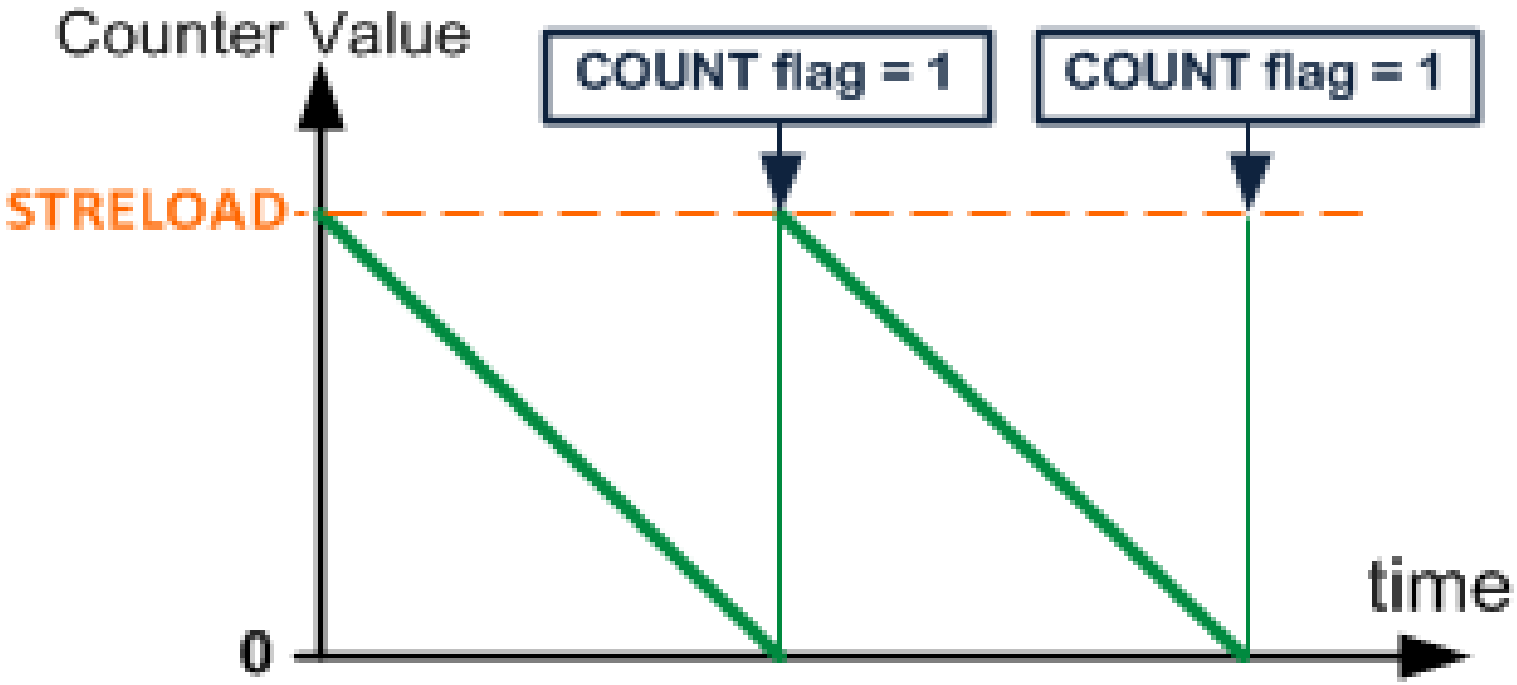


System Tick Timer

- Available in all Cortex-M MCUs
- It is a 24-bit down counter. It counts down from an initial value to 0.
- Used to initiate an action on a periodic basis
  - OS ticks



System Tick Counting

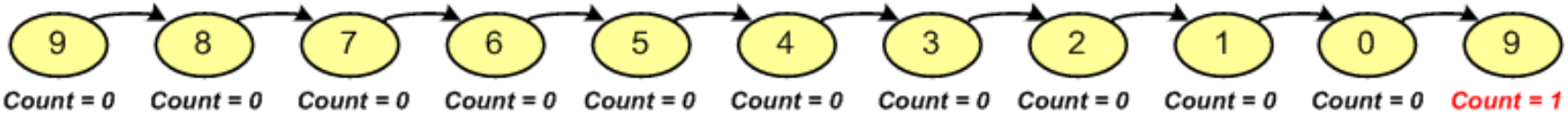


STCTRL (System Tick Control) Register

Name	bit	Description
ENABLE	0	0: The counter is disabled, 1: enables SysTick to begin counting down
TICKINT	1	Interrupt Enable 0: Interrupt generation is disabled. 1: when SysTick counts to 0 an interrupt is generated
CLKSOURCE	2	Clock Source 0: AHB clock divided by 8 1: AHB clock
COUNTFLAG	16	Count flag 0: The SysTick has not counted down to zero since the last time this bit was read 1: The SysTick has counted down to zero Note: This flag is cleared by reading the STRCTRL or writing to STCURRENT.

Example: Assuming system clock = 8 MHz, calculate the delay which is made by the following function.

```
void delay() {
  SysTick->LOAD = 9;
  SysTick->CTRL = 5; /*Enable the timer and choose system clock as the
clock source */

  while((SysTick->CTRL &0x10000) == 0) /*wait until the Count flag is set */
  { }
  SysTick->CTRL = 0; /*Stop the timer (Enable = 0) */
```



Since the AHB clock is chosen as the clock source, each clock lasts $\frac{1}{sysclk} = \frac{1}{8MHz} = 0.125\mu s$.

So, the program makes a delay of $10 \times 0.125\mu s = 1.25\mu s = 1250ns$.

Example

- In an ARM microcontroller a clock with frequency of clk is fed to the sysTick timer. Calculate the delay which is made by the timer if the STRELOAD register is loaded with N.

Solution:

The timer is initialized with N. So, it goes through N+1 stages.
Since the system clock is chosen as the clock source, each clock lasts 1 / clk
So, the program makes a delay of (N + 1) × (1 / clk) = (N + 1) / clk.

Example: Using the System Tick timer, write a function that makes a delay of 1 ms. Assume APB clock = 72 MHz.

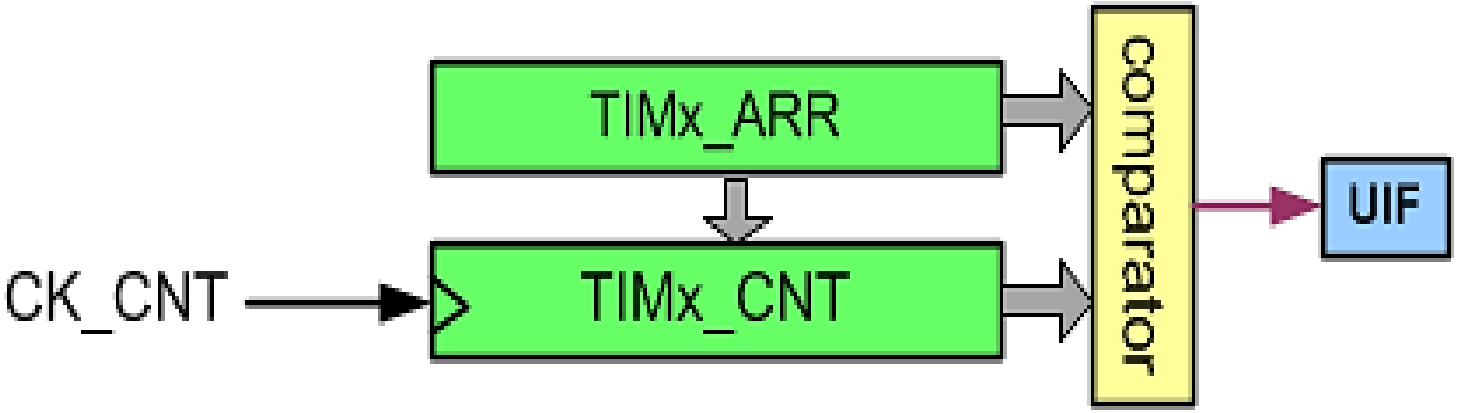
Solution:

delay = (N + 1) / clk ➔ (N + 1) = delay × clk = 0.001 sec × 72 MHz = 72,000 ➔ N = 72,000 – 1 = 71999

```
void delay1ms(void)
{
   SysTick->LOAD = 71999;
   SysTick->CTRL = 0x5;   /* Enable the timer and choose sysclk as the clock source */

   while((SysTick->CTRL & 0x10000) == 0) /* wait until the COUNT flag is set */
   { }
   SysTick->CTRL = 0; /* Stop the timer (Enable = 0) */
}
```

STM32 Timers

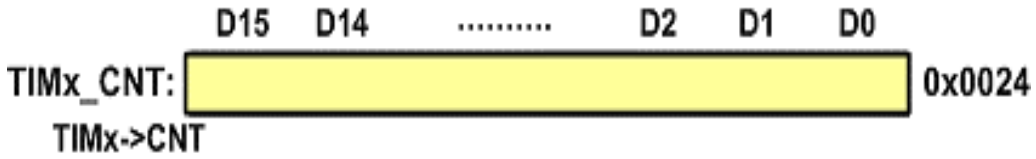
STM32 Timers



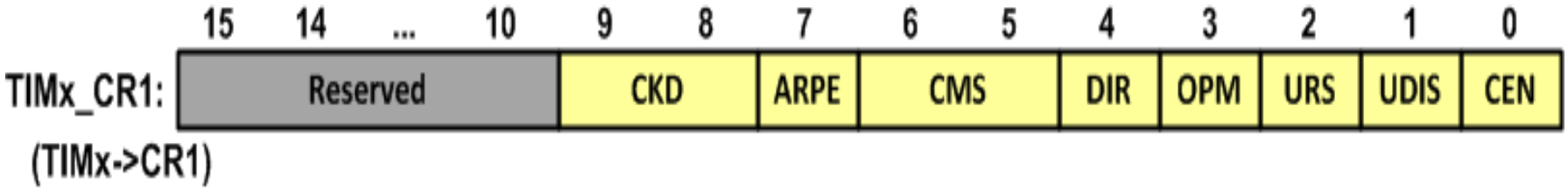
- ARR (Auto-Reload Register)

Note:
In STM32 microcontrollers, all the timer registers begin with TIMx. So, for simplicity, just consider the letters which come after TIM. For example, consider TIMx_CNT as CNT (Counter).

- CNT (Counter)



TIMx_CR1 (Control Register)



- CMS (Center-Aligned Mode Selection)
  - 00: Edge-aligned mode
  - 01, 10, 11: Center-aligned mode
- DIR (Direction)
  - 0: count up
  - 1: count down
- OPM (One Pulse Mode)
  - 0: Counter is not stopped at update event
  - 1: Counter stops counting at the next update event.

CMS	DIR	Counting mode
00	0	Counting up
00	1	Counting down
01	X	Count up and down
10	X	Count up and down
11	X	Count up and down

Example: Find the TIMx_CR1 value to: (a) count up continuously (b) count down continuously (c) stop counting.

