UNIVERSITEIT iYUNIVESITHI STELLENBOSCH UNIVERSITY 100 1918 · 2018

forward together · saam vorentoe · masiye phambili

Computer Systems / Rekenaarstelsels 245 - 2020

Lecture 13

Timers and Counter: Part 1
Tydhouers en Tellers: Part 1

Dr Rensu Theart & Dr Lourens Visagie

Lecture Overview

- What is a Timer/Counter?
- Timers/Counters
  - Used as a Counter
  - Used as a Timer
  - Counting modes (Upcounting, downcounting, center-aligned mode)
  - Output compare
  - Input capture
  - Timers on the STM32F411VE

Introduction

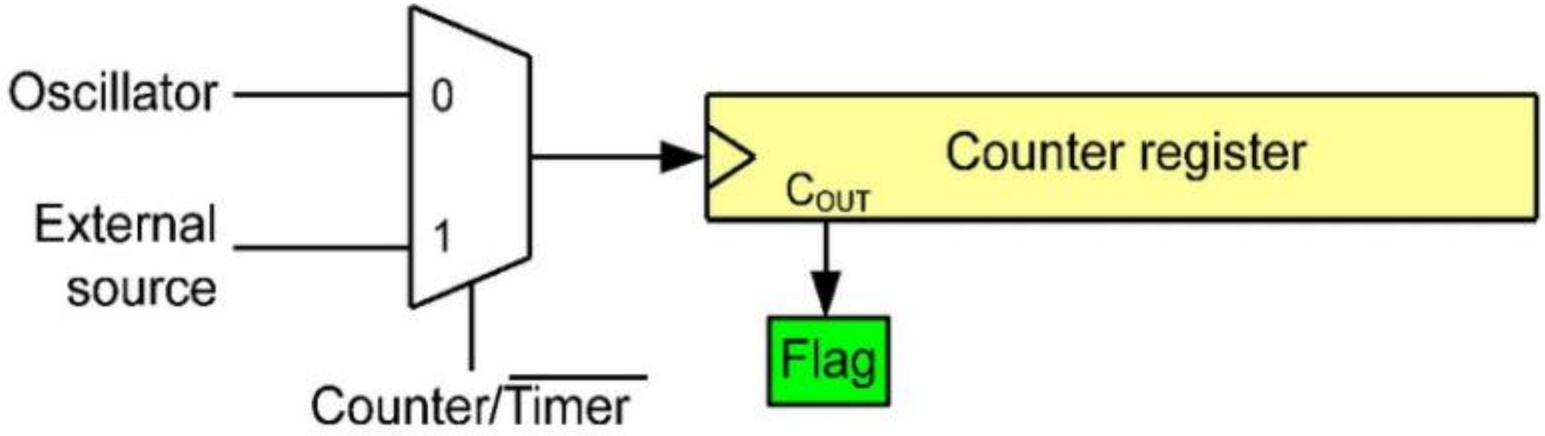
- Embedded systems commonly need to measure time.
- For example, a microwave oven needs a timer:
  1. to keep track of the time of day
  2. another to measure how long to cook.
  3. It might use yet another to generate pulses to the motor spinning the platter, and
  4. a fourth to control the power setting by only activating the microwave's energy for a fraction of every second.

- Embedded systems also commonly need to count the number of occurrence of something.

- Timers for the STM32F4 is a very big topic. Covered in pages 238-461 of the reference manual. (That's more than 25% of the entire reference manual).

What is a Timer/Counter?

- A register that counts up (or down) with every transition on an input signal.
- **Option 1**: Input signal is a periodic clock signal with known frequency: Count register can be used for time-keeping or performing actions with controlled timing. This is usually called a **Timer**
- **Option 2:** Input signal is an external signal without known timing: Count register can be used to count number of pulses on input signal. This is usually called a **Counter**.



Timer/Counters

- Set-up of timer/counter module is through memory mapped registers
  - Counter/timer clock source selection
  - Up/down direction
  - Prescale value
  - Auto-reload value
  - Interrupt enable
- One-shot or periodic
- Timers are also used as input to other peripherals:
  - Control analog-to digital conversion rate
  - Bit-rate of serial communications

Timer/Counters

- The timer count registers (TIMx_CNT) are usually 16 or 32-bits wide.
  - The more bits that hold the count the greater the counting range.
- Timer will count up (or down) until maximum value of the register or an **auto-reload value**.
  - If the counter increases past this point, the count register will be reset (go to zero) and interrupt is generated (if enabled).
- Auto-reload value is another register (TIMx_ARR), which has the same width as the count register that can be used in different ways:
  - In **upcounting** mode, the counter counts from 0 to the auto-reload value, then restarts.
  - In **downcounting** mode, the counter counts from the auto-reload value down to 0, then restarts.
  - Interrupt generated every **update event (UEV)**, e.g. when counter overflows/underflows.
- Counters also have a 16-bit **prescaler** allowing for the counter clock frequency to be divided by any factor between 1 and 65536.

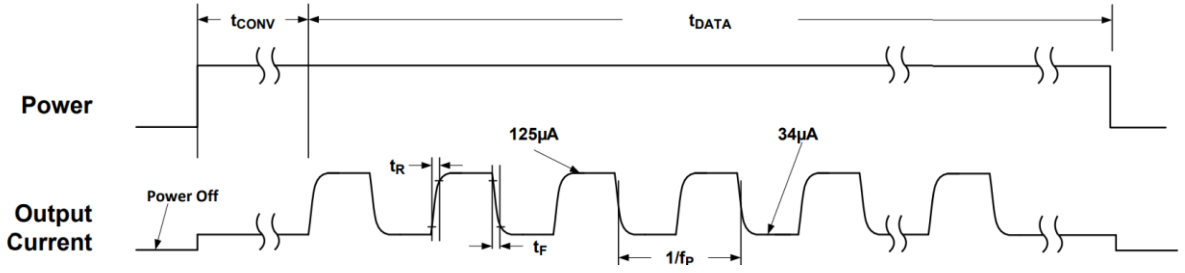
Timer/Counters

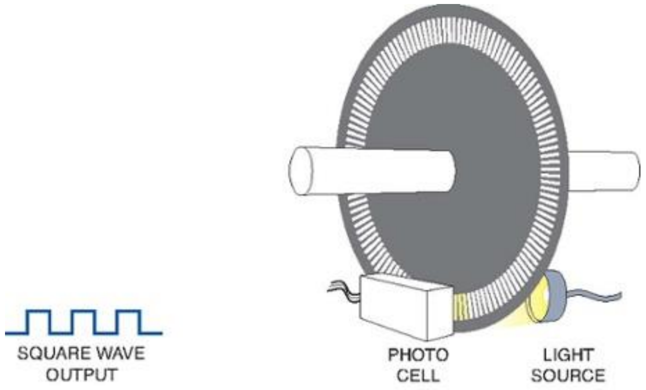
- Most Timers have 4 independent channels.
- Different channels can be used for different purposes:
  - Output Compare
  - Input Capture
  - PWM generation
  - One-pulse mode output
- Interrupts can be generated on the following events:
  - Update: counter overflow/underflow, counter initialization
  - Trigger event (counter start, stop, count by trigger)
  - Output compare
  - Input capture
  - (Break input)

Used as a Counter

- Count the number of pulses (in a certain time duration).
- LMT01 Temperature sensor – outputs a number of pulses in 51ms window – no. of pulses varies with temperature.

