

Primena mikrokontrolera MS1PMK EXTI, Timers, PWM

2017/2018

Nenad Jovičić

Marija Janković

EXTI

Extended Interrupt and Events Controller

- Mogućnost generisanja do 40 interrupt/events zahteva
- 26 konfigurabilnih linija (od čega 16 za GPIO)
- 14 direktnih linija
- Nezavisni registar maske za interrupt/event za svaku liniju
- Zasebni status bit
- Mogućnost softverskog emuliranja interrupt/event-a

Konfigurabilne linije

- Koriste ih **spoljašnji prekidi** dovedeni na GPIO kao i par periferija
- Da bi se podesio prekid potrebno je:
 - Odabrati na koju ivicu se prekid generiše (uzlaznu, silaznu ili obe)
 - U registru maske obezbediti da prekid na toj liniji nije maskiran
- Kada dođe do željene ivice generiše se prekidni zahtev.
- Zahtev se beleži u odgovarajućem bitu pending registra i čeka brisanje od strane korisnika nakon izvršenja prekidne rutine.

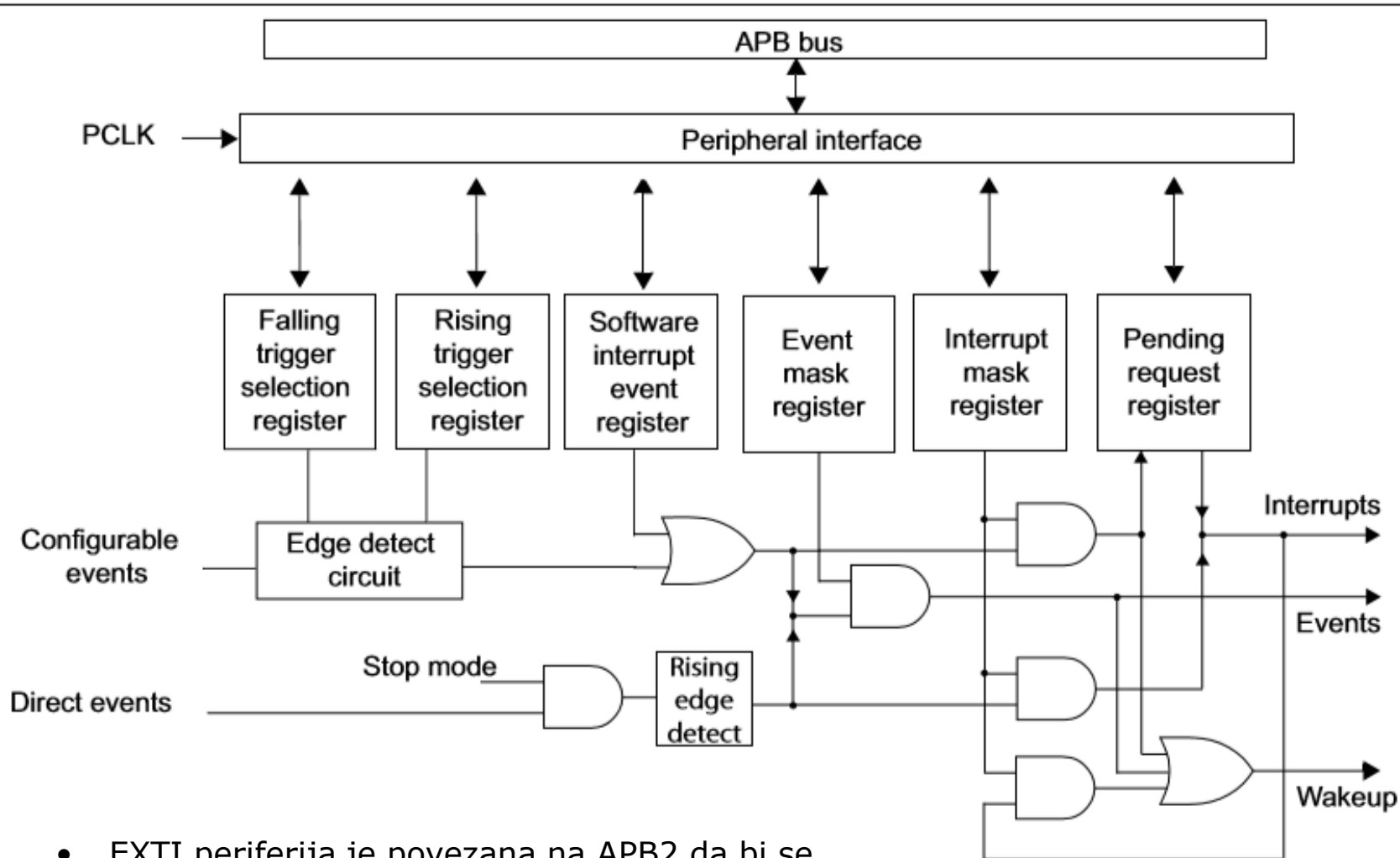
Konfigurabilne linije

- Na konfigurabilnim linijama je moguće aktivirati softverski interrupt/event
 - Podešavanjem da prekid ili događaj nisu maskirani
 - Upisom u odgovarajući software interrupt registar emulira se zahtev za prekidom ili događajem.

Direktne linije

- Koriste ih određene periferije uglavnom za generisanje zahteva za izlazak iz Stop moda ili nekog prekida.
- Prekid je automatski dozvoljen za direktne linije pa nije potrebno upisivati ništa u registre
- Ne postoji pending bit za direktne linije.
- Ukoliko sistem nije u STOP modu rada zahtev za prekid ili događaj preko direktnih linija neće biti prosleđen

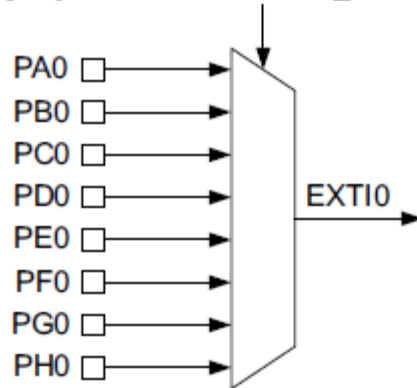
Figure 33. Configurable interrupt/event block diagram



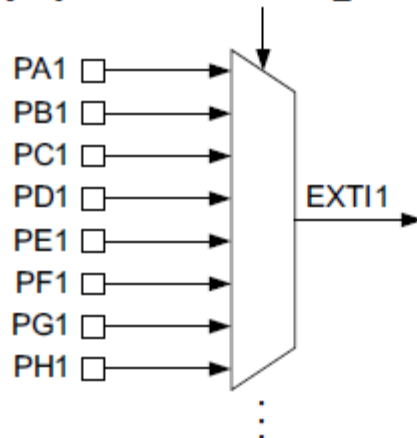
- EXTI periferija je povezana na APB2 da bi se skratilo vreme reagovanja (APB2 je u opštem slučaju brža od APB1 magistrale)

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 ⁽¹⁾	Line type
0-15	GPIO	configurable
16	PVD	configurable
17	OTG_FS wakeup event ⁽²⁾	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 ⁽²⁾	direct
24	I2C2 wakeup ⁽²⁾	direct
25	I2C3 wakeup	direct
26	USART1 wakeup ⁽²⁾	direct
27	USART2 wakeup ⁽²⁾	direct
28	USART3 wakeup ⁽²⁾	direct
29	UART4 wakeup ⁽²⁾	direct
30	UART5 wakeup ⁽²⁾	direct
31	LPUART1 wakeup	direct
32	LPTIM1	direct
33	LPTIM2 ⁽²⁾	direct
34	SWPMI1 wakeup ⁽²⁾	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

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

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

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)

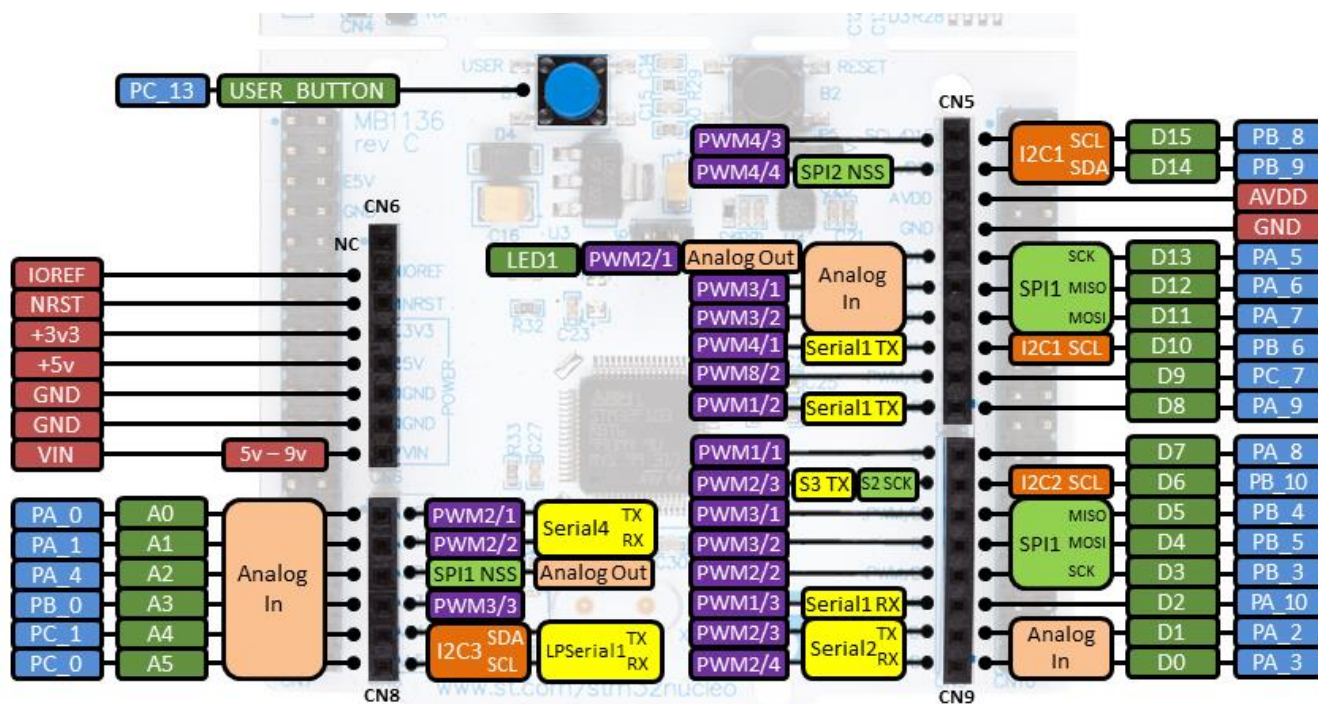
0: Falling trigger disabled (for Event and Interrupt) for input line

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

STM CUBE

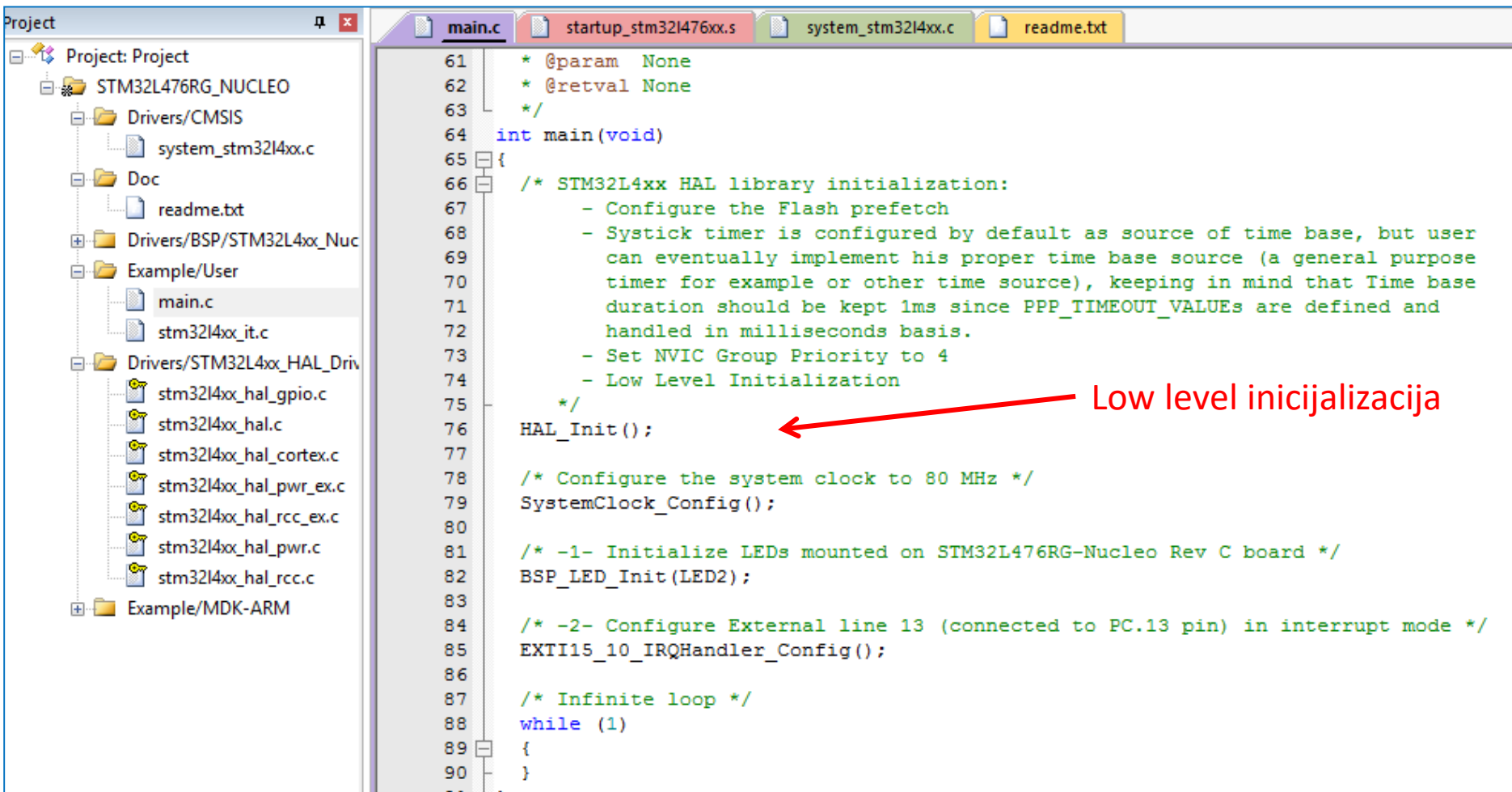
Projekat GPIO_EXTI

- ...\\STM32Cube_FW_L4_V1.4.0\\Projects\\STM32L476RG-Nucleo\\Examples\\GPIO\\GPIO_EXTI\\MDK-ARM
- Obezbediti izmenu stanja diode pritiskom na USER_BUTTON



STM CUBE

Projekat GPIO_EXTI



The image shows a screenshot of an IDE (likely Keil) with a project named 'Project' for the STM32L476RG_NUCLEO. The project structure is visible on the left, showing folders for Drivers/CMSIS, Doc, Drivers/BSP/STM32L4xx_Nuc, Example/User, and Drivers/STM32L4xx_HAL_Driv. The main.c file is open in the editor, showing the following code:

```
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     }
```

A red arrow points to the `HAL_Init();` line (line 76) with the text "Low level inicijalizacija".

Inicijalizacija EXTI prekida

```
main.c | stm32l4xx_hal_gpio.h  
5 typedef struct  
6 {  
7     uint32_t Pin;          /*!< Specifies the GPIO pins to be configured.  
8                           This parameter can be any value of @ref GPIO_pins */  
9  
10    uint32_t Mode;         /*!< Specifies the operating mode for the selected pins.  
11                           This parameter can be a value of @ref GPIO_mode */  
12  
13    uint32_t Pull;        /*!< Specifies the Pull-up or Pull-Down activation for the selected pins.  
14                           This parameter can be a value of @ref GPIO_pull */  
15  
16    uint32_t Speed;       /*!< Specifies the speed for the selected pins.  
17                           This parameter can be a value of @ref GPIO_speed */  
18  
19    uint32_t Alternate;   /*!< Peripheral to be connected to the selected pins  
20                           This parameter can be a value of @ref GPIOEx_Alternate_function_selection */  
21 } GPIO_InitTypeDef;
```

```
Project | main.c | stm32l4xx_hal_gpio.h  
151 * @param None  
152 * @retval None  
153 */  
154 static void EXTI15_10_IRQHandler_Config(void)  
155 {  
156     GPIO_InitTypeDef GPIO_InitStructure;  
157  
158     /* Enable GPIOC clock */  
159     __HAL_RCC_GPIOC_CLK_ENABLE();  
160  
161     /* Configure PC.13 pin as input floating */  
162     GPIO_InitStructure.Mode = GPIO_MODE_IT_RISING;  
163     GPIO_InitStructure.Pull = GPIO_NOPULL;  
164     GPIO_InitStructure.Pin = GPIO_PIN_13;  
165     HAL_GPIO_Init(GPIOC, &GPIO_InitStructure);  
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 }
```

```
main.c | stm32l4xx_hal_gpio.h  
*/  
#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
```

Drajverske funkcije – korisnik ne menja

```
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 EXTI line
 * @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 EXTI line
 * @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 */
}

```

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 to the EXTI lines
 * @retval None
 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if (GPIO_Pin == GPIO_PIN_13)
    {
        /* Toggle LED2 */
        BSP_LED_Toggle(LED2);
    }
}

```

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	EXTI9_5_IRQHandler	[WEAK]
283	EXPORT	EXTI4_3_IRQHandler	[WEAK]
284	EXPORT	EXTI0_IRQHandler	[WEAK]

```
startup_stm321476xx.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.

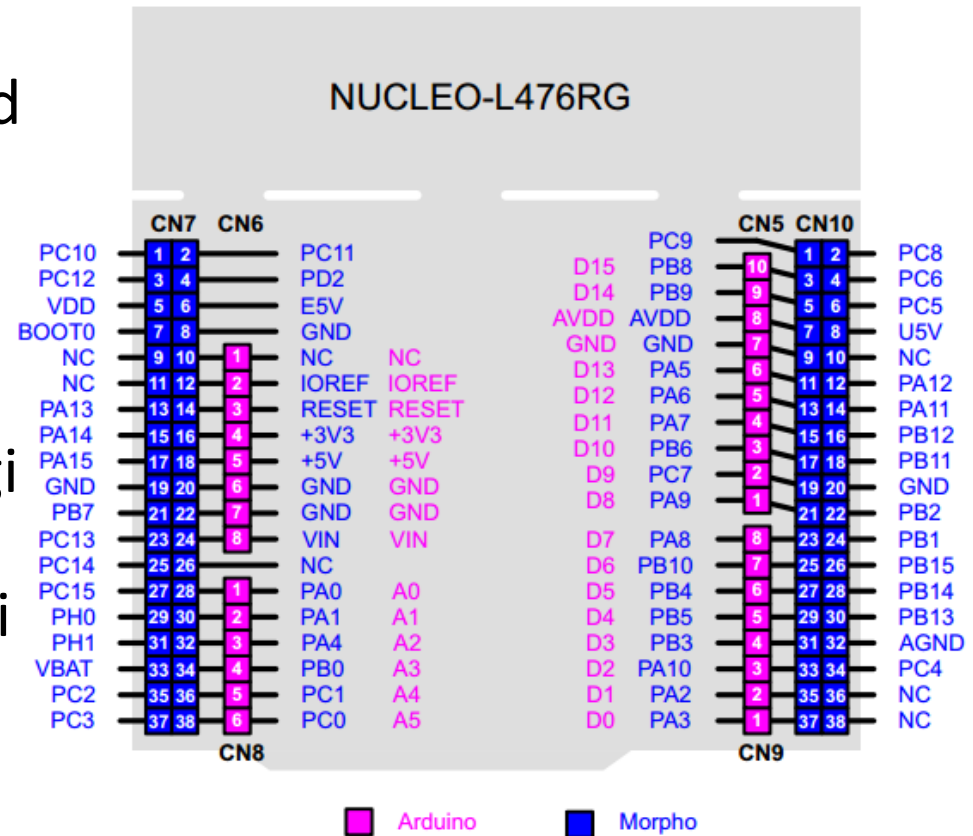
```
startup_stm321476xx.s
```

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

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

ZADATAK

- 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.



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` ()

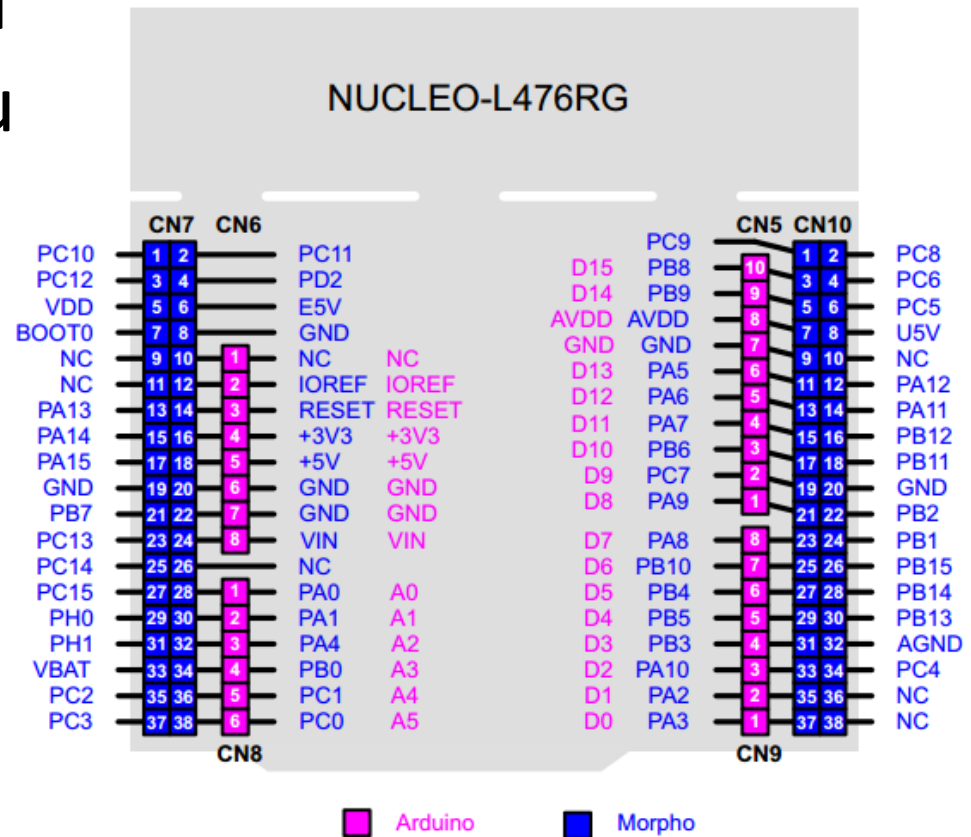
Enable IRQ.

void `disable_irq` ()

Disable IRQ.

ZADATAK – sada u mbed-u

- Napisati program koji obezbeđuje promenu stanja diode LED2 na prekid na pinu PA_8.
- Problem 1 – PA_8 je floating
- Rešenje – aktivirati pull-up



STM32Fxx - Tajmeri

- STM32 arhitektura poseduje nekoliko vrsta tajmera:
 - **Tajmeri opšte namene** koji se koriste za 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, automatsko 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 ⁽¹⁾	-
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

- General-purpose timer [cookbook](#)

Tajmeri opšte namene TIMx (x=2,3,4,5)

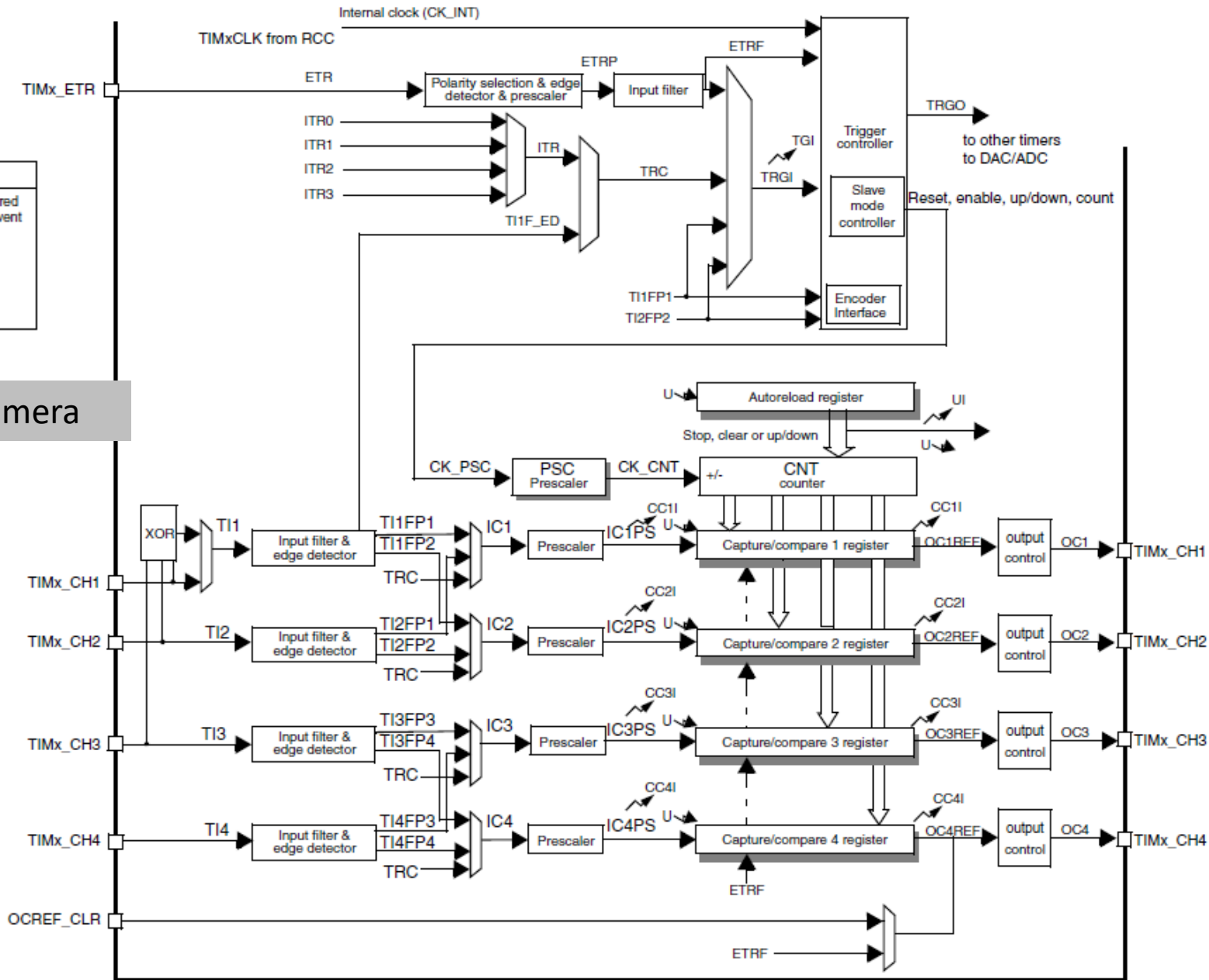
- 16-bitni (TIM3 i TIM4) ili 32-bitni brojač (TIM2 i TIM5) na gore, dole ili gore/dole.
- 16-bitni preskaler za ulazni takt
- Do 4 nezavisna kanala koji mogu da rade u izlaznom (output compare), ulaznom (input capture), PWM ili pojedinačnom impulsnom modu.
- Mogućnost sinhronizacije sa ostalim tajmerima.
- 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



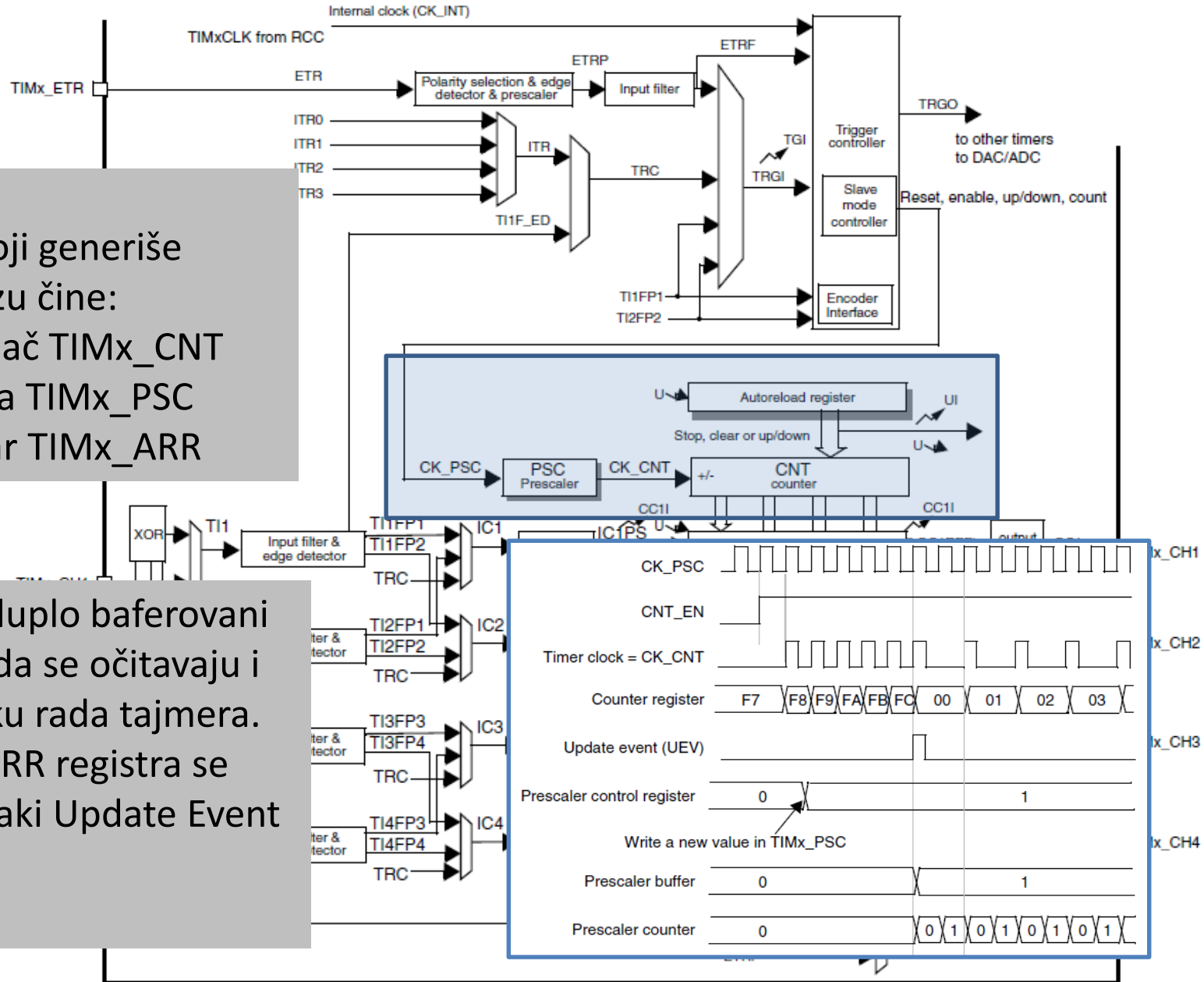
Vremenska baza

Vremenska baza

Deo tajmera koji generiše vremensku bazu čine:

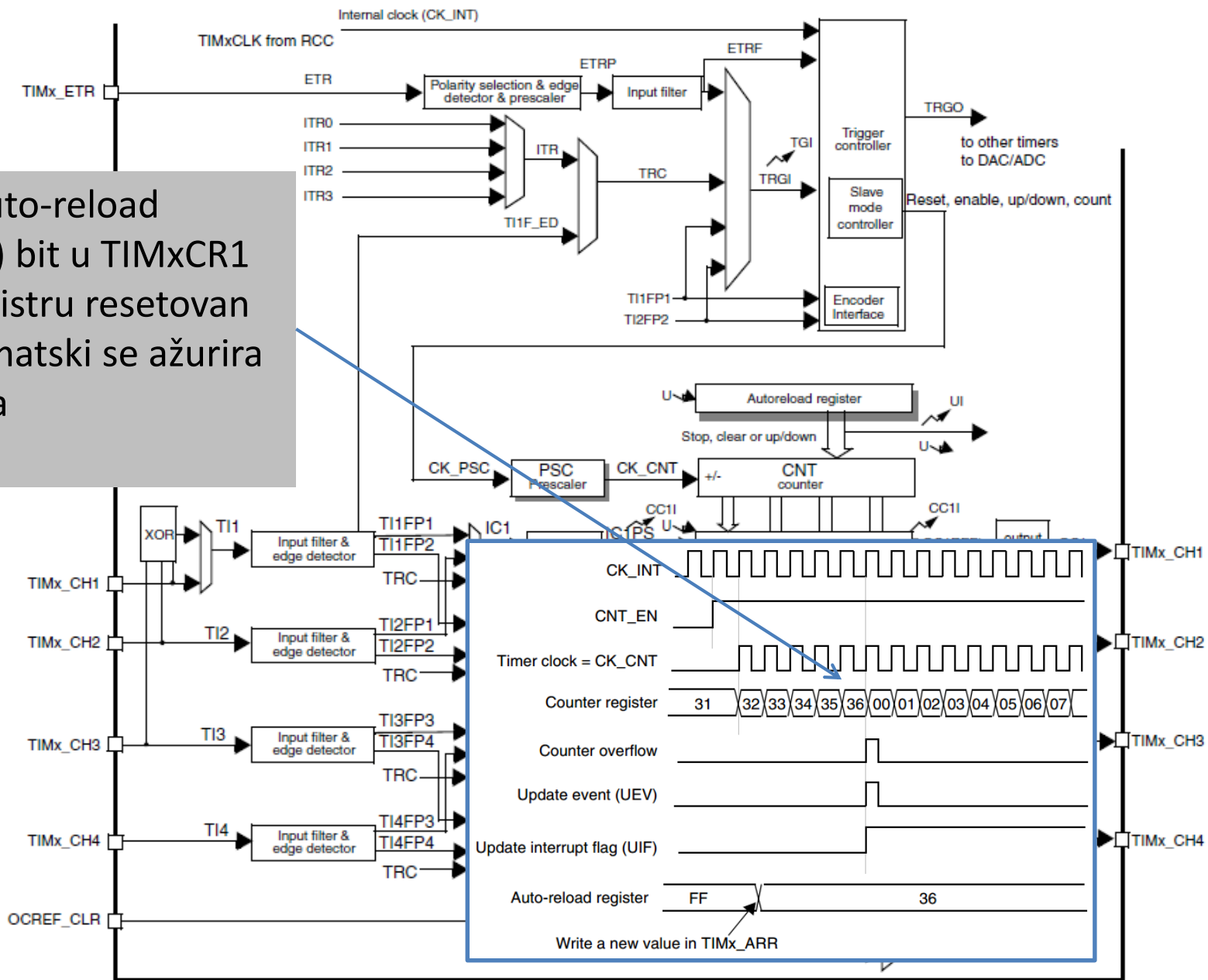
- Tajmerski brojač TIMx_CNT
- Preskaler takta TIMx_PSC
- Reload registar TIMx_ARR

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



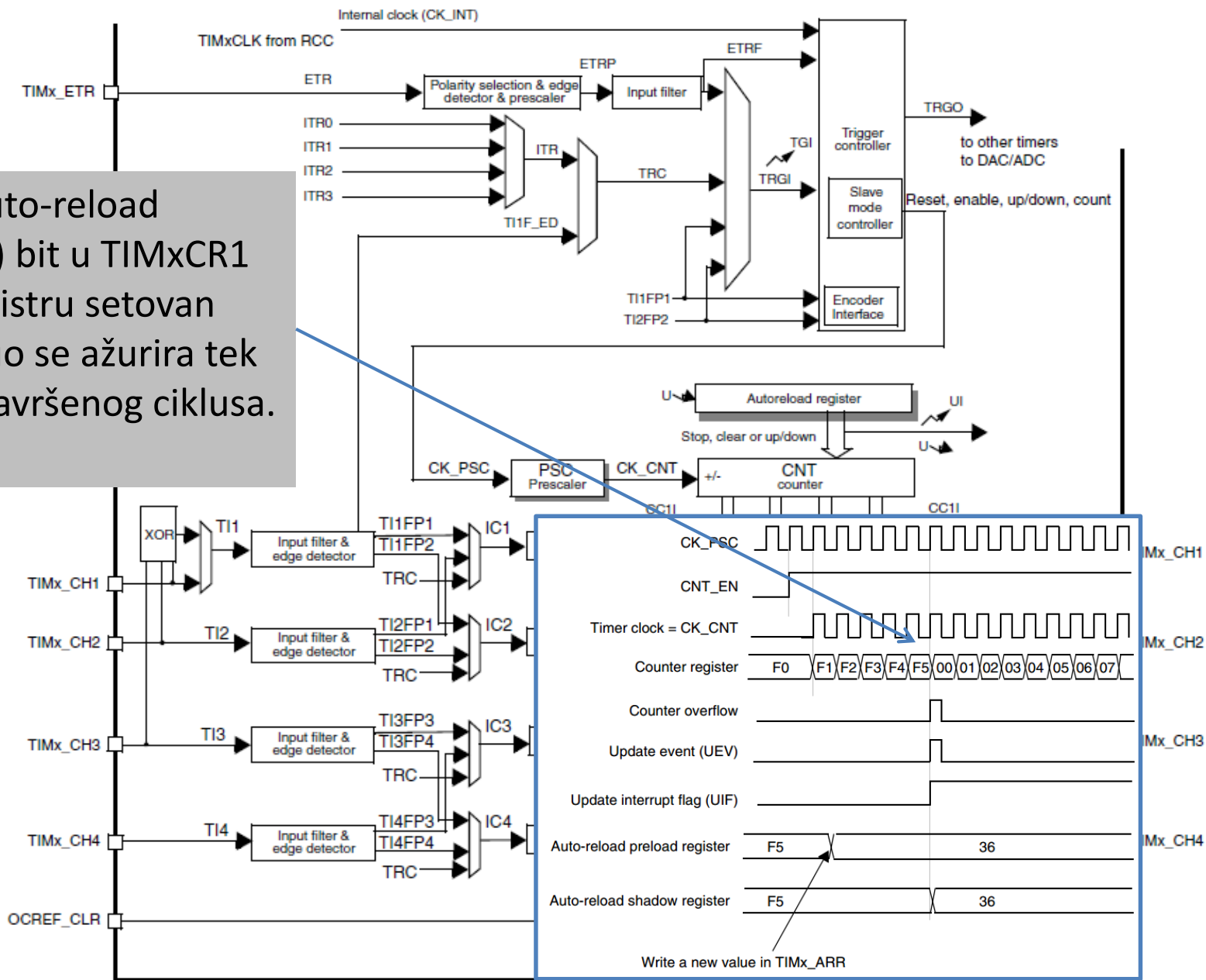
Brojač na gore – UP counting

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

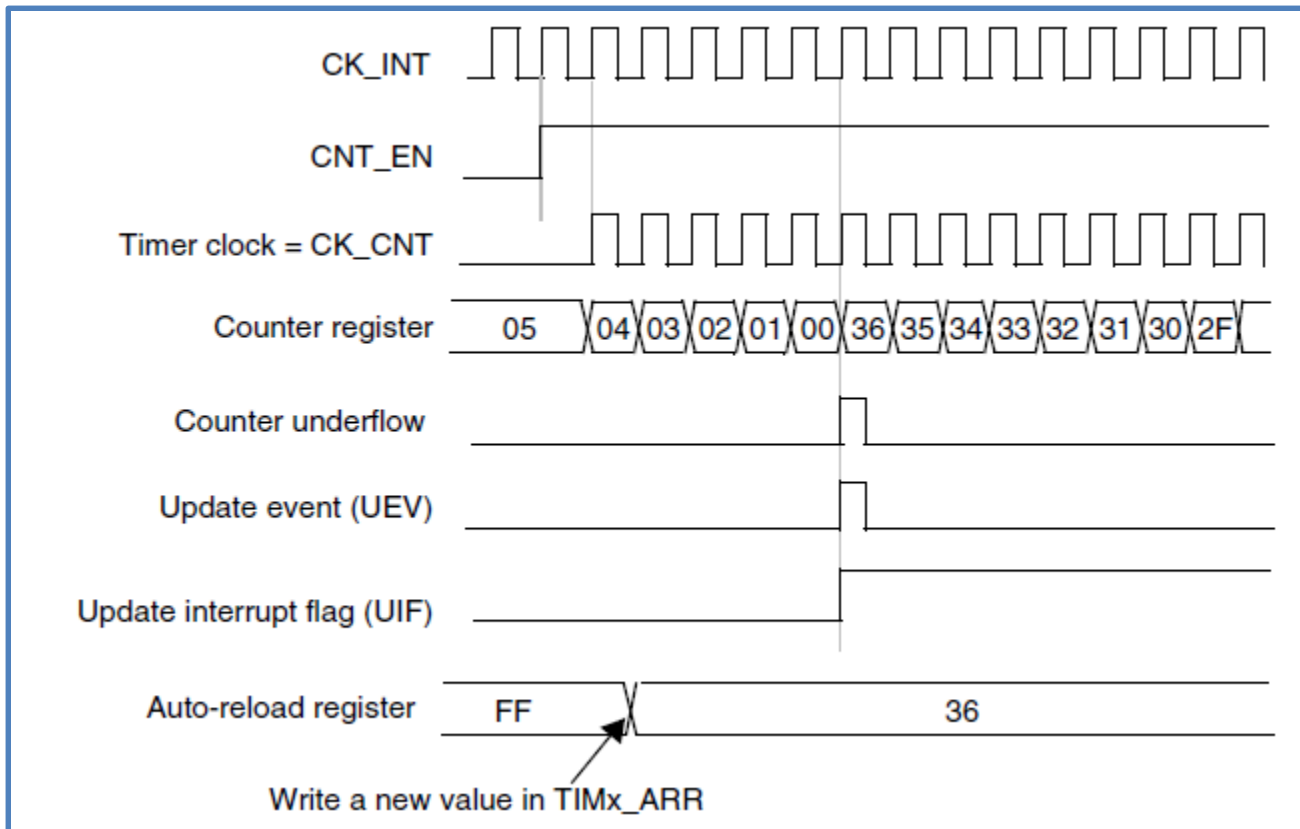


Brojač na gore – UP counting

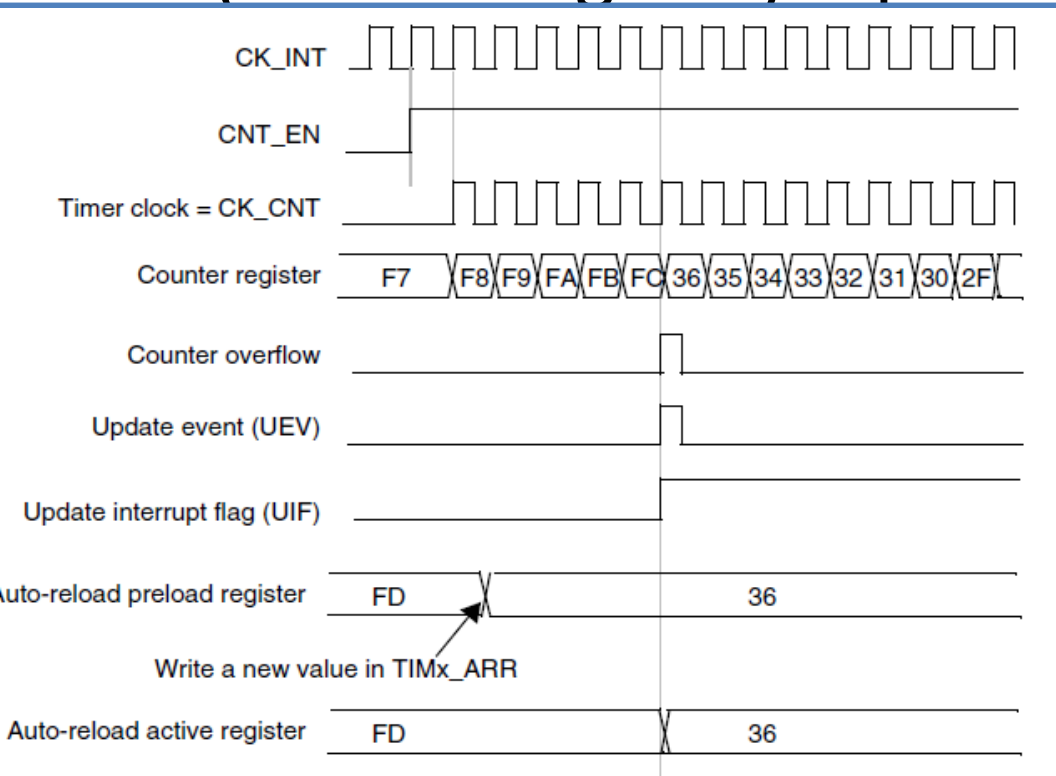
Ako je ARPE (auto-reload preload enable) bit u TIMxCR1 kontrolnom registru setovan (ARPE=1) moduo se ažurira tek nakon jednog završenog ciklusa.



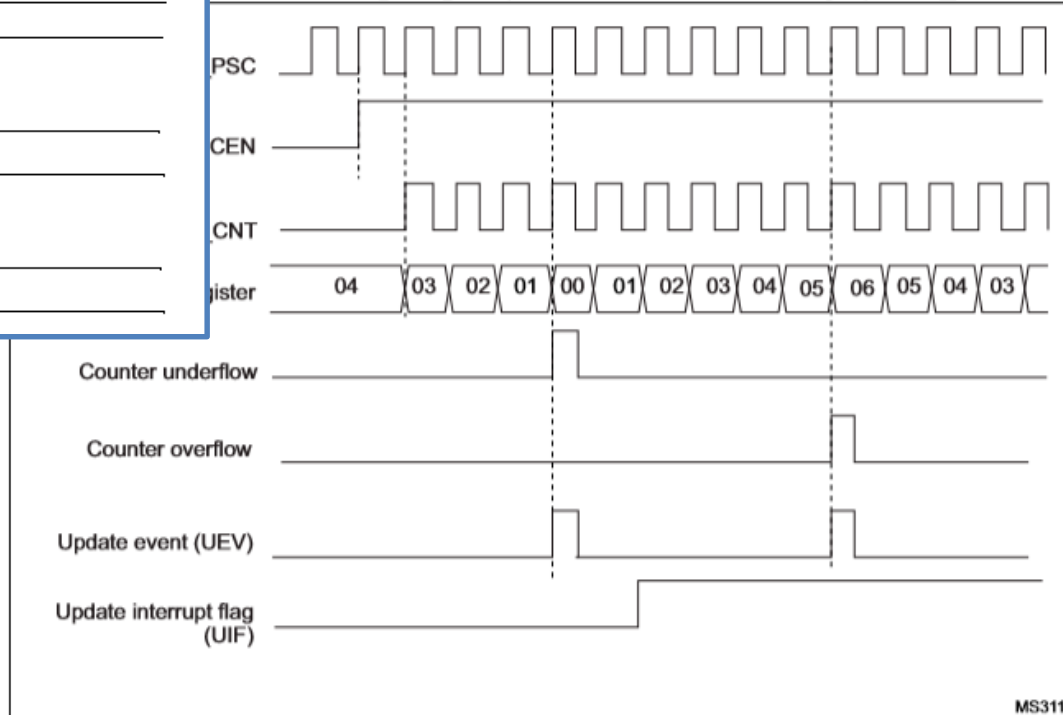
Downcounting mode



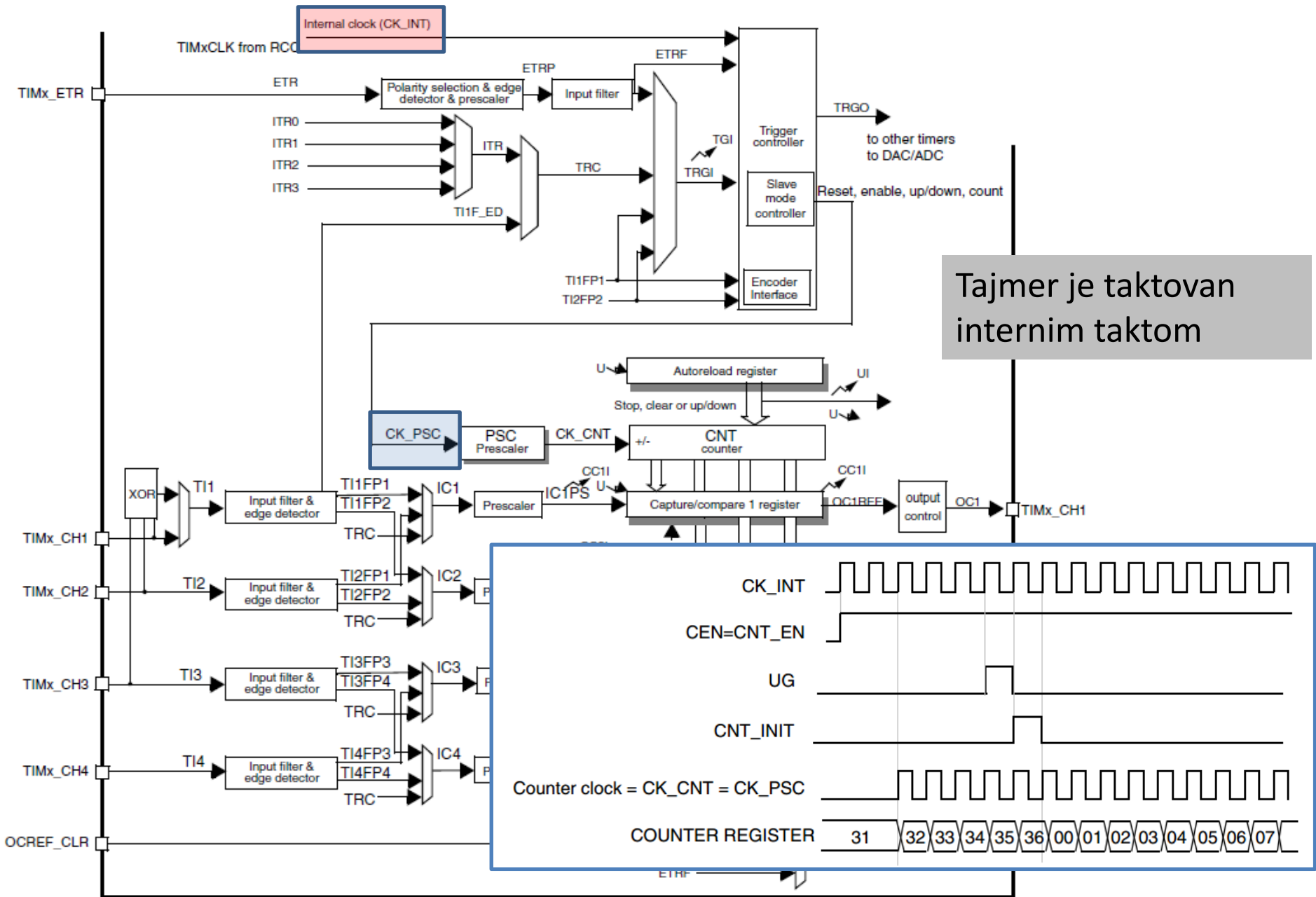
Brojač gore-dole (Center-aligned / Up-down counting mode)



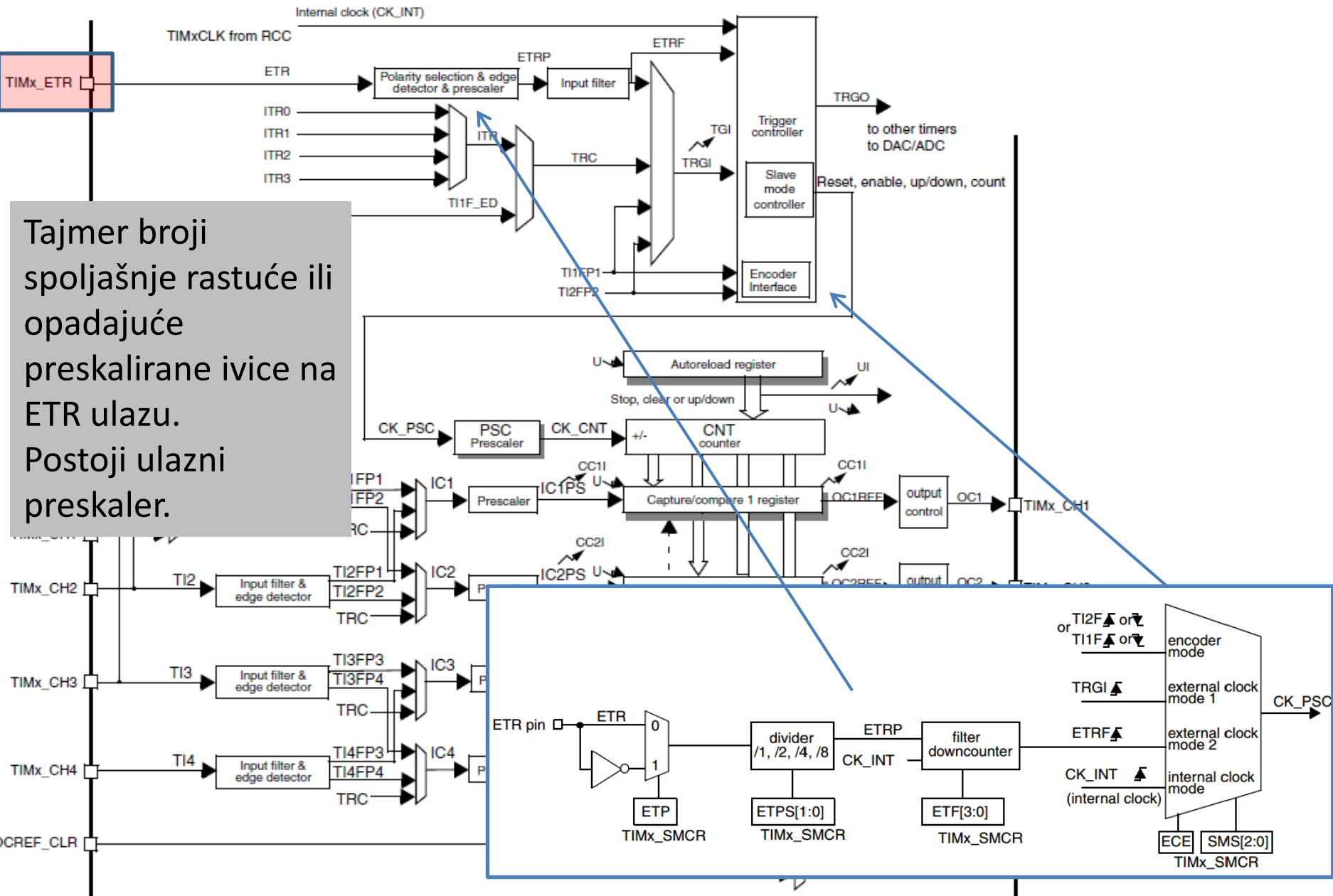
Counter timing diagram, internal clock divided by 1, TIMx_ARR=0x



Taktovanje - Internal mode

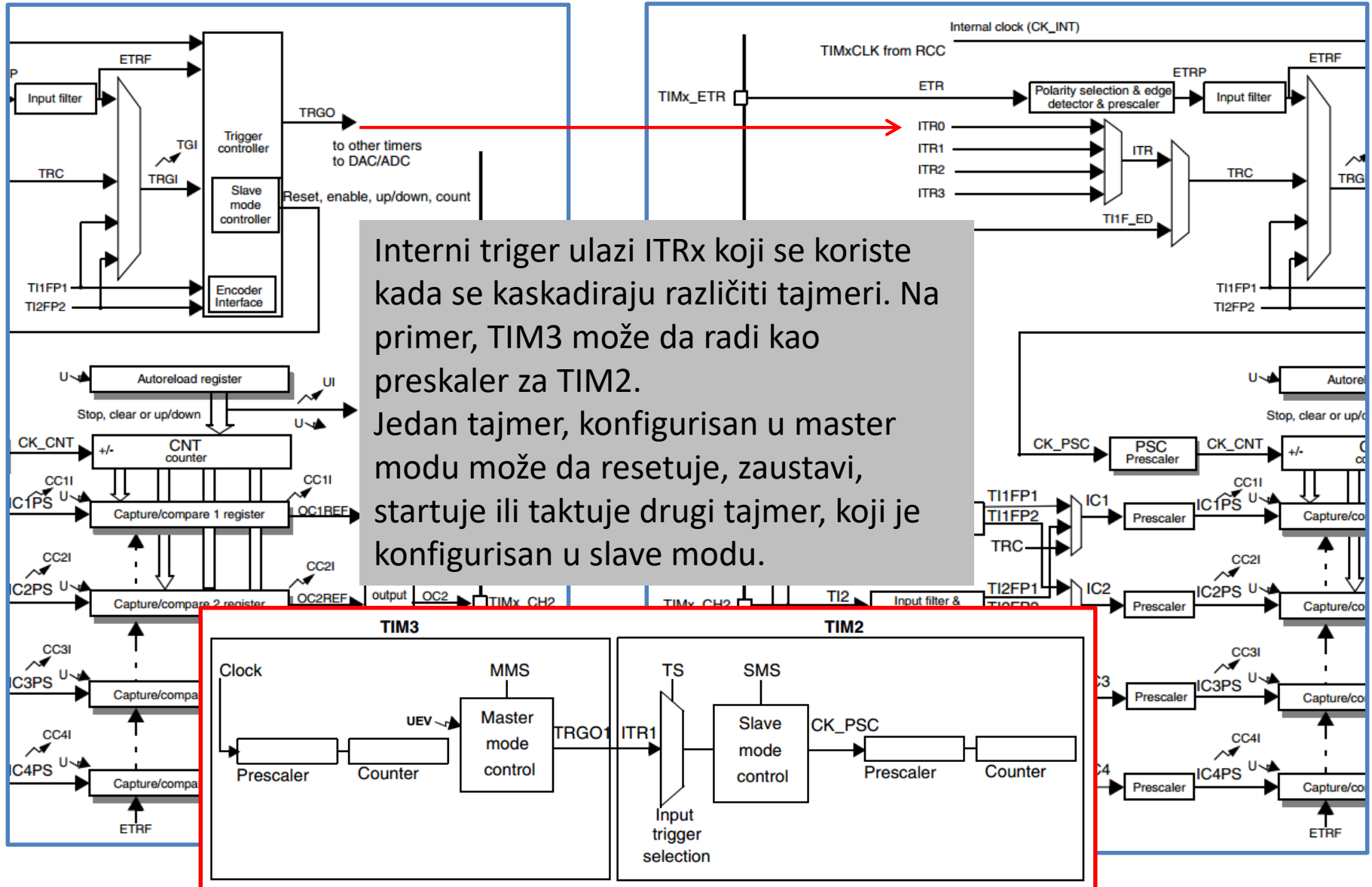


External mode 2



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

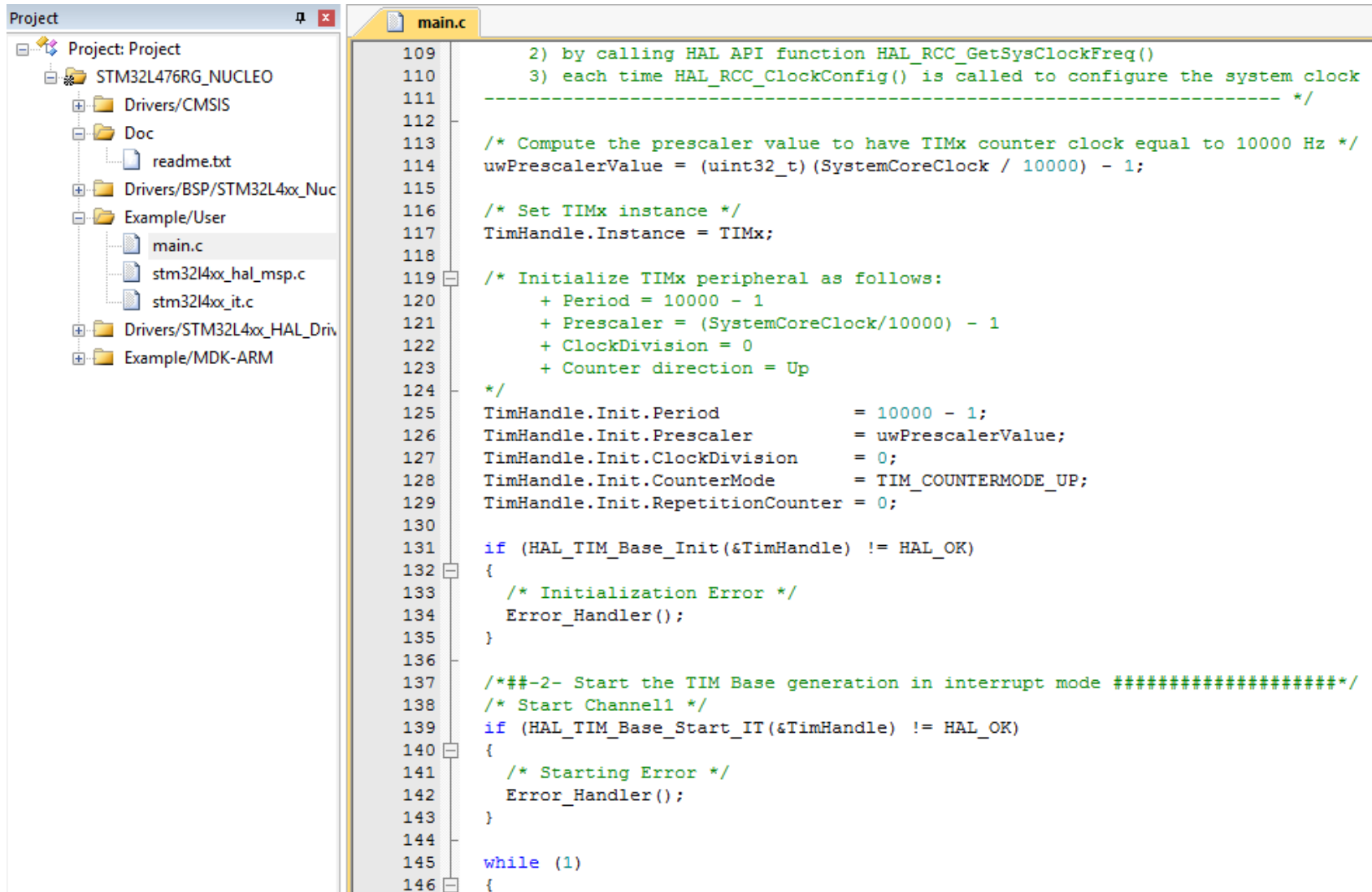
Kaskadna veza tajmera



Interni triger ulazi ITRx koji se koriste kada se kaskadiraju različiti tajmeri. Na primer, TIM3 može da radi kao prescaler za TIM2. Jedan tajmer, konfigurisan u master modu može da resetuje, zaustavi, startuje ili taktuje drugi tajmer, koji je konfigurisan u slave modu.

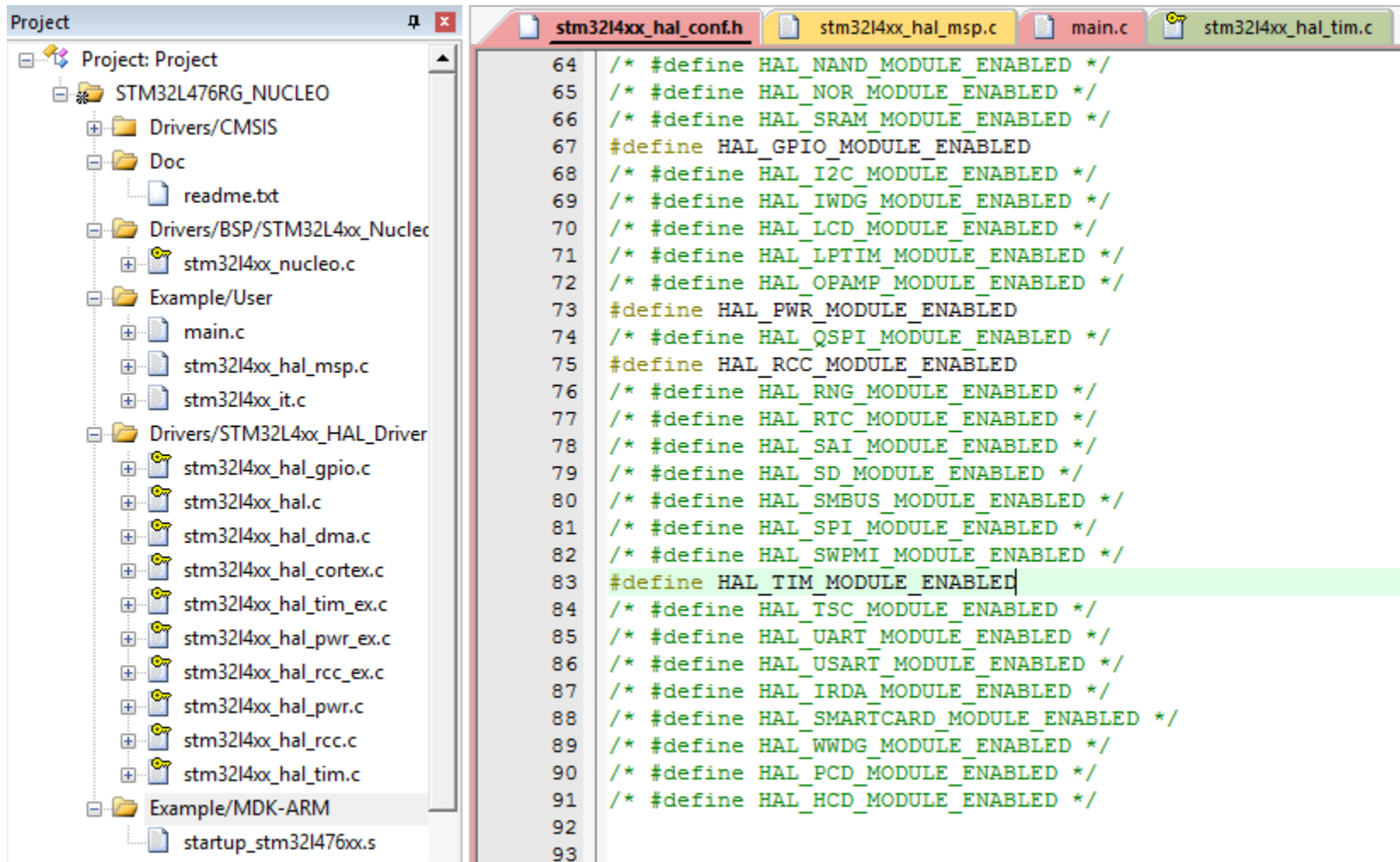
STM CUBE

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 for 'Project: Project'. The tree structure includes:

- 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 'stm32l4xx_hal_conf.h' file is open, showing a list of HAL module enablement macros. The macro `#define HAL_TIM_MODULE_ENABLED` on line 83 is highlighted in green.

```
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 features:
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()
```

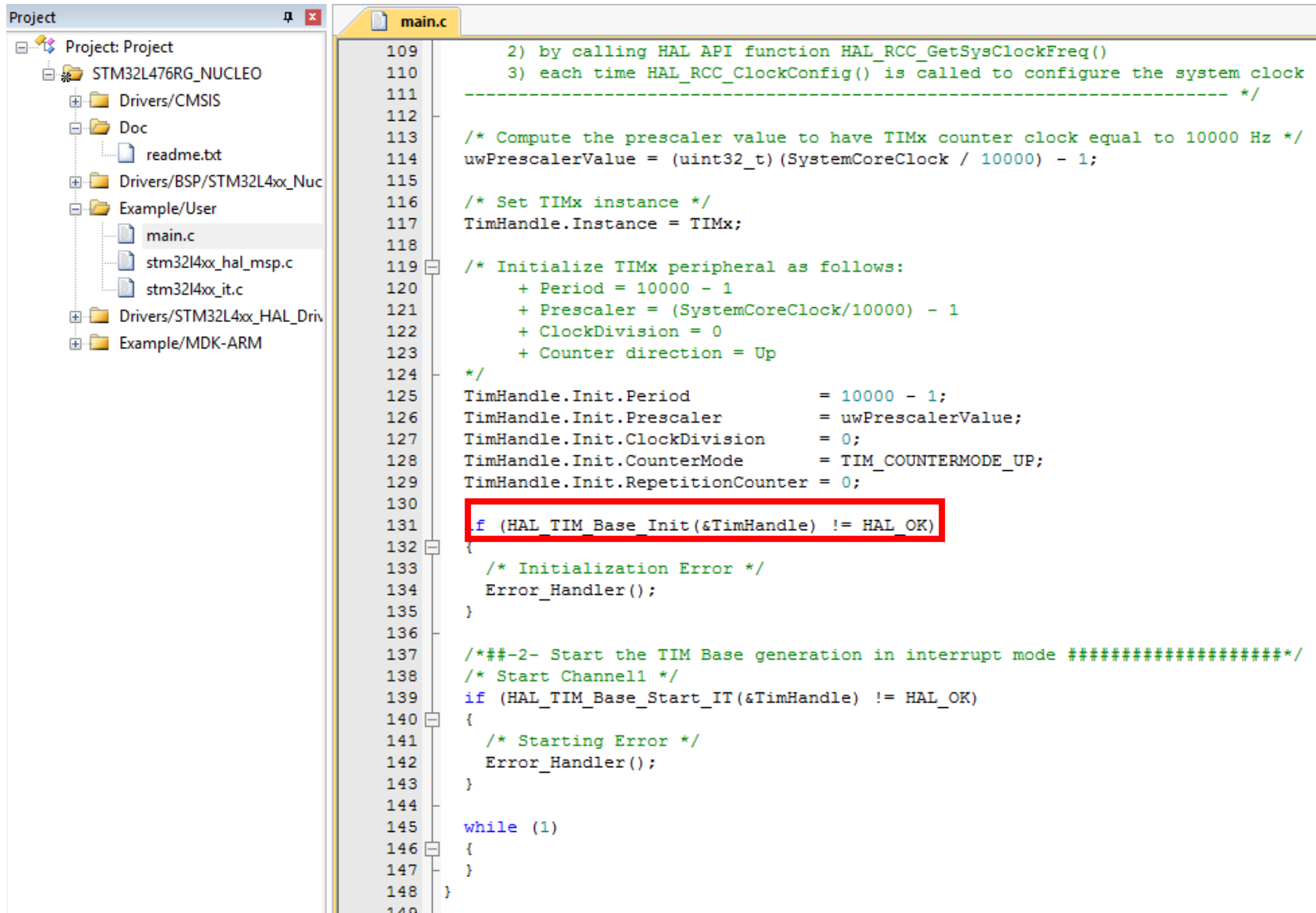


```

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);
  }

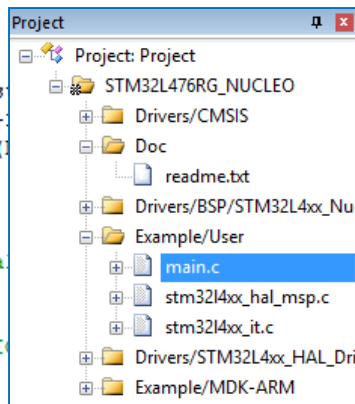
  /* Set the TIM state */
  htim->State= HAL_TIM_STATE_BUSY;

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

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

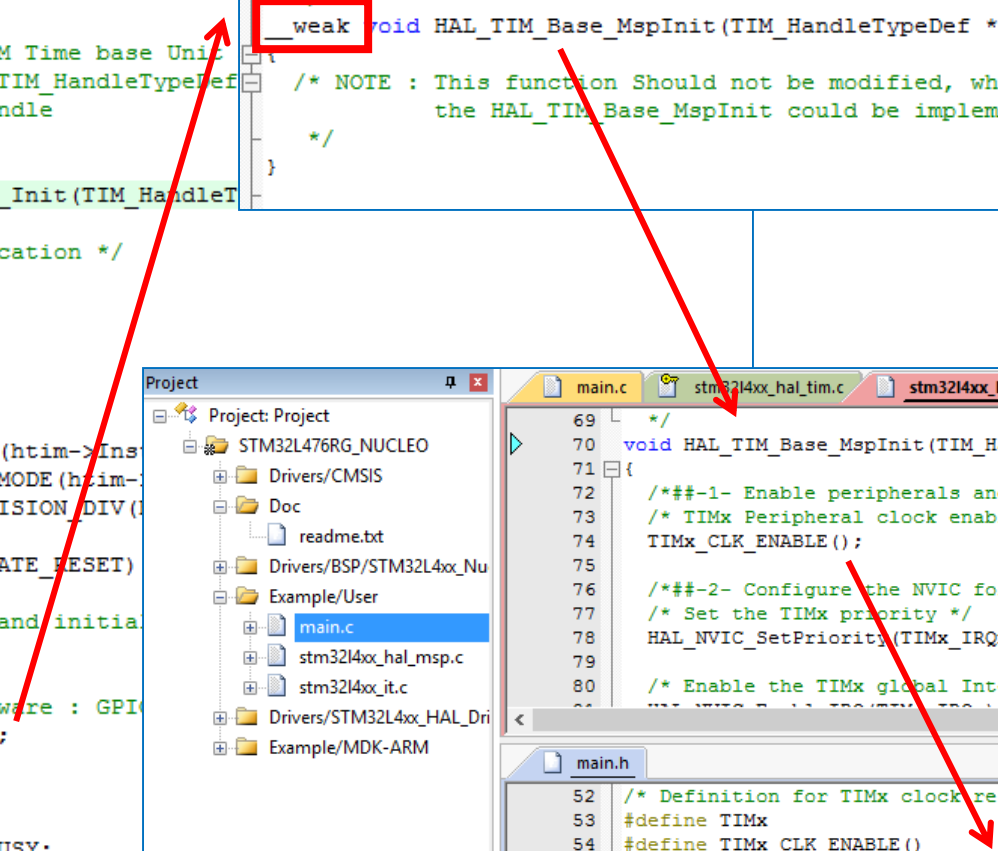
  return HAL_OK;
}
```

```
n.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   TIMx->CR1 |= TIM_CR1_CEN;
82 }
```

```
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 */
```

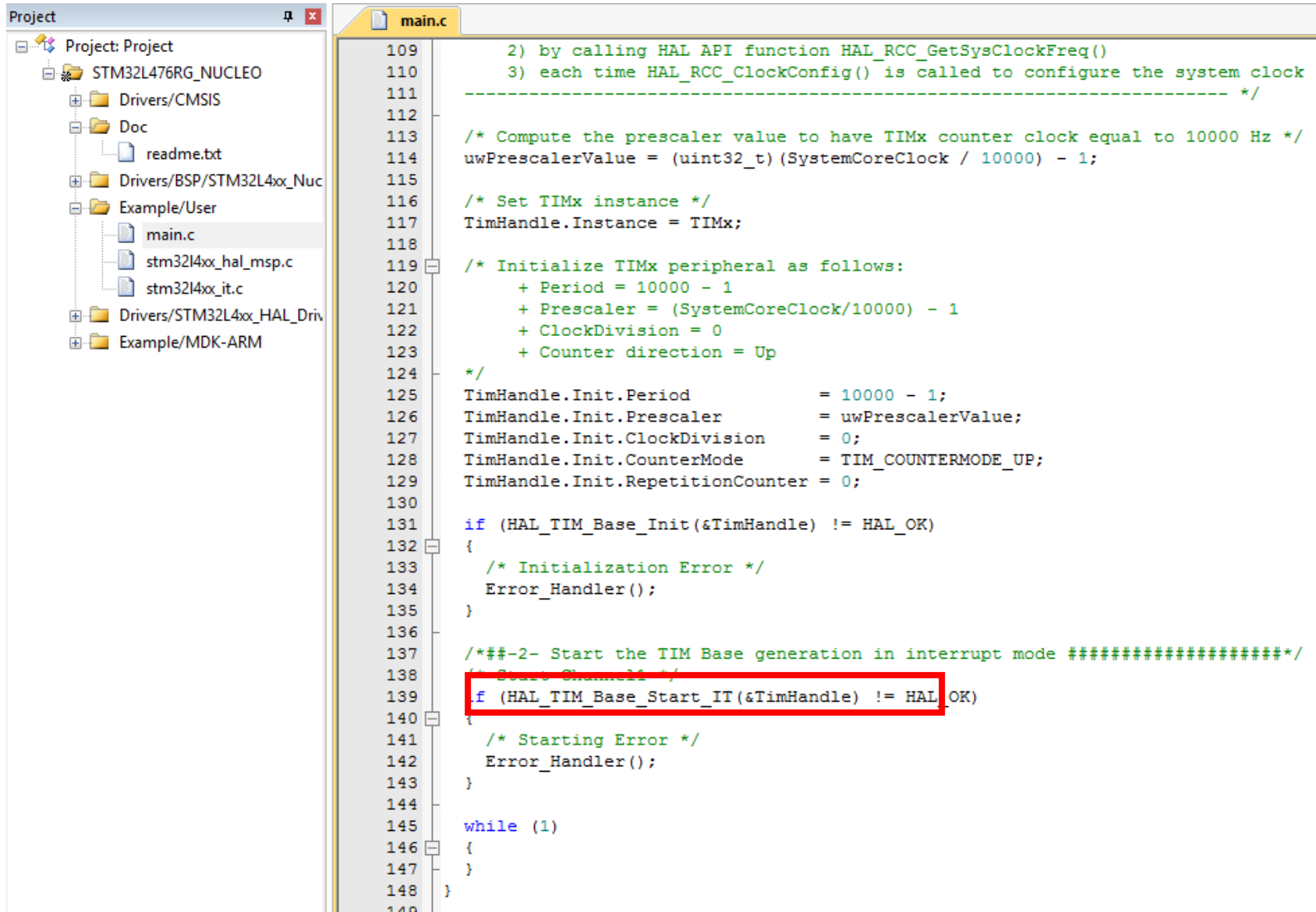


MspInit()

- MSP – MCU Support Package
- 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 where the timer is started in interrupt mode.

```
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 shows an IDE window with a project tree on the left and three code files open on the right. The project tree is titled 'Project' and contains 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

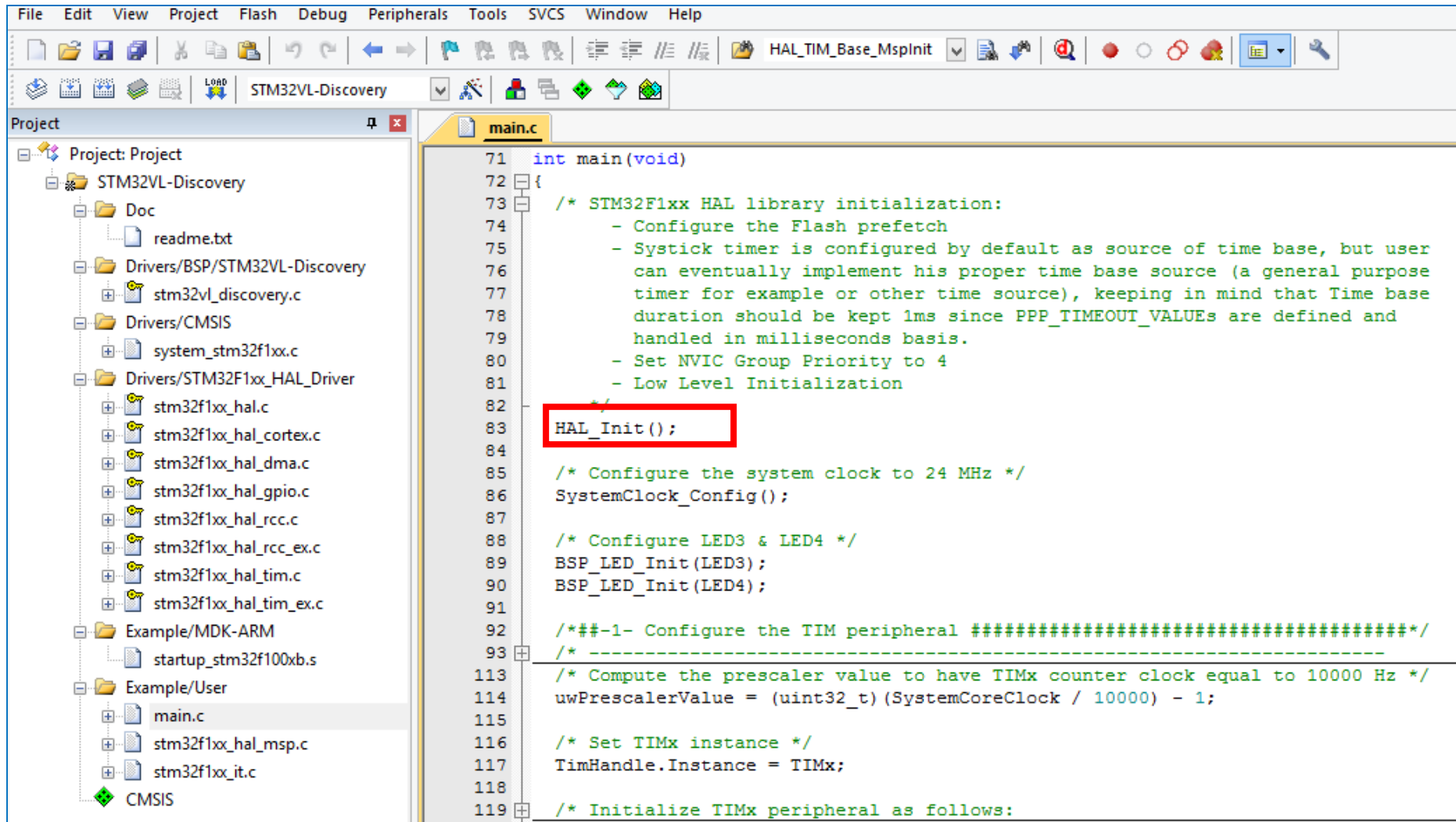
The three code files are:

```
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
160 /**
```

Generalna Inicijalizacija hardvera



The image shows a screenshot of an IDE (Integrated Development Environment) with a project named "STM32VL-Discovery". The main window displays the source code for "main.c". The code is in C and shows the initialization of the HAL (Hardware Abstraction Layer) library. The function `HAL_Init();` is highlighted with a red box. The code includes comments in green and function calls in black. The project structure is visible in the left sidebar, showing folders for "Doc", "Drivers/BSP/STM32VL-Discovery", "Drivers/CMSIS", "Drivers/STM32F1xx_HAL_Driver", "Example/MDK-ARM", and "Example/User".

```
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_0);

    /* 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_Mspltit()` koja inicijalizuje low-level hardverske resurse.
3. Ponekad je potrebno inicijalizovati i neke posebne delove periferije posebnim 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....

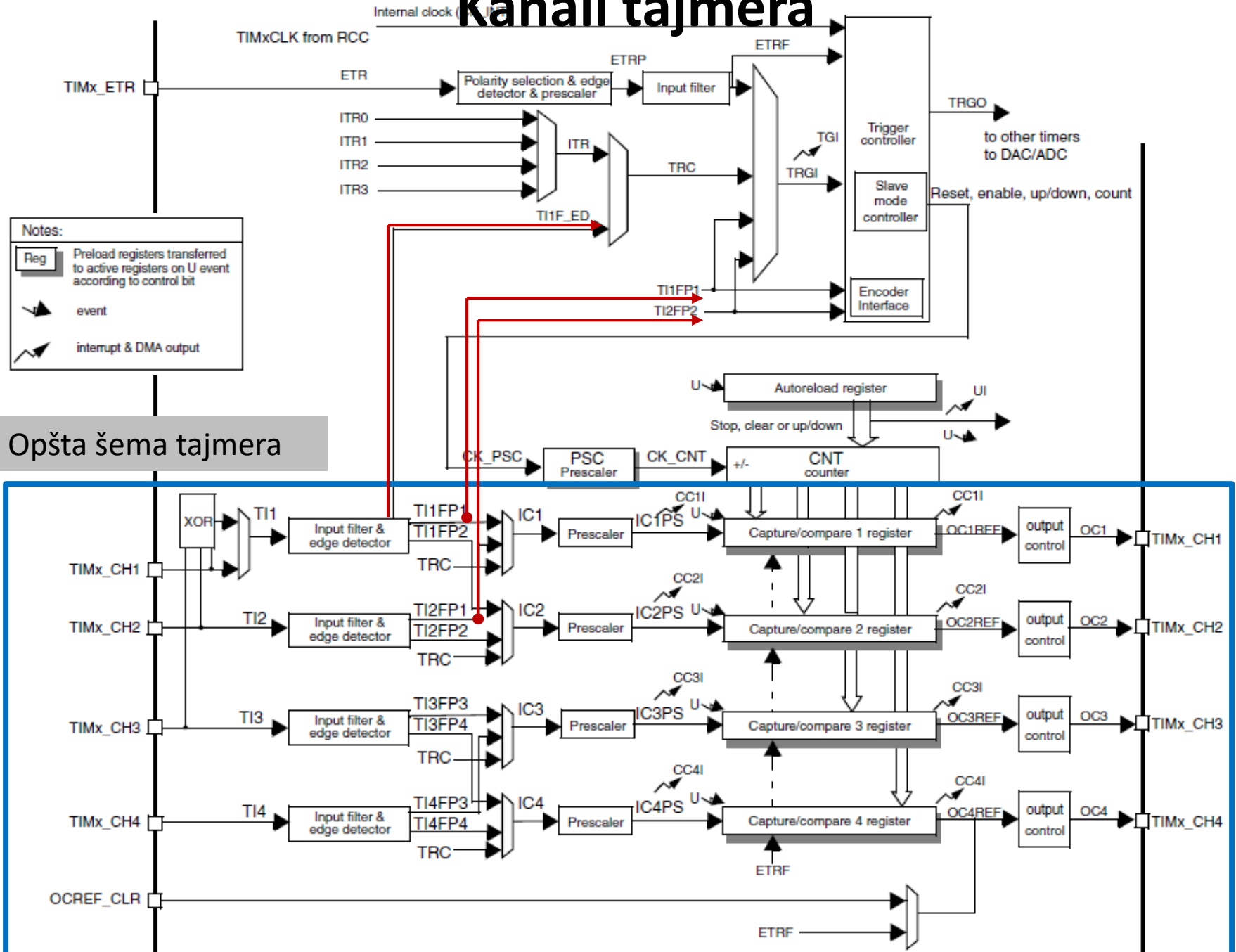
- Hint:

<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
```

Kanali tajmera



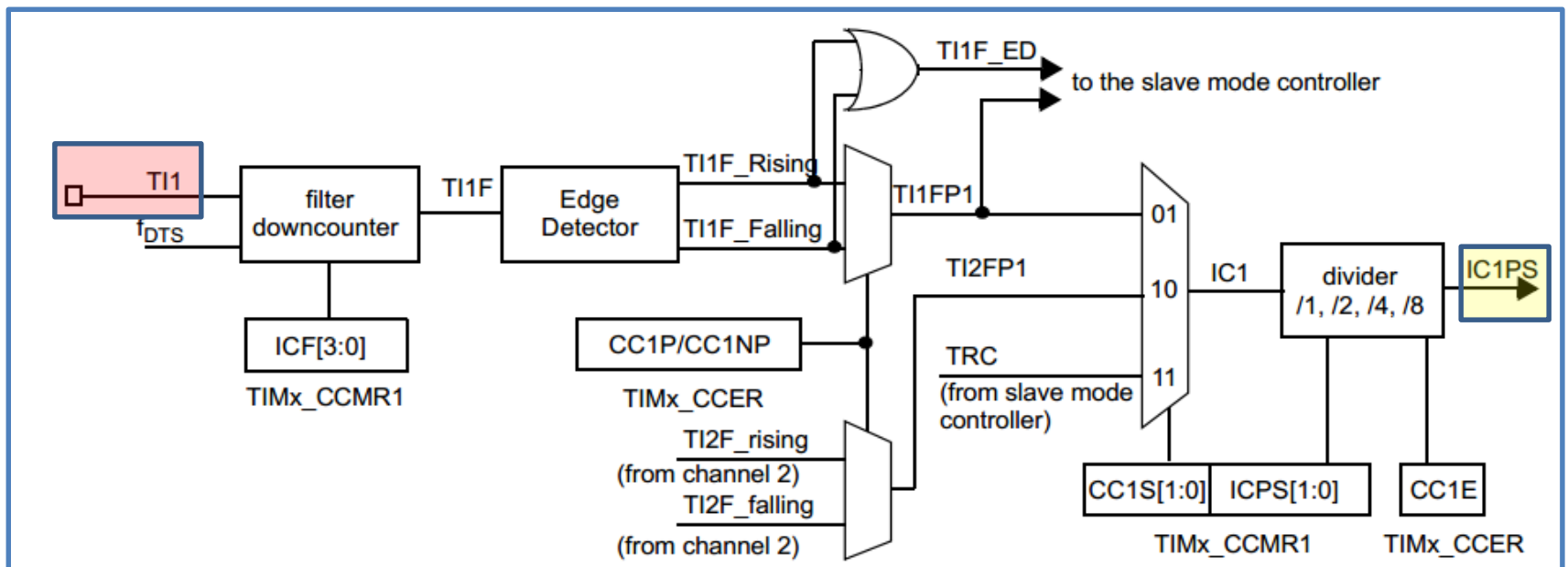
Notes:

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

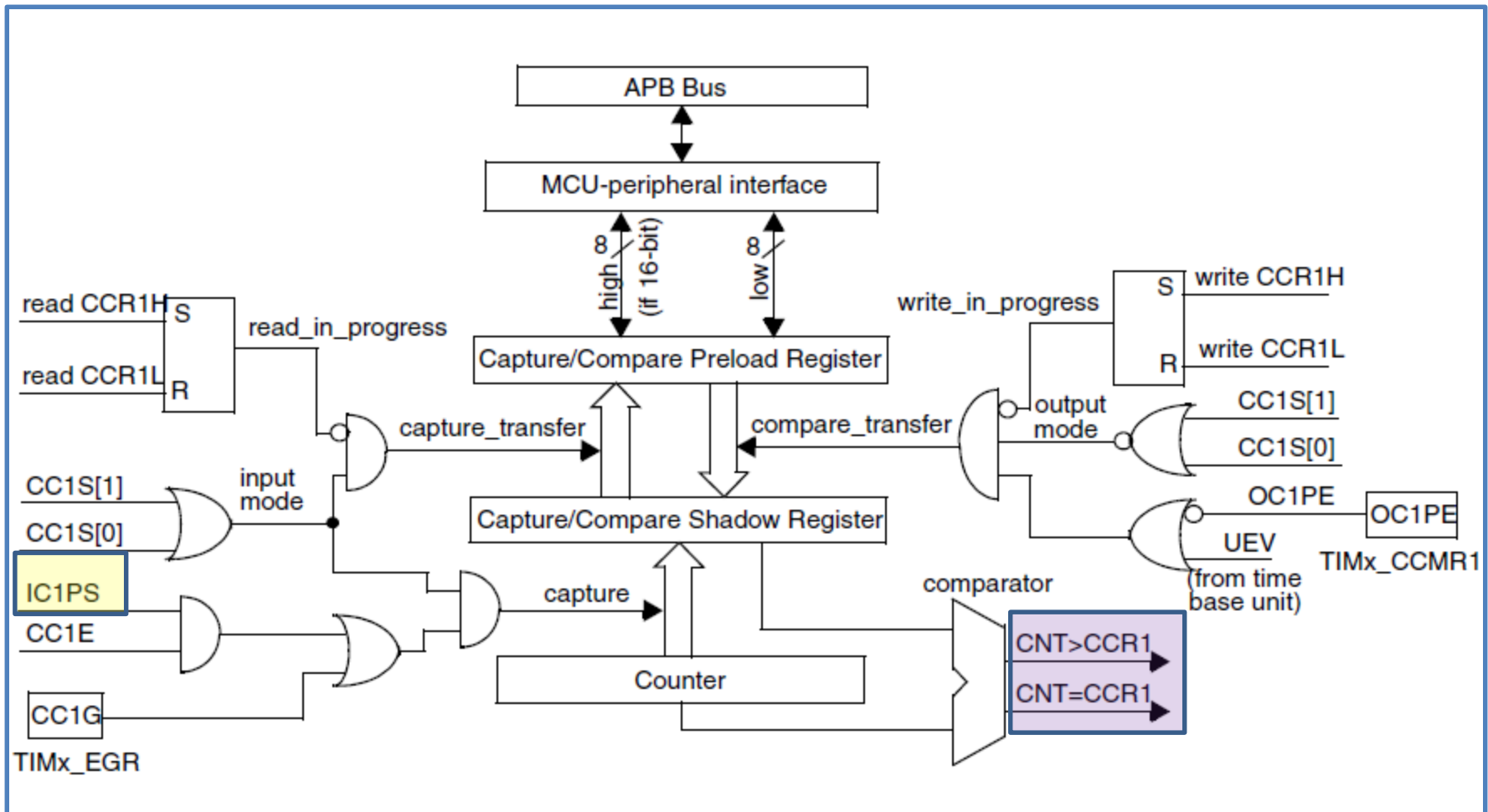
Opšta šema tajmera

Capture/compare jedinica input capture deo

- Svaki capture događaj može da generiše prekid ili DMA zahtev.

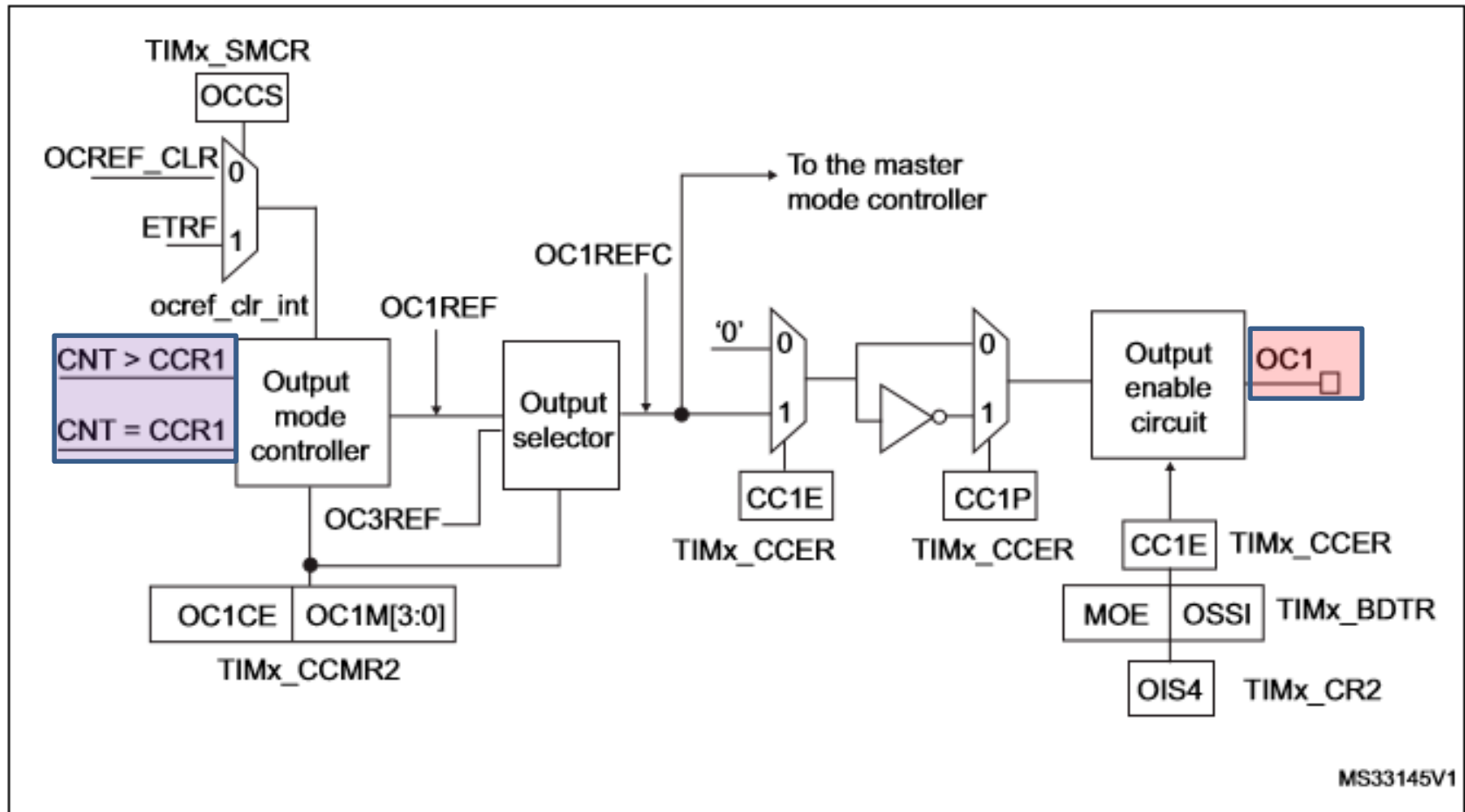


Capture/compare jedinica centralni deo

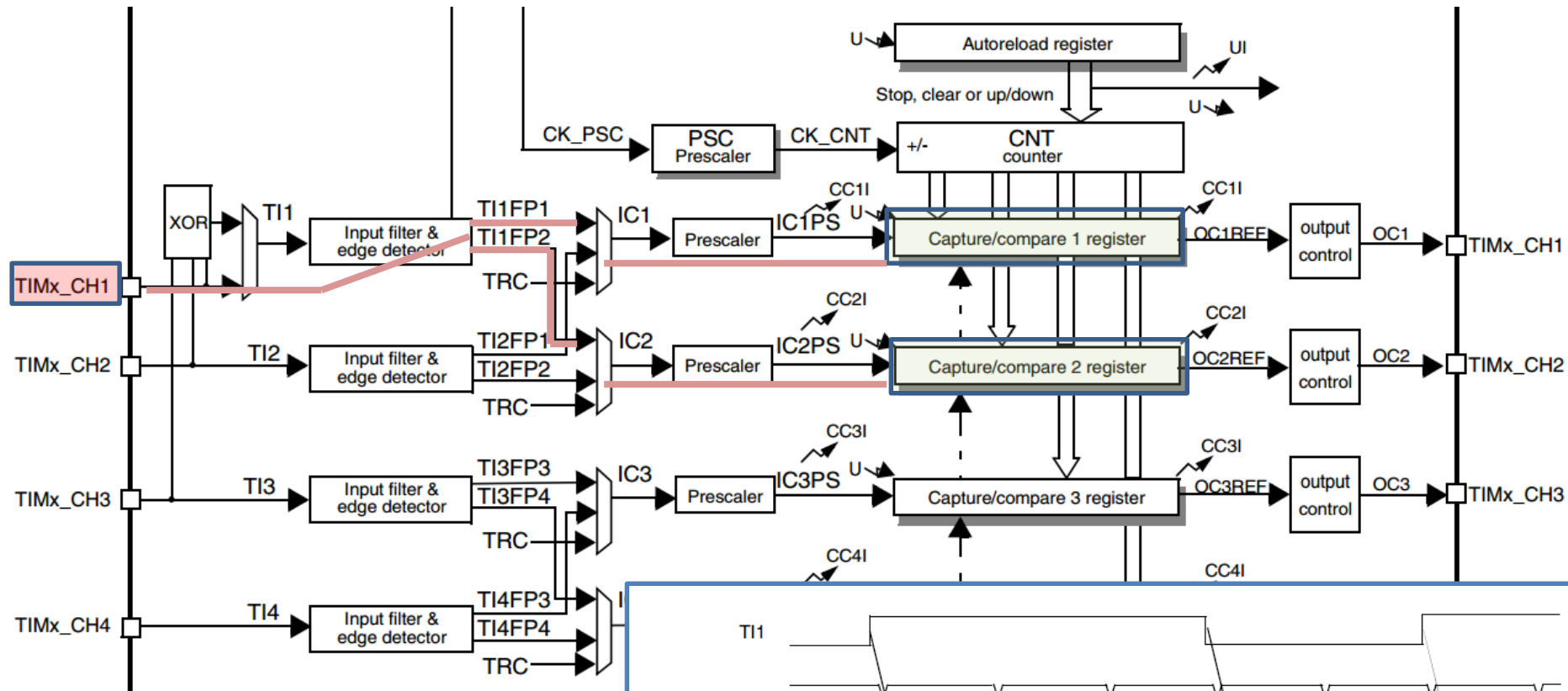


Capture/compare jedinica output compare deo

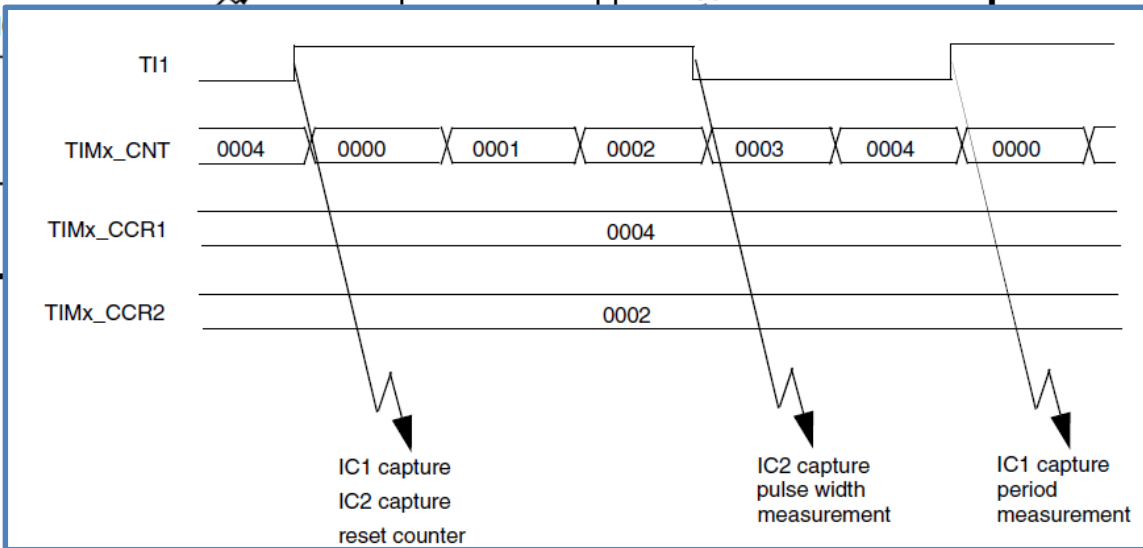
Figure 299. Output stage of capture/compare channel (channel 1)



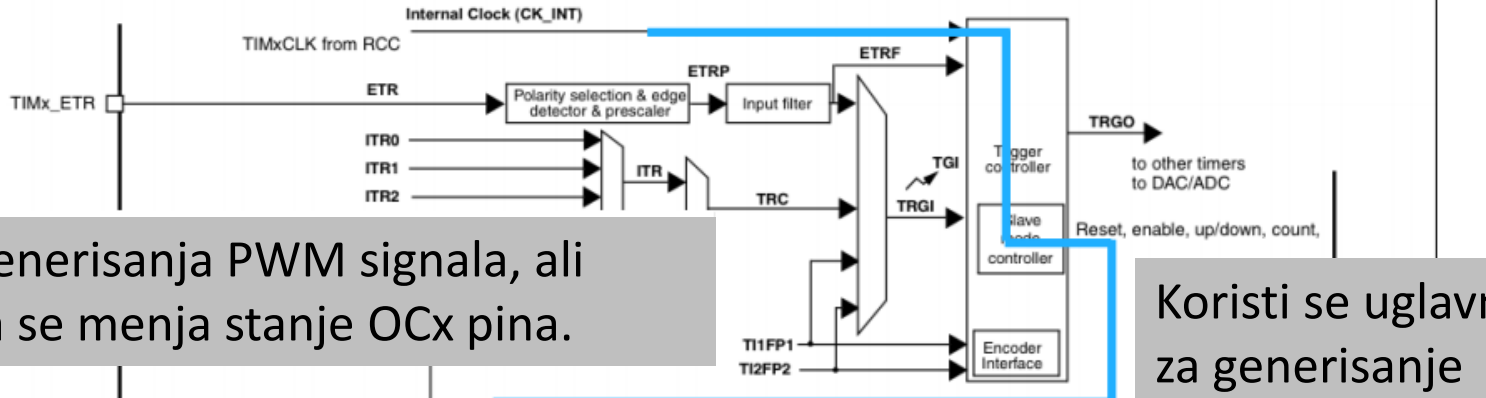
PWM input capture



Posebna input capture konfiguracija u kojoj se kombinuju dva kanala tako da može da se meri perioda i duty-cycle pwm signala na ulazu.

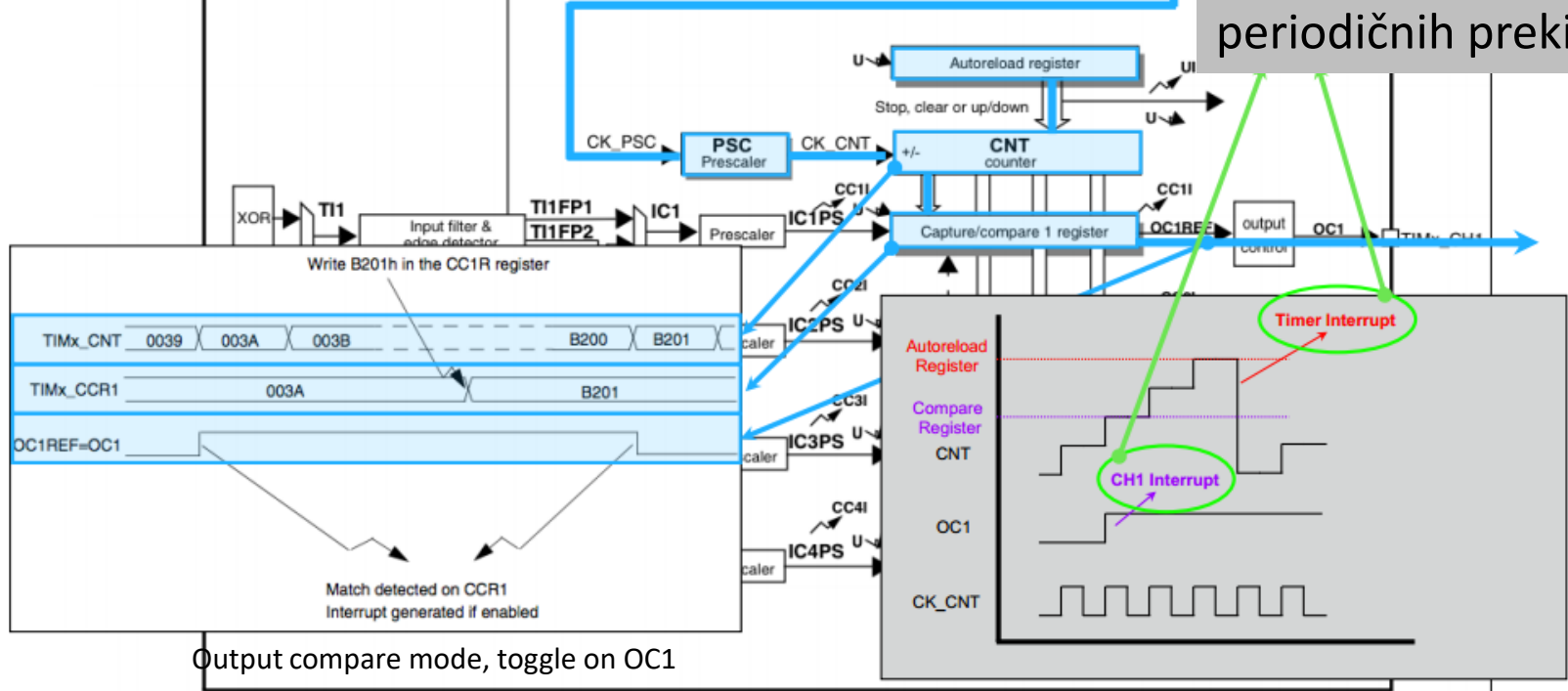


Output compare mod



Nema generisanja PWM signala, ali može da se menja stanje OCx pina.

Koristi se uglavnom za generisanje periodičnih prekida.



OCx MODE – šta generišemo na izlazu na osnovu OCxREF signala

RM0351

Advanced-control timers (TIM1/TIM8)

Bits 6:4 **OC1M**: Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

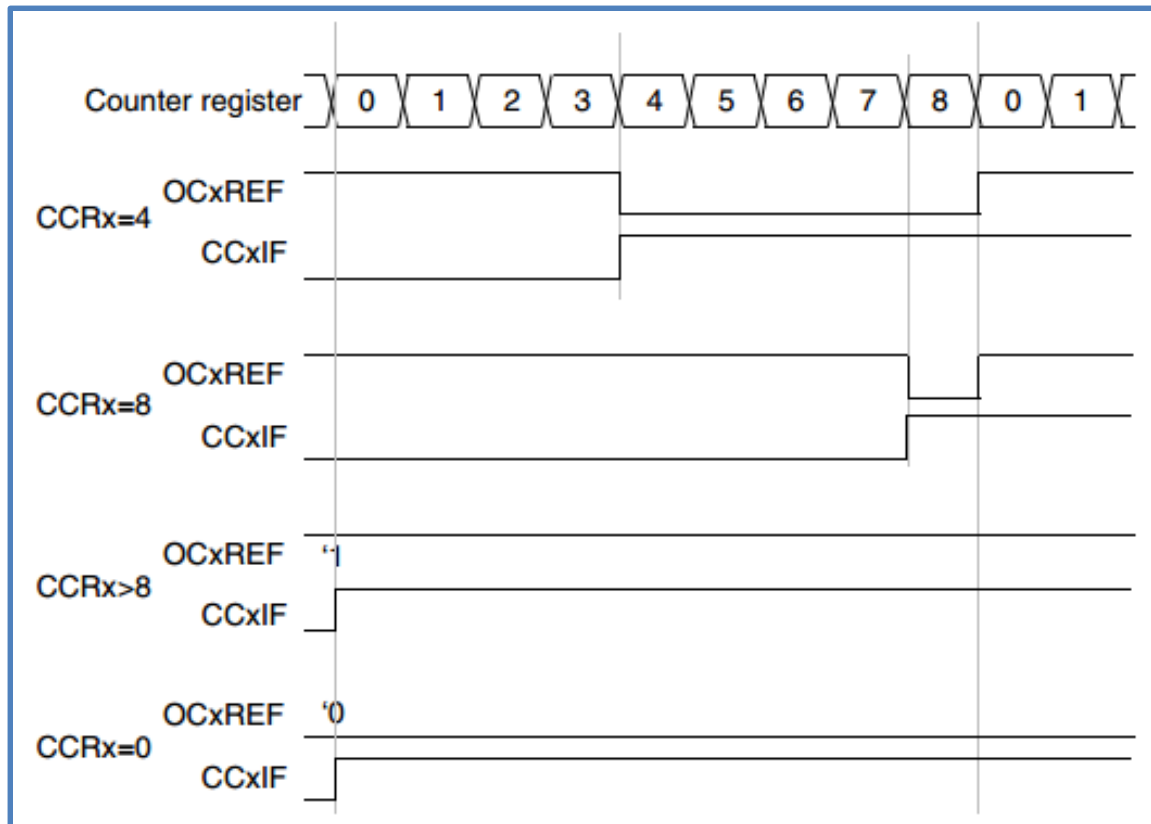
0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF='0') as long as TIMx_CNT>TIMx_CCR1 else active (OC1REF='1').

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

1000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is

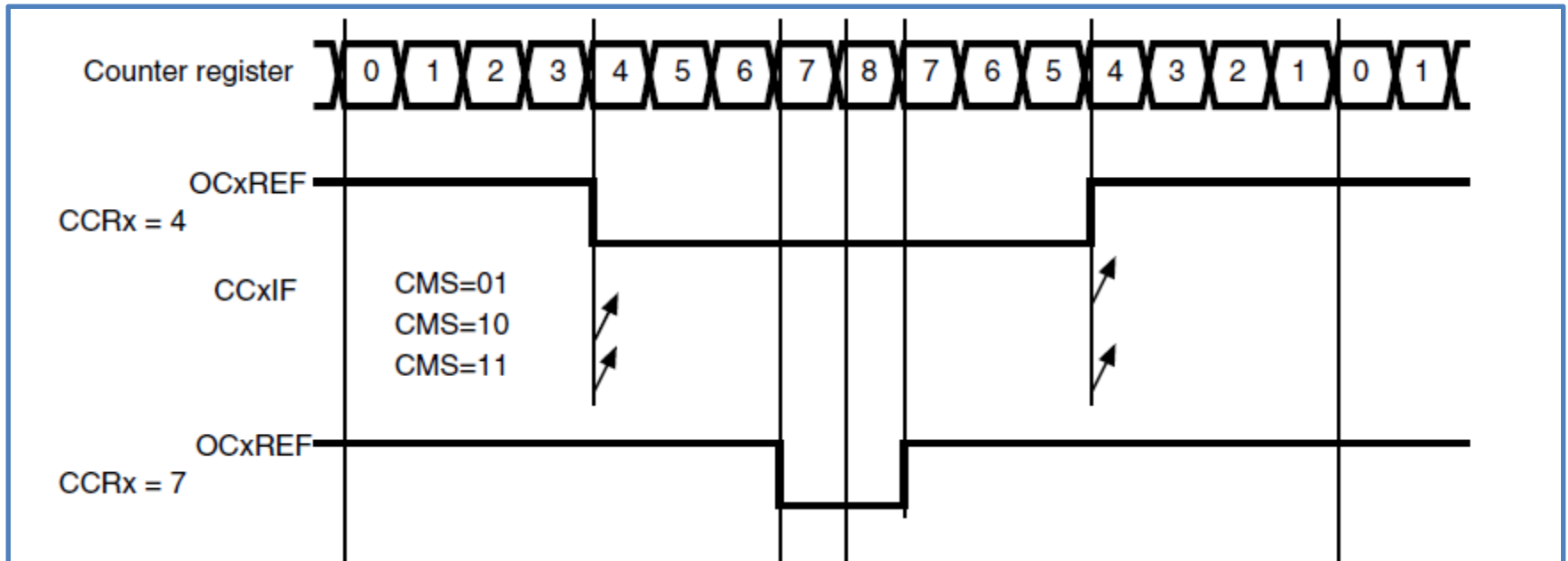
Edge-aligned PWM

- U ovom modu brojač može da radi ili u UP ili u DOWN modu.
- U primeru je podešeno da signal OCxREF bude aktivan dok god je $TIMx_CNT < TIMx_CCRx$ (PWM1 mod), u suprotnom pada na nizak nivo.



Center-aligned PWM

- Slično onome što se kod MSP-a zove phase-correct PWM mod.
- Brojač radi u up/down modu, OCxM je PWM1.



MBED

Klasa PwmOut

PwmOut Class Reference

```
#include <PwmOut.h>
```

Public Member Functions

[PwmOut](#) (PinName pin)

Create a **PwmOut** connected to the specified pin.

void [write](#) (float value)

Set the output duty-cycle, specified as a percentage (float)

float [read](#) ()

Return the current output duty-cycle setting, measured as a percentage (float)

void [period](#) (float seconds)

Set the PWM period, specified in seconds (float), keeping the duty cycle the same.

void [period_ms](#) (int ms)

Set the PWM period, specified in milli-seconds (int), keeping the duty cycle the same.

void [period_us](#) (int us)

Set the PWM period, specified in micro-seconds (int), keeping the duty cycle the same.

void [pulsewidth](#) (float seconds)

Set the PWM pulsewidth, specified in seconds (float), keeping the period the same.

void [pulsewidth_ms](#) (int ms)

Set the PWM pulsewidth, specified in milli-seconds (int), keeping the period the same.

void [pulsewidth_us](#) (int us)

Set the PWM pulsewidth, specified in micro-seconds (int), keeping the period the same.

PwmOut & [operator=](#) (float value)

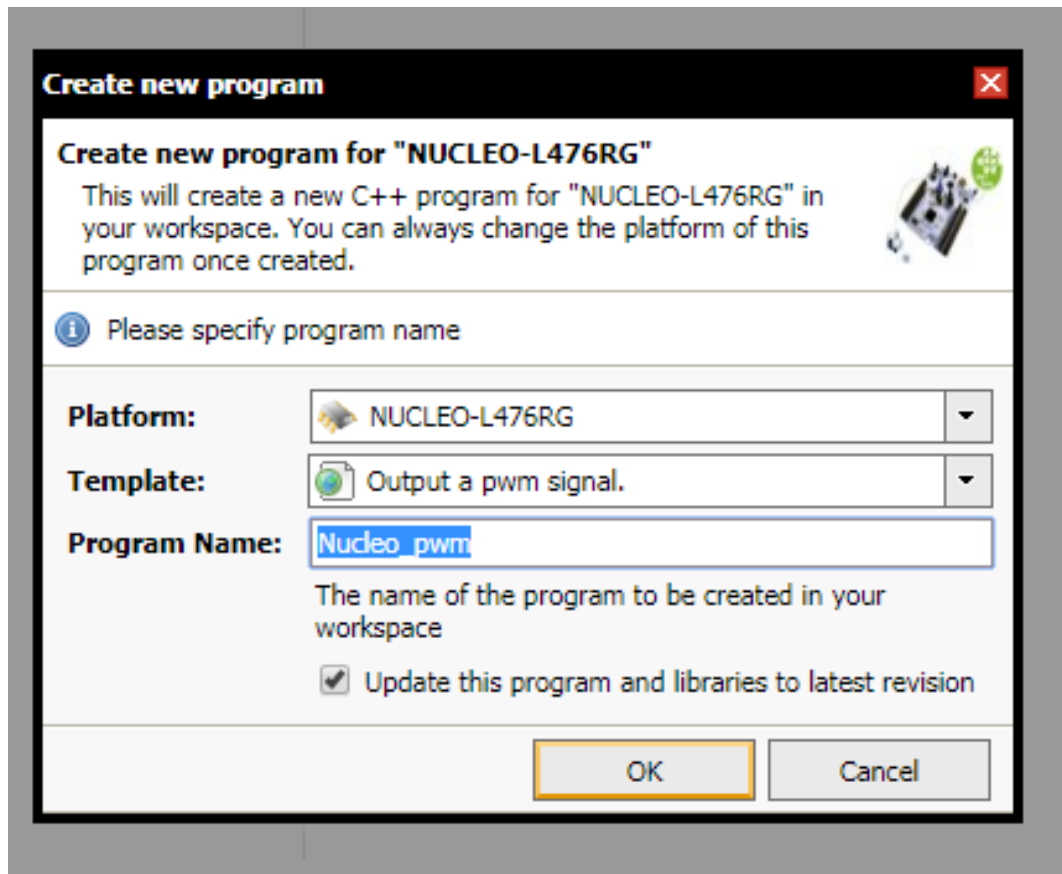
A operator shorthand for [write\(\)](#)

[operator float](#) ()

An operator shorthand for [read\(\)](#)

MBED

Template: Output a pwm signal



Create new program

Create new program for "NUCLEO-L476RG"

This will create a new C++ program for "NUCLEO-L476RG" in your workspace. You can always change the platform of this program once created.

Please specify program name

Platform: NUCLEO-L476RG

Template: Output a pwm signal.

Program Name: Nucleo_pwm

The name of the program to be created in your workspace

Update this program and libraries to latest revision

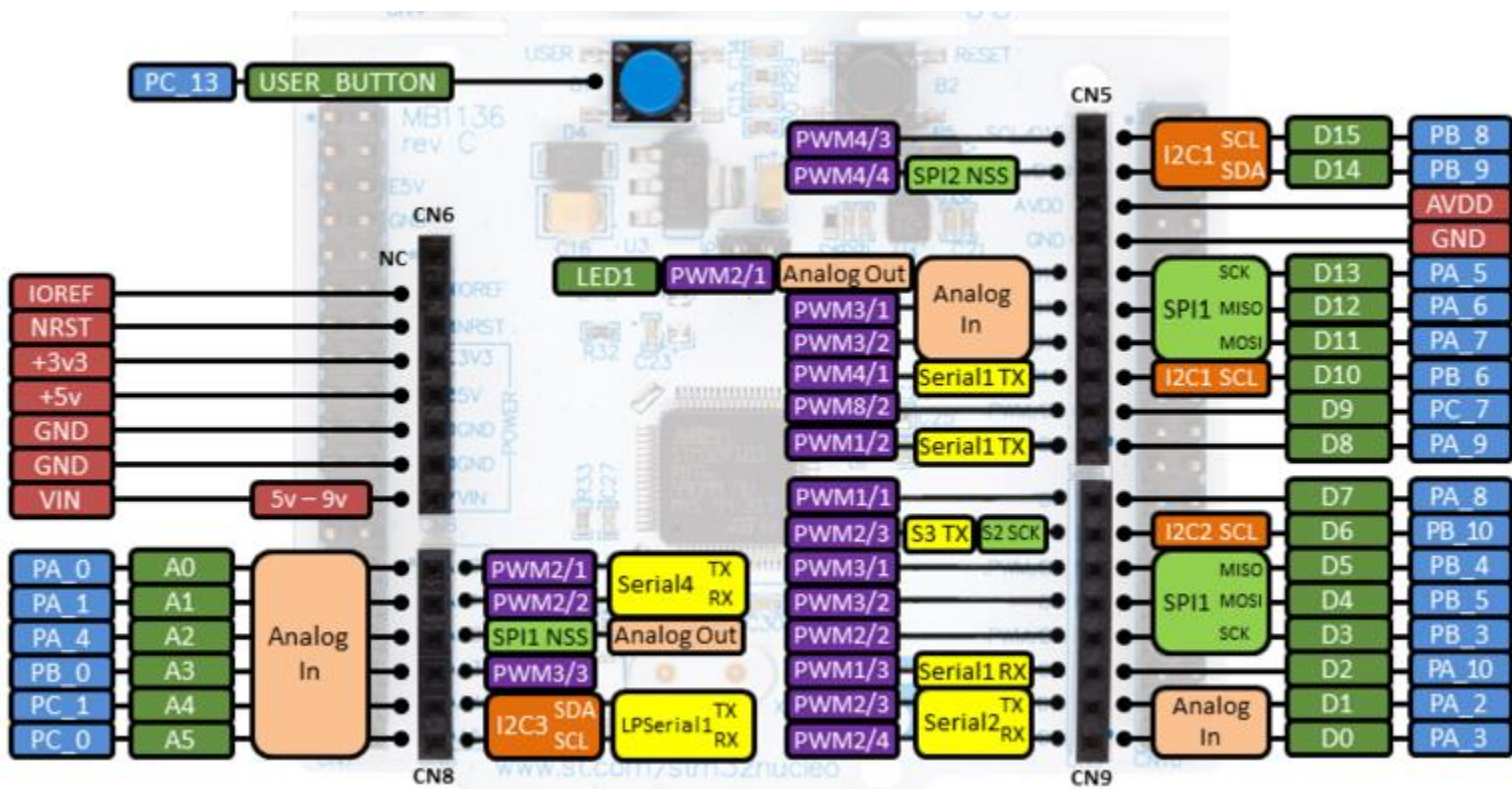
OK Cancel

MBED

Template: Output a pwm signal

- Na kom pinu se generiše PWM signal?
- Kako da proverimo da li je generisanje ispravno?

Na kojim to pinovima može da se generiše PWM



MBED

Digitalni izlazi sa mogućnošću generisanja PWM signala

- Pristup preko MBED biblioteke je brutalno jednostavan i intuitivan.

```
#include "mbed.h"
PwmOut led(LED2);
int main() {
    while(1) {
        for(float p = 0.0f; p < 1.0f; p += 0.1f) {
            led = p; wait(0.1);
        }
    }
}
```

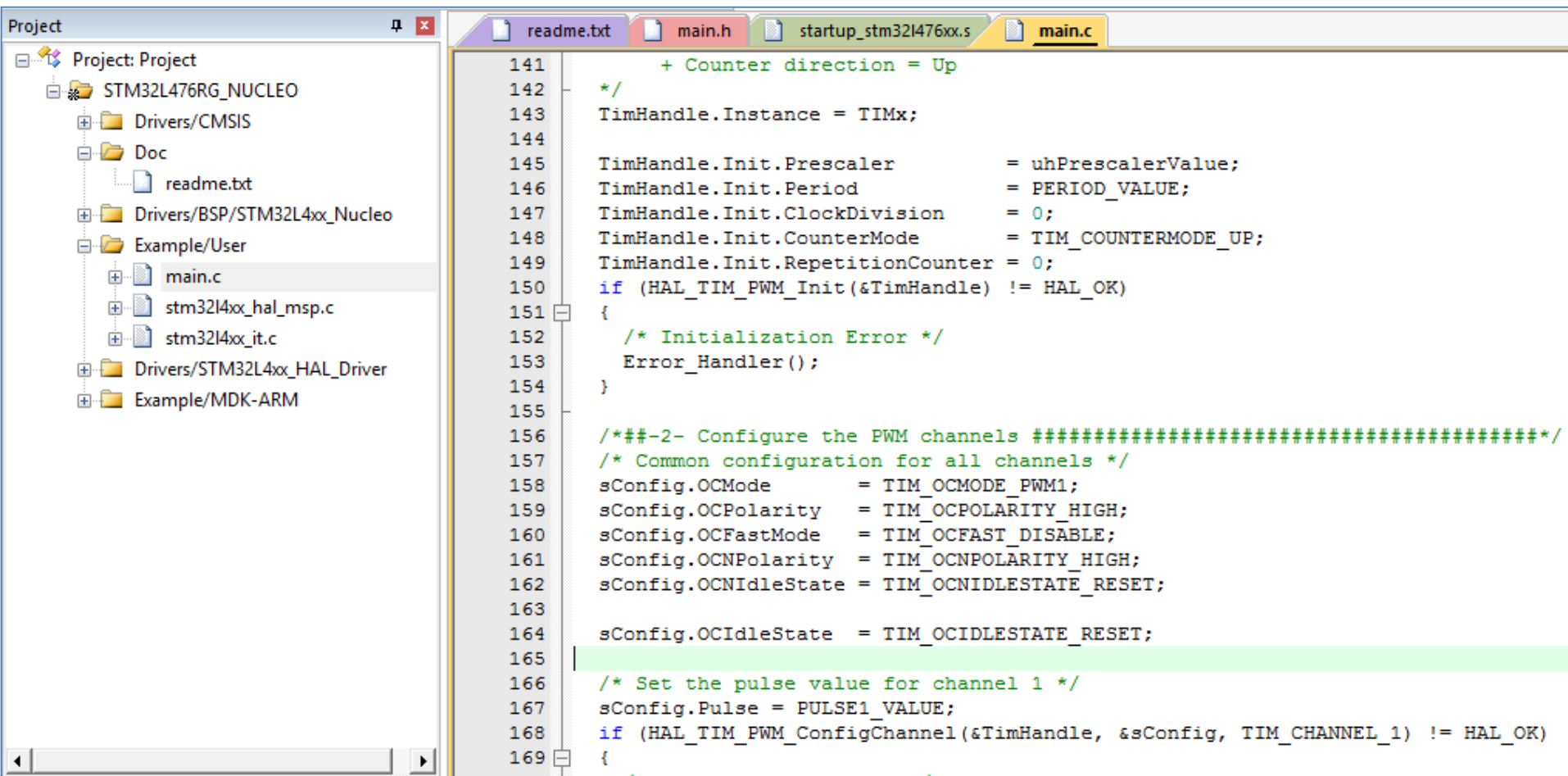
Zadatak

- Generisati PWM izlaz na pinu LED-a.
- Obezbediti da se na pritisak tastera (USER BUTTON) inkrementira duty ratio za 10% pri svakom pritisku.

STM CUBE

Projekat TIM_PWMOutput

- Generisanje PWM signala na 4 OC kanala sa različitim duty cycle-om.



The image shows a screenshot of an IDE (likely STM32CubeIDE) with a project structure on the left and C code in the main editor. The project structure includes folders for Drivers/CMSIS, Doc, Drivers/BSP/STM32L4xx_Nucleo, Example/User, Drivers/STM32L4xx_HAL_Driver, and Example/MDK-ARM. The main editor displays the following C code:

```
141     + Counter direction = Up
142     */
143     TimHandle.Instance = TIMx;
144
145     TimHandle.Init.Prescaler      = uhPrescalerValue;
146     TimHandle.Init.Period        = PERIOD_VALUE;
147     TimHandle.Init.ClockDivision = 0;
148     TimHandle.Init.CounterMode   = TIM_COUNTERMODE_UP;
149     TimHandle.Init.RepetitionCounter = 0;
150     if (HAL_TIM_PWM_Init(&TimHandle) != HAL_OK)
151     {
152         /* Initialization Error */
153         Error_Handler();
154     }
155
156     /*##-2- Configure the PWM channels #####*/
157     /* Common configuration for all channels */
158     sConfig.OCMode          = TIM_OCMODE_PWM1;
159     sConfig.OCpolarity      = TIM_OCPOLARITY_HIGH;
160     sConfig.OCFastMode      = TIM_OCFAST_DISABLE;
161     sConfig.OCNPolarity     = TIM_OCNPOLARITY_HIGH;
162     sConfig.OCNIdleState    = TIM_OCNIDLESTATE_RESET;
163
164     sConfig.OCIdleState     = TIM_OCIDLESTATE_RESET;
165
166     /* Set the pulse value for channel 1 */
167     sConfig.Pulse = PULSE1_VALUE;
168     if (HAL_TIM_PWM_ConfigChannel(&TimHandle, &sConfig, TIM_CHANNEL_1) != HAL_OK)
169     {
```

Struktura TIM_OC_InitTypeDef

```
readme.txt main.h startup_stm32i476xx.s main.c stm32i4xx_hal_tim.h
46 /** @addtogroup TIM_PWMOutput
47     * @{
48     */
49
50 /* Private typedef -----*/
51 #define PERIOD_VALUE          (uint32_t)(666 - 1) /* Period Value */
52 #define PULSE1_VALUE          (uint32_t)(PERIOD_VALUE/2) /* Capture Compare 1 Value */
53 #define PULSE2_VALUE          (uint32_t)(PERIOD_VALUE*37.5/100) /* Capture Compare 2 Value */
54 #define PULSE3_VALUE          (uint32_t)(PERIOD_VALUE/4) /* Capture Compare 3 Value */
55 #define PULSE4_VALUE          (uint32_t)(PERIOD_VALUE*12.5/100) /* Capture Compare 4 Value */
56
57 /* Private define -----*/
58 /* Private macro -----*/
59 /* Private variables -----*/
60 /* Timer handler declaration */
61 TIM_HandleTypeDef TimHandle;
62
63 /* Timer Output Compare Configuration Structure declaration */
64 TIM_OC_InitTypeDef sConfig;
65
66
67
68
69
70
71
72 */
73 typedef struct
74 {
75     uint32_t OCMode; /*!< Specifies the output mode.
76                     This parameter can be a value of @ref TIM_Output_Compare_Mode */
77
78     uint32_t Pulse; /*!< Specifies the pulse width (in counter clock)
79                    This parameter can be a value of @ref TIM_Output_Compare_Pulse */
80
81     uint32_t OCPolarity; /*!< Specifies the output polarity.
82                          This parameter can be a value of @ref TIM_Output_Compare_Polarity */
83
84     uint32_t OCNPolarity; /*!< Specifies the complementary output polarity.
85                           This parameter can be a value of @ref TIM_Output_Compare_N_Polarity
86                           @note This parameter is valid only for TIM1 and TIM8. */
87
88     uint32_t OCFastMode; /*!< Specifies the Fast mode state.
89                          This parameter can be a value of @ref TIM_Output_Fast_State
90                          @note This parameter is valid only in PWM1 and PWM2 mode. */
91
92     uint32_t OCIdleState; /*!< Specifies the TIM Output Compare pin state during Idle state.
93                           This parameter can be a value of @ref TIM_Output_Compare_Idle_State
94                           @note This parameter is valid only for TIM1 and TIM8. */
95
96     uint32_t OCNIdleState; /*!< Specifies the TIM Output Compare pin state during Idle state.
97                             This parameter can be a value of @ref TIM_Output_Compare_N_Idle_State
98                             @note This parameter is valid only for TIM1 and TIM8. */
99
100     TIM_OC_InitTypeDef;
101 };
```

Bits 6:4 **OC1M**: Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF='0') as long as TIMx_CNT>TIMx_CCR1 else active (OC1REF='1').

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

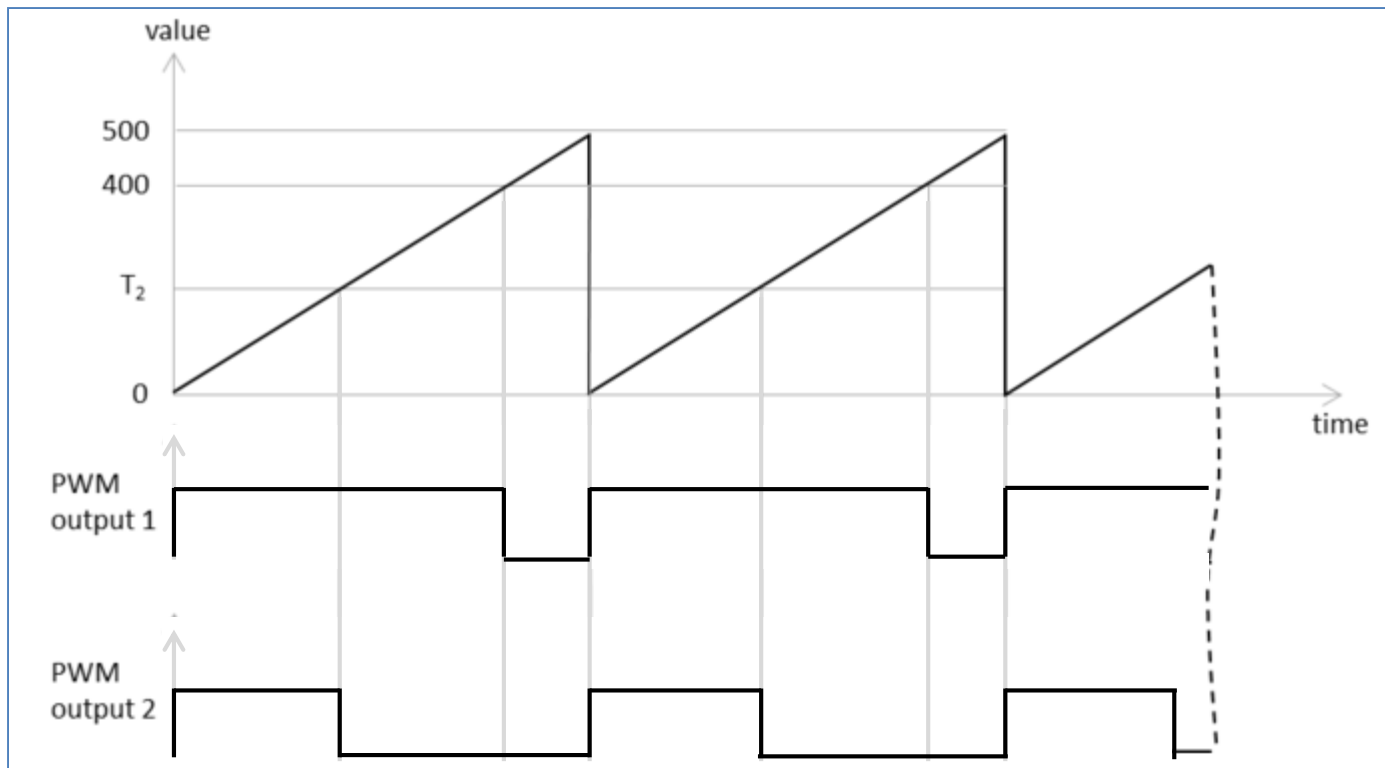
1000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger

```

860 /** @defgroup TIM_Output_Compare_and_PWM_modes TIM Output Compare and PWM Modes
861     * @{
862     */
863 #define TIM_OCMODE_TIMING                ((uint32_t)0x0000)
864 #define TIM_OCMODE_ACTIVE                ((uint32_t)TIM_CCMR1_OC1M_0)
865 #define TIM_OCMODE_INACTIVE              ((uint32_t)TIM_CCMR1_OC1M_1)
866 #define TIM_OCMODE_TOGGLE                ((uint32_t)TIM_CCMR1_OC1M_1 | TIM_CCMR1_OC1M_0)
867 #define TIM_OCMODE_PWM1                  ((uint32_t)TIM_CCMR1_OC1M_2 | TIM_CCMR1_OC1M_1)
868 #define TIM_OCMODE_PWM2                  ((uint32_t)TIM_CCMR1_OC1M_2 | TIM_CCMR1_OC1M_1 | TIM_CCMR1_OC1M_0)
869 #define TIM_OCMODE_FORCED_ACTIVE         ((uint32_t)TIM_CCMR1_OC1M_2 | TIM_CCMR1_OC1M_0)
870 #define TIM_OCMODE_FORCED_INACTIVE       ((uint32_t)TIM_CCMR1_OC1M_2)
871
872 #define TIM_OCMODE_RETRIGERRABLE_OPM1    ((uint32_t)TIM_CCMR1_OC1M_3)
873 #define TIM_OCMODE_RETRIGERRABLE_OPM2    ((uint32_t)TIM_CCMR1_OC1M_3 | TIM_CCMR1_OC1M_0)
874 #define TIM_OCMODE_COMBINED_PWM1        ((uint32_t)TIM_CCMR1_OC1M_3 | TIM_CCMR1_OC1M_2)

```

Što se u suštini svodi na ovo



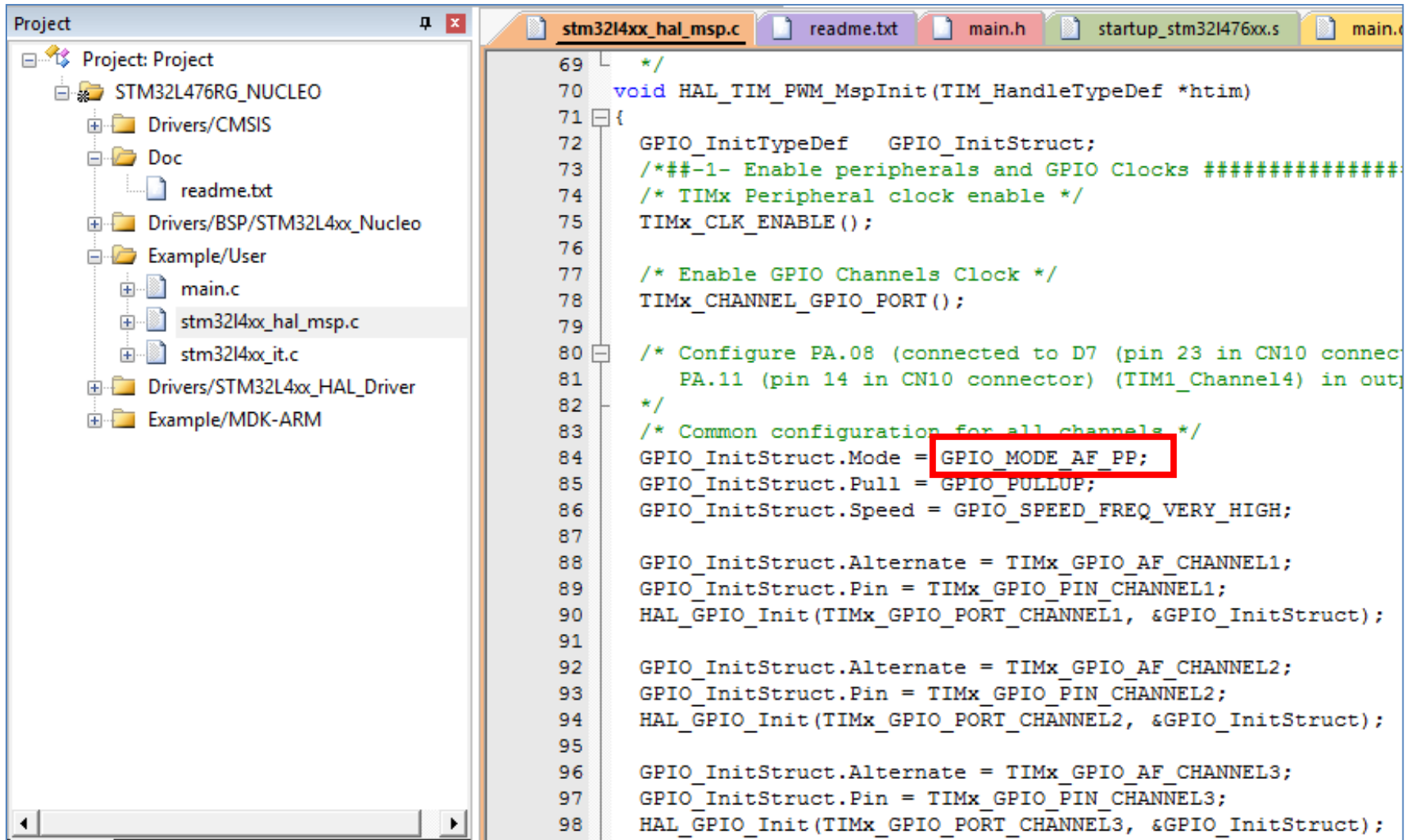
Podేశavanje svakog OC kanala

```
readme.txt  main.h  startup_stm32i476xx.s  main.c
166  /* Set the pulse value for channel 1 */
167  sConfig.Pulse = PULSE1_VALUE;
168  if (HAL_TIM_PWM_ConfigChannel(&TimHandle, &sConfig, TIM_CHANNEL_1) != HAL_OK)
169  {
170      /* Configuration Error */
171      Error_Handler();
172  }
173
174  /* Set the pulse value for channel 2 */
175  sConfig.Pulse = PULSE2_VALUE;
176  if (HAL_TIM_PWM_ConfigChannel(&TimHandle, &sConfig, TIM_CHANNEL_2) != HAL_OK)
177  {
178      /* Configuration Error */
179      Error_Handler();
180  }
181
182  /* Set the pulse value for channel 3 */
183  sConfig.Pulse = PULSE3_VALUE;
184  if (HAL_TIM_PWM_ConfigChannel(&TimHandle, &sConfig, TIM_CHANNEL_3) != HAL_OK)
185  {
186      /* Configuration Error */
187      Error_Handler();
188  }
189
190  /* Set the pulse value for channel 4 */
191  sConfig.Pulse = PULSE4_VALUE;
192  if (HAL_TIM_PWM_ConfigChannel(&TimHandle, &sConfig, TIM_CHANNEL_4) != HAL_OK)
193  {
194      /* Configuration Error */
195      Error_Handler();
196  }
197
```

Startovanje OC kanala

```
readme.txt  main.h  startup_stm32i476xx.s  main.c
197
198  /*##-3- Start PWM signals generation #####
199  /* Start channel 1 */
200  if (HAL_TIM_PWM_Start(&TimHandle, TIM_CHANNEL_1) != HAL_OK)
201  {
202      /* PWM Generation Error */
203      Error_Handler();
204  }
205  /* Start channel 2 */
206  if (HAL_TIM_PWM_Start(&TimHandle, TIM_CHANNEL_2) != HAL_OK)
207  {
208      /* PWM Generation Error */
209      Error_Handler();
210  }
211  /* Start channel 3 */
212  if (HAL_TIM_PWM_Start(&TimHandle, TIM_CHANNEL_3) != HAL_OK)
213  {
214      /* PWM generation Error */
215      Error_Handler();
216  }
217  /* Start channel 4 */
218  if (HAL_TIM_PWM_Start(&TimHandle, TIM_CHANNEL_4) != HAL_OK)
219  {
220      /* PWM generation Error */
221      Error_Handler();
222  }
223
224  while (1)
225  {
226  }
227
```


A šta je sa low level inicijalizacijom?



The image shows a screenshot of an IDE with two main windows. The left window displays a project tree for 'Project: Project' with the following structure:

- Project: Project
 - STM32L476RG_NUCLEO
 - Drivers/CMSIS
 - Doc
 - readme.txt
 - Drivers/BSP/STM32L4xx_Nucleo
 - Example/User
 - main.c
 - stm32l4xx_hal_msp.c (selected)
 - stm32l4xx_it.c
 - Drivers/STM32L4xx_HAL_Driver
 - Example/MDK-ARM

The right window shows the code for `stm32l4xx_hal_msp.c`. The code is as follows:

```
69  /*
70  void HAL_TIM_PWM_MspInit(TIM_HandleTypeDef *htim)
71  {
72      GPIO_InitTypeDef  GPIO_InitStructure;
73      /*##-1- Enable peripherals and GPIO Clocks #####
74      /* TIMx Peripheral clock enable */
75      TIMx_CLK_ENABLE();
76
77      /* Enable GPIO Channels Clock */
78      TIMx_CHANNEL_GPIO_PORT();
79
80      /* Configure PA.08 (connected to D7 (pin 23 in CN10 connect
81      PA.11 (pin 14 in CN10 connector) (TIM1_Channel4) in outp
82      */
83      /* Common configuration for all channels */
84      GPIO_InitStructure.Mode = GPIO_MODE_AF_PP;
85      GPIO_InitStructure.Pull = GPIO_PULLUP;
86      GPIO_InitStructure.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
87
88      GPIO_InitStructure.Alternate = TIMx_GPIO_AF_CHANNEL1;
89      GPIO_InitStructure.Pin = TIMx_GPIO_PIN_CHANNEL1;
90      HAL_GPIO_Init(TIMx_GPIO_PORT_CHANNEL1, &GPIO_InitStructure);
91
92      GPIO_InitStructure.Alternate = TIMx_GPIO_AF_CHANNEL2;
93      GPIO_InitStructure.Pin = TIMx_GPIO_PIN_CHANNEL2;
94      HAL_GPIO_Init(TIMx_GPIO_PORT_CHANNEL2, &GPIO_InitStructure);
95
96      GPIO_InitStructure.Alternate = TIMx_GPIO_AF_CHANNEL3;
97      GPIO_InitStructure.Pin = TIMx_GPIO_PIN_CHANNEL3;
98      HAL_GPIO_Init(TIMx_GPIO_PORT_CHANNEL3, &GPIO_InitStructure);
```

Ultra-low-power ARM[®] Cortex[®]-M4 32-bit MCU+FPU, 100DMIPS,
up to 1MB Flash, 128 KB SRAM, USB OTG FS, LCD, analog, audio

Datasheet - production data

Features

- Ultra-low-power with FlexPowerControl
 - 1.71 V to 3.6 V power supply
 - -40 °C to 85/105/125 °C temperature range
 - 300 nA in V_{BAT} mode: supply for RTC and

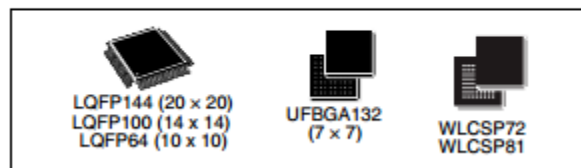


Table 15. STM32L476xxSTM32L476xx pin definitions (continued)

Pin Number						Pin name (function after reset)	Pin type	I/O structure	Notes	Pin functions	
LQFP64	WLCSP72	WLCSP81	LQFP100	UFBGA132	LQFP144					Alternate functions	Additional functions
41	E2	E2	67	D11	100	PA8	I/O	FT_I	-	MCO, TIM1_CH1, USART1_CK, OTG_FS_SOF, LCD_COM0, LPTIM2_OUT, EVENTOUT	-
42	E3	E3	68	D10	101	PA9	I/O	FT_Iu	-	TIM1_CH2, USART1_TX, LCD_COM1, TIM15_BKIN, EVENTOUT	OTG_FS_VBUS
43	D2	D2	69	C12	102	PA10	I/O	FT_Iu	-	TIM1_CH3, USART1_RX, OTG_FS_ID, LCD_COM2, TIM17_BKIN, EVENTOUT	-
44	D1	D1	70	B12	103	PA11	I/O	FT_u	-	TIM1_CH4, TIM1_BKIN2, USART1_CTS, CAN1_RX, OTG_FS_DM, TIM1_BKIN2_COMP1, EVENTOUT	-

MCU Datasheet

Table 17. Alternate function AF0 to AF7 (for AF8 to AF15 see [Table 18](#))

Port		AF0	AF1	AF2	AF3	AF4	AF5	AF6	AF7
		SYS_AF	TIM1/TIM2/ TIM5/TIM8/ LPTIM1	TIM1/TIM2/ TIM3/TIM4/ TIM5	TIM8	I2C1/I2C2/I2C3	SPI1/SPI2	SPI3/DFSDM	USART1/ USART2/ USART3
Port A	PA0	-	TIM2_CH1	TIM5_CH1	TIM8_ETR	-	-	-	USART2_CTS
	PA1	-	TIM2_CH2	TIM5_CH2	-	-	-	-	USART2_RTS_ DE
	PA2	-	TIM2_CH3	TIM5_CH3	-	-	-	-	USART2_TX
	PA3	-	TIM2_CH4	TIM5_CH4	-	-	-	-	USART2_RX
	PA4	-	-	-	-	-	SPI1_NSS	SPI3_NSS	USART2_CK
	PA5	-	TIM2_CH1	TIM2_ETR	TIM8_CH1N	-	SPI1_SCK	-	-
	PA6	-	TIM1_BKIN	TIM3_CH1	TIM8_BKIN	-	SPI1_MISO	-	USART3_CTS
	PA7	-	TIM1_CH1N	TIM3_CH2	TIM8_CH1N	-	SPI1_MOSI	-	-
	PA8	MCO	TIM1_CH1	-	-	-	-	-	USART1_CK
	PA9	-	TIM1_CH2	-	-	-	-	-	USART1_TX
	PA10	-	TIM1_CH3	-	-	-	-	-	USART1_RX
	PA11	-	TIM1_CH4	TIM1_BKIN2	-	-	-	-	USART1_CTS
	PA12	-	TIM1_ETR	-	-	-	-	-	USART1_RTS_ DE
	PA13	JTMS-SWDIO	IR_OUT	-	-	-	-	-	-
	PA14	JTCK-SWCLK	-	-	-	-	-	-	-
PA15	JTDI	TIM2_CH1	TIM2_ETR	-	-	SPI1_NSS	SPI3_NSS	-	

Kako da vidimo generisane signale?

- Simulator
 - Za sada ne postoji potpun simulator za STM32L476RG
- Osciloskop
- Logički analizator
- Vratimo nazad na naš mikrokontroler?

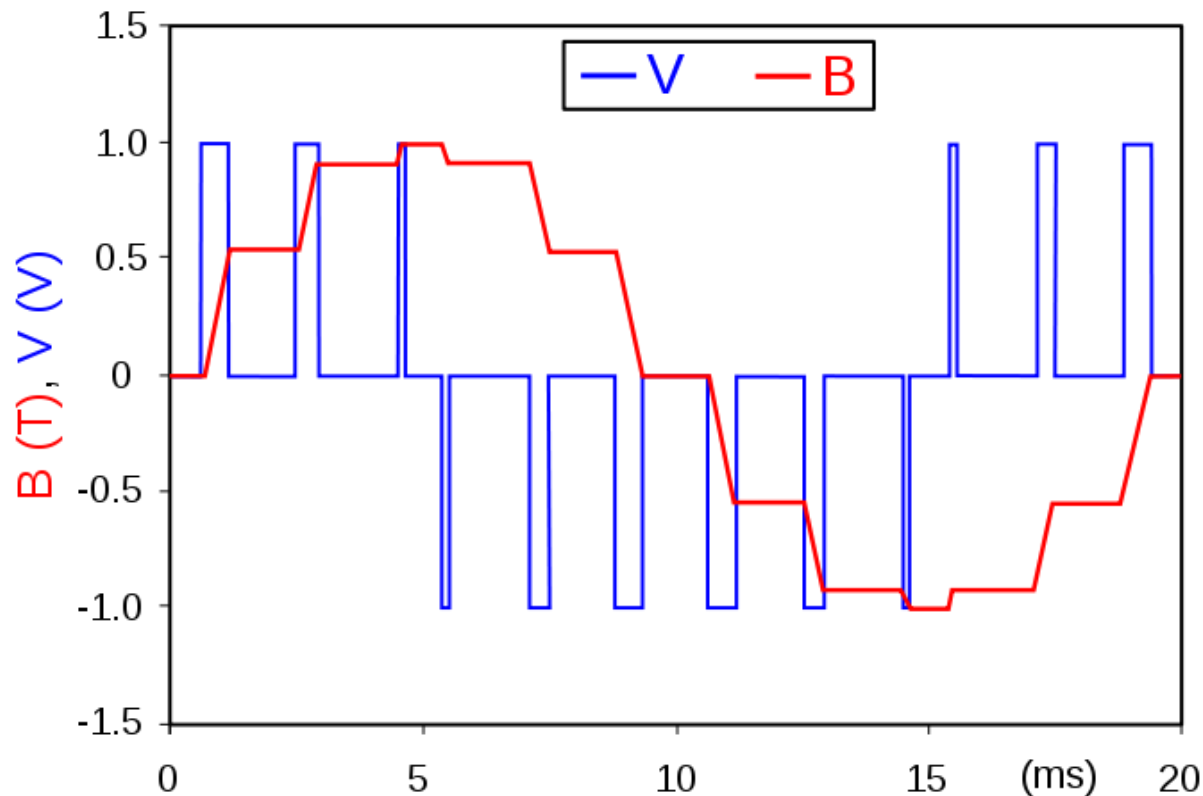
Zadatak za vežbu

- Izmeniti PWMOutput projekat tako da se Capture na kanalu jedan startuje sa dozvoljenim prekidom. U prekidnoj rutini CC1 treba toggle-ovati diodu.
- Voditi računa
 - TIM1 ima više IRQ Handler-a, izabrati odgovarajući
 - Dozvoliti prekide u Msplnit() funkciji
 - U stm32xxxx_it.h i _it.c fajlu definisati potrebnu IRQ handler funkciju
 - Startovati kanal 1 sa Start_IT
 - Proveriti koji interapt Flag se onda enable-uje
 - Naći koja se callback funkcija poziva i prepisati je u main.c fajlu da poziva BSP_LED_Toggle(LED2)

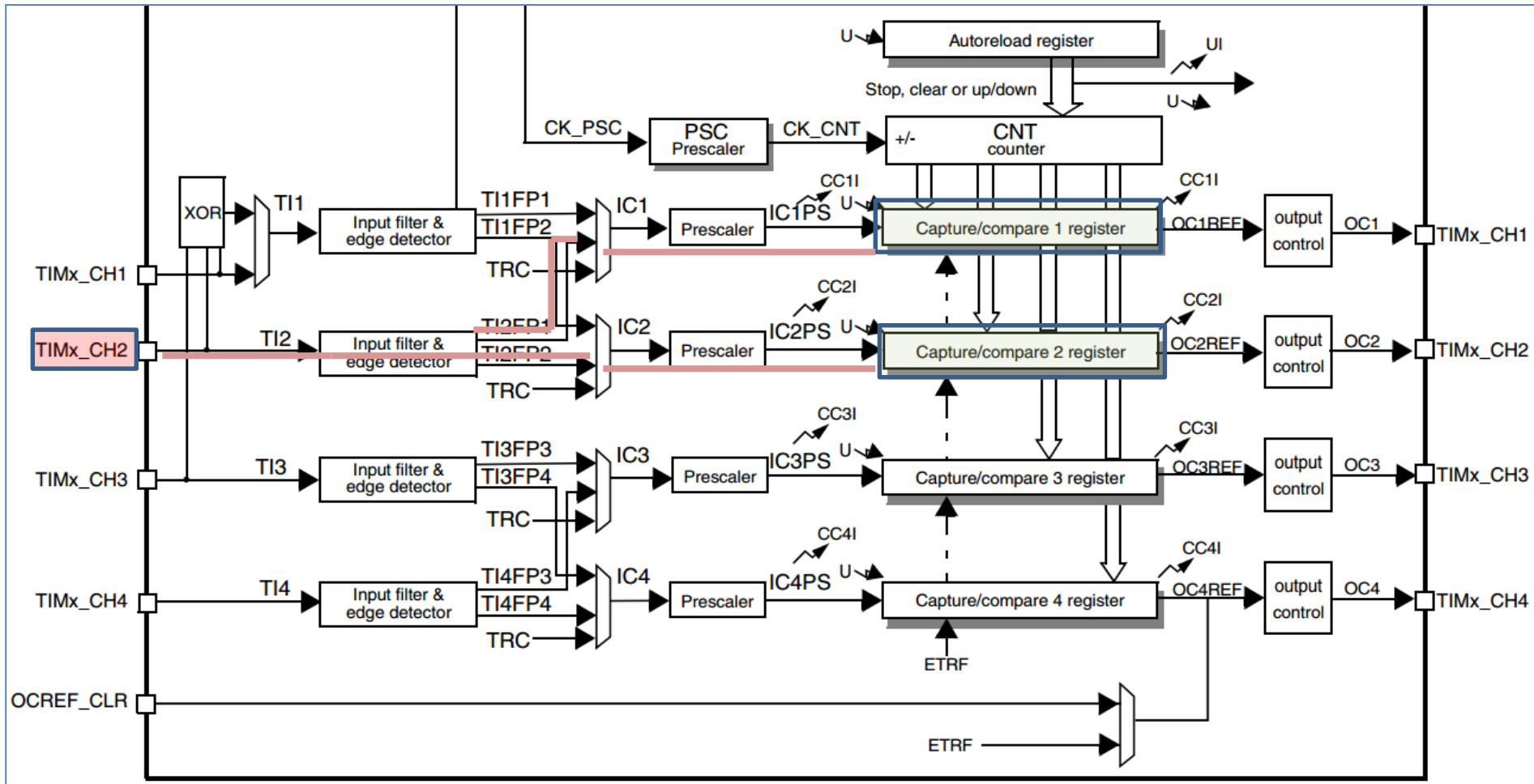
STM CUBE

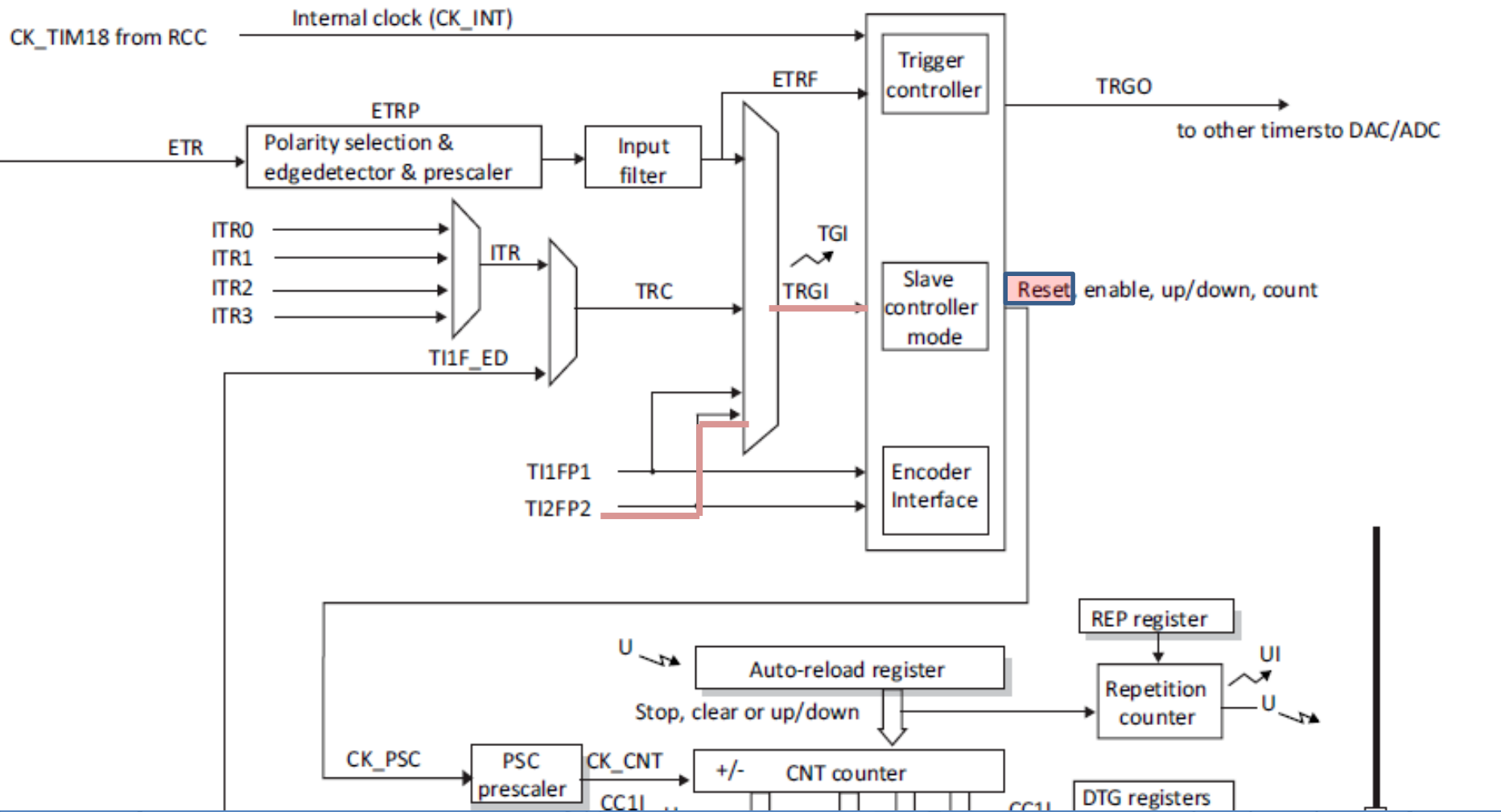
Projekat TIM_PWMInput

- Merenje karakteristika PWM signala (frekvencija i faktor ispunjenosti) uz pomoć tajmera

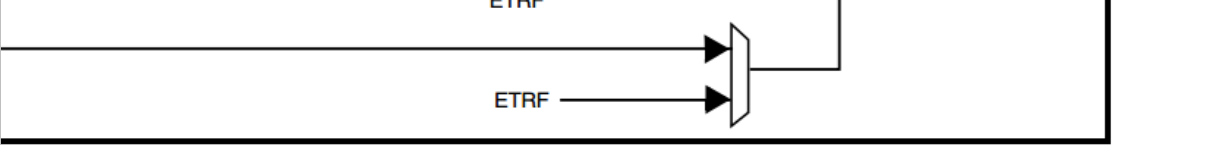


PWM input capture





IC2 signal se koristi i za resetovanje tajmera...



Prametri za IC

Koje su vrednosti za Polarity i ICSelection za CH1, a koje za CH2?

```
51 /* Private typedef -----*/
52 /* Private macro -----*/
53 /* Private variables -----*/
54 /* Timer handler declaration */
55 TIM_HandleTypeDef    TimHandle;
56
57 /* Timer Input Capture Configuration Structure declaration */
58 TIM_IC_InitTypeDef   sConfig;
59
60 /* Slave configuration structure */
61 TIM_SlaveConfigTypeDef  sSlaveConfig;
62
63 /* Captured Value */
64 __IO uint32_t         uwIC2Value = 0;
65 /* Duty Cycle Value */
66 __IO uint32_t         uwDutyCycle = 0;
67 /* Frequency Value */
68 __IO uint32_t         uwFrequency = 0;
```

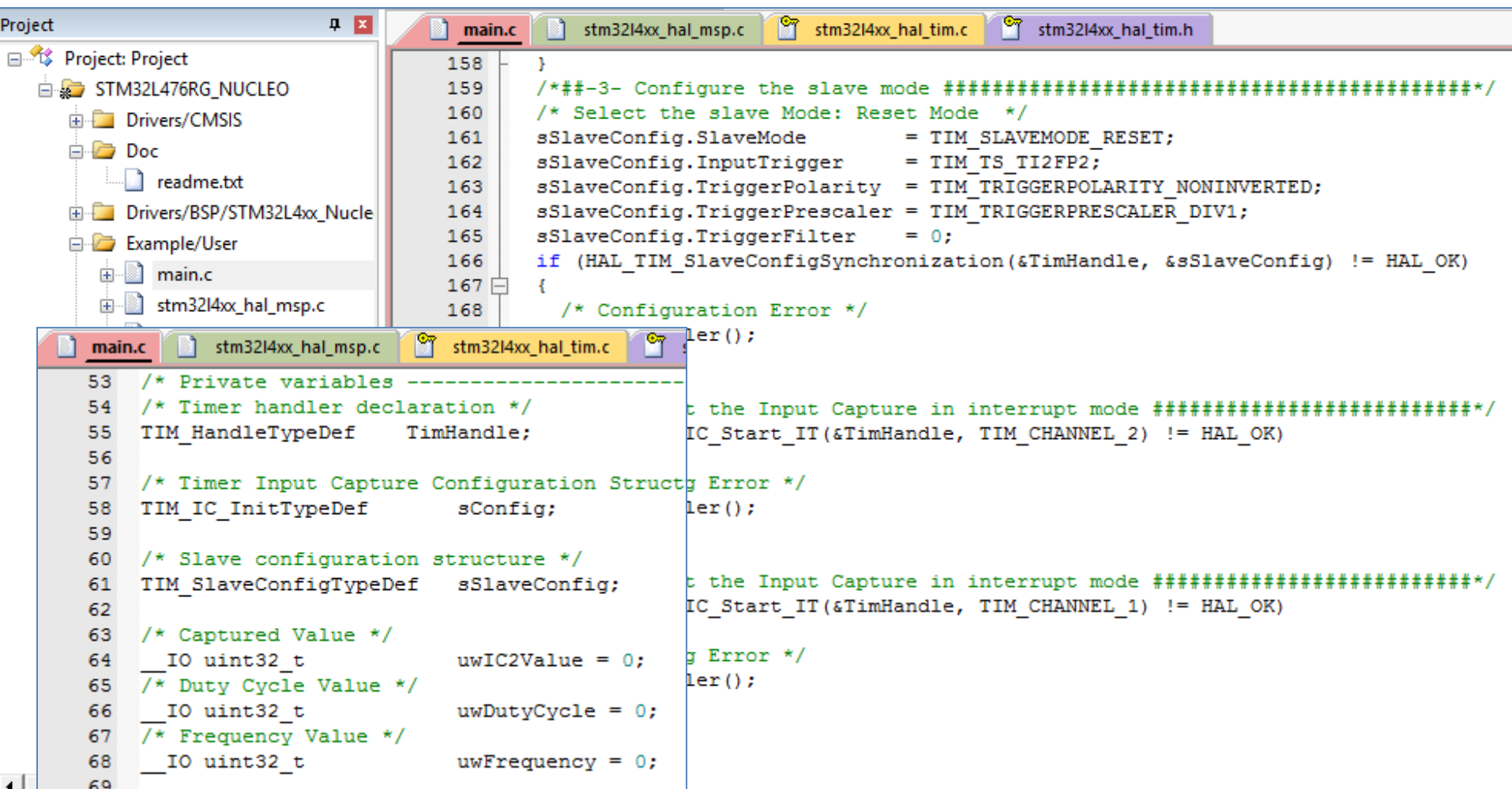
```
in.c  stm32l4xx_hal_msp.c  stm32l4xx_hal_tim.c  stm32l4xx_hal_tim.h
/**
 * @brief TIM Input Capture Configuration Structure definition
 */
typedef struct
{
    uint32_t  ICPolarity;    /*!< Specifies the active edge of the input signal.
                             This parameter can be a value of @ref TIM_Input_Capture_Polarity */
    uint32_t  ICSelection;   /*!< Specifies the input.
                             This parameter can be a value of @ref TIM_Input_Capture_Selection */
    uint32_t  ICPrescaler;  /*!< Specifies the Input Capture Prescaler.
                             This parameter can be a value of @ref TIM_Input_Capture_Prescaler */
    uint32_t  ICFilter;     /*!< Specifies the input capture filter.
                             This parameter can be a number between Min_Data = 0x0 and Max_Data = 0xF */
} TIM_IC_InitTypeDef;
```

```
/** @defgroup TIM_Input_Capture_Selection TIM Input Capture Selection
 * @{
 */
#define TIM_ICSELECTION_DIRECTTI    (TIM_CCMR1_CC1S_0) /*!< TIM Input 1, 2, 3 or 4 is selected to be
                                                         connected to IC1, IC2, IC3 or IC4, respectively
 */
#define TIM_ICSELECTION_INDIRECTTI  (TIM_CCMR1_CC1S_1) /*!< TIM Input 1, 2, 3 or 4 is selected to be
                                                         connected to IC2, IC1, IC4 or IC3, respectively
 */
#define TIM_ICSELECTION_TRC         (TIM_CCMR1_CC1S)  /*!< TIM Input 1, 2, 3 or 4 is selected to be connected to TRC
 */
/**
```

Podేశavanje svakog kanala

```
/*##-2- Configure the Input Capture channels #####*/  
/* Common configuration */  
sConfig.ICPrescaler = TIM_ICPSC_DIV1;  
sConfig.ICFilter = 0;  
  
/* Configure the Input Capture of channel 1 */  
sConfig.ICPolarity = TIM_ICPOLARITY_FALLING;  
sConfig.ICSelection = TIM_ICSELECTION_INDIRECTTI;  
if (HAL_TIM_IC_ConfigChannel(&TimHandle, &sConfig, TIM_CHANNEL_1) != HAL_OK)  
{  
    /* Configuration Error */  
    Error_Handler();  
}  
  
/* Configure the Input Capture of channel 2 */  
sConfig.ICPolarity = TIM_ICPOLARITY_RISING;  
sConfig.ICSelection = TIM_ICSELECTION_DIRECTTI;  
if (HAL_TIM_IC_ConfigChannel(&TimHandle, &sConfig, TIM_CHANNEL_2) != HAL_OK)  
{  
    /* Configuration Error */  
    Error_Handler();  
}
```

Posebna podešavanja za PWM input Slave mode CH2 IC treba da resetuje tajmer



The image shows a screenshot of an IDE with two windows displaying C code for configuring a timer in slave mode. The top window shows the configuration of the slave mode and the start of the input capture channel 2. The bottom window shows the private variables and the start of the input capture channel 1.

```
Project
├── Project
│   ├── STM32L476RG_NUCLEO
│   │   ├── Drivers/CMSIS
│   │   ├── Doc
│   │   │   └── readme.txt
│   │   ├── Drivers/BSP/STM32L4xx_Nucleo
│   │   └── Example/User
│   │       ├── main.c
│   │       └── stm32l4xx_hal_msp.c
└── ...

main.c
stm32l4xx_hal_msp.c
stm32l4xx_hal_tim.c
stm32l4xx_hal_tim.h

158 }
159 /*##-3- Configure the slave mode #####*/
160 /* Select the slave Mode: Reset Mode */
161 sSlaveConfig.SlaveMode = TIM_SLAVEMODE_RESET;
162 sSlaveConfig.InputTrigger = TIM_TS_TI2FP2;
163 sSlaveConfig.TriggerPolarity = TIM_TRIGGERPOLARITY_NONINVERTED;
164 sSlaveConfig.TriggerPrescaler = TIM_TRIGGERPRESCALER_DIV1;
165 sSlaveConfig.TriggerFilter = 0;
166 if (HAL_TIM_SlaveConfigSynchronization(&TimHandle, &sSlaveConfig) != HAL_OK)
167 {
168     /* Configuration Error */
169     HAL_TIM_ErrorHandler();
170 }

53 /* Private variables -----*/
54 /* Timer handler declaration */
55 TIM_HandleTypeDef TimHandle;
56
57 /* Timer Input Capture Configuration Structure Error */
58 TIM_IC_InitTypeDef sConfig;
59
60 /* Slave configuration structure */
61 TIM_SlaveConfigTypeDef sSlaveConfig;
62
63 /* Captured Value */
64 __IO uint32_t uwIC2Value = 0;
65 /* Duty Cycle Value */
66 __IO uint32_t uwDutyCycle = 0;
67 /* Frequency Value */
68 __IO uint32_t uwFrequency = 0;
69
```

Slave mode konfiguracija

```
main.c  stm32l4xx_hal_msp.c  stm32l4xx_hal_tim.c  stm32l4xx_hal_tim.h
260 typedef struct {
261     uint32_t  SlaveMode;          /*!< Slave mode selection
262                                     This parameter can be a value of @ref TIM_Slave_Mode */
263     uint32_t  InputTrigger;      /*!< Input Trigger source
264                                     This parameter can be a value of @ref TIM_Trigger_Selection */
265     uint32_t  TriggerPolarity;   /*!< Input Trigger polarity
266                                     This parameter can be a value of @ref TIM_Trigger_Polarity */
267     uint32_t  TriggerPrescaler;  /*!< Input trigger prescaler
268                                     This parameter can be a value of @ref TIM_Trigger_Prescaler */
269     uint32_t  TriggerFilter;     /*!< Input trigger filter
270                                     This parameter can be a number between Min_Data = 0x0 and Max_Data = 0xF */
271
272 }TIM_SlaveConfigTypeDef;
```

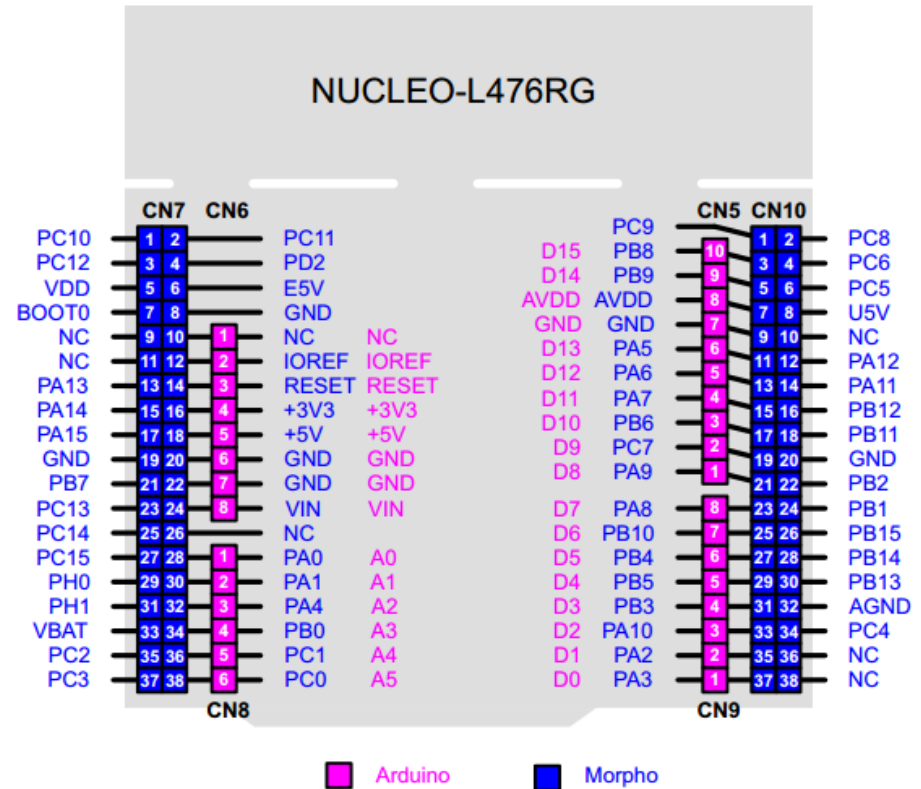
```
main.c  stm32l4xx_hal_msp.c  stm32l4xx_hal_tim.c  stm32l4xx_hal_tim.h
46
47 /** @defgroup TIM_Slave_Mode TIM Slave mode
48     * @{
49     */
50 #define TIM_SLAVEMODE_DISABLE          ((uint32_t)0x0000)
51 #define TIM_SLAVEMODE_RESET            ((uint32_t)(TIM_SMCR_SMS_2))
52 #define TIM_SLAVEMODE_GATED            ((uint32_t)(TIM_SMCR_SMS_2 | TIM_SMCR_SMS_0))
53 #define TIM_SLAVEMODE_TRIGGER         ((uint32_t)(TIM_SMCR_SMS_2 | TIM_SMCR_SMS_1))
54 #define TIM_SLAVEMODE_EXTERNAL1       ((uint32_t)(TIM_SMCR_SMS_2 | TIM_SMCR_SMS_1 | TIM_SMCR_SMS_0))
55 #define TIM_SLAVEMODE_COMBINED_RESETTRIGGER ((uint32_t)(TIM_SMCR_SMS_3))
56 /**
```

I na kraju to sve treba startovati

```
stm32i4xx_it.c  readme.txt  main.c*  stm32i4xx_hal_msp.c  stm32i4xx_hal_tim.c  stm32i4xx_hal_tim

163     sSlaveConfig.TriggerPolarity = TIM_TRIGGERPOLARITY_NONINVERTED;
164     sSlaveConfig.TriggerPrescaler = TIM_TRIGGERPRESCALER_DIV1;
165     sSlaveConfig.TriggerFilter    = 0;
166     if (HAL_TIM_SlaveConfigSynchronization(&TimHandle, &sSlaveConfig) != HAL_OK)
167     {
168         /* Configuration Error */
169         Error_Handler();
170     }
171
172     /*##-4- Start the Input Capture in interrupt mode #####*/
173     if (HAL_TIM_IC_Start_IT(&TimHandle, TIM_CHANNEL_2) != HAL_OK)
174     {
175         /* Starting Error */
176         Error_Handler();
177     }
178
179     /*##-5- Start the Input Capture in interrupt mode #####*/
180     if (HAL_TIM_IC_Start(&TimHandle, TIM_CHANNEL_1) != HAL_OK)
181     {
182         /* Starting Error */
183         Error_Handler();
184     }
185
186     while (1)
187     {
188     }
189 }
```

Zadatak



- Jedan kontroler izvršava projekat PWMOutput, a druga grupa izvršava projekat PWMInput
- Povezati dva kontrola pomoću dva provodnika – masa i signal.
- Pitanje: Gde se računaju parametri ulaznog PWM signala?
- Da li će projekat PMWInput korektno da radi ako se CH1 inicijalizuje bez prekidne funkcije tj. sa:
`HAL_TIM_IC_Start(&TimHandle, TIM_CHANNEL_1)?`

KEIL Osvežava promenljive u realnom vremenu

The screenshot displays the Keil IDE interface with a C source file open. A context menu is visible over the code, with the 'Add 'uwDutyCycle' to...' option selected. This option has opened a sub-menu where 'Watch 1' is chosen. The 'Watch 1' window at the bottom right shows the current values of variables: uwFrequency (24009), uwDutyCycle (49), and uwIC2Value (3328). The Command window at the bottom left shows the command 'WS 1, 'uwDutyCycle, 0x0A' being entered. The Find in Files window at the bottom shows a search for 'HAL_TIM_IC_Start_IT'.

```
54 /* Timer handler declaration */
55 TIM_HandleTypeDef TimHandle;
56
57 /* Timer Input Capture Configuration
58 TIM_IC_InitTypeDef sConfig;
59
60 /* Slave configuration structure */
61 TIM_SlaveConfigTypeDef sSlaveConf
62
63 /* Captured Value */
64 __IO uint32_t uwIC2Value
65 /* Duty Cycle Value */
66 __IO uint32_t uwDutyCycle
67 /* Frequency Value */
68 __IO uint32_t uwFrequency
69
70 /* Private function prototypes -----*/
71 void SystemClock_Config(void);
72 static void Error_Handler(void);
73
```

Command

```
WS 1, 'uwFrequency, 0x0A
WS 1, 'uwDutyCycle, 0x0A
WS 1, 'uwIC2Value, 0x0A
```

Watch 1

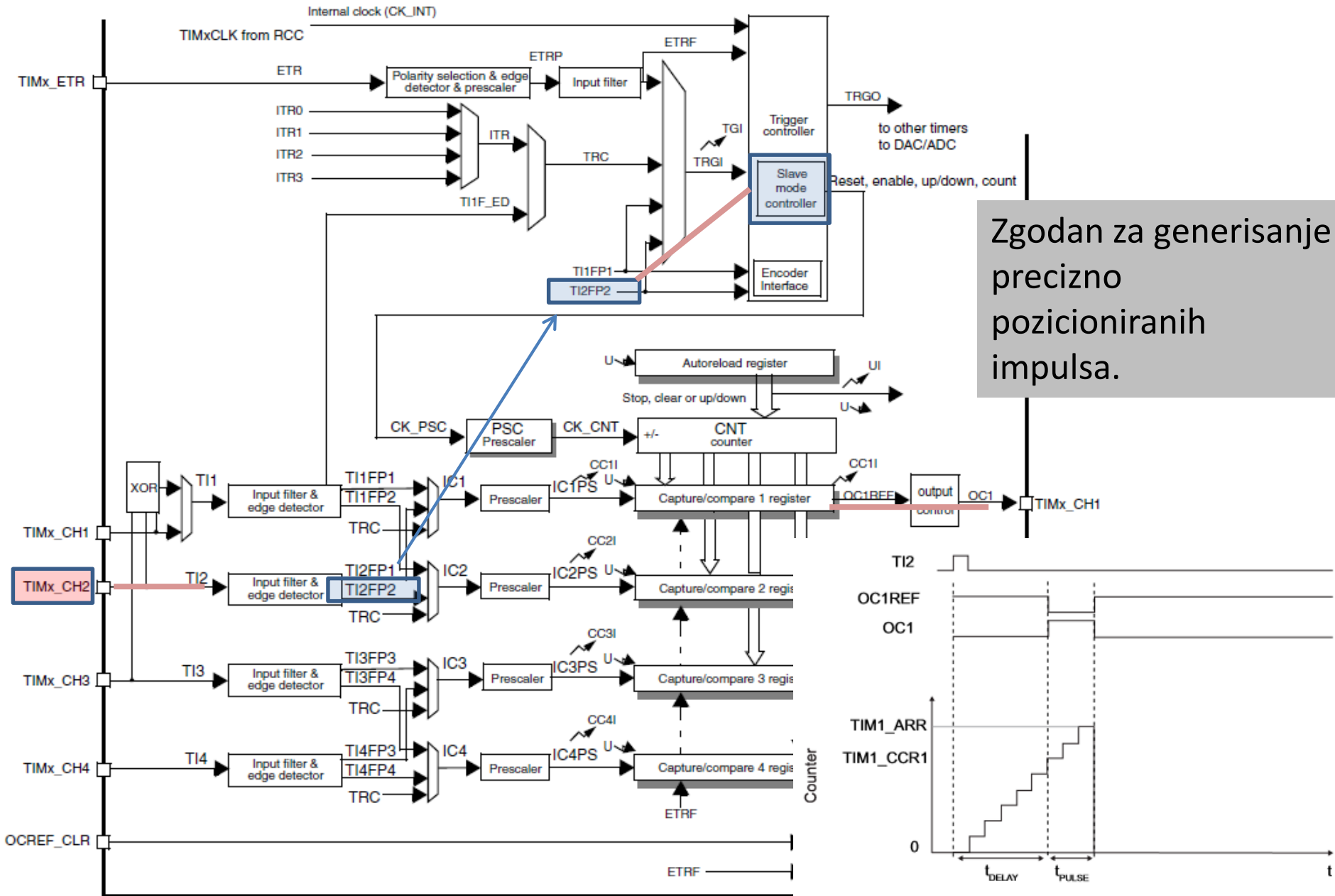
Name	Value	Type
uwFrequency	24009	unsign...
uwDutyCycle	49	unsign...
uwIC2Value	3328	unsign...

ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE

Find in Files

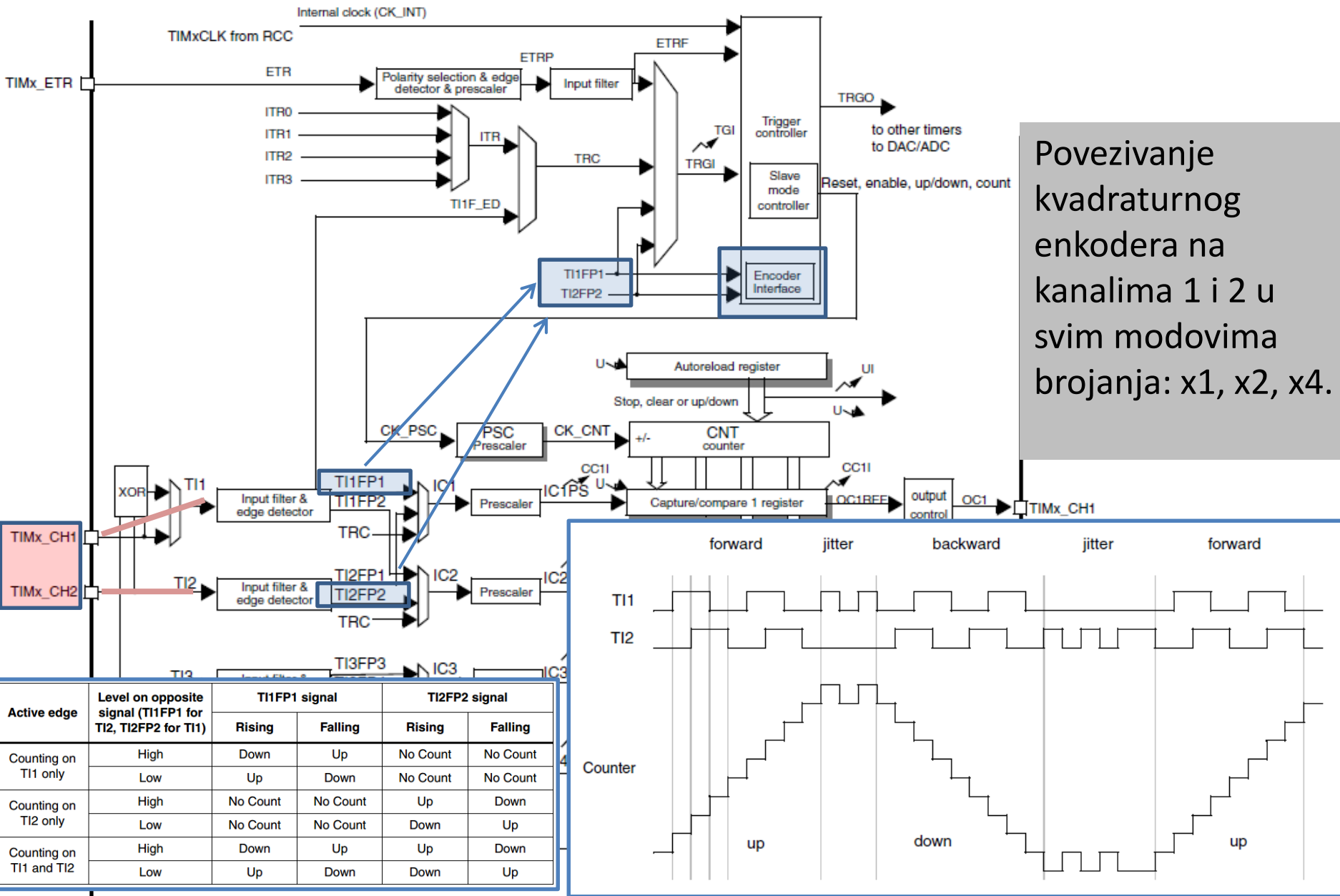
```
Searching for 'HAL_TIM_IC_Start_IT'...
C:\stm32cube14\STM32Cube_FW_L4_V1.4.0\Projects\STM32L476RG-Nucleo\Examples\TIM\TIM_PWMInput\Src\main.c(173) : if (HAL_TIM_IC_Sta
C:\stm32cube14\STM32Cube_FW_L4_V1.4.0\Projects\STM32L476RG-Nucleo\Examples\TIM\TIM_PWMInput\Src\main.c(180) : if (HAL_TIM_IC_Sta
C:\stm32cube14\STM32Cube_FW_L4_V1.4.0\Drivers\STM32L4xx_HAL_Driver\Src\stm32l4xx_hal_tim.c(87) : /* Input Capture
```

One pulse mod



Zgodan za generisanje precizno pozicioniranih impulsa.

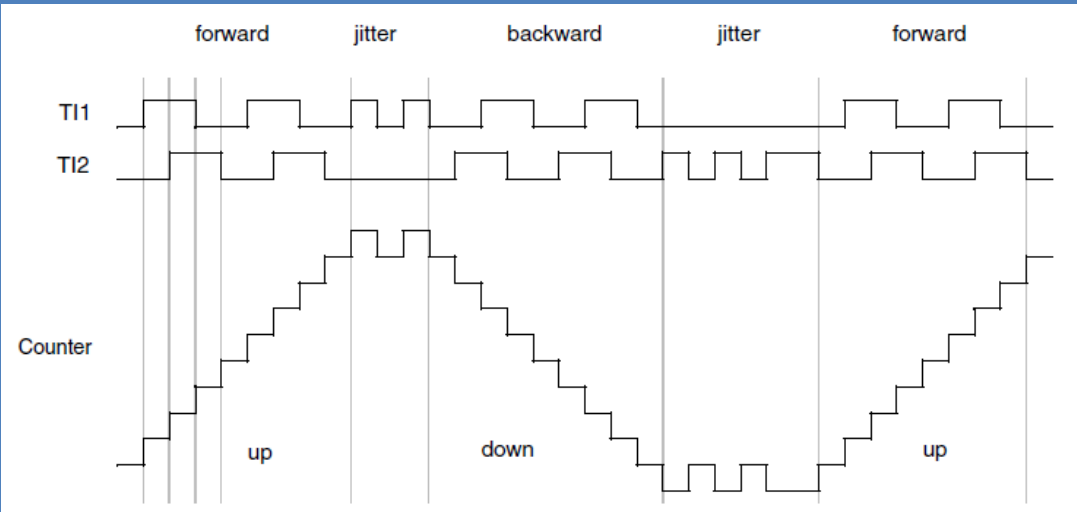
Encoder interface



Povezivanje kvadraturnog enkodera na kanalima 1 i 2 u svim modovima brojanja: x1, x2, x4.

TIMx_CH1
TIMx_CH2

Active edge	Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on TI2 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up



Encoder i MBED????

- Osnovna MBED biblioteka ne definiše klasu za encoder interfejs, ali postoje projekti i biblioteke u pokviru MBED baze. Doduše za LPC1768...

The screenshot shows the mbed IDE interface. The main window is titled "Import Wizard" and displays a list of libraries matching the search term "QEI". A dialog box titled "Import Library" is open, showing the source URL "http://mbed.org/users/Fairy_Paolina/code/QEI/" and the target path "LPC1768_blinky_32014".

Program Workspace

- main.cpp
- mbed
- Classes
- Files
- Structs
- Groups
- 2ak1016_web_HTTPServer
- ADC_spike
- AmpereMeter
- BlueUSB
- C3picasso
- EncoderRead
- ethsnif
- finalDS18S20
- GPS_HelloWorld
- hel_test
- Hello_World_HMC5843
- HMC5843
- HTTPClientStreamingExample
- HTTPServer
- HTTPServer-1
- HTTPServerHelloWorld
- HTTPServerN1
- HTTPServerTimeHandlerTest
- ICRSEurobot13
- IMUfilter
- IMUfilter_HelloWorld
- IMUfilter_RPYExample
- L3G4200
- LCDTFT_Library
- LPC1768_blinky_32014
- mbed-rpc
- Classes

Import Wizard

Import a library from mbed.org

Select library from the list. You can also drag&drop them in your workspace. [Click here](#) to import from URL.

Programs Libraries Bookmarked Upload

QEI Search

Listing published libraries on mbed.org matching "QEI". [Clear Search](#)

Name	Tags
QEI	Encoder QEI quadrature
QEI_hw	Encoder QEI quadrature ro
mRotaryEncoder	Encoder QEI quadrature ro
Robot	H-bridge Moror PID QEI S
QEI_hw_m	
QEI	
enc	Encoder QEI quadrature
QEI	Encoder QEI quadrature

Import Library

Import a library from mbed.org into a program in your workspace.

Please specify name

Source URL:

Import As: Program Library

Import Name:

Target Path:

New Program:

Import Cancel

Library Details

Name: QEI

Author: [Paolina Povolotskaya](#)

Published: 15 Mar 2014

Last Updated: 15 Mar 2014

Imports: 0

Forks: 0

Commits: 2

Dependents: 1

Dependencies: 0

Followers: 1

Library Homepage

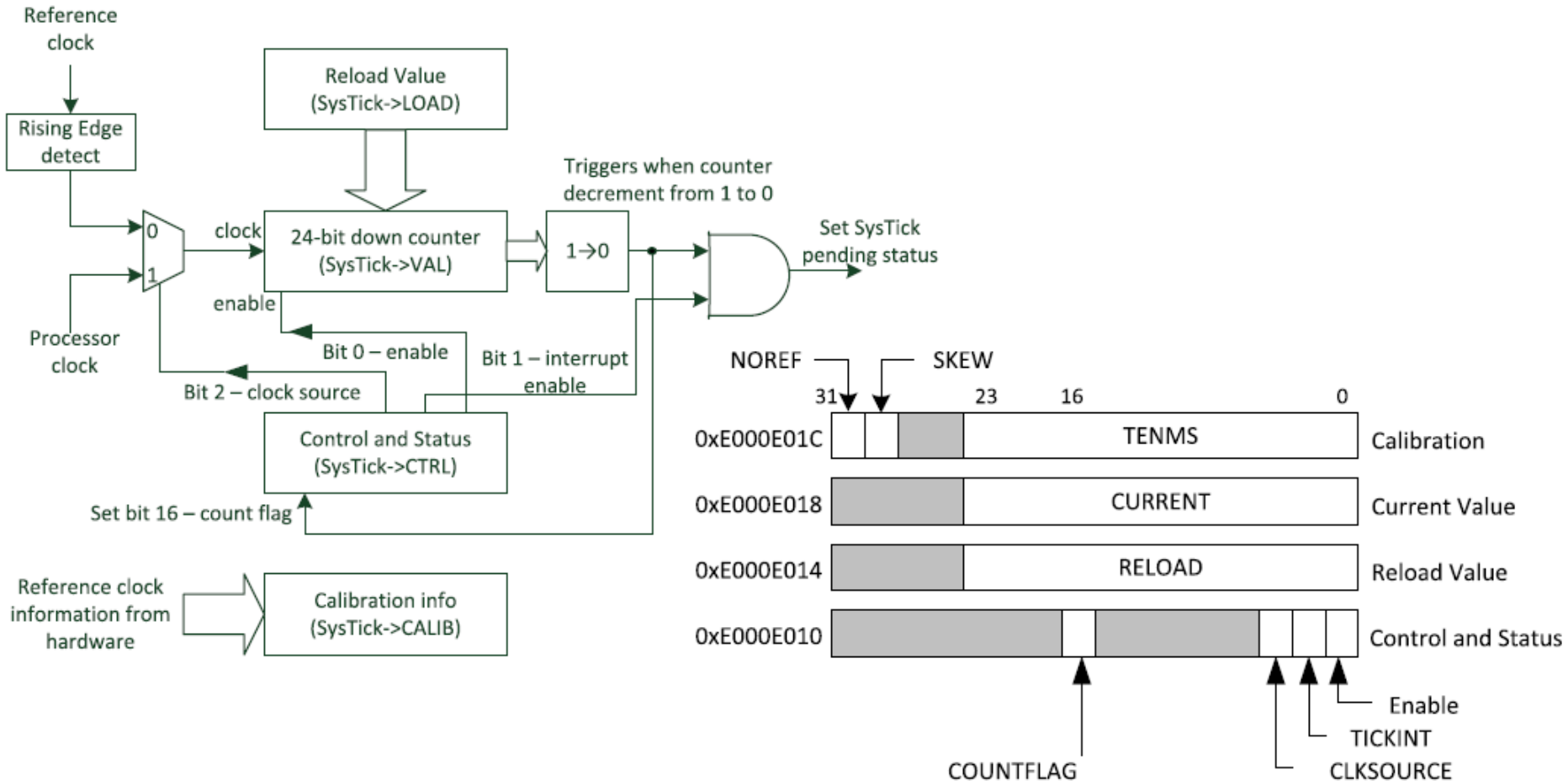
Tags: [Encoder QEI quadrature](#)

Description: [Quadrature encoder interface library.](#)

Systick tajmer

- Fleksibilni sistemski tajmer
- Sastavni deo procesorskog CORTEX-M4 jezgra.
- 24-bit auto-reload brojač na dole sa posebnim prekidom.
- 2 konfigurabilna izvora takta
- Pogodan za realizaciju real-time operativnih sistema.
- U STM32L4x5/6 seriji takt za ovu periferiju može biti ili CPU takt ili CPU/8 takt. Ovo se konfigurira u RCC grupi registara.
- S obzirom da je deo procesorskog jezgra definicije funkcija koje konfiguriraju rad časovnika se nalaze u okviru `core_cm4.h` fajla koji obezbeđuje ARM.
- Neke od high level funkcija može da obezbedi i proizvođač mikrokontrolera i one se nalaze u okviru peripheral drajver biblioteke i to u sastavu `misc.h` fajla.

Systick timer



MBED

Sistemski časovnik

<http://mbed.org/handbook/Ticker>

- Klasa Ticker daje jednostavan način realizacije periodičnog poziva korisničke funkcije, što praktično predstavlja sistemski časovnik.

Ticker Class Reference

#include <[Ticker.h](#)>

Inherits [mbed::TimerEvent](#).

Inherited by [Timeout](#).

Public Member Functions

void [attach](#) (void(*fptr)(void), float t)

Attach a function to be called by the **Ticker**, specifying the interval in seconds.

template<typename T >

void [attach](#) (T *tptr, void(T::*mptr)(void), float t)

Attach a member function to be called by the **Ticker**, specifying the interval in seconds.

void [attach_us](#) (void(*fptr)(void), unsigned int t)

Attach a function to be called by the **Ticker**, specifying the interval in micro-seconds.

template<typename T >

void [attach_us](#) (T *tptr, void(T::*mptr)(void), unsigned int t)

Attach a member function to be called by the **Ticker**, specifying the interval in micro-seconds.

void [detach](#) ()

Detach the function.

Static Public Member Functions

static void [irq](#) (uint32_t id)

The handler registered with the underlying timer interrupt.

Ticker - primer programa

```
#include "mbed.h"
Ticker flipper;
DigitalOut myled(LED1);

void flip() {
    myled = !myled;
}

int main() {
    myled = 1;
    flipper.attach(&flip, 0.5); // the address of the function to be attached (flip)
                                //and the interval (0.5 seconds)
                                // spin in a main loop. flipper will interrupt it to call flip
    while(1) ;
}
```

Povezivanje sa funkcijom članicom neke klase

```
#include "mbed.h"

// A class for flip()-ing a DigitalOut
class Flipper {
public:
    Flipper(PinName pin) : _pin(pin) {
        _pin = 0;
    }
    void flip() {
        _pin = !_pin;
    }
private:
    DigitalOut _pin;
};

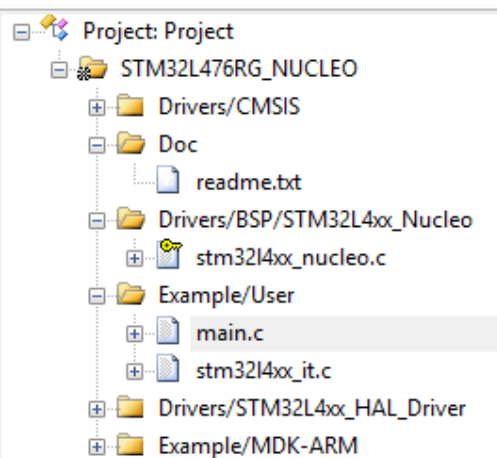
Flipper f(LED1);
Ticker t;

int main() {
    t.attach(&f, &Flipper::flip, 0.5); // the address of the object, member function, and interval
                                        // spin in a main loop. flipper will interrupt it to call flip
    while(1);
}
```

STM CUBE

Projekat CORTEXM_SysTick

\\stm32cubel4\STM32Cube_FW_L4_V1.4.0\Projects\STM32L476RG-Nucleo\Examples\Cortex\CORTEXM_SysTick\MDK-ARM



```
110
111  /* Infinite loop */
112  while (1)
113  {
114      /* Toggle LED2 */
115      BSP_LED_Toggle(LED2);
116
117      /* Insert 50 ms delay */
118      HAL_Delay(50);
119  }
120
121 }
```

Naizmenična promena stanja dioda sa promenljivim intervalima čekanja

```
stm32l4xx_hal.c
321  * @brief Provide accurate delay (in milliseconds) based on variable incremented.
322  * @note In the default implementation , SysTick timer is the source of time base.
323  *       It is used to generate interrupts at regular time intervals where uwTick
324  *       is incremented.
325  * @note This function is declared as __weak to be overwritten in case of other
326  *       implementations in user file.
327  * @param Delay: specifies the delay time length, in milliseconds.
328  * @retval None
329  */
330  __weak void HAL_Delay(uint32_t Delay)
331  {
332      uint32_t tickstart = 0;
333      tickstart = HAL_GetTick();
334      while((HAL_GetTick() - tickstart) < Delay)
335      {
336      }
337  }
```

Delay() funkcija ne implementira čekanje kao dummy petlju, ali je ipak čekanje u mestu.

Realizacija preko prekida

The image shows a screenshot of an IDE with two code windows. The top window, titled 'stm32l4xx_it.c', shows the implementation of the SysTick interrupt handler. The function 'SysTick_Handler' is defined, and it calls 'HAL_IncTick()'. A red box highlights the function definition, and a red dashed arrow points from a red text annotation to it. The bottom window, titled 'stm32l4xx_hal.c', shows the implementation of the HAL driver function 'HAL_IncTick'. A blue box highlights the function definition, and a blue dashed arrow points from a blue text annotation to it. The left sidebar shows the project structure, including folders for Drivers, Doc, and Example, and files like 'main.c' and 'stm32l4xx_it.c'.

```
154 * @retval None
155 */
156 void SysTick_Handler(void)
157 {
158     HAL_IncTick();
159 }
160
161 /*****
162 /*
163 /* Add here the Interrupt Handler for the used peripheral(s) (PPP), for the
164 /* available peripheral interrupt handler's name please refer to the startup
165 /* file (startup_stm32l4xx.s).
166 */
167 */
168 */
169 */
170 */
171 */
172 */
173 */
174 */
175 */
176 */
177 */
178 */
179 */
180 */
181 */
182 */
183 */
184 */
185 */
186 */
187 */
188 */
189 */
190 */
191 */
192 */
193 */
194 */
195 */
196 */
197 */
198 */
199 */
200 */
201 */
202 */
203 */
204 */
205 */
206 */
207 */
208 */
209 */
210 */
211 */
212 */
213 */
214 */
215 */
216 */
217 */
218 */
219 */
220 */
221 */
222 */
223 */
224 */
225 */
226 */
227 */
228 */
229 */
230 */
231 */
232 */
233 */
234 */
235 */
236 */
237 */
238 */
239 */
240 */
241 */
242 */
243 */
244 */
245 */
246 */
247 */
248 */
249 */
250 */
251 */
252 */
253 */
254 */
255 */
256 */
257 */
258 */
259 */
260 */
261 */
262 */
263 */
264 */
265 */
266 */
267 */
268 */
269 */
270 */
271 */
272 */
273 */
274 */
275 */
276 */
277 */
278 */
279 */
280 */
281 */
282 */
283 */
284 */
285 */
286 */
287 */
288 */
289 */
290 */
291 */
292 */
293 */
294 */
295 */
296 */
297 */
298 */
299 */
300 */
301 */
302 */
303 */
304 */
305 */
306 */
307 */
308 */
309 */
310 */
311 */
312 */
313 */
314 */
315 */
316 */
317 */
318 */
319 */
320 */
321 */
322 */
323 */
324 */
325 */
326 */
327 */
328 */
329 */
330 */
331 */
332 */
333 */
334 */
335 */
336 */
337 */
338 */
339 */
340 */
341 */
342 */
343 */
344 */
345 */
346 */
347 */
348 */
349 */
350 */
351 */
352 */
353 */
354 */
355 */
356 */
357 */
358 */
359 */
360 */
361 */
362 */
363 */
364 */
365 */
366 */
367 */
368 */
369 */
370 */
371 */
372 */
373 */
374 */
375 */
376 */
377 */
378 */
379 */
380 */
381 */
382 */
383 */
384 */
385 */
386 */
387 */
388 */
389 */
390 */
391 */
392 */
393 */
394 */
395 */
396 */
397 */
398 */
399 */
400 */
401 */
402 */
403 */
404 */
405 */
406 */
407 */
408 */
409 */
410 */
411 */
412 */
413 */
414 */
415 */
416 */
417 */
418 */
419 */
420 */
421 */
422 */
423 */
424 */
425 */
426 */
427 */
428 */
429 */
430 */
431 */
432 */
433 */
434 */
435 */
436 */
437 */
438 */
439 */
440 */
441 */
442 */
443 */
444 */
445 */
446 */
447 */
448 */
449 */
450 */
451 */
452 */
453 */
454 */
455 */
456 */
457 */
458 */
459 */
460 */
461 */
462 */
463 */
464 */
465 */
466 */
467 */
468 */
469 */
470 */
471 */
472 */
473 */
474 */
475 */
476 */
477 */
478 */
479 */
480 */
481 */
482 */
483 */
484 */
485 */
486 */
487 */
488 */
489 */
490 */
491 */
492 */
493 */
494 */
495 */
496 */
497 */
498 */
499 */
500 */
501 */
502 */
503 */
504 */
505 */
506 */
507 */
508 */
509 */
510 */
511 */
512 */
513 */
514 */
515 */
516 */
517 */
518 */
519 */
520 */
521 */
522 */
523 */
524 */
525 */
526 */
527 */
528 */
529 */
530 */
531 */
532 */
533 */
534 */
535 */
536 */
537 */
538 */
539 */
540 */
541 */
542 */
543 */
544 */
545 */
546 */
547 */
548 */
549 */
550 */
551 */
552 */
553 */
554 */
555 */
556 */
557 */
558 */
559 */
560 */
561 */
562 */
563 */
564 */
565 */
566 */
567 */
568 */
569 */
570 */
571 */
572 */
573 */
574 */
575 */
576 */
577 */
578 */
579 */
580 */
581 */
582 */
583 */
584 */
585 */
586 */
587 */
588 */
589 */
590 */
591 */
592 */
593 */
594 */
595 */
596 */
597 */
598 */
599 */
600 */
601 */
602 */
603 */
604 */
605 */
606 */
607 */
608 */
609 */
610 */
611 */
612 */
613 */
614 */
615 */
616 */
617 */
618 */
619 */
620 */
621 */
622 */
623 */
624 */
625 */
626 */
627 */
628 */
629 */
630 */
631 */
632 */
633 */
634 */
635 */
636 */
637 */
638 */
639 */
640 */
641 */
642 */
643 */
644 */
645 */
646 */
647 */
648 */
649 */
650 */
651 */
652 */
653 */
654 */
655 */
656 */
657 */
658 */
659 */
660 */
661 */
662 */
663 */
664 */
665 */
666 */
667 */
668 */
669 */
670 */
671 */
672 */
673 */
674 */
675 */
676 */
677 */
678 */
679 */
680 */
681 */
682 */
683 */
684 */
685 */
686 */
687 */
688 */
689 */
690 */
691 */
692 */
693 */
694 */
695 */
696 */
697 */
698 */
699 */
700 */
701 */
702 */
703 */
704 */
705 */
706 */
707 */
708 */
709 */
710 */
711 */
712 */
713 */
714 */
715 */
716 */
717 */
718 */
719 */
720 */
721 */
722 */
723 */
724 */
725 */
726 */
727 */
728 */
729 */
730 */
731 */
732 */
733 */
734 */
735 */
736 */
737 */
738 */
739 */
740 */
741 */
742 */
743 */
744 */
745 */
746 */
747 */
748 */
749 */
750 */
751 */
752 */
753 */
754 */
755 */
756 */
757 */
758 */
759 */
760 */
761 */
762 */
763 */
764 */
765 */
766 */
767 */
768 */
769 */
770 */
771 */
772 */
773 */
774 */
775 */
776 */
777 */
778 */
779 */
780 */
781 */
782 */
783 */
784 */
785 */
786 */
787 */
788 */
789 */
790 */
791 */
792 */
793 */
794 */
795 */
796 */
797 */
798 */
799 */
800 */
801 */
802 */
803 */
804 */
805 */
806 */
807 */
808 */
809 */
810 */
811 */
812 */
813 */
814 */
815 */
816 */
817 */
818 */
819 */
820 */
821 */
822 */
823 */
824 */
825 */
826 */
827 */
828 */
829 */
830 */
831 */
832 */
833 */
834 */
835 */
836 */
837 */
838 */
839 */
840 */
841 */
842 */
843 */
844 */
845 */
846 */
847 */
848 */
849 */
850 */
851 */
852 */
853 */
854 */
855 */
856 */
857 */
858 */
859 */
860 */
861 */
862 */
863 */
864 */
865 */
866 */
867 */
868 */
869 */
870 */
871 */
872 */
873 */
874 */
875 */
876 */
877 */
878 */
879 */
880 */
881 */
882 */
883 */
884 */
885 */
886 */
887 */
888 */
889 */
890 */
891 */
892 */
893 */
894 */
895 */
896 */
897 */
898 */
899 */
900 */
901 */
902 */
903 */
904 */
905 */
906 */
907 */
908 */
909 */
910 */
911 */
912 */
913 */
914 */
915 */
916 */
917 */
918 */
919 */
920 */
921 */
922 */
923 */
924 */
925 */
926 */
927 */
928 */
929 */
930 */
931 */
932 */
933 */
934 */
935 */
936 */
937 */
938 */
939 */
940 */
941 */
942 */
943 */
944 */
945 */
946 */
947 */
948 */
949 */
950 */
951 */
952 */
953 */
954 */
955 */
956 */
957 */
958 */
959 */
960 */
961 */
962 */
963 */
964 */
965 */
966 */
967 */
968 */
969 */
970 */
971 */
972 */
973 */
974 */
975 */
976 */
977 */
978 */
979 */
980 */
981 */
982 */
983 */
984 */
985 */
986 */
987 */
988 */
989 */
990 */
991 */
992 */
993 */
994 */
995 */
996 */
997 */
998 */
999 */
```

Prekidna rutina SysTick_Handler() realizuje odbrojavanje vremena pozivom HAL funkcije

Jedna od osnovnih HAL drajver funkcija

Inicijalizacija SysTick tajmera

The image shows a screenshot of an IDE with a project tree on the left and three code files open in the main editor area. The project tree shows a project named 'Project' with a sub-project 'STM32L476RG_NUCLEO'. The main editor area displays the following code:

```
main.c
74     - Set NVIC Group Priority to 4
75     - Low Level Initialization
76     */
77     HAL_Init();
78
```

```
stm32l4xx_hal.c
256  * @param TickPriority: Tick interrupt priority.
257  * @retval HAL status
258  */
259  __weak HAL_StatusTypeDef HAL_InitTick(uint32_t TickPriority)
260  {
261     /*Configure the SysTick to have interrupt in 1ms time basis*/
262     HAL_SYSTICK_Config(SystemCoreClock/1000);
263
264     /*Configure the SysTick IRQ priority */
265     HAL_NVIC_SetPriority(SysTick_IRQn, TickPriority ,0);
```

```
stm32l4xx_hal_cortex.c
267  */
268  uint32_t HAL_SYSTICK_Config(uint32_t TicksNumb)
269  {
270     return SysTick_Config(TicksNumb);
271  }
272  /**
```

```
core_cm4.h
1829  */
1830  __STATIC_INLINE uint32_t SysTick_Config(uint32_t ticks)
1831  {
1832     if ((ticks - 1UL) > SysTick_LOAD_RELOAD_Msk)
1833     {
1834         return (1UL); /* Reload value impossible */
1835     }
1836
1837     SysTick->LOAD = (uint32_t)(ticks - 1UL); /* set reload register */
1838     NVIC_SetPriority (SysTick_IRQn, (1UL << __NVIC_PRIO_BITS) - 1UL); /* set Priority for SysTick Interrupt */
1839     SysTick->VAL = 0UL; /* Load the SysTick Counter Value */
1840     SysTick->CTRL = SysTick_CTRL_CLKSOURCE_Msk |
1841                   SysTick_CTRL_TICKINT_Msk |
1842                   SysTick_CTRL_ENABLE_Msk; /* Enable SysTick IRQ and SysTick Timer */
1843     return (0UL); /* Function successful */
1844 }
1845
1846 #endif
```

Zadatak

- Izmeniti program tako da se prekid sistemskog tajmera generiše sa učestanošću od 500Hz.
- Dakle u pitanju je perioda od 2ms. Gde je najjednostavnije izvršiti izmenu?

Klasa Timeout

<http://mbed.org/handbook/Timeout>

- Klasa Timeout se koristi kada je potrebno generisati akciju nakon isteka određenog vremena.
- Praktično ima isti interfejs kao klasa Ticker, ali se izvršava samo jedan interval.

Timeout Class Reference

```
#include <Timeout.h>
```

Inherits [mbed::Ticker](#).

Public Member Functions

```
void attach (void(*fptr)(void), float t)
```

Attach a function to be called by the [Ticker](#), specifying the interval in seconds.

```
template<typename T >
```

```
void attach (T *tptr, void(T::*mptr)(void), float t)
```

Attach a member function to be called by the [Ticker](#), specifying the interval in seconds.

```
void attach\_us (void(*fptr)(void), unsigned int t)
```

Attach a function to be called by the [Ticker](#), specifying the interval in micro-seconds.

```
template<typename T >
```

```
void attach\_us (T *tptr, void(T::*mptr)(void), unsigned int t)
```

Attach a member function to be called by the [Ticker](#), specifying the interval in micro-seconds.

```
void detach ()
```

Detach the function.

Static Public Member Functions

```
static void irq (uint32_t id)
```

The handler registered with the underlying timer interrupt.

Timeout

```
#include "mbed.h"

Timeout flipper;
DigitalOut led1(LED1);

void flip() {
    led1 = !led1;
}

int main() {
    led1 = 1;
    flipper.attach(&flip, 2.0); // setup flipper to call flip after 2
seconds

    // spin in a main loop. flipper will interrupt it to call flip
    while(1);
}
```

Klasa Timer – merenje vremena

<http://mbed.org/handbook/Timer>

- Klasa koja može da se koristi prilikom testiranja za merenje vremena.

```
#include "mbed.h"

Timer t;

int main() {
    t.start();
    printf("Hello World!\n");
    t.stop();
    printf("The time taken was %f
seconds\n", t.read());
}
```

Timer Class Reference

```
#include <Timer.h>
```

Public Member Functions

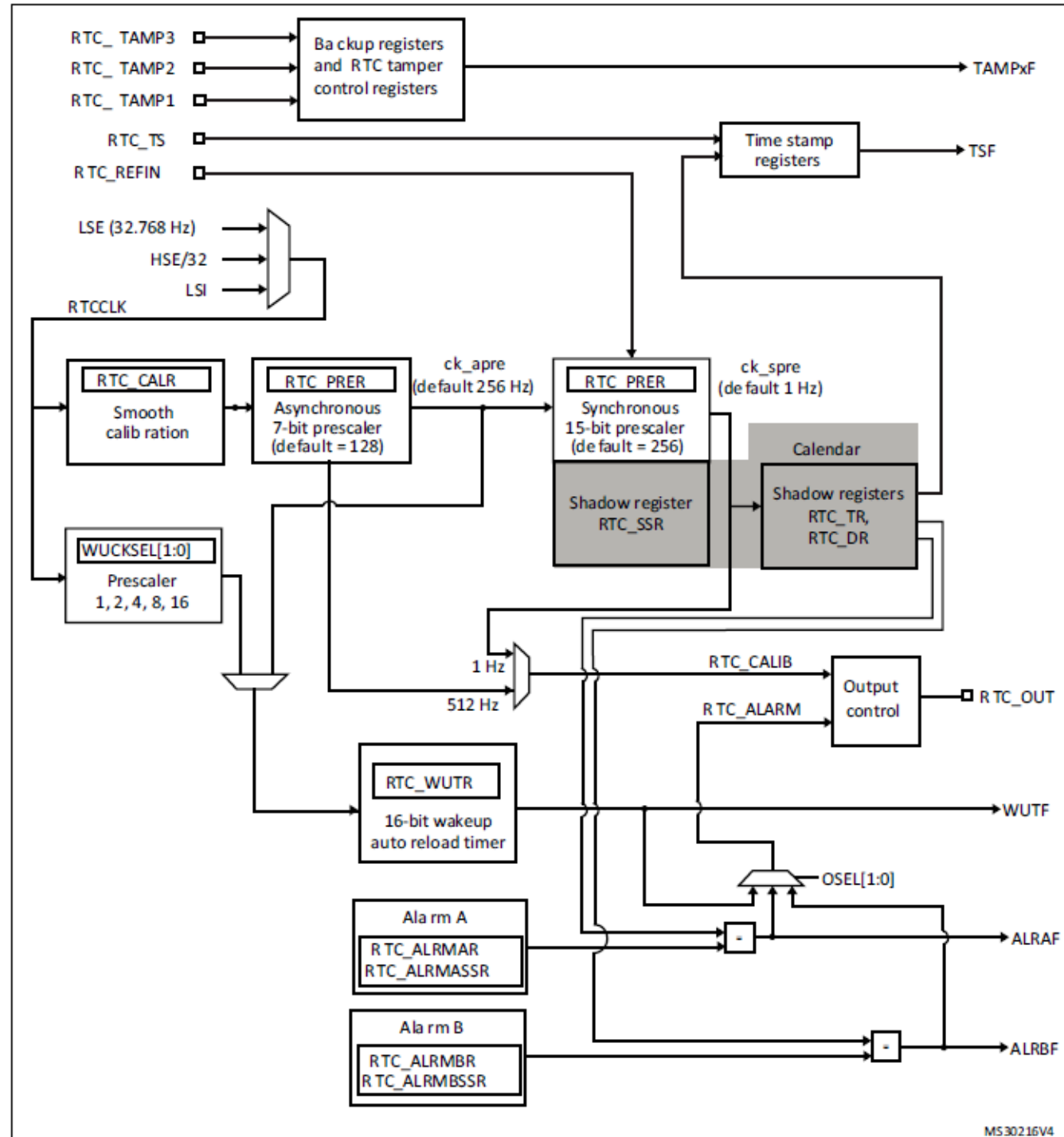
```
void start ()
    Start the timer.
void stop ()
    Stop the timer.
void reset ()
    Reset the timer to 0.
float read ()
    Get the time passed in seconds.
int read\_ms ()
    Get the time passed in milli-seconds.
int read\_us ()
    Get the time passed in micro-seconds.
```

Real Time Clock - RTC

- Real-time clock (RTC) je nezavisni BCD tajmer, tj. brojač. RTC obezbeđuje realno vreme vezano za dan, mesec, godinu kao i podesive alarme vezane za datum i vreme.
- Dva 32-bitna registra sadrže informaciju o realnom vremenu i to u trenutnim sekundama, minutima, satima, danima (u nedelji), datumu (dan, mesec, godina). Podaci su predstavljeni u BCD formatu.
- Dok god je napajanje u nominalnom opsegu RTC nikada ne prestaje da radi, nevezano za trenutno stanje uređaja (aktivno, low-power, ili pod resetom).

RTC

- Dva alarma
- Tri tamper događaja



Sat realnog vremena – RTC

- Za podršku su kreirane MBED funkcije, ali ne postoji klasa RTC. Funkcije su deklarirane u `time.h` i `rtc_time.h`.

[time](#) [FUNCTIONS](#)

Implementation of the C `time.h` functions

[time](#)

Get the current time

[set_time](#)

Set the current time

[mktime](#)

Converts a `tm` structure in to a timestamp

[localtime](#)

Converts a timestamp in to a `tm` structure

[ctime](#)

Converts a timestamp to a human-readable string

[strftime](#)

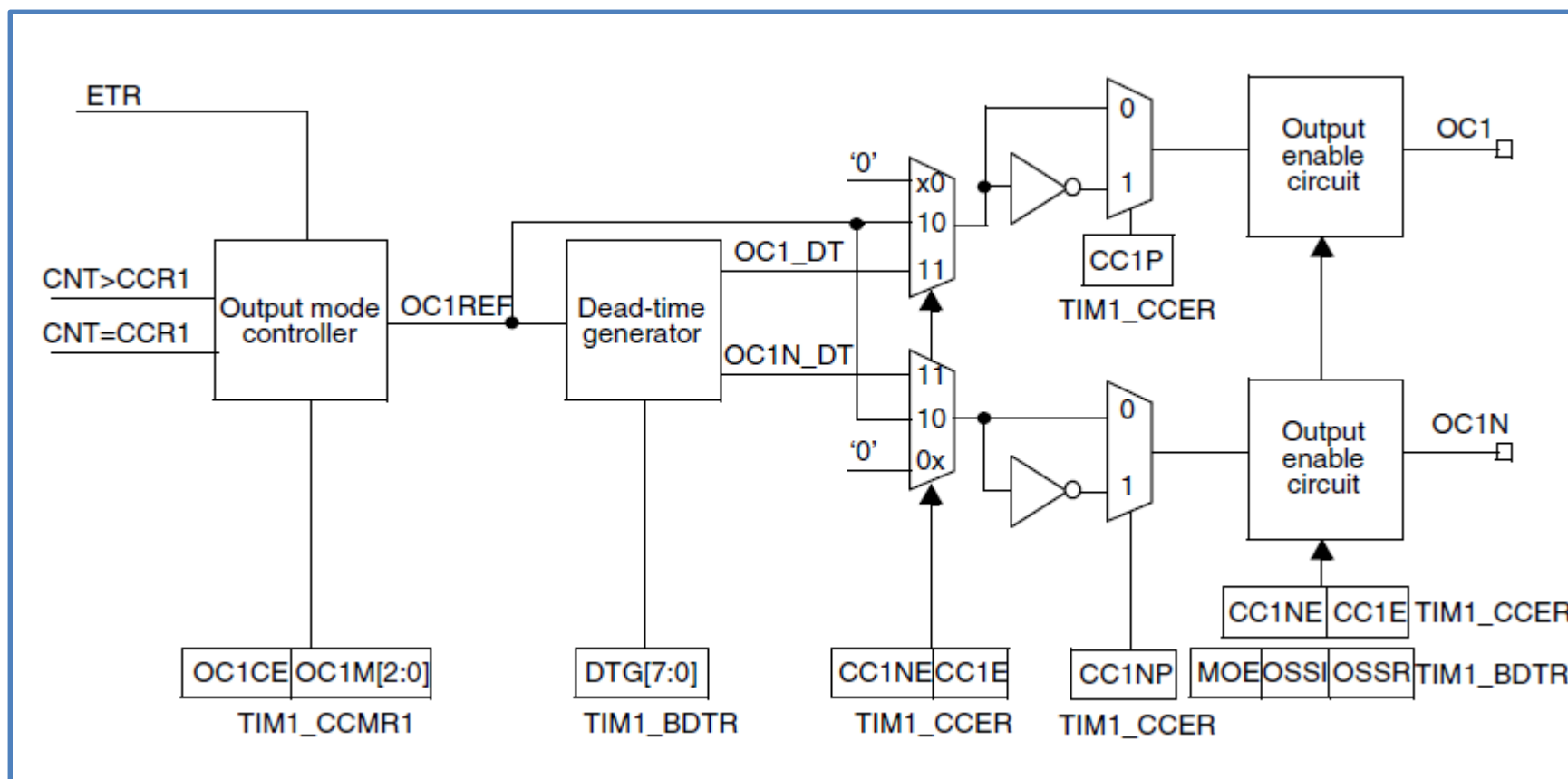
Converts a `tm` structure to a custom format human-readable string

RTC

```
#include "mbed.h"
int main() {
    set_time(1256729737); // Set RTC time to Wed, 28 Oct 2009
11:35:37
    while(1) {
        time_t seconds = time(NULL);
        printf("Time as seconds since January 1, 1970 = %d\n",
            seconds);
        printf("Time as a basic string = %s", ctime(&seconds));
        char buffer[32];
        strftime(buffer, 32, "%I:%M %p\n", localtime(&seconds));
        printf("Time as a custom formatted string = %s", buffer);
        wait(1);
    }
}
```

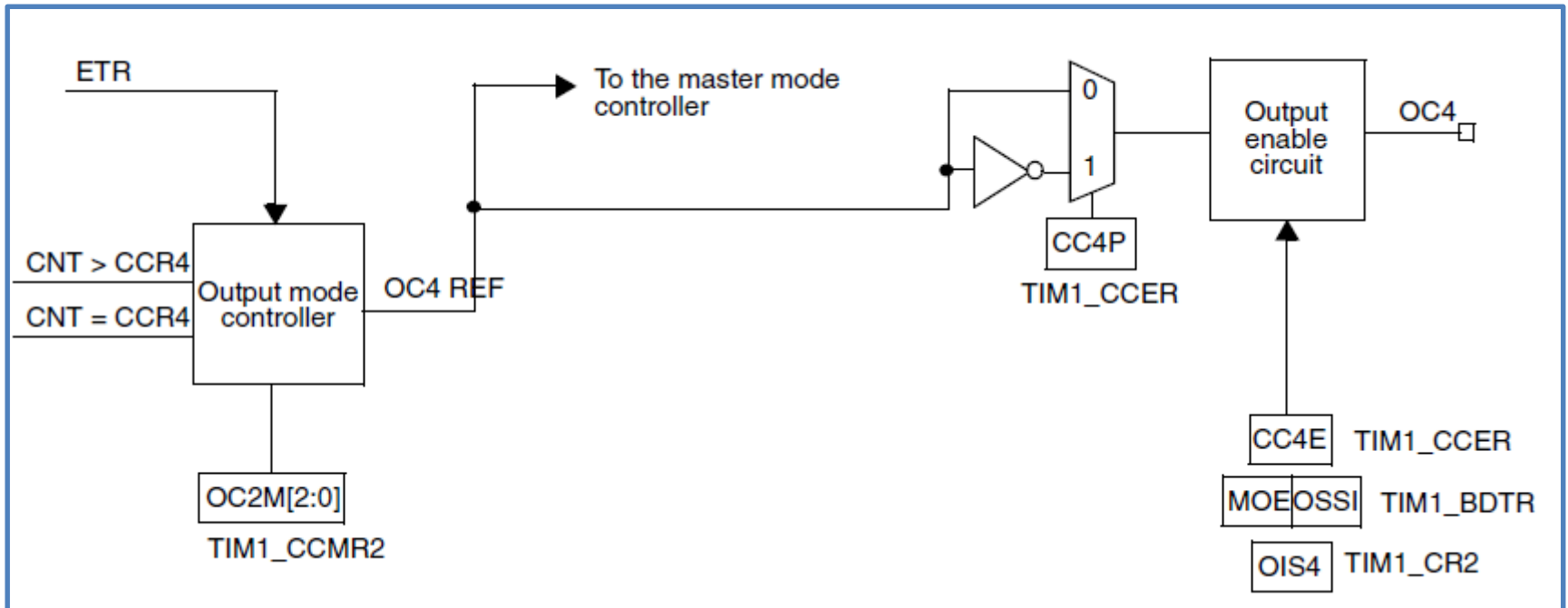

Izlazni stepen Output Capture jedinice

– Kanali 1,2,3



TIM1 - Kanal 4

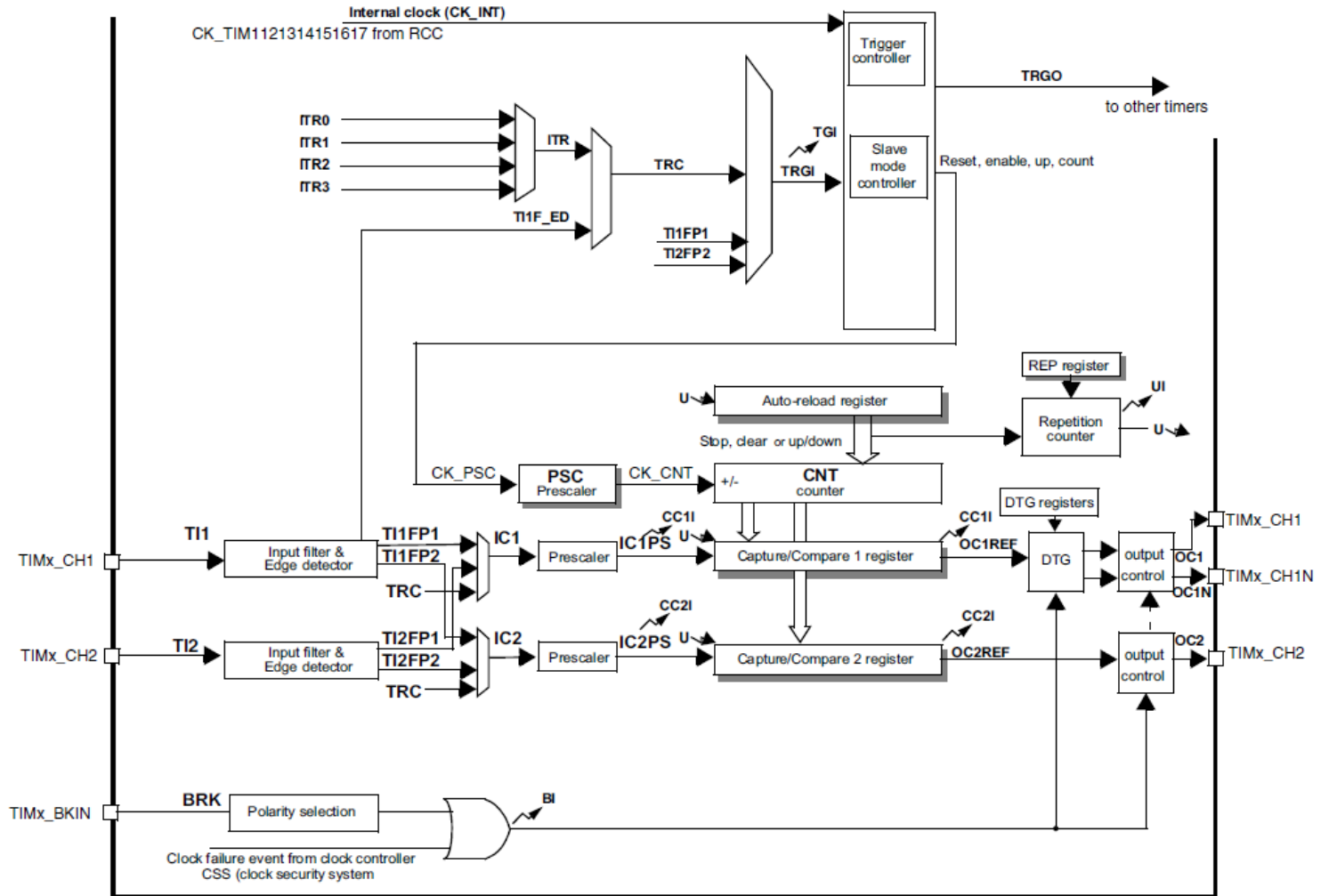
- Kanal 4 je pojednostavljene strukture jer se TIM1 obično koristi u trofaznim PWM generatorima u kojima se sedmi kanal koristi uglavnom za potrebe “kočenja”.



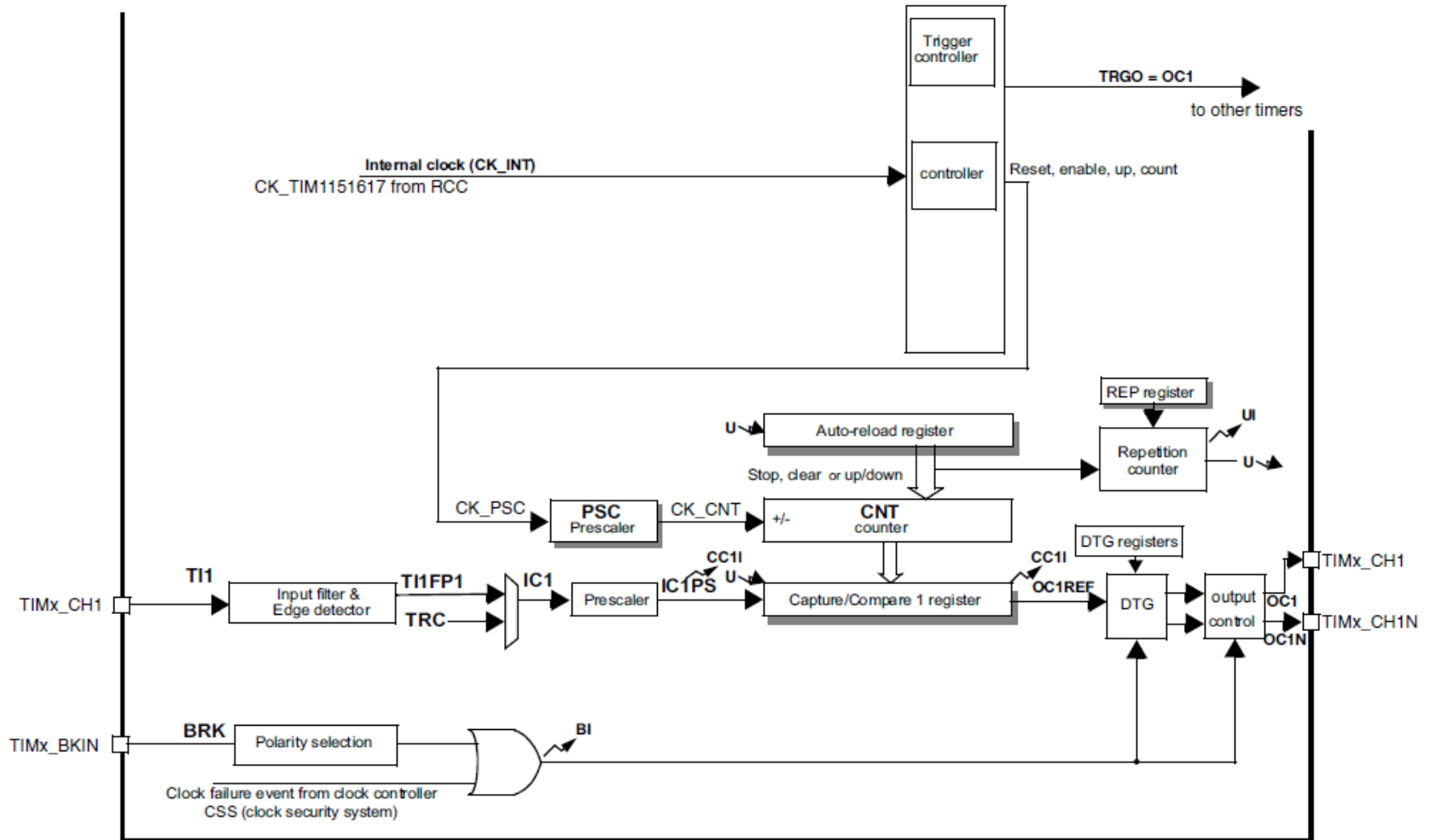
Tajmeri TIM15, TIM16, TIM17

- 16-bitni brojač na gore.
- 16-bitni preskaler za ulazni takt
- 1 (tim15) ili 2 (tim16, tim17) nezavisna kanala koji mogu da rade u izlaznom (output compare), ulazno (input capture), PWM ili pojedinačnom impulsnom modu.
- Mogućnost sinhronizacije sa ostalim tajmerima.
- Prekid zahtev za sledeće događaje:
 - Input capture
 - Output compare
 - Reload tajmera, inicijalizacija (softverska ili spoljašnja)
- Podržan je DMA prenos
- Uvek postoji jedan komplementarni izlaz.
- Brojač ponavljanja.

TIM15



TIM16, TIM17



Osnovni tajmeri (Basic Timer)

TIM6 i TIM7

- 16-bitni brojač na gore.
- 16-bitni preskaler za ulazni takt.
- Mogućnost startovanja DAC-a.
- Prekid i DMA zahtev overflow događaj.
- Podrжан je DMA prenos.
- Uvek postoji jedan komplementarni izlaz.
- Brojač ponavljanja.