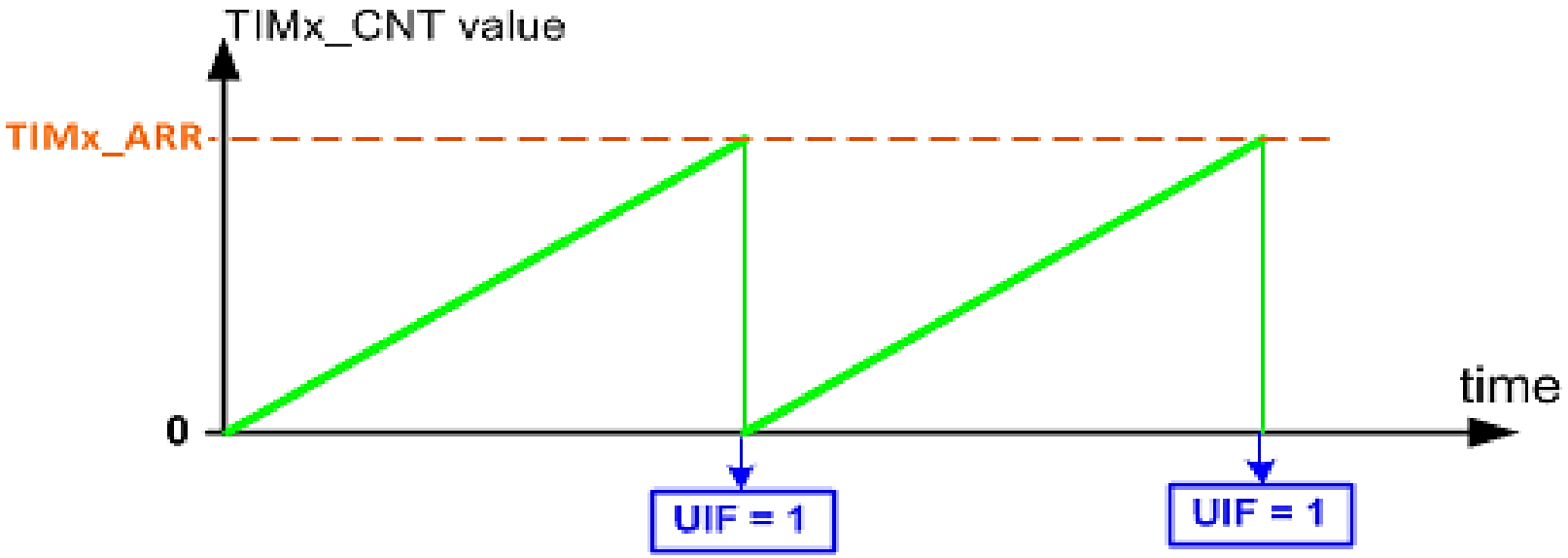
TIMx_	CKD	ARPE	CMS	DIR	OPM	URS	UDIS	CEN
CR1:	0	0	00	0	0	0	0	1

TIMx_	CKD	ARPE	CMS	DIR	OPM	URS	UDIS	CEN
CR1:	0	0	00	1	0	0	0	1

TIMx_	CKD	ARPE	CMS	DIR	OPM	URS	UDIS	CEN
CR1:	0	0	00	0	0	0	0	0

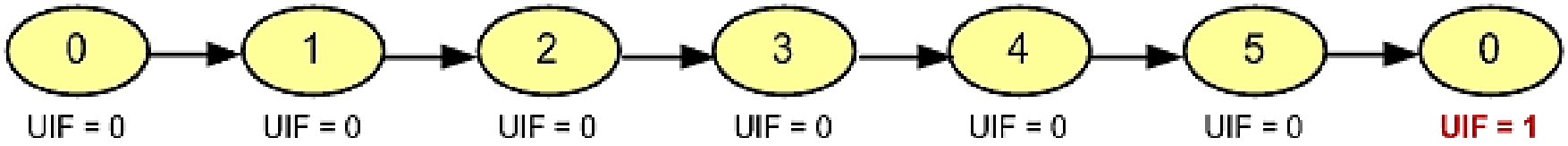
TIMx_SR (Status Register)

Counting Up



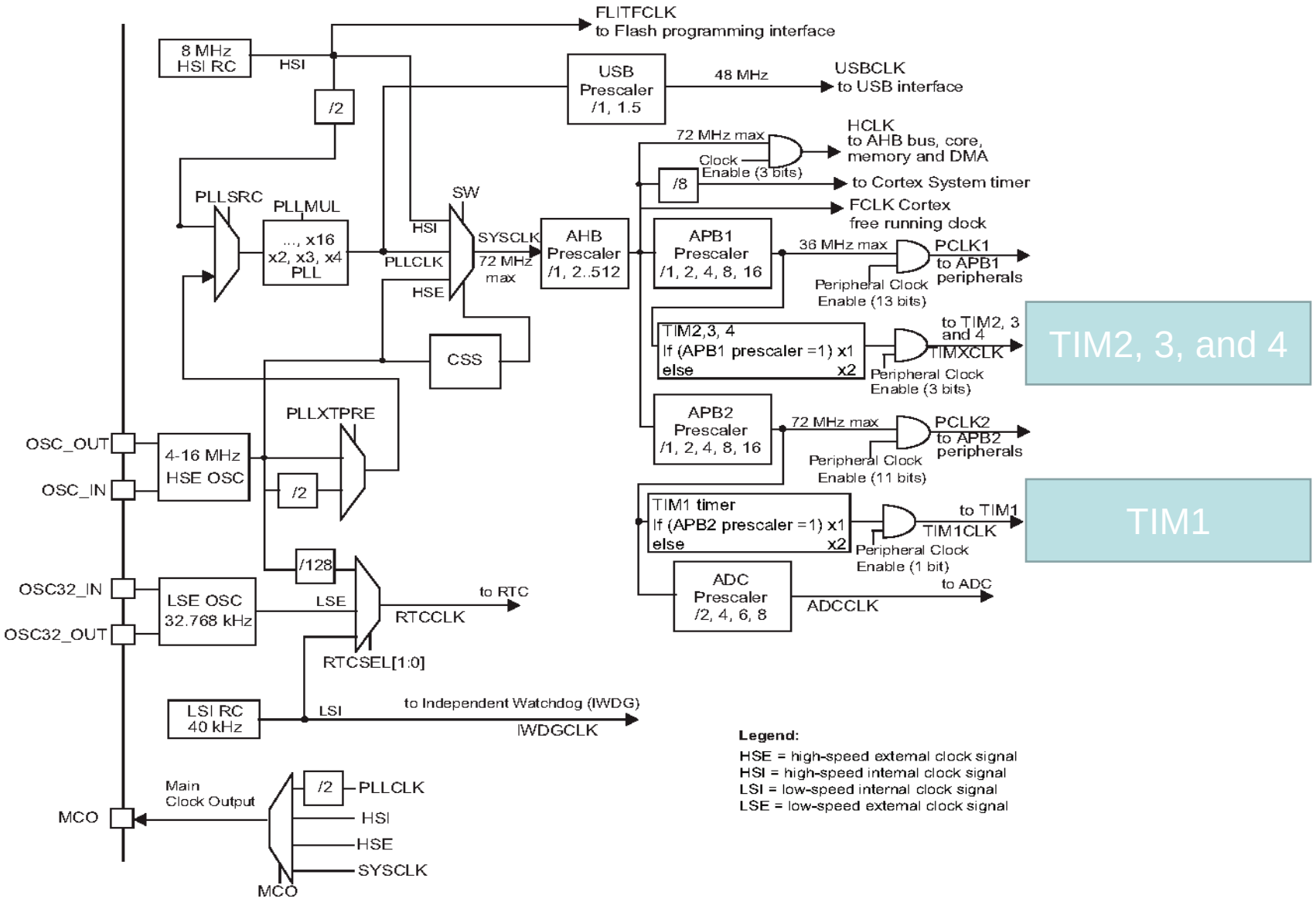
Example

- Assume TIM2_ARR = 5 and TIM2_CNT is counting up. (a) Explain when the UIF flag is raised. (b) How many clocks does it take until the UIF flag rises?



(b) When the counter starts counting, it goes through 6 states (ARR+1 states) until the flag rises.

STM32F10X Clock



TIM2, 3, and 4

TIM1

By default, the CPU clock frequency is 72MHz; the APB1 clock is set to 36MHz, and APB2 is 72MHz. Calculate the frequency of the clock that is fed to the timers.

- Since the CPU clock is 72MHz and APB1 clock is 36MHz, the prescaler for APB1 is set to 2 (other than 1). So the APB2 clock is multiplied by 2 and fed to the timers which are connected to APB1 bus. 36MHz × 2 = 72MHz.
- APB2 clock has the same frequency as the CPU. So, its prescaler is set to 1 and the timer clocks are the same as the APB2 clock. According to Figure 8-8, TIM1 and TIM8 are connected to APB2. So, a clock with frequency of 72MHz are fed to the timers.
- So, the clocks for all timers are 72MHz by default, unless we change the APB prescalers.

Enabling Clocks

RCC_APB1ENR:
(RCC->APB1ENR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved		DACEN	PWR EN	BKPEN	Res.	CAN EN	Res.	USBEN	I2C2EN	I2C1EN	USART 5EN	USART 4EN	USART 3EN	USART 2EN	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3EN	SPI2EN	reserved		WWDG EN	reserved		TIM14 EN	TIM13 EN	TIM12 EN	TIM7 EN	TIM6 EN	TIM5 EN	TIM4 EN	TIM3 EN	TIM2 EN

RCC_APB2ENR:
(RCC->APB2ENR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved										TIM11 EN	TIM10 EN	TIM9 EN	Res.		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC3 EN	USART 1EN	TIM8 EN	SPI1 EN	TIM1 EN	ADC2 EN	ADC1 EN	IOPG EN	IOPF EN	IOPE EN	IOPD EN	IOPC EN	IOPB EN	IOPA EN	Res.	AFIO EN

Label	Description	Label	Description
IOPx	I/O port x clock enable	ADCnEN	ADCn clock enable
USARTnEN	USARTn clock enable	DACnEN	DACn clock enable
USBEN	USB clock enable	TIMnEN	TIMn timer clock enable
CANEN	CAN clock enable	SPInEN	SPI n clock enable
PWREN	Power interface clock enable	BKPEN	Backup interface clock enable
WWDG	Window watchdog clock enable	SDIOEN	SDIO clock enable
DMAnEN	DMAn clock enable	FSMCEN	FSMC clock enable
CRCEN	CRC clock enable	I2CnEN	I2Cn clock enable
Note: (0: clock disabled, 1: clock enabled)			

Example: Calculate the delay which is made by the following function.

```
void delay()
{
        RCC->APB1ENR |= (1<<0);          /* enable TIM2 clock */
        TIM2->ARR = 71;
        TIM2->SR = 0;     /* clear the UIF flag */
        TIM2->CR1 = 1; /* up counting */
        while((TIM2->SR & 1) == 0); /* wait until the UIF flag is set */
        TIM2->CR1 = 0; /*stop counting */
}
```

- Solution:
  - It goes through 72 stages.
  - Since the timer clock is 72MHz by default, each clock lasts $\frac{1}{72MHz}$.
  - So, the program makes a delay of $72 \times \frac{1}{72MHz} = 1\mu s$.

