

PRIMENA MIKROKONTROLERA- MS1PMK

6. deo

2017

Nenad Jovičić

MBED

Sistemiški 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 flash(LED2);
void flip() {
    flash = !flash;
}
int main() {
    flash = 1;
    flipper.attach(&flip, 0.5);           // the address of the function to be attached
                                        // 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(LED2);
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) ;
}
```

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 led2(LED2);

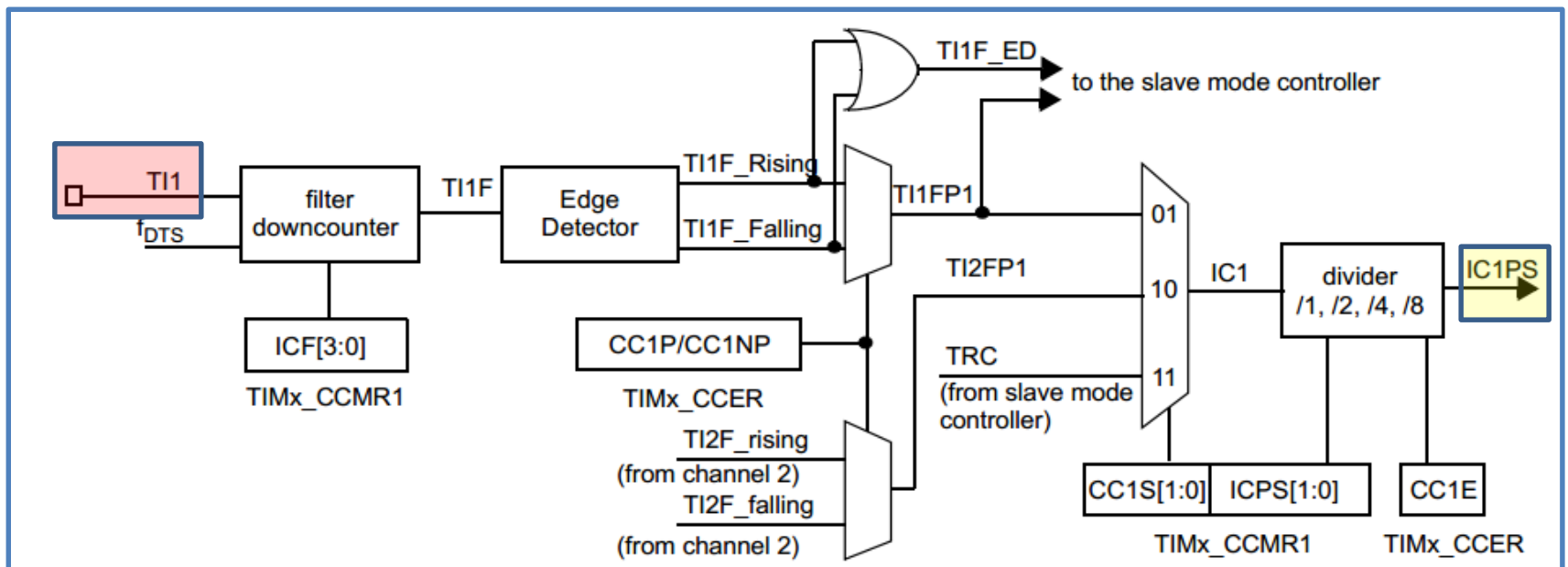
void flip() {
    led2 = !led2;
}

int main() {
    led2 = 1;
    flipper.attach(&flip, 2.0); // setup flipper to call flip after 2
    seconds

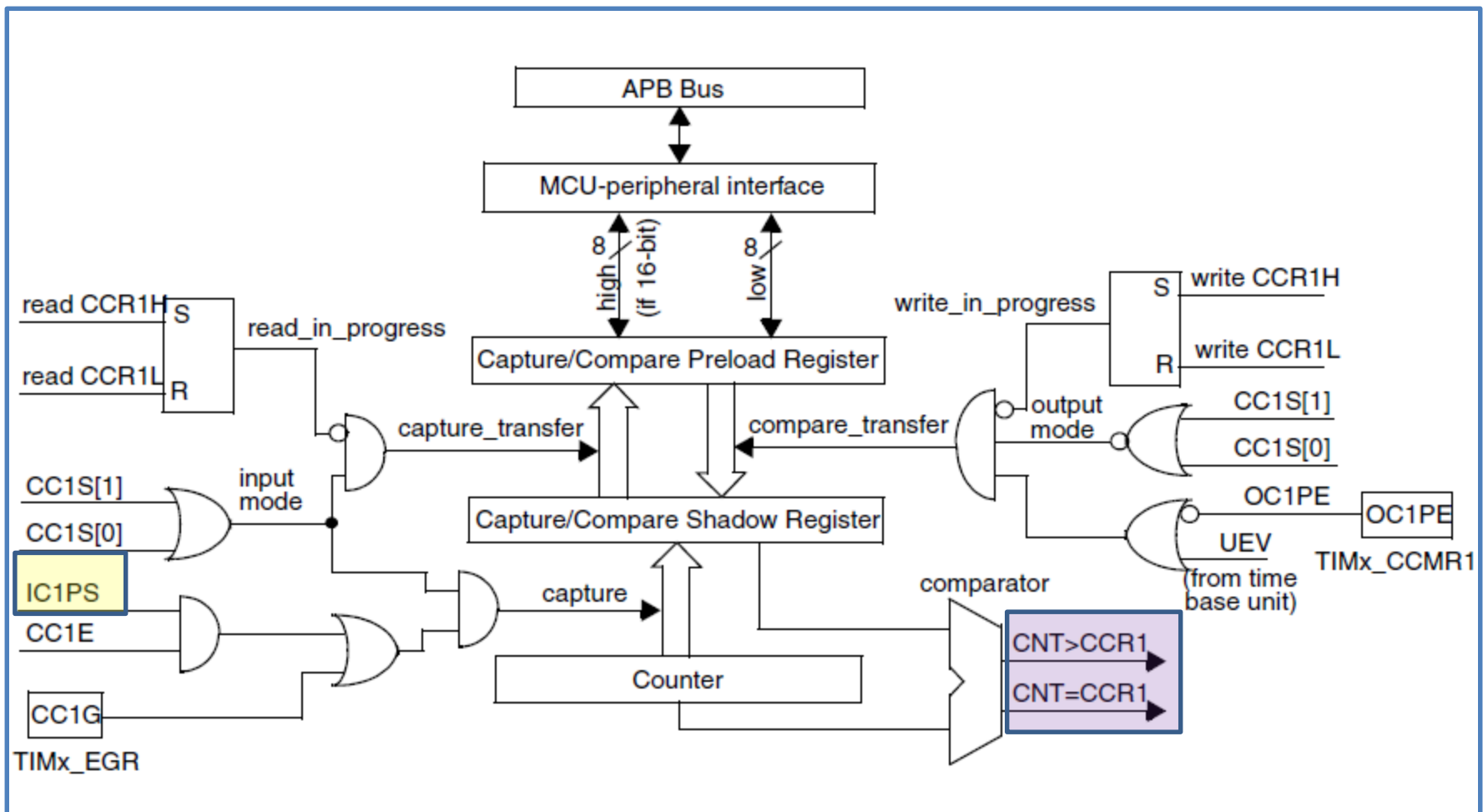
    // spin in a main loop. flipper will interrupt it to call flip
    while(1) ;
}
```

Capture/compare jedinica input capture deo

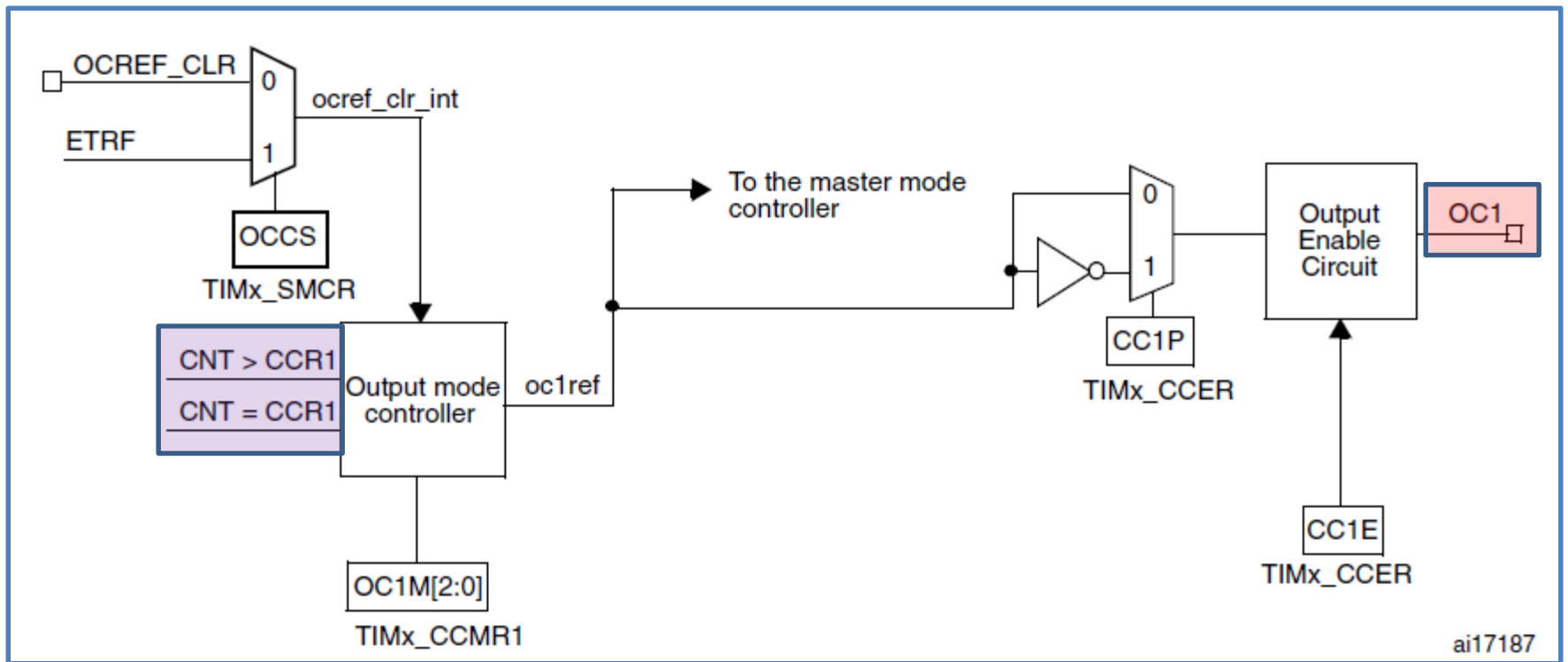
- Svaki capture događaj može da generiše prekid ili DMA zahtev.



