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.

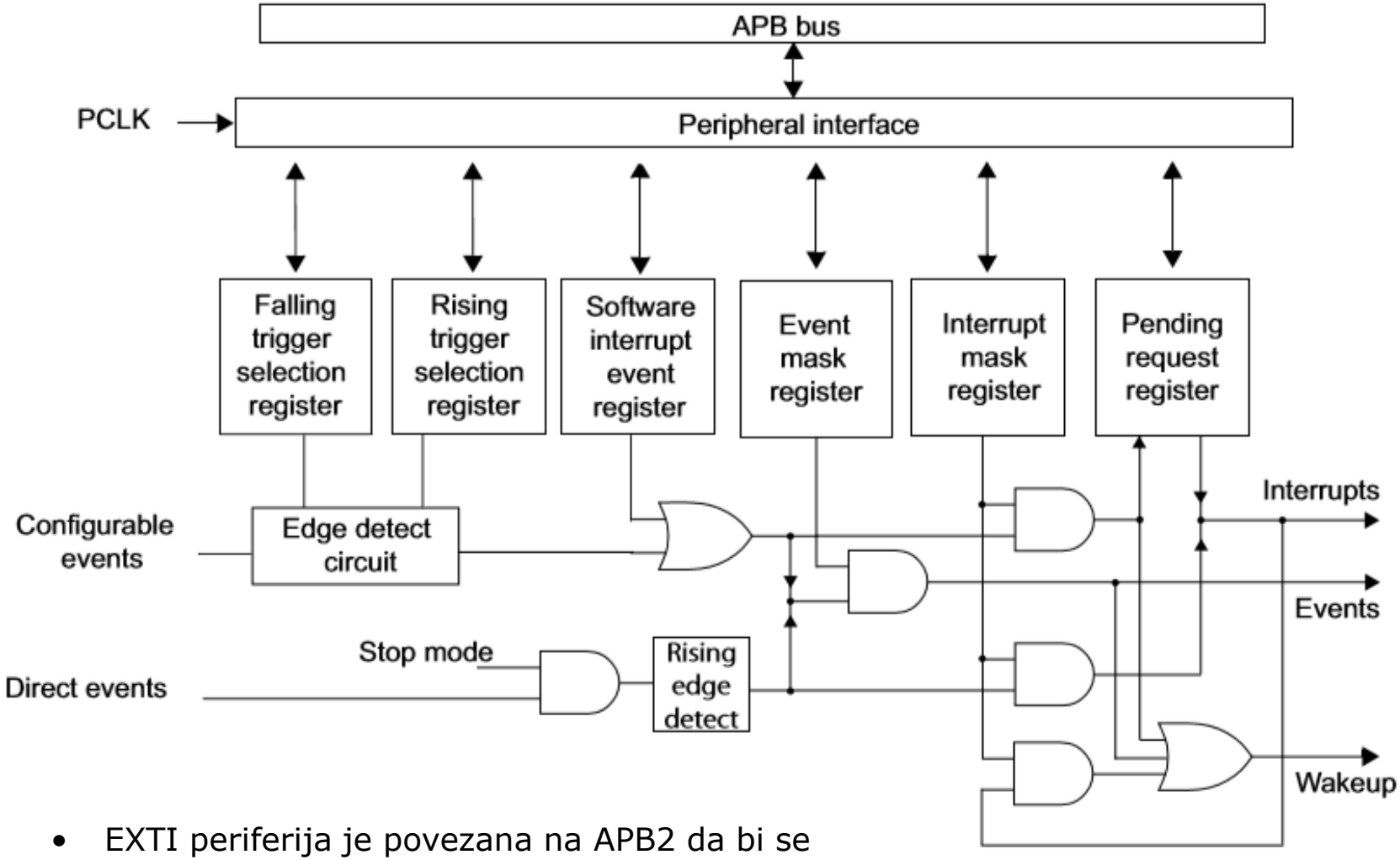
Konfigurabilne linije

- 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

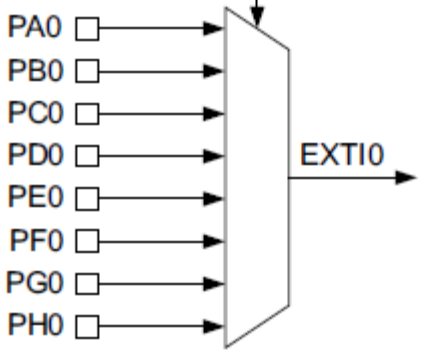
Figure 33. Configurable interrupt/event block diagram



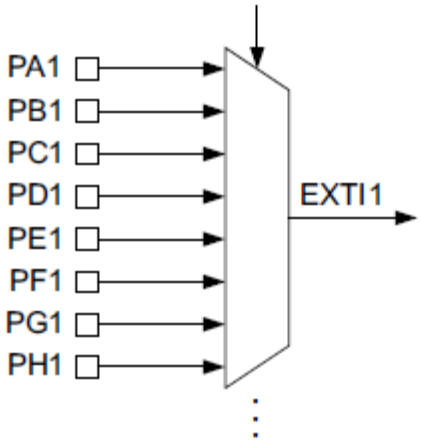
- EXTI periferija je povezana na APB2 da bi se skratilo vreme reagovanja (APB2 je u opštem slučaju brža od APB1 magistrale)

EXTI prekidi

EXTI0[3:0] bits in the SYSCFG_EXTICR1 register



EXTI1[3:0] bits in the SYSCFG_EXTICR1 register




```
DCD     PVD_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
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
EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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.

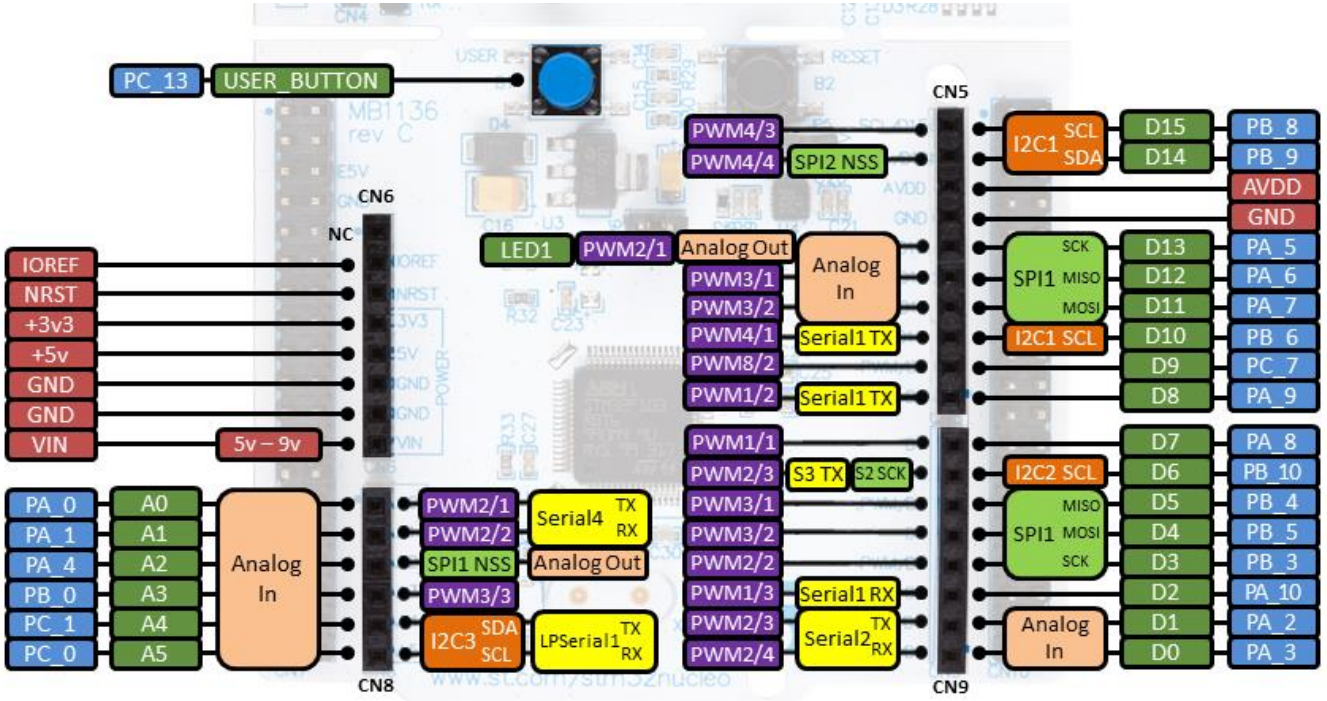
Bits 22:18 **FTx:** Falling trigger event configuration bit of line x (x = 22 to 18)
0: Falling trigger disabled (for Event and Interrupt) for input line
1: Falling trigger enabled (for Event and Interrupt) for input line

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

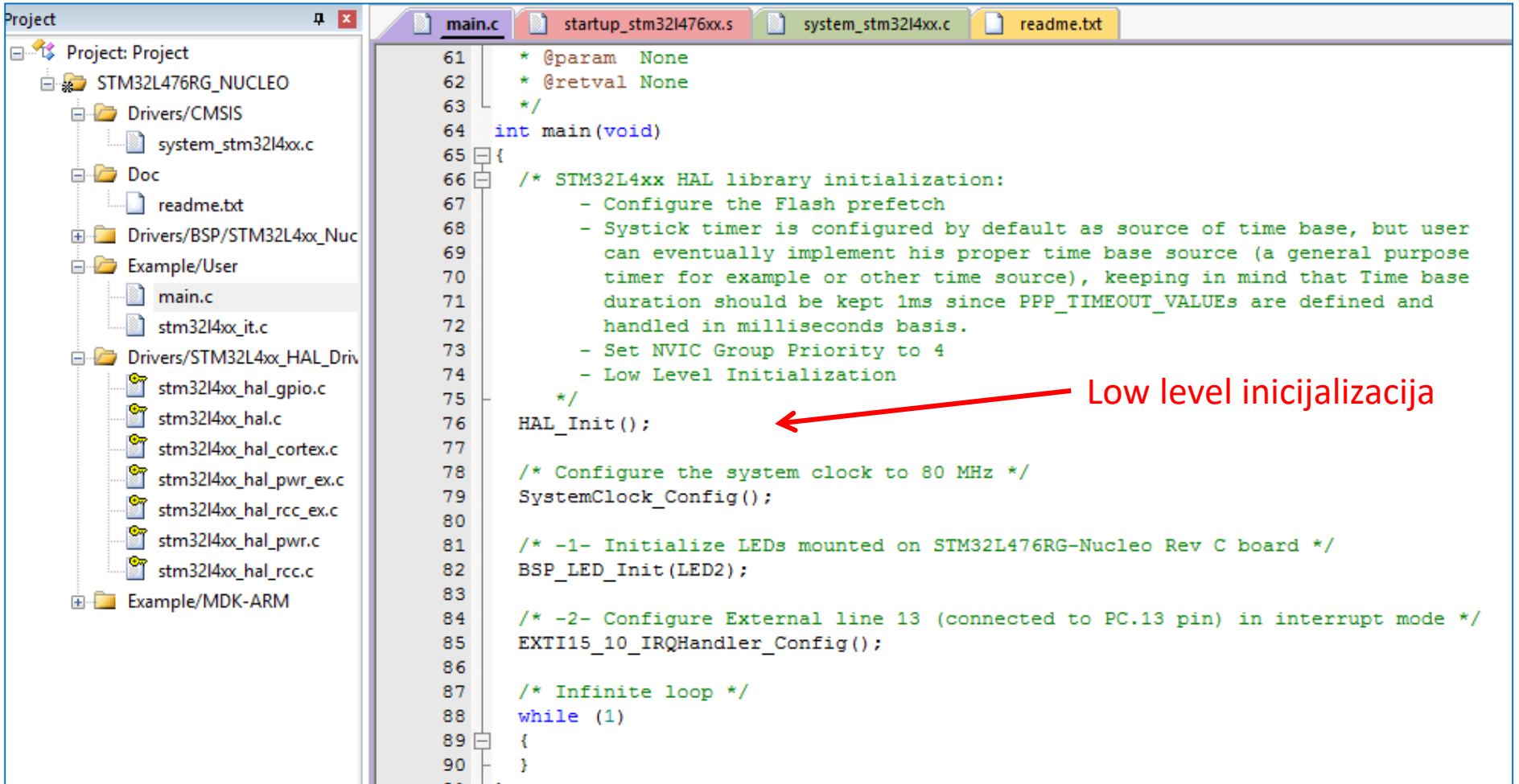
STM CUBE
Projekat GPIO_EXTI

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

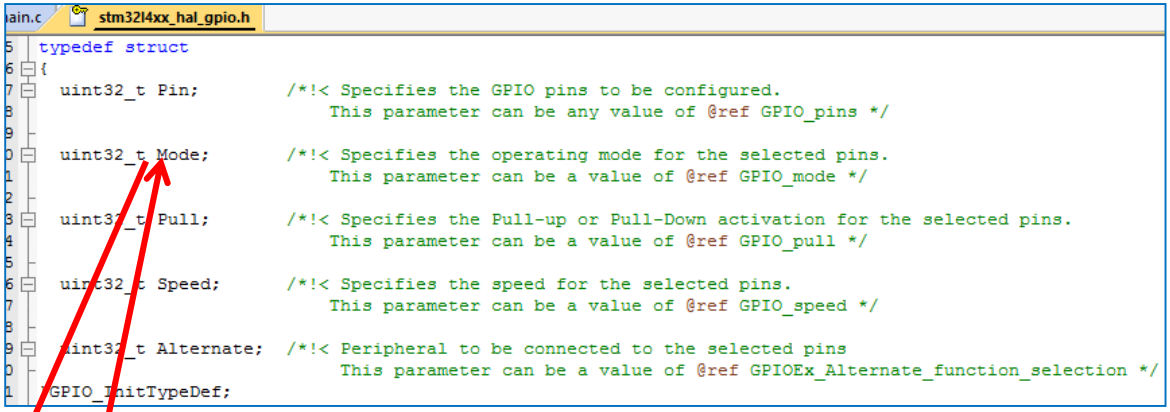


STM CUBE

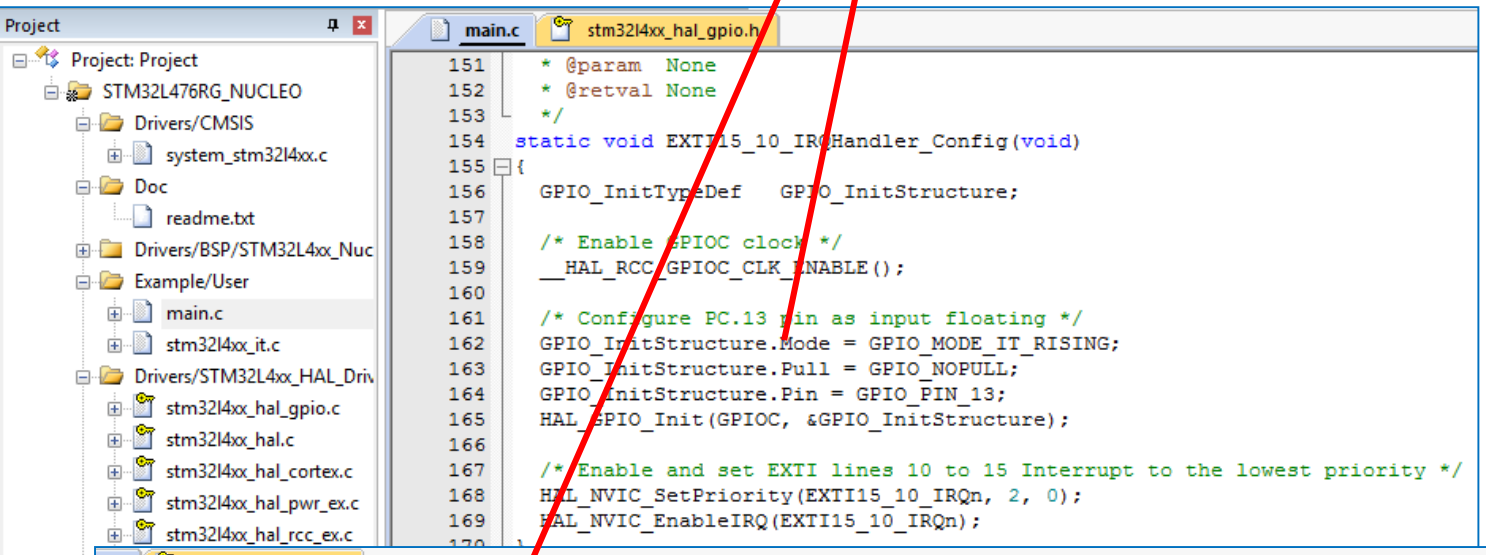
Projekat GPIO_EXTI



Inicijalizacija EXTI prekida



```
typedef struct
{
  uint32_t Pin;        /*!< Specifies the GPIO pins to be configured.
                            This parameter can be any value of @ref GPIO_pins */

  uint32_t Mode;       /*!< Specifies the operating mode for the selected pins.
                            This parameter can be a value of @ref GPIO_mode */

  uint32_t Pull;       /*!< Specifies the Pull-up or Pull-Down activation for the selected pins.
                            This parameter can be a value of @ref GPIO_pull */

  uint32_t Speed;      /*!< Specifies the speed for the selected pins.
                            This parameter can be a value of @ref GPIO_speed */

  uint32_t Alternate;  /*!< Peripheral to be connected to the selected pins
                            This parameter can be a value of @ref GPIOEx_Alternate_function_selection */
}GPIO_InitTypeDef;
```



```
Project
Project: Project
STM32L476RG_NUCLEO
Drivers/CMSIS
system_stm32l4xx.c
Doc
readme.txt
Drivers/BSP/STM32L4xx_Nuc
Example/User
main.c
stm32l4xx_it.c
Drivers/STM32L4xx_HAL_Driv
stm32l4xx_hal_gpio.c
stm32l4xx_hal.c
stm32l4xx_hal_cortex.c
stm32l4xx_hal_pwr_ex.c
stm32l4xx_hal_rcc_ex.c
```

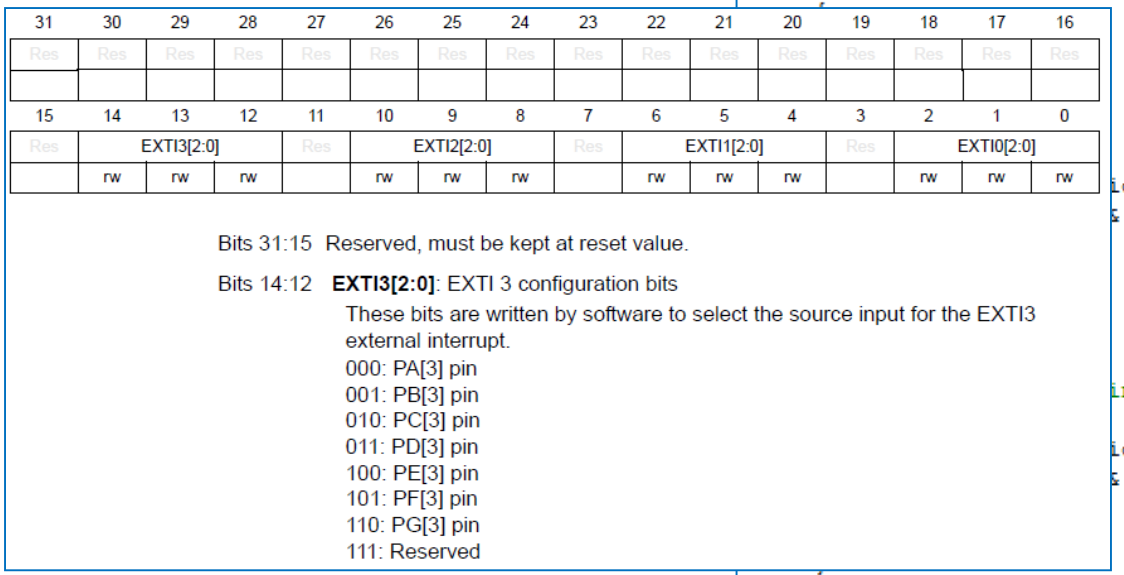
main.c | stm32l4xx_hal_gpio.h

```
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);
```

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()



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

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)

iocurrent);
& GPIO_MODE_EVT) == GPIO_MODE_EVT)

ing edge configuration */

iocurrent);
& RISING_EDGE) == RISING_EDGE)

  EXTI->RTSR1 = temp;
```

Reakcija na prekid!

startup_stm32l476xx.s | stm32l4xx_it.c | main.c | stm32l4xx_hal_gpio.h | stm

```
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_stm32l476xx.s | stm32l4xx_it.c | main.c | stm32l4xx_hal_gp

```
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

artup_stm32l476xx.s | stm32l4xx_it.c | main.c | stm32l4xx_hal_gpio.h | stm32l4xx_hal_gpio.c

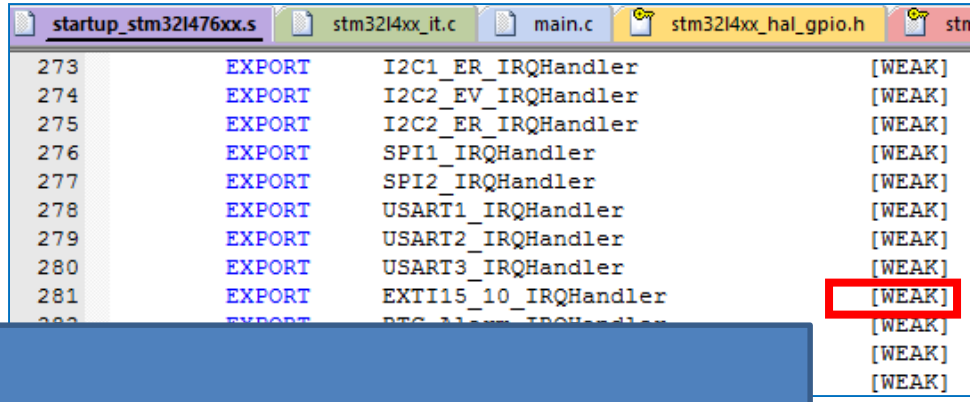
```
  * @brief  Handle EXTI interrupt request.
  * @param  GPIO_Pin: Specifies the port pin connected to corresponding E
  * @retval None
  */
void HAL_GPIO_EXTI_IRQHandler(uint16_t GPIO_Pin)
{
  /* EXTI line interrupt detected */
  if(__HAL_GPIO_EXTI_GET_IT(GPIO_Pin) != RESET)
  {
    __HAL_GPIO_EXTI_CLEAR_IT(GPIO_Pin);
    HAL_GPIO_EXTI_Callback(GPIO_Pin);
  }
}

/**
  * @brief  EXTI line detection callback.
  * @param  GPIO_Pin: Specifies the port pin connected to corresponding E
  * @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 nee
           the HAL_GPIO_EXTI_Callback could be implemented in the user fi
   */
}
```

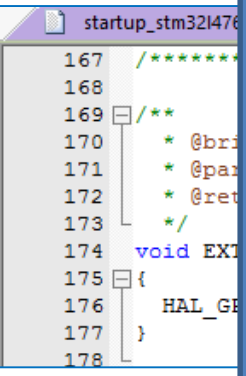
Korisničke funkcije

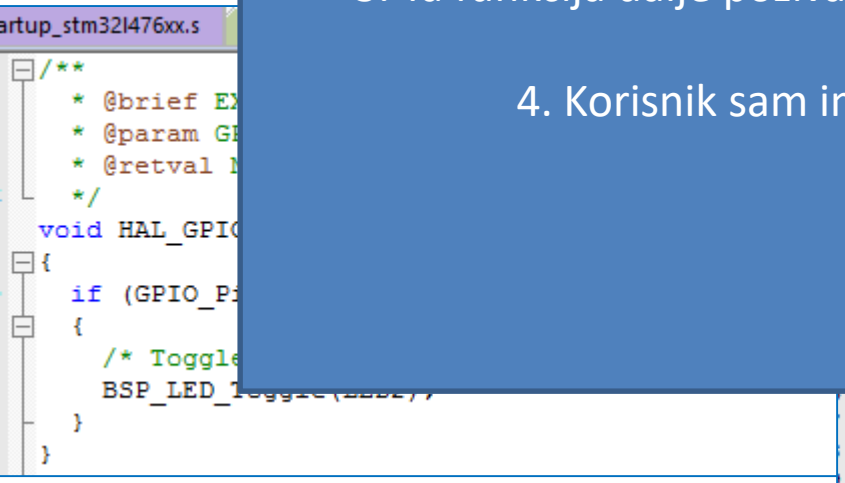
artup_stm32l476xx.s | stm32l4xx_it.c | main.c | stm32l4xx_hal_g

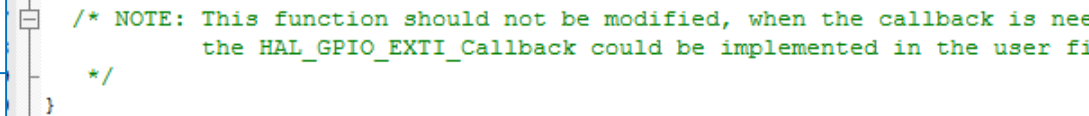
```
/**
  * @brief EXTI line detection callbacks
  * @param GPIO_Pin: Specifies the pins connected
  * @retval None
  */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
  if (GPIO_Pin == GPIO_PIN_13)
  {
    /* Toggle LED2 */
    BSP_LED_Toggle(LED2);
  }
}
```

Reakcija na prekid!







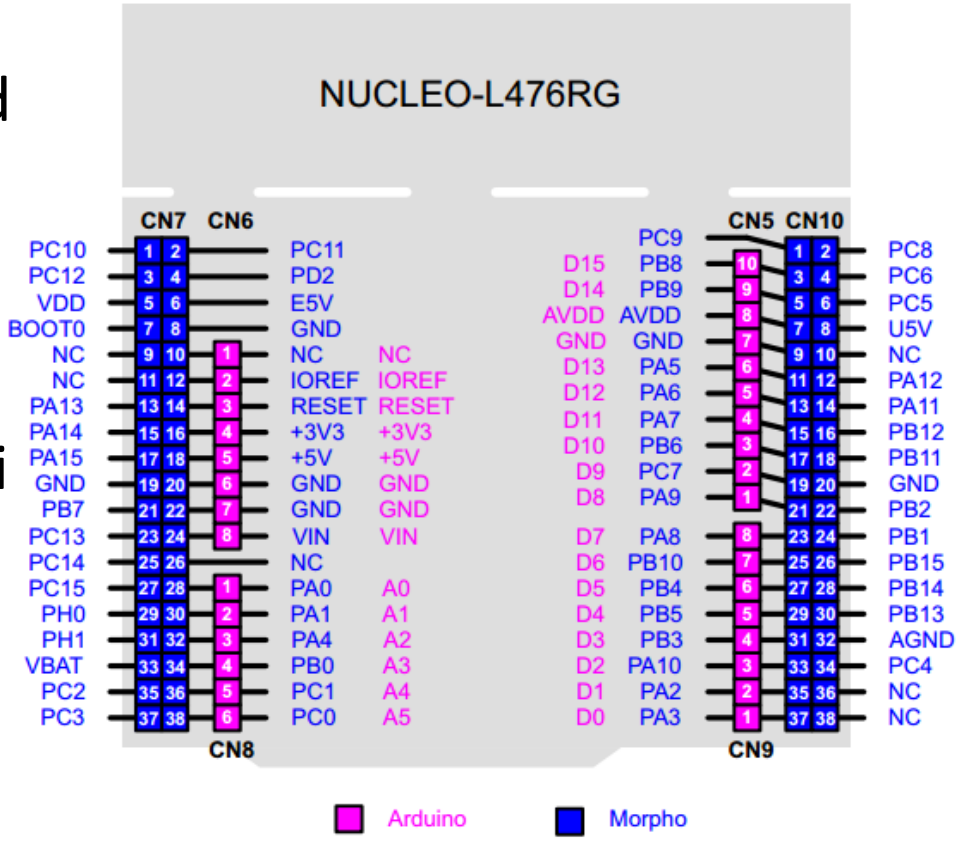


Principi HAL drajvera kada su u pitanju prekidi su sledeći:

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.

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.