⇒Use timer/counter with external input to count number of pulses



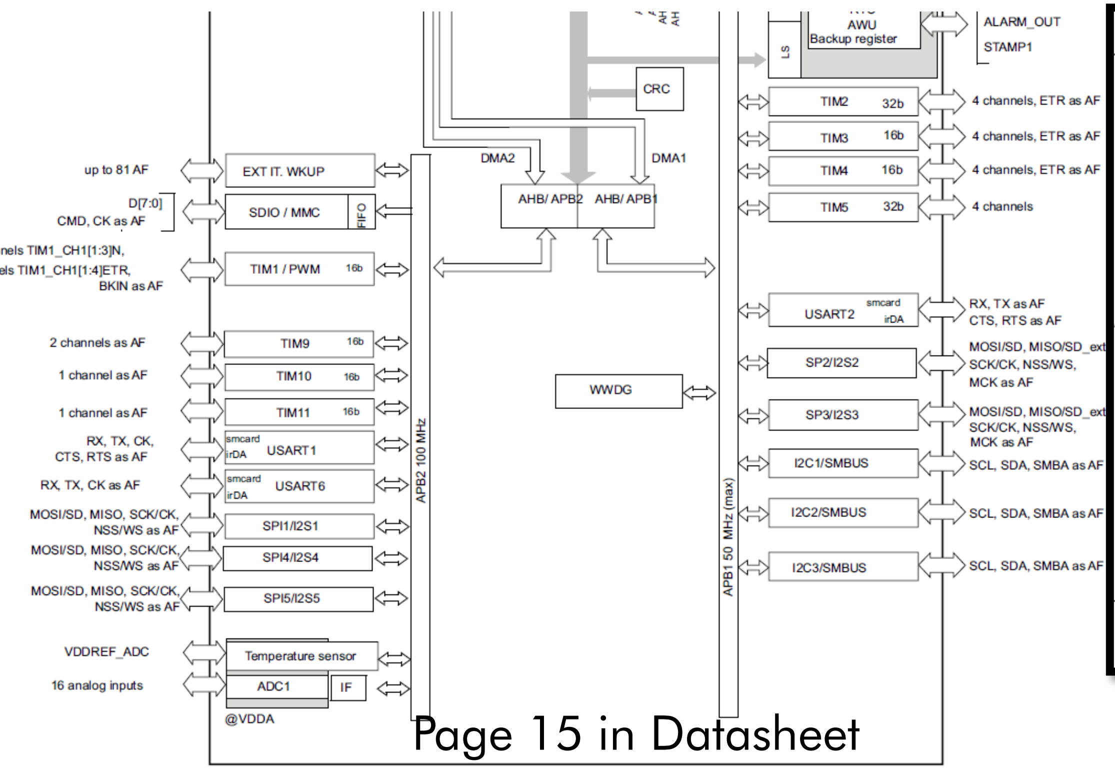


- Optical encoder – outputs pulses at a rate relative to rotation rate.

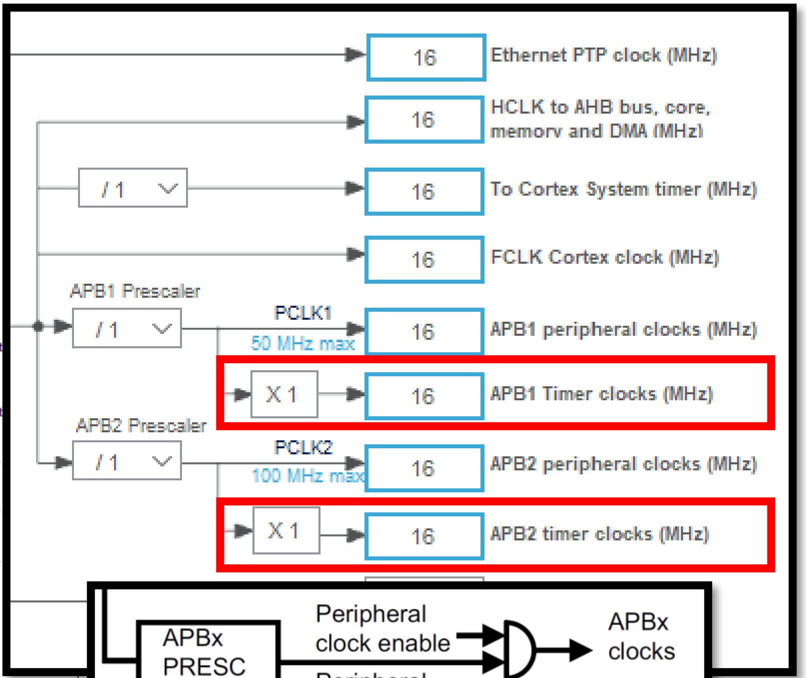
⇒Use timer/counter with external input to measure rotation rate.

Used as a Timer

- There are several timers in the STM32F4.
- They can all driven from the **Advanced Peripheral Bus (APB)** clock signals.
  - TIM2, TIM3, TIM4, TIM5, IWDG, WWDG, RTC connected to **APB1**
  - TIM1, TIM9, TIM10, TIM11 connected to **APB2**



Page 15 in Datasheet



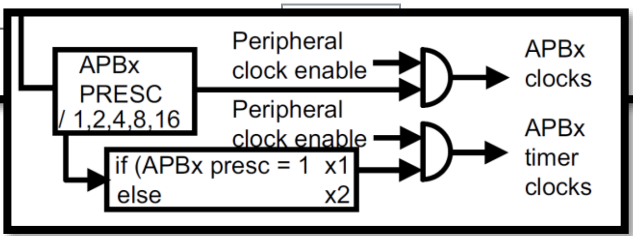
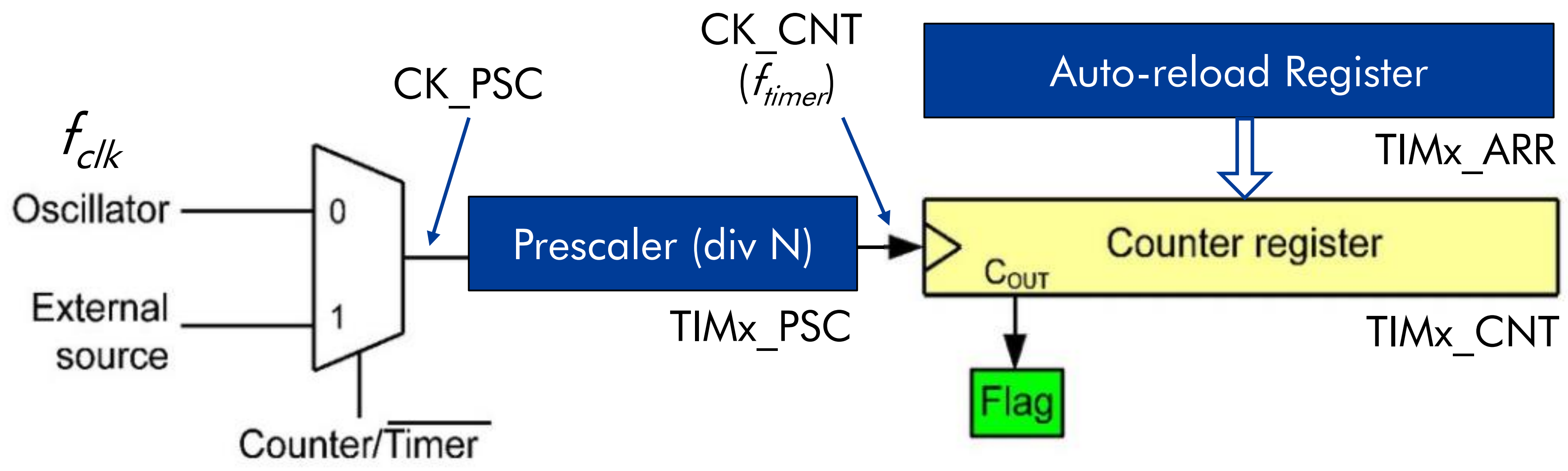


Table 4. Timer feature comparison

Timer type	Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/ compare channels	Complemen- tary output	Max. interface clock (MHz)	Max. timer clock (MHz)
Advanced -control	TIM1	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	Yes	100 APB2	100
General purpose	TIM2, TIM5	32-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	50 APB1	100
	TIM3, TIM4	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	50 APB1	100
	TIM9	16-bit	Up	Any integer between 1 and 65536	No	2	No	100 APB2	100
	TIM10, TIM11	16-bit	Up	Any integer between 1 and 65536	No	1	No	100 APB2	100

Used as a Timer



- When input signal is a periodic signal with fixed frequency (f_{clk}), then the clock counter frequency is equal to:

$$f_{timer} = \frac{f_{clk}}{N}$$

Note: N = TIMx_PSC +1, to ensure divide by 0 is not possible

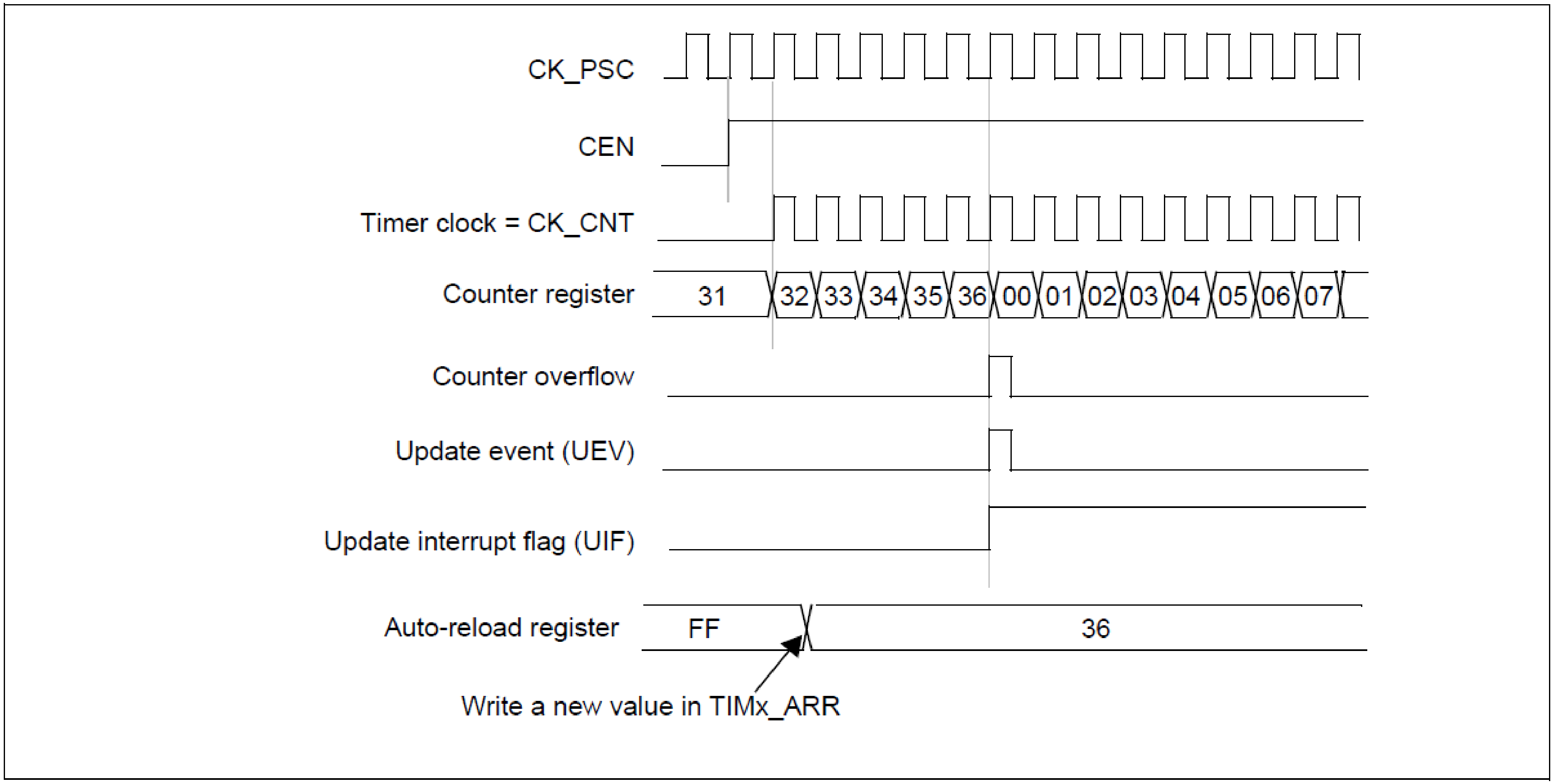
- The time interval with which interrupts are generated, if counting up from 0 to auto-reload value (R) is:

$$T = \frac{(R+1)}{f_{timer}}$$

Note: R + 1, since both 0 and R are included in the count.

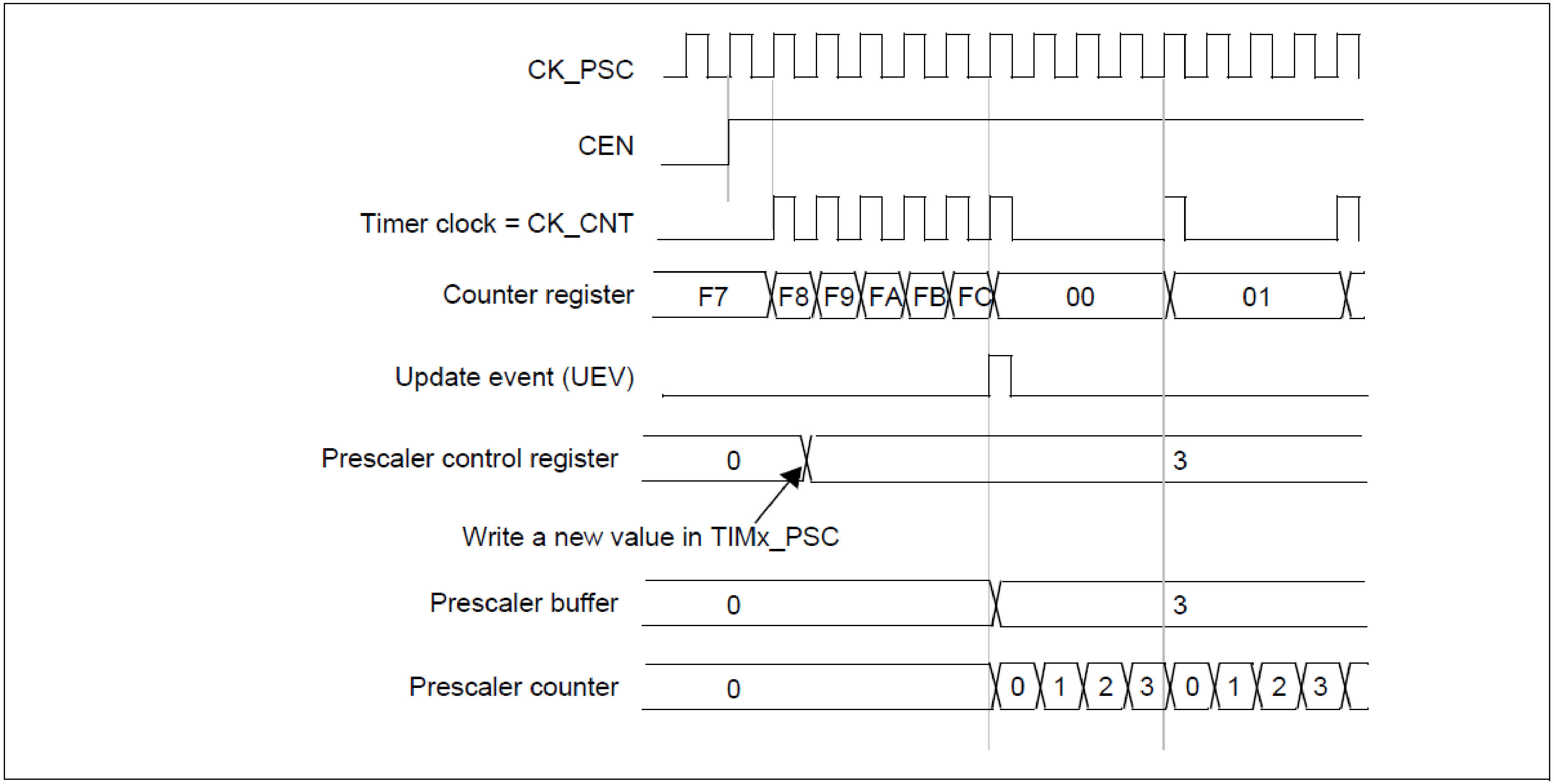
Timers/Counters: Upcounting mode

Figure 46. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)