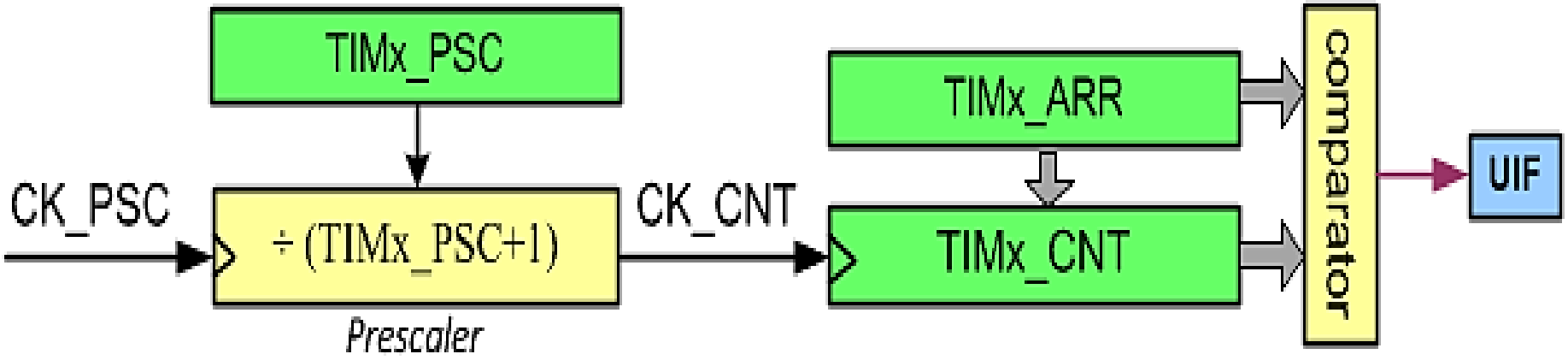
Example: Using TIM2 make a delay of 50µs. The clock frequencies are set by default.

Solution:

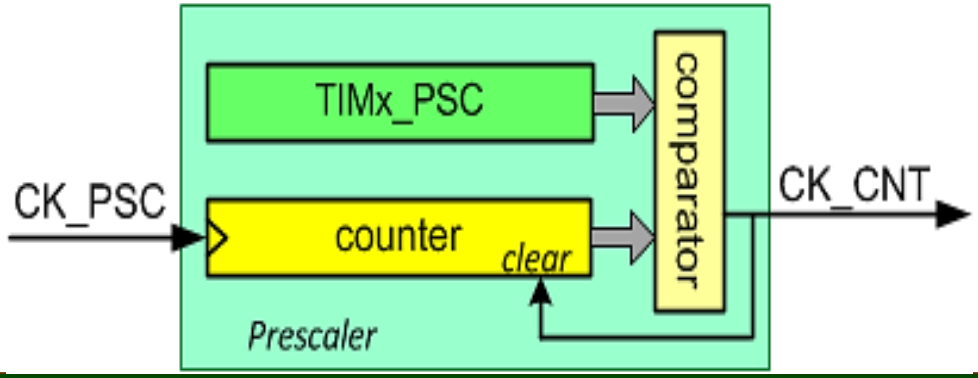
Delay = ARR+1 /72MHz ➔ ARR+1 = delay × 72MHz = 50 µs × 72MHz = 3600
➔ ARR = 3600 – 1 = 3599.

```
void delay()
{
        RCC->APB1ENR |= (1<<0);          /* enable TIM2 clock */
        TIM2->ARR = 3599;
        TIM2->CR1 = 1; /* up counting */
        while((TIM2->SR & 1) == 0); /* wait until the UIF flag is set */
        TIM2->CR1 = 0; /*stop counting */
        TIM2->SR = 0;     /* clear the UIF flag */
}
```

Prescaler



- Prescaler is a 16-bit up-counter and a comparator



Example: Calculate the delay which is made by the following function.

```
void delay() {
        RCC->APB1ENR |= (1<<0);          /* enable TIM2 clock */
        TIM2->PSC = 7200-1;              /* PSC = 7199 */
        TIM2->ARR = 500-1;
        TIM2->SR = 0;     /* clear the UIF flag */
        TIM2->CR1 = 1; /* up counting */
        while((TIM2->SR & 1) == 0); /* wait until the UIF flag is set */
        TIM2->CR1 = 0; /*stop counting */
}
```

Solution:

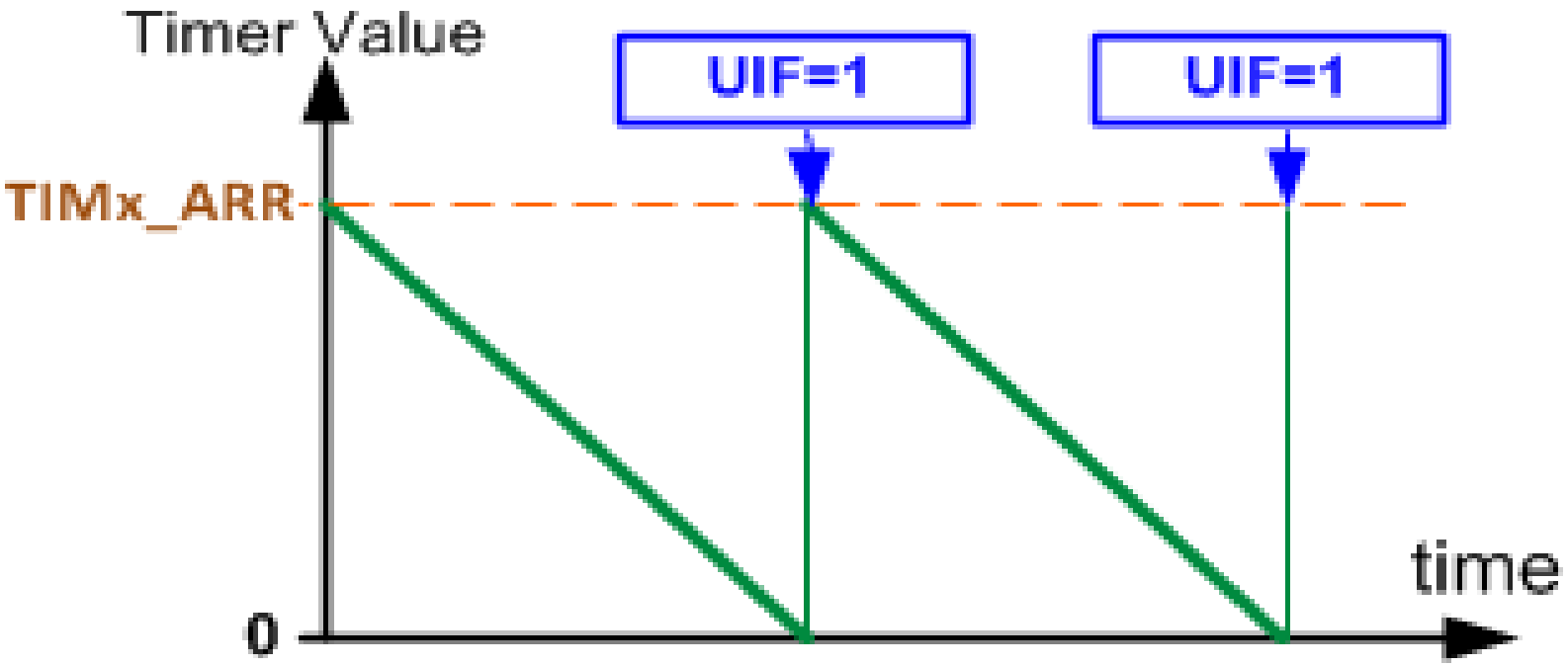
The clock is divided by 7200 ➔ 72MHz/7200 = 10KHz.

Each clock lasts $\frac{1}{10KHz} = 0.1ms$ and the program makes a delay of $500 \times 0.1\text{ms} = 50\text{ms}$.

Example: Using TIM2 write a program that toggles PC13, every second.

```
#include <stm32f10x.h>
void delay(void);
int main( ) {
        RCC->APB2ENR |= 0xFC;           /* enable GPIO clocks */
        RCC->APB1ENR |= (1<<0);         /* enable TIM2 clock */
        GPIOC->CRH = 0x44344444;        /* PC13 as output */

        while(1)  {
                GPIOC->ODR ^= (1<<13); /* toggle PC13 */
                delay();
        }
}
void delay() {
        TIM2->PSC = 7200-1;             /* PSC = 7199 */
        TIM2->ARR = 10000-1;
        TIM2->SR = 0;     /* clear the UIF flag */
        TIM2->CR1 = 1; /* up counting */
        while((TIM2->SR & 1) == 0); /* wait until the UIF flag is set */
        TIM2->CR1 = 0; /*stop counting */
}
```

Down Counting



Example: Calculate the delay which is made by the following function.

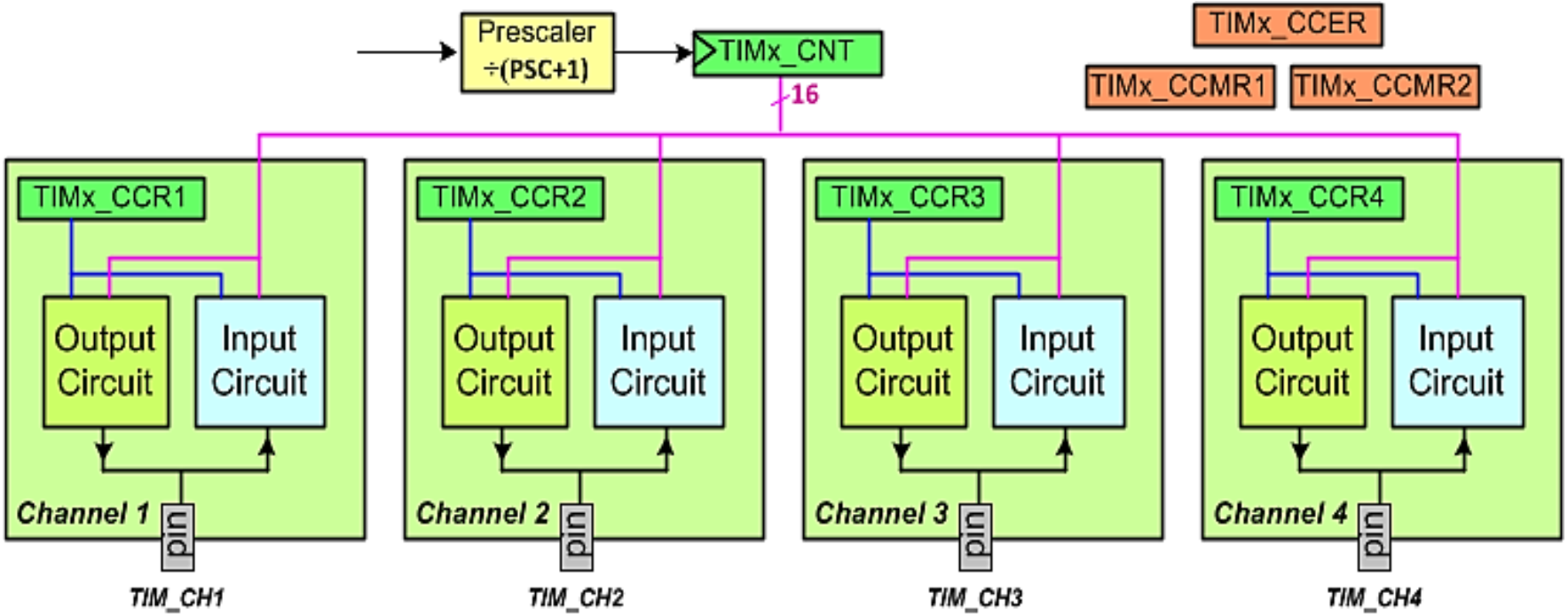
```
void delay()
{
        RCC->APB1ENR |= (1<<0);       /* enable TIM2 clock */
        TIM2->ARR = 999;
        TIM2->CR1 = 1; /* up counting */
        while((TIM2->SR & 1) == 0); /* wait until the UIF flag is set */
        TIM2->CR1 = 0; /*stop counting */
        TIM2->SR = 0;  /* clear the UIF flag */
}
```

