

# PRIMENA MIKROKONTROLERA- MS1PMK

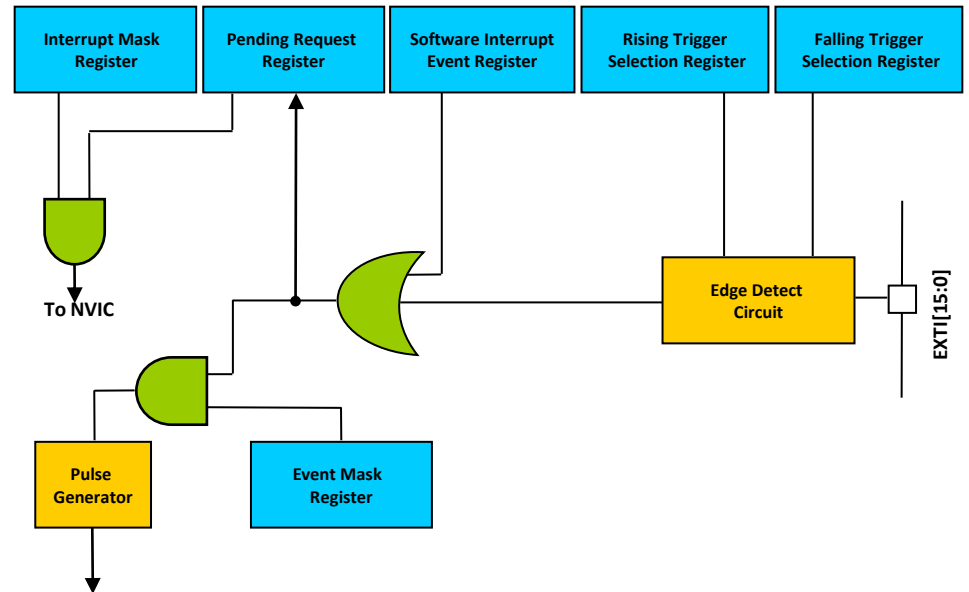
5. deo

2017

Nenad Jovičić

# EXTI Spoljašnji prekidi

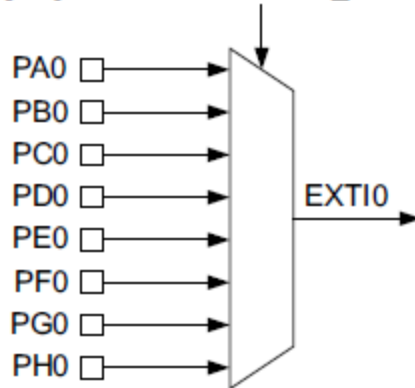
- Do 40 Interrupt/Events zahteva
  - Do 80 pinova se može koristiti kao EXTI ulaz.
  - 16 linija povezanih na GPIO pinove
  - Ostale linije poveyane na specifične periferije
- Dva konfiguraciona moda:
  - Interrupt mode: generisanje prekida
  - Event mode: generisanje buđenja sistema koji je u SLEEP modu.
- Nezavisni trigeri (rastuća, opadajuća ili obe ivice)
- Status bit svake linije
- Mogućnost softverskog forsiranja bilo kog prekida/događaja.



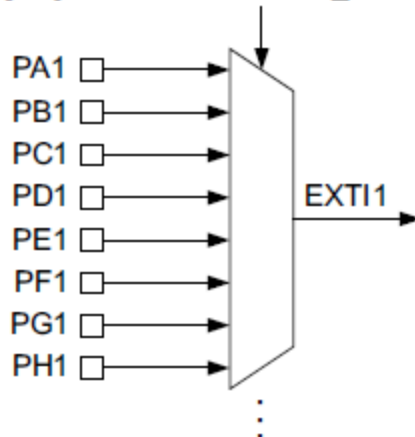
- EXTI periferija je povezana na APB2 da bi se skratilo vreme ragovanja (APB2 je u opštem slučaju brža od APB1 magistrale)

# EXTI prekidi

EXTI0[3:0] bits in the SYSCFG\_EXTICR1 register



EXTI1[3:0] bits in the SYSCFG\_EXTICR1 register



DCD	FVD_IRQHandler	; PVD through EXTI Line detect
DCD	TAMPER_IRQHandler	; Tamper
DCD	RTC_IRQHandler	; RTC
DCD	FLASH_IRQHandler	; Flash
DCD	RCC_IRQHandler	; RCC
DCD	EXTI0_IRQHandler	; EXTI Line 0
DCD	EXTI1_IRQHandler	; EXTI Line 1
DCD	EXTI2_IRQHandler	; EXTI Line 2
DCD	EXTI3_IRQHandler	; EXTI Line 3
DCD	EXTI4_IRQHandler	; EXTI Line 4
DCD	DMA1_Channel1_IRQHandler	; DMA1 Channel 1
DCD	DMA1_Channel2_IRQHandler	; DMA1 Channel 2
DCD	DMA1_Channel3_IRQHandler	; DMA1 Channel 3
DCD	DMA1_Channel4_IRQHandler	; DMA1 Channel 4
DCD	DMA1_Channel5_IRQHandler	; DMA1 Channel 5
DCD	DMA1_Channel6_IRQHandler	; DMA1 Channel 6
DCD	DMA1_Channel7_IRQHandler	; DMA1 Channel 7
DCD	ADC1_IRQHandler	; ADC1
DCD	0	; Reserved
DCD	0	; Reserved
DCD	0	; Reserved
DCD	0	; Reserved
DCD	EXTI9_5_IRQHandler	; EXTI Line 9..5
DCD	TIM1_BRK_TIM15_IRQHandler	; TIM1 Break and TIM15
DCD	TIM1_UP_TIM16_IRQHandler	; TIM1 Update and TIM16
DCD	TIM1_TRG_COM_TIM17_IRQHandler	; TIM1 Trigger and Commutation and TIM17
DCD	TIM1_CC_IRQHandler	; TIM1 Capture Compare
DCD	TIM2_IRQHandler	; TIM2
DCD	TIM3_IRQHandler	; TIM3
DCD	TIM4_IRQHandler	; TIM4
DCD	I2C1_EV_IRQHandler	; I2C1 Event
DCD	I2C1_ER_IRQHandler	; I2C1 Error
DCD	I2C2_EV_IRQHandler	; I2C2 Event
DCD	I2C2_ER_IRQHandler	; I2C2 Error
DCD	SPI1_IRQHandler	; SPI1
DCD	SPI2_IRQHandler	; SPI2
DCD	USART1_IRQHandler	; USART1
DCD	USART2_IRQHandler	; USART2
DCD	USART3_IRQHandler	; USART3
DCD	EXTI15_10_IRQHandler	; EXTI Line 15..10
DCD	RTCAlarm_IRQHandler	; RTC Alarm through EXTI Line
DCD	CEC_IRQHandler	; HDMI-CEC

# External interrupt configuration register 1 (2,3,4) (SYSCFG\_EXTICR1 (2,3,4))

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	EXTI3[2:0]			Res	EXTI2[2:0]			Res	EXTI1[2:0]			Res	EXTI0[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:12 **EXTI3[2:0]**: EXTI 3 configuration bits

These bits are written by software to select the source input for the EXTI3 external interrupt.

- 000: PA[3] pin
- 001: PB[3] pin
- 010: PC[3] pin
- 011: PD[3] pin
- 100: PE[3] pin
- 101: PF[3] pin
- 110: PG[3] pin
- 111: Reserved

**Na EXTIx ide  
uvek pinx**

# Ostale EXTI konekcije

Table 43. EXTI lines connections

EXTI line	Line source <sup>(1)</sup>	Line type
0-15	GPIO	configurable
16	PVD	configurable
17	OTG_FS wakeup event <sup>(2)</sup>	direct
18	RTC alarms	configurable
19	RTC tamper or timestamp or CSS_LSE	configurable
20	RTC wakeup timer	configurable
21	COMP1 output	configurable
22	COMP2 output	configurable
23	I2C1 wakeup <sup>(2)</sup>	direct
24	I2C2 wakeup <sup>(2)</sup>	direct
25	I2C3 wakeup	direct
26	USART1 wakeup <sup>(2)</sup>	direct
27	USART2 wakeup <sup>(2)</sup>	direct
28	USART3 wakeup <sup>(2)</sup>	direct
29	UART4 wakeup <sup>(2)</sup>	direct
30	UART5 wakeup <sup>(2)</sup>	direct
31	LPUART1 wakeup	direct
32	LPTIM1	direct
33	LPTIM2 <sup>(2)</sup>	direct
34	SWPMI1 wakeup <sup>(2)</sup>	direct
35	PVM1 wakeup	configurable
36	PVM2 wakeup	configurable
37	PVM3 wakeup	configurable
38	PVM4 wakeup	configurable
39	LCD wakeup	direct

# Interrupt mask register (EXTI\_IMR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IM31	IM30	IM29	IM28	IM27	IM26	IM25	IM24	IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IMx**: Interrupt Mask on line x (x = 31 to 0)

0: Interrupt request from Line x is masked

1: Interrupt request from Line x is not masked

# Interrupt mask register 2 (EXTI\_IMR2)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IM39	IM38	IM37	IM36	IM35	IM34	IM33	IM32
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value

Bits 7:0 **IMx**: Interrupt mask on line x (x = 39 to 32)

0: Interrupt request from line x is masked

1: Interrupt request from line x is not masked

# Rising trigger selection register (EXTI\_RTSR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT22	RT21	RT20	RT19	RT18	Res.	RT16
									rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:18 **RTx**: Rising trigger event configuration bit of line x (x = 22 to 18)

0: Rising trigger disabled (for Event and Interrupt) for input line

1: Rising trigger enabled (for Event and Interrupt) for input line

Bit 17 Reserved, must be kept at reset value.

Bits 16:0 **RTx**: Rising trigger event configuration bit of line x (x = 16 to 0)

0: Rising trigger disabled (for Event and Interrupt) for input line

1: Rising trigger enabled (for Event and Interrupt) for input line



# Wakeup event mask register (EXTI\_EMR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EM31	EM30	EM29	EM28	EM27	EM26	EM25	EM24	EM23	EM22	EM21	EM20	EM19	EM18	EM17	EM16
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:0 **EMx**: Event mask on line x (x = 31 to 0)

0: Event request from line x is masked

1: Event request from line x is not masked

# Falling trigger selection register (EXTI\_FTSR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT22	FT21	FT20	FT19	FT18	Res.	FT16
									rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:18 **FTx**: Falling trigger event configuration bit of line x (x = 22 to 18)

0: Falling trigger disabled (for Event and Interrupt) for input line

1: Falling trigger enabled (for Event and Interrupt) for input line

Bit 17 Reserved, must be kept at reset value.

Bits 16:0 **FTx**: Falling trigger event configuration bit of line x (x = 16 to 0)

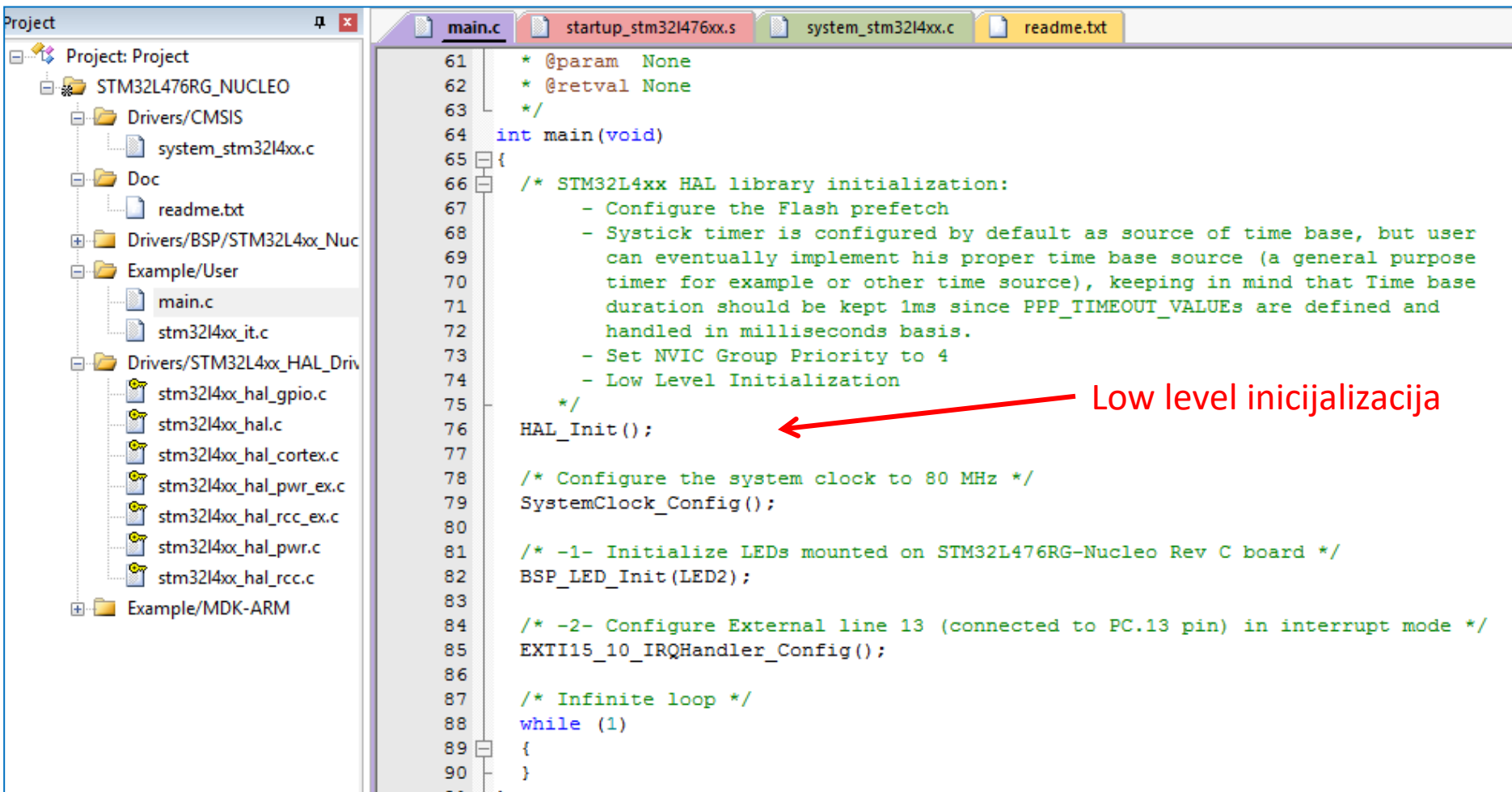
0: Falling trigger disabled (for Event and Interrupt) for input line

1: Falling trigger enabled (for Event and Interrupt) for input line

# STM CUBE

## Projekat GPIO\_EXTI

- ...\\STM32Cube\_FW\_L4\_V1.4.0\\Projects\\STM32L476RG-Nucleo\\Examples\\GPIO\\GPIO\_EXTI\\MDK-ARM



```
61  * @param None
62  * @retval None
63  */
64  int main(void)
65  {
66      /* STM32L4xx HAL library initialization:
67       - Configure the Flash prefetch
68       - SysTick timer is configured by default as source of time base, but user
69       can eventually implement his proper time base source (a general purpose
70       timer for example or other time source), keeping in mind that Time base
71       duration should be kept 1ms since PPP_TIMEOUT_VALUES are defined and
72       handled in milliseconds basis.
73       - Set NVIC Group Priority to 4
74       - Low Level Initialization
75     */
76     HAL_Init();
77
78     /* Configure the system clock to 80 MHz */
79     SystemClock_Config();
80
81     /* -1- Initialize LEDs mounted on STM32L476RG-Nucleo Rev C board */
82     BSP_LED_Init(LED2);
83
84     /* -2- Configure External line 13 (connected to PC.13 pin) in interrupt mode */
85     EXTI15_10_IRQHandler_Config();
86
87     /* Infinite loop */
88     while (1)
89     {
90     }
```