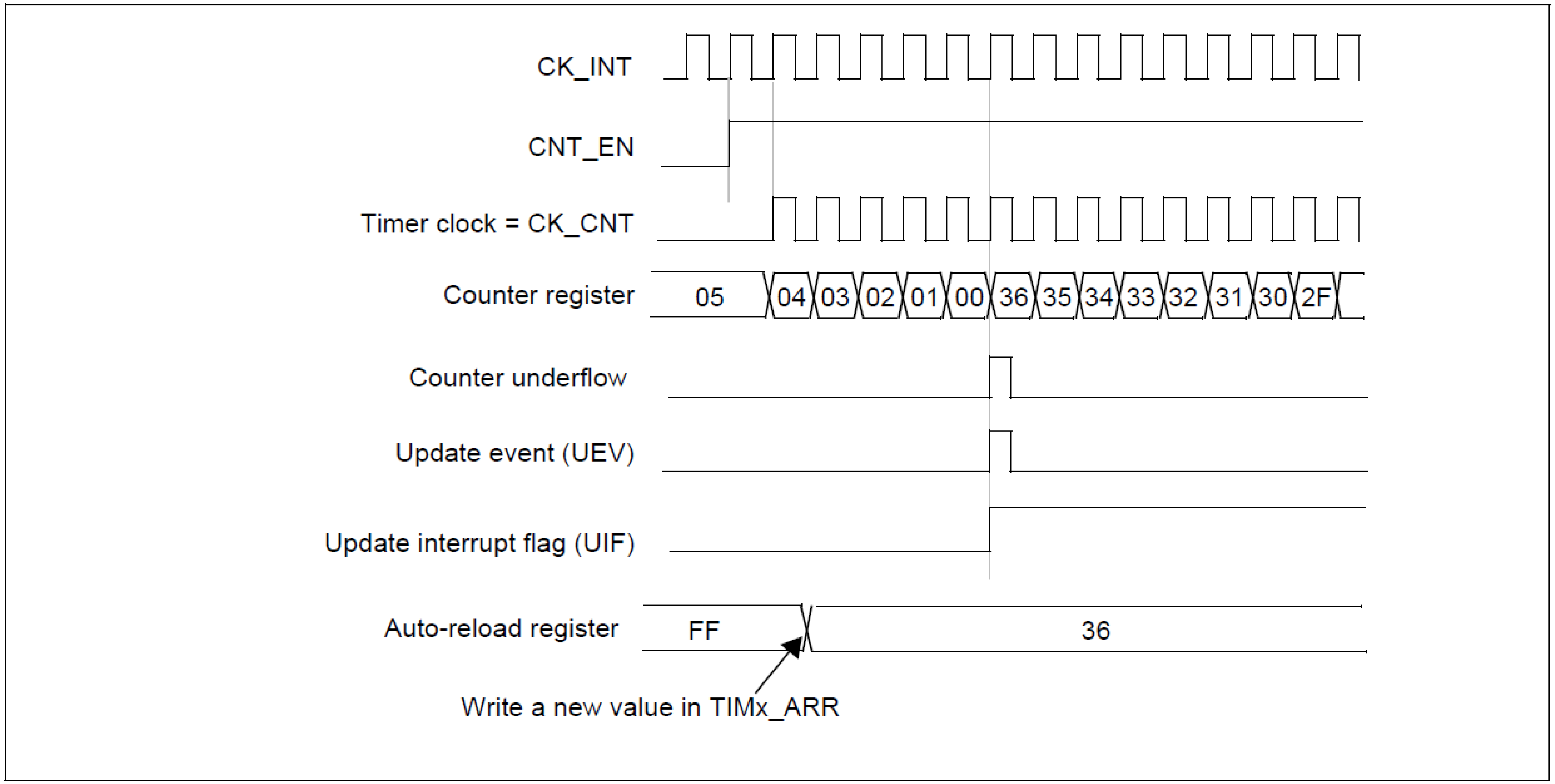
Timers/Counters: Upcounting mode

Figure 41. Counter timing diagram with prescaler division change from 1 to 4



Timers/Counters: Downcounting mode

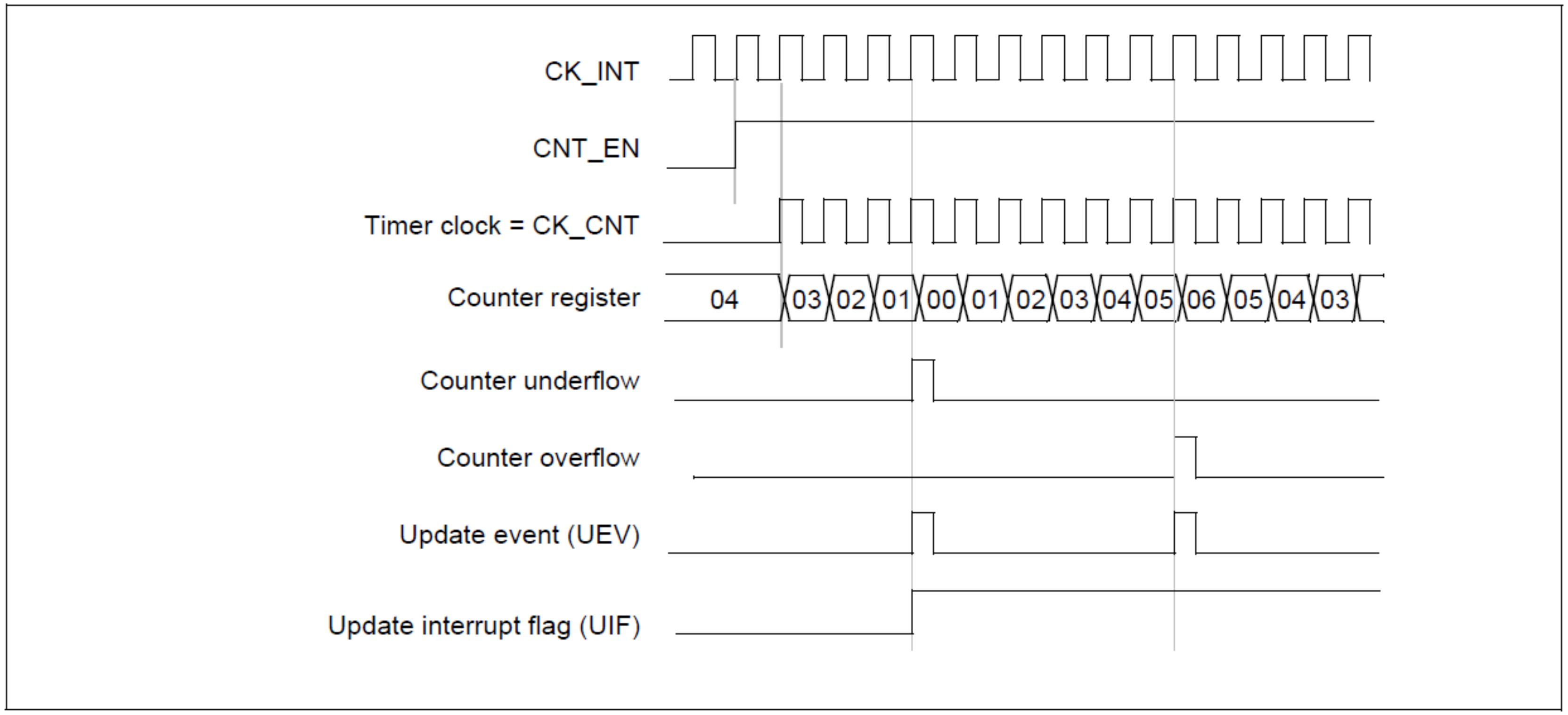
Figure 100. Counter timing diagram, Update event



Timers/Counters: Center-aligned mode (up/down counting)

- In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register) minus 1, generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Figure 101. Counter timing diagram, internal clock divided by 1, TIMx_ARR=0x6



Used as a Timer

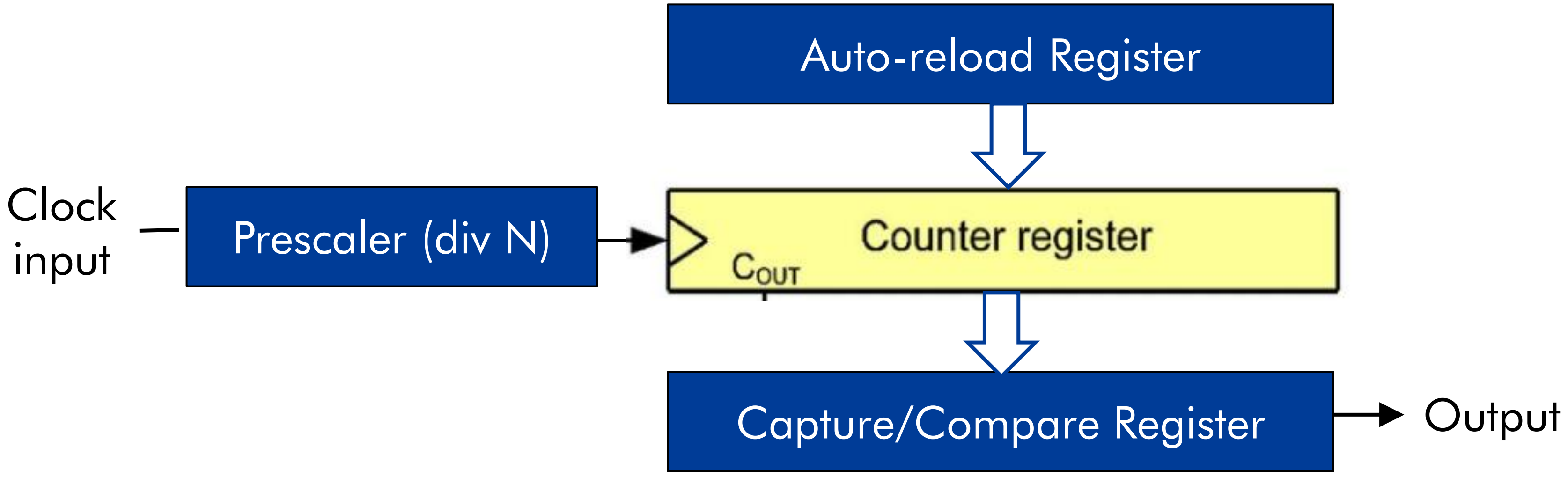
- **Example:** The input clock to the timer module (CK_PSC) is 10MHz, and a prescale value (TIMx_PSC) of 1024 is used. The timer counts up from zero to the reload value (TIMx_ARR). In order to generate interrupts every 1s, what must the reload value be?
- TIMx_ARR $= \frac{10^7}{(1024+1)} - 1 = 9746\ldots$

- The resulting timer interrupt period will not be exactly 1s because of integer round-of. In this case, 0.9990675s .