Solution:

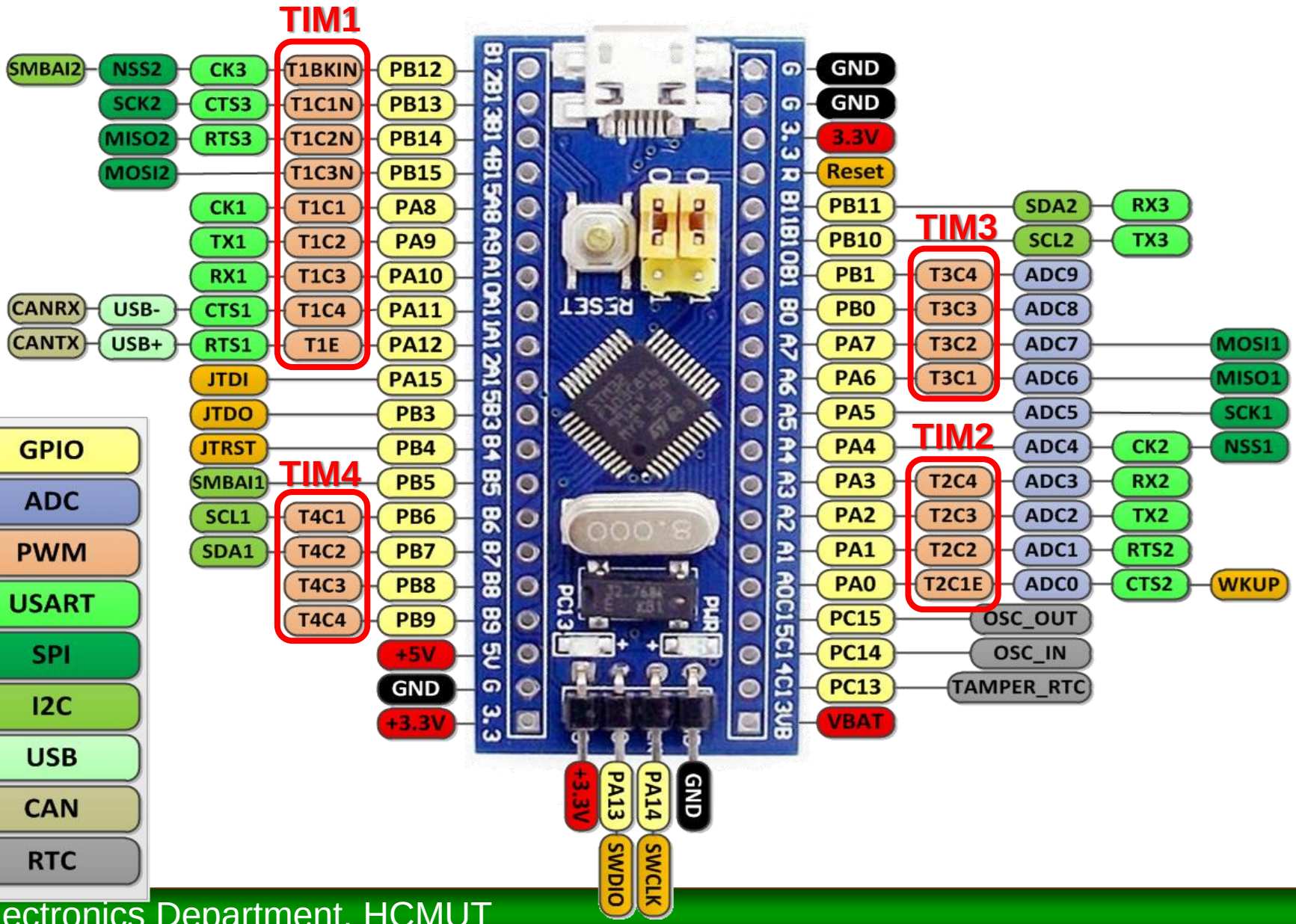
It takes 1000 clocks to rise the flag.
Each clock lasts 1/72MHz
Delay = 1000×1/72MHz = 1/72K = 13.8µs.

Output Compare

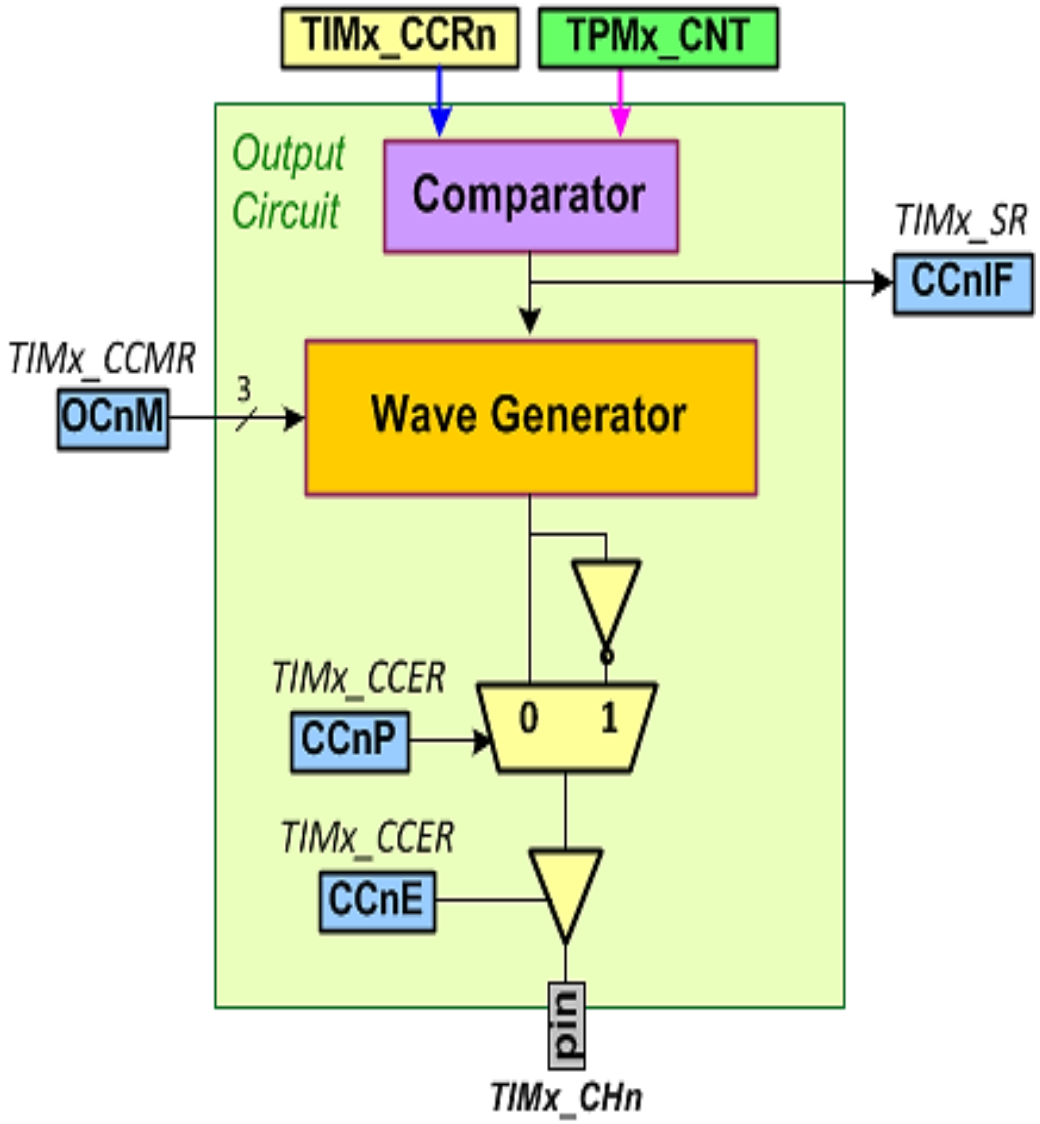
The Channels of TIMx



Timer Pins in the Blue Pill



Output Circuit

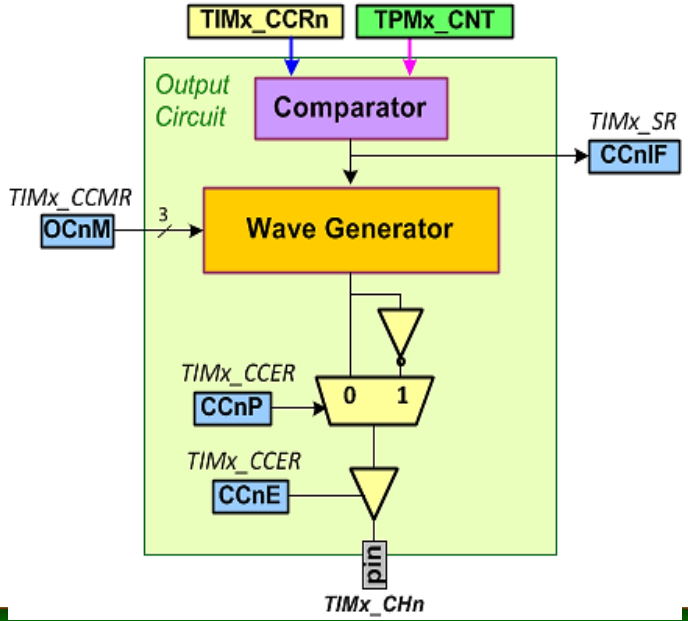


CCER (Compare/Capture Enable Reg.)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
TIMx_CCER:	Reserved		CC4P	CC4E	Reserved		CC3P	CC3E	Reserved		CC2P	CC2E	Reserved		CC1P	CC1E	0x20

(TIMx->CCER)