Low level inicijalizacija

# Inicijalizacija EXTI prekida

```
Project: Project
├── STM32L476RG_NUCLEO
│   ├── Drivers\CMSIS
│   │   └── system_stm32l4xx.c
│   ├── Doc
│   │   └── readme.txt
│   ├── Drivers\BSP\STM32L4xx_Nuc
│   ├── Example\User
│   │   ├── main.c
│   │   ├── stm32l4xx_it.c
│   │   └── Drivers\STM32L4xx_HAL_Drv
│   │       ├── stm32l4xx_hal_gpio.c
│   │       ├── stm32l4xx_hal.c
│   │       ├── stm32l4xx_hal_cortex.c
│   │       ├── stm32l4xx_hal_pwr_ex.c
│   │       └── stm32l4xx_hal_rcc_ex.c
└── main.c

151 * @param None
152 * @retval None
153 */
154 static void EXTI15_10_IRQHandler(void)
155 {
156     GPIO_InitTypeDef GPIO_InitStruct = {0};
157
158     /* Enable GPIOC clock */
159     __HAL_RCC_GPIOC_CLK_ENABLE();
160
161     /* Configure PC.13 pin as input floating */
162     GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
163     GPIO_InitStruct.Pull = GPIO_NOPULL;
164     GPIO_InitStruct.Pin = GPIO_PIN_13;
165     HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
166
167     /* Enable and set EXTI lines 10 to 15 Interrupt to the lowest priority */
168     HAL_NVIC_SetPriority(EXTI15_10_IRQn, 2, 0);
169     HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);
170 }

stm32l4xx_hal_gpio.h

/*
 * Define GPIO modes
 */
#define GPIO_MODE_INPUT ((uint32_t)0x00000000) /*< Input Floating Mode */
#define GPIO_MODE_OUTPUT_PP ((uint32_t)0x00000001) /*< Output Push Pull Mode */
#define GPIO_MODE_OUTPUT_OD ((uint32_t)0x00000011) /*< Output Open Drain Mode */
#define GPIO_MODE_AF_PP ((uint32_t)0x00000002) /*< Alternate Function Push Pull Mode */
#define GPIO_MODE_AF_OD ((uint32_t)0x00000012) /*< Alternate Function Open Drain Mode */
#define GPIO_MODE_ANALOG ((uint32_t)0x00000003) /*< Analog Mode */
#define GPIO_MODE_ANALOG_ADC_CONTROL ((uint32_t)0x0000000B) /*< Analog Mode for ADC conversion */
#define GPIO_MODE_IT_RISING ((uint32_t)0x10110000) /*< External Interrupt Mode with Rising edge trigger detection */
#define GPIO_MODE_IT_FALLING ((uint32_t)0x10210000) /*< External Interrupt Mode with Falling edge trigger detection */
#define GPIO_MODE_IT_RISING_FALLING ((uint32_t)0x10310000) /*< External Interrupt Mode with Rising/Falling edge trigger detection */
#define GPIO_MODE_EVT_RISING ((uint32_t)0x10120000) /*< External Event Mode with Rising edge trigger detection */
#define GPIO_MODE_EVT_FALLING ((uint32_t)0x10220000) /*< External Event Mode with Falling edge trigger detection */
#define GPIO_MODE_EVT_RISING_FALLING ((uint32_t)0x10320000) /*< External Event Mode with Rising/Falling edge trigger detection */
```

# HAL\_GPIO\_Init()

```

it.c  main.c  stm32l4xx_hal_gpio.h  stm32l4xx_hal_gpio.c

/*----- EXTI Mode Configuration -----*/
/* Configure the External Interrupt or event for the current IO */
if((GPIO_Init->Mode & EXTI_MODE) == EXTI_MODE)
{
    /* Enable SYSCFG Clock */
    __HAL_RCC_SYSCFG_CLK_ENABLE();

    temp = SYSCFG->EXTICR[position >> 2];
    temp &= ~((uint32_t)0x0F) << (4 * (position & 0x03));
    temp |= (GPIO_GET_INDEX(GPIOx) << (4 * (position & 0x03)));
    SYSCFG->EXTICR[position >> 2] = temp;

    /* Clear EXTI line configuration */
    temp = EXTI->IMR1;
    temp &= ~((uint32_t)iocurrent);
    if((GPIO_Init->Mode & GPIO_MODE_IT) == GPIO_MODE_IT)

```

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	EXTI3[2:0]			Res	EXTI2[2:0]			Res	EXTI1[2:0]			Res	EXTI0[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:12 **EXTI3[2:0]**: EXTI 3 configuration bits

These bits are written by software to select the source input for the EXTI3 external interrupt.

- 000: PA[3] pin
- 001: PB[3] pin
- 010: PC[3] pin
- 011: PD[3] pin
- 100: PE[3] pin
- 101: PF[3] pin
- 110: PG[3] pin
- 111: Reserved

```

iocurrent);
& GPIO_MODE_EVT) == GPIO_MODE_EVT)

```

```

ing edge configuration */

```

```

iocurrent);
& RISING_EDGE) == RISING_EDGE)

```

```

EXTI->RTSR1 = temp;

```

# Reakcija na prekid!

```
startup_stm321476xx.s  stm3214xx_it.c  main.c  stm3214xx_hal_gpio.h  stm3214xx_hal_gpio.c
273 EXPORT I2C1_ER_IRQHandler [WEAK]
274 EXPORT I2C2_EV_IRQHandler [WEAK]
275 EXPORT I2C2_ER_IRQHandler [WEAK]
276 EXPORT SPI1_IRQHandler [WEAK]
277 EXPORT SPI2_IRQHandler [WEAK]
278 EXPORT USART1_IRQHandler [WEAK]
279 EXPORT USART2_IRQHandler [WEAK]
280 EXPORT USART3_IRQHandler [WEAK]
281 EXPORT EXTI15_10_IRQHandler [WEAK]
282 EXPORT RTC_Alarm_IRQHandler [WEAK]
283 EXPORT DFSDM3_IRQHandler [WEAK]
284 EXPORT TIM8_BRK_IRQHandler [WEAK]
```

```
startup_stm321476xx.s  stm3214xx_it.c  main.c  stm3214xx_hal_gpio.h  stm3214xx_hal_gpio.c
167 /*****
168
169 /**
170  * @brief This function handles external lines 10
171  * @param None
172  * @retval None
173  */
174 void EXTI15_10_IRQHandler(void)
175 {
176     HAL_GPIO_EXTI_IRQHandler(USER_BUTTON_PIN);
177 }
178
```

Korisničke funkcije

```
startup_stm321476xx.s  stm3214xx_it.c  main.c  stm3214xx_hal_gpio.h  stm3214xx_hal_gpio.c
/**
 * @brief EXTI line detection callbacks
 * @param GPIO_Pin: Specifies the pins connected
 * @retval None
 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if (GPIO_Pin == GPIO_PIN_13)
    {
        /* Toggle LED2 */
        BSP_LED_Toggle(LED2);
    }
}
```

```
startup_stm321476xx.s  stm3214xx_it.c  main.c  stm3214xx_hal_gpio.h  stm3214xx_hal_gpio.c
 * @brief Handle EXTI interrupt request.
 * @param GPIO_Pin: Specifies the port pin connected to corresponding
 * @retval None
 */
void HAL_GPIO_EXTI_IRQHandler(uint16_t GPIO_Pin)
{
    /* EXTI line interrupt detected */
    if( __HAL_GPIO_EXTI_GET_IT(GPIO_Pin) != RESET)
    {
        __HAL_GPIO_EXTI_CLEAR_IT(GPIO_Pin);
        HAL_GPIO_EXTI_Callback(GPIO_Pin);
    }
}
/**
 * @brief EXTI line detection callback.
 * @param GPIO_Pin: Specifies the port pin connected to corresponding
 * @retval None
 */
weak void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    /* Prevent unused argument(s) compilation warning */
    UNUSED(GPIO_Pin);

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

Drajverske funkcije – korisnik ne menja

weak

# Reakcija na prekid!

```
startup_stm32i476xx.s  stm32i4xx_it.c  main.c  stm32i4xx_hal_gpio.h  stm32i4xx_hal_gpio.c
```

273	EXPORT	I2C1_ER_IRQHandler	[WEAK]
274	EXPORT	I2C2_EV_IRQHandler	[WEAK]
275	EXPORT	I2C2_ER_IRQHandler	[WEAK]
276	EXPORT	SPI1_IRQHandler	[WEAK]
277	EXPORT	SPI2_IRQHandler	[WEAK]
278	EXPORT	USART1_IRQHandler	[WEAK]
279	EXPORT	USART2_IRQHandler	[WEAK]
280	EXPORT	USART3_IRQHandler	[WEAK]
281	EXPORT	EXTI15_10_IRQHandler	[WEAK]
282	EXPORT	EXTI9_5_IRQHandler	[WEAK]
283	EXPORT	EXTI4_3_IRQHandler	[WEAK]
284	EXPORT	EXTI0_IRQHandler	[WEAK]

```
startup_stm32i476xx.s
```

```
167  /*****
168
169  /**
170   * @brief
171   * @param
172   * @retval
173   */
174  void EXTI
175  {
176   HAL_G
177  }
178
```

Principi HAL drajvera kada su u pitanju prekidi su sledeći:

1. Uvek postoji default handler u startup fajlu
2. Korisnik sam piše svoju prekidnu funkciju i u njoj poziva HAL\_PPP\_IRQHandler() funkciju u kojoj se “servisira” prekid
3. Ta funkcija dalje poziva HAL\_PPP\_Callback() funkciju koja ustvari “reaguje” na prekid
4. Korisnik sam implementira tu Callback funkciju.

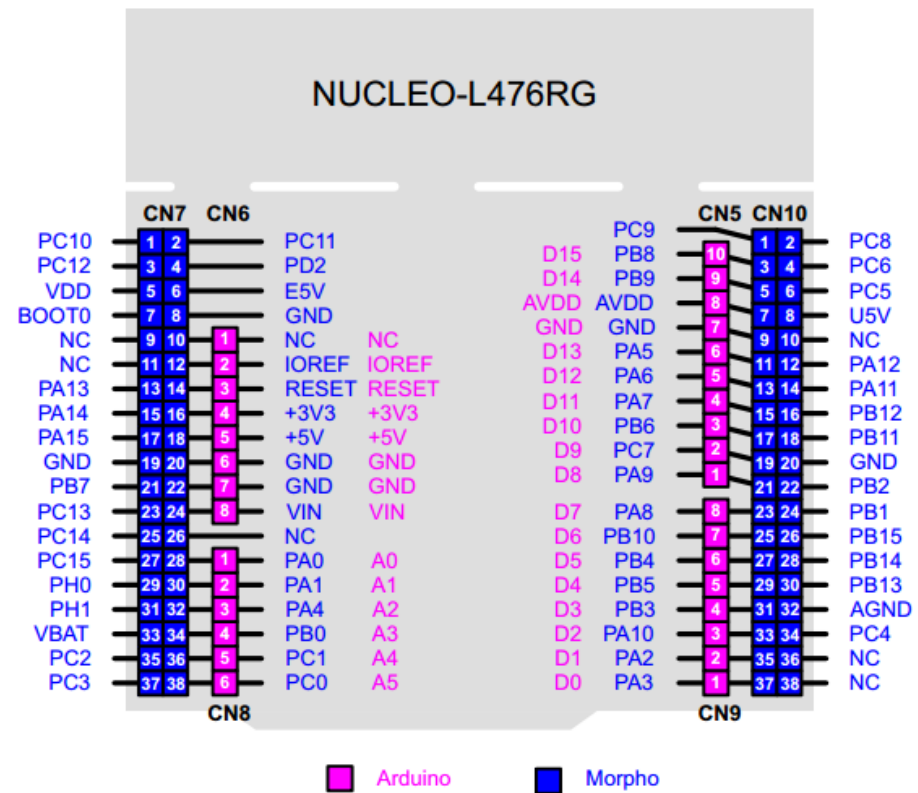
```
startup_stm32i476xx.s
```

```
/**
 * @brief EX
 * @param GE
 * @retval M
 */
void HAL_GPIO
{
  if (GPIO_P
  {
    /* Toggle
    BSP_LED_T
  }
}
```

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

# ZADATAK 1

- Napisati program koji obezbeđuje promenu stanja diode LED2 na prekid na pinu PA\_8.
- Problem 1 – PA\_8 je floating
- Rešenje – aktivirati pull-up
- Problem 2 – PA\_8 ima drugi prekidni vektor
- Rešenje – Identifikovati koji je to drugi prekidni vektor i na osnovu toga izvršiti adekvatna prilagođenja.





# MBED

## Prekidi digitalnih portova

- Klasa `InterruptIn` implementira na jednostavan način odavno prisutnu funkcionalnost digitalnih ulaza mikrokontrolera

### InterruptIn Class Reference