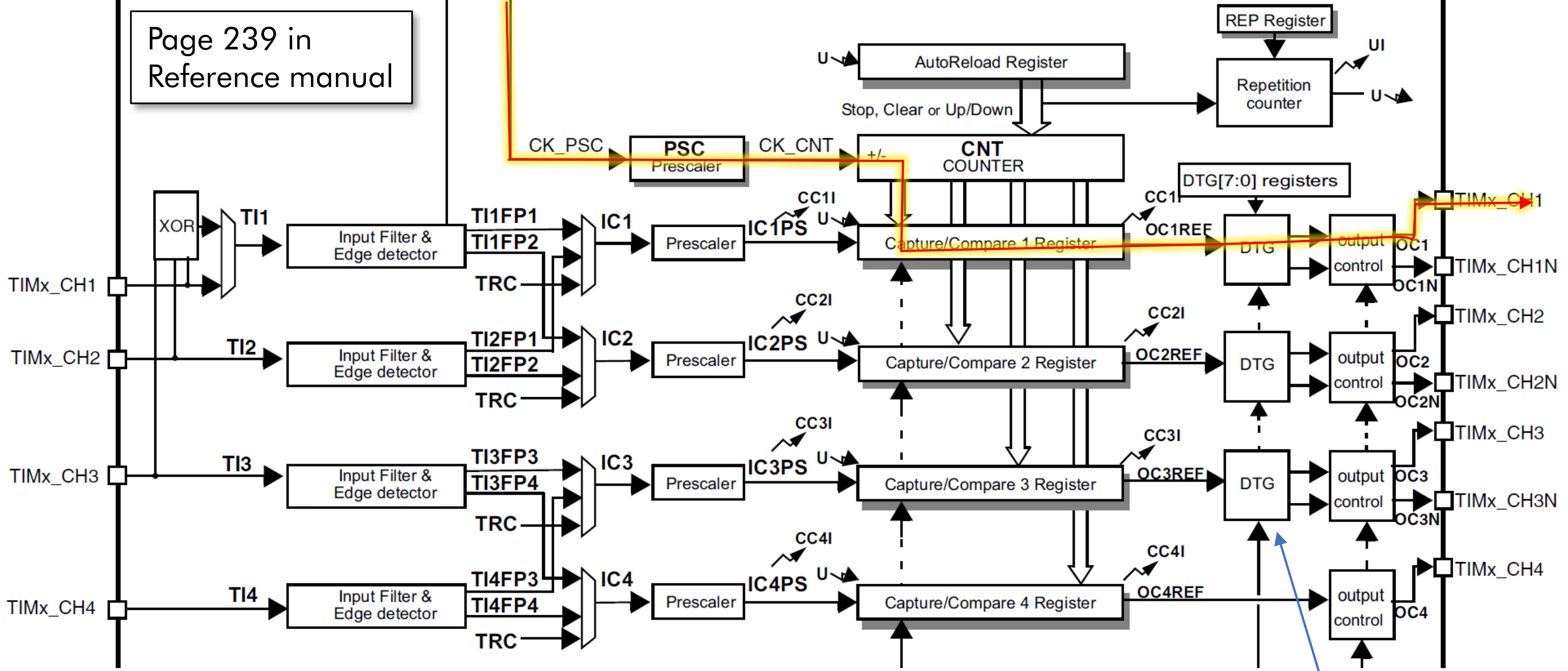
- Sometimes you might need a +1 or -1 in the formula :
  - Counting from 0 to 3: 0, 1, 2, 3 = 4 iterations. I.e. if the reload value is R, the number of input clock intervals between interrupts is R+1
  - Sometimes timer/counter hardware compensates. (check the datasheet for the device for specific implementation)

Output Compare

- **Output compare** is the ability to trigger an output using a timer, without having to interrupt the execution of the processor.
  - Used to control an output waveform or indicate when a period of time has elapsed.
- In output compare mode, counter register value is continuously compared to value in Capture/Compare Register (TIMx_CCRx). If it matches, an output logic level is changed (set to high, clear to low, or toggle output).
- Can also generate an interrupt if enabled.

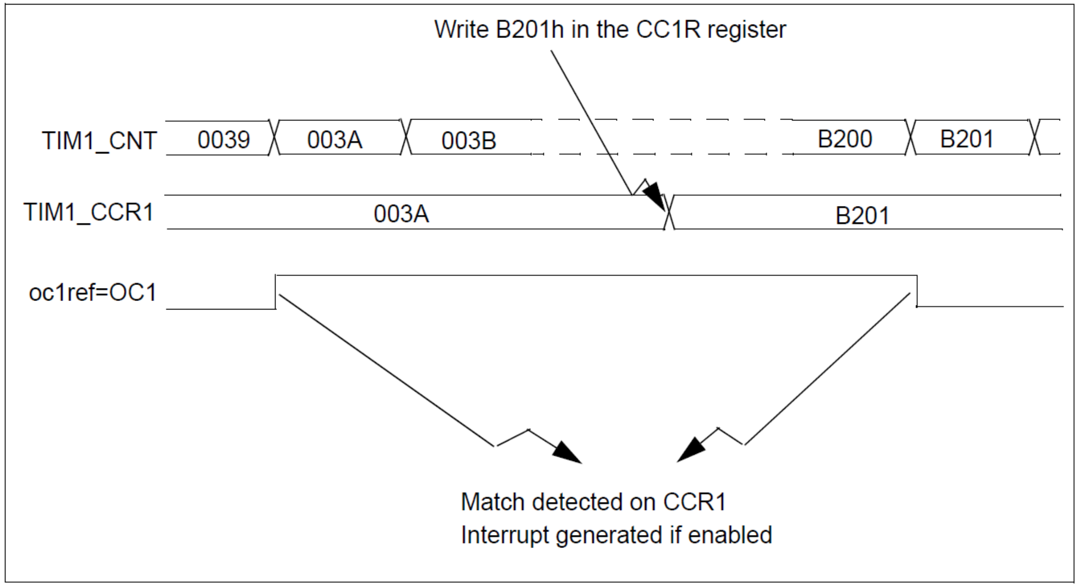


Page 239 in Reference manual



Dead time generation

Figure 70. Output compare mode, toggle on OC1.



Output Compare

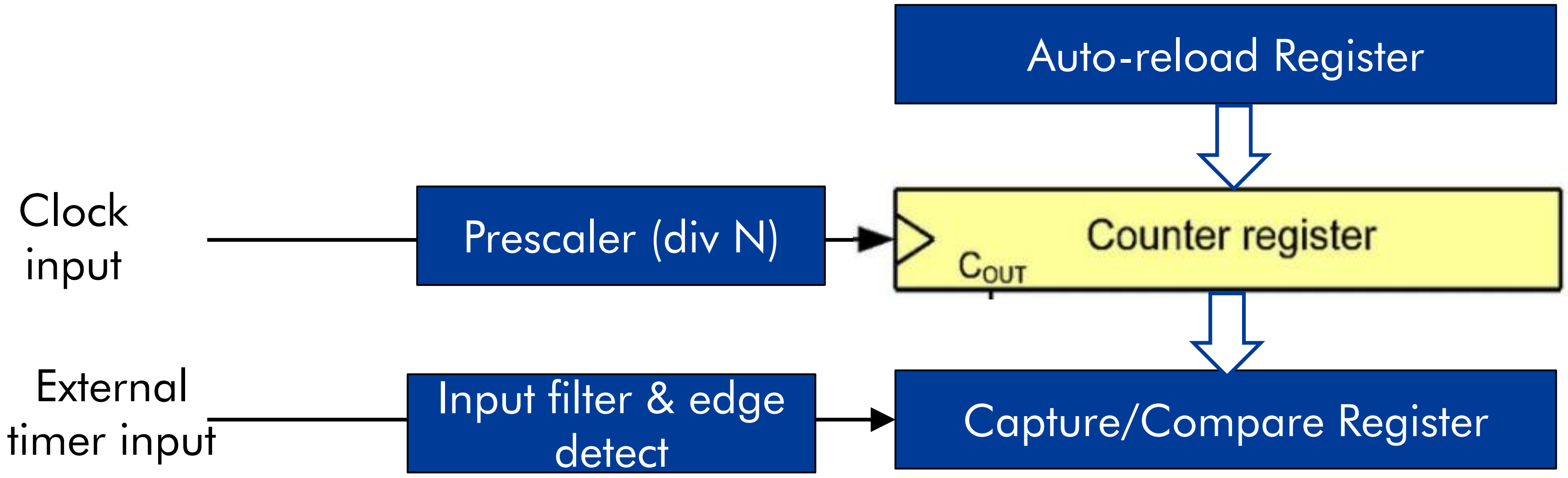
- To start the timer we must include the following line outside our while(1) loop:

```
HAL_TIM_OC_Start(&htim4, TIM_CHANNEL_1);
```

- `htim4` is auto generated by STM32CubeIDE when you set the timer up in the device configuration, matches the Timer set of that output.
- The channel should match the channel that the output pin was set up to use.