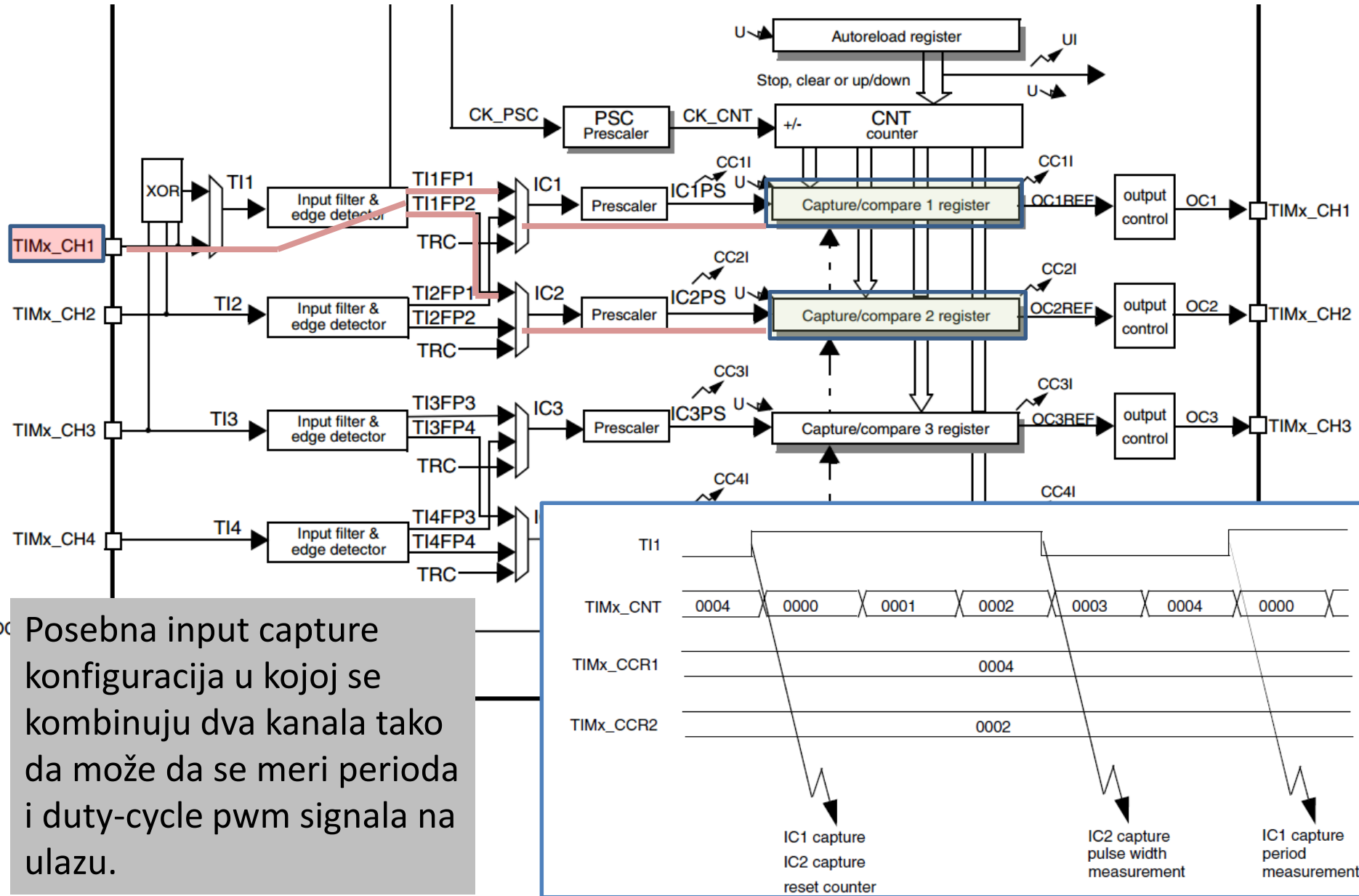
Capture/compare jedinica centralni deo



Capture/compare jedinica output compare deo



PWM input capture

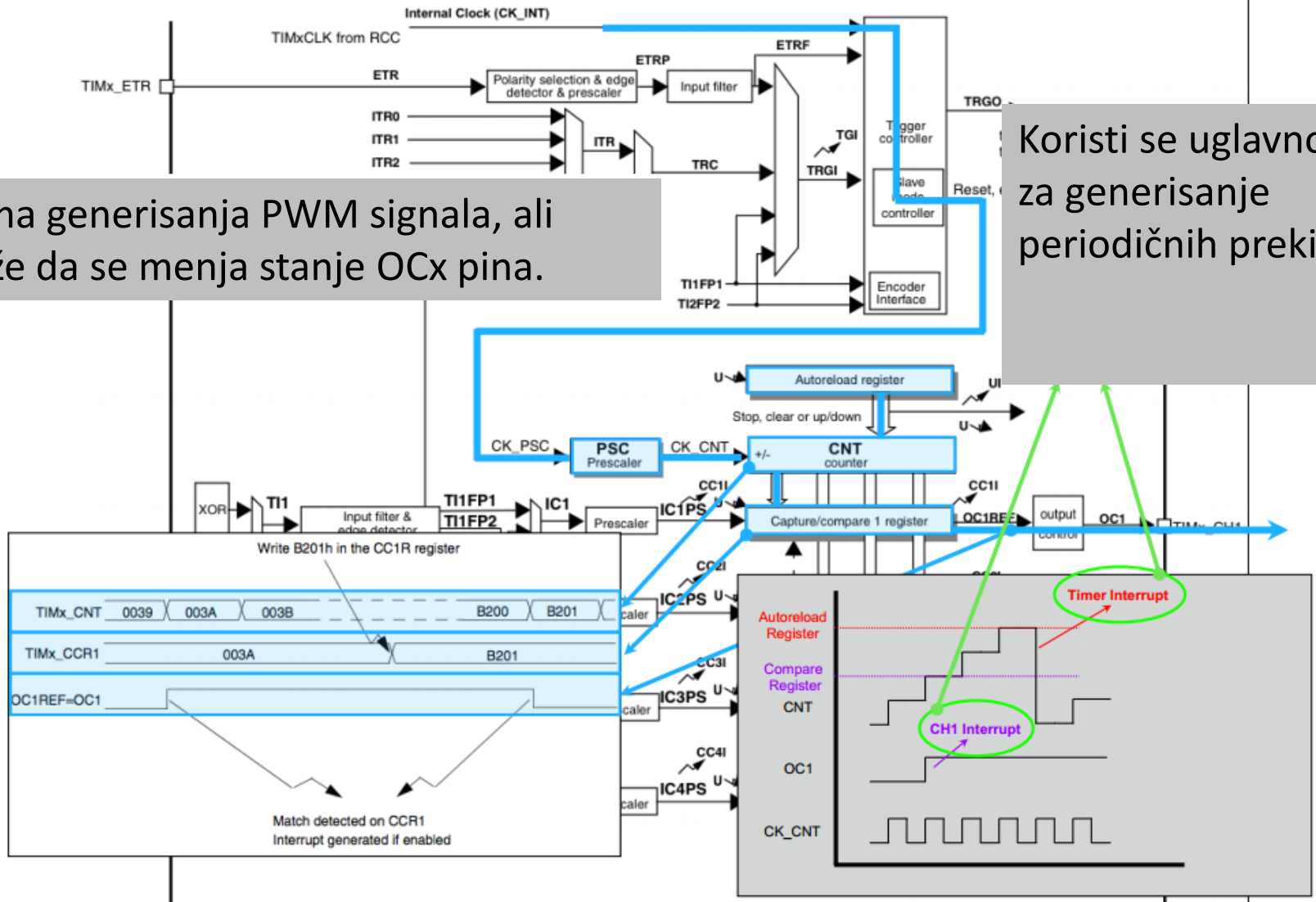


Posebna input capture konfiguracija u kojoj se kombinuju dva kanala tako da može da se meri perioda i duty-cycle pwm signala na ulazu.

Output compare mod

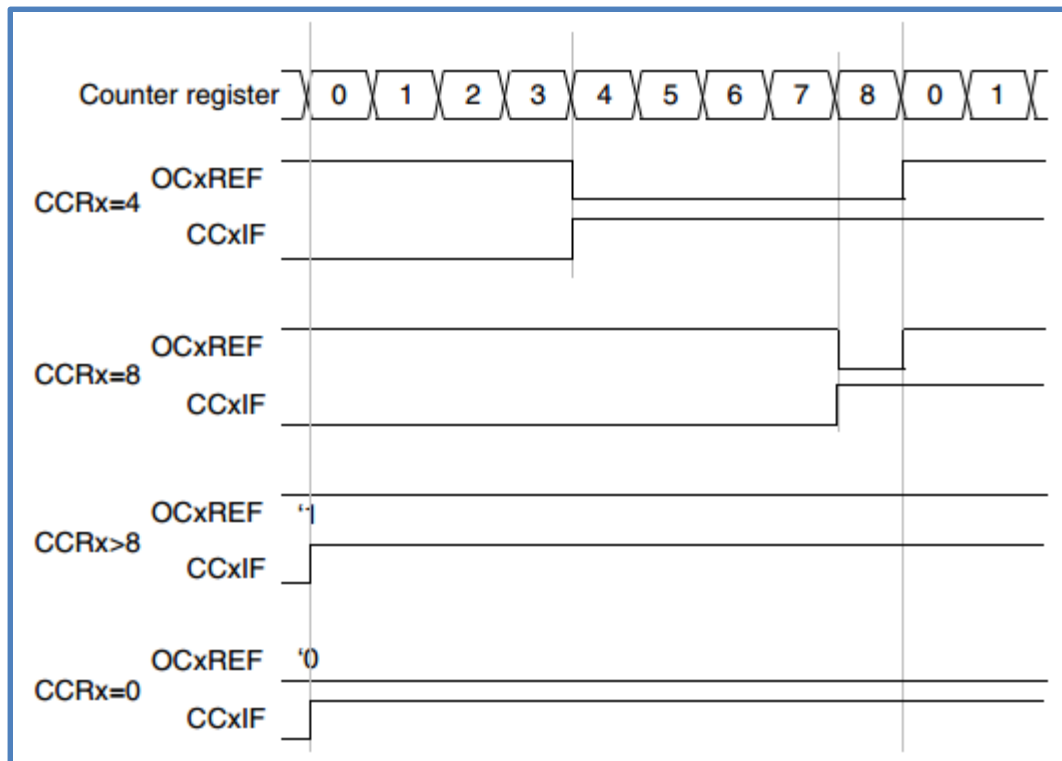
Nema generisanja PWM signala, ali može da se menja stanje OCx pina.

Koristi se uglavnom za generisanje periodičnih prekida.