```
#include <InterruptIn.h>
```

#### Public Member Functions

`InterruptIn` (PinName pin)

Create an **InterruptIn** connected to the specified pin.

void `rise` (void(\*fptr)(void))

Attach a function to call when a rising edge occurs on the input.

template<typename T >

void `rise` (T \*tptr, void(T::\*mptr)(void))

Attach a member function to call when a rising edge occurs on the input.

void `fall` (void(\*fptr)(void))

Attach a function to call when a falling edge occurs on the input.

template<typename T >

void `fall` (T \*tptr, void(T::\*mptr)(void))

Attach a member function to call when a falling edge occurs on the input.

void `mode` (PinMode pull)

Set the input pin mode.

void `enable_irq` ()

Enable IRQ.

void `disable_irq` ()

Disable IRQ.

# Klasa InterruptIn

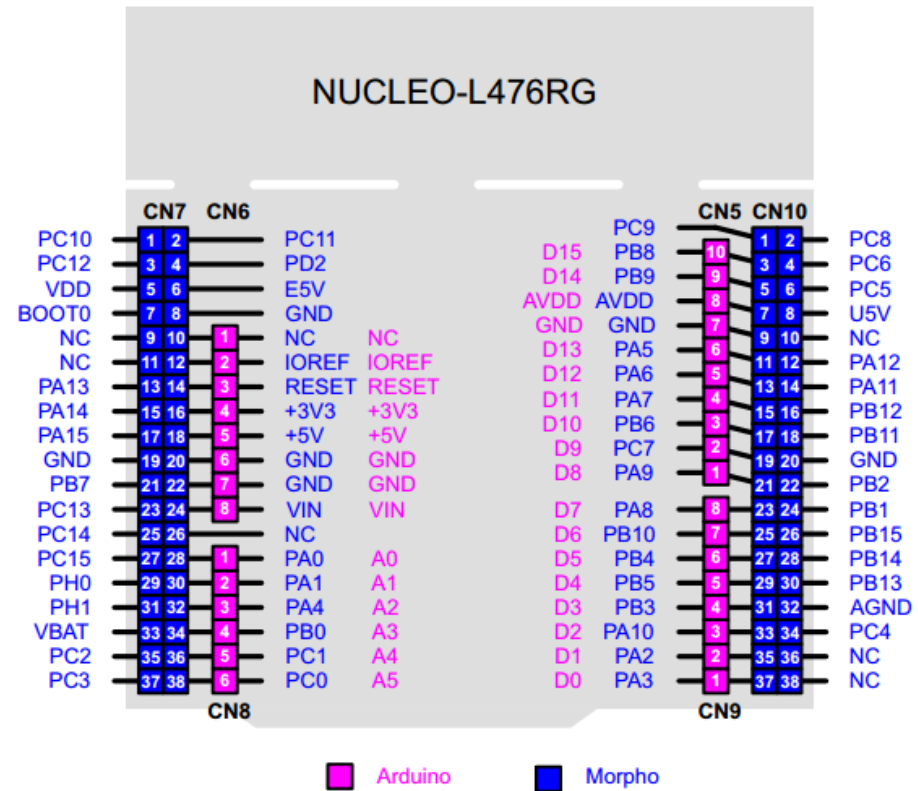
```
#include "mbed.h"
InterruptIn button(USER_BUTTON);
DigitalOut led(LED2);