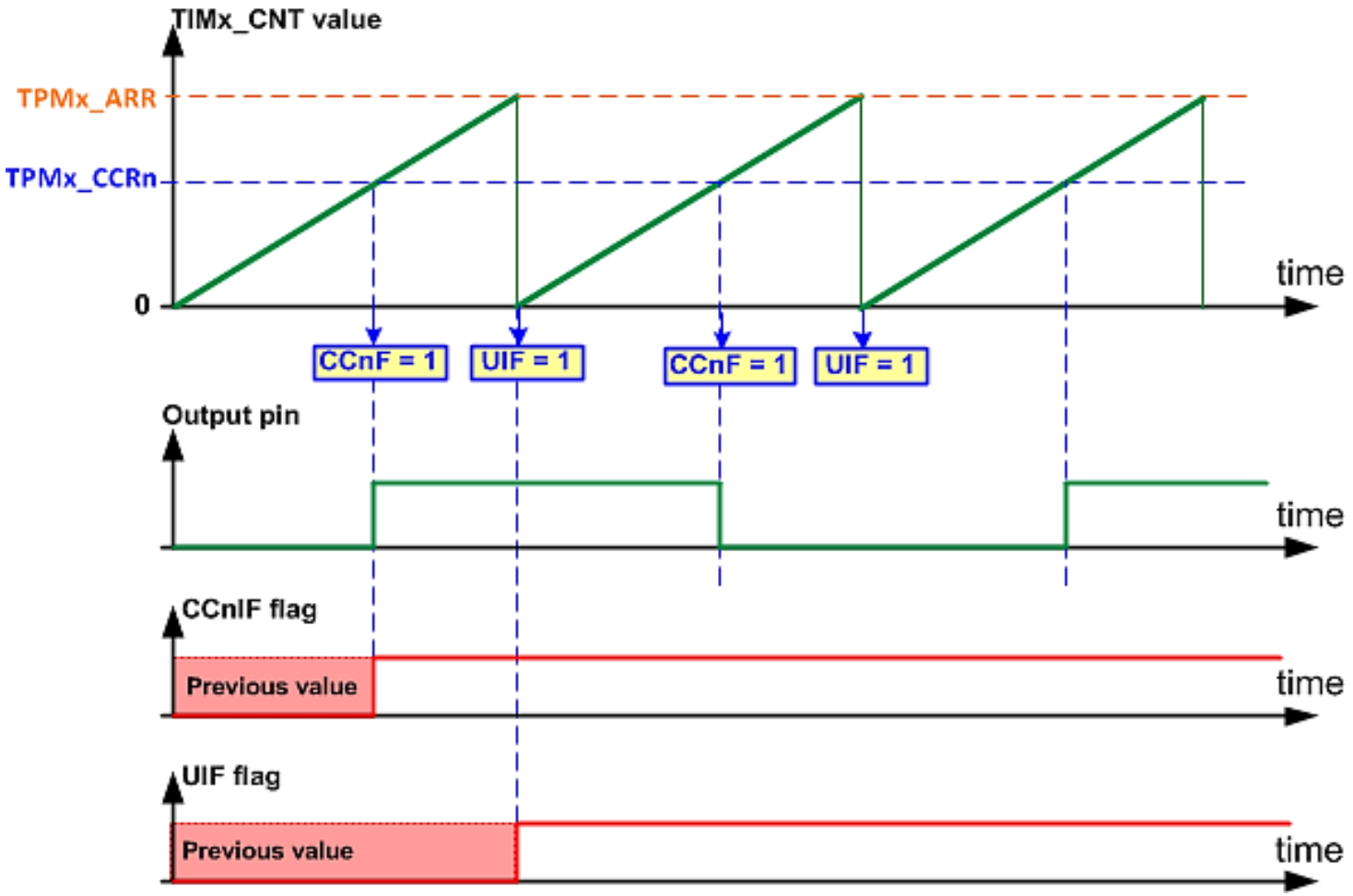
- CCnP (Compare/Capture n output Polarity)
  - 0: Active high (directly)
  - 1: Active low (inverted)
- CCnE (Compare/Capture n output Enable
  - 0: the output is disabled.
  - 1: the output is enabled



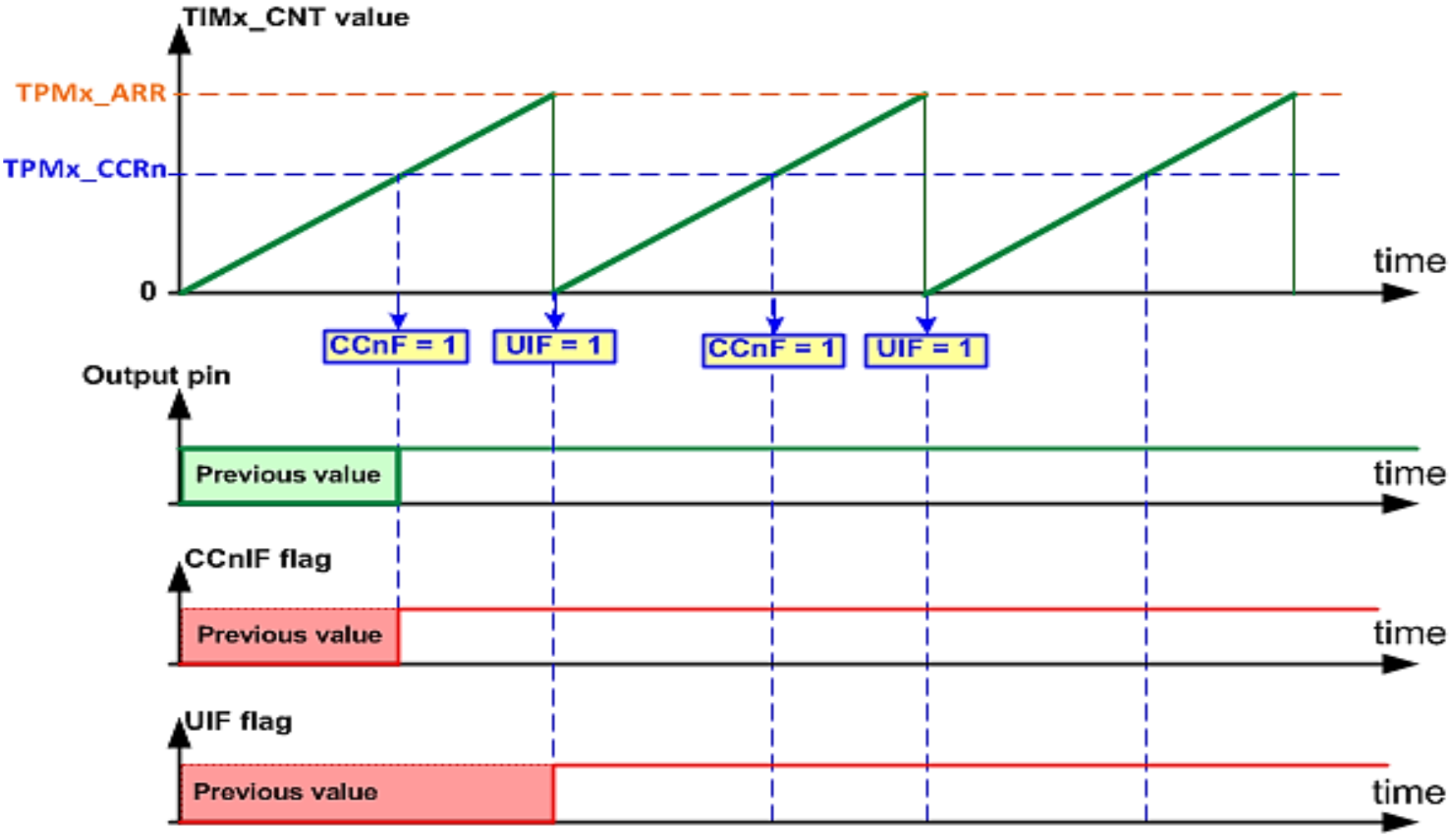
CCMR1 and CCMR2

OCnM	Mode	Description
000	Frozen	Compare match has no effect on the GPIO pin
001	Active on match	When CNT=CCRn, the wave generator makes its output high and the GPIO pin (TIMx_CHn) becomes active.
010	Inactive on match	When CNT=CCRn, the wave generator makes its output low and the GPIO pin (TIMx_CHn) changes to inactive level.
011	Toggle on match	When CNT=CCRn, it toggles the GPIO pin (TIMx_CHn).
100	Force inactive	It forces the GPIO pin to inactive level without considering the values of the TIMx_CNT and TIMx_CCRn registers.
101	Force active	It forces the GPIO pin to active level without considering the values of the TIMx_CNT and TIMx_CCRn registers.
110	PWM 1	It is discussed in the PWM chapter.
111	PWM 2	It is discussed in the PWM chapter.

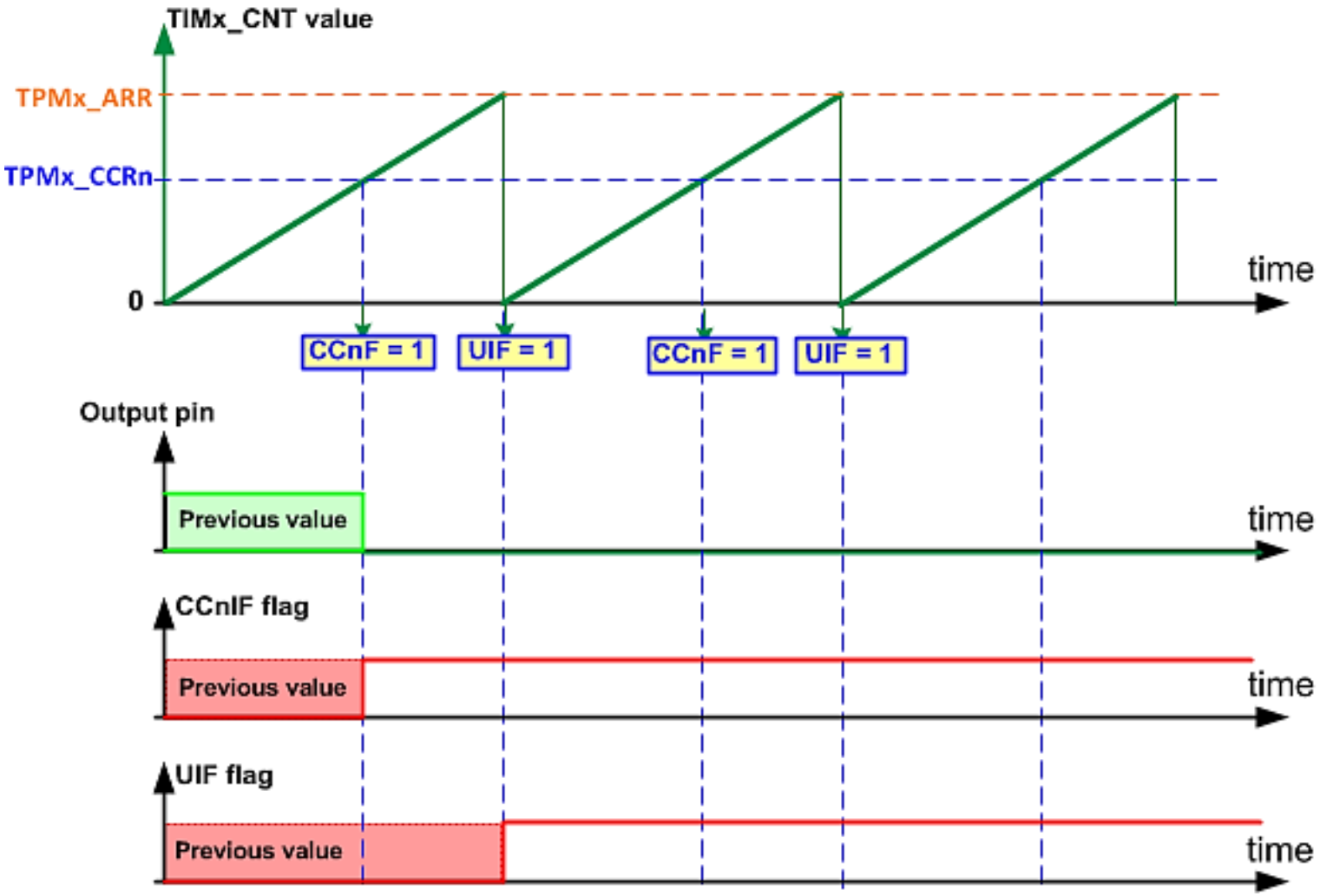
Toggle mode (OCnM = 011)