Input Capture

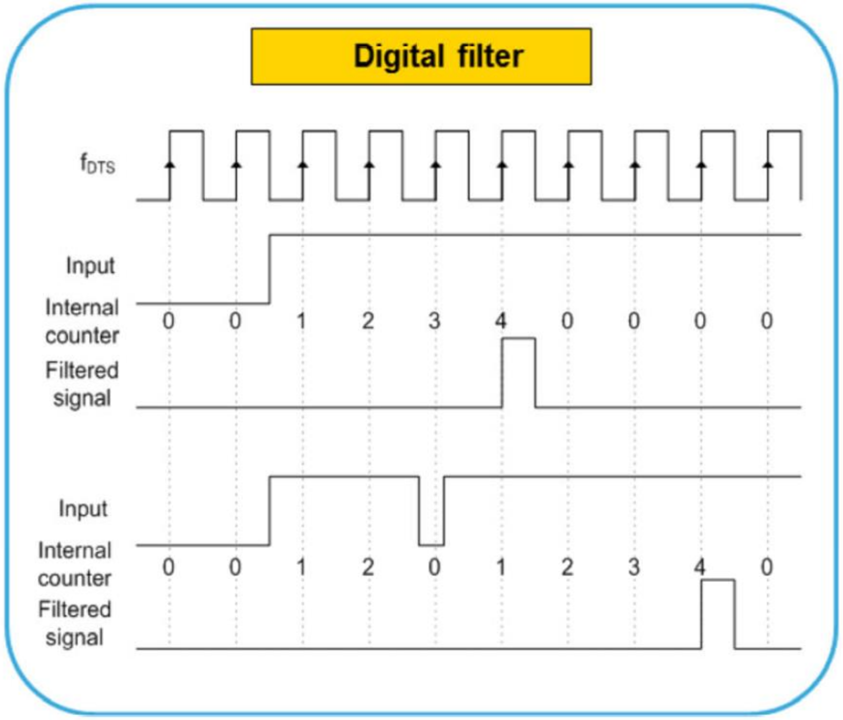
- Timer/Counter is usually set-up to count with clock input – counts up in known intervals.
- The Counter register value is latched to Capture/Compare Registers (TIMx_CCRx) when an external event occurs.
  - A transition detected by the corresponding Input Capture (ICx) signal.
- Can use this to measure time between leading (or trailing) edges in an external signal.
- Can enable interrupt when timer value is captured.



Input Capture

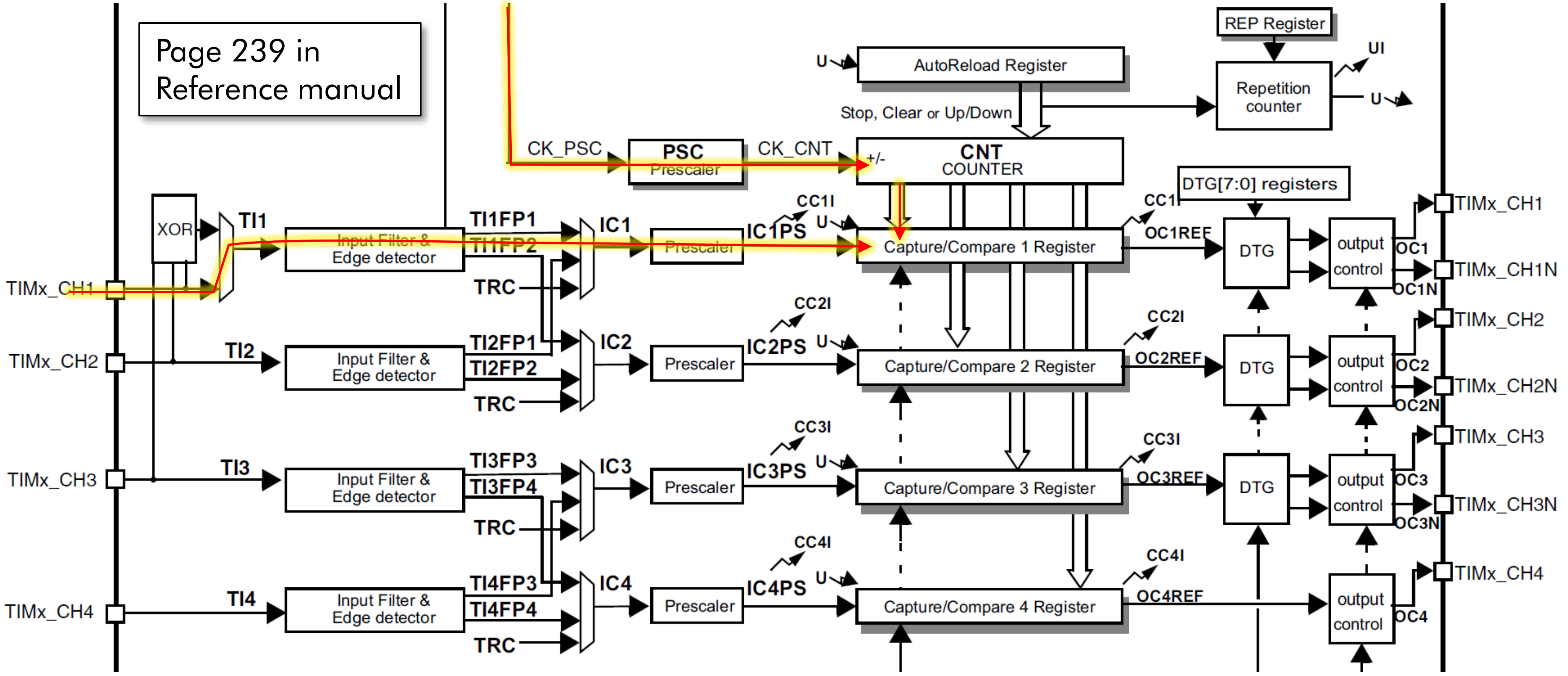
Some extra features include:

- Programmable edge-sensitivity (rising / falling / both)
- Event prescaler (1 capture every 1/2/4/8 events)
  - This can decrease the CPU burden when processing high frequency signals and allows the measurements to be more accurate.
- 4-bit digital filter (for debouncing and noise removal)
  - Allows a clean sample edge to be captured up to 15 sampling period after the active edge.



Input Capture

Page 239 in Reference manual

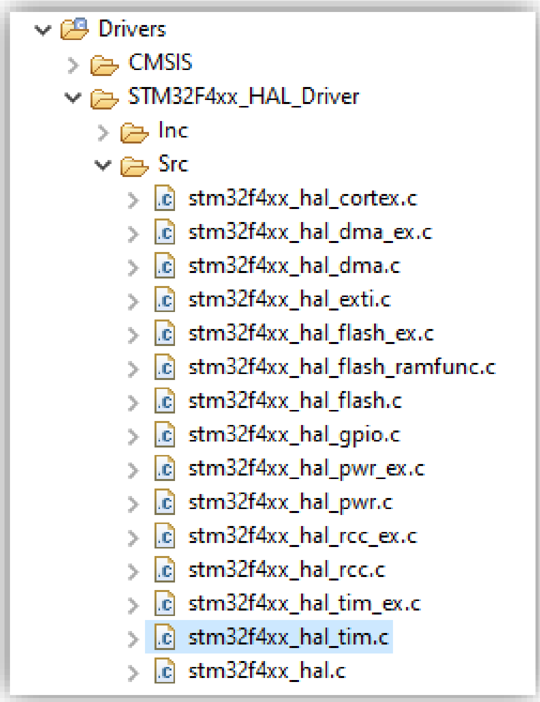


Input Capture

- To start the timer to cause an interrupt we must include the following line outside our while(1) loop:

```
HAL_TIM_IC_Start_IT(&htim5, TIM_CHANNEL_1);
```