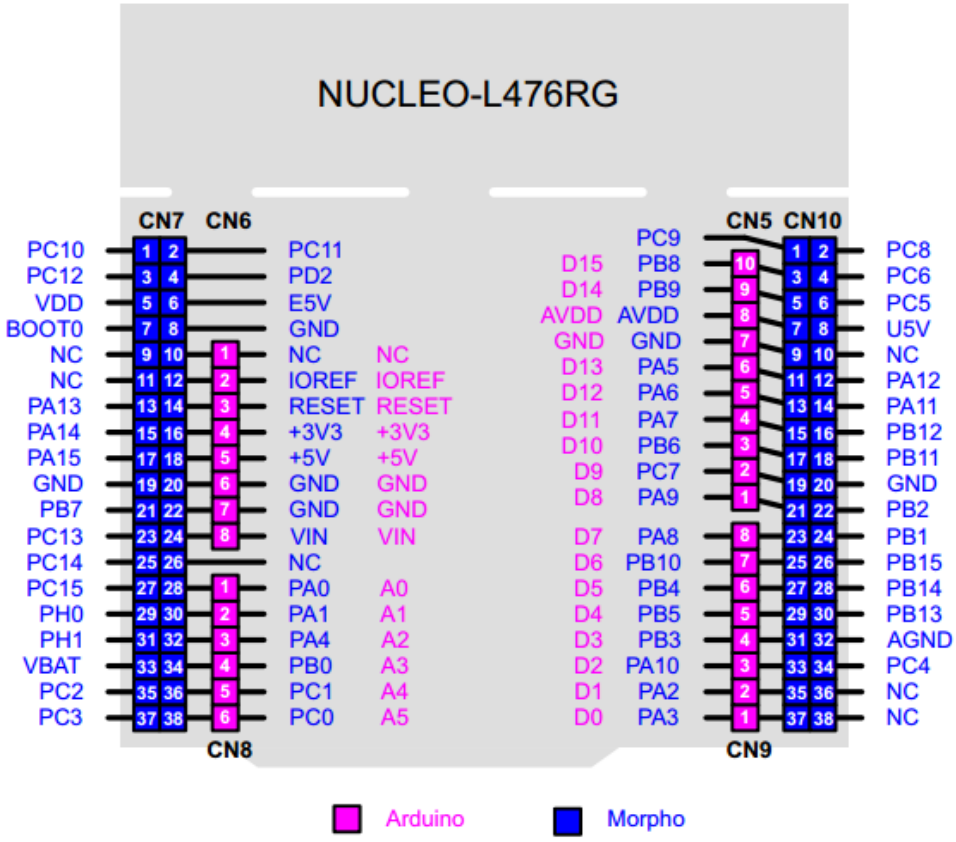
void mode (PinMode pull)
Set the input pin mode.

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 ahitektura 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 iterfejsi (**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: komlementarni 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 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
		-	-	-	-	-	-	TIM19
	32-bit	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

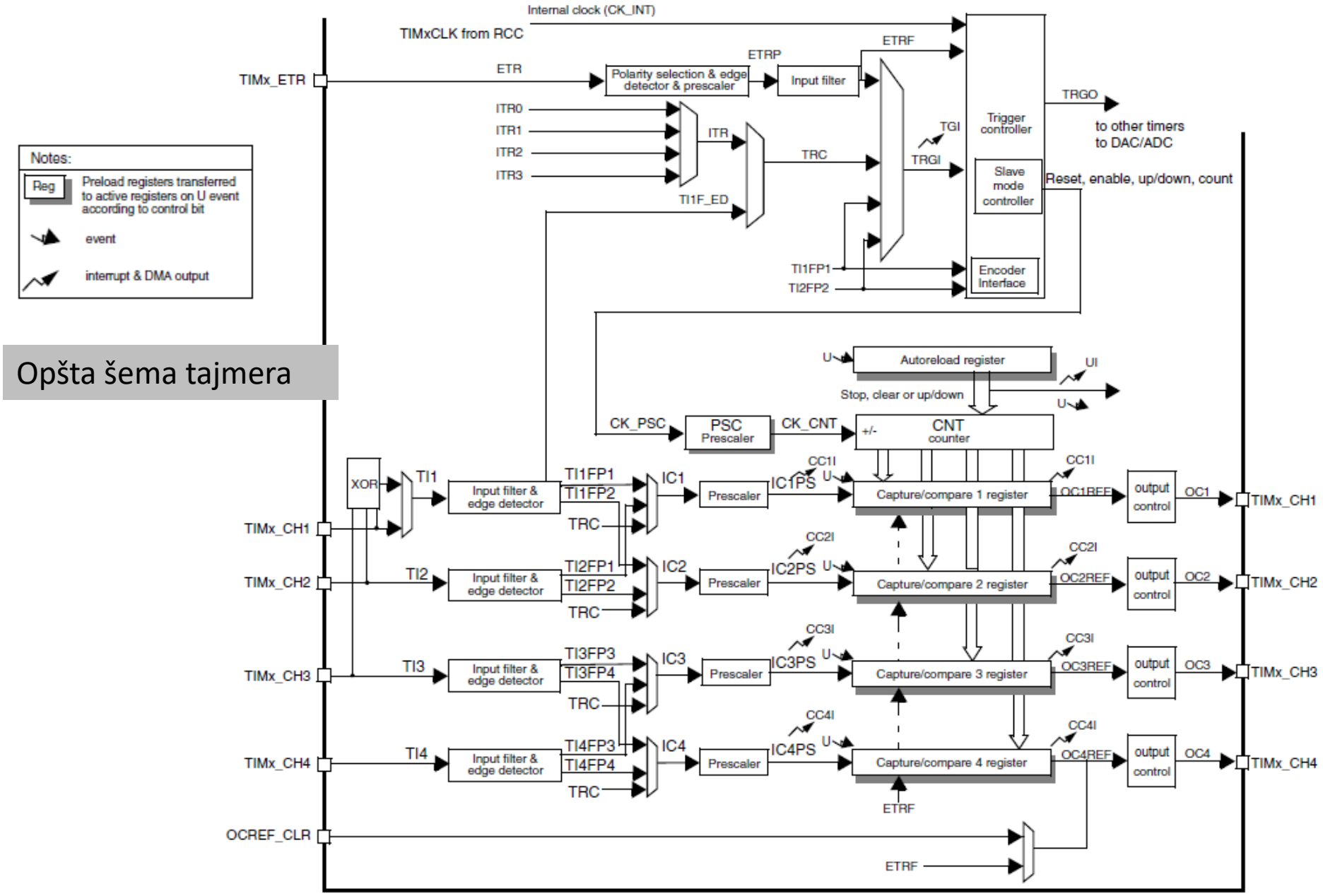
Timer type	Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/ compare channels	Complementary outputs
Advanced control	TIM1, TIM8	16-bit	Up, down, Up/down	Any integer between 1 and 65536	Yes	4	3
General-purpose	TIM2, TIM5	32-bit	Up, down, Up/down	Any integer between 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

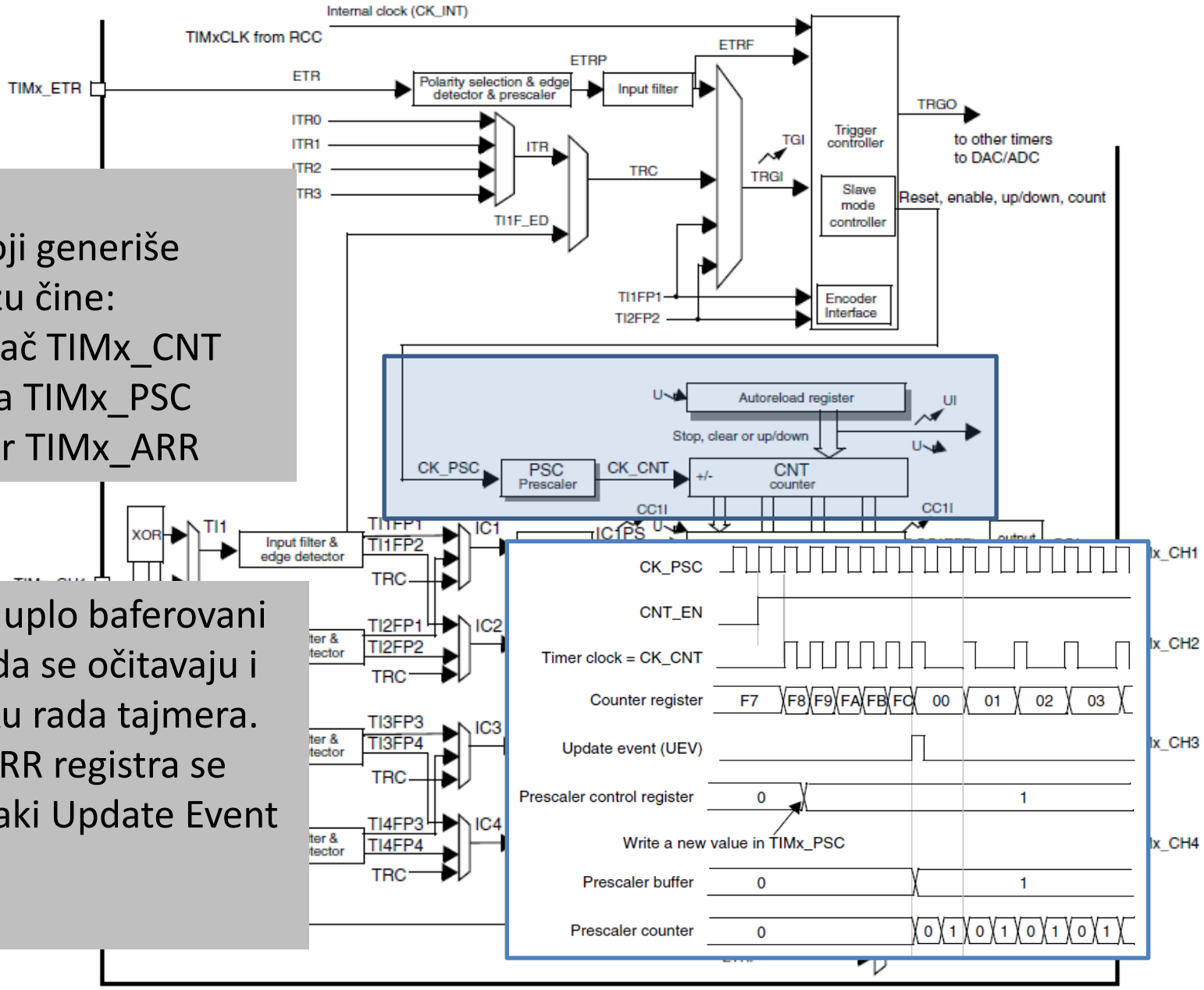


Opšta šema tajmera

Vremenska baza

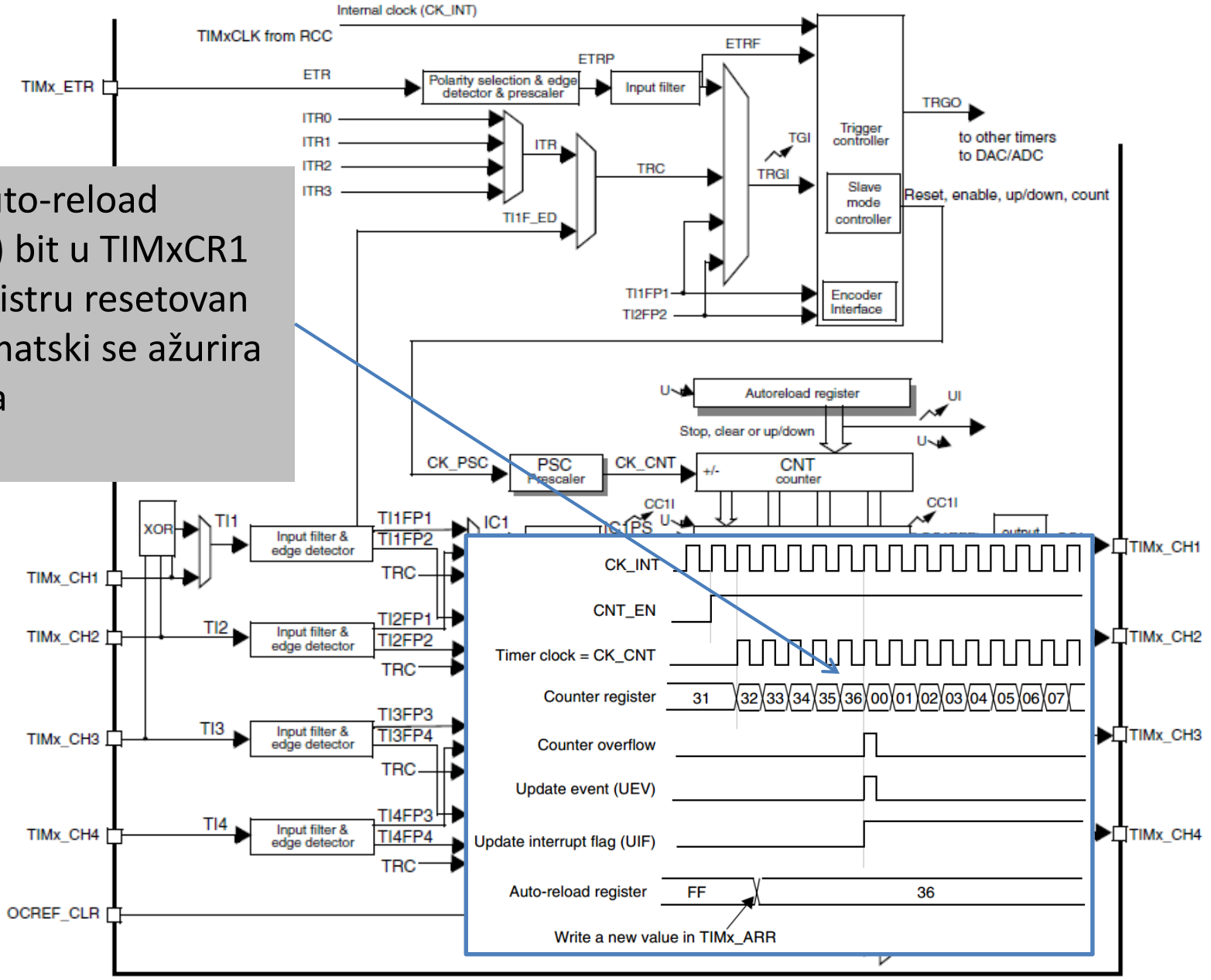
Vremenska baza
Deo tajmera koji generiše vremensku bazu čine:
-Tajmerski brojač TIMx_CNT
-Preskaler takta TIMx_PSC
-Reload registar TIMx_ARR

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



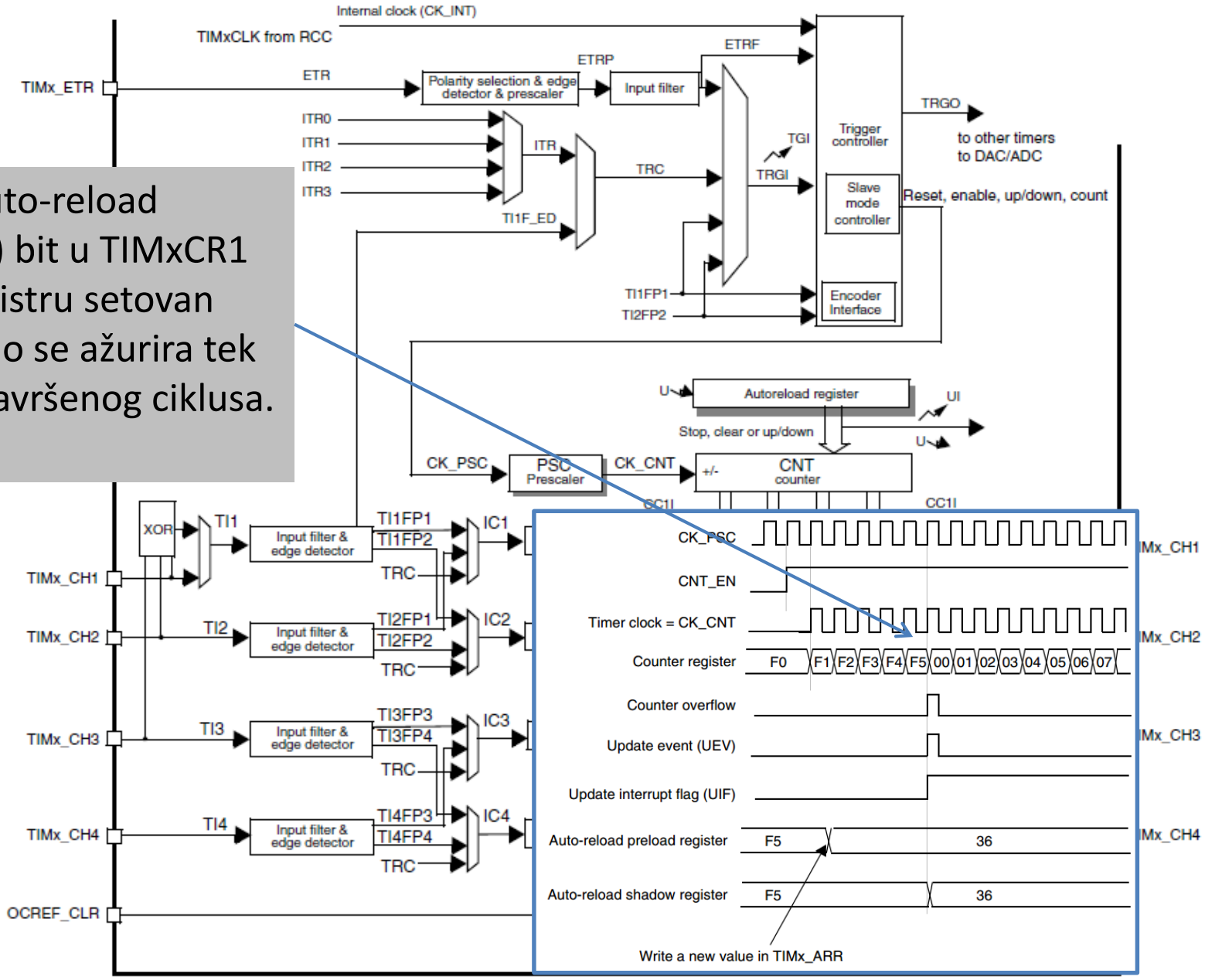
Brojač na gore – UP counting

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

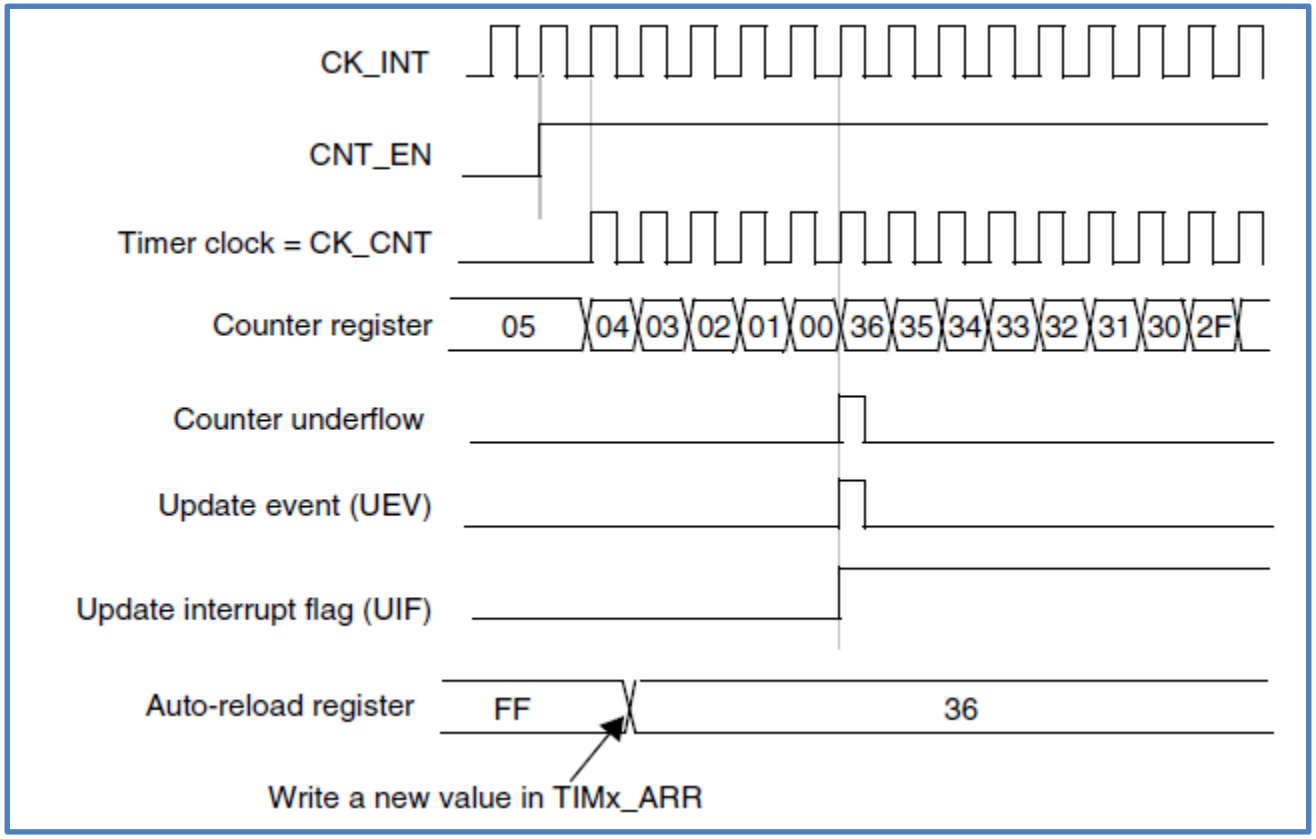


Brojač na gore – UP counting

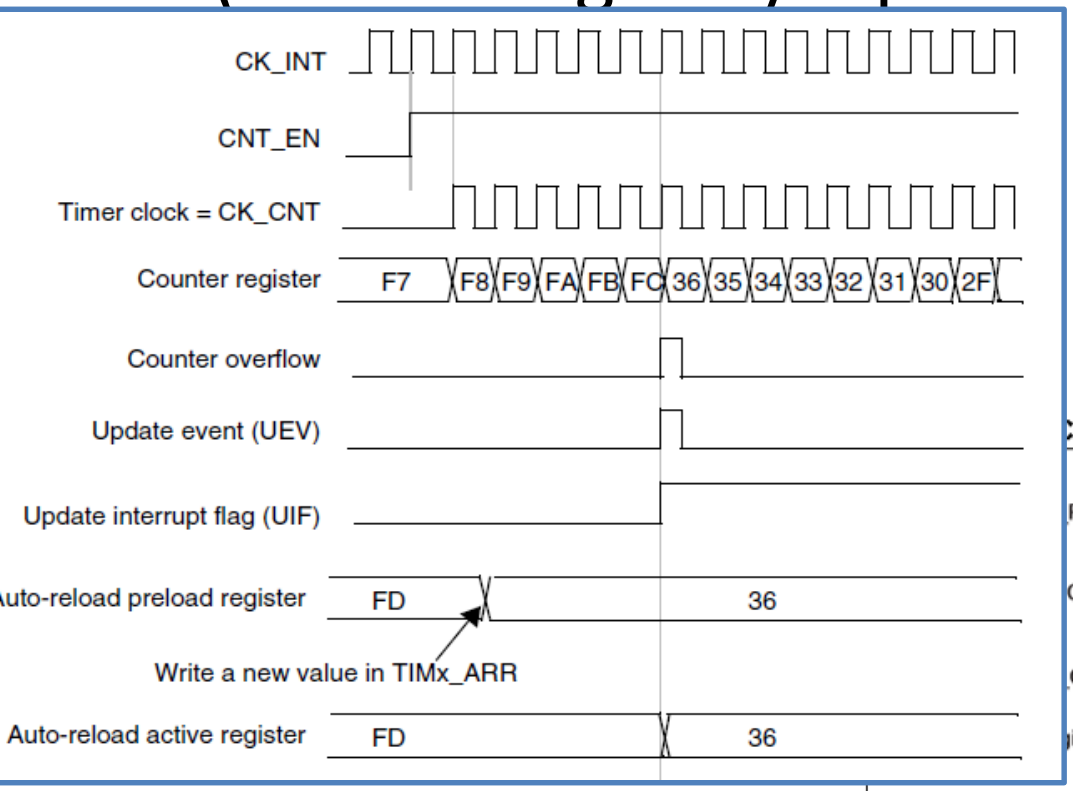
Ako je ARPE (auto-reload preload enable) bit u TIMxCR1 kontrolnom registru setovan (ARPE=1) moduo se ažurira tek nakon jednog završenog ciklusa.