Set Mode (OCnM = 001)



Clear Mode (OCnM =010)



Example: Find the CCMR value to: (a) Set high channel 2 on compare match, (b) toggle channel 2 on compare match, (c) toggle channel 3 on compare match

OCnM	Mode
000	Frozen
001	Active on match
010	Inactive on match
011	Toggle on match
100	Force inactive
101	Force active
110	PWM 1
111	PWM 2

(a)

TIMx_CCMR1:	OC2CE	OC2M	OC2PE	OC2FE	CC2S	OC1CE	OC1M	OC1PE	OC1FE	CC1S
	0	001	0	0	00	0	000	0	0	00

TIMx->CCMR1 = 0x1000;

(b)

TIMx_CCMR1:	OC2CE	OC2M	OC2PE	OC2FE	CC2S	OC1CE	OC1M	OC1PE	OC1FE	CC1S
	0	011	0	0	00	0	000	0	0	00

TIMx->CCMR1 = 0x3000;

(c)

TIMx_CCMR2:	OC4CE	OC4M	OC4PE	OC4FE	CC4S	OC3CE	OC3M	OC3PE	OC3FE	CC3S
	0	000	0	0	00	0	011	0	0	00

TIMx->CCMR2 = 0x0030;

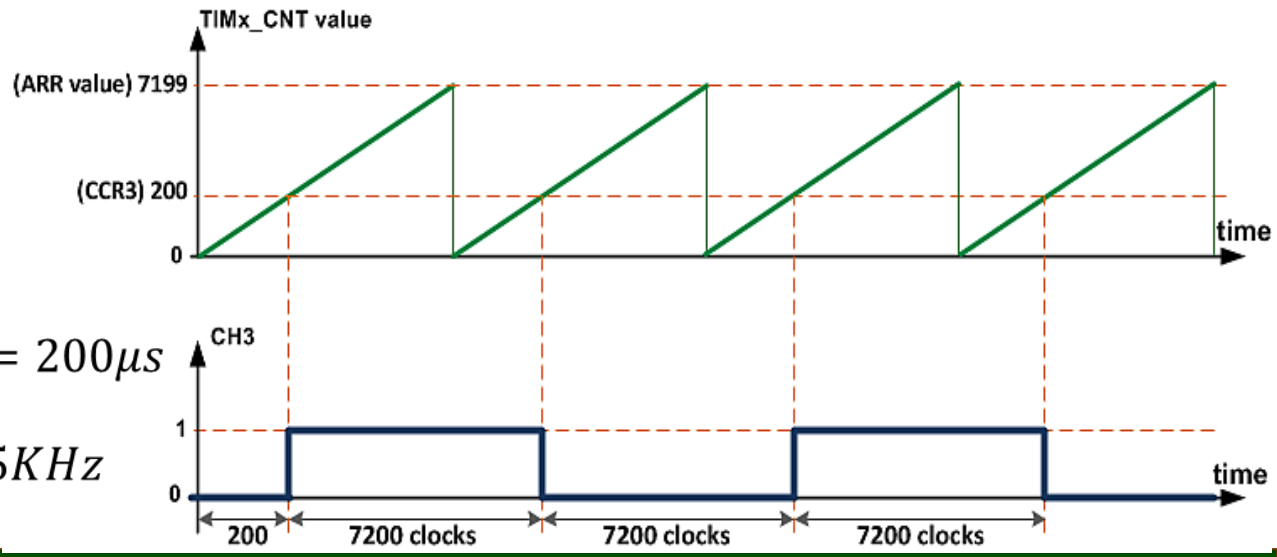
Example: Draw the wave generated by the following program. Calculate the frequency of the generated wave.

```
#include <stm32f10x.h>
int main( ) {
        RCC->APB2ENR |= 0xFC;      /* enable GPIO clocks */
        RCC->APB1ENR |= (1<<0);    /* enable TIM2 clock */
        GPIOA->CRL = 0x44444B44;  /* PA2: alternate func. output */

        TIM2->CCR3 = 200;
        TIM2->CCER = 0x1 << 8; /* CC3P = 0, CC3E = 1 */
        TIM2->CCMR2 = 0x0030;  /* toggle channel 3 */
        TIM2->ARR = 10000-1;
        TIM2->CR1 = 1;      /* start counting up */

        while(1)   { }
}
```

$$T_{Timer\ clock} = \frac{1}{72MHz}$$

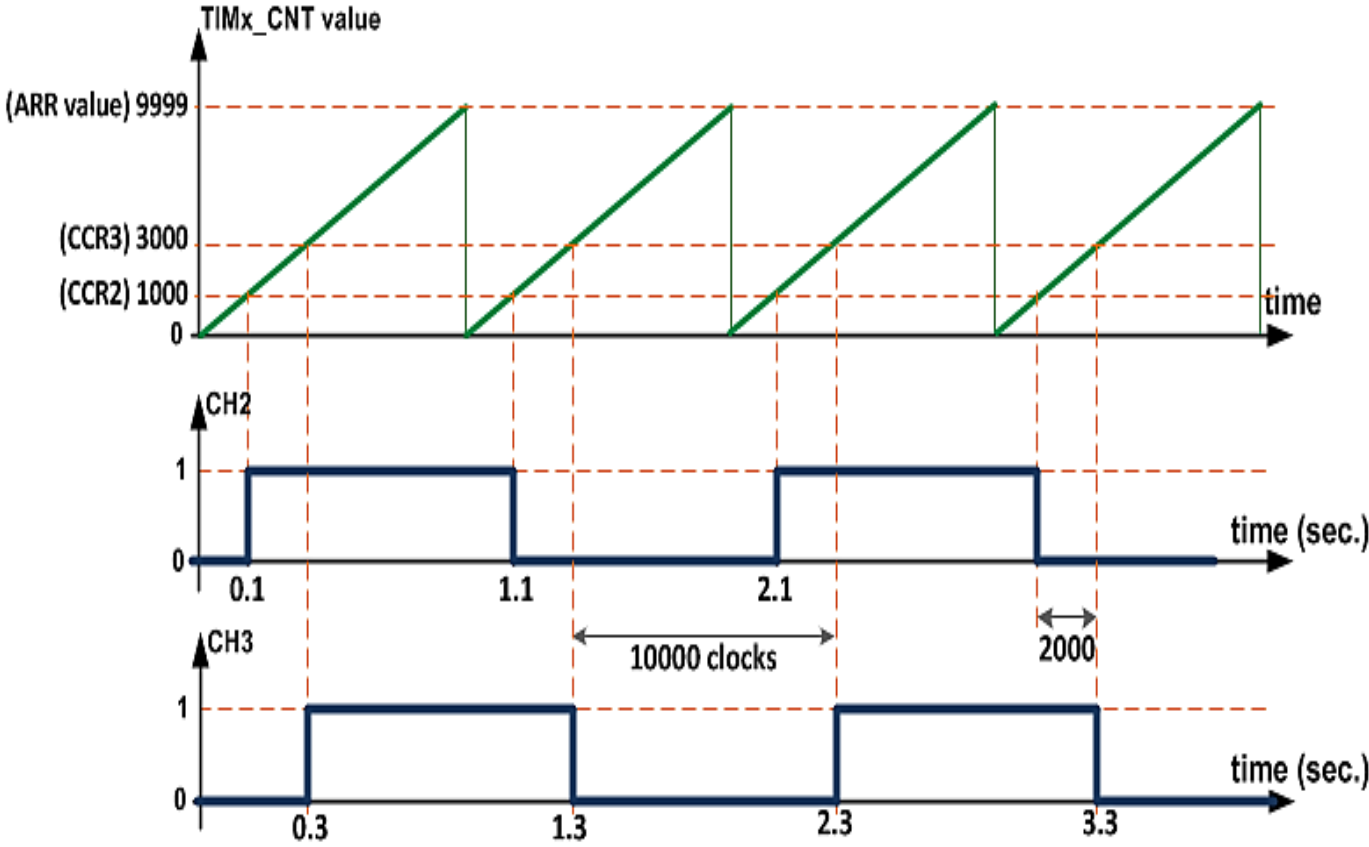
$$\Rightarrow T_{wave} = 2 \times 7200 \times \frac{1}{72M} = 200\mu s$$

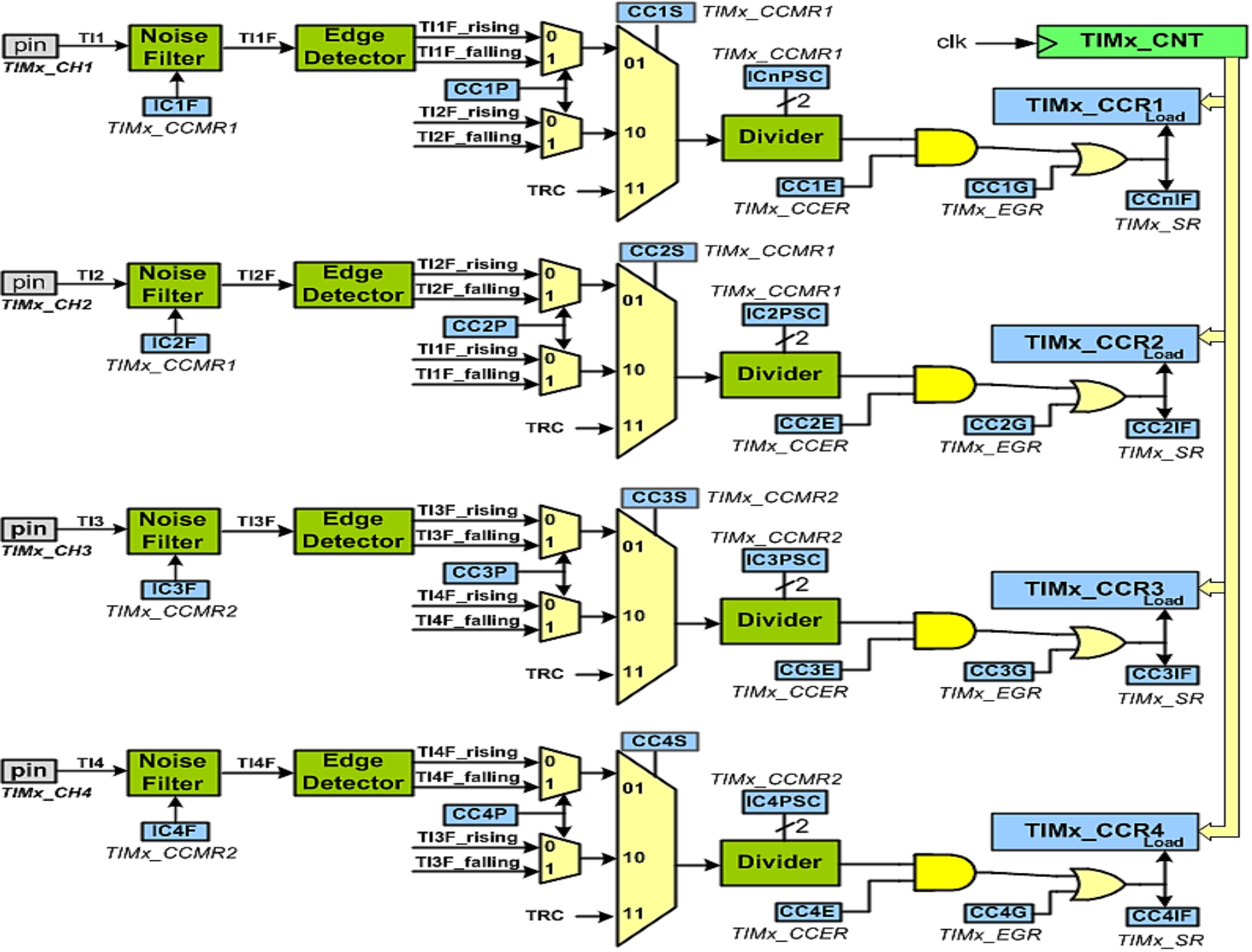
$$\Rightarrow F_{wave} = \frac{1}{T_{wave}} = \frac{1}{200\mu} = 5KHz$$