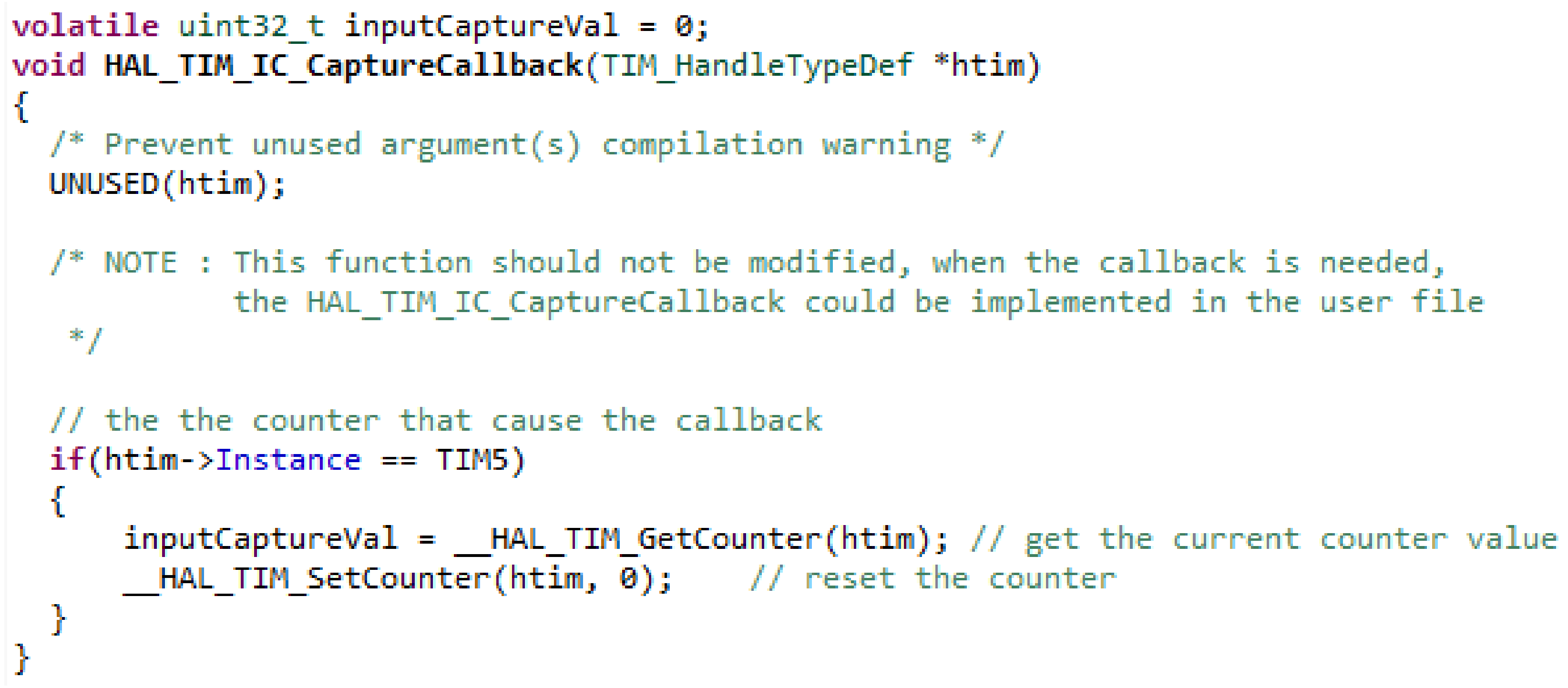
- `htim5` is auto generated by STM32CubeIDE when you set the timer up in the device configuration, matches the Timer set of that input.
- The channel should match the channel that the output pin was set up to use.
- We also need an interrupt callback function.
  - A template can be found in stm32f4xx_hal_tim.c
  - `HAL_TIM_IC_CaptureCallback()` copy and paste your own definition in main.c (without __weak)
  - Callbacks are universal for all timers, so you must check the counter that caused the callback.



```
Drivers
  CMSIS
  STM32F4xx_HAL_Driver
    Inc
    Src
      stm32f4xx_hal_cortex.c
      stm32f4xx_hal_dma_ex.c
      stm32f4xx_hal_dma.c
      stm32f4xx_hal_exti.c
      stm32f4xx_hal_flash_ex.c
      stm32f4xx_hal_flash_ramfunc.c
      stm32f4xx_hal_flash.c
      stm32f4xx_hal_gpio.c
      stm32f4xx_hal_pwr_ex.c
      stm32f4xx_hal_pwr.c
      stm32f4xx_hal_rcc_ex.c
      stm32f4xx_hal_rcc.c
      stm32f4xx_hal_tim_ex.c
      stm32f4xx_hal_tim.c
      stm32f4xx_hal.c
```

Input Capture

- Example: Read the value from the counter when an event occurs and store it in a global variable.
  - Note the `volatile` keyword



```
volatile uint32_t inputCaptureVal = 0;
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{
  /* Prevent unused argument(s) compilation warning */
  UNUSED(htim);

  /* NOTE : This function should not be modified, when the callback is needed,
            the HAL_TIM_IC_CaptureCallback could be implemented in the user file
   */

  // the the counter that cause the callback
  if(htim->Instance == TIM5)
  {
      inputCaptureVal = __HAL_TIM_GetCounter(htim); // get the current counter value
      __HAL_TIM_SetCounter(htim, 0);    // reset the counter
  }
}
```

Timers on the STM32F411VE

- There are a lot of registers associated to timers, we rarely set any of them manually (except maybe for the TIMx_CNT register).
- Use STM32CubeIDE.

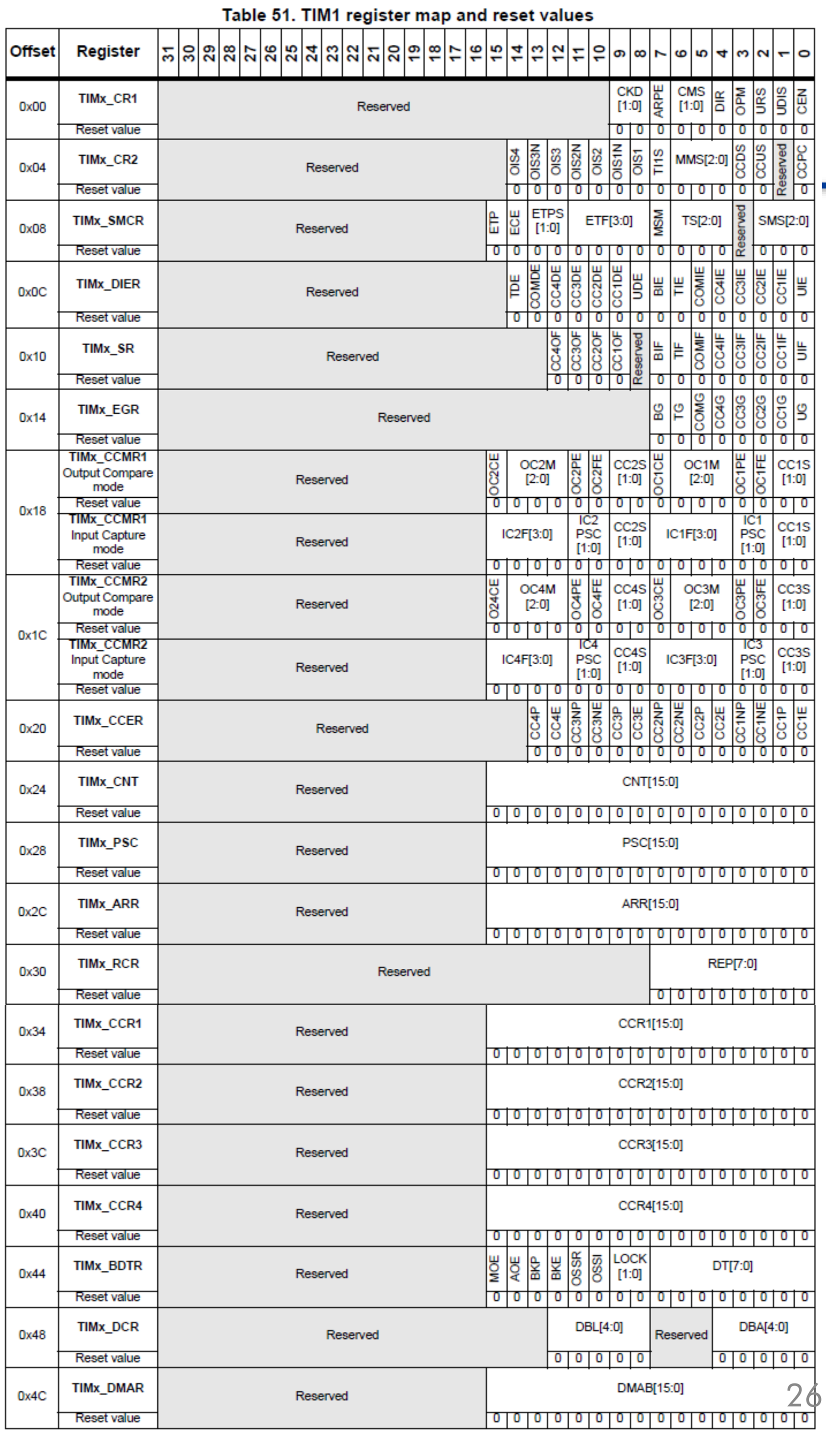
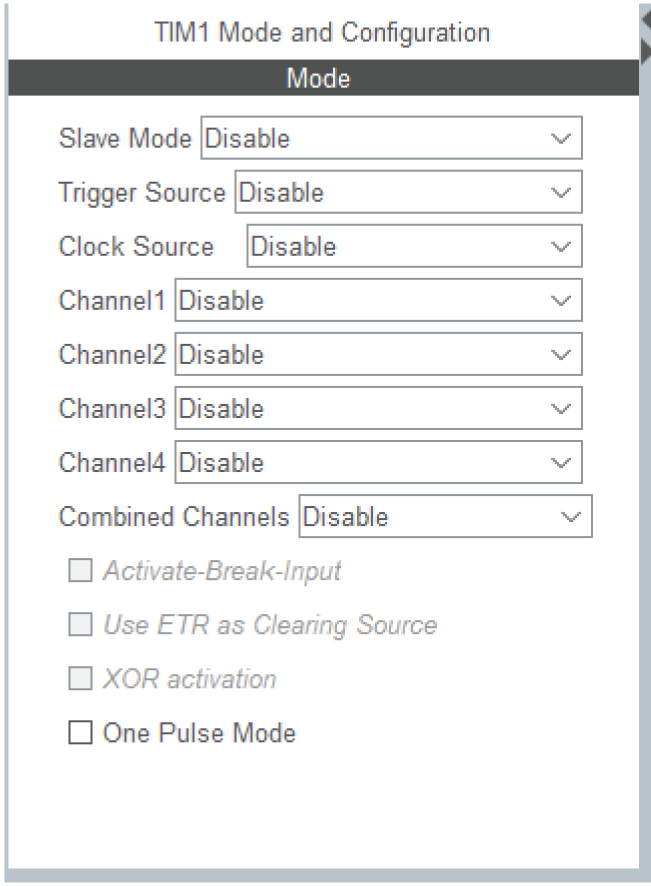