void flip() {
    led = !led;
}

int main() {
    button.fall(&flip); // attach the address of the flip function to the rising
    edge
    while(1);
}
```

# ZADATAK 2

- Napisati program koji obezbeđuje promenu stanja diode LED2 na prekid na pinu PA\_8.
- Problem – PA\_8 je floating
- Rešenje – aktivirati pull-up



# STM32Fxx - Tajmeri

- STM32 arhitektura poseduje nekoliko vrsta tajmera:
  - Tajmeri opšte namene koji se koriste iz generisanje običnih PWM signala (output compare), pojedinačnih impulsa (one-pulse), hvatanje ulaznih signala (input capture), specifični senzorski interfejsi (enkoder, hall-effect senzor)
  - Napredni tajmeri (advanced timers) koji osim opštih funkcija imaju neke prednosti za generisanje signala koji se koriste u motornim pogonima ili digitalnom upravljanju pretvaračima. Primer: komplementarni izlazi sa regulisanjem mrtvog vremena, automatski isključivanje svih kanala i slično.
  - N-kanalni tajmeri (N-channel timer), koji imaju karakteristike tajmera opšte namene ali imaju ograničen broj kanala.
  - N-kanalni tajmeri sa komplementarnim izlazima, i sa regulacijom mrtvog vremena samo na jednom kanalu.
  - Osnovni tajmer (basic timer), koji nema izlaze i ulaze već se koristi za generisanje vremenske baze, ili periodično trigerovanje DAC periferije.

# Tajmeri – pregled po STM32 familiji

Timer type		STM32F0 series	STM32F101 /102/103/ 105/107 lines	STM32F100 value line	STM32L1 series	STM32F2 and STM32F4 series	STM32F30x and STM32F3x8	STM32F37x lines
Advanced		TIM1	TIM1	TIM1	-	TIM1	TIM1	-
		-	TIM8	-	-	TIM8	TIM8	-
		-	-	-	-	-	TIM20 <sup>(1)</sup>	-
General purpose	16-bit	-	TIM2	TIM2	TIM2	-	TIM2	TIM2
		TIM3	TIM3	TIM3	TIM3	TIM3	TIM3	TIM3
		-	TIM4	TIM4	TIM4	TIM4	TIM4	TIM4
		-	TIM5	TIM5	-	-	-	TIM5
	32-bit	-	-	-	-	-	-	TIM19
		TIM2	-	-	-	TIM2	TIM2	TIM2
		-	-	-	-	TIM5	-	TIM5
Basic		TIM6	TIM6	TIM6	TIM6	TIM6	TIM6	TIM6
		-	TIM7	TIM7	TIM7	TIM7	TIM7	TIM7
		-	-	-	-	-	-	TIM18
1-channel		-	TIM10	-	TIM10	TIM10	-	-
		-	TIM11	-	TIM11	TIM11	-	-
		-	TIM13	TIM13	-	TIM13	-	TIM13
		TIM14	TIM14	TIM14	-	TIM14	-	TIM14
2-channel		-	TIM9	-	TIM9	TIM9	-	-
		-	TIM12	TIM12	-	TIM12	-	TIM12
1-channel with one complementary output		TIM15	-	TIM15	-	-	TIM15	TIM15
		-	-	-	-	-	-	-
2-channel with one complementary output		TIM16	-	TIM16	-	-	TIM16	TIM16
		TIM17	-	TIM17	-	-	TIM17	TIM17

# STM32L476 tajmeri

Timer type	Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/compare channels	Complementary outputs
Advanced control	TIM1, TIM8	16-bit	Up, down, Up/down	Any integer between 1 and 65536	Yes	4	3
General-purpose	TIM2, TIM5	32-bit	Up, down, Up/down	Any integer between 1 and 65536	Yes	4	No
General-purpose	TIM3, TIM4	16-bit	Up, down, Up/down	Any integer between 1 and 65536	Yes	4	No
General-purpose	TIM15	16-bit	Up	Any integer between 1 and 65536	Yes	2	1
General-purpose	TIM16, TIM17	16-bit	Up	Any integer between 1 and 65536	Yes	1	1
Basic	TIM6, TIM7	16-bit	Up	Any integer between 1 and 65536	Yes	0	No

# Tajmeri opšte namene TIMx (x=2,3,4,5)

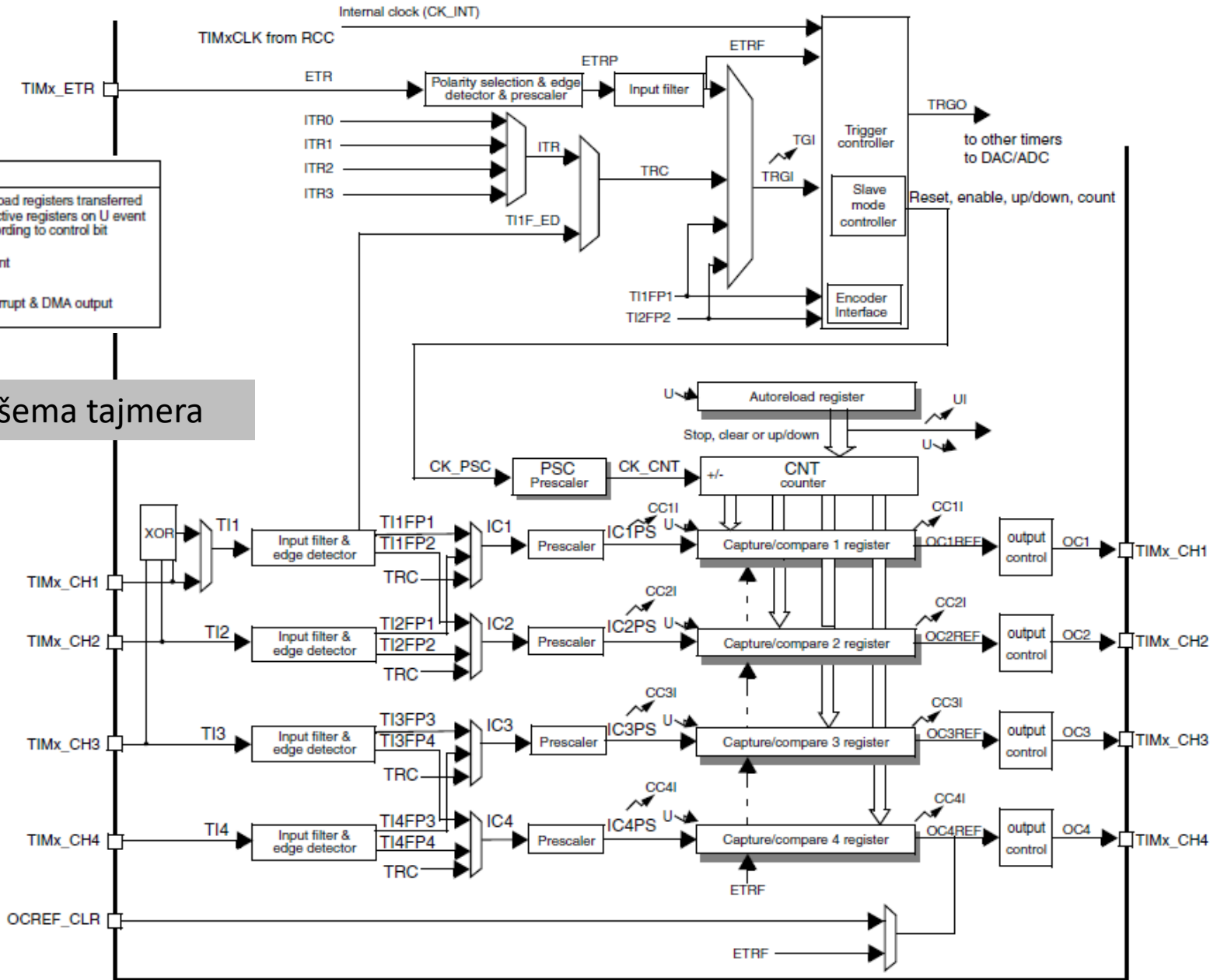
- 16-bitni (TIM3 i TIM4) ili 32-bitni brojač (TIM2 i TIM5) na gore, dole ili gore/dole.
- 16-bitni preskaler za ulazni takt
- Do 4 nezavisna kanala koji mogu da rade u izlaznom (output compare), ulaznom (input capture), PWM ili pojedinačnom impulsnom modu.
- Mogućnost sinhronizacije sa ostalim tajmerima.
- Prekid/DMA zahtev za sledeće događaje:
  - Input capture
  - Output compare
  - Reload tajmera, inicijalizacija (softverska ili spoljašnja)
- Podržavaju kvadraturne inkrementalne enkodere i hall-effect senzore.

# Tajmer opšte namene

**Notes:**

- Reg** Preload registers transferred to active registers on U event according to control bit
- event
- interrupt & DMA output

## Opšta šema tajmera





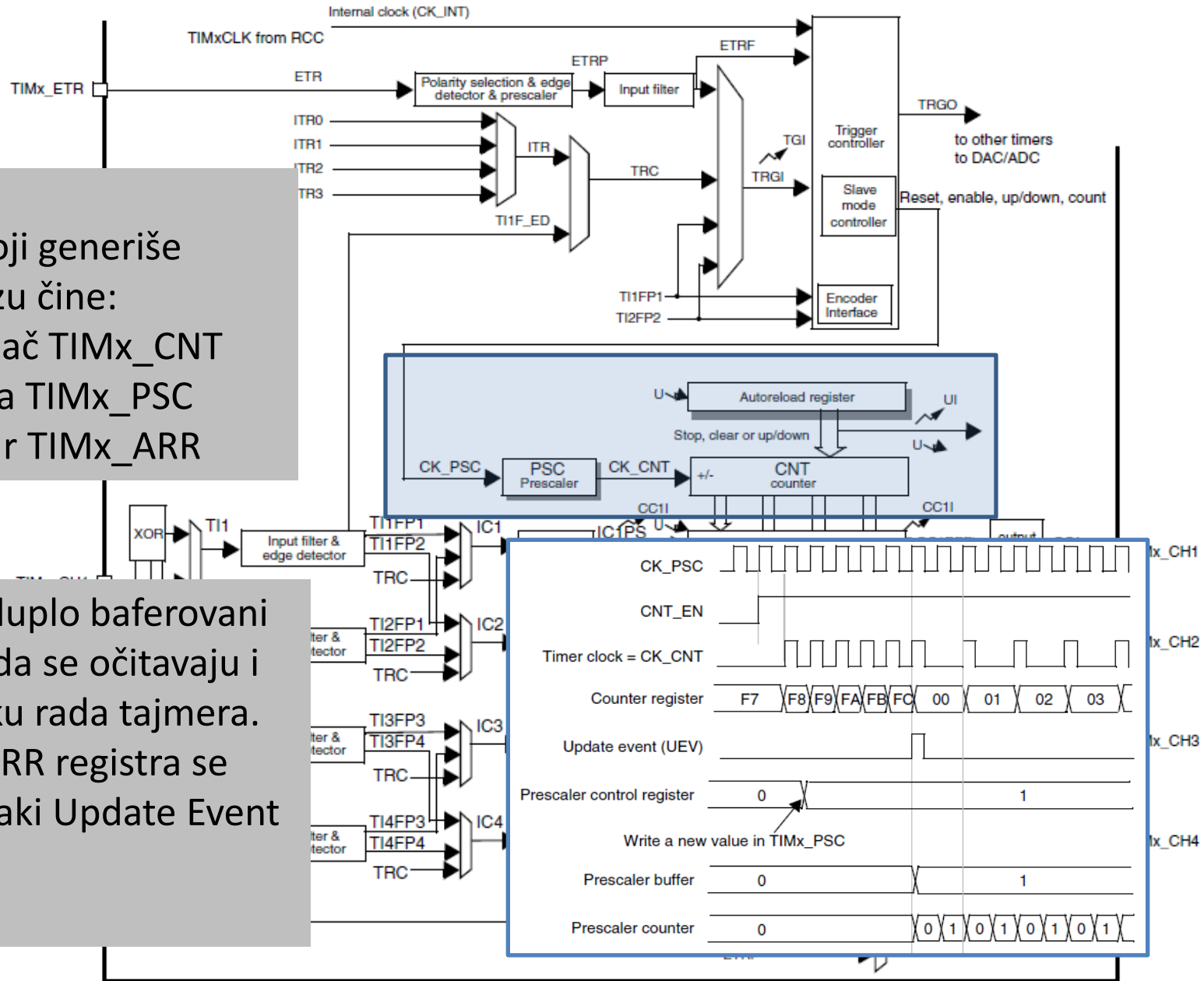
# Vremenska baza

Vremenska baza

Deo tajmera koji generiše vremensku bazu čine:

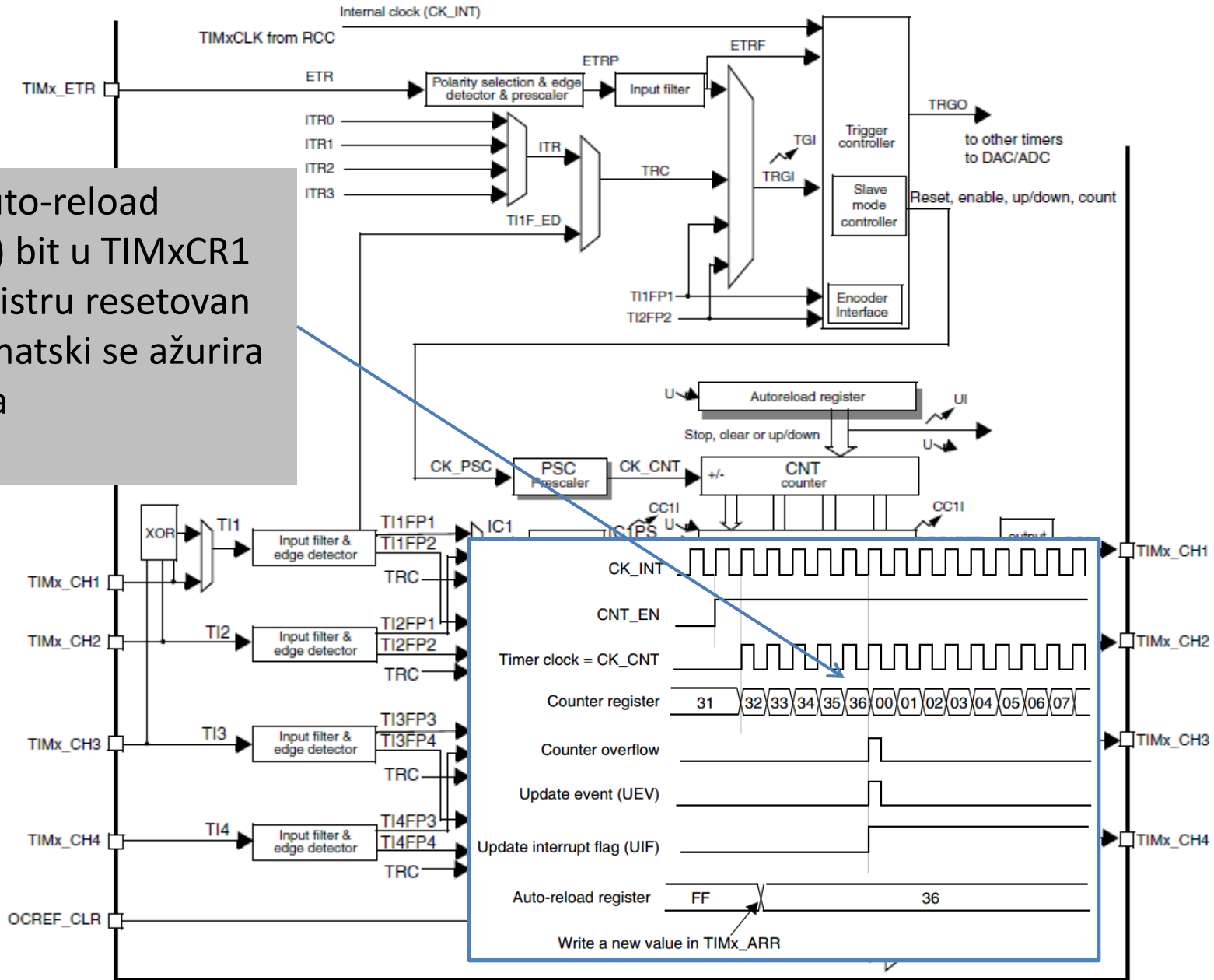
- Tajmerski brojač TIMx\_CNT
- Preskaler takta TIMx\_PSC
- Reload registar TIMx\_ARR

Svi registri su duplo baferovani tako da mogu da se očitavaju i menjaju i u toku rada tajmera. Izmene PSC i ARR registra se dešavaju na svaki Update Event UEV



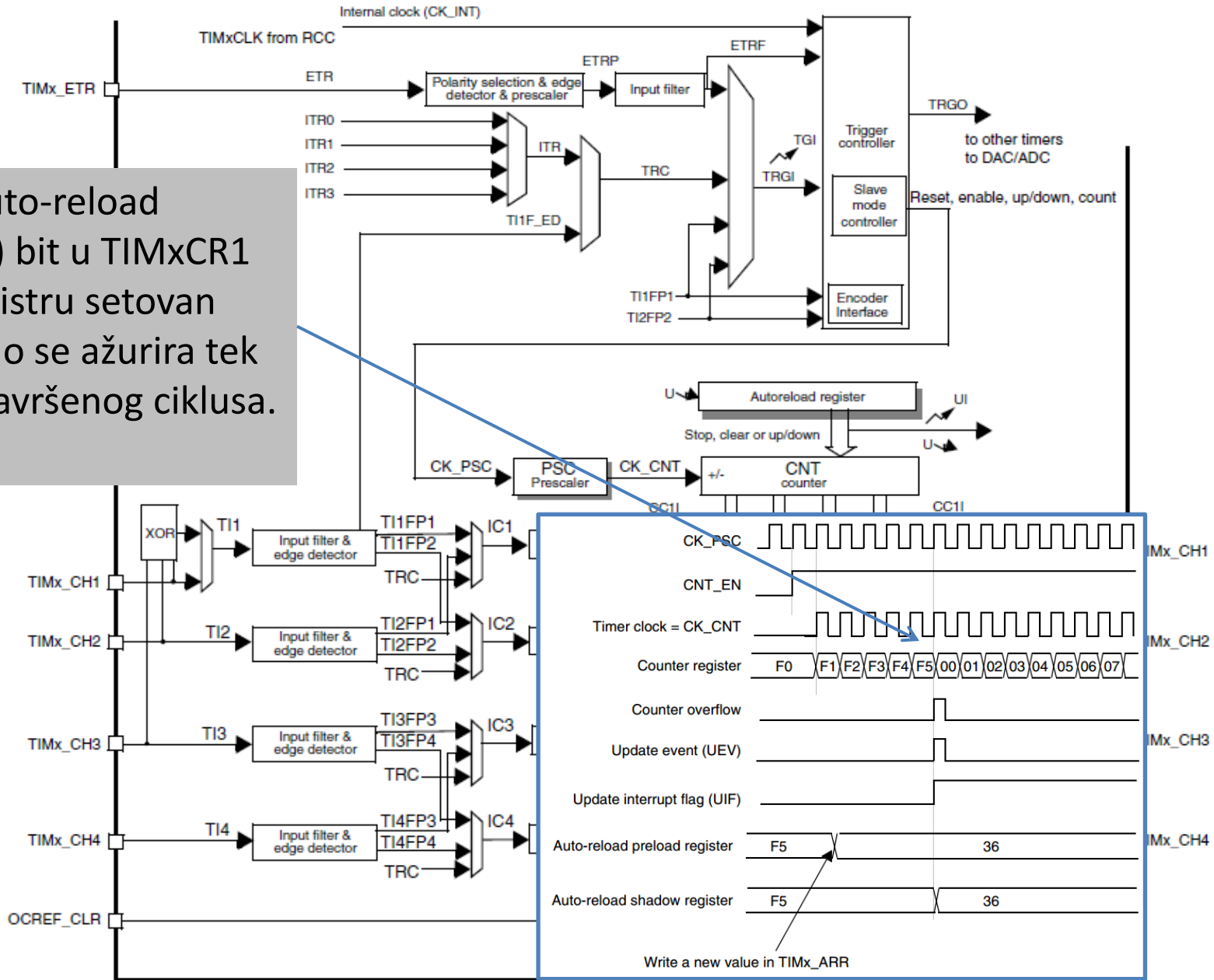
# Brojač na gore – UP counting

Ako je ARPE (auto-reload preload enable) bit u TIMxCR1 kontrolnom registru resetovan (ARPE=0) automatski se ažurira moduo brojanja

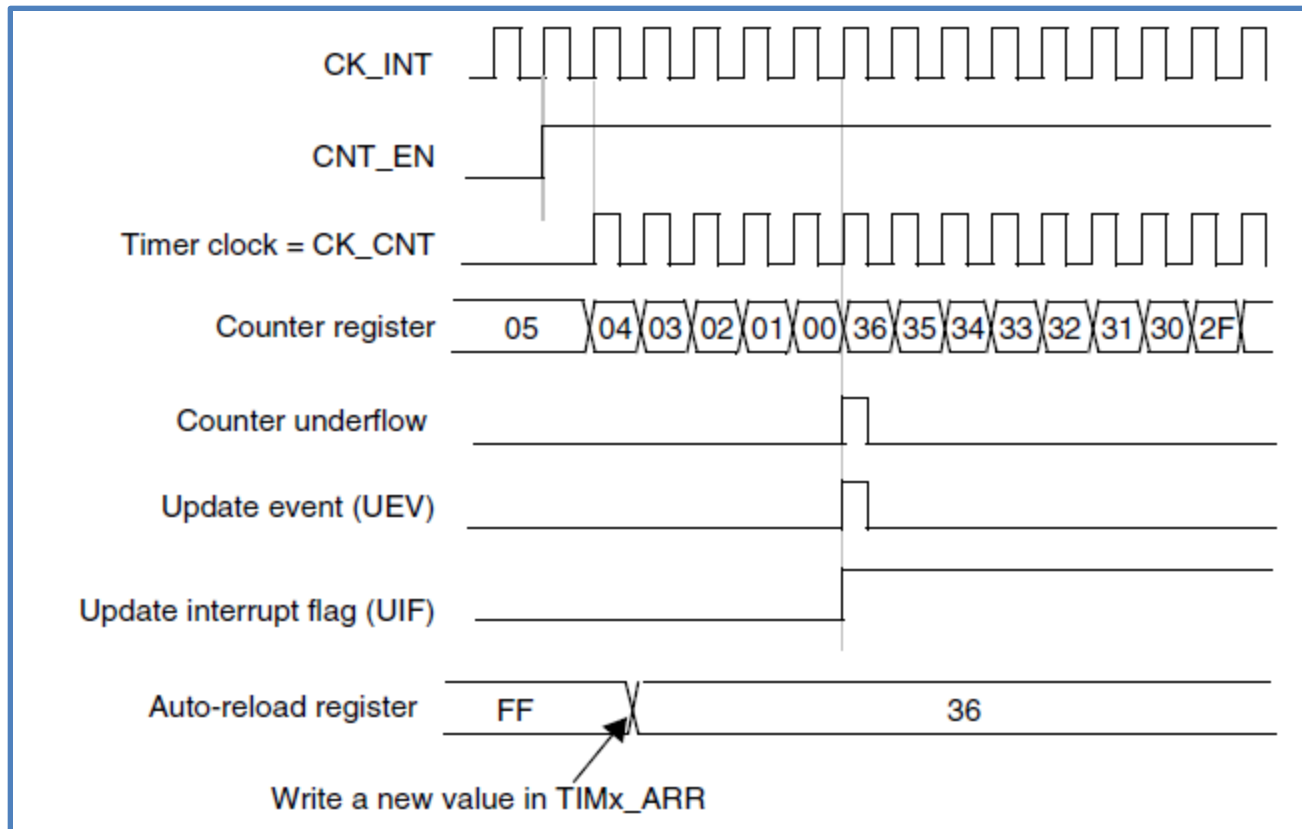


# Brojač na gore – UP counting

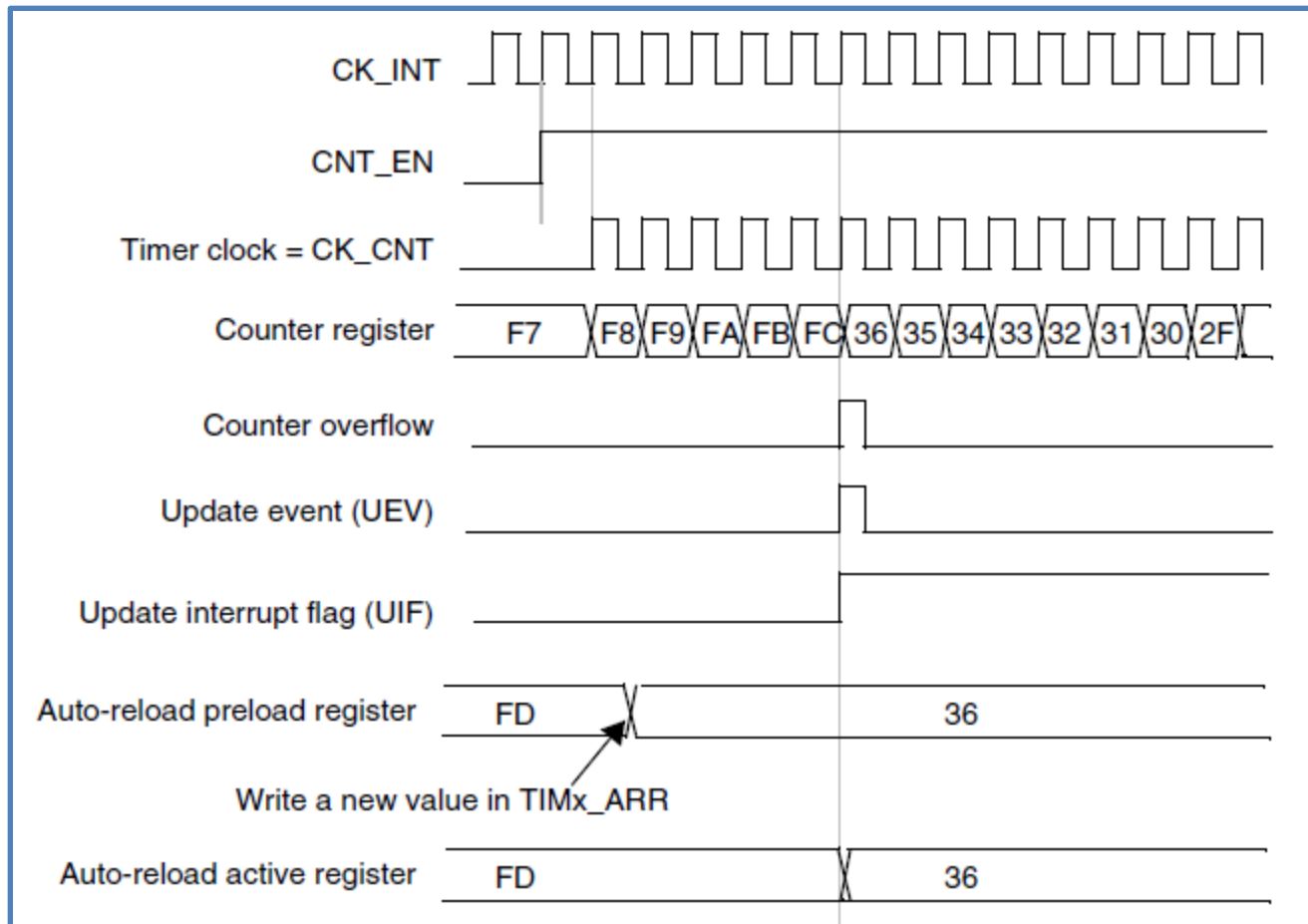
Ako je ARPE (auto-reload preload enable) bit u TIMxCR1 kontrolnom registru setovan (ARPE=1) moduo se ažurira tek nakon jednog završenog ciklusa.



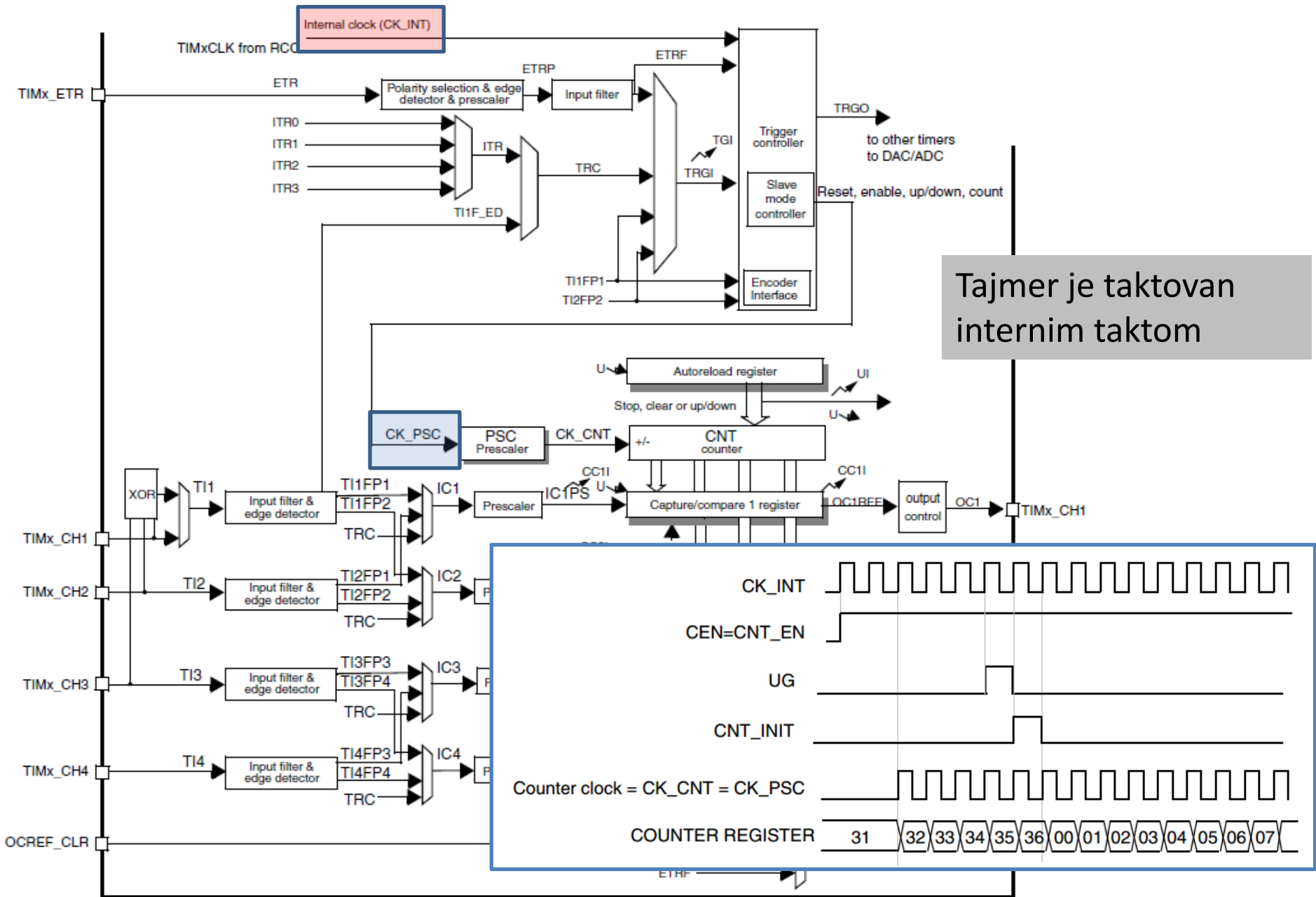
# Downcounting mode



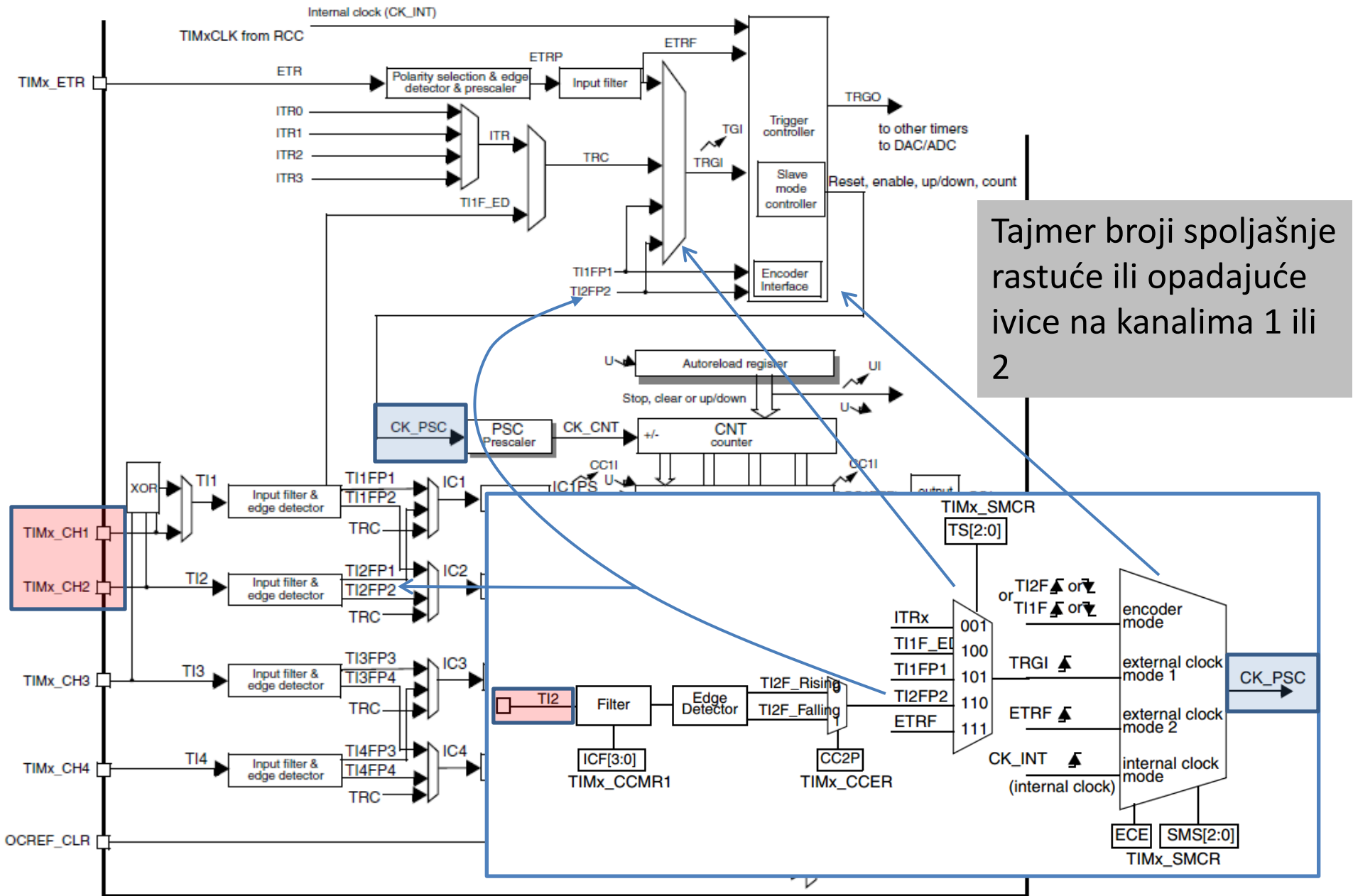
# Brojač gore-dole (Center-aligned / Up-down counting mode)



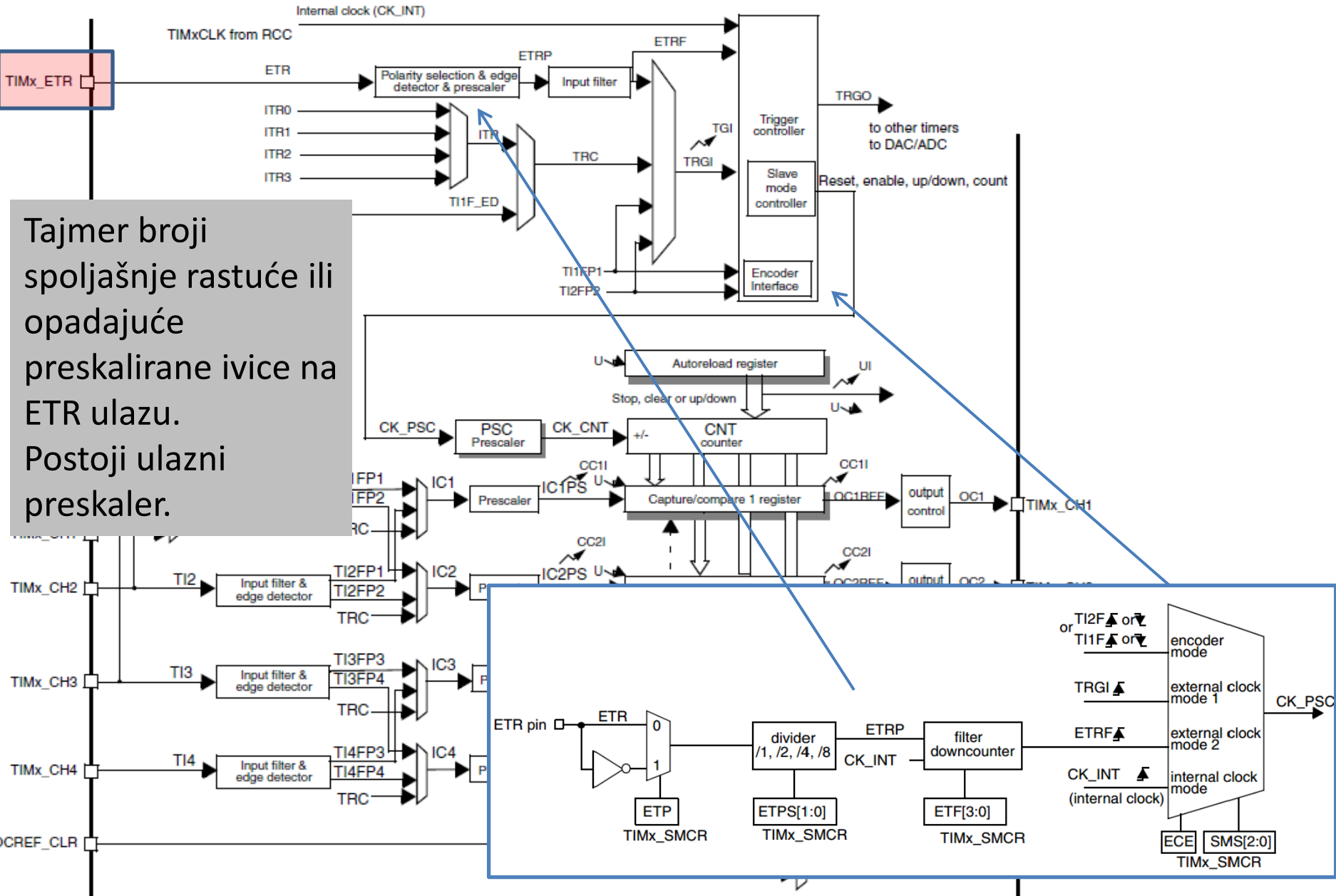
# Taktovanje - Internal mode



# External mode 1



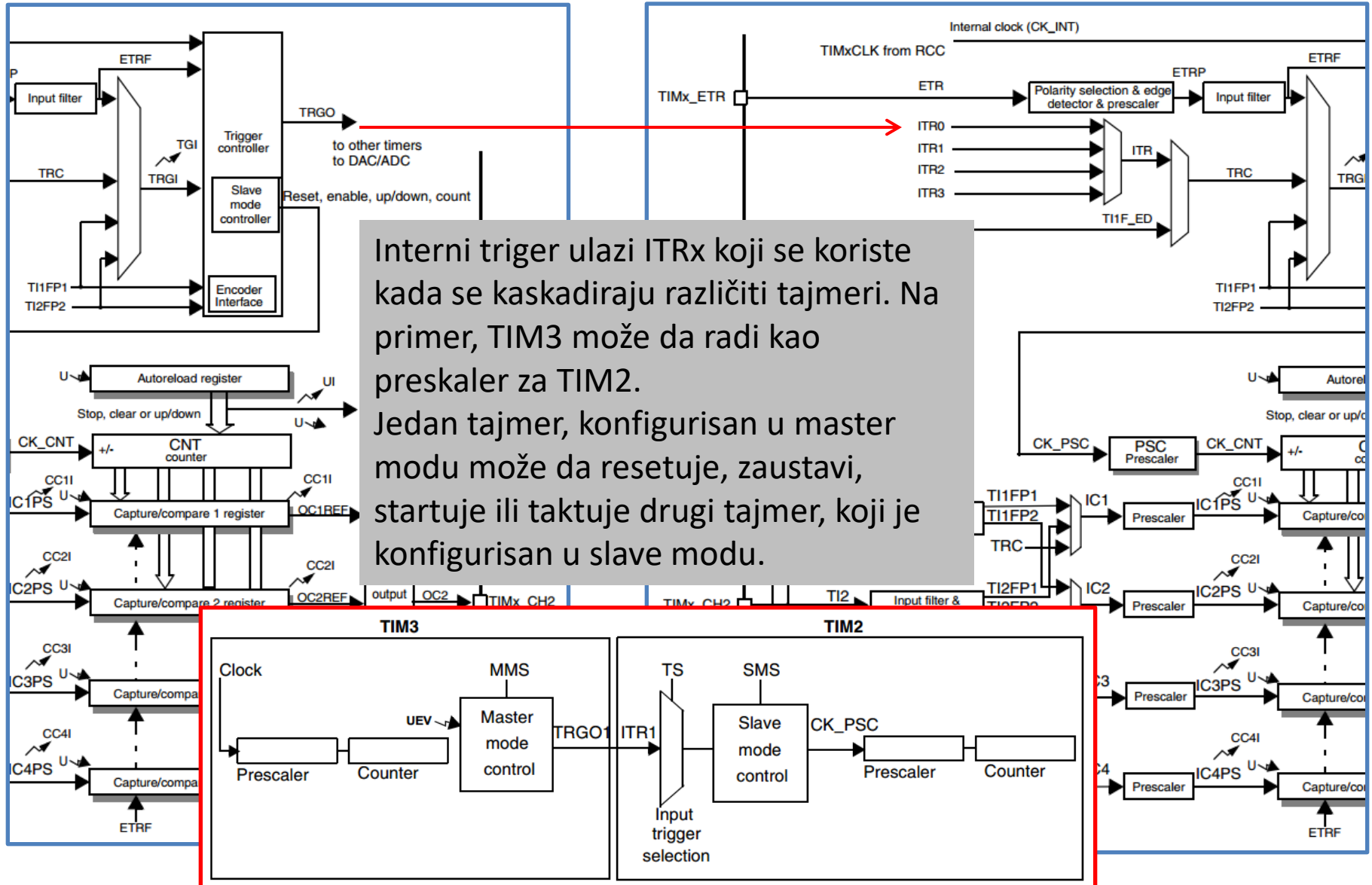
# External mode 2



Tajmer broji spoljašnje rastuće ili opadajuće preskalirane ivice na ETR ulazu. Postoji ulazni preskaler.



# Kaskadna veza tajmera

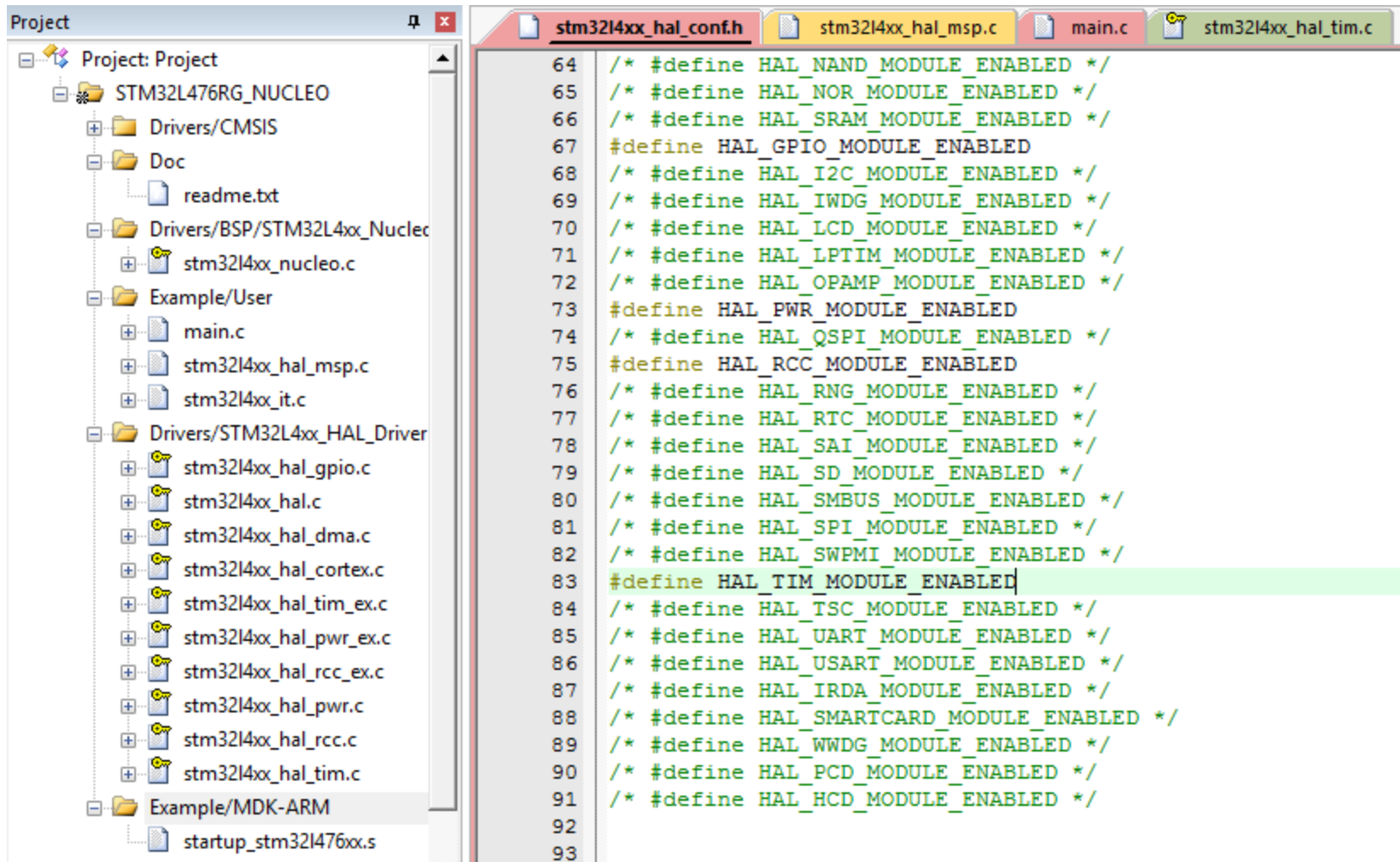


# STM CUBE

## Projektat TIM\_TimeBase