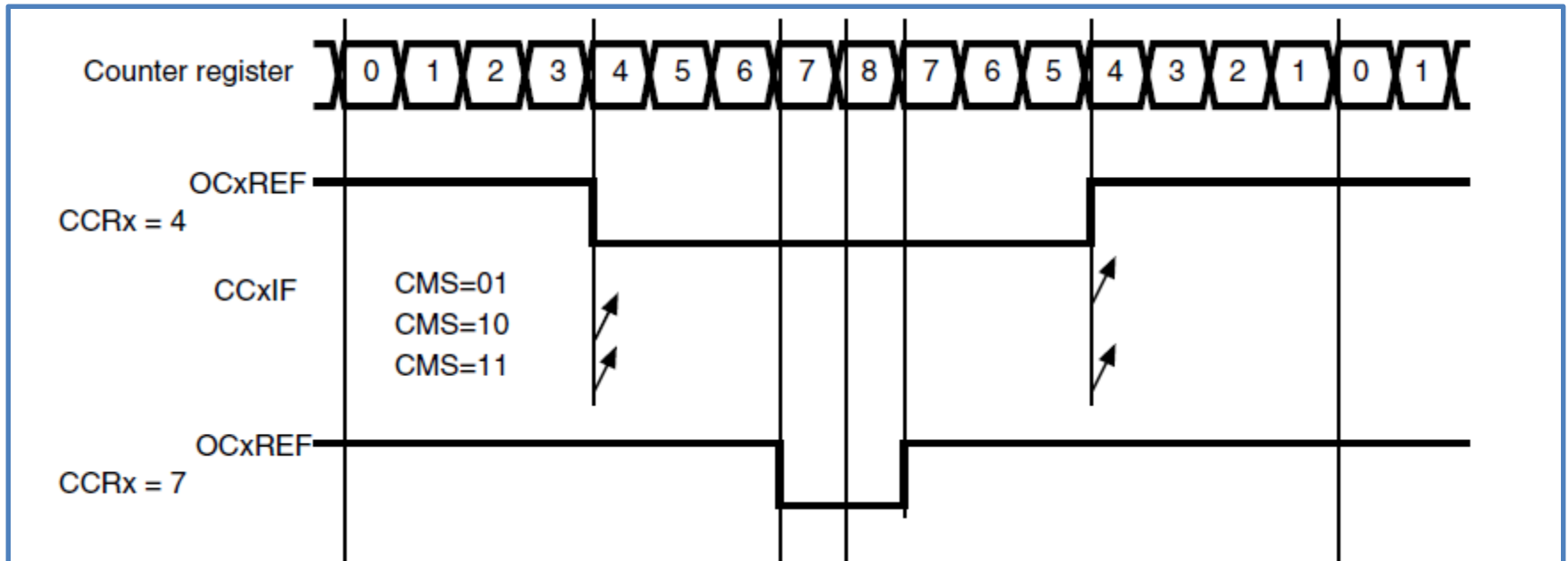
Edge-aligned PWM

- U ovom modu brojač može da radi ili u UP ili u DOWN modu.



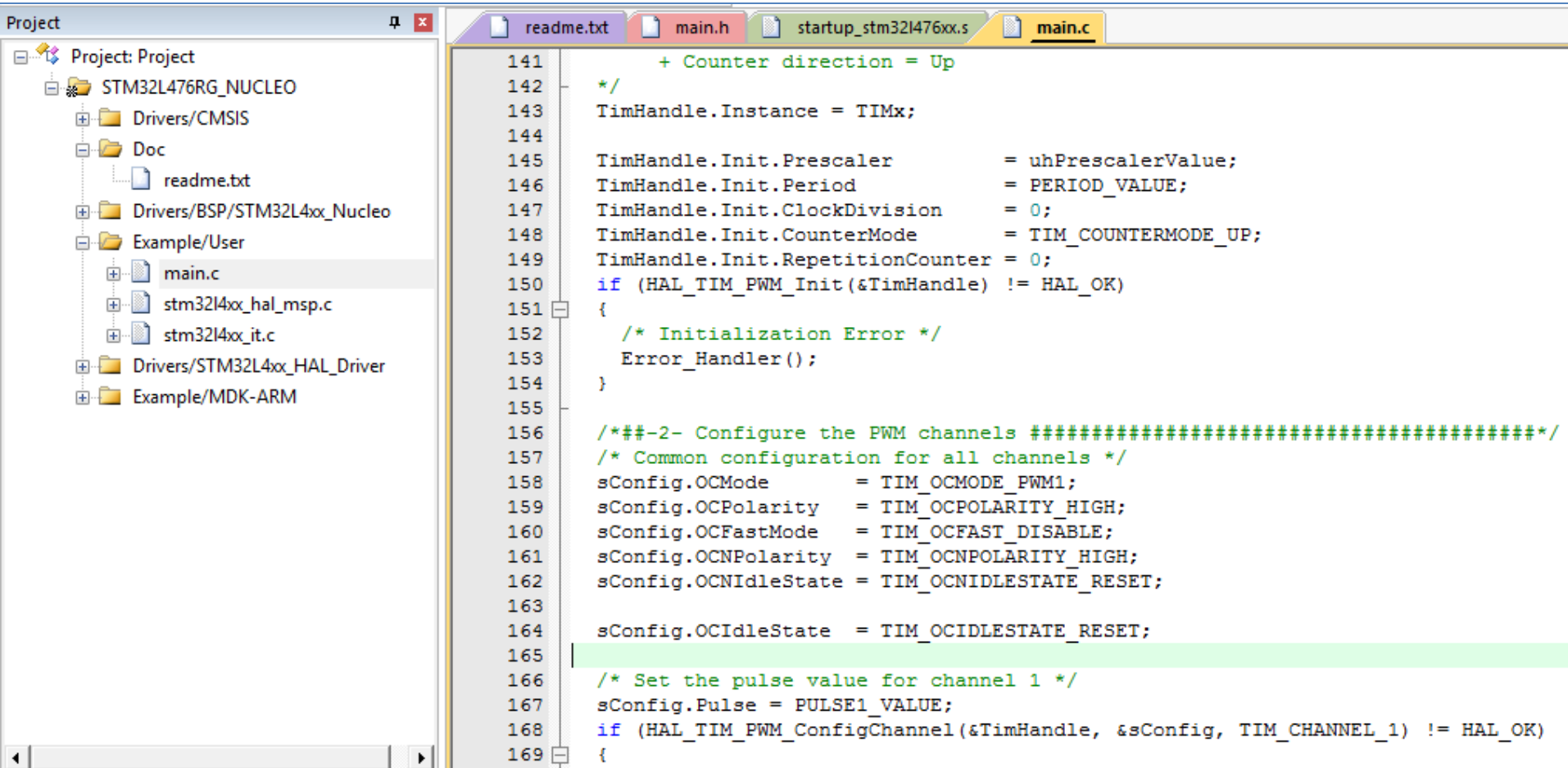
Center-aligned PWM

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



STM CUBE

Projektat TIM_PWMOutput



The screenshot displays the STM32CubeIDE interface. On the left, the 'Project' pane shows the project structure for 'Project: Project' on an STM32L476RG_NUCLEO board. The project files include 'Drivers/CMSIS', 'Doc', 'Drivers/BSP/STM32L4xx_Nucleo', and 'Example/User'. The 'main.c' file is selected in the 'Example/User' folder.

The main editor window shows the code in 'main.c'. The code is as follows:

```
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

```

92  */
93  typedef struct
94  {
95      uint32_t OCMODE;          /*!< Specifies the output compare mode.
96                               This parameter can be a value of @ref TIM_Output_Compare_Mode */
97
98      uint32_t Pulse;         /*!< Specifies the pulse width for the output compare.
99                               This parameter can be a value of @ref TIM_Output_Compare_Pulse */
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;

```

```

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;

```

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.

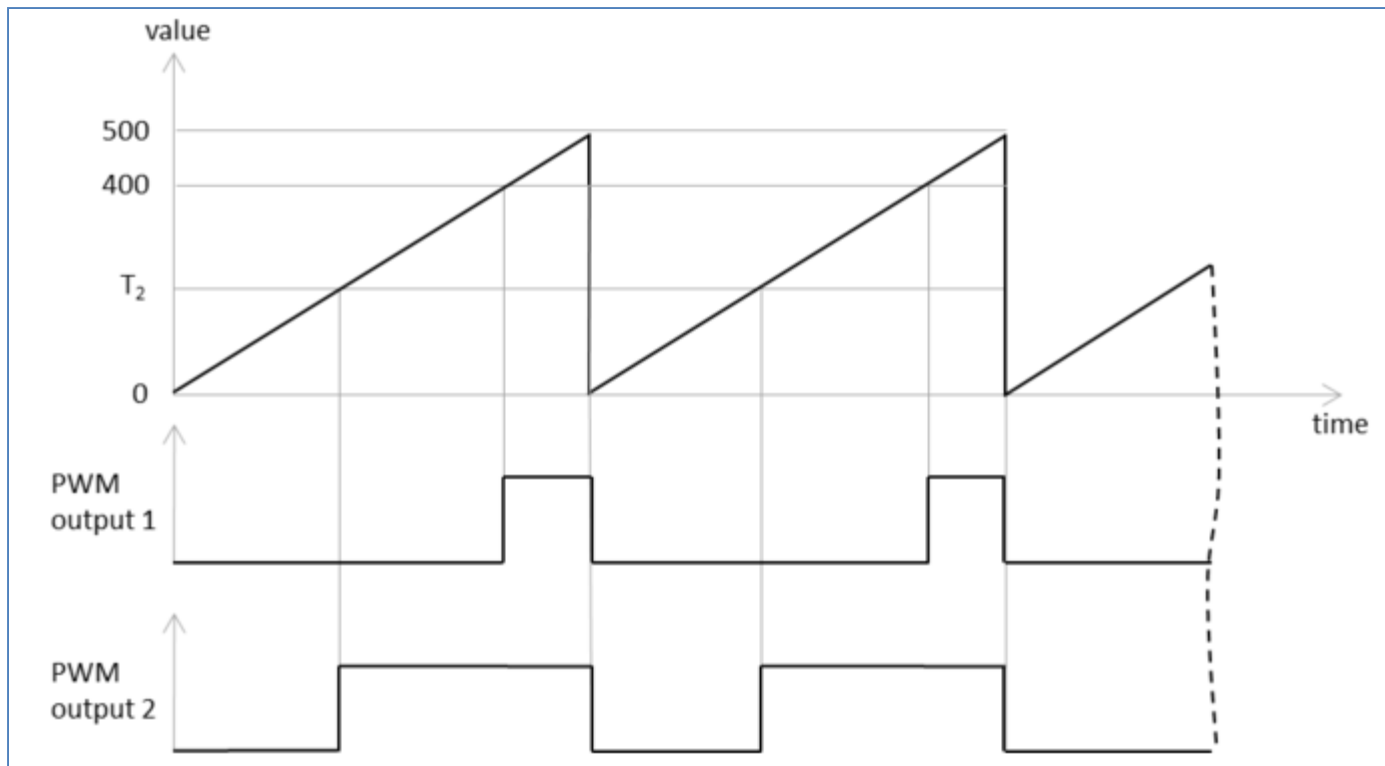
1000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger

```

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



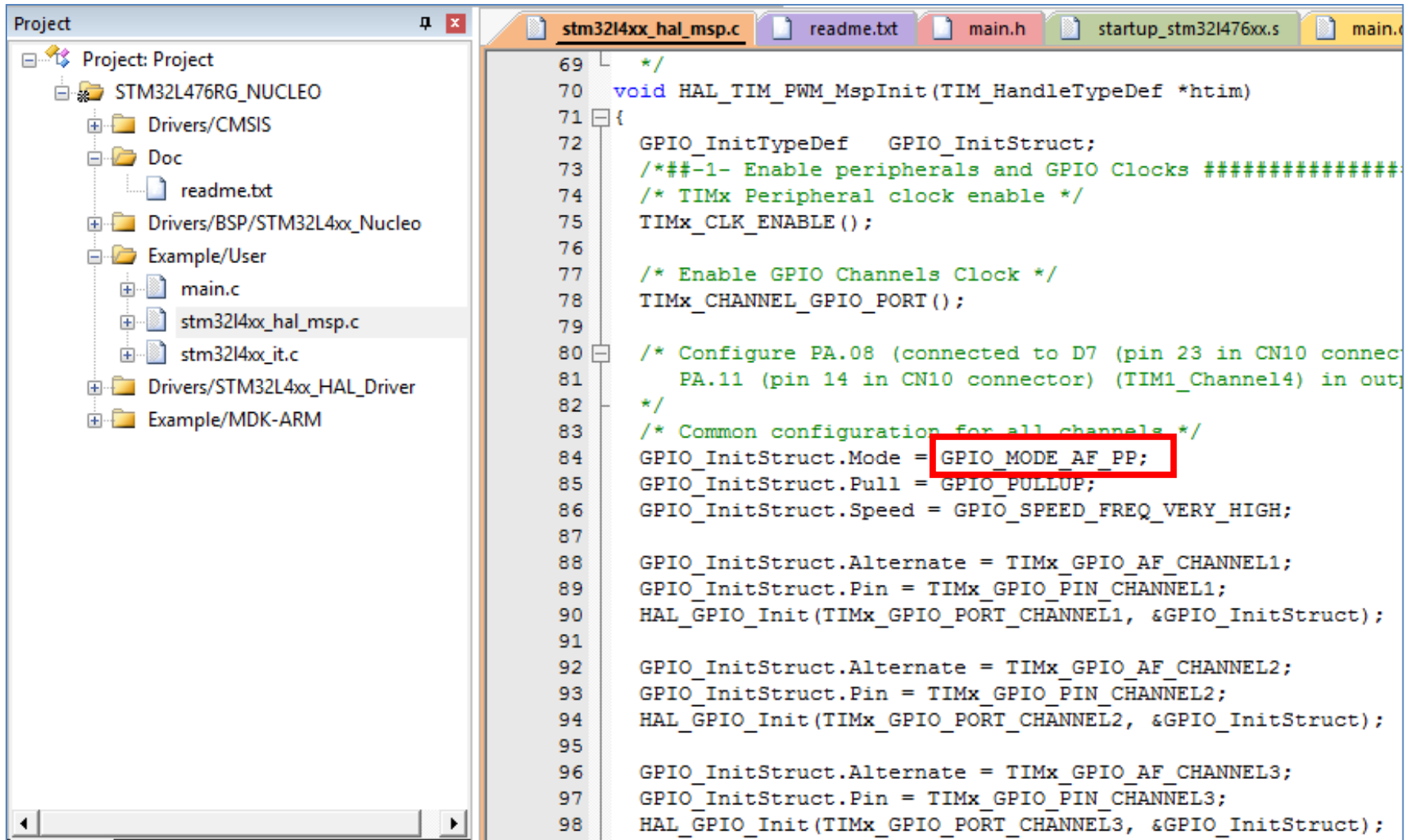
Podlašavanje svakog OC kanala

```
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_stm321476xx.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 panes. The left pane 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 pane 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);
```



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

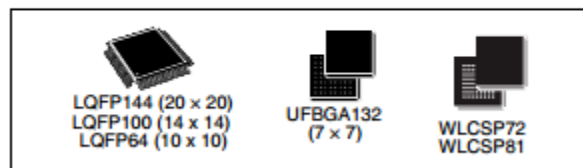


Table 15. STM32L476xxSTM32L476xx pin definitions (continued)

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_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	-

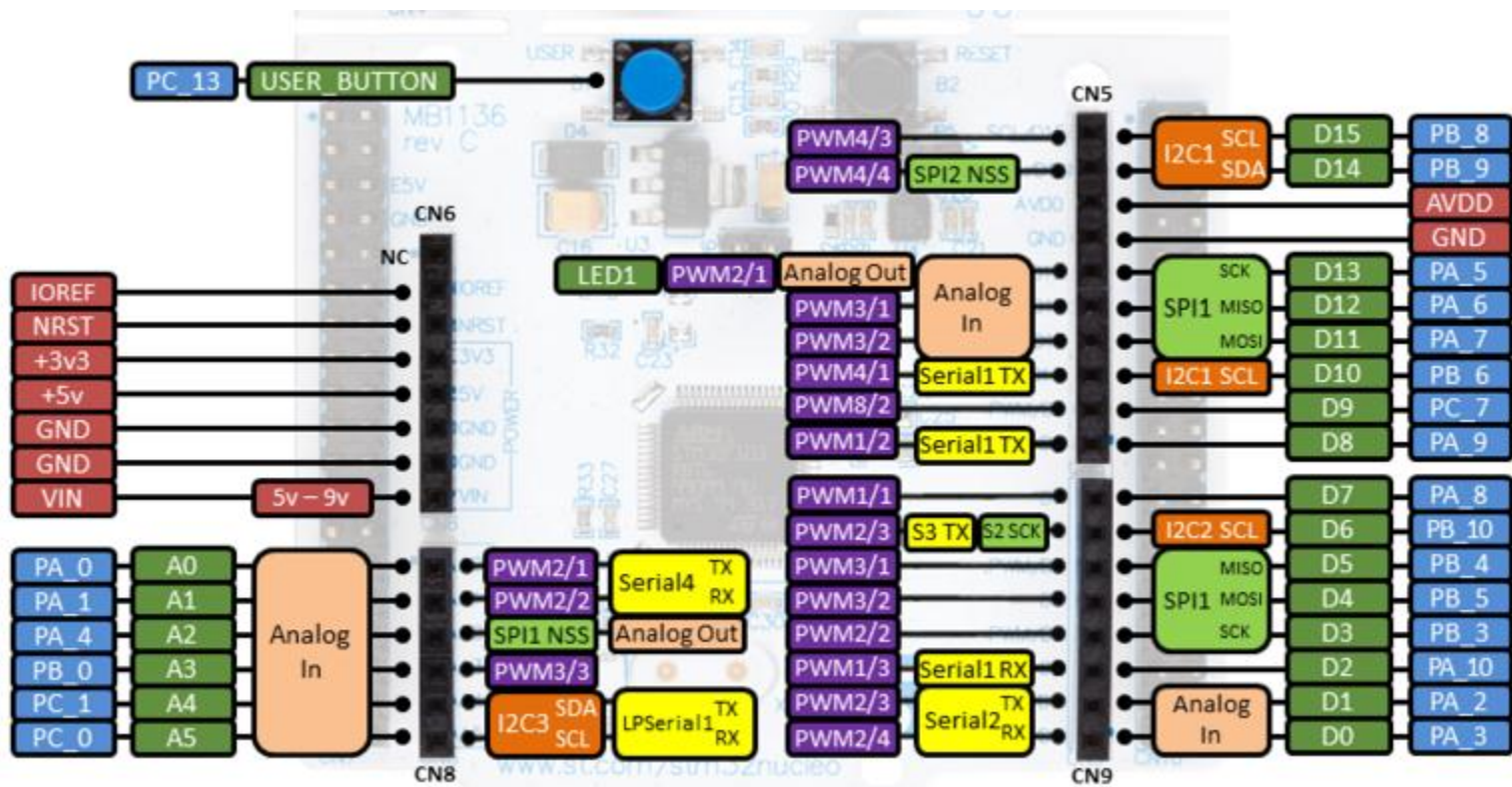
MBED

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);
        }
    }
}
```

Na kojim to pinovima može da se generiše PWM



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\(\)](#)

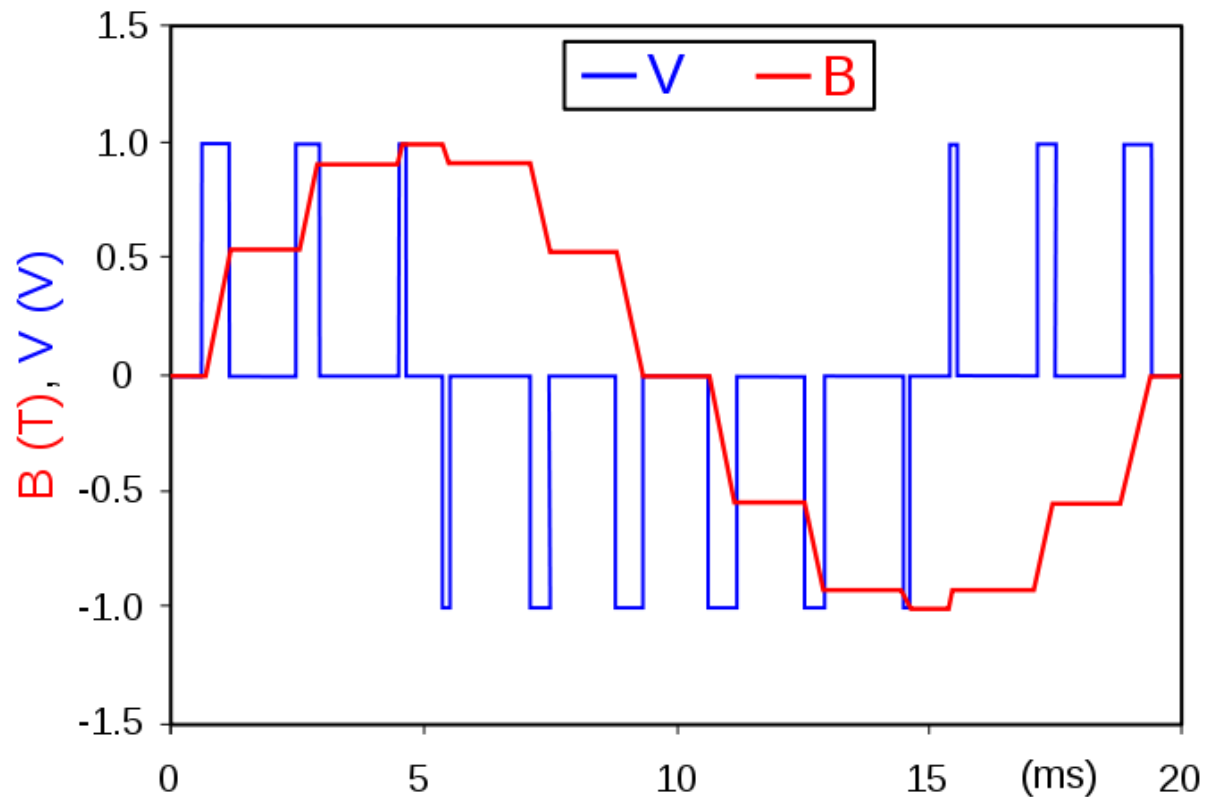
[operator float](#) ()

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

STM CUBE

Projekat TIM_PWMInput

- Merenje karakteristika PWM signala uz pomoć tajmera



Prametri za IC

```
main.c  stm3214xx_hal_msp.c  stm3214xx_hal_tim.c  startup_stm321476xx.s
49
50 /* Private typedef -----*/
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  stm3214xx_hal_msp.c  stm3214xx_hal_tim.c  stm3214xx_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;
}
```

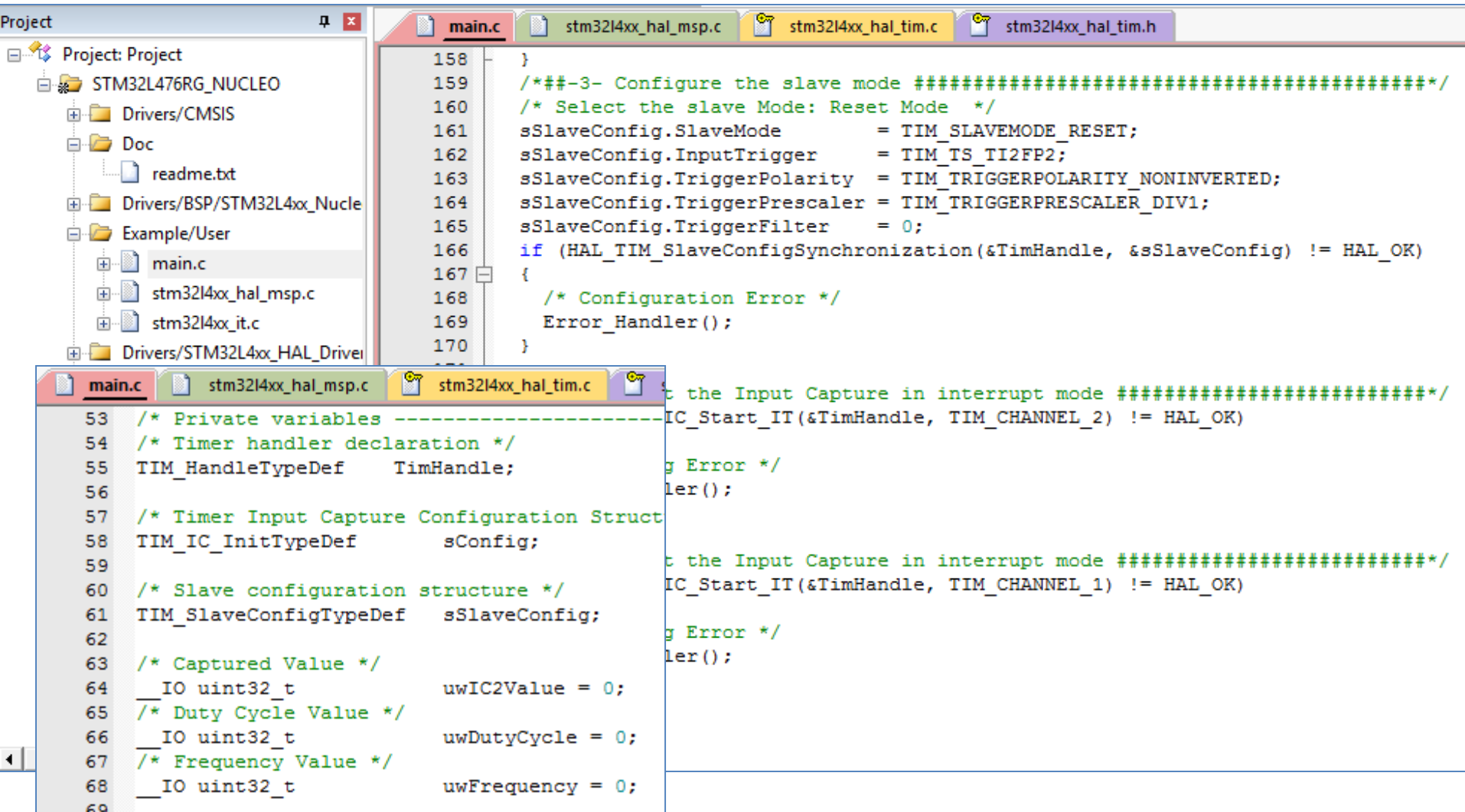
Podేశavanje svakog kanala

```
/*##-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



The image shows a screenshot of an IDE with a project tree on the left and code editors on the right. The project tree shows a project named 'Project' with a sub-project 'STM32L476RG_NUCLEO'. The code editors show the following code:

```
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     Error_Handler();
170 }
```

```
53 /* Private variables -----*/
54 /* Timer handler declaration */
55 TIM_HandleTypeDef TimHandle;
56
57 /* Timer Input Capture Configuration Structure */
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
```

```
/* the Input Capture in interrupt mode ******/
IC_Start_IT(&TimHandle, TIM_CHANNEL_2) != HAL_OK)
g Error */
ler();

/* the Input Capture in interrupt mode ******/
IC_Start_IT(&TimHandle, TIM_CHANNEL_1) != HAL_OK)
g Error */
ler();
```

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 /**
```

Neki slave modovi

Figure 245. Control circuit in reset mode

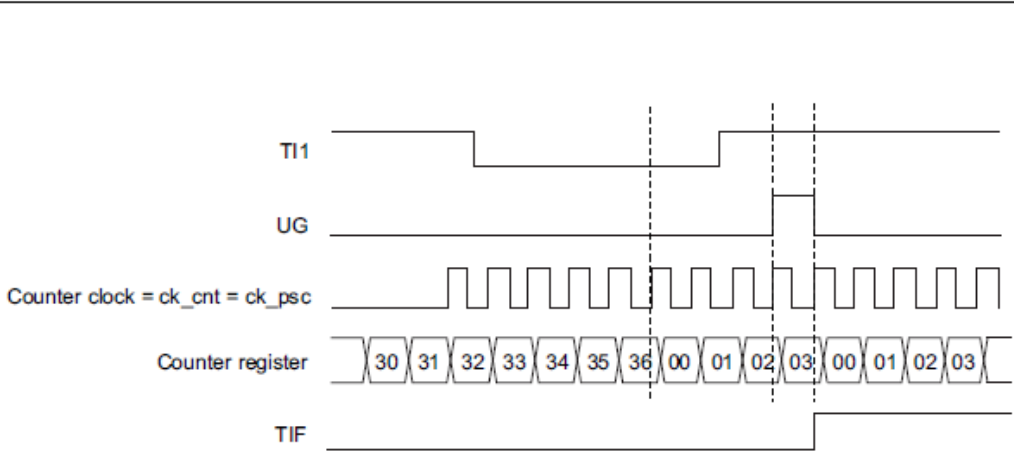


Figure 246. Control circuit in Gated mode

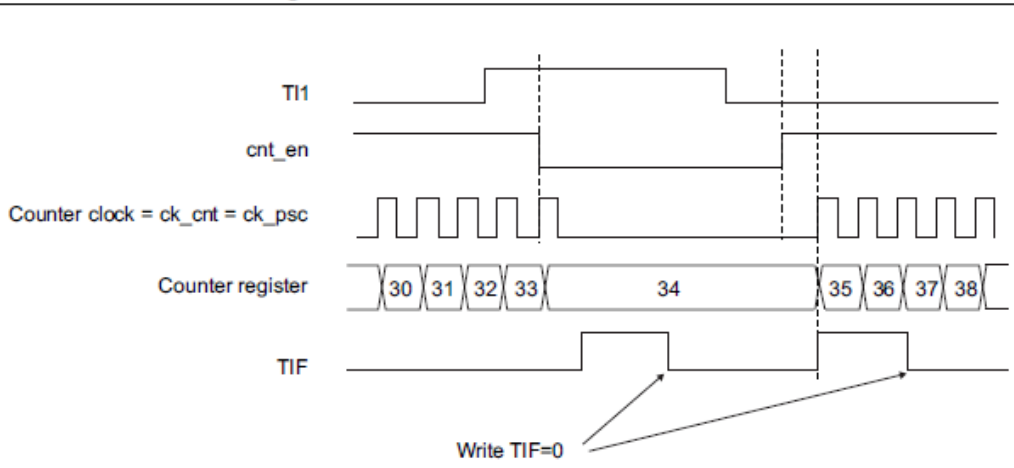
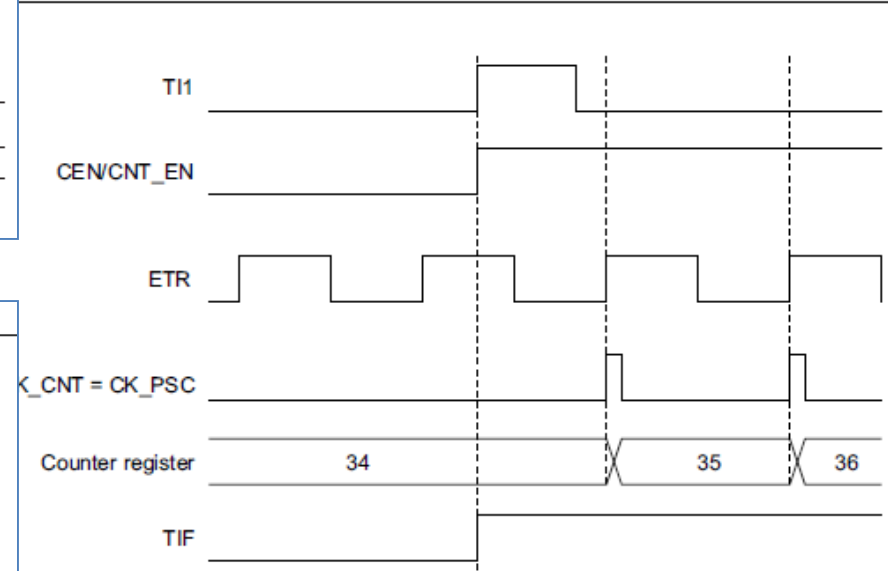
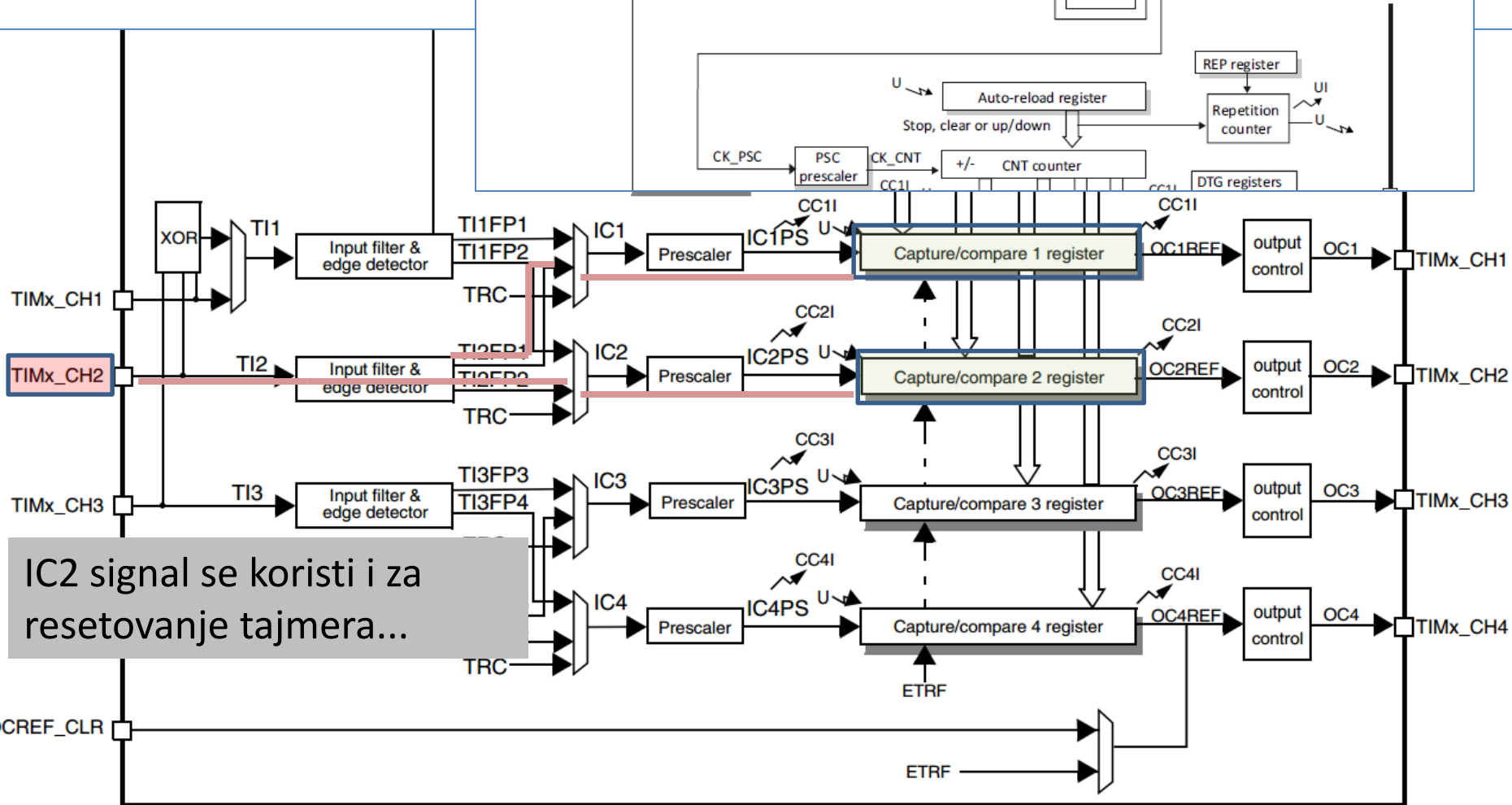
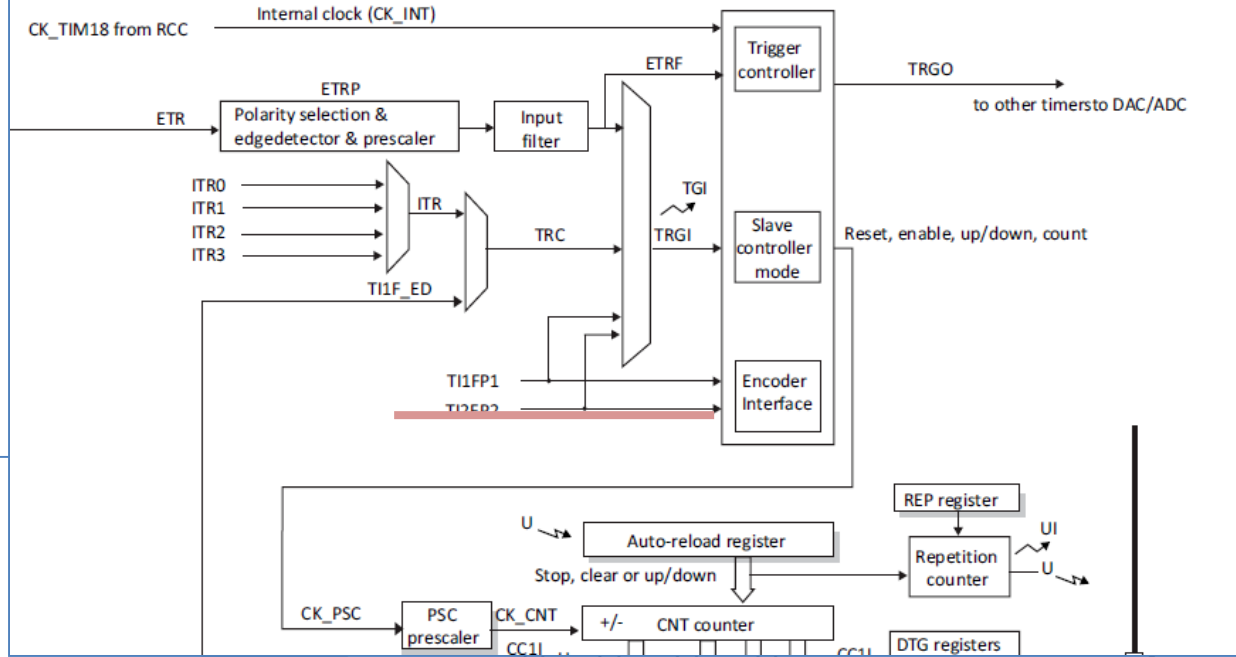


Figure 248. Control circuit in external clock mode 2 + trigger mode



PWM input capture



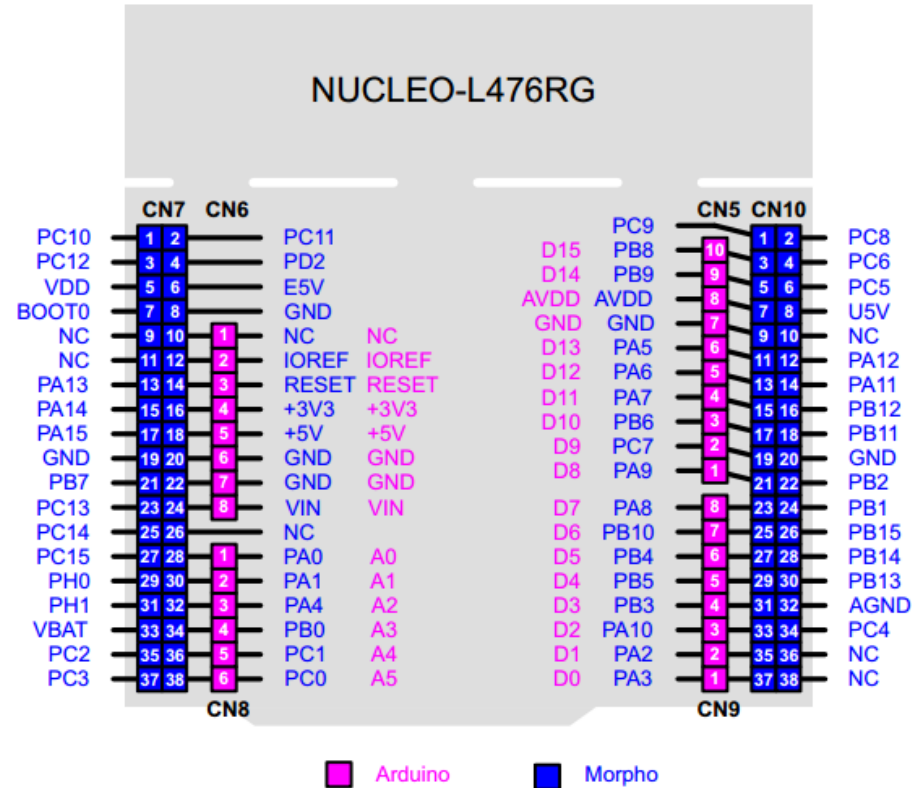
IC2 signal se koristi i za resetovanje tajmera...

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



- Jedna 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 illustrates the process of refreshing a variable in real-time within the KEIL IDE. The main window shows the source code for 'main.c' with the variable 'uwDutyCycle' highlighted. A context menu is open over the variable, with 'Add 'uwDutyCycle' to...' selected, which has opened a sub-menu showing 'Watch 1' as the chosen option. The 'Watch 1' window at the bottom right shows the current values: uwFrequency (24009), uwDutyCycle (49), and uwIC2Value (3328). The Command window shows the watchlist configuration, and the Find in Files window shows the search results 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

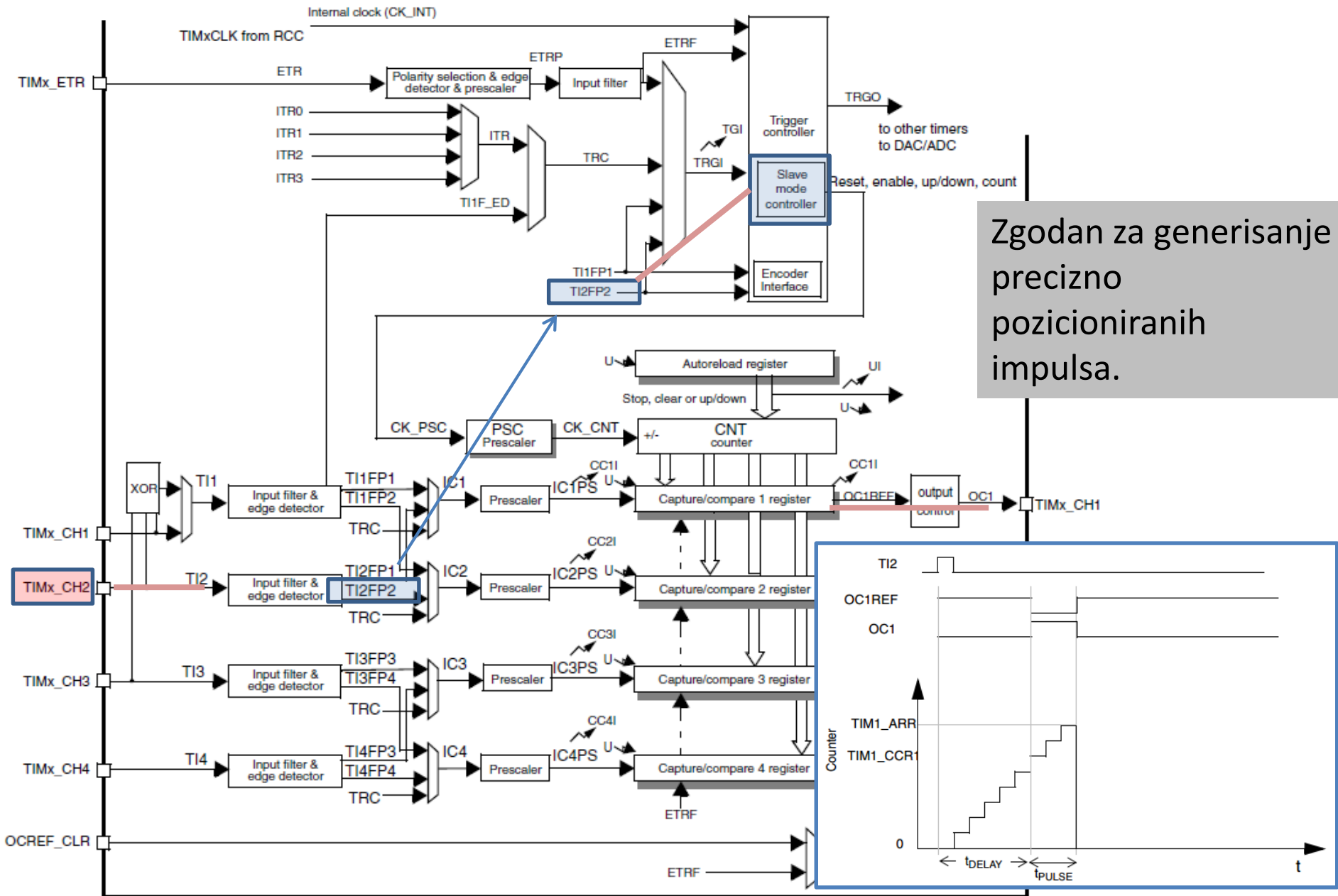
Find in Files

```
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
```

Zadatak

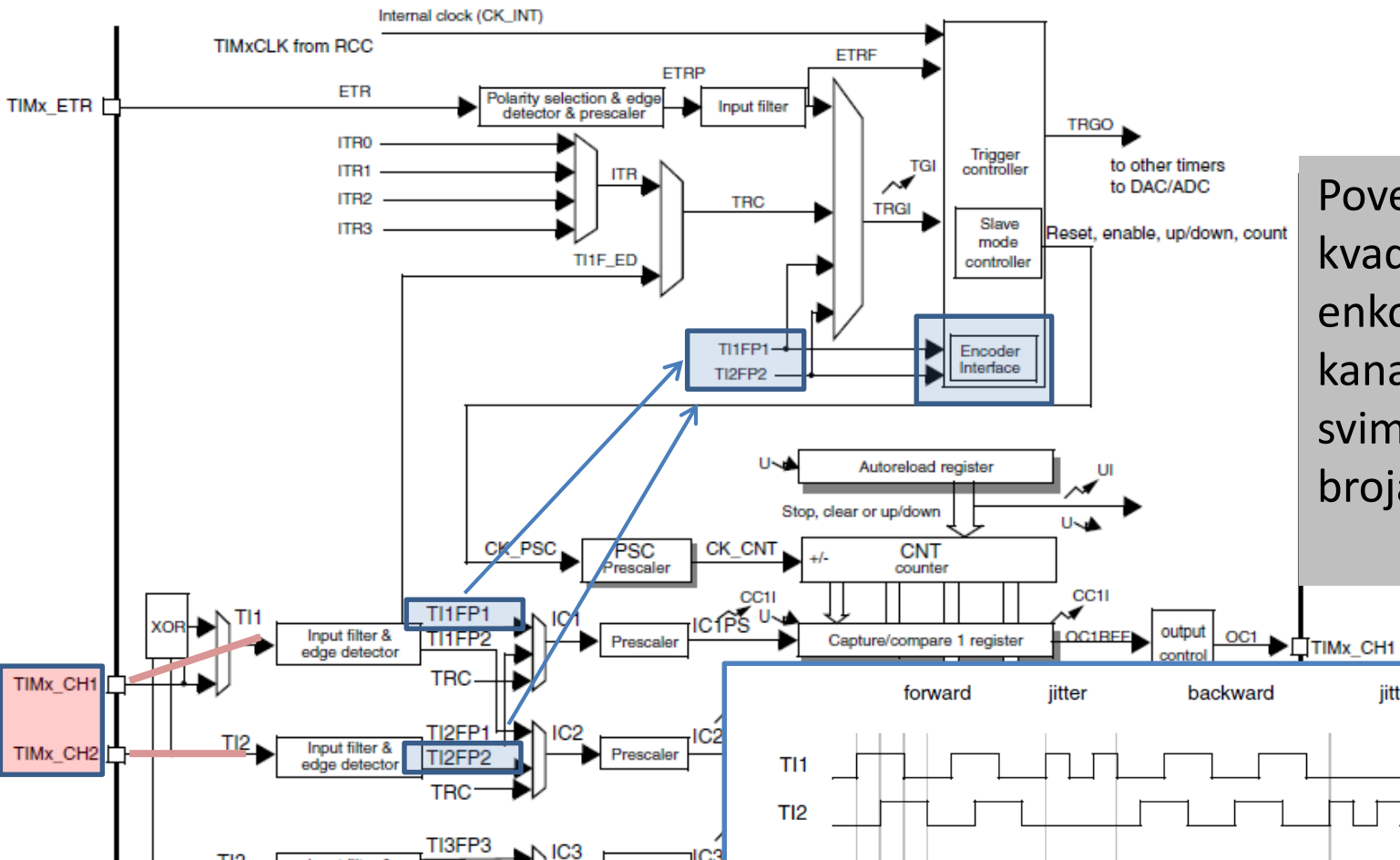
- Implementirati obe funkcionalnosti na MBED platformi.
- PWMOutput je izuzetno lako implementirati.
- PWMInput nema direktnu podršku...
- Ideja za PWMInput:
 - a) Uvede se jedan Ticker tajmer koji radi nekom velikom brzinom i inkrementira neku promenljivu *time*.
 - b) Ulazni signal se poveže na neka dva InterruptIn ulaza koji generišu prekide na rastuću i opadajuću ivicu.
 - c) U prekidima se očitava promenljiva *time* i na osnovu sukcesivnih vrednosti se računaju parametri..

One pulse mod

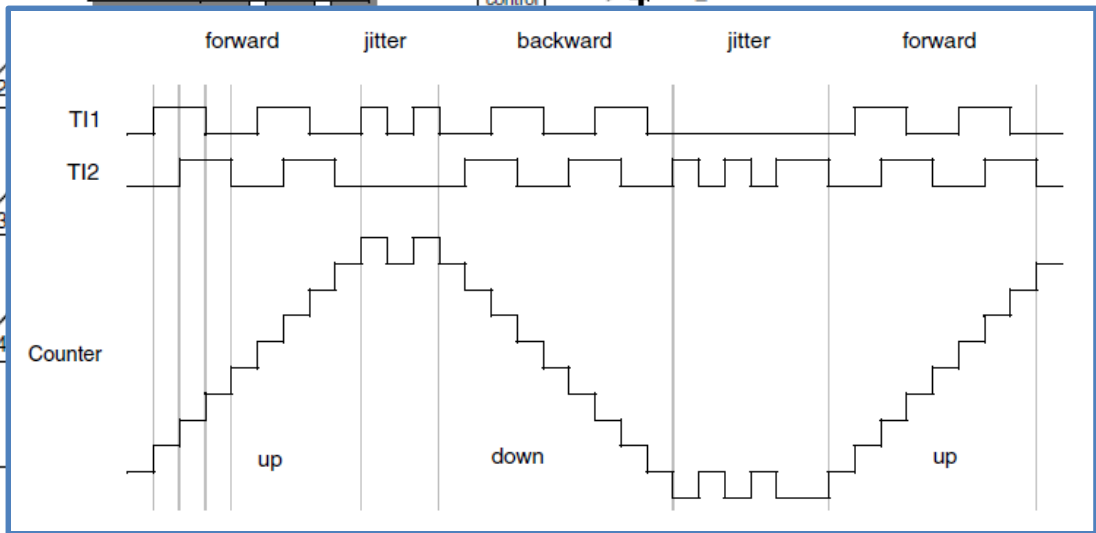


Zgodan za generisanje precizno pozicioniranih impulsa.

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...

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 import name "QEI". The target path is set to "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	
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

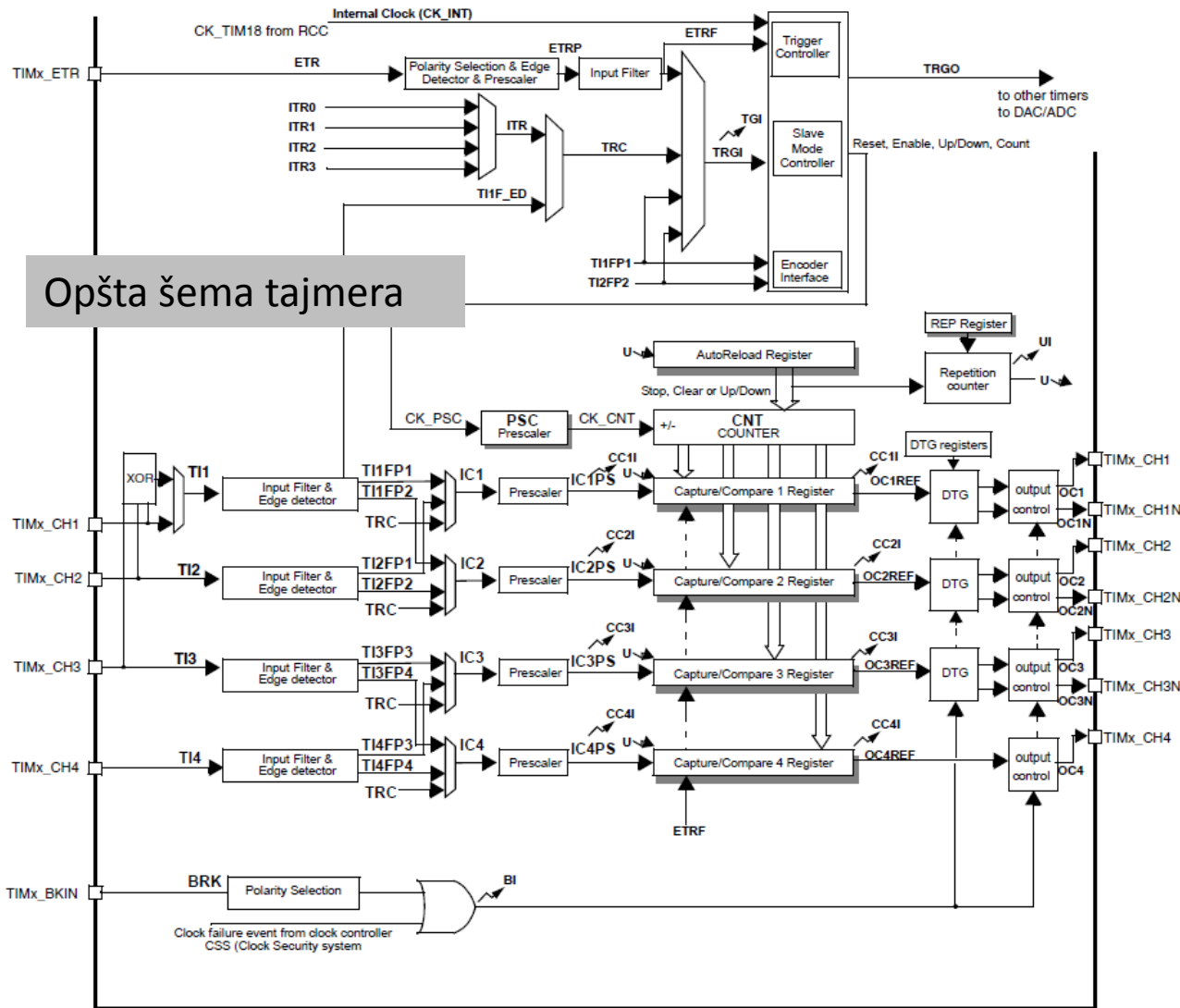
Tags

[Encoder QEI quadrature](#)

Description

Quadrature encoder interface library.

TIM1– Advanced control timer

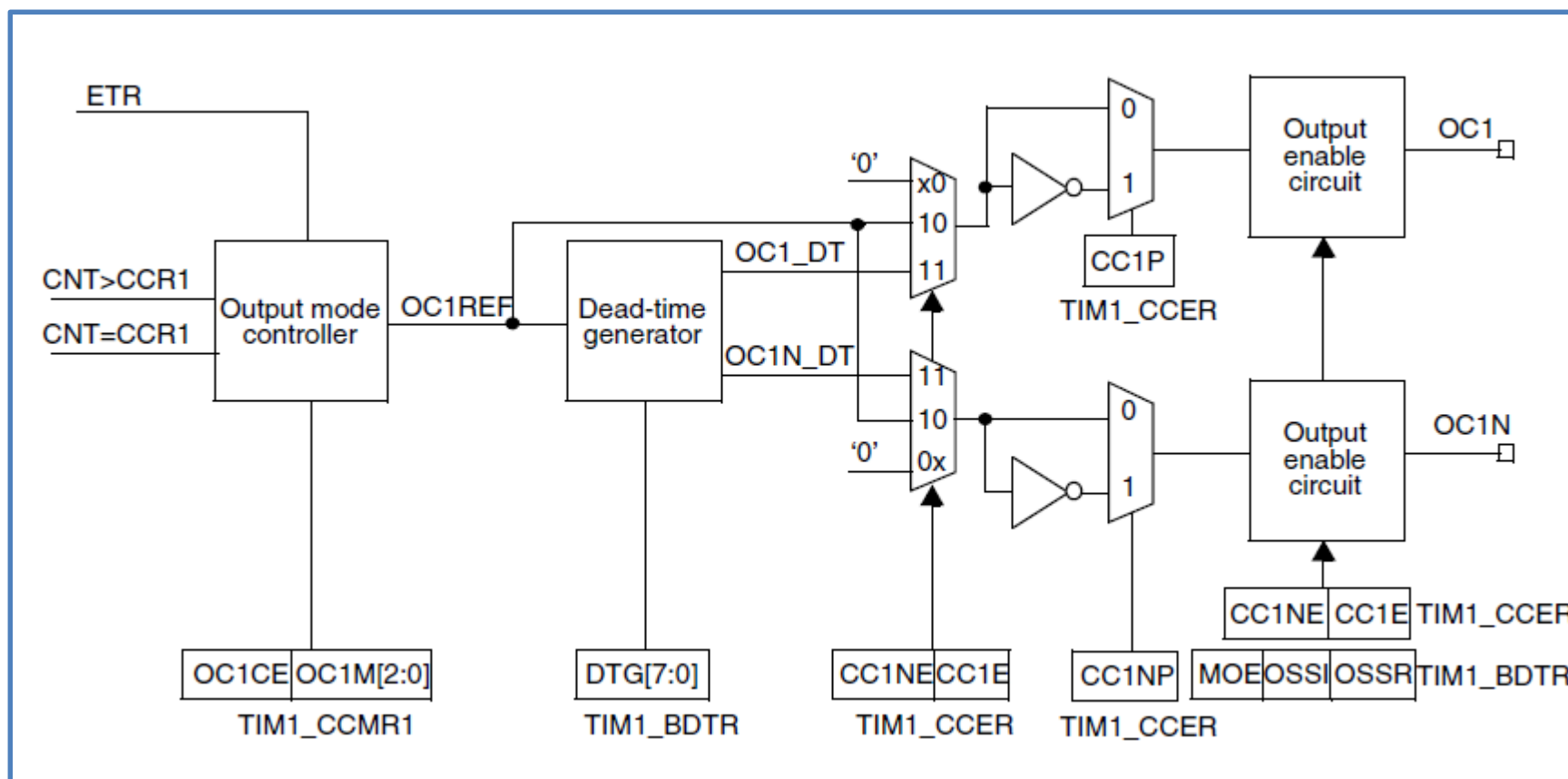


Opšta šema tajmera

- 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.

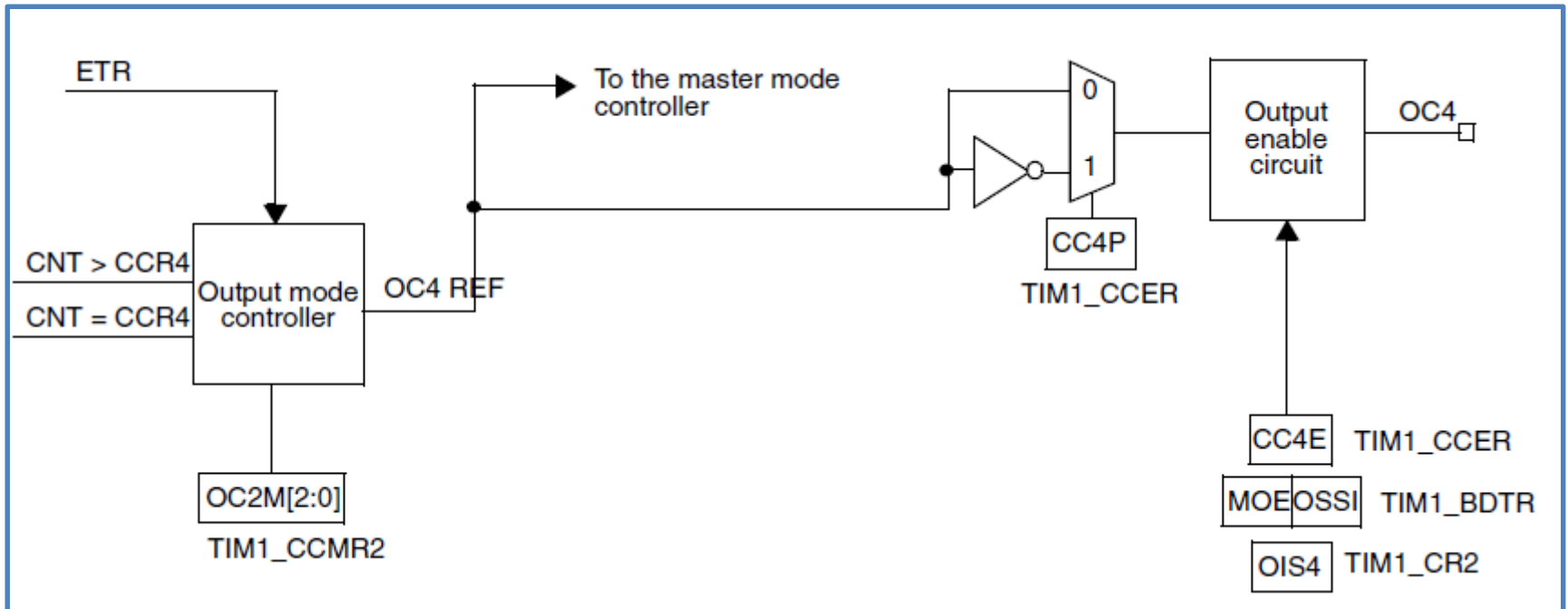
Izlazni stepen Output Capture jedinice

– Kanali 1,2,3



TIM1 - Kanal 4

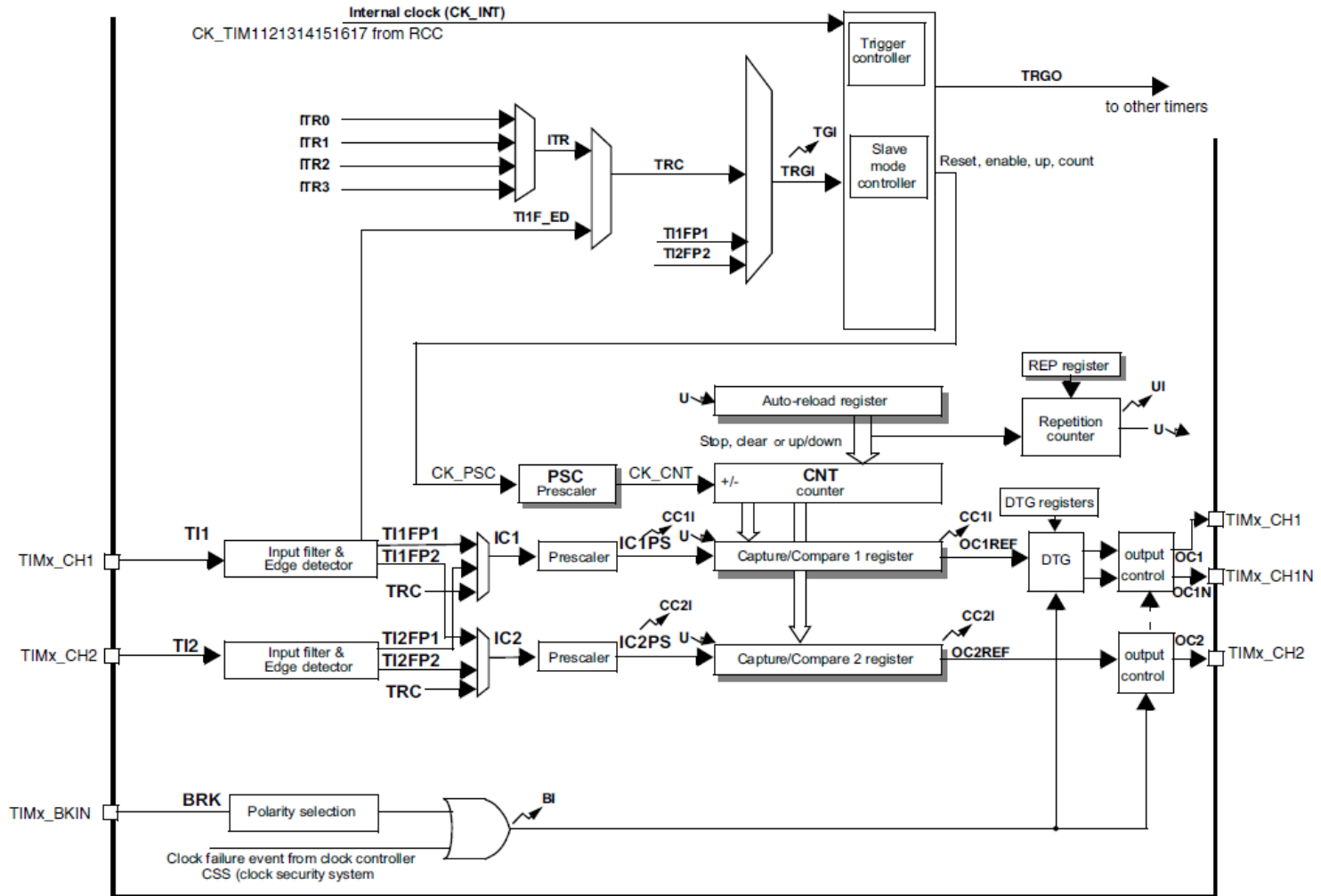
- Kanal 4 je pojednostavljene strukture jer se TIM1 obično koristi u trofaznim PWM generatorima u kojima se sedmi kanal koristi uglavnom za potrebe “kočenja”.