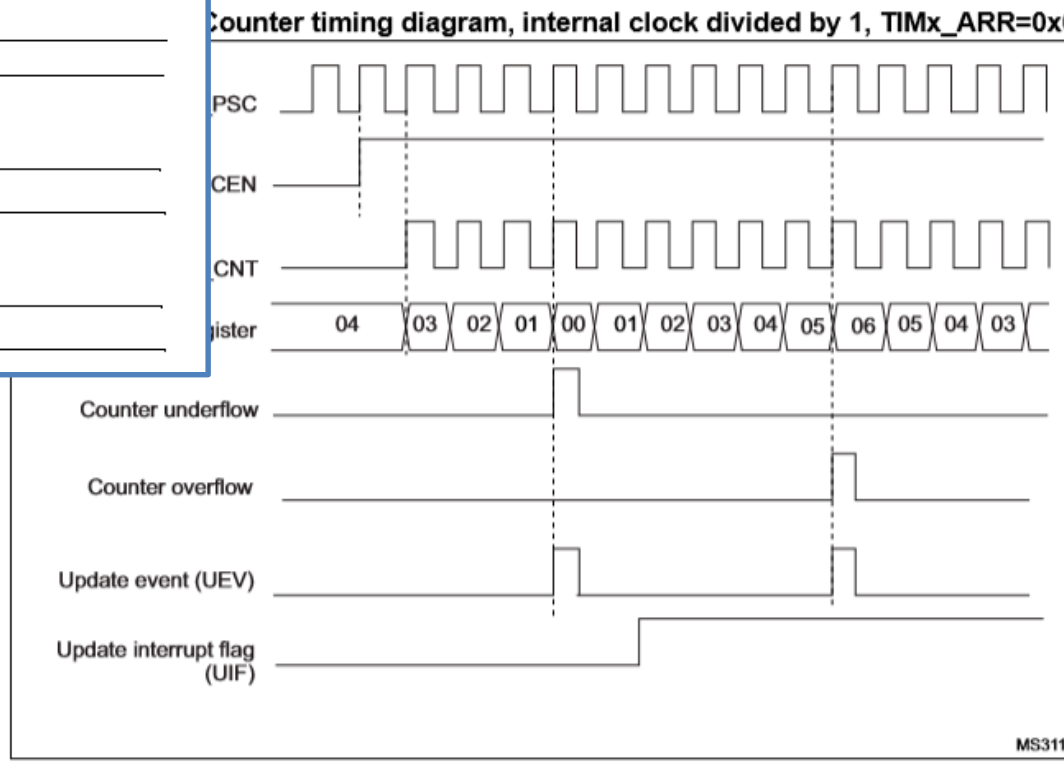


Downcounting mode

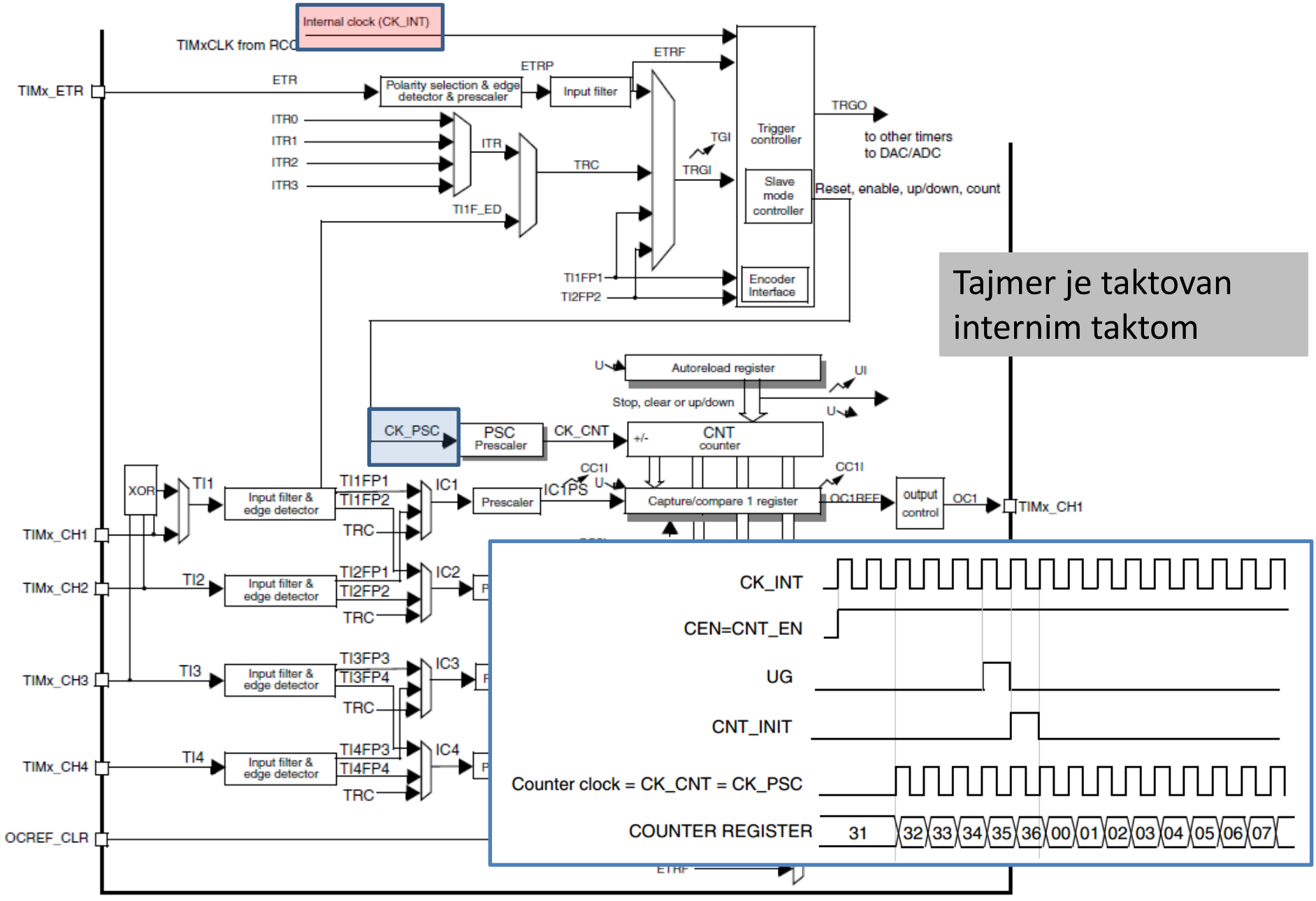


Brojač gore-dole
(Center-aligned / Up-down counting mode)

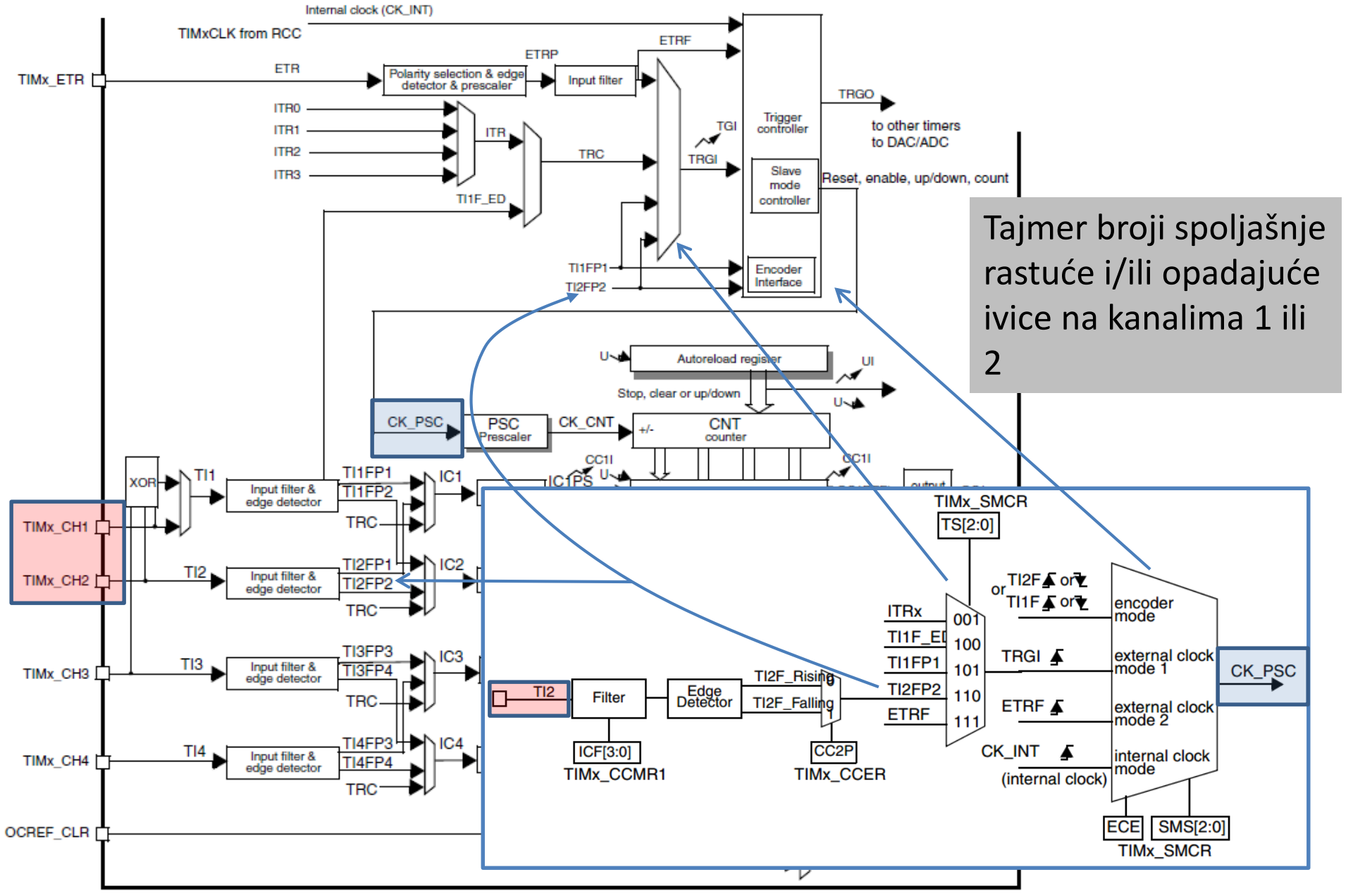




Taktovanje - Internal mode



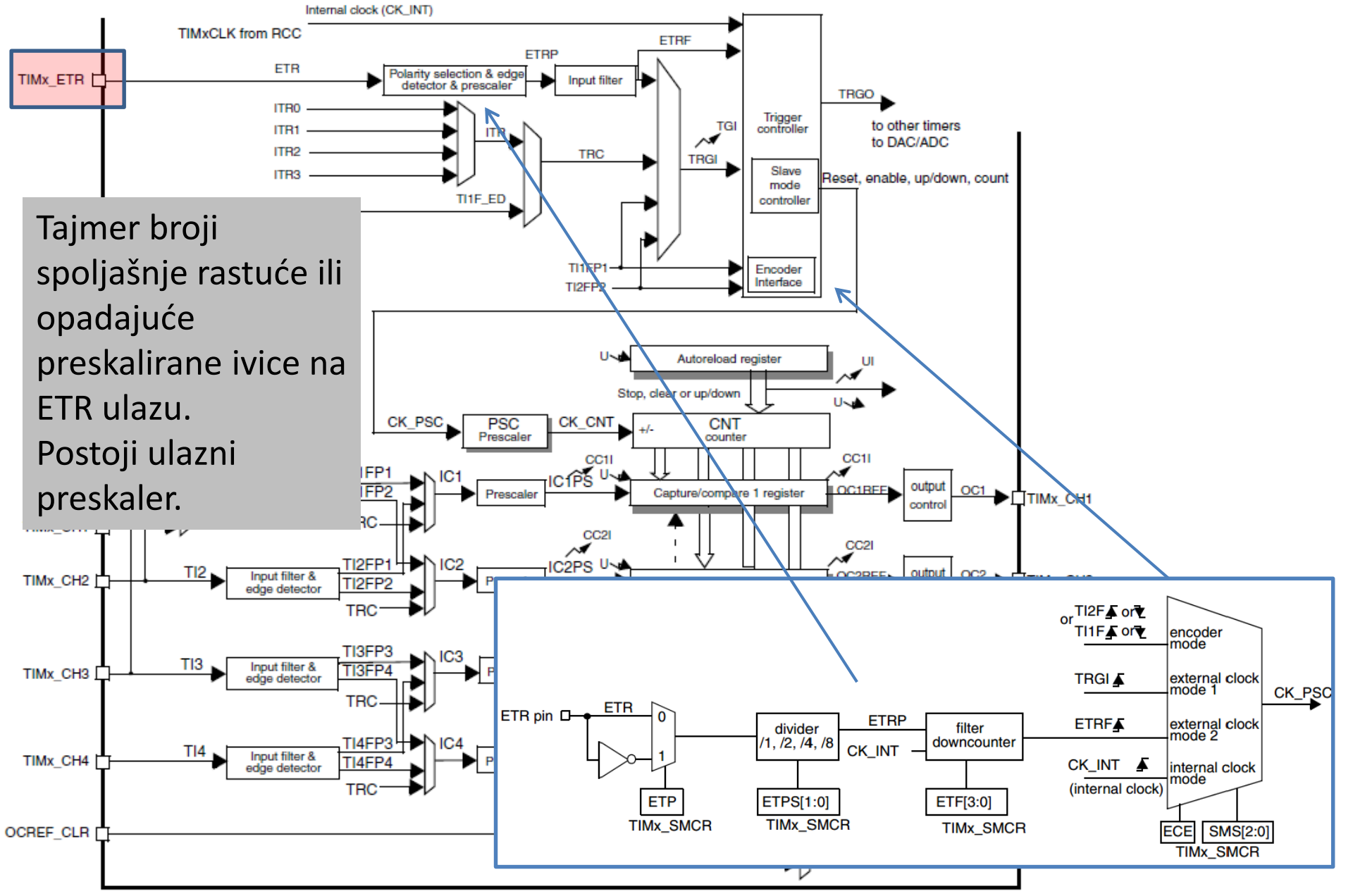
External mode 1



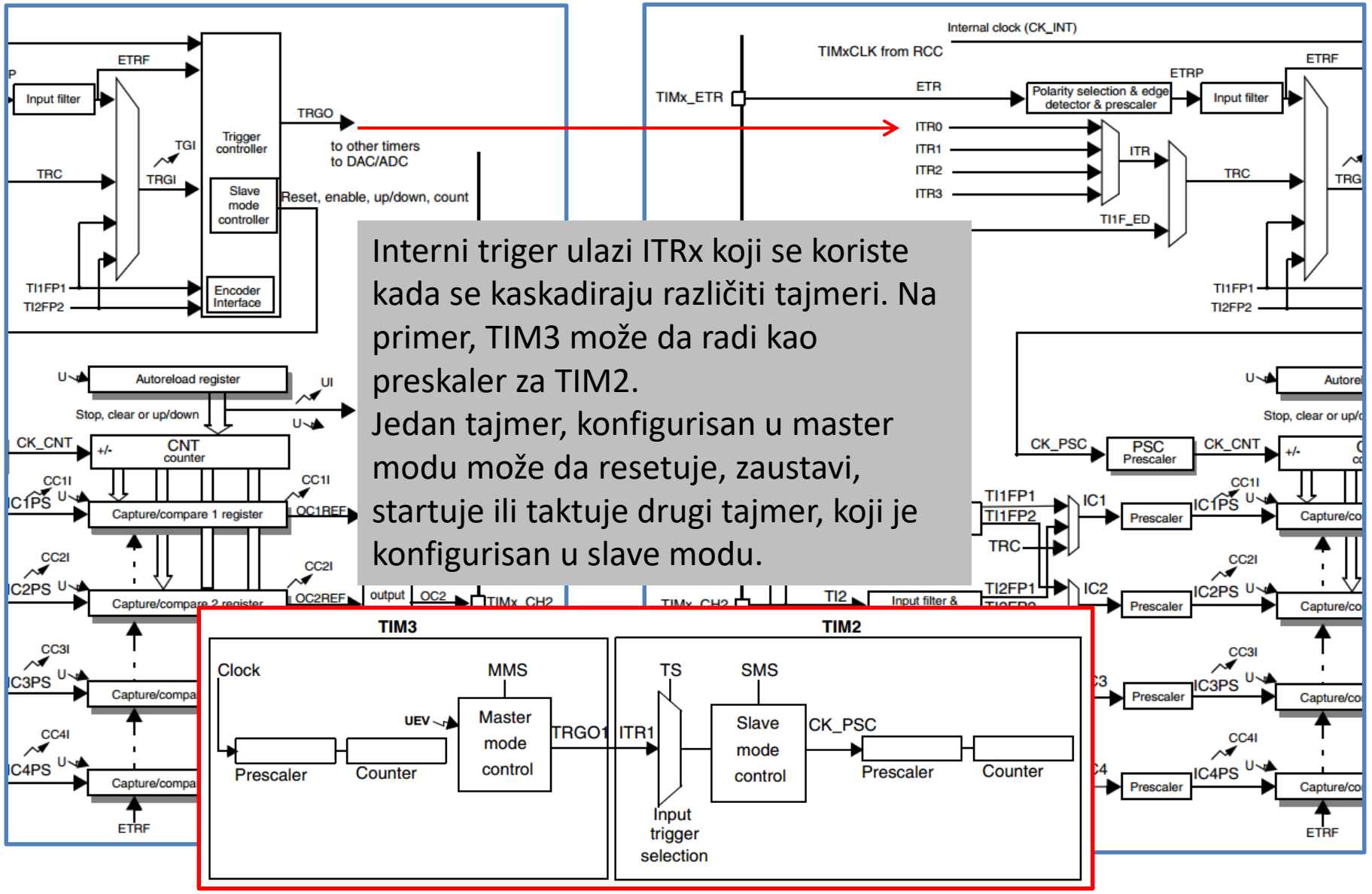
Tajmer broji spoljašnje rastuće i/ili opadajuće ivice na kanalima 1 ili 2

External mode 2

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



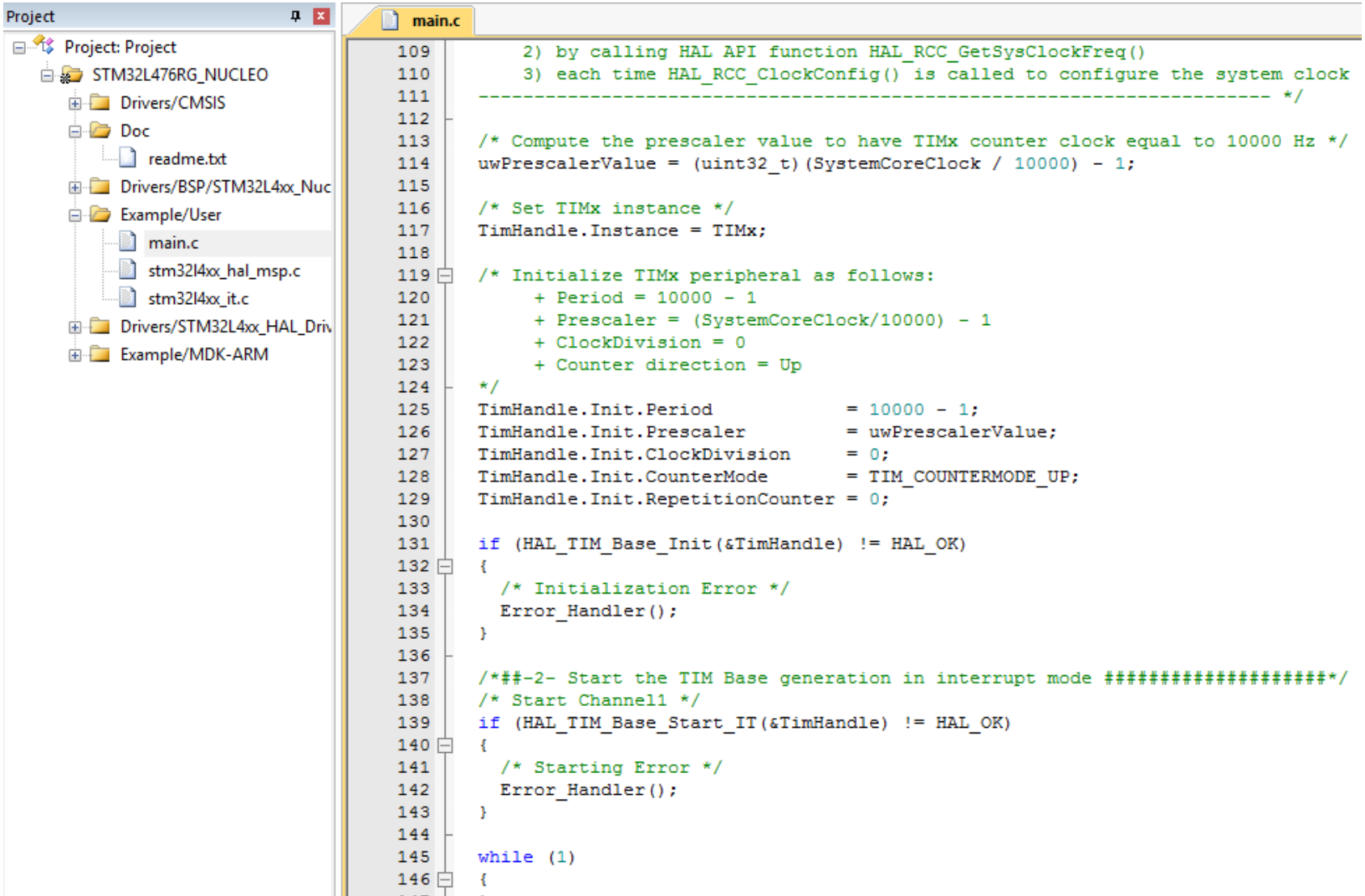
Kaskadna veza tajmera



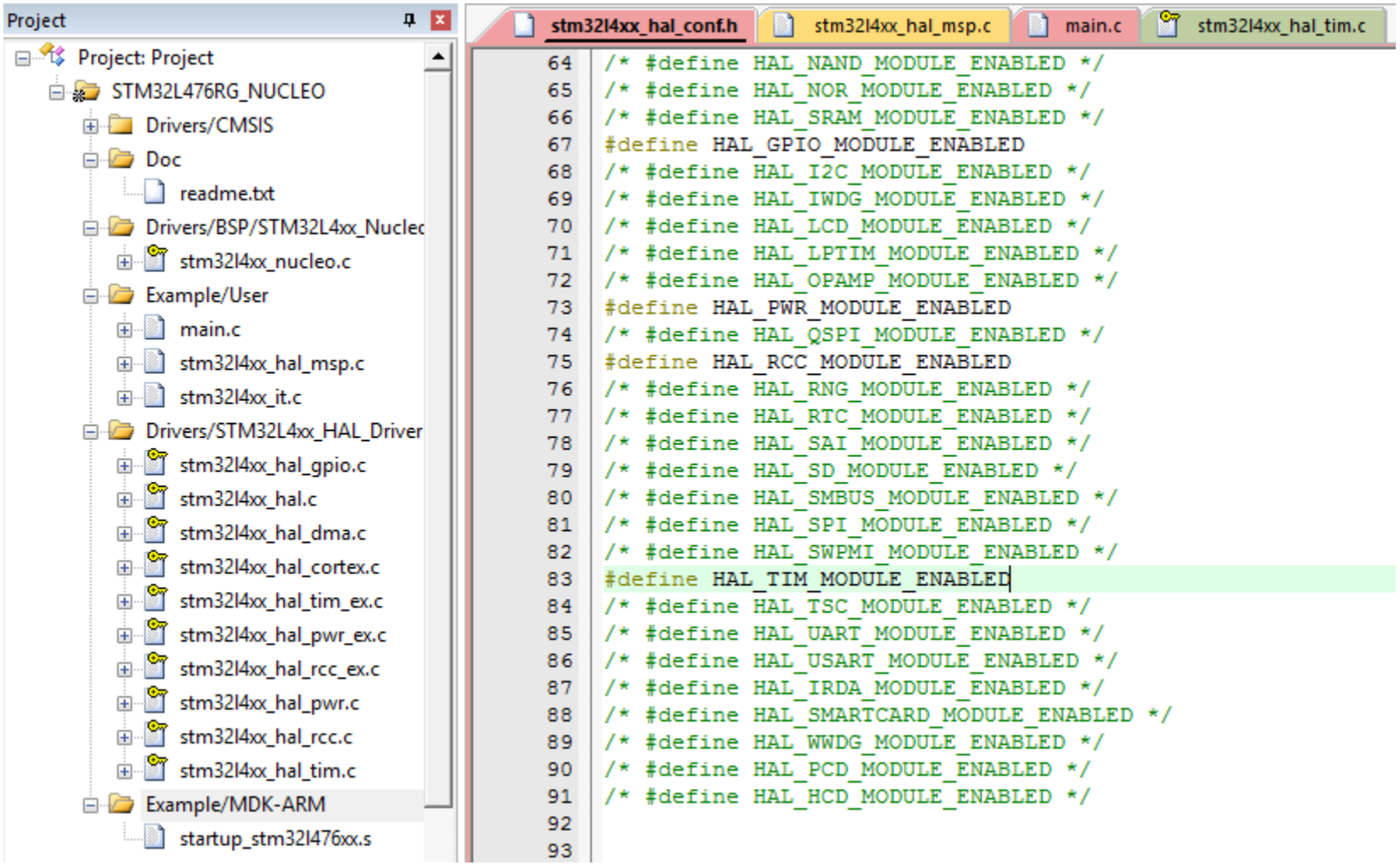
Interni triger ulazi ITRx koji se koriste kada se kaskadiraju različiti tajmeri. Na primer, TIM3 može da radi kao preskaler za TIM2.
Jedan tajmer, konfigurisan u master modu može da resetuje, zaustavi, startuje ili taktuje drugi tajmer, koji je konfigurisan u slave modu.

STM CUBE

Projekat TIM_TimeBase



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



Šta to piše u drajverskim fajlovima?

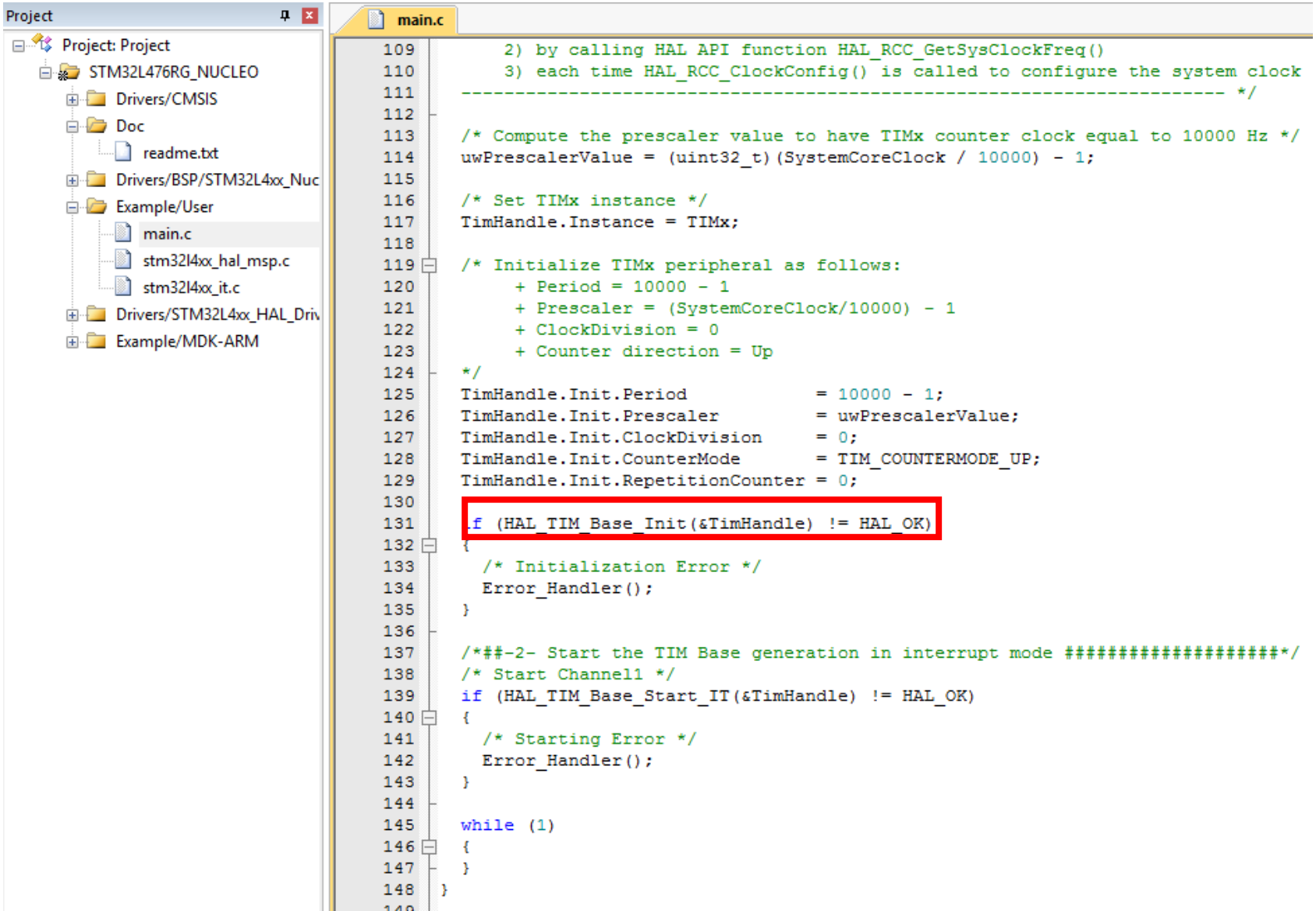
stm32l4xx_hal_tim.c

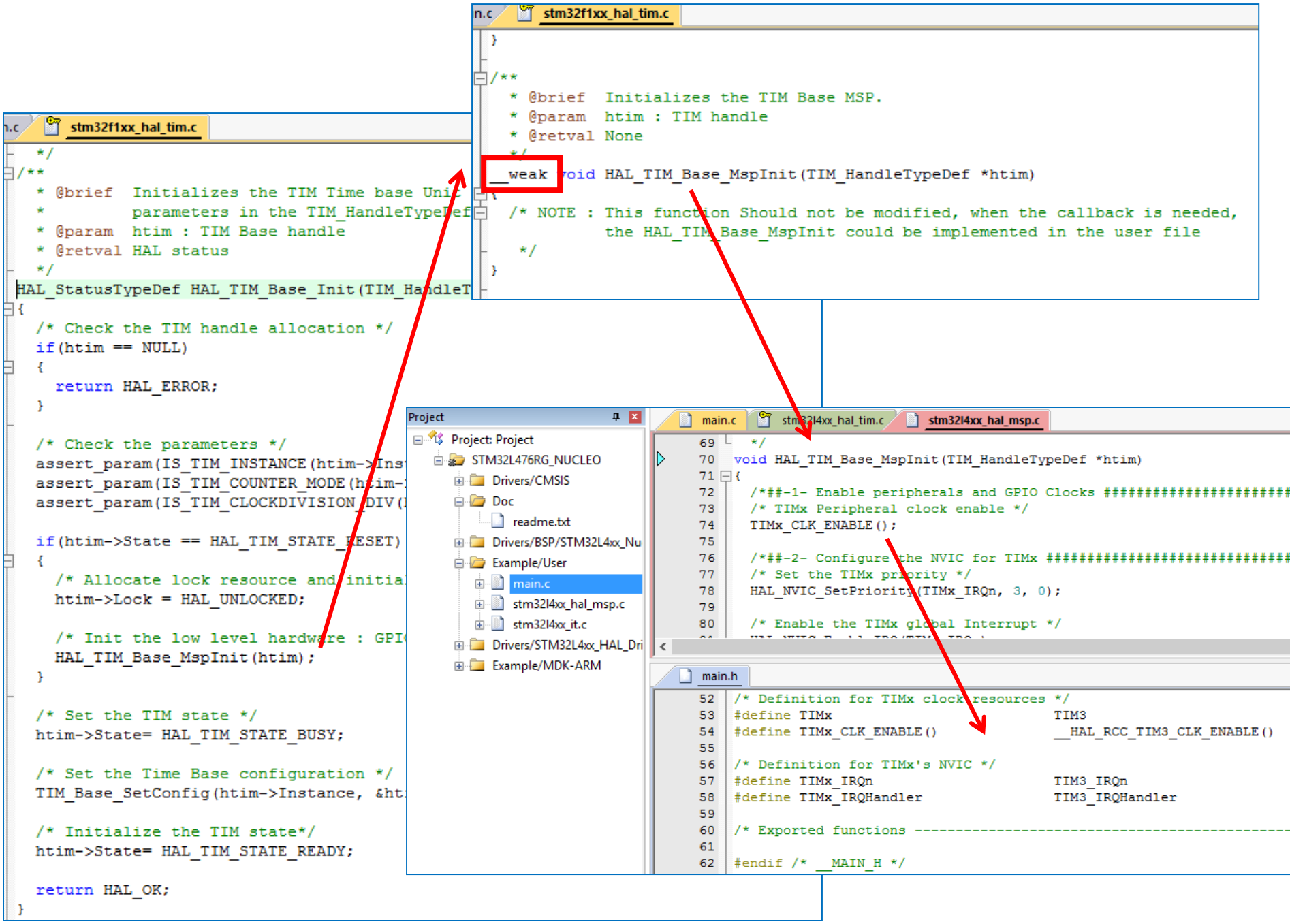
```
/**
  ******************************************************************************
  * @file    stm32l4xx_hal_tim.c
  * @author  MCD Application Team
  * @version V1.4.0
  * @date    26-February-2016
  * @brief   TIM HAL module driver.
  *          This file provides firmware functions to manage the following
  *          functionalities of the Timer (TIM) peripheral:
  *           + Time Base Initialization
  *           + Time Base Start
  *           + Time Base Start Interruption
  *           + Time Base Start DMA
  *           + Time Output Compare/PWM Initialization
  *           + Time Output Compare/PWM Channel Configuration
  *           + Time Output Compare/PWM  Start
  *           + Time Output Compare/PWM  Start Interruption
  *           + Time Output Compare/PWM Start DMA
  *           + Time Input Capture Initialization
  *           + Time Input Capture Channel Configuration
  *           + Time Input Capture Start
  *           + Time Input Capture Start Interruption
  *           + Time Input Capture Start DMA
  *           + Time One Pulse Initialization
  *           + Time One Pulse Channel Configuration
  *           + Time One Pulse Start
  *           + Time Encoder Interface Initialization
  *           + Time Encoder Interface Start
  *           + Time Encoder Interface Start Interruption
  *           + Time Encoder Interface Start DMA
```