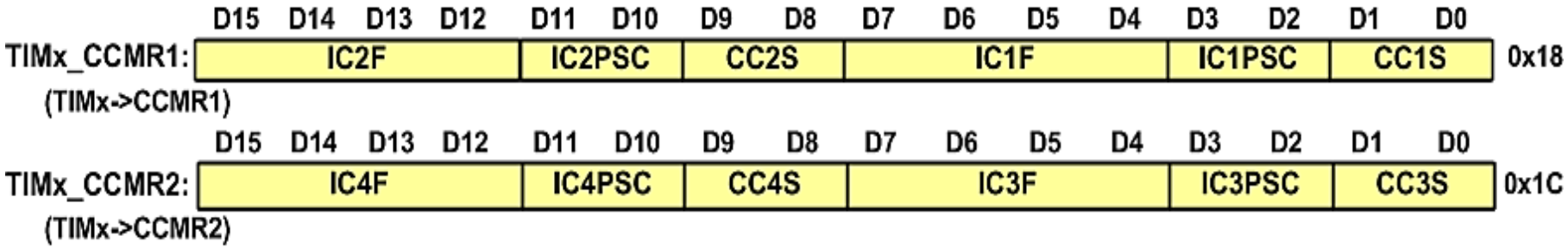
Example: Draw the waves generated by the following program.

```
#include <stm32f10x.h>
int main( ) {
  RCC->APB2ENR |= 0xFC;          /* enable GPIO clocks */
  RCC->APB1ENR |= (1<<0);        /* enable TIM2 clock */
  GPIOA->CRL = 0x44444BB4;       /* PA2(CH3), PA1(CH2): alternate func. output */
  TIM2->CCR2 = 1000;
  TIM2->CCR3 = 3000;
  TIM2->CCER = (0x1<
  TIM2->CCMR1 = 0x3
  TIM2->CCMR2 = 0x0
  TIM2->PSC = 7200-1
  TIM2->ARR = 10000-
  TIM2->CR1 = 1;
  while(1) { }
}
```




pin
TI1
TIMx_CH1
Noise Filter
IC1F
TIMx_CCMR1
TI1F
Edge Detector
TI1F_rising
TI1F_falling
CC1P
TI2F_rising
TI2F_falling
TRC
CC1S
TIMx_CCMR1
01
10
11
ICnPSC
2
Divider
CC1E
TIMx_CCER
clk
TIMx_CNT
TIMx_CCR1
Load
CC1G
TIMx_EGR
CCnIF
TIMx_SR
pin
TI2
TIMx_CH2
Noise Filter
IC2F
TIMx_CCMR1
TI2F
Edge Detector
TI2F_rising
TI2F_falling
CC2P
TI1F_rising
TI1F_falling
TRC
CC2S
TIMx_CCMR1
01
10
11
IC2PSC
2
Divider
CC2E
TIMx_CCER
TIMx_CCR2
Load
CC2G
TIMx_EGR
CC2IF
TIMx_SR
pin
TI3
TIMx_CH3
Noise Filter
IC3F
TIMx_CCMR2
TI3F
Edge Detector
TI3F_rising
TI3F_falling
CC3P
TI4F_rising
TI4F_falling
TRC
CC3S
TIMx_CCMR2
01
10
11
IC3PSC
2
Divider
CC3E
TIMx_CCER
TIMx_CCR3
Load
CC3G
TIMx_EGR
CC3IF
TIMx_SR
pin
TI4
TIMx_CH4
Noise Filter
IC4F
TIMx_CCMR2
TI4F
Edge Detector
TI4F_rising
TI4F_falling
CC4P
TI3F_rising
TI3F_falling
TRC
CC4S
TIMx_CCMR2
01
10
11
IC4PSC
2
Divider
CC4E
TIMx_CCER
TIMx_CCR4
Load
CC4G
TIMx_EGR
CC4IF
TIMx_SR

CCMR1 and CCMR2

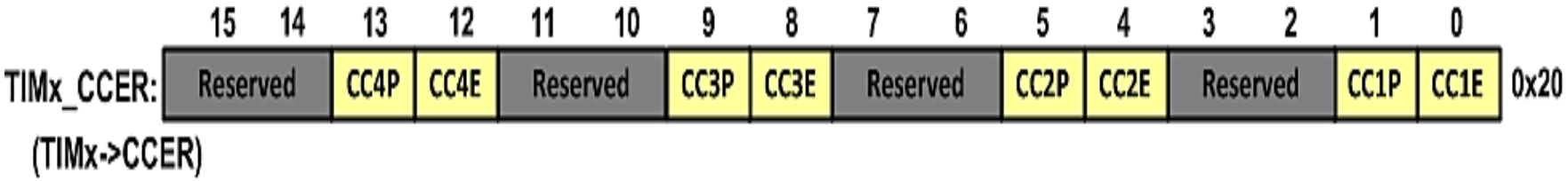


- CCnS (Compare/Capture n Selection)
- ICnPSC (Input Capture Prescaler)
- ICnF (Input Capture Filter)

CC1S	Mode
00	Output Compare mode
01	Input from TIMx_CH1
10	Input from TIMx_CH2
11	Input from TRC

CC2S	Mode
00	Output Compare mode
01	Input from TIMx_CH2
10	Input from TIMx_CH1
11	Input from TRC

ICnF	N	Fsampling	ICnF	N	Fsampling	ICnF	N	Fsampling	ICnF	N	Fsampling
0000	1	f_{DTS}	0100	6	$f_{DTS}/2$	1000	6	$f_{DTS}/8$	1100	8	$f_{DTS}/16$
0001	2	f_{CK_INT}	0101	8	$f_{DTS}/2$	1001	8	$f_{DTS}/8$	1101	5	$f_{DTS}/32$
0010	4	f_{CK_INT}	0110	6	$f_{DTS}/4$	1010	5	$f_{DTS}/16$	1110	6	$f_{DTS}/32$
0011	8	f_{CK_INT}	0111	8	$f_{DTS}/4$	1011	6	$f_{DTS}/16$	1111	8	$f_{DTS}/32$



- CCnE (Compare/Capture n Enable)
  - 0: disable
  - 1: enable

TIM_SR (Status Register)

- CCnIF (Capture/Compare n Input Flag)
  - 0: No capture occurred,
  - 1: capture occurred
- CCnOF (Capture/Compare n Over-capture flag)
  - 0: No over-capture,
  - 1: over-capture detected)

Example

- Find the CCMR1 and CCER values to configure channel 1 for capturing pin TIMx_CH1 on rising edge, and no division. The noise filter should accept signals after 4 timer clocks.

Solution:

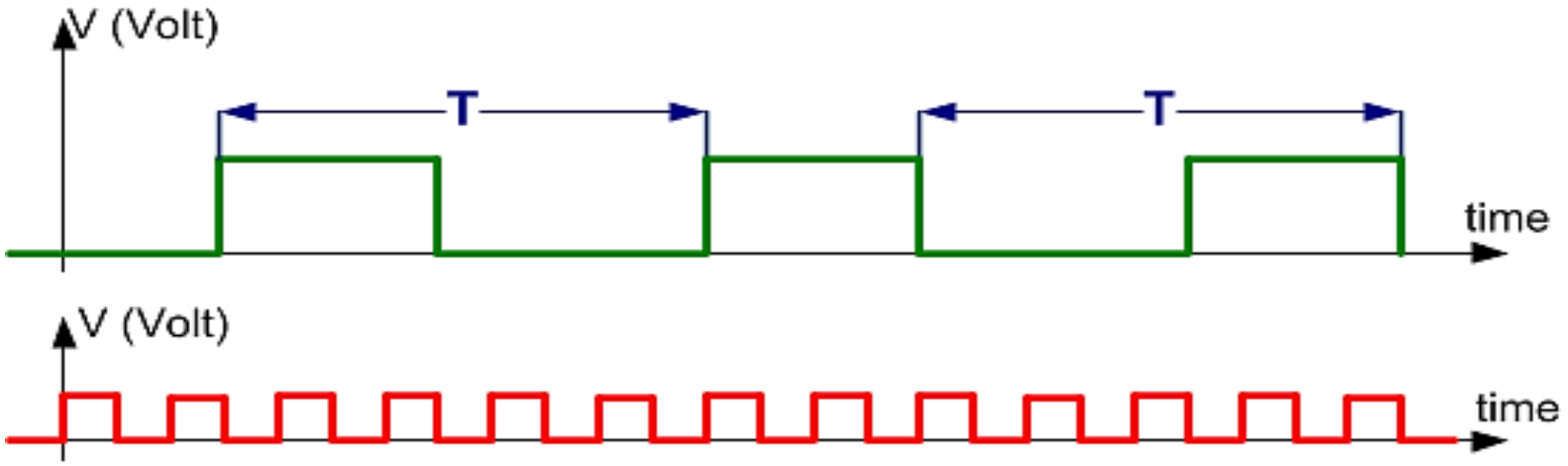
TIMx_CCMR1:	IC2F	IC2PSC	CC2S	IC1F	IC1PSC	CC1S
	0000	00	00	0010	00	01

```
TIMx->CCMR1=0x0021;
```