```
109     2) by calling HAL API function HAL_RCC_GetSysClockFreq()
110     3) each time HAL_RCC_ClockConfig() is called to configure the system clock
111     ----- */
112
113     /* Compute the prescaler value to have TIMx counter clock equal to 10000 Hz */
114     uwPrescalerValue = (uint32_t)(SystemCoreClock / 10000) - 1;
115
116     /* Set TIMx instance */
117     TimHandle.Instance = TIMx;
118
119     /* Initialize TIMx peripheral as follows:
120     + Period = 10000 - 1
121     + Prescaler = (SystemCoreClock/10000) - 1
122     + ClockDivision = 0
123     + Counter direction = Up
124     */
125     TimHandle.Init.Period           = 10000 - 1;
126     TimHandle.Init.Prescaler        = uwPrescalerValue;
127     TimHandle.Init.ClockDivision    = 0;
128     TimHandle.Init.CounterMode     = TIM_COUNTERMODE_UP;
129     TimHandle.Init.RepetitionCounter = 0;
130
131     if (HAL_TIM_Base_Init(&TimHandle) != HAL_OK)
132     {
133         /* Initialization Error */
134         Error_Handler();
135     }
136
137     /*##-2- Start the TIM Base generation in interrupt mode #####*/
138     /* Start Channell */
139     if (HAL_TIM_Base_Start_IT(&TimHandle) != HAL_OK)
140     {
141         /* Starting Error */
142         Error_Handler();
143     }
144
145     while (1)
146     {
```

# Ako koristimo periferije potrebno je to da označimo u .conf fajlu



The image shows a screenshot of an IDE. On the left, a project tree is visible with the following structure:

- Project: Project
  - STM32L476RG\_NUCLEO
    - Drivers/CMSIS
    - Doc
      - readme.txt
    - Drivers/BSP/STM32L4xx\_Nucleo
      - stm32l4xx\_nucleo.c
    - Example/User
      - main.c
      - stm32l4xx\_hal\_msp.c
      - stm32l4xx\_it.c
    - Drivers/STM32L4xx\_HAL\_Driver
      - stm32l4xx\_hal\_gpio.c
      - stm32l4xx\_hal.c
      - stm32l4xx\_hal\_dma.c
      - stm32l4xx\_hal\_cortex.c
      - stm32l4xx\_hal\_tim\_ex.c
      - stm32l4xx\_hal\_pwr\_ex.c
      - stm32l4xx\_hal\_rcc\_ex.c
      - stm32l4xx\_hal\_pwr.c
      - stm32l4xx\_hal\_rcc.c
      - stm32l4xx\_hal\_tim.c
    - Example/MDK-ARM
      - startup\_stm32l476xx.s

On the right, the editor shows the file `stm32l4xx_hal_conf.h` with the following code:

```
64 /* #define HAL_NAND_MODULE_ENABLED */
65 /* #define HAL_NOR_MODULE_ENABLED */
66 /* #define HAL_SRAM_MODULE_ENABLED */
67 #define HAL_GPIO_MODULE_ENABLED
68 /* #define HAL_I2C_MODULE_ENABLED */
69 /* #define HAL_IWDG_MODULE_ENABLED */
70 /* #define HAL_LCD_MODULE_ENABLED */
71 /* #define HAL_LPTIM_MODULE_ENABLED */
72 /* #define HAL_OPAMP_MODULE_ENABLED */
73 #define HAL_PWR_MODULE_ENABLED
74 /* #define HAL_QSPI_MODULE_ENABLED */
75 #define HAL_RCC_MODULE_ENABLED
76 /* #define HAL_RNG_MODULE_ENABLED */
77 /* #define HAL_RTC_MODULE_ENABLED */
78 /* #define HAL_SAI_MODULE_ENABLED */
79 /* #define HAL_SD_MODULE_ENABLED */
80 /* #define HAL_SMBUS_MODULE_ENABLED */
81 /* #define HAL_SPI_MODULE_ENABLED */
82 /* #define HAL_SWPMI_MODULE_ENABLED */
83 #define HAL_TIM_MODULE_ENABLED
84 /* #define HAL_TSC_MODULE_ENABLED */
85 /* #define HAL_UART_MODULE_ENABLED */
86 /* #define HAL_USART_MODULE_ENABLED */
87 /* #define HAL_IRDA_MODULE_ENABLED */
88 /* #define HAL_SMARTCARD_MODULE_ENABLED */
89 /* #define HAL_WWDG_MODULE_ENABLED */
90 /* #define HAL_PCD_MODULE_ENABLED */
91 /* #define HAL_HCD_MODULE_ENABLED */
92
93
```

# Šta to piše u drajverskim fajlovima?

```
stm3214xx_hal_tim.c
1  /**
2  *
3  * @file   stm3214xx_hal_tim.c
4  * @author MCD Application Team
5  * @version V1.4.0
6  * @date   26-February-2016
7  * @brief  TIM HAL module driver.
8  *
9  * This file provides firmware functions to manage the following
10 * functionalities of the Timer (TIM) peripheral:
11 *
12 *   + Time Base Initialization
13 *   + Time Base Start
14 *   + Time Base Start Interruption
15 *   + Time Base Start DMA
16 *   + Time Output Compare/PWM Initialization
17 *   + Time Output Compare/PWM Channel Configuration
18 *   + Time Output Compare/PWM Start
19 *   + Time Output Compare/PWM Start Interruption
20 *   + Time Output Compare/PWM Start DMA
21 *   + Time Input Capture Initialization
22 *   + Time Input Capture Channel Configuration
23 *   + Time Input Capture Start
24 *   + Time Input Capture Start Interruption
25 *   + Time Input Capture Start DMA
26 *   + Time One Pulse Initialization
27 *   + Time One Pulse Channel Configuration
28 *   + Time One Pulse Start
29 *   + Time Encoder Interface Initialization
30 *   + Time Encoder Interface Start
31 *   + Time Encoder Interface Start Interruption
32 *   + Time Encoder Interface Start DMA
```

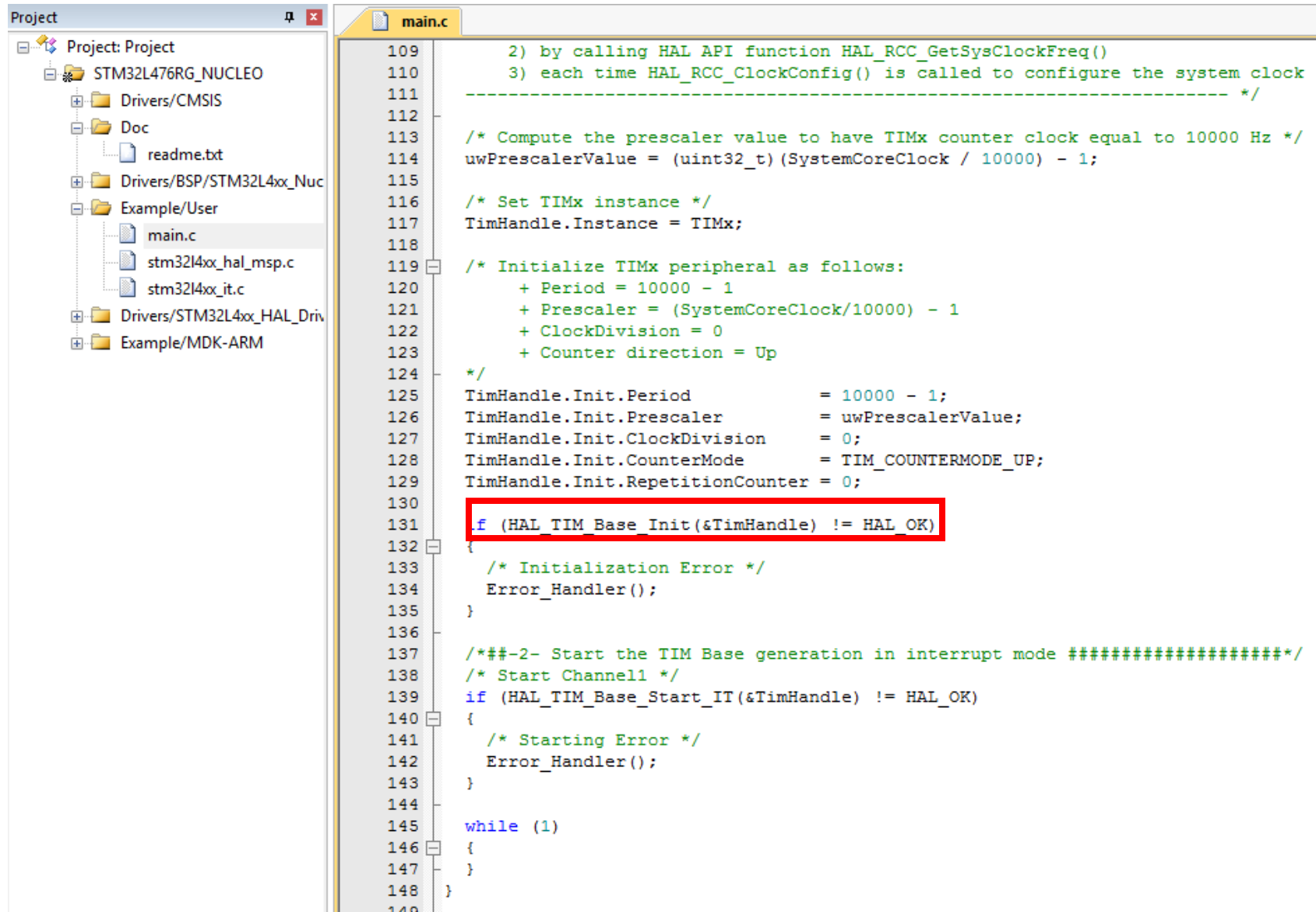
```
31 *           + Commutation Event configuration with Interruption and DMA
32 *           + Time OCREf clear configuration
33 *           + Time External Clock configuration
34 @verbatim
35 =====
36          ##### TIMER Generic features #####
37 =====
38 [...] The Timer features include:
39     (#) 16-bit up, down, up/down auto-reload counter.
40     (#) 16-bit programmable prescaler allowing dividing (also on the fly) the
41         counter clock frequency either by any factor between 1 and 65536.
42     (#) Up to 4 independent channels for:
43         (++) Input Capture
44         (++) Output Compare
45         (++) PWM generation (Edge and Center-aligned Mode)
46         (++) One-pulse mode output
47
48     ##### How to use this driver #####
49 =====
50 [...]
51     (#) Initialize the TIM low level resources by implementing the following functions
52         depending on the selected feature:
53         (++) Time Base : HAL_TIM_Base_MspInit()
54         (++) Input Capture : HAL_TIM_IC_MspInit()
55         (++) Output Compare : HAL_TIM_OC_MspInit()
56         (++) PWM generation : HAL_TIM_PWM_MspInit()
57         (++) One-pulse mode output : HAL_TIM_OnePulse_MspInit()
58         (++) Encoder mode output : HAL_TIM_Encoder_MspInit()
```

```

59
60 (#) Initialize the TIM low level resources :
61 (##) Enable the TIM interface clock using __HAL_RCC_TIMx_CLK_ENABLE();
62 (##) TIM pins configuration
63 (++) Enable the clock for the TIM GPIOs using the following function:
64 __HAL_RCC_GPIOx_CLK_ENABLE();
65 (++) Configure these TIM pins in Alternate function mode using HAL_GPIO_Init();
66
67 (#) The external Clock can be configured, if needed (the default clock is the
68 internal clock from the APBx), using the following function:
69 HAL_TIM_ConfigClockSource, the clock configuration should be done before
70 any start function.
71
72 (#) Configure the TIM in the desired functioning mode using one of the
73 Initialization function of this driver:
74 (++) HAL_TIM_Base_Init: to use the Timer to generate a simple time base
75 (++) HAL_TIM_OC_Init and HAL_TIM_OC_ConfigChannel: to use the Timer to generate an
76 Output Compare signal.
77 (++) HAL_TIM_PWM_Init and HAL_TIM_PWM_ConfigChannel: to use the Timer to generate a
78 PWM signal.
79 (++) HAL_TIM_IC_Init and HAL_TIM_IC_ConfigChannel: to use the Timer to measure an
80 external signal.
81 (++) HAL_TIM_OnePulse_Init and HAL_TIM_OnePulse_ConfigChannel: to use the Timer
82 in One Pulse Mode.
83 (++) HAL_TIM_Encoder_Init: to use the Timer Encoder Interface.
84
85 (#) Activate the TIM peripheral using one of the start functions depending from the feature used:
86 (++) Time Base : HAL_TIM_Base_Start(), HAL_TIM_Base_Start_DMA(), HAL_TIM_Base_Start_IT()
87 (++) Input Capture : HAL_TIM_IC_Start(), HAL_TIM_IC_Start_DMA(), HAL_TIM_IC_Start_IT()
88 (++) Output Compare : HAL_TIM_OC_Start(), HAL_TIM_OC_Start_DMA(), HAL_TIM_OC_Start_IT()
89 (++) PWM generation : HAL_TIM_PWM_Start(), HAL_TIM_PWM_Start_DMA(), HAL_TIM_PWM_Start_IT()
90 (++) One-pulse mode output : HAL_TIM_OnePulse_Start(), HAL_TIM_OnePulse_Start_IT()
91 (++) Encoder mode output : HAL_TIM_Encoder_Start(), HAL_TIM_Encoder_Start_DMA(), HAL_TIM_Encoder_Start_IT()
92
93 (#) The DMA Burst is managed with the two following functions:
94 HAL_TIM_DMABurst_WriteStart()
95 HAL_TIM_DMABurst_ReadStart()

```

# Inicijalizacija vremenske baze tajmera