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

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



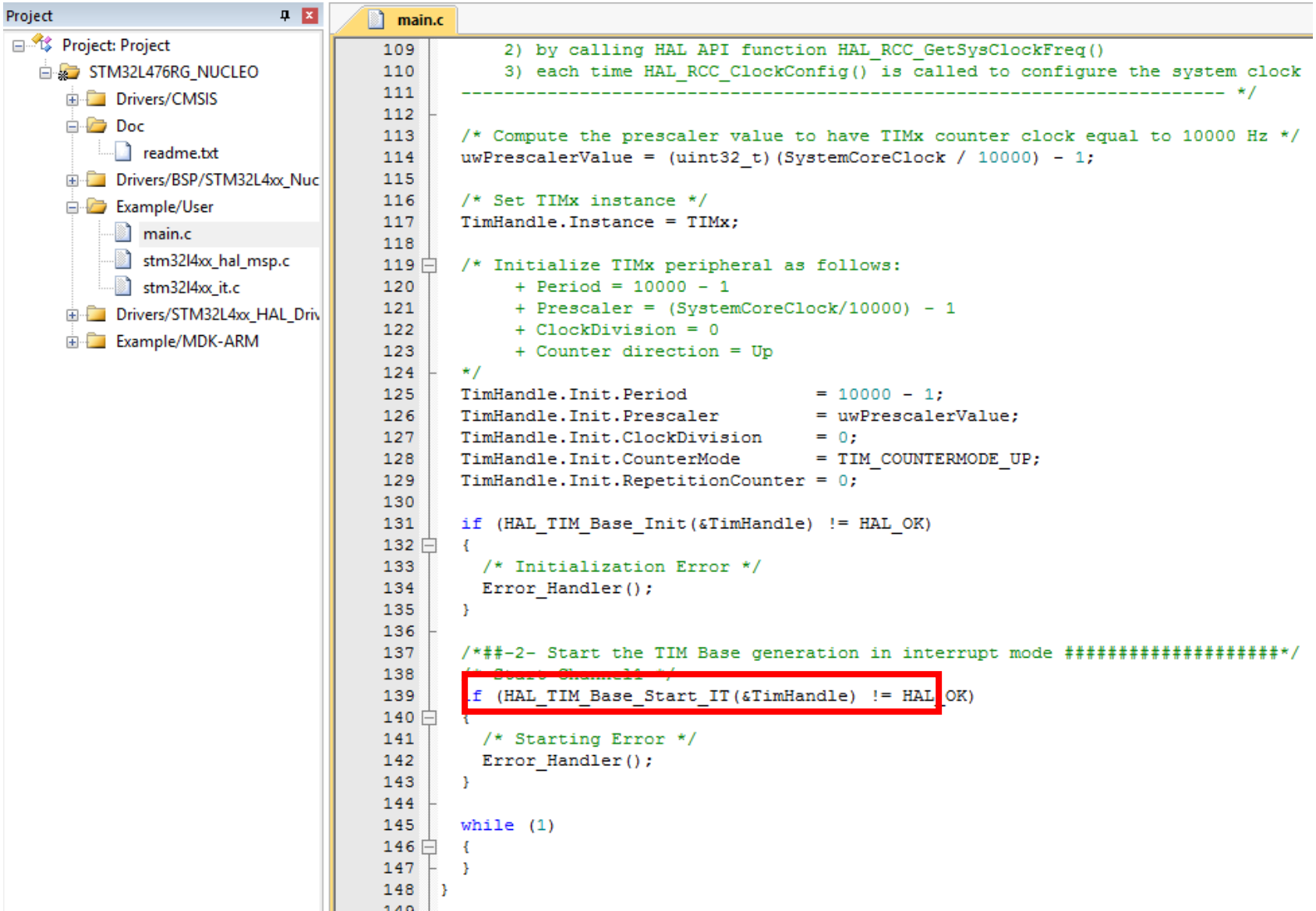

stm32f1xx_hal_tim.c
*/
/**
* @brief Initializes the TIM Time base Unit
* parameters in the TIM_HandleTypeDef
* @param htim : TIM Base handle
* @retval HAL status
*/
HAL_StatusTypeDef HAL_TIM_Base_Init(TIM_HandleT
{
/* Check the TIM handle allocation */
if(htim == NULL)
{
return HAL_ERROR;
}
/* Check the parameters */
assert_param(IS_TIM_INSTANCE(htim->Ins
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;
}
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
*/
}
Project
Project: Project
STM32L476RG_NUCLEO
Drivers/CMSIS
Doc
readme.txt
Drivers/BSP/STM32L4xx_Nu
Example/User
main.c
stm32l4xx_hal_msp.c
stm32l4xx_it.c
Drivers/STM32L4xx_HAL_Dri
Example/MDK-ARM
main.c
stm32l4xx_hal_tim.c
stm32l4xx_hal_msp.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 */
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()

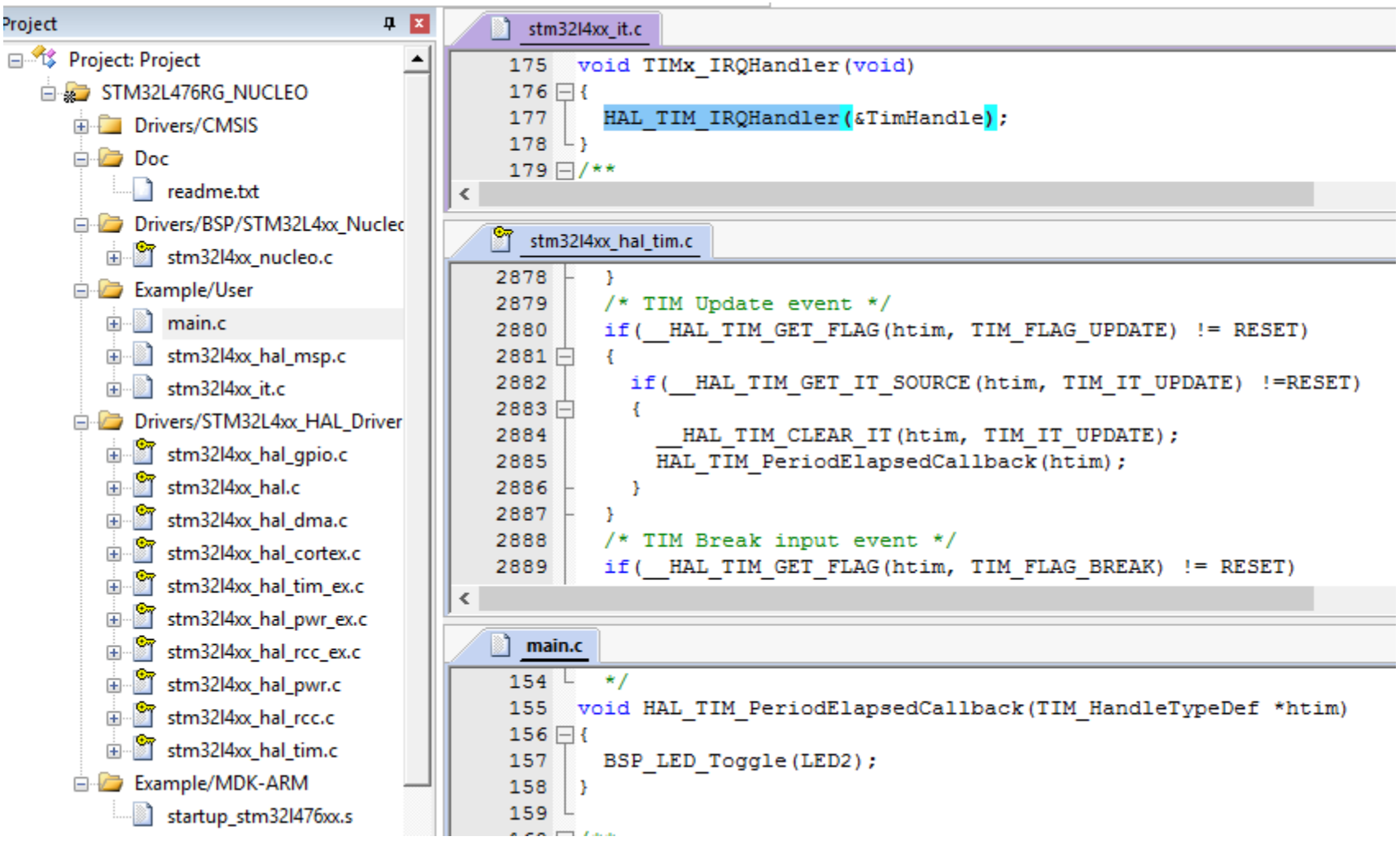
- MSP – MCU Support Package
- Dovodimo takt tajmeru (ovakav ili onakav)
- Konfigurišemo prekide
- Dozvoljavamo prekide

```
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  }
```

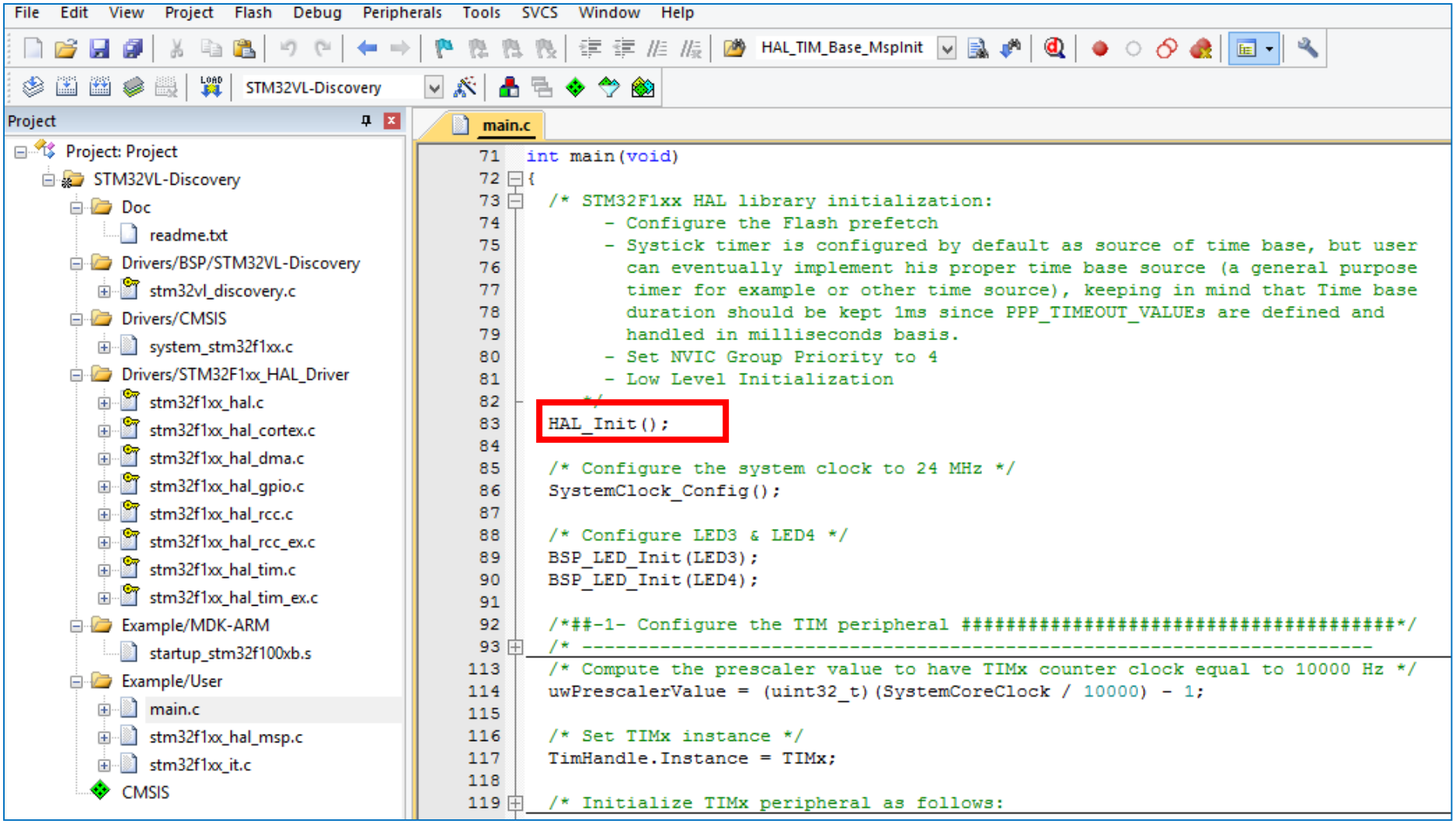
Startujmo tajmer



Imamo i prekid....



Generalna Inicijalizacija hardvera



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_

  /* Use systick as time base source and configure
  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:

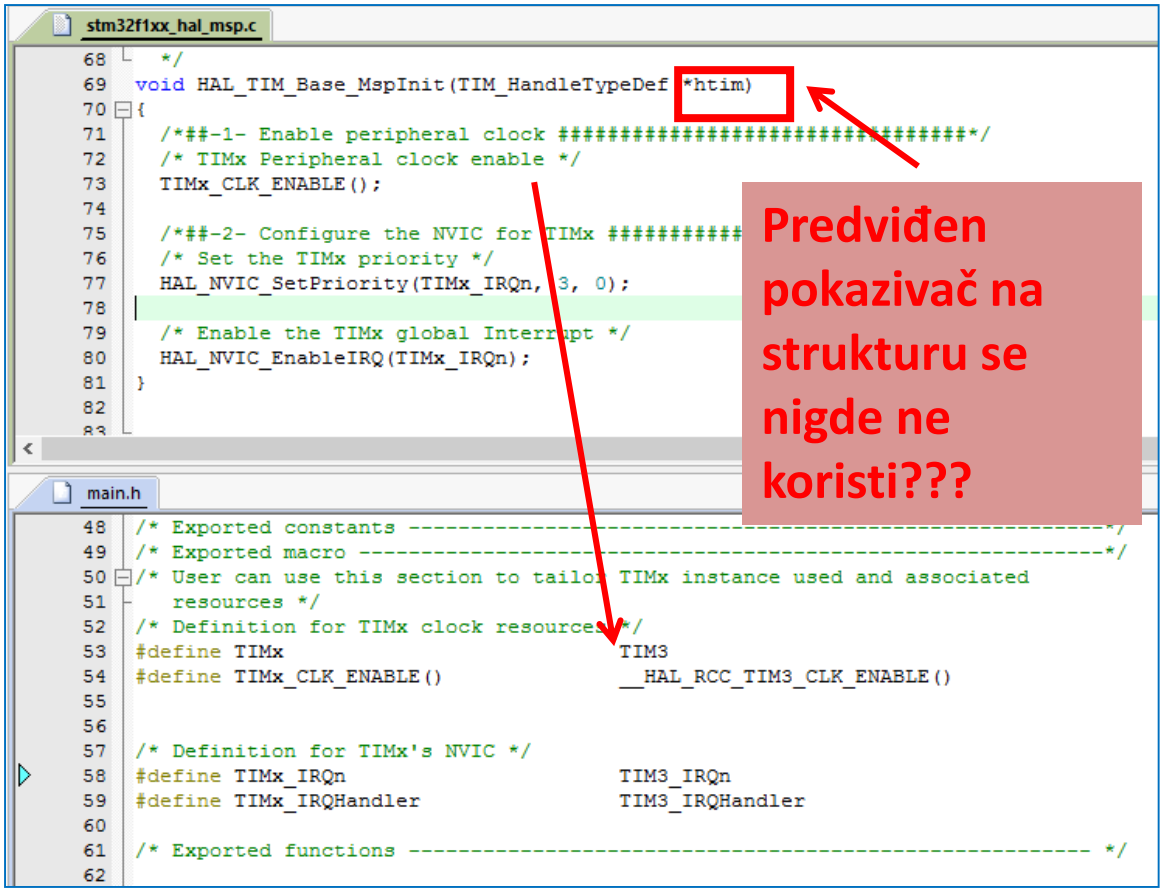
1. Periferiju je najpre potrebno inicijalizovati
2. Inicijalizacija periferije se obavlja u drajveru ali se kao sporedni efekat poziva funkcija HAL_PPP_MspInit() koja inicijalizuje low-level hardverske resurse.
3. Ponekad je potrebno inicijalizovati i neke posebne delove periferije posebnim funkcijama 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

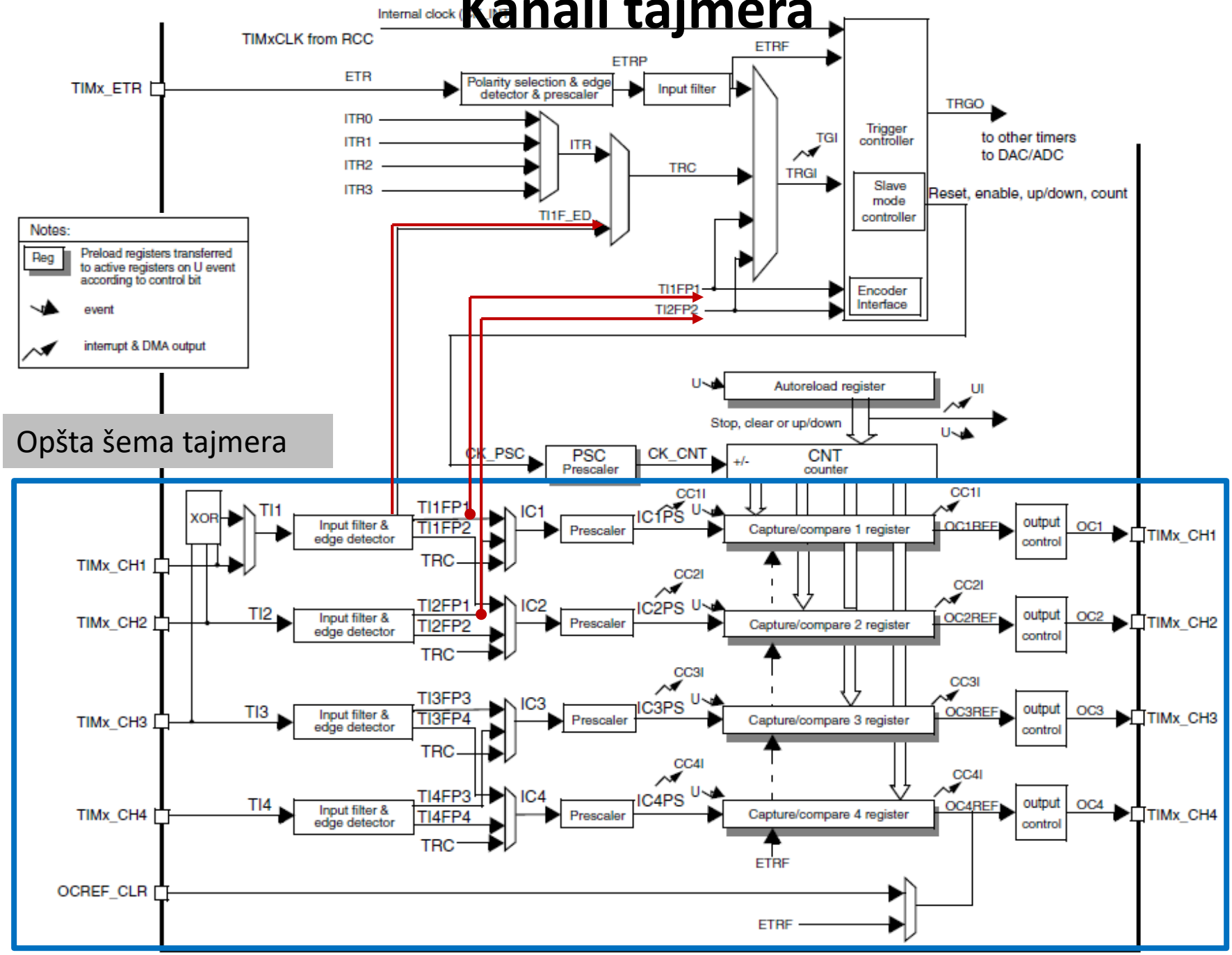
- Najpre pitanje: Koji tajmer je aktivan?
- Zadatak 1: promeniti podešavanja tako da sve ovo radi tajmer 4.
- Zadatak 2: Proširiti projekat tako da rade oba tajmera istovremeno, ali sa neki različitim podešavanjima.
- Problem: MspInit() funkciju će pozvati drajver prilikom inicijalizacije svakog tajmera. 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/STe2ecommunities/mcu/Lists/STM32Java/Attachments/1249/tim.c