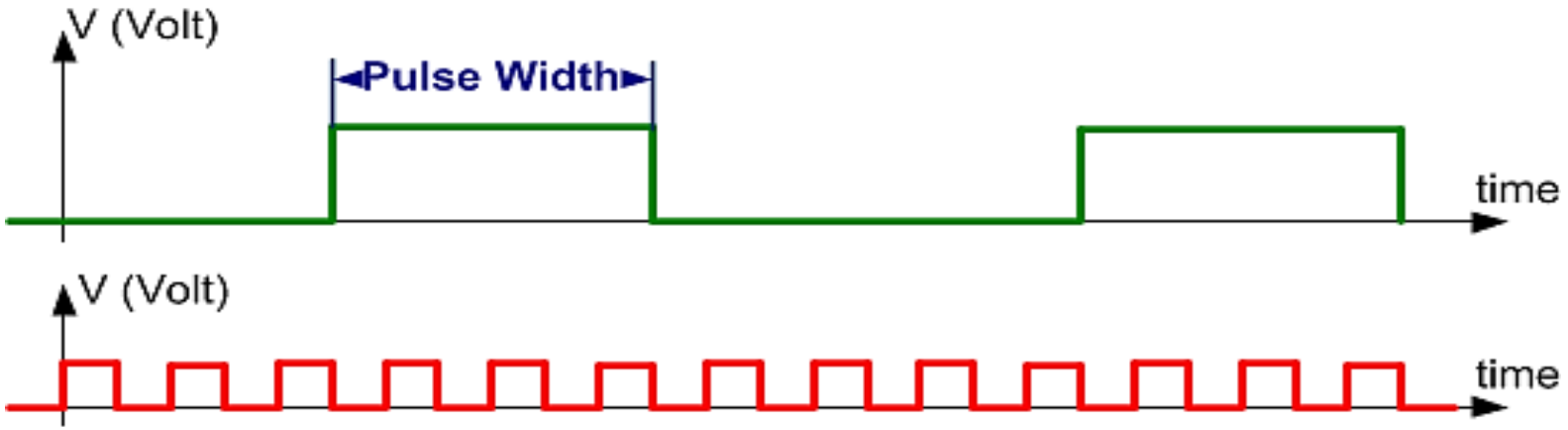
TIMx_CCER	Res.	CC4P	CC4E	Res.	CC3P	CC3E	Res.	CC2P	CC2E	Res.	CC1P	CC1E
	00	0	0	00	0	0	00	0	0	00	0	1

```
TIMx->CCER=0x0001;
```

Measuring Period and Pulse width



(a) Measuring Period in Terms of the Number of Clocks Counted by the Timer



(b) Measuring Pulse Width in Terms of the Number of Clocks Counted by the Timer

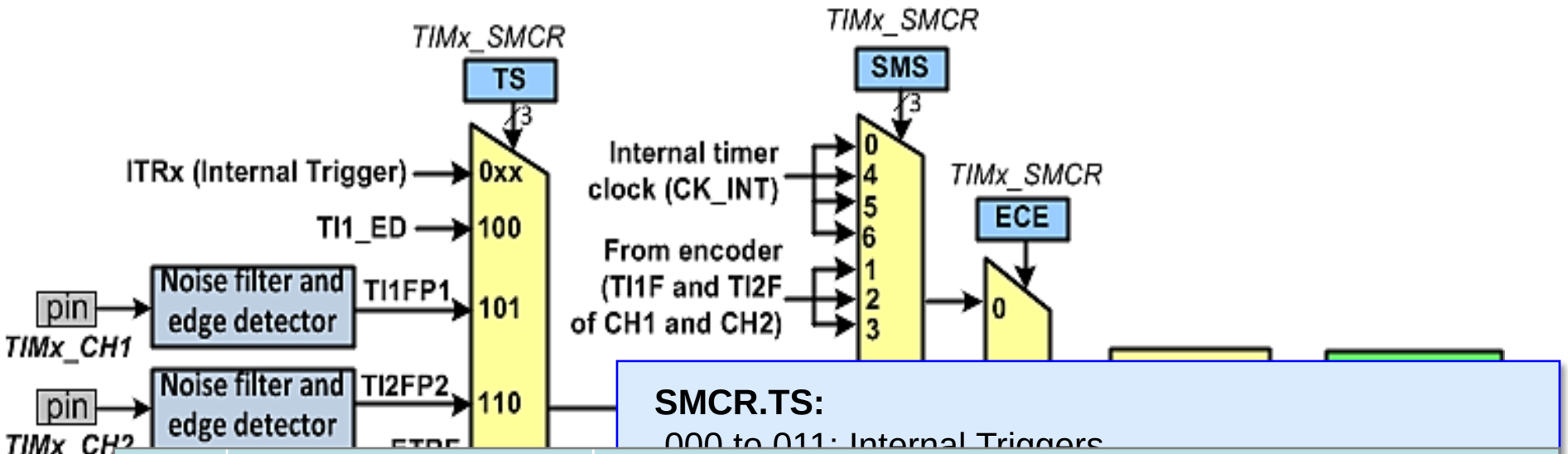
Example: The program measures the frequency of the wave and reports through usart1

```
#include <stm32f10x.h>
#include <stdio.h>

void usart1_init(void);
void usart1_sendByte(unsigned char c);
void usart1_sendInt(unsigned int i);
void usart1_sendStr(char *str);

int main( ) {
  RCC->APB2ENR |= (0xFC| (1<<14)); /* enable GPIO clocks and USART1 clock */
  RCC->APB1ENR |= (1<<0);          /* enable TIM2 clock */

  GPIOA->CRL = 0x44444844;         /* PA2(CH3): input pull-up */
  GPIOA->ODR |= (1<<2);

  TIM2->CCMR2 = 0x001;  /* Pin TIM2_CH3 as input for channel 3 */
  TIM2->CCER = 0x1 << 8; /*CC3P = 0 (rising), CC3E = 1 */
  TIM2->PSC = 7200-1;
  TIM2->ARR = 50000-1;
  TIM2->CR1 = 1;               /* start counting up */
```

usart1_init()

Using Timer as a Counter

Timers Clock Sources

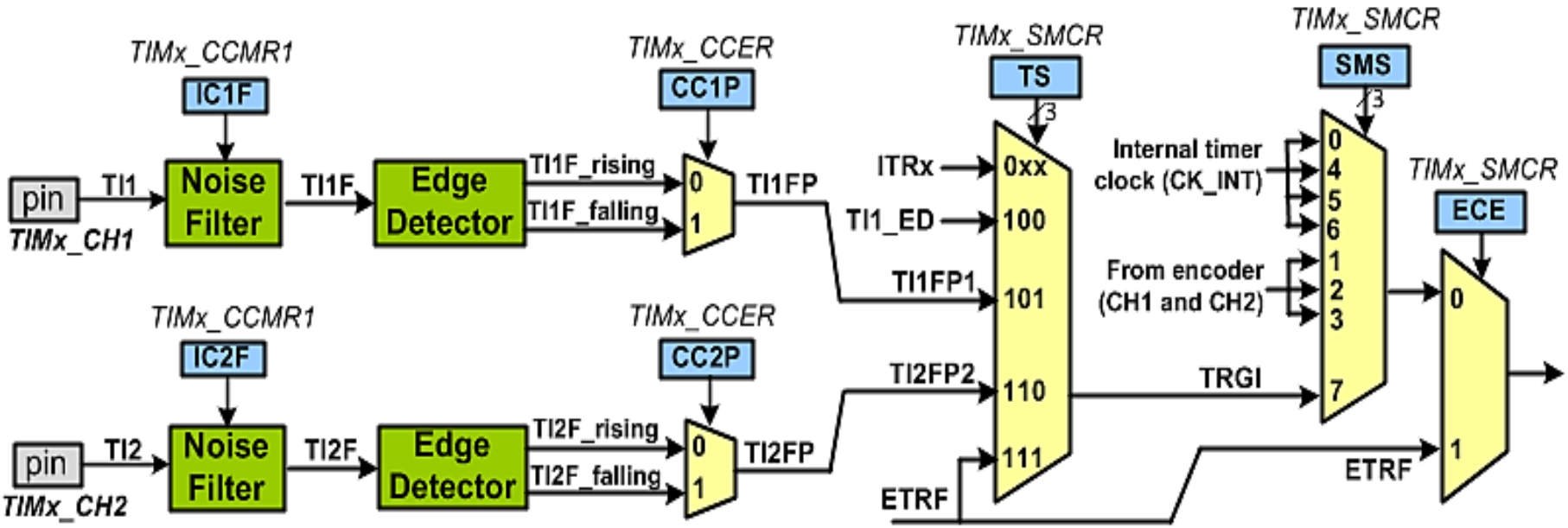


SMCR.TS:

000 to 011: Internal Triggers

SMS	Mode	Description
0	Timer mode	clocked by the internal clock
1	Encoder mode 1	counter counts up/down on TI2FP1 edge depending on TI1FP2 level.
2	Encoder mode 2	counter counts up/down on TI1FP2 edge depending on TI2FP1 level.
3	Encoder mode 3	counter counts up/down on both TI1FP2 and TI2FP2 edges depending on the level of the other.
4	Reset mode	The counter is reset and the registers are updated on each rising edge of TRGI signal. (See the manual.)
5	Gated mode	As long as the TRGI signal is high, the timer counts. The timer stops counting when the TRGI signal is low.
6	Trigger mode	at the rising edge of the TRGI signal, it starts counting.
7	External clock mode 1	Counts the rising edges of TRGI signal.

TIMx_CH1 and TIMx_CH2 External Clock Circuit



Example: Find the SMCR value to choose (a) TIMx_CH1 (b) TIMx_CH2 as the clock source for the counter.

(a)

TIMx_SMCR	ETP	ECE	ETPS	ETF	MSM	TS	Res.	SMS
	0	0	00	0000	0	101	0	111

TIMx_SMCR = 0x57;

(b)

TIMx_SMCR	ETP	ECE	ETPS	ETF	MSM	TS	Res.	SMS
	0	0	00	0000	0	110	0	111

TIMx_SMCR = 0x67;

Example: Write a program that counts the input pulses of TIMx_CH2 on rising edge and sends the value of counter through USART1.

```
int main( ) {
  RCC->APB2ENR |= (0xFC| (1<<14)); // enable GPIO clocks and USART1 clock
  RCC->APB1ENR |= (1<<0);          /* enable TIM2 clock */

  usart1_init();                   /* initialize the usart1 */

  GPIOA->CRL = 0x44444484;         /* PA1(CH2): input pull-up */
  GPIOA->ODR |= (1<<1);

  TIM2->CCMR1 = 0x0000;  /* no filter */
  TIM2->CCER = 0; /* CC2P = 0 (rising) */
  TIM2->SMCR = 0x67;      /* TIM2_CH2 as clock source */
  TIM2->ARR = 50000-1; /* count from 0 to 49999 then roll over to 0 */
  TIM2->CR1 = 1; /* start counting up */

  while(1) {
    usart1_sendInt(TIM2->CNT);       /* send the counter value through serial */
    usart1_sendStr("\n\r"); /* go to new line */
    delay_ms(100);
  }
}
```

Example: A clock pulse is fed into pin TIM2_CH1(PA0). Write a program that toggles PC13 every 100 pulses.

```
#include <stm32f10x.h>
int main( ) {
  RCC->APB2ENR |= 0xFC;          /* enable GPIO clocks and USART1 clock */
  RCC->APB1ENR |= (1<<0);        /* enable TIM2 clock */

  GPIOA->CRL = 0x44444448;       /* PA0(CH1): input pull-up */
  GPIOA->ODR |= (1<<0);

  GPIOC->CRH = 0x44344444;       /* PC13 as output */

  TIM2->CCMR1 = 0x0000;          /* no filter */
  TIM2->CCER = 0x1 << 1;         /* CC0P = 1 (falling) */
  TIM2->SMCR = 0x57;             /* TIM2_CH1 as clock source */
  TIM2->ARR = 100-1;
  TIM2->CR1 = 1;
  while(1) {
    if((TIM2->SR&1) != 0) {
      TIM2->SR = 0;
      GPIOC->ODR ^= (1<<13);
    }
  }
}
```

ETR (External clock) input Block