```
109     2) by calling HAL API function HAL_RCC_GetSysClockFreq()
110     3) each time HAL_RCC_ClockConfig() is called to configure the system clock
111     ----- */
112
113     /* Compute the prescaler value to have TIMx counter clock equal to 10000 Hz */
114     uwPrescalerValue = (uint32_t)(SystemCoreClock / 10000) - 1;
115
116     /* Set TIMx instance */
117     TimHandle.Instance = TIMx;
118
119     /* Initialize TIMx peripheral as follows:
120     + Period = 10000 - 1
121     + Prescaler = (SystemCoreClock/10000) - 1
122     + ClockDivision = 0
123     + Counter direction = Up
124     */
125     TimHandle.Init.Period           = 10000 - 1;
126     TimHandle.Init.Prescaler        = uwPrescalerValue;
127     TimHandle.Init.ClockDivision    = 0;
128     TimHandle.Init.CounterMode      = TIM_COUNTERMODE_UP;
129     TimHandle.Init.RepetitionCounter = 0;
130
131     if (HAL_TIM_Base_Init(&TimHandle) != HAL_OK)
132     {
133         /* Initialization Error */
134         Error_Handler();
135     }
136
137     /*##-2- Start the TIM Base generation in interrupt mode #####*/
138     /* Start Channell */
139     if (HAL_TIM_Base_Start_IT(&TimHandle) != HAL_OK)
140     {
141         /* Starting Error */
142         Error_Handler();
143     }
144
145     while (1)
146     {
147     }
148 }
```

```
h.c | stm32f1xx_hal_tim.c
*/
/**
 * @brief Initializes the TIM Time base Unit
 * parameters in the TIM_HandleTypeDef
 * @param htim : TIM Base handle
 * @retval HAL status
 */
HAL_StatusTypeDef HAL_TIM_Base_Init(TIM_HandleTypeDef
{
/* Check the TIM handle allocation */
if(htim == NULL)
{
return HAL_ERROR;
}

/* Check the parameters */
assert_param(IS_TIM_INSTANCE(htim->Insta
assert_param(IS_TIM_COUNTER_MODE(htim-
assert_param(IS_TIM_CLOCKDIVISION_DIV(

if(htim->State == HAL_TIM_STATE_RESET)
{
/* Allocate lock resource and initia
htim->Lock = HAL_UNLOCKED;

/* Init the low level hardware : GPI
HAL_TIM_Base_MspInit(htim);
}

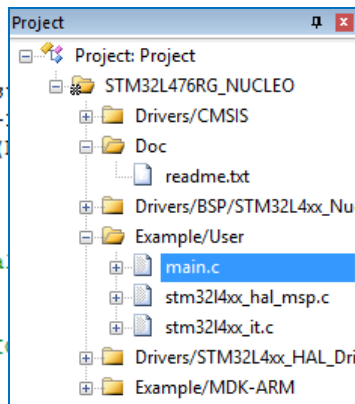
/* Set the TIM state */
htim->State= HAL_TIM_STATE_BUSY;

/* Set the Time Base configuration */
TIM_Base_SetConfig(htim->Instance, &ht

/* Initialize the TIM state*/
htim->State= HAL_TIM_STATE_READY;

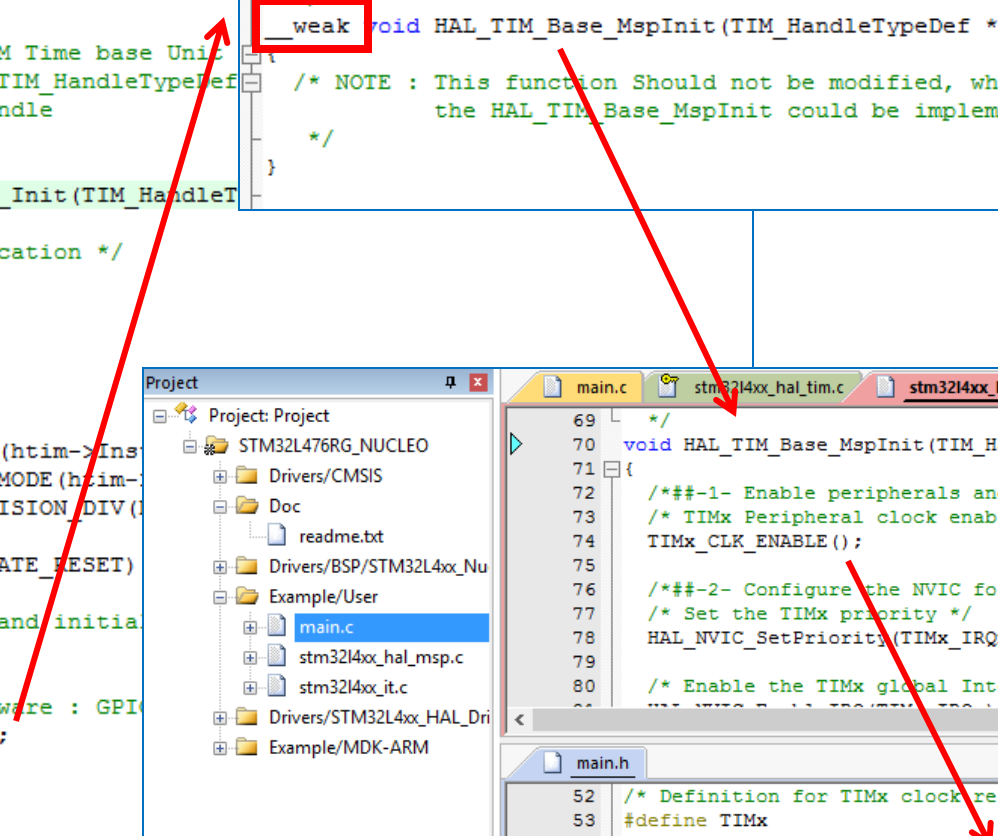
return HAL_OK;
}
}
```

```
h.c | stm32f1xx_hal_tim.c
}
/**
 * @brief Initializes the TIM Base MSP.
 * @param htim : TIM handle
 * @retval None
 */
__weak void HAL_TIM_Base_MspInit(TIM_HandleTypeDef *htim)
{
/* NOTE : This function Should not be modified, when the callback is needed,
the HAL_TIM_Base_MspInit could be implemented in the user file
*/
}
}
```



```
main.c | stm32l4xx_hal_tim.c
69 */
70 void HAL_TIM_Base_MspInit(TIM_HandleTypeDef *htim)
71 {
72 /**-1- Enable peripherals and GPIO Clocks #####
73 /* TIMx Peripheral clock enable */
74 TIMx_CLK_ENABLE();
75
76 /**-2- Configure the NVIC for TIMx #####
77 /* Set the TIMx priority */
78 HAL_NVIC_SetPriority(TIMx_IRQn, 3, 0);
79
80 /* Enable the TIMx global Interrupt */
81
```

```
main.h
52 /* Definition for TIMx clock resources */
53 #define TIMx TIM3
54 #define TIMx_CLK_ENABLE() __HAL_RCC_TIM3_CLK_ENABLE()
55
56 /* Definition for TIMx's NVIC */
57 #define TIMx_IRQn TIM3_IRQn
58 #define TIMx_IRQHandler TIM3_IRQHandler
59
60 /* Exported functions -----
61
62 #endif /* __MAIN_H */
```



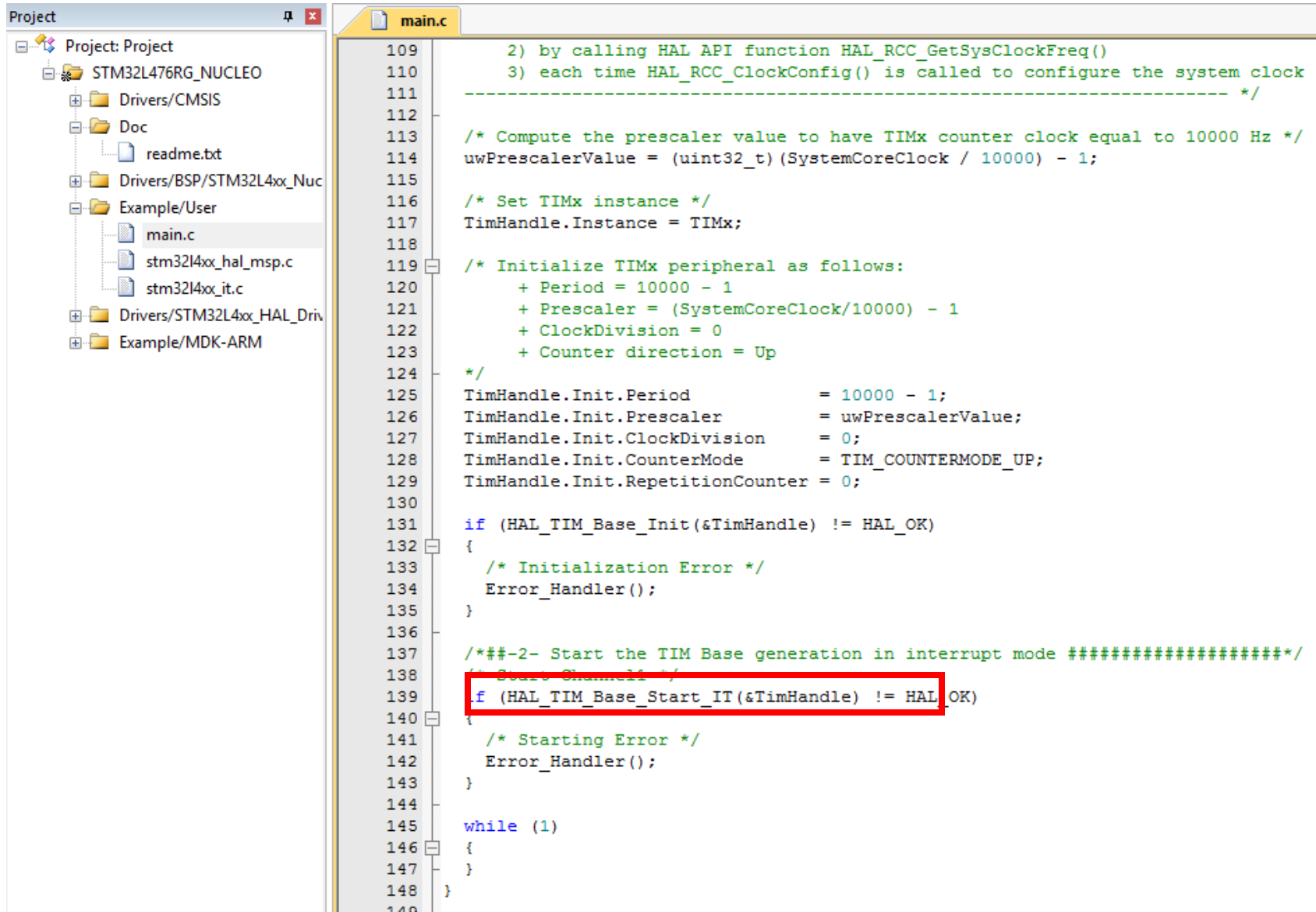


# MspInit()

- Dovodimo takt tajmeru (ovakav ili onakav)
- Konfiguriramo prekide
- Dozvoljavamo prekide

```
70 void HAL_TIM_Base_MspInit(TIM_HandleTypeDef *htim)
71 {
72     /*##-1- Enable peripherals and GPIO Clocks #####*/
73     /* TIMx Peripheral clock enable */
74     TIMx_CLK_ENABLE();
75
76     /*##-2- Configure the NVIC for TIMx #####*/
77     /* Set the TIMx priority */
78     HAL_NVIC_SetPriority(TIMx_IRQn, 3, 0);
79
80     /* Enable the TIMx global Interrupt */
81     HAL_NVIC_EnableIRQ(TIMx_IRQn);
82 }
```

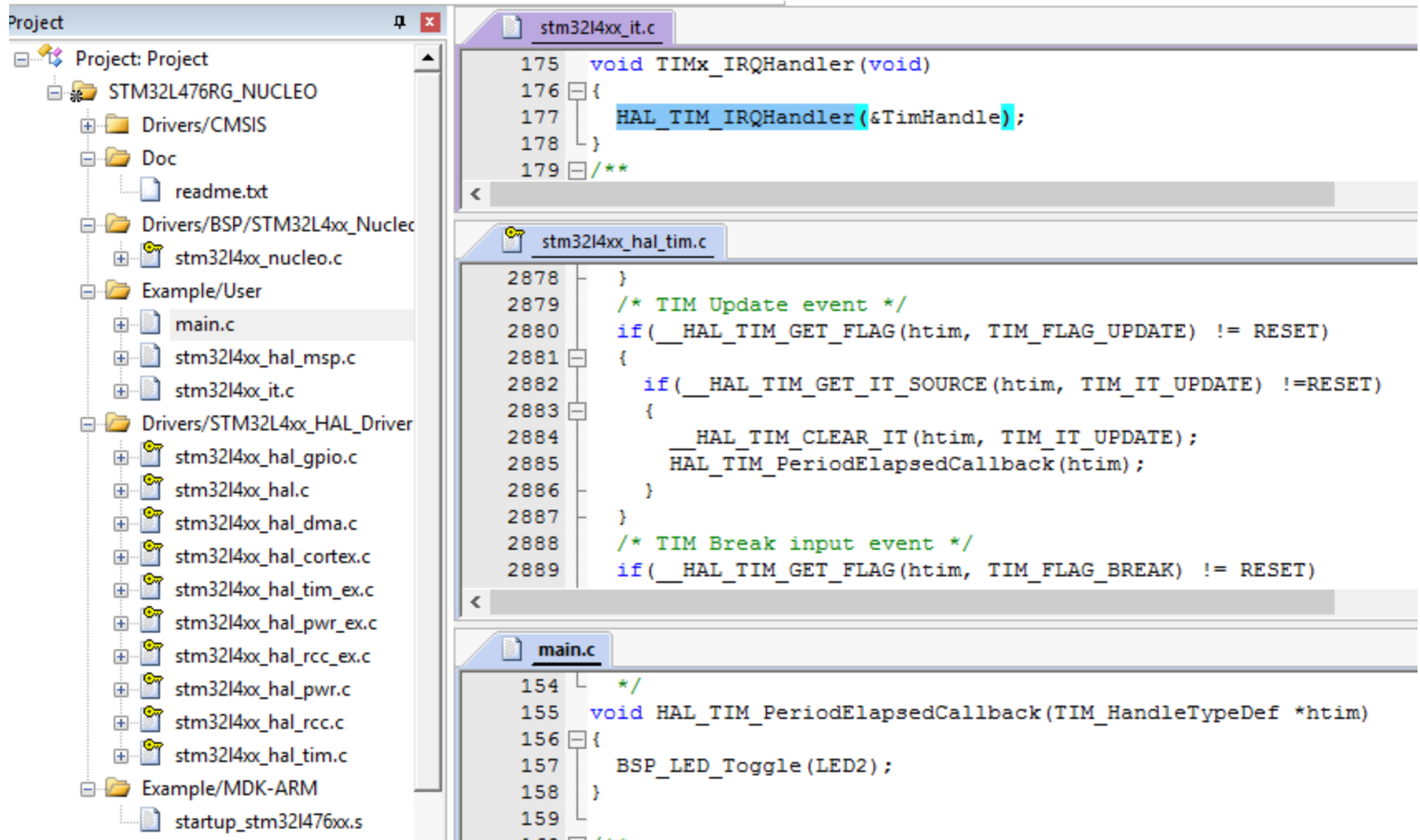
# Startujmo tajmer



The image shows a screenshot of an IDE with a project structure on the left and C code in the main editor. The project is named 'Project' and is located on an STM32L476RG\_NUCLEO. The code in 'main.c' is for initializing and starting a timer. A red box highlights the line `if (HAL_TIM_Base_Start_IT(&TimHandle) != HAL_OK)`.

```
Project
├── Project
│   ├── STM32L476RG_NUCLEO
│   │   ├── Drivers/CMSIS
│   │   ├── Doc
│   │   │   └── readme.txt
│   │   ├── Drivers/BSP/STM32L4xx_Nuc
│   │   ├── Example/User
│   │   │   ├── main.c
│   │   │   ├── stm32l4xx_hal_msp.c
│   │   │   └── stm32l4xx_it.c
│   │   ├── Drivers/STM32L4xx_HAL_Driv
│   │   └── Example/MDK-ARM
└── main.c
    109     2) by calling HAL API function HAL_RCC_GetSysClockFreq()
    110     3) each time HAL_RCC_ClockConfig() is called to configure the system clock
    111     ----- */
    112
    113     /* Compute the prescaler value to have TIMx counter clock equal to 10000 Hz */
    114     uwPrescalerValue = (uint32_t)(SystemCoreClock / 10000) - 1;
    115
    116     /* Set TIMx instance */
    117     TimHandle.Instance = TIMx;
    118
    119     /* Initialize TIMx peripheral as follows:
    120         + Period = 10000 - 1
    121         + Prescaler = (SystemCoreClock/10000) - 1
    122         + ClockDivision = 0
    123         + Counter direction = Up
    124     */
    125     TimHandle.Init.Period           = 10000 - 1;
    126     TimHandle.Init.Prescaler        = uwPrescalerValue;
    127     TimHandle.Init.ClockDivision    = 0;
    128     TimHandle.Init.CounterMode      = TIM_COUNTERMODE_UP;
    129     TimHandle.Init.RepetitionCounter = 0;
    130
    131     if (HAL_TIM_Base_Init(&TimHandle) != HAL_OK)
    132     {
    133         /* Initialization Error */
    134         Error_Handler();
    135     }
    136
    137     /*##-2- Start the TIM Base generation in interrupt mode #####*/
    138     /* Start Channel1 */
    139     if (HAL_TIM_Base_Start_IT(&TimHandle) != HAL_OK)
    140     {
    141         /* Starting Error */
    142         Error_Handler();
    143     }
    144
    145     while (1)
    146     {
    147     }
    148 }
```

# Imamo i prekid....



The screenshot displays an IDE interface with a project tree on the left and three code editors on the right. The project tree shows a project named 'Project' with a sub-project 'STM32L476RG\_NUCLEO'. Inside this sub-project, there are folders for 'Drivers/CMSIS', 'Doc', 'Drivers/BSP/STM32L4xx\_Nucleo', 'Example/User', and 'Drivers/STM32L4xx\_HAL\_Driver'. The 'Example/User' folder contains 'main.c', 'stm32l4xx\_hal\_msp.c', and 'stm32l4xx\_it.c'. The 'Drivers/STM32L4xx\_HAL\_Driver' folder contains several HAL driver files, including 'stm32l4xx\_hal\_gpio.c', 'stm32l4xx\_hal.c', 'stm32l4xx\_hal\_dma.c', 'stm32l4xx\_hal\_cortex.c', 'stm32l4xx\_hal\_tim\_ex.c', 'stm32l4xx\_hal\_pwr\_ex.c', 'stm32l4xx\_hal\_rcc\_ex.c', 'stm32l4xx\_hal\_pwr.c', 'stm32l4xx\_hal\_rcc.c', and 'stm32l4xx\_hal\_tim.c'. The 'Example/MDK-ARM' folder contains 'startup\_stm32l476xx.s'.

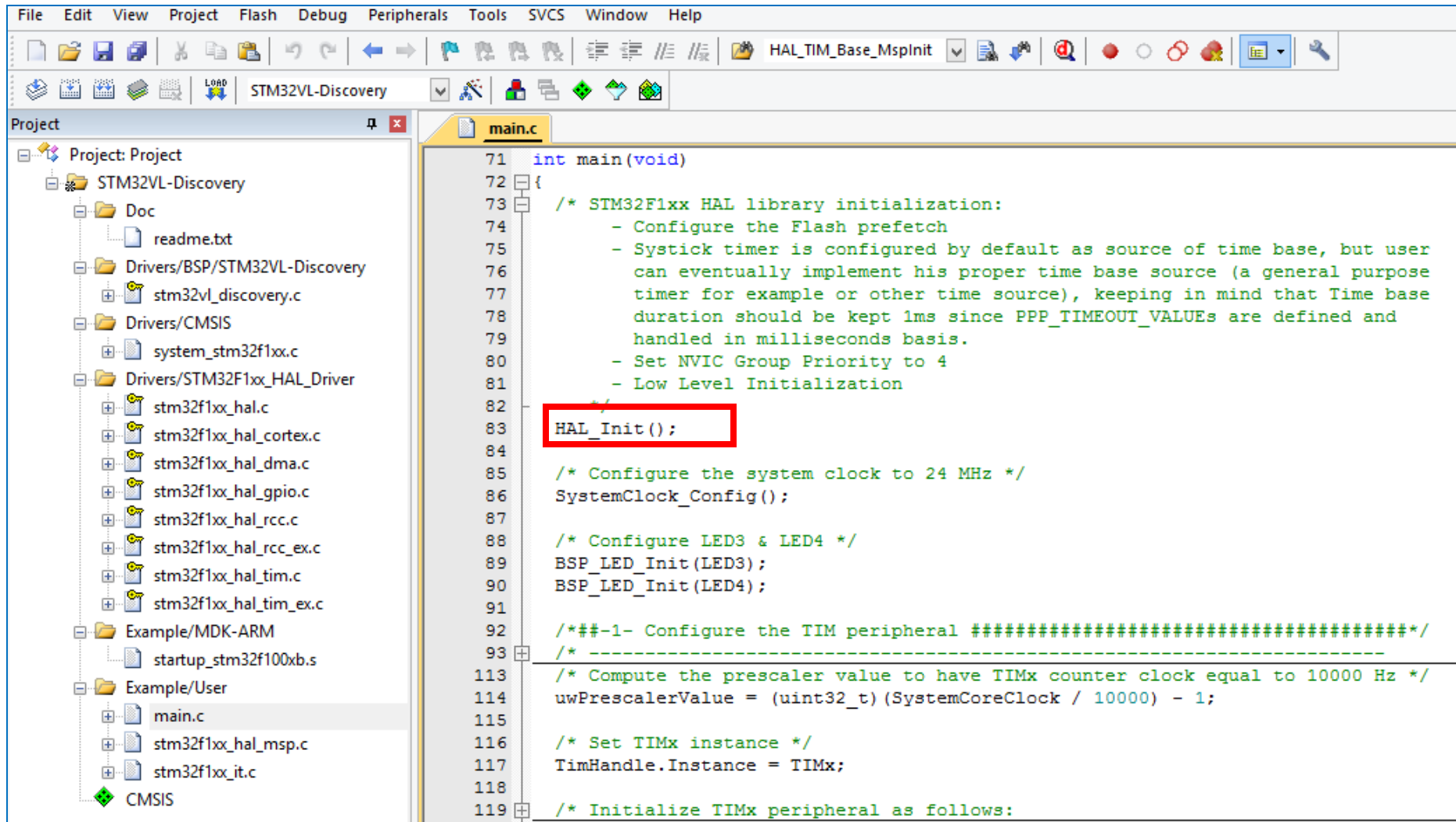
The three code editors show the following code:

```
stm32l4xx_it.c
175 void TIMx_IRQHandler(void)
176 {
177     HAL_TIM_IRQHandler(&TimHandle);
178 }
179 /**

stm32l4xx_hal_tim.c
2878 }
2879 /* TIM Update event */
2880 if(__HAL_TIM_GET_FLAG(htim, TIM_FLAG_UPDATE) != RESET)
2881 {
2882     if(__HAL_TIM_GET_IT_SOURCE(htim, TIM_IT_UPDATE) !=RESET)
2883     {
2884         __HAL_TIM_CLEAR_IT(htim, TIM_IT_UPDATE);
2885         HAL_TIM_PeriodElapsedCallback(htim);
2886     }
2887 }
2888 /* TIM Break input event */
2889 if(__HAL_TIM_GET_FLAG(htim, TIM_FLAG_BREAK) != RESET)

main.c
154 */
155 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
156 {
157     BSP_LED_Toggle(LED2);
158 }
159
```

# Generalna Inicijalizacija hardvera



```
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
STM32VL-Discovery
Project
  STM32VL-Discovery
    Doc
      readme.txt
    Drivers/BSP/STM32VL-Discovery
      stm32vl_discovery.c
    Drivers/CMSIS
      system_stm32f1xx.c
    Drivers/STM32F1xx_HAL_Driver
      stm32f1xx_hal.c
      stm32f1xx_hal_cortex.c
      stm32f1xx_hal_dma.c
      stm32f1xx_hal_gpio.c
      stm32f1xx_hal_rcc.c
      stm32f1xx_hal_rcc_ex.c
      stm32f1xx_hal_tim.c
      stm32f1xx_hal_tim_ex.c
    Example/MDK-ARM
      startup_stm32f100xb.s
    Example/User
      main.c
      stm32f1xx_hal_msp.c
      stm32f1xx_it.c
    CMSIS
main.c
71 int main(void)
72 {
73     /* STM32F1xx HAL library initialization:
74      - Configure the Flash prefetch
75      - SysTick timer is configured by default as source of time base, but user
76      can eventually implement his proper time base source (a general purpose
77      timer for example or other time source), keeping in mind that Time base
78      duration should be kept 1ms since PPP_TIMEOUT_VALUES are defined and
79      handled in milliseconds basis.
80      - Set NVIC Group Priority to 4
81      - Low Level Initialization
82     */
83     HAL_Init();
84
85     /* Configure the system clock to 24 MHz */
86     SystemClock_Config();
87
88     /* Configure LED3 & LED4 */
89     BSP_LED_Init(LED3);
90     BSP_LED_Init(LED4);
91
92     /*##-1- Configure the TIM peripheral #####*/
93     /* -----
113    /* Compute the prescaler value to have TIMx counter clock equal to 10000 Hz */
114    uwPrescalerValue = (uint32_t)(SystemCoreClock / 10000) - 1;
115
116    /* Set TIMx instance */
117    TimHandle.Instance = TIMx;
118
119    /* Initialize TIMx peripheral as follows:
```

# Generalna Inicijalizacija hardvera

```
in.c stm32f1xx_hal.c
/**
 * @brief This function configures the Flash prefetch,
 *        Configures time base source, NVIC and Low level hardware
 * @note This function is called at the beginning of program after reset and before
 *        the clock configuration
 * @note The time base configuration is based on MSI clock when exiting from Reset.
 *        Once done, time base tick start incrementing.
 *        In the default implementation, SysTick is used as source of time base.
 *        The tick variable is incremented each 1ms in its ISR.
 * @retval HAL status
 */
HAL_StatusTypeDef HAL_Init(void)
{
    /* Configure Flash prefetch */
    #if (PREFETCH_ENABLE != 0)
    #if defined(STM32F101x6) || defined(STM32F101xB) || defined(STM32F101xE) || defined(STM32F101xG) || \
        defined(STM32F102x6) || defined(STM32F102xB) || \
        defined(STM32F103x6) || defined(STM32F103xB) || defined(STM32F103xE) || defined(STM32F103xG) || \
        defined(STM32F105xC) || defined(STM32F107xC)

        /* Prefetch buffer is not available on value line devices */
        __HAL_FLASH_PREFETCH_BUFFER_ENABLE();
    #endif
    #endif /* PREFETCH_ENABLE */

    /* Set Interrupt Group Priority */
    HAL_NVIC_SetPriorityGrouping(NVIC_PRIORITYGROUP_4);

    /* Use systick as time base source and configure systick */
    HAL_InitTick(TICK_INT_PRIORITY);

    /* Init the low level hardware */
    HAL_MspInit();

    /* Return function status */
    return HAL_OK;
}
```

```
in.c stm32f1xx_hal.c
/**
 * @brief Initializes the MSP.
 * @retval None
 */
__weak void HAL_MspInit(void)
{
    /* NOTE : This function Should not be modified, when the callback is needed,
     * the HAL_MspInit could be implemented in the user file
     */
}
```

# Principi HAL drajvera kada su u pitanju kompleksne periferije:

1. Periferiju je najpre potrebno inicijalizovati
2. Inicijalizacija periferije se obavlja u drajveru ali se kao sporedni efekat poziva funkcija `HAL_PPP_Msplnit()` koja inicijalizuje low-level hardverske resurse.
3. Ponekad je potrebno inicijalizovati i neke posebne delove periferije posebni funkcijama ali to zavisi od aplikacije.
4. Periferije tipično kreću sa željenim radom tek pošto se pokrenu funkcijom `HAL_PPP_start()`
5. Ukoliko periferija generiše prekide korisnik sam piše svoju prekidnu funkciju i u njoj poziva `HAL_PPP_IRQHandler()` funkciju u kojoj se “servisira” prekid.
6. Ta funkcija dalje poziva `HAL_PPP_Callback()` funkciju koja ustvari “reaguje” na prekid
7. Korisnik sam implementira tu Callback funkciju.

# Zadatak

- Najpre pitanje: Koji tajmer je aktivan?
- Zadatak 1: promeniti podešavanja tako da sve ovo radi tajmer 4.
- Zadatak 2: Proširiti projekat tako da rade oba tajmera istovremeno, ali sa neki različitim podešavanjima.
- Problem: MspInit() funkciju će pozvati drajver prilikom inicijalizacije svakog tajmera. Kako da znam za potrebe čije inicijalizacije se poziva MspInit()?

# Zato se uvek prosleđuju i pokazivači na objekte koji se inicijalizuju....

- Hint:

<https://my.st.com/public/STe2ecomunities/mcu/Lists/STM32Java/Attachments/1249/tim.c>

```
stm32f1xx_hal_msp.c
68  /*
69  void HAL_TIM_Base_MspInit(TIM_HandleTypeDef *htim)
70  {
71  /**-1- Enable peripheral clock #####*/
72  /* TIMx Peripheral clock enable */
73  TIMx_CLK_ENABLE();
74
75  /**-2- Configure the NVIC for TIMx #####*/
76  /* Set the TIMx priority */
77  HAL_NVIC_SetPriority(TIMx_IRQn, 3, 0);
78
79  /* Enable the TIMx global Interrupt */
80  HAL_NVIC_EnableIRQ(TIMx_IRQn);
81  }
82
83

main.h
48  /* Exported constants -----*/
49  /* Exported macro -----*/
50  /* User can use this section to tailor TIMx instance used and associated
51  resources */
52  /* Definition for TIMx clock resources */
53  #define TIMx          TIM3
54  #define TIMx_CLK_ENABLE()  _HAL_RCC_TIM3_CLK_ENABLE()
55
56
57  /* Definition for TIMx's NVIC */
58  #define TIMx_IRQn      TIM3_IRQn
59  #define TIMx_IRQHandler TIM3_IRQHandler
60
61  /* Exported functions ----- */
62
```

**Predviđen pokazivač na strukturu se nigde ne koristi???**