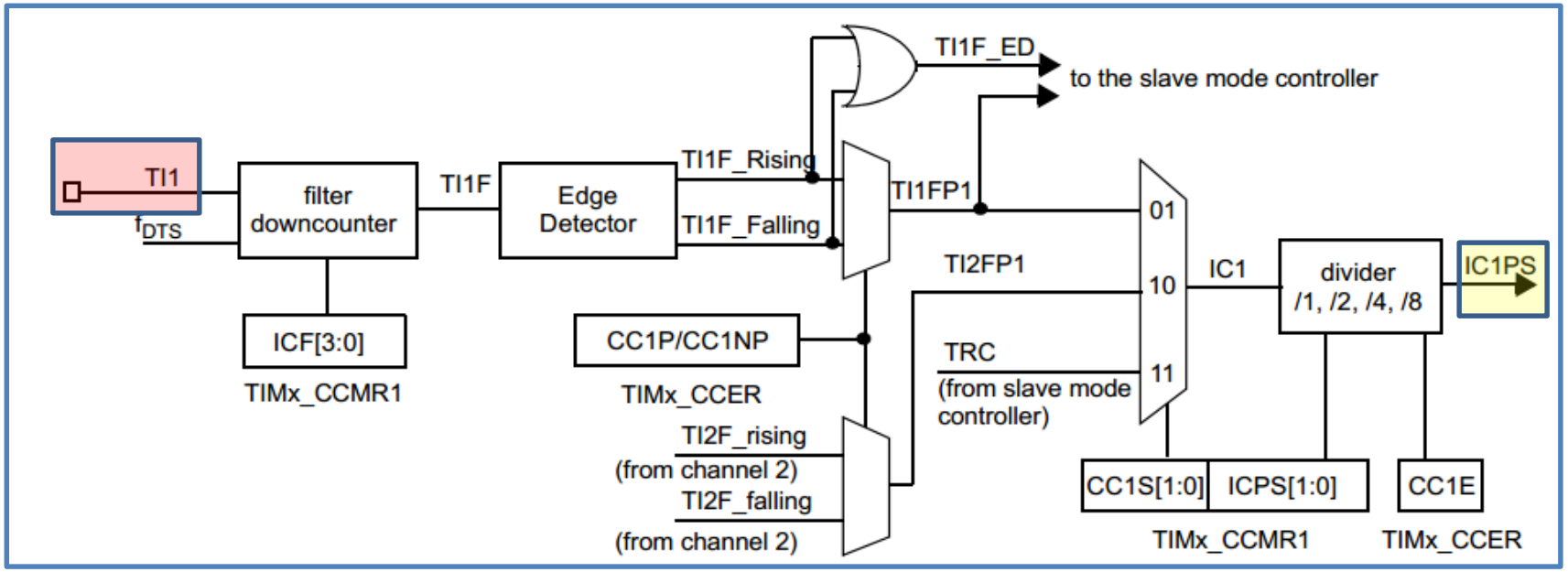
Kanali tajmera



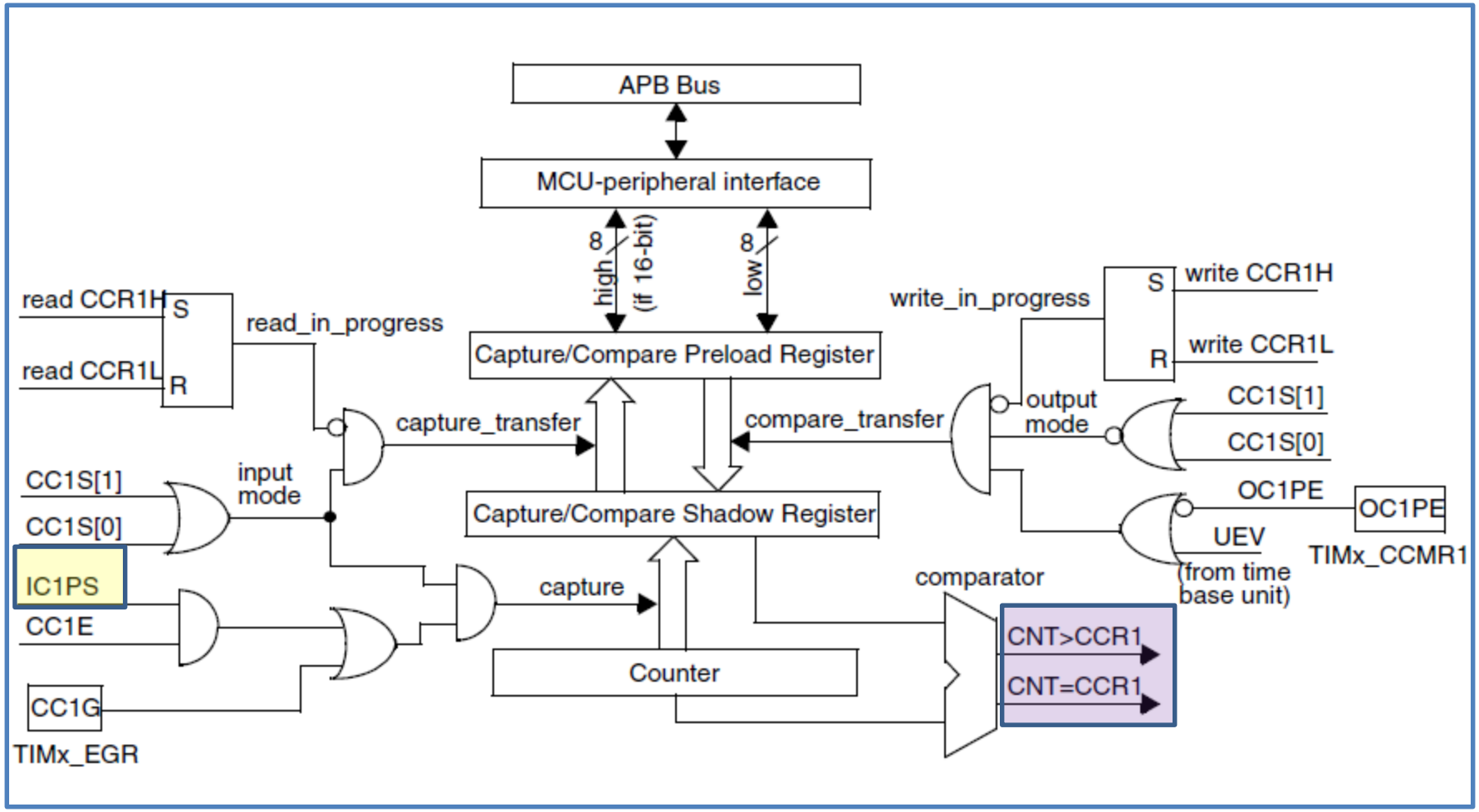
Opšta šema tajmera

Capture/compare jedinica input capture deo

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

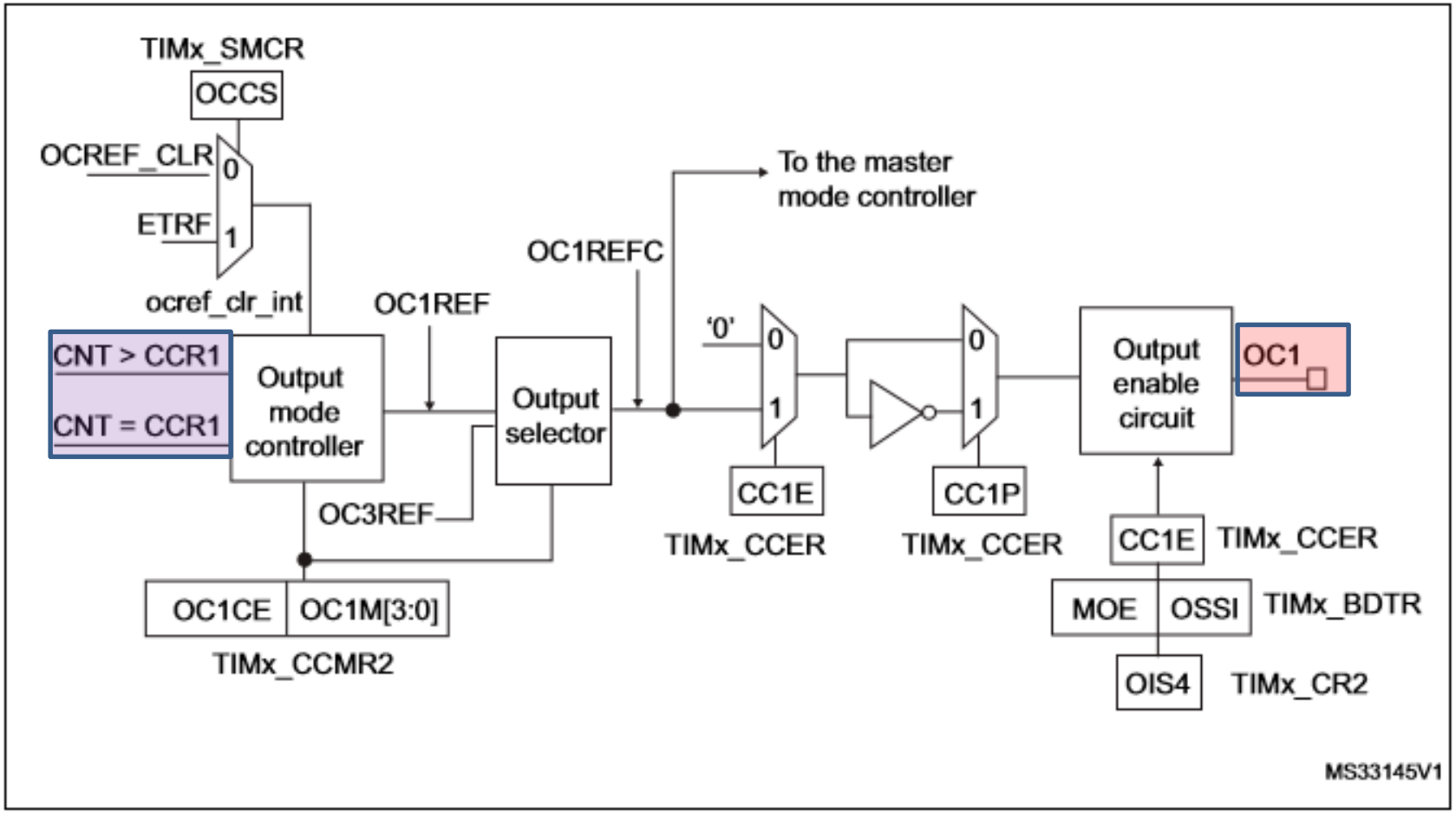


Capture/compare jedinica centralni deo

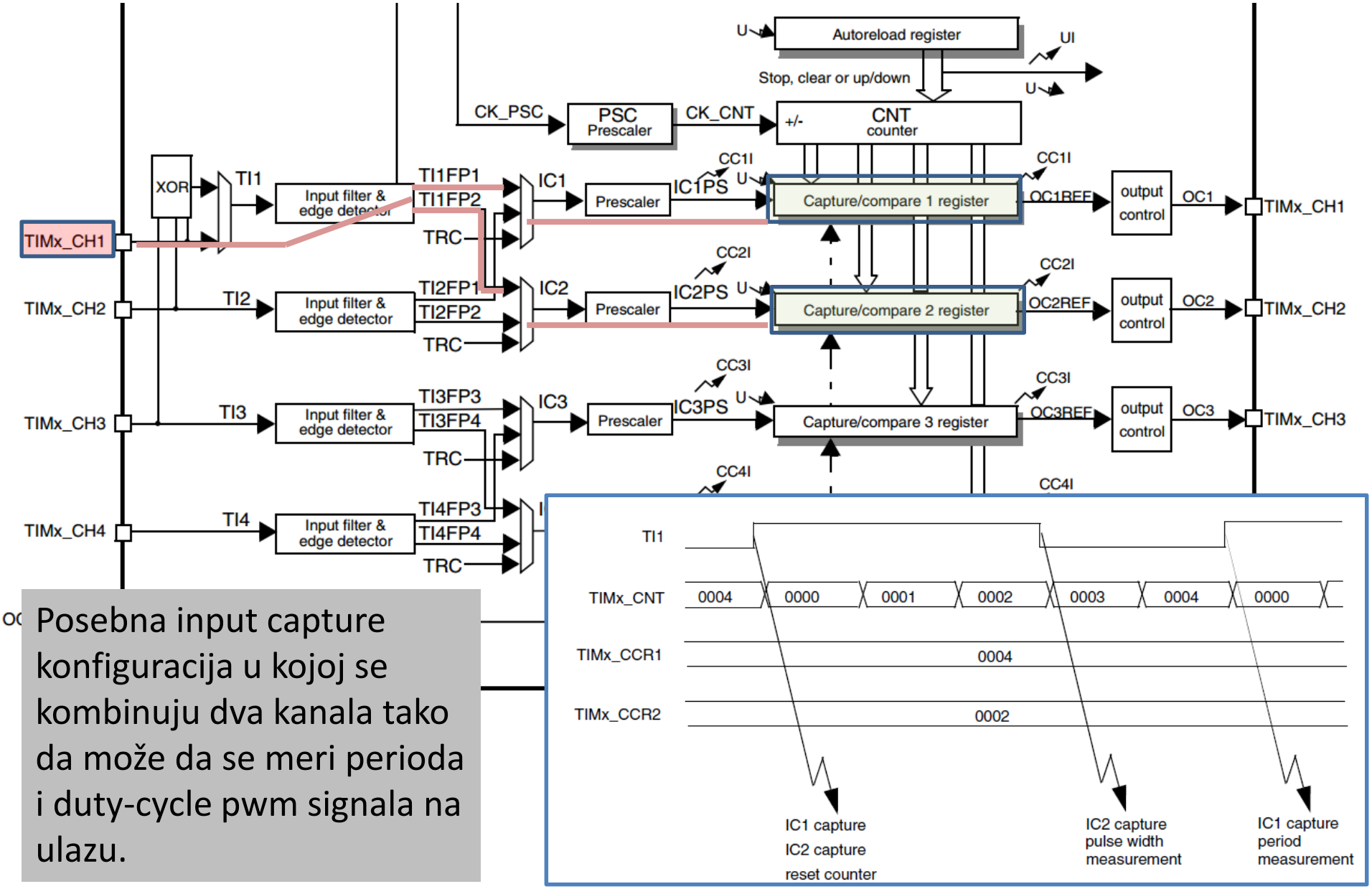


Capture/compare jedinica output compare deo

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



PWM input capture

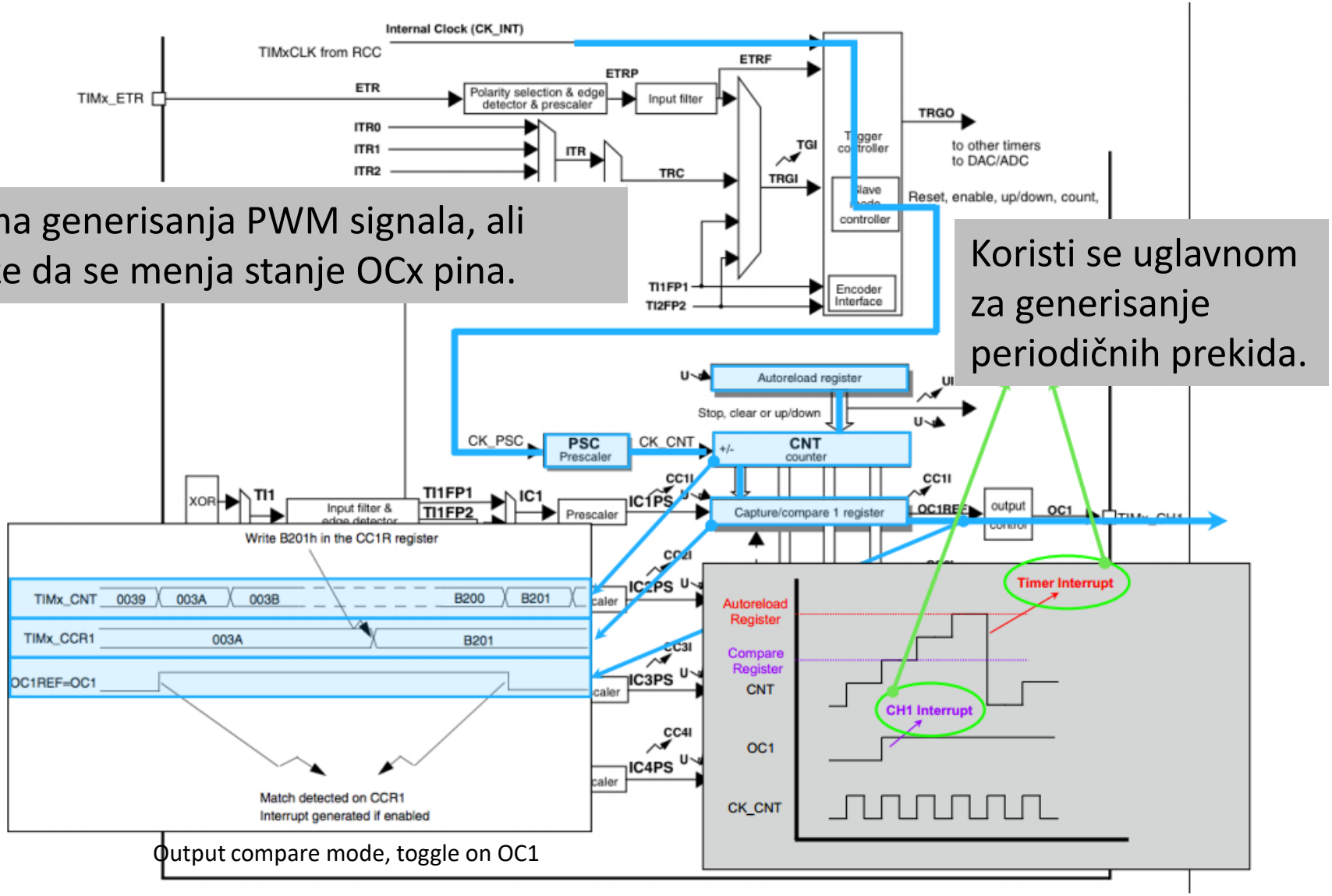


Posebna input capture konfiguracija u kojoj se kombinuju dva kanala tako da može da se meri perioda i duty-cycle pwm signala na ulazu.

Output compare mod

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

Koristi se uglavnom za generisanje periodičnih prekida.



Output compare mode, toggle on OC1

OCx MODE – šta generišemo na izlazu na osnovu OCxREF signala

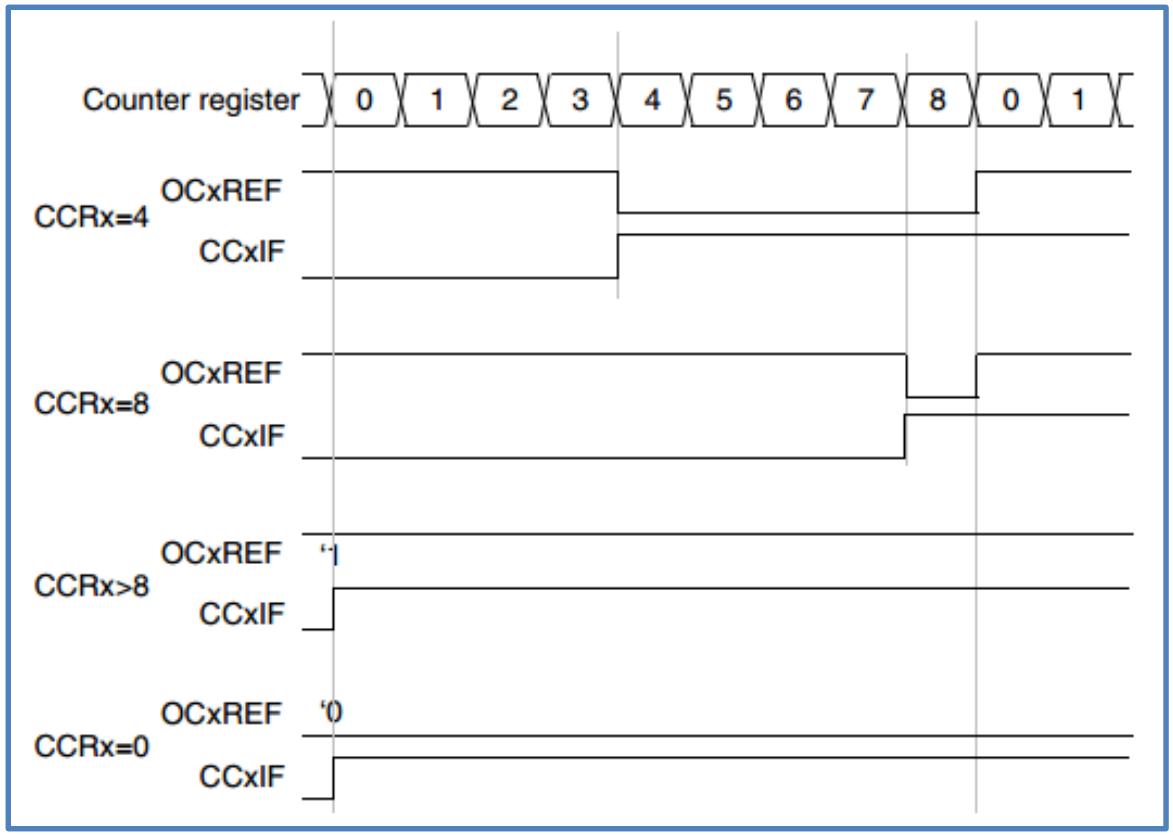
Bits 6:4 **OC1M**: Output Compare 1 mode

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: Retrigerrable 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

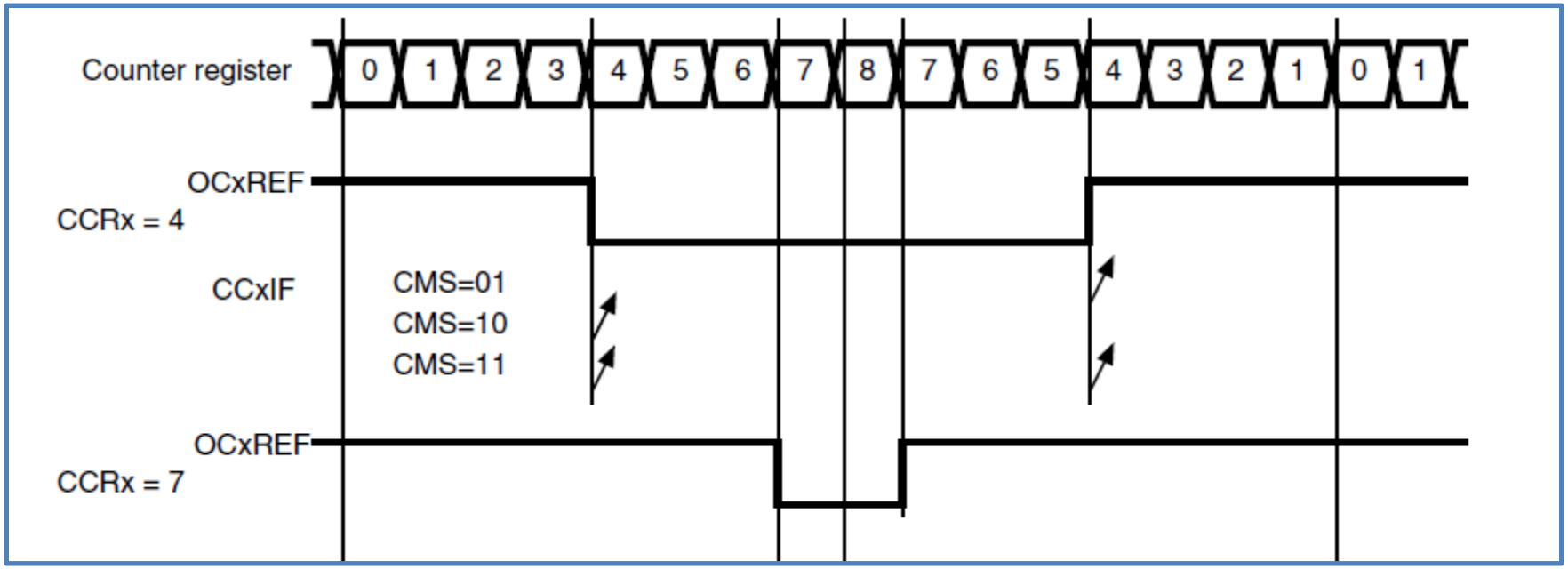
Edge-aligned PWM

- 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 ouput 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.

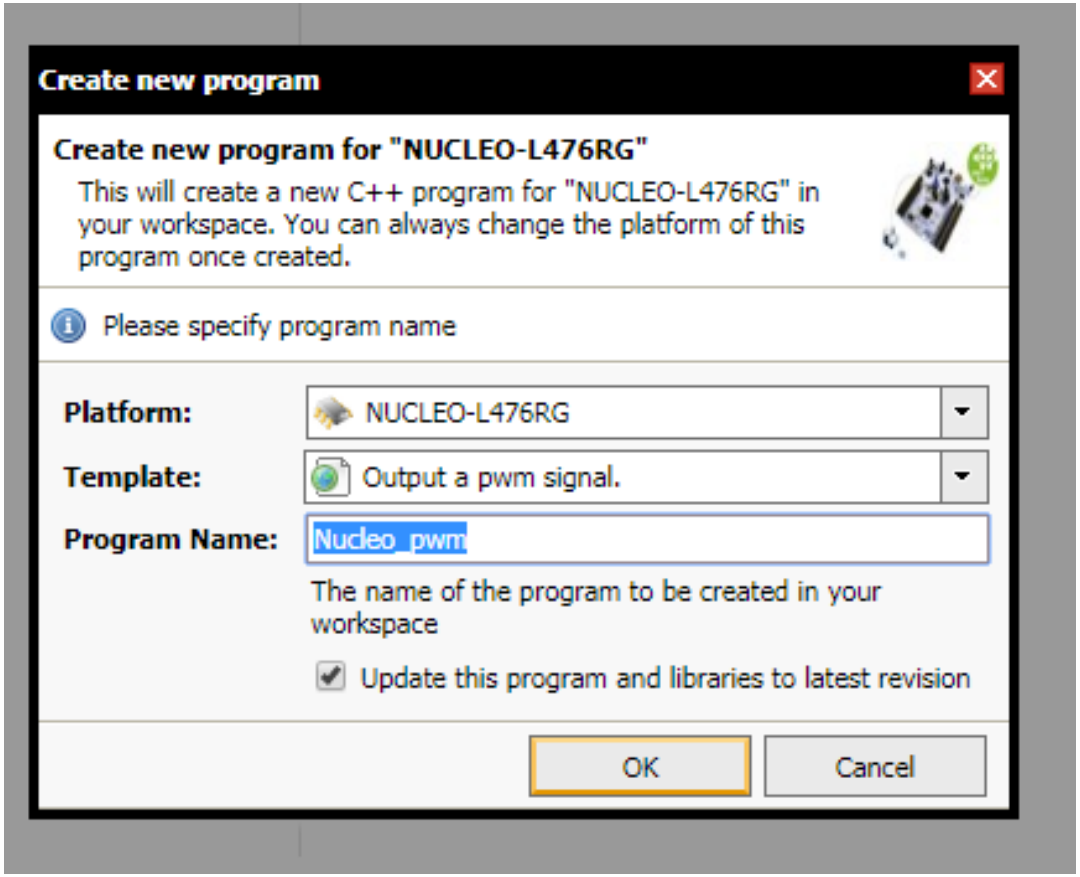
void pulsewidth_us (int us)
Set the PWM pulsewidth, specified in micro-seconds (int), keeping the period the same.

PwmOut & operator= (float value)
A operator shorthand for write()

operator float ()
An operator shorthand for read()

MBED

Template: Output a pwm signal

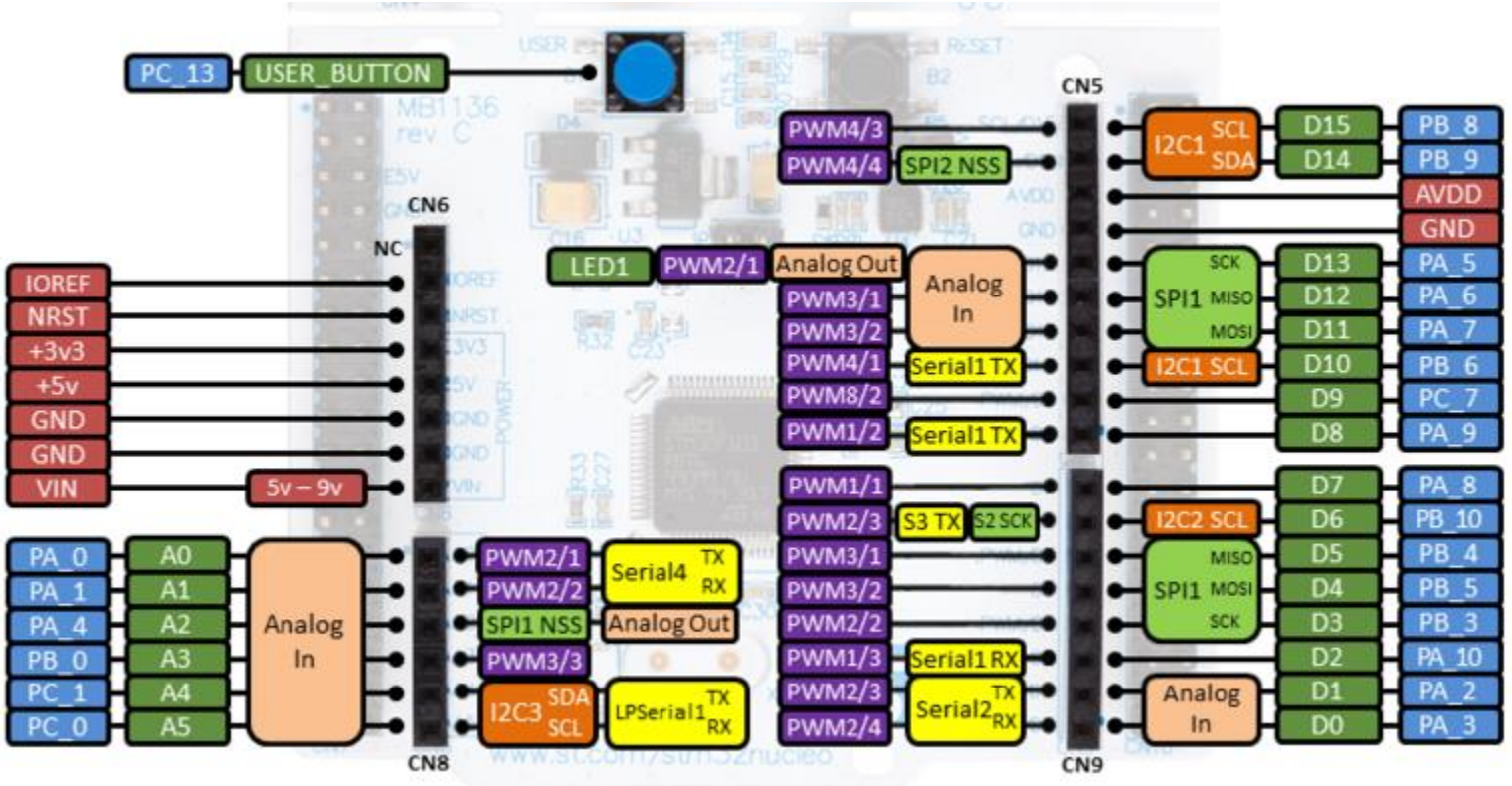


MBED

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);
                }
        }
}
```

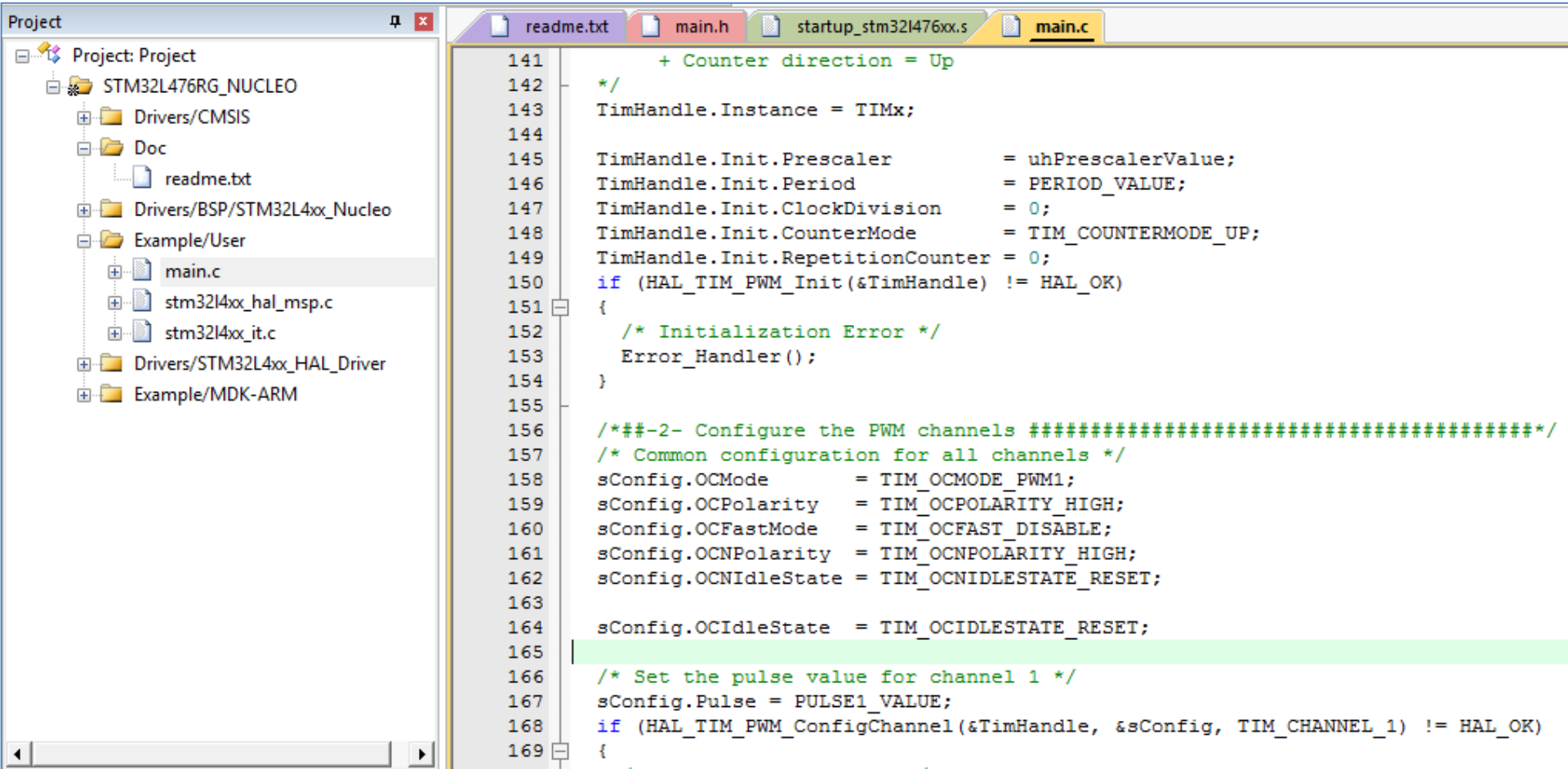
Zadatak

- 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.



Struktura **TIM_OC_InitTypeDef**

readme.txt | main.h | startup_stm32l476xx.s | main.c | stm32l4xx_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;
```

readme.txt | main.h | startup_stm32l476xx.s

```
92   */
93 typedef struct
94 {
95   uint32_t OCMode;         /*!< S
96                                T
97
98   uint32_t Pulse;          /*!< S
99                                T
100
101  uint32_t OCPolarity;     /*!< Specifies the output polarity.
102                                This parameter can be a value of @ref TIM_Output_Compare_Polarity */
103
104  uint32_t OCNPolarity;    /*!< Specifies the complementary output polarity.
105                                This parameter can be a value of @ref TIM_Output_Compare_N_Polarity
106                                @note This parameter is valid only for TIM1 and TIM8. */
107
108  uint32_t OCFastMode;    /*!< Specifies the Fast mode state.
109                                This parameter can be a value of @ref TIM_Output_Fast_State
110                                @note This parameter is valid only in PWM1 and PWM2 mode. */
111
112
113  uint32_t OCIdleState;    /*!< Specifies the TIM Output Compare pin state during Idle state.
114                                This parameter can be a value of @ref TIM_Output_Compare_Idle_State
115                                @note This parameter is valid only for TIM1 and TIM8. */
116
117  uint32_t OCNIdleState;  /*!< Specifies the TIM Output Compare pin state during Idle state.
118                                This parameter can be a value of @ref TIM_Output_Compare_N_Idle_State
119                                @note This parameter is valid only for TIM1 and TIM8. */
120 } TIM_OC_InitTypeDef;
```

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

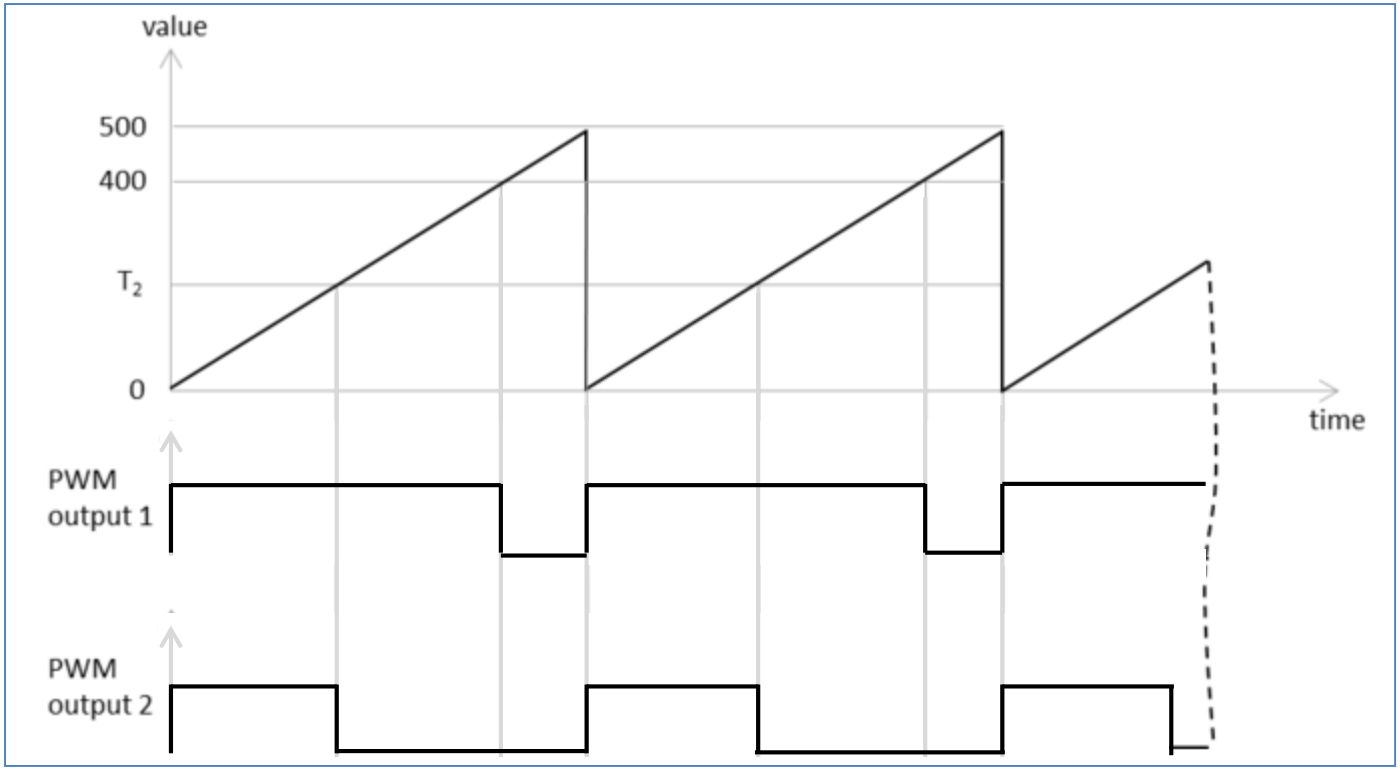
These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

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

readme.txt | main.h | startup_stm32l476xx.s | main.c | stm32l4xx_hal_tim.h

```
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



Podešavanje svakog OC kanala

readme.txt | main.h | startup_stm32l476xx.s | main.c

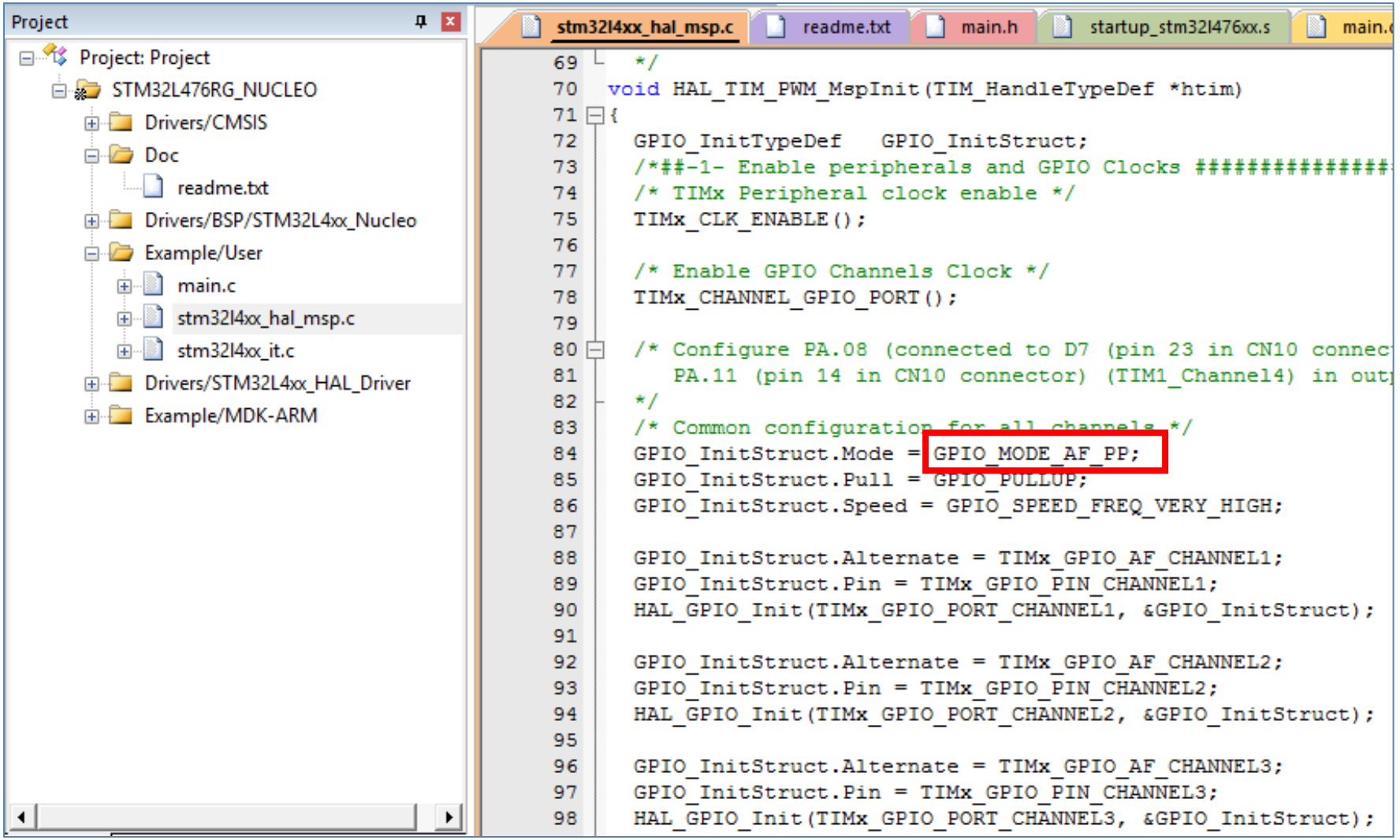
```
166   /* Set the pulse value for channel 1 */
167   sConfig.Pulse = PULSE1_VALUE;
168   if (HAL_TIM_PWM_ConfigChannel(&TimHandle, &sConfig, TIM_CHANNEL_1) != HAL_OK)
169   {
170     /* Configuration Error */
171     Error_Handler();
172   }
173
174   /* Set the pulse value for channel 2 */
175   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   }
```

Startovanje OC kanala

readme.txt | main.h | startup_stm32l476xx.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?



life.augmented

STM32L476xx

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

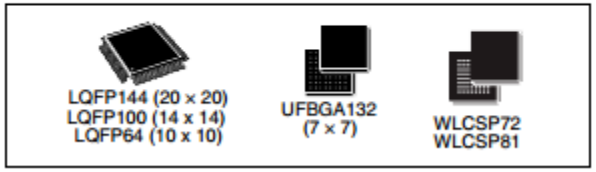


Table 15. STM32L476xxSTM32L476xx pin definitions (continued)

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_l	-	MCO, TIM1_CH1, USART1_CK, OTG_FS_SOF, LCD_COM0, LPTIM2_OUT, EVENTOUT	-
42	E3	E3	68	D10	101	PA9	I/O	FT_lu	-	TIM1_CH2, USART1_TX, LCD_COM1, TIM15_BKIN, EVENTOUT	OTG_FS_ VBUS
43	D2	D2	69	C12	102	PA10	I/O	FT_lu	-	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?

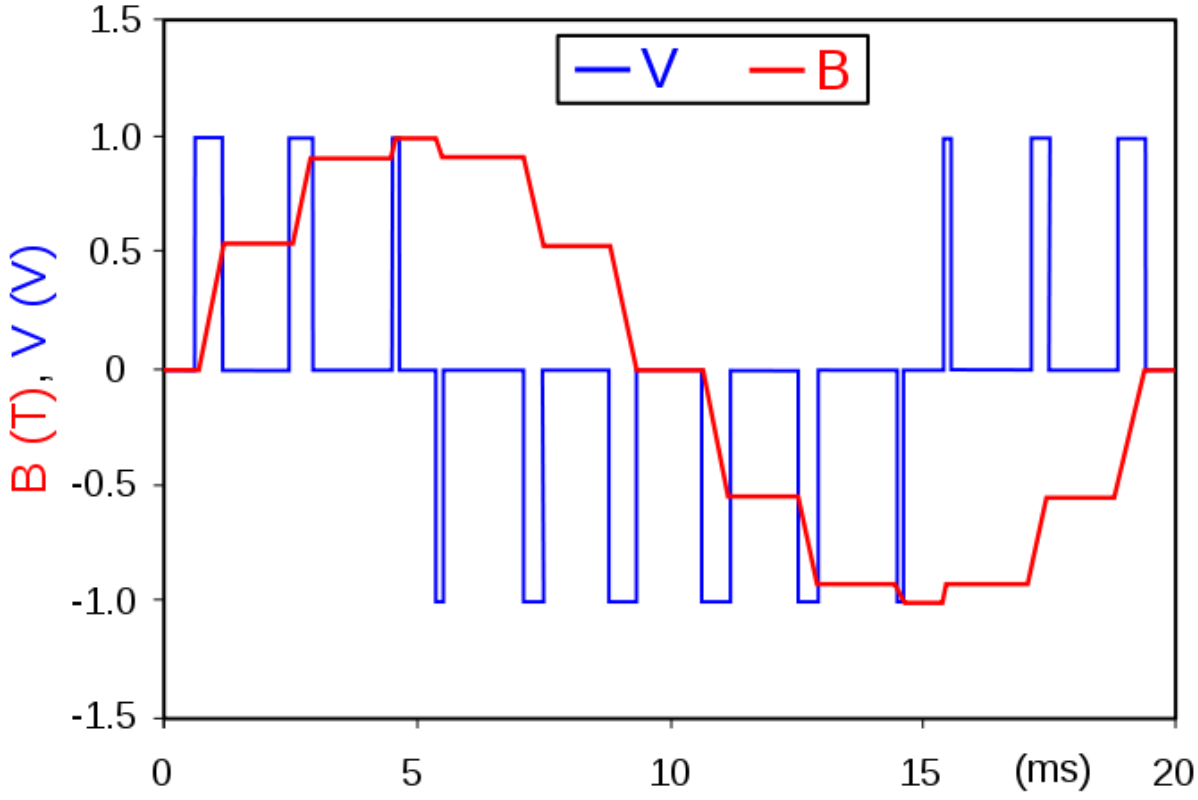
Zadatak za vežbu

- 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 MspInit() 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)

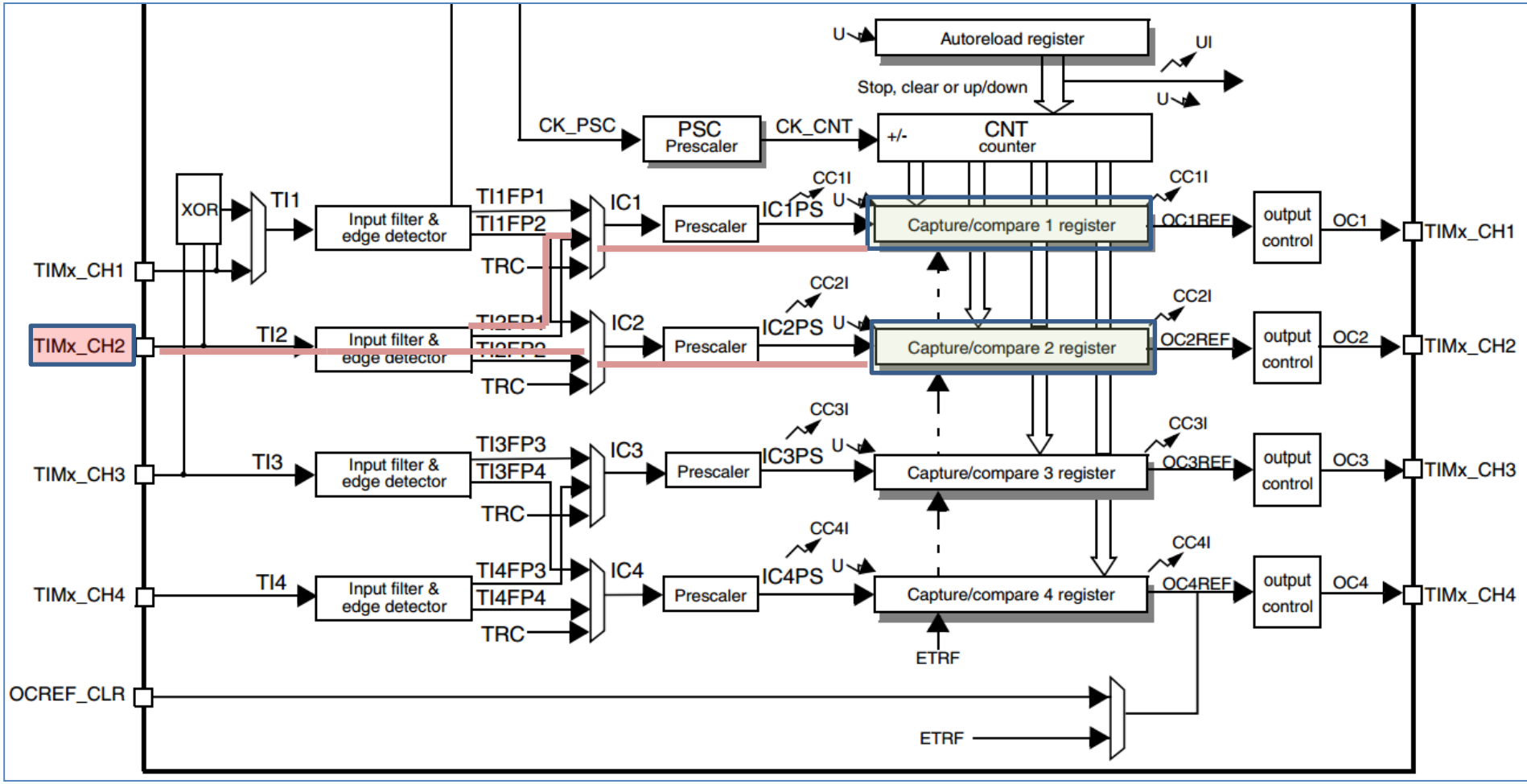
STM CUBE

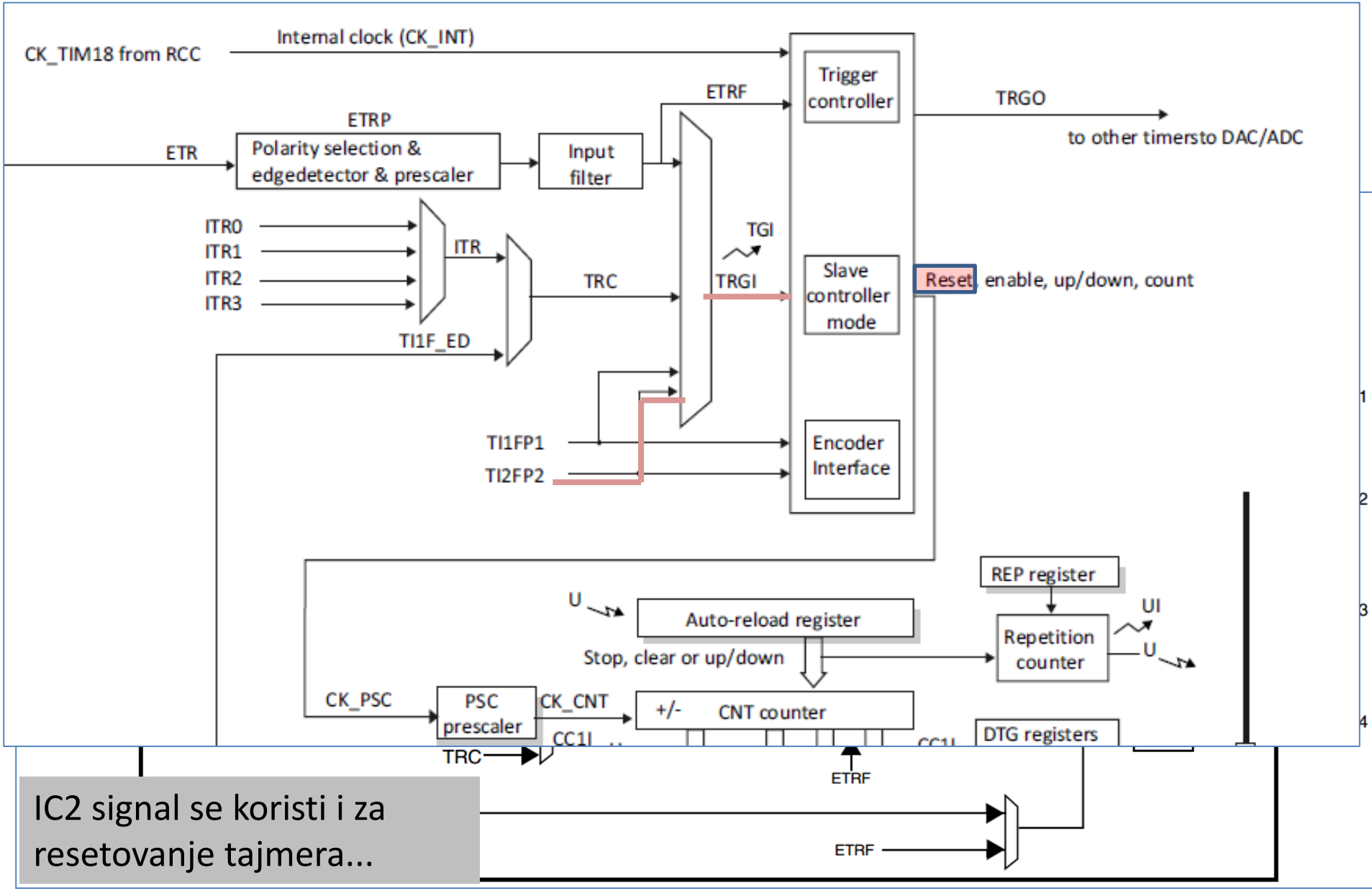
Projekat TIM_PWmInput

- 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

```
51  /* Private define ------------------------------------------------------------*/
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;
```

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

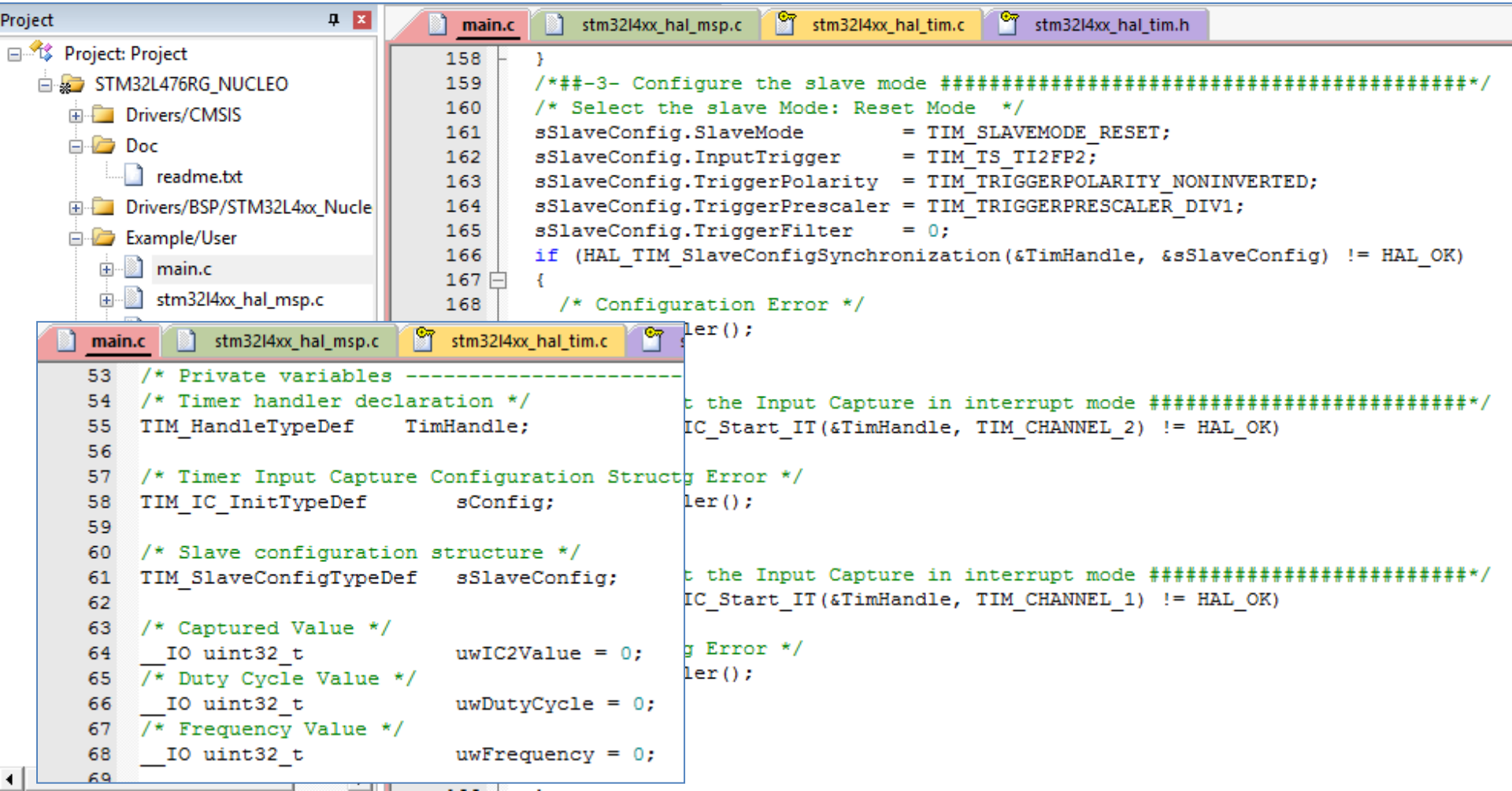
```
/** @defgroup TIM_Input_Capture_Selection TIM Input Capture Selection
  * @{
  */
#define TIM_ICSELECTION_DIRECTTI           (TIM_CCMR1_CC1S_0)    /*!< TIM Input 1, 2, 3 or 4 is selected to be
                                                                    connected to IC1, IC2, IC3 or IC4, respectively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

Podešavanje 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



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

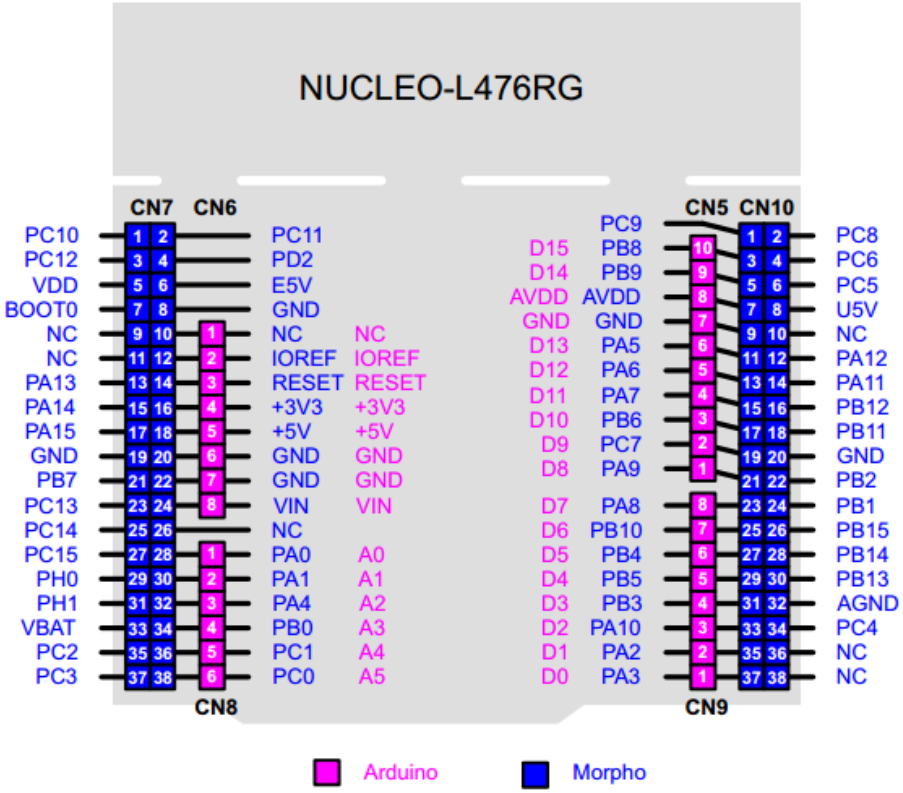
```
46
47 /** @defgroup TIM_Slave_Mode TIM Slave mode
48   * @{
49   */
50 #define TIM_SLAVEMODE_DISABLE                ((uint32_t)0x0000)
51 #define TIM_SLAVEMODE_RESET                  ((uint32_t)(TIM_SMCR_SMS_2))
52 #define TIM_SLAVEMODE_GATED                  ((uint32_t)(TIM_SMCR_SMS_2 | TIM_SMCR_SMS_0))
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

stm32l4xx_it.c | readme.txt | main.c* | stm32l4xx_hal_msp.c | stm32l4xx_hal_tim.c | stm32l4xx_hal_tim

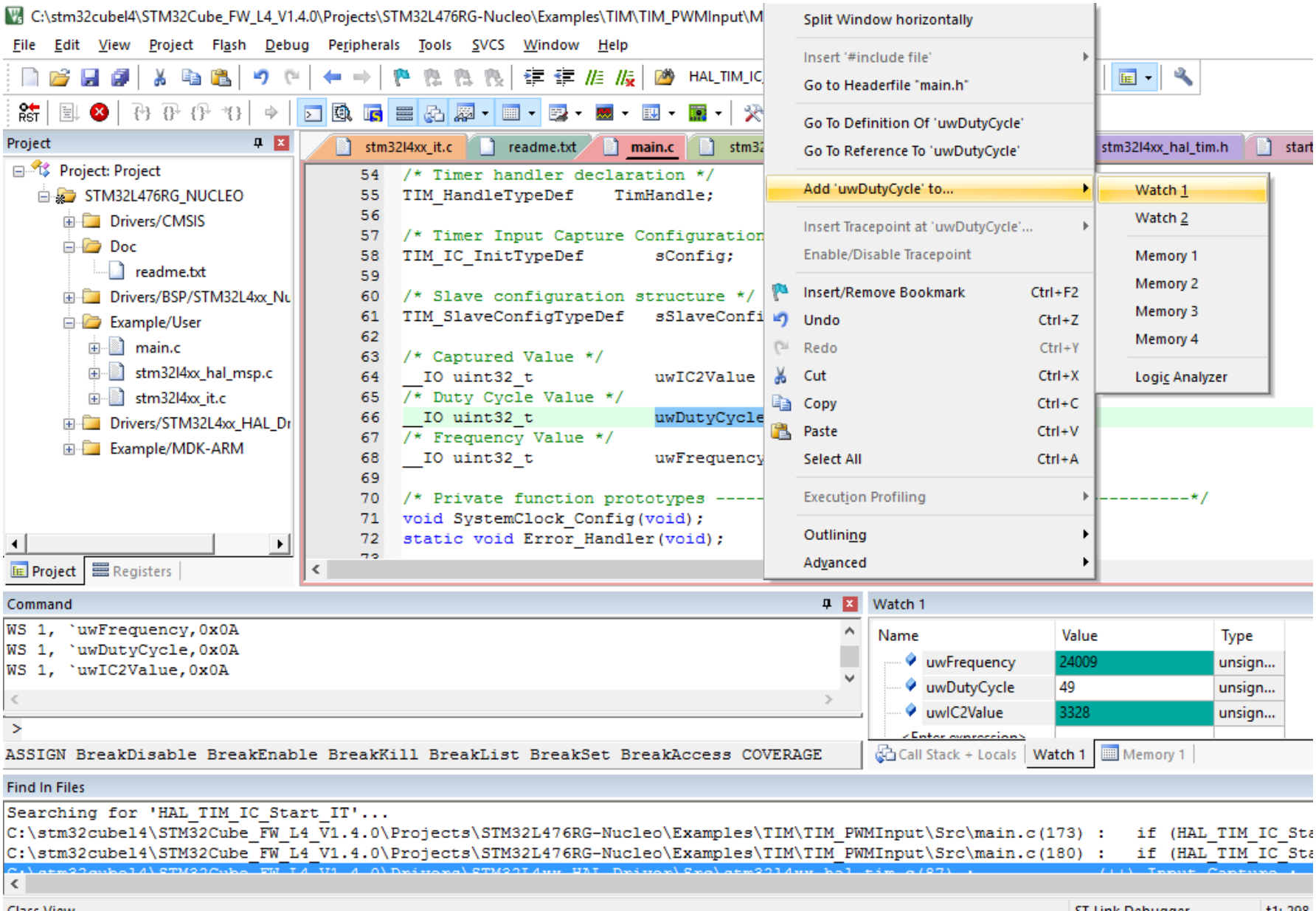
```
163   sSlaveConfig.TriggerPolarity  = TIM_TRIGGERPOLARITY_NONINVERTED;
164   sSlaveConfig.TriggerPrescaler = TIM_TRIGGERPRESCALER_DIV1;
165   sSlaveConfig.TriggerFilter    = 0;
166   if (HAL_TIM_SlaveConfigSynchronization(&TimHandle, &sSlaveConfig) != HAL_OK)
167   {
168     /* Configuration Error */
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

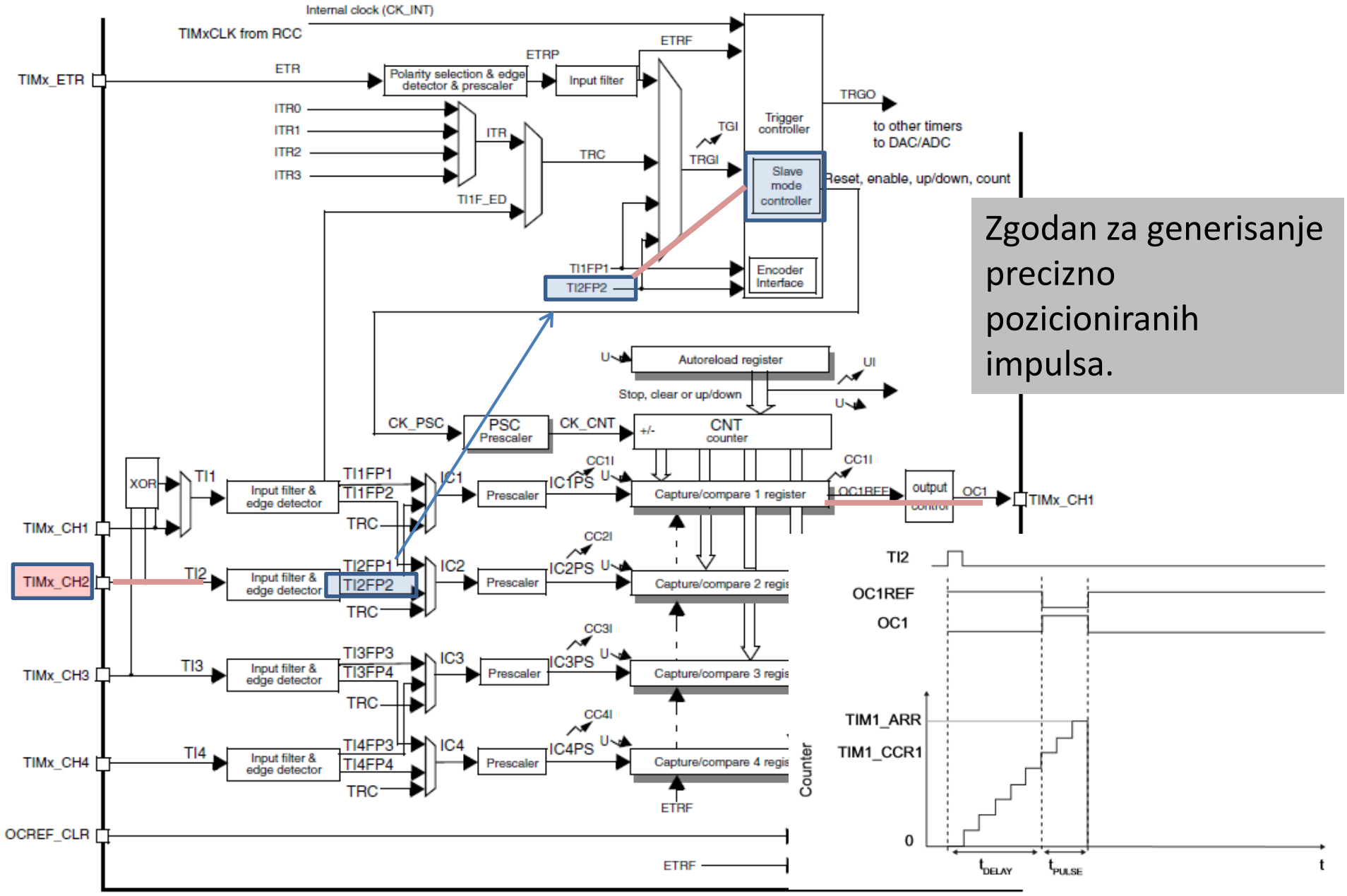


- Jedan kontroler izvršava projekat PWMOutput, a druga grupa izvršava projekat PWMInput
- Povezati dva kontrola pomoću dva provodnika – masa i signal.
- Pitanje: Gde se računaju parametri ulaznog PWM signala?
- Da li će projekat PMWInput korektno da radi ako se CH1 inicijalizuje bez prekidne funcije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

KEIL Osvežava promenljive u realnom vremenu

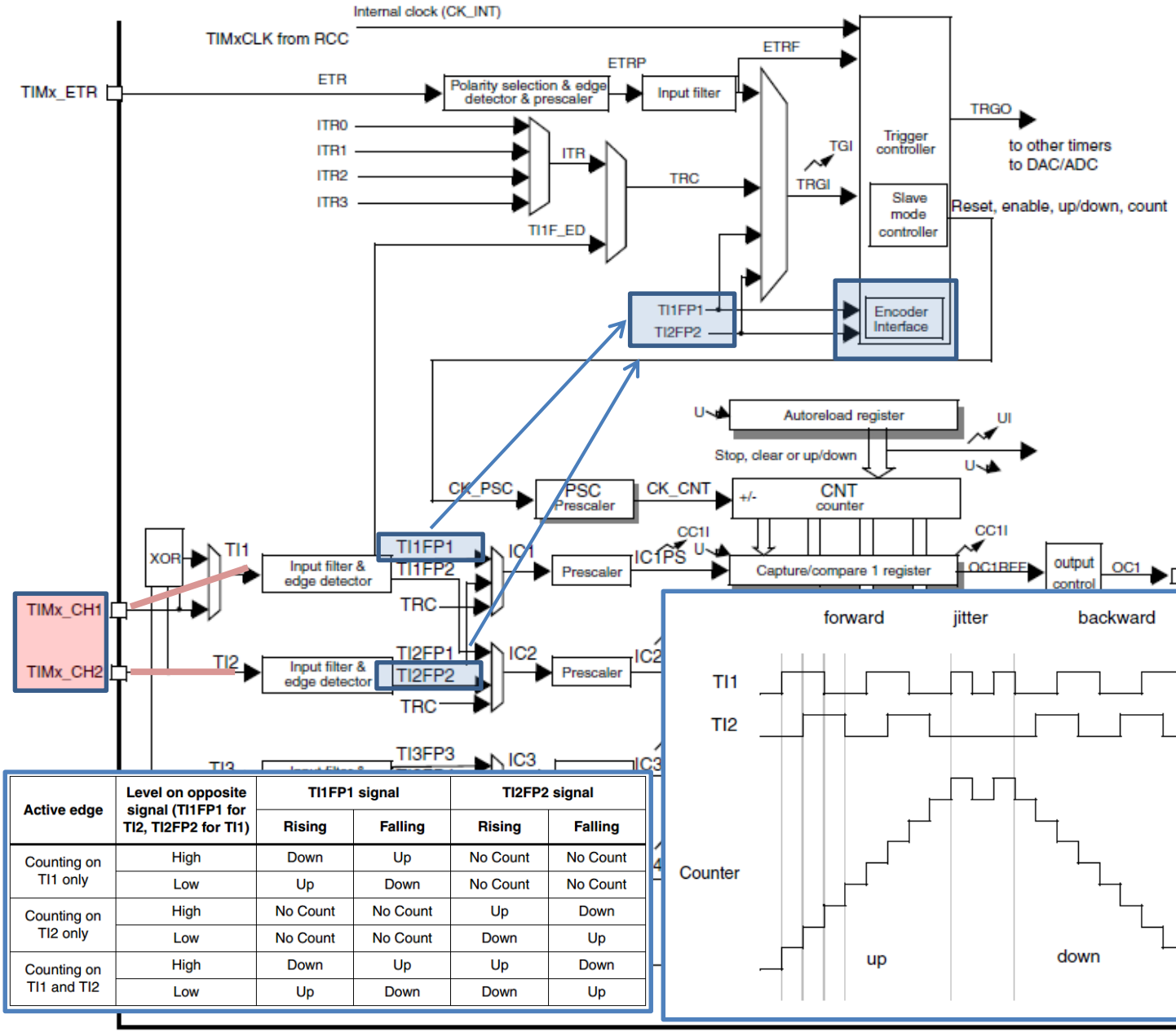


One pulse mod



Zgodan za generisanje precizno pozicioniranih impulsa.

Encoder interface

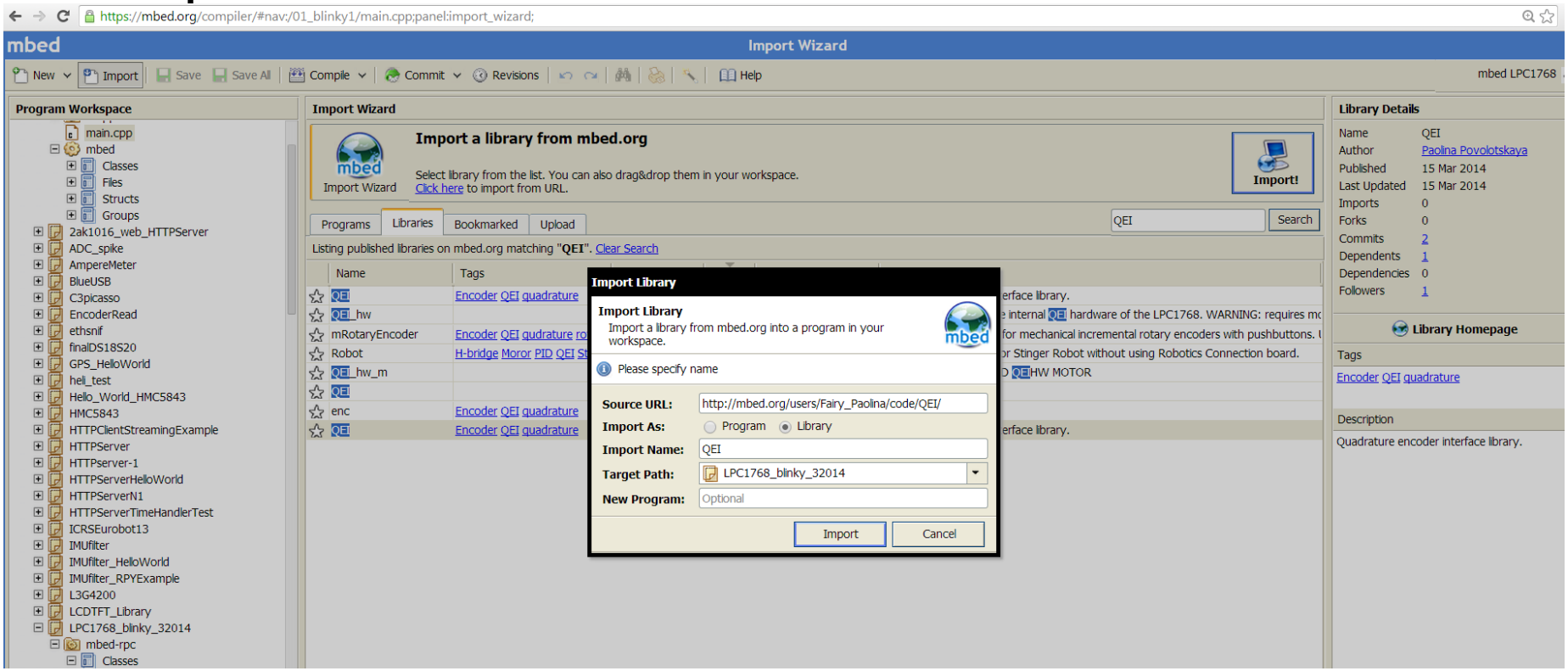


Povezivanje kvadraturnog enkodera na kanalima 1 i 2 u svim modovima brojanja: x1, x2, x4.

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????

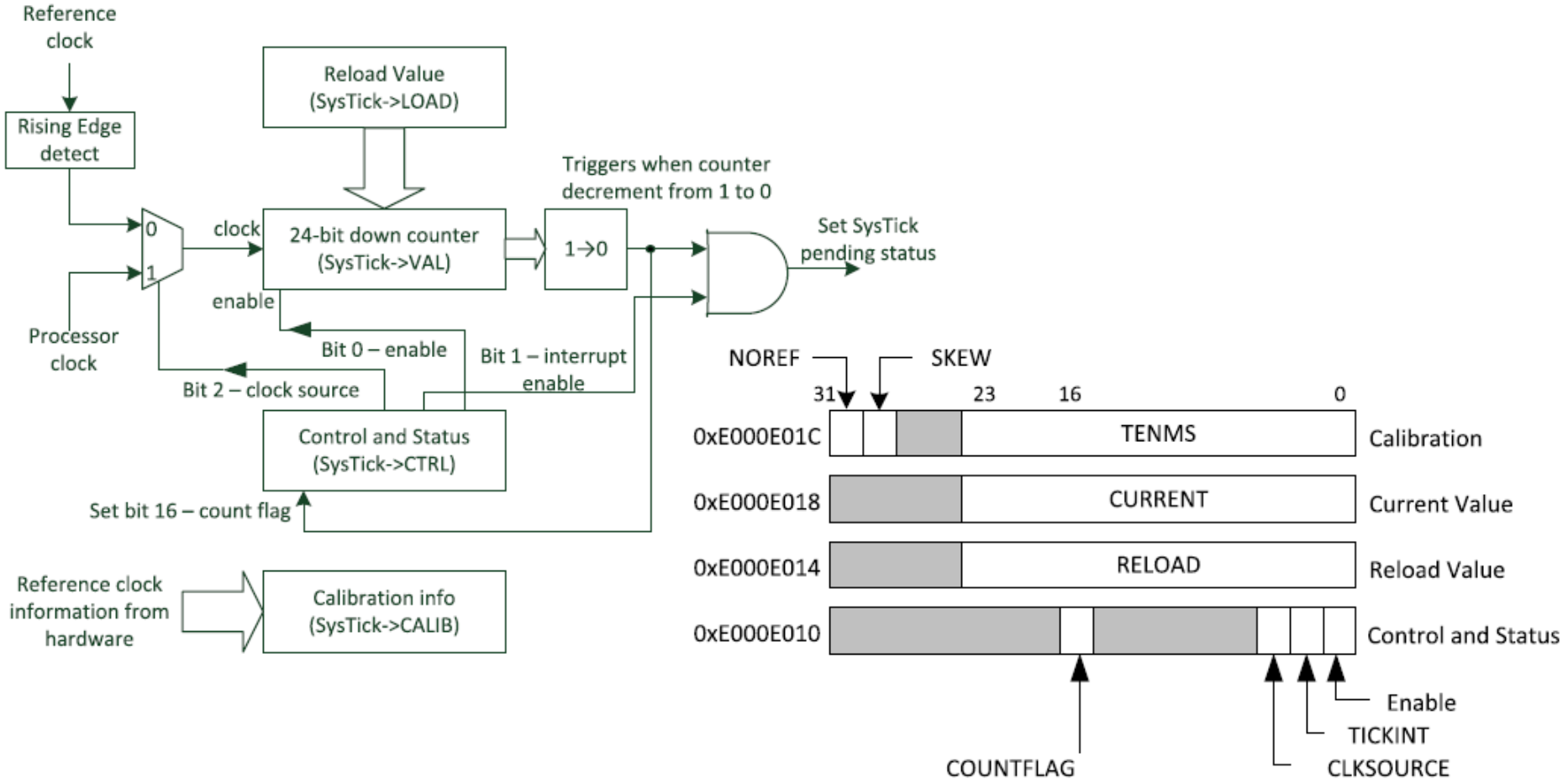
- Osnovna MBED biblioteka ne definiše klasu za encoder interfejs, ali postoje projekti i biblioteke u pokviru MBED baze. Doduše za LPC1768...



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 bili ili CPU takt ili CPU/8 takt. Ovo se konfiguriše u RCC grupi registara.
- S obzirom da je deo procesorskog jezgra definicije funkcija koje konfigurišu 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**, specifiying 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**, specifiying the interval in seconds.

void attach_us (void(*fptr)(void), unsigned int t)
Attach a function to be called by the **Ticker**, specifiying 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**, specifiying 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) ;
}
```

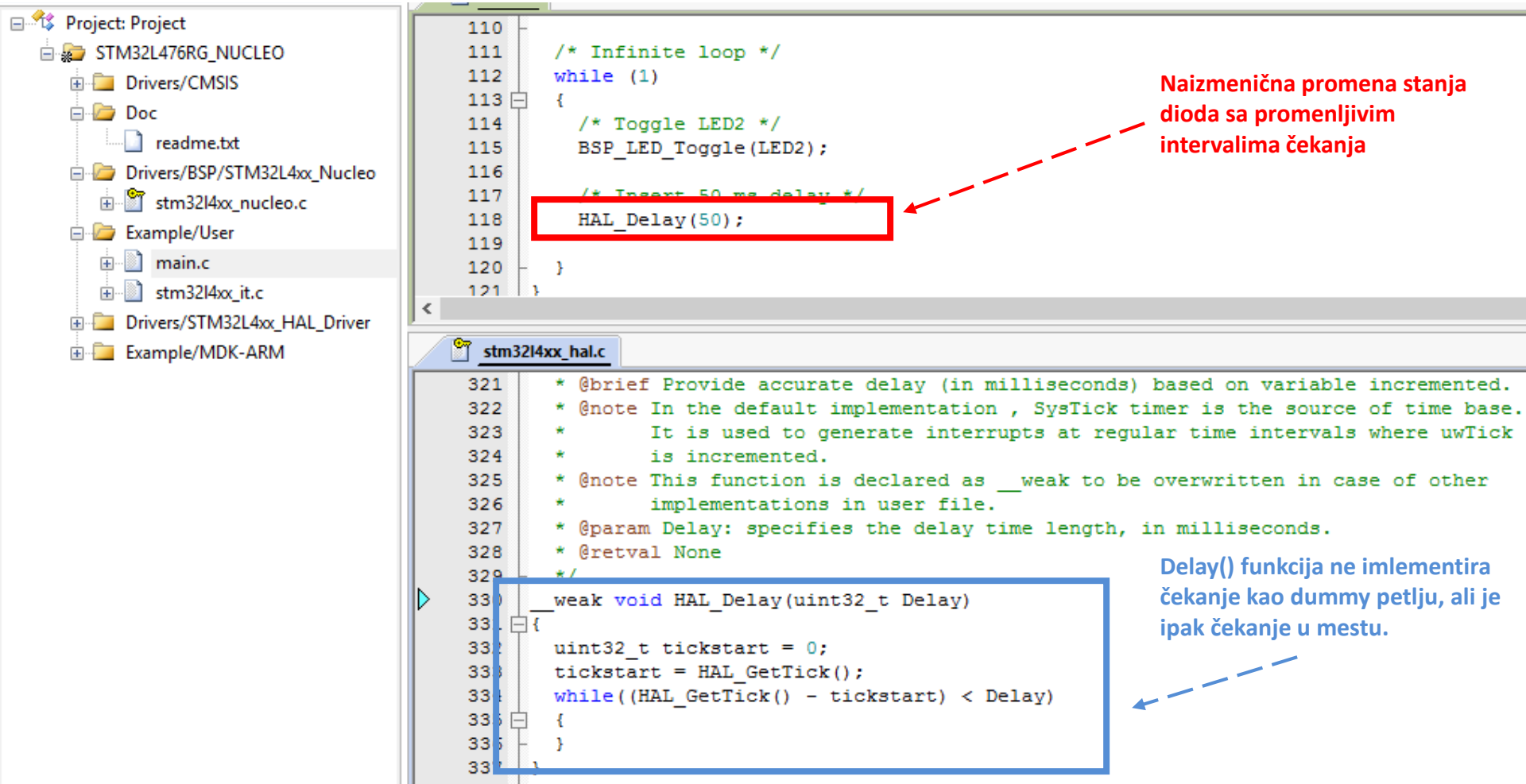
Povezivanje sa funkcijom članicom neke klase

```
#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);
}
```

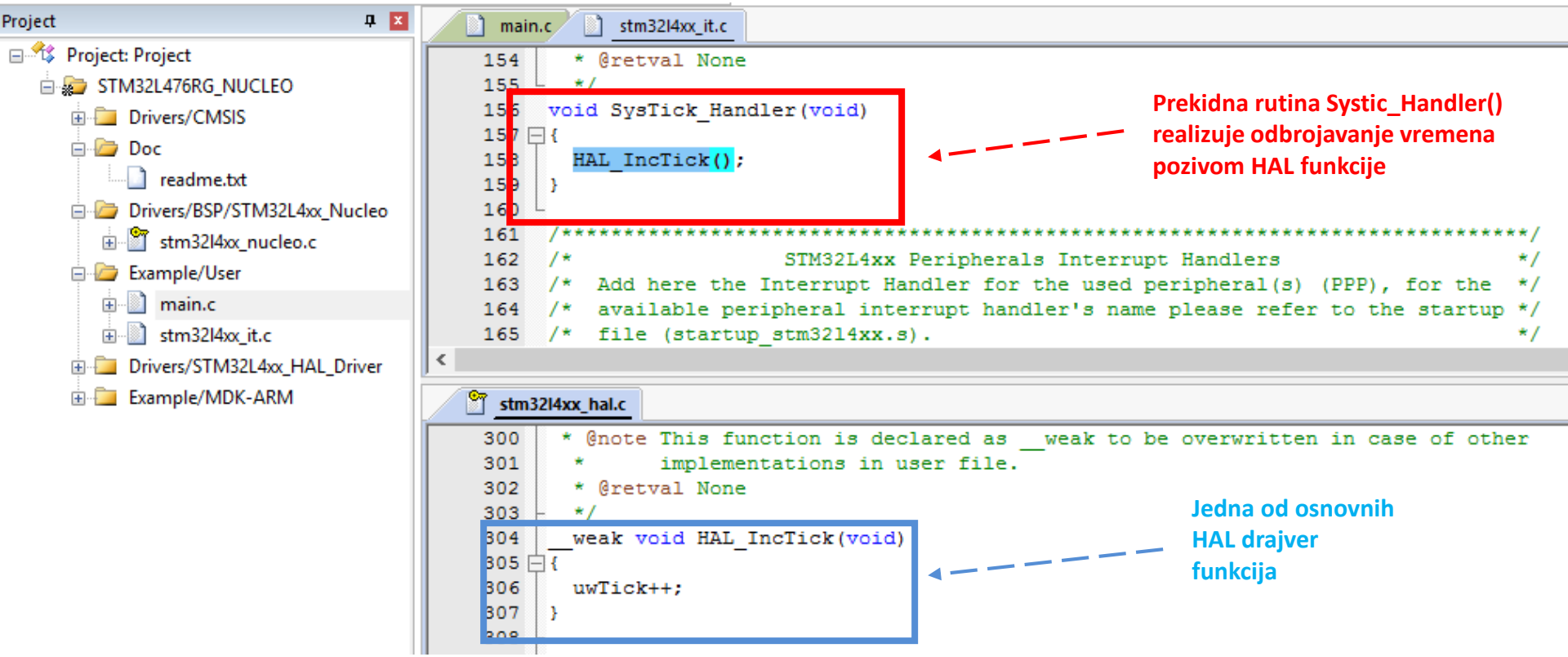
STM CUBE

Projekat CORTEXM_SysTick

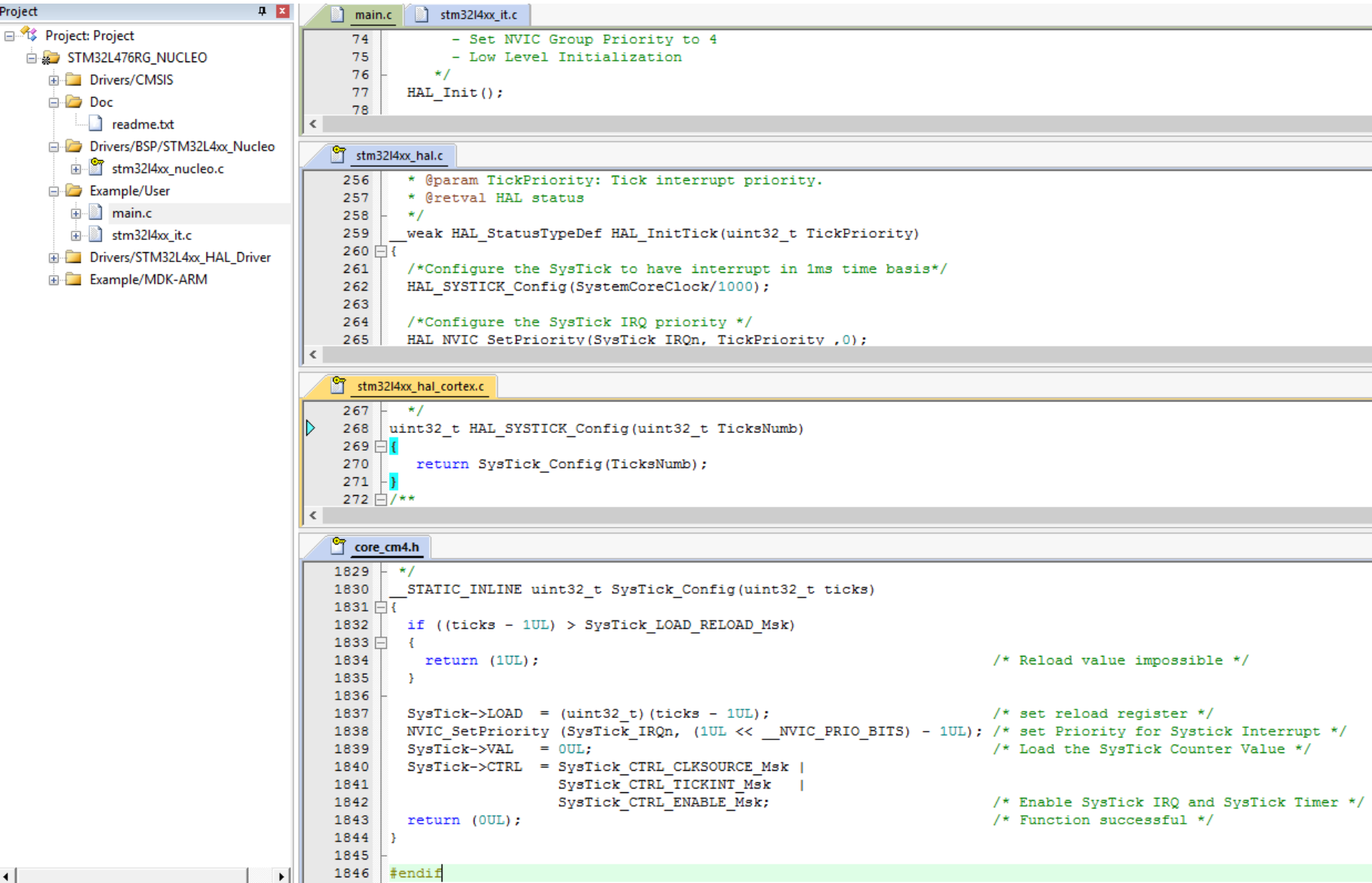
\\stm32cubel4\STM32Cube_FW_L4_V1.4.0\Projects\STM32L476RG-Nucleo\Examples\Cortex\CORTEXM_SysTick\MDK-ARM



Realizacija preko prekida



Inicijalizacija Systick tajmera



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, specifiying 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, specifiying the interval in seconds.

void attach_us (void(*fptr)(void), unsigned int t)
Attach a function to be called by the Ticker, specifiying 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, specifiying 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 mili-seconds.

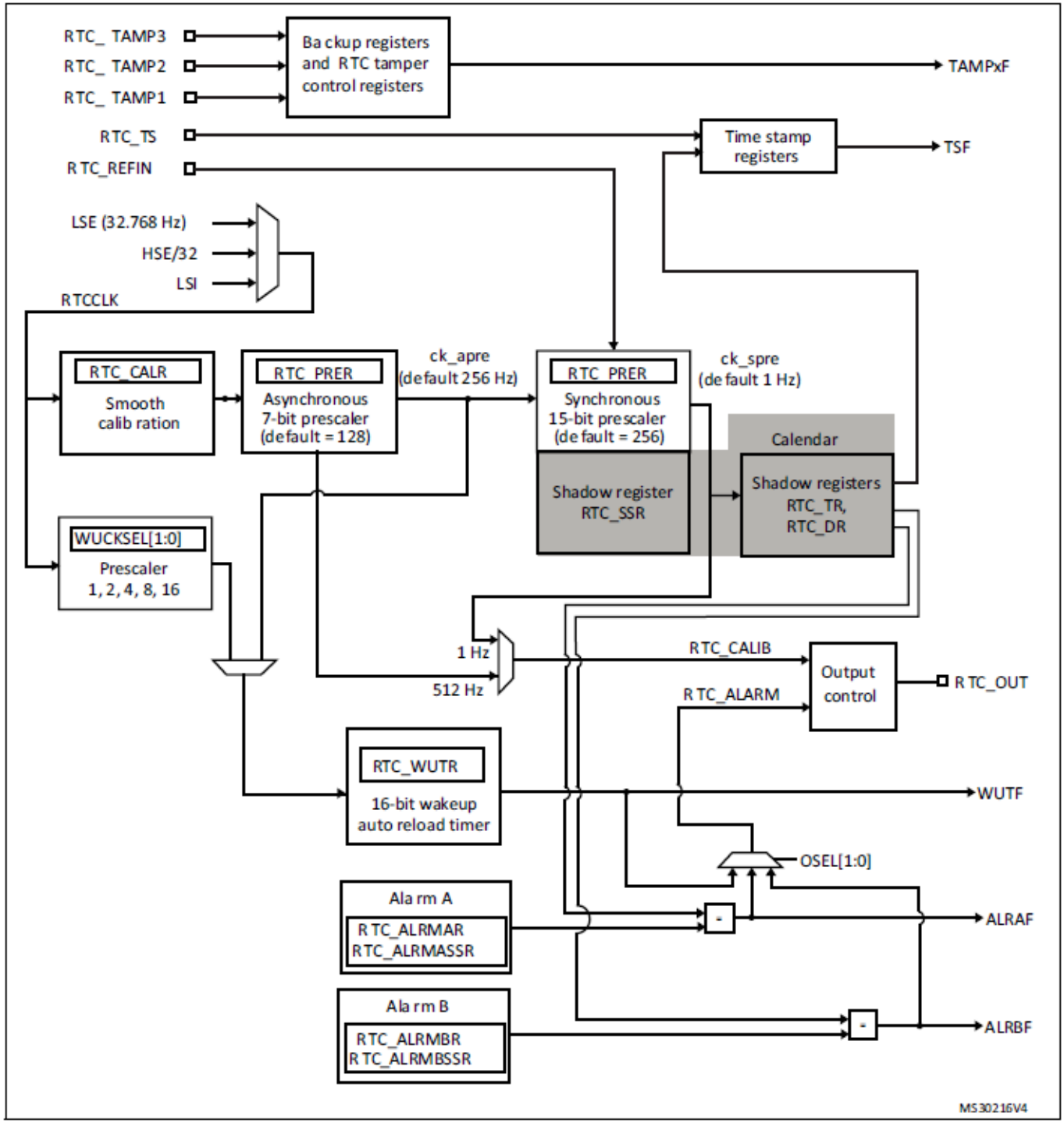
int read_us ()
Get the time passed in micro-seconds.

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

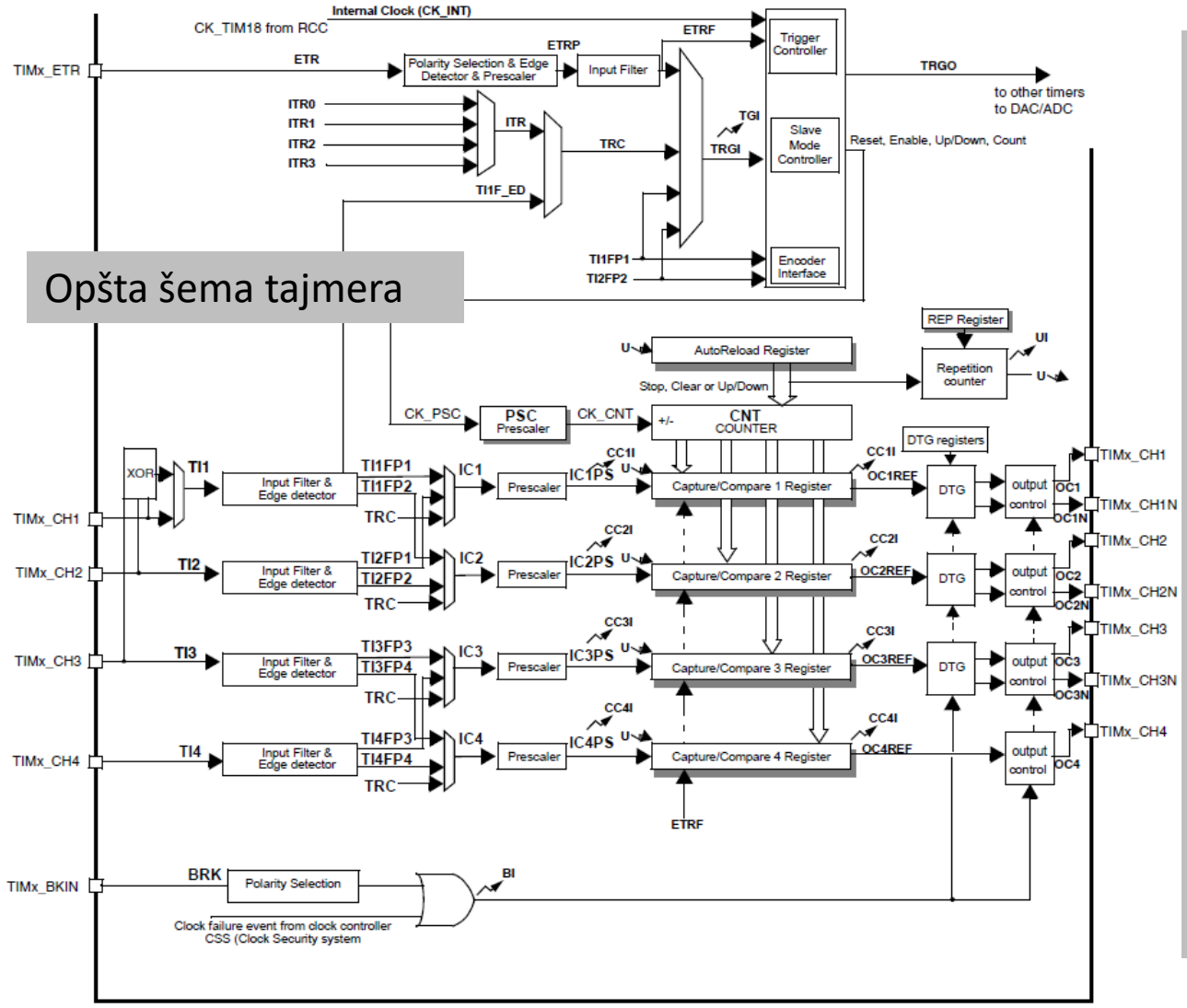
- Za podršku su kreirane MBED funkcije, ali ne postoji klasa RTC. Funkcije su deklarisane u time.h i rtc_time.h.

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);
    }
}
```

TIM1– Advanced control timer

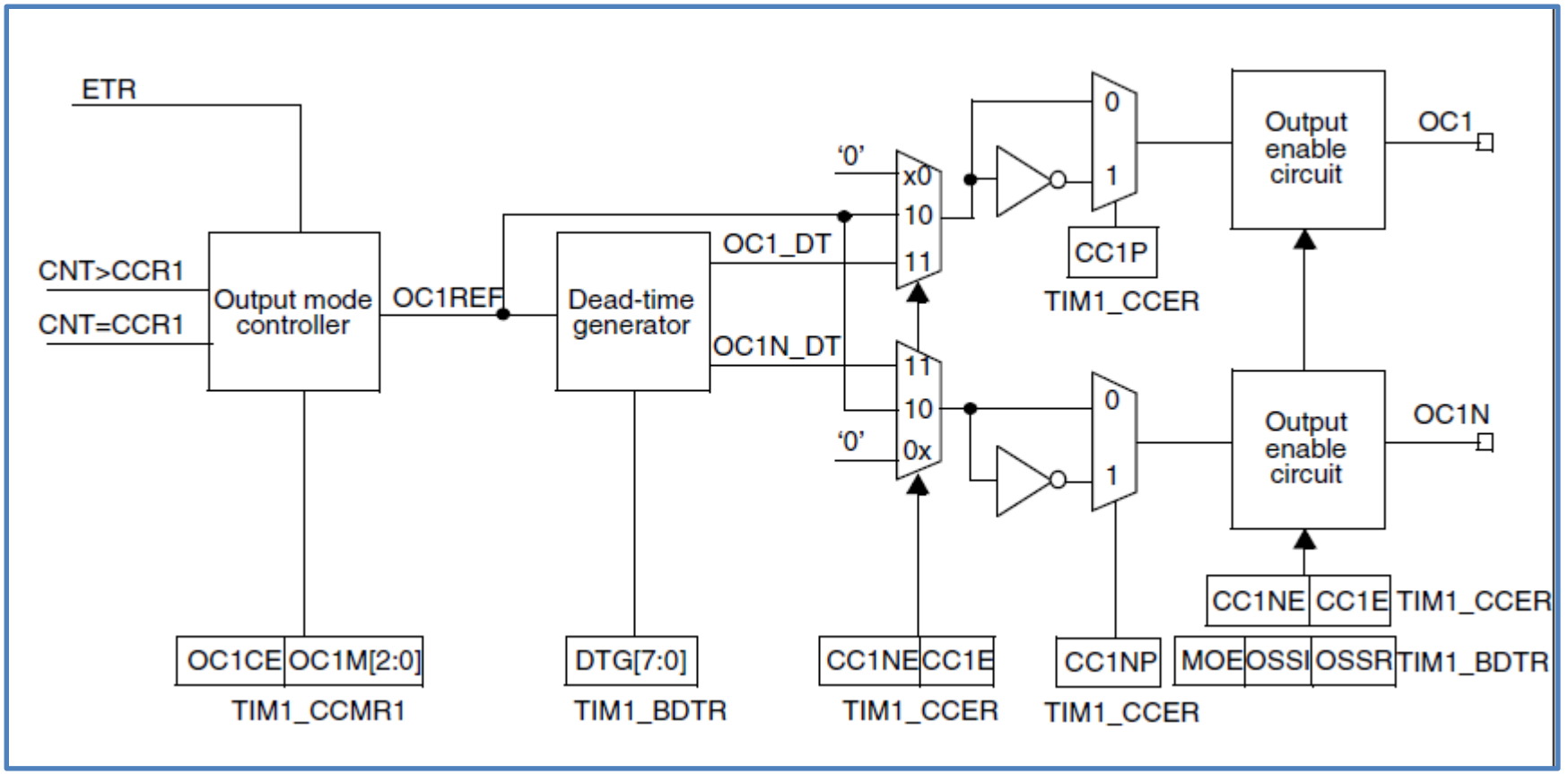


Opšta šema tajmera

Pored osobina tajmera opšte namene poseduje sledeće specifičnosti:

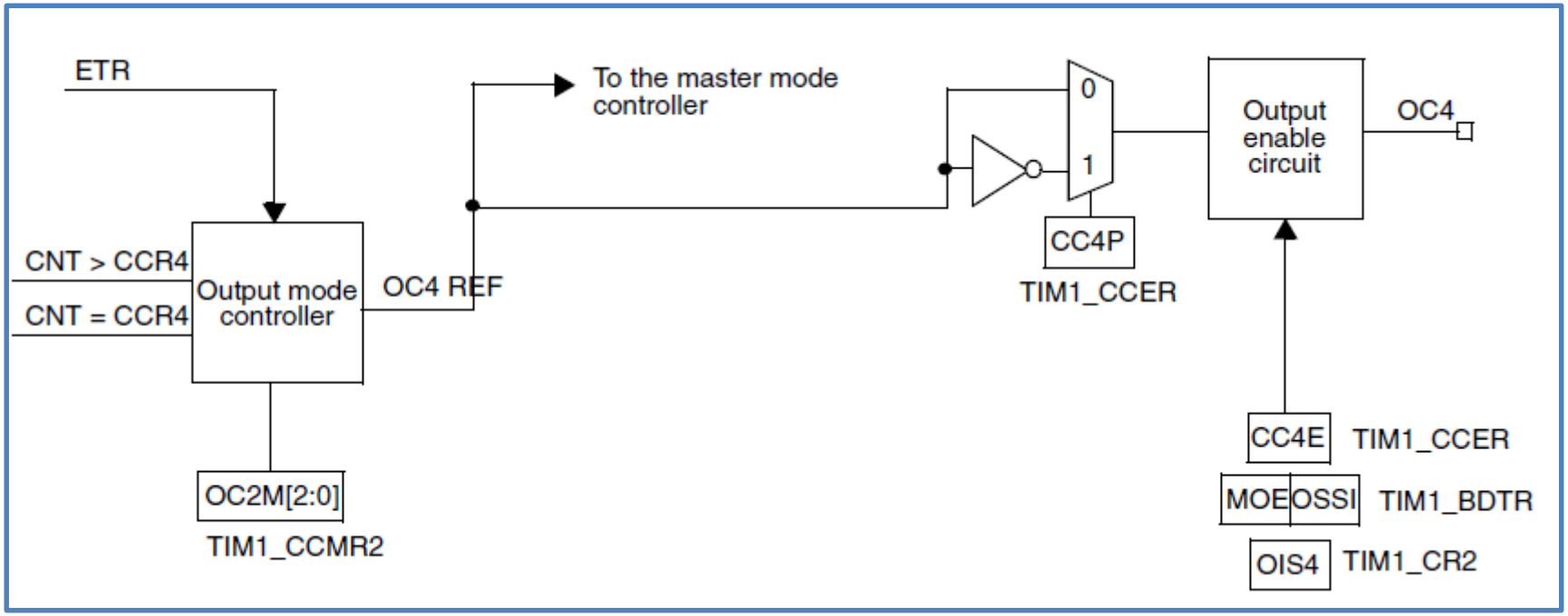
- Komplementarni izlazi sa programabilnim mrtvim vremenom.
- Break signal koji jednovremeno deaktivira sve izlaze.
- Brojač ponavljanja koji obezbeđuje ažuriranje tajmerskih registara tek posle određenog broja ciklusa.
- Interfejs prema trofaznom hall-effect senzoru.

Izlazni stepen Output Capture jedinice – Kanali 1,2,3



TIM1 - Kanal 4

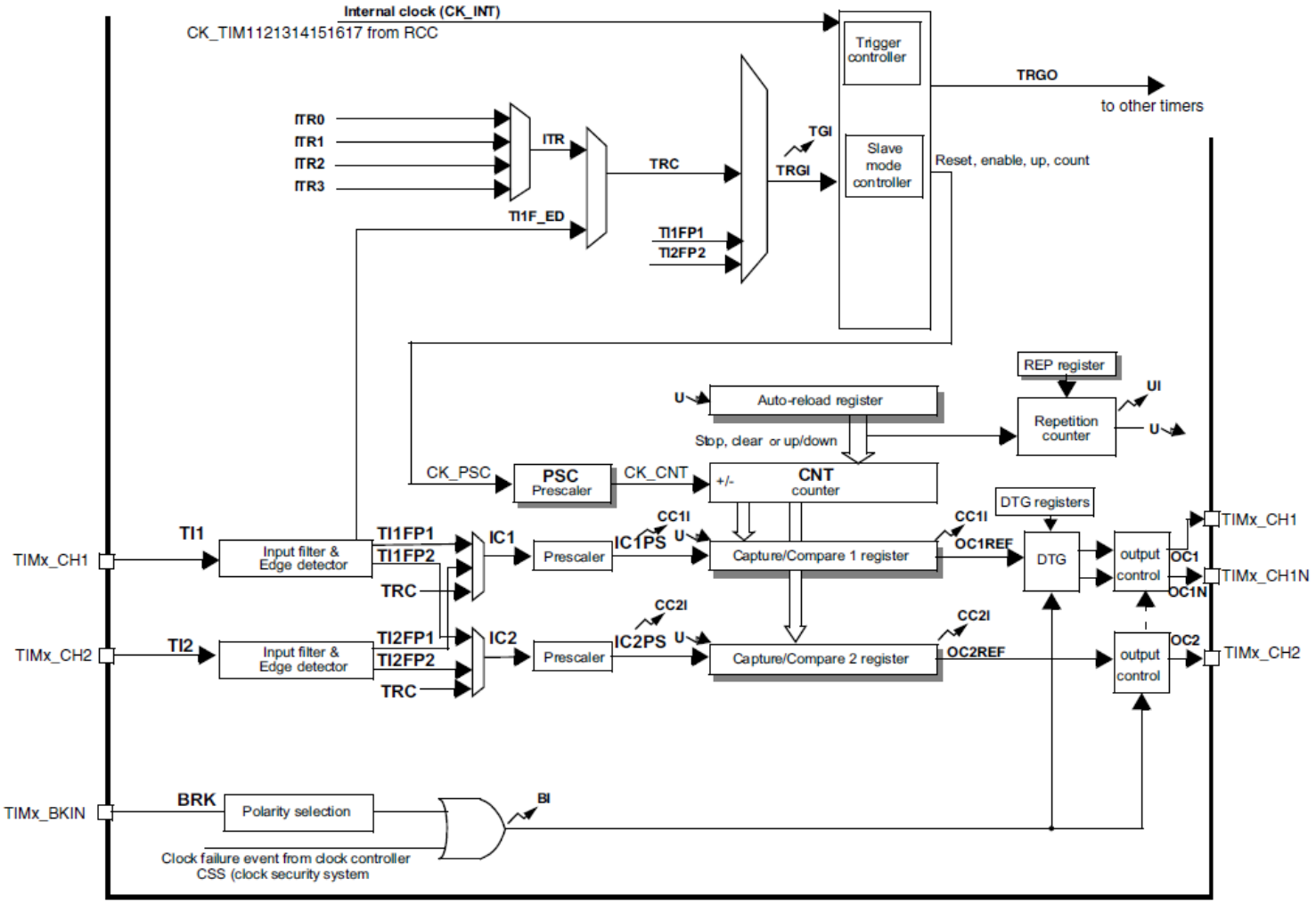
- Kanal 4 je pojednostavljene strukture jer se TIM1 obično koristi u trofaznim PWM generatorima u kojima se sedmi kanal koristi uglavnom za potrebe “kočenja”.



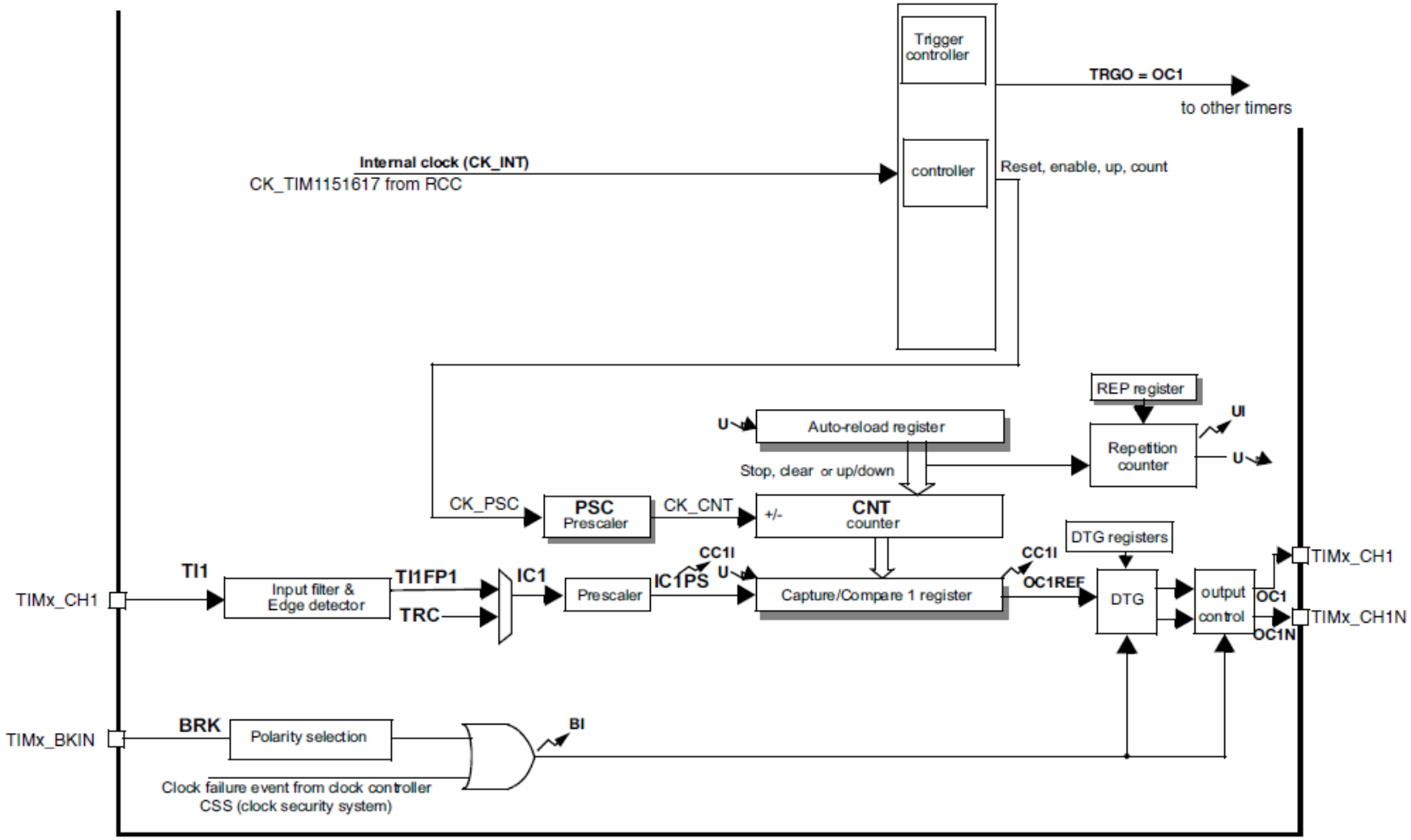
Tajmeri TIM15, TIM16, TIM17

- 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 comapre), 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


Internal clock (CK_INT)
CK_TIM1121314151617 from RCC
Trigger controller
TRGO
to other timers
ITR0
ITR1
ITR2
ITR3
ITR
TRC
TGI
TRGI
Slave mode controller
Reset, enable, up, count
TI1F_ED
TI1FP1
TI2FP2
REP register
UI
U
Auto-reload register
Repetition counter
Stop, clear or up/down
CK_PSC
PSC Prescaler
CK_CNT
+/-
CNT counter
DTG registers
TI1
TIMx_CH1
Input filter & Edge detector
TI1FP1
TI1FP2
TRC
IC1
CC1I
IC1PS
Prescaler
Capture/Compare 1 register
OC1REF
DTG
output control
OC1
OC1N
TIMx_CH1
TIMx_CH1N
TI2
TIMx_CH2
Input filter & Edge detector
TI2FP1
TI2FP2
TRC
IC2
CC2I
IC2PS
Prescaler
Capture/Compare 2 register
OC2REF
output control
OC2
TIMx_CH2
TIMx_BKIN
BRK
Polarity selection
BI
Clock failure event from clock controller
CSS (clock security system

TIM16, TIM17


Trigger controller
TRGO = OC1
to other timers
Internal clock (CK_INT)
CK_TIM1151617 from RCC
controller
Reset, enable, up, count
REP register
UI
U
Auto-reload register
Repetition counter
U
Stop, clear or up/down
CK_PSC
PSC
Prescaler
CK_CNT
+/-
CNT
counter
DTG registers
CC1I
CC1I
TI1
TIMx_CH1
Input filter & Edge detector
TI1FP1
TRC
IC1
Prescaler
IC1PS
U
Capture/Compare 1 register
OC1REF
DTG
output control
OC1
OC1N
TIMx_CH1
TIMx_CH1N
BRK
TIMx_BKIN
Polarity selection
BI
Clock failure event from clock controller
CSS (clock security system)

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.
- Podržan je DMA prenos.
- Uvek postoji jedan komplementarni izlaz.
- Brojač ponavljanja.