Table 51. TIM1 register map and reset values

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
0x04	TIMx_CR2	Reserved																	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Reserved	CCPC
	Reset value																		0	0	0	0	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	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	TIMx_SR	Reserved																			CC4OF	CC3OF	CC2OF	CC1OF	Reserved	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value																				0	0	0	0		0	0	0	0	0	0	0	0
0x14	TIMx_EGR	Reserved																								BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																									0	0	0	0	0	0	0	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	Reserved																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	Reserved																		CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	TIMx_CNT	Reserved																CNT[15:0]															
	Reset value																	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	Reserved																ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	TIMx_RCR	Reserved																								REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	TIMx_CCR1	Reserved																CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	TIMx_CCR2	Reserved																CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	TIMx_CCR3	Reserved																CCR3[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	TIMx_CCR4	Reserved																CCR4[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	TIMx_BDTR	Reserved																MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	TIMx_DCR	Reserved																			DBL[4:0]					Reserved			DBA[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x4C	TIMx_DMAR	Reserved																DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Timers on the STM32F411VE

- Registers controlling timer modules mapped to memory (as before)
- Lots of registers to set up. Make use of STM32CubeIDE instead
- Slave mode
  - Reset: counter register is reinitialized on trigger input
  - Gated: timer/counter is enabled depending on level of selected input
  - Triggered: Timer/counter starts in response to event on selected input
- Trigger source: Source signal for above slave modes
- Clock source: The clock source for this timer/counter (internal clock or ETR – External Trigger Input)
  - ETR linked to specific pins (e.g. PE7 for TIM1)
- Channel 1 to 4 – Capture/Compare Channel selections


TIM1 Mode and Configuration

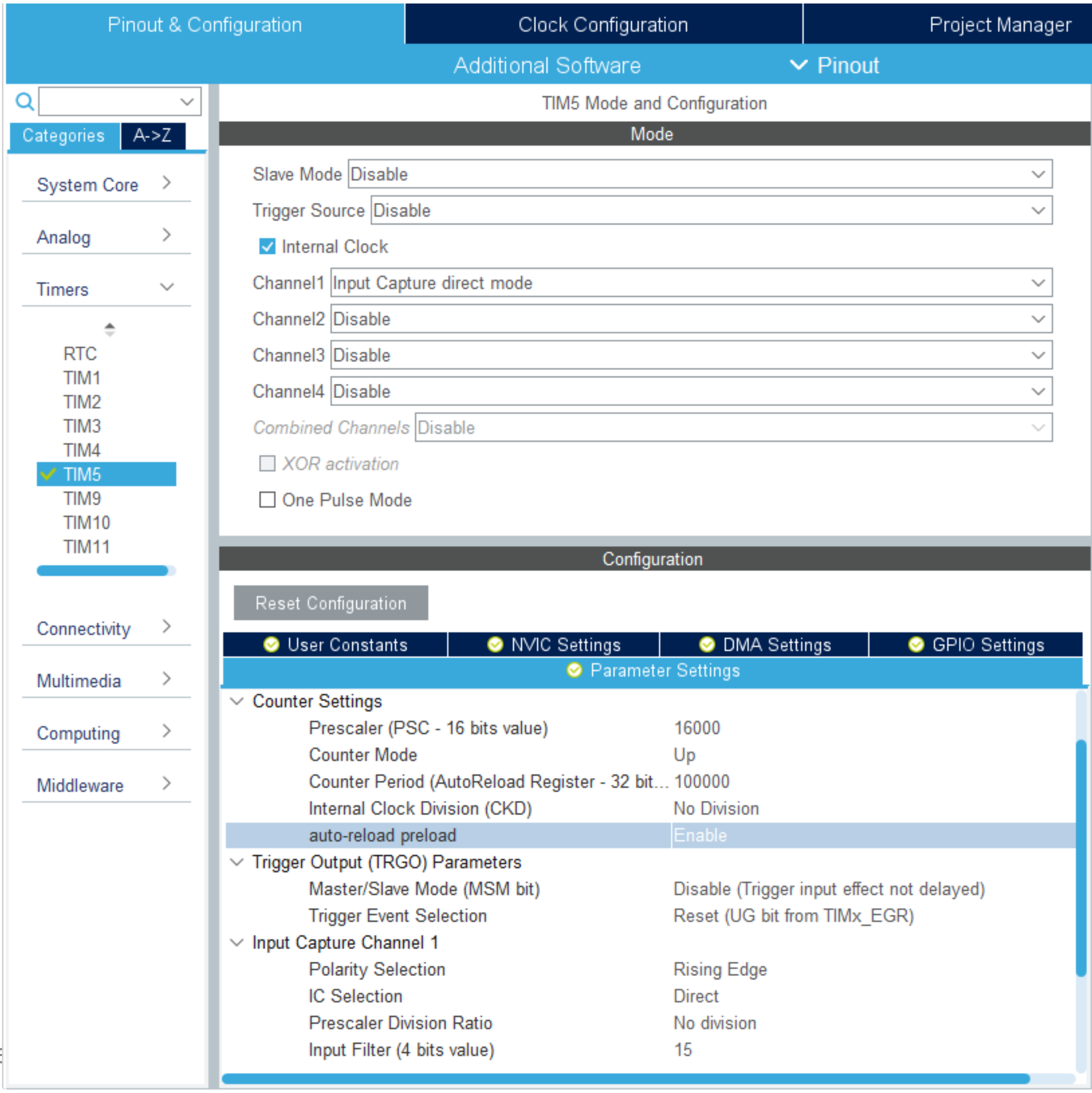
Mode

Slave Mode	Disable
Trigger Source	Disable
Clock Source	Disable
Channel1	Disable
Channel2	Disable
Channel3	Disable
Channel4	Disable
Combined Channels	Disable

☐ *Activate-Break-Input*
☐ *Use ETR as Clearing Source*
☐ *XOR activation*
☐ One Pulse Mode

Timers on the STM32F411VE

- 'Pinout' will update to show which external pins are being used
- More settings on the 'Configuration' tab
  - Here you can set timer parameters.
  - Interrupt settings (NVIC)
  - GPIO settings (pull up, etc.)



TIM5 Mode and Configuration

Mode

Slave Mode: Disable
Trigger Source: Disable
✓ Internal Clock
Channel1: Input Capture direct mode
Channel2: Disable
Channel3: Disable
Channel4: Disable
Combined Channels: Disable
☐ *XOR activation*
☐ One Pulse Mode

Configuration

Counter Settings	
Prescaler (PSC - 16 bits value)	16000
Counter Mode	Up
Counter Period (AutoReload Register - 32 bit...	100000
Internal Clock Division (CKD)	No Division
auto-reload preload	Enable
Trigger Output (TRGO) Parameters	
Master/Slave Mode (MSM bit)	Disable (Trigger input effect not delayed)
Trigger Event Selection	Reset (UG bit from TIMx_EGR)
Input Capture Channel 1	
Polarity Selection	Rising Edge
IC Selection	Direct
Prescaler Division Ratio	No division
Input Filter (4 bits value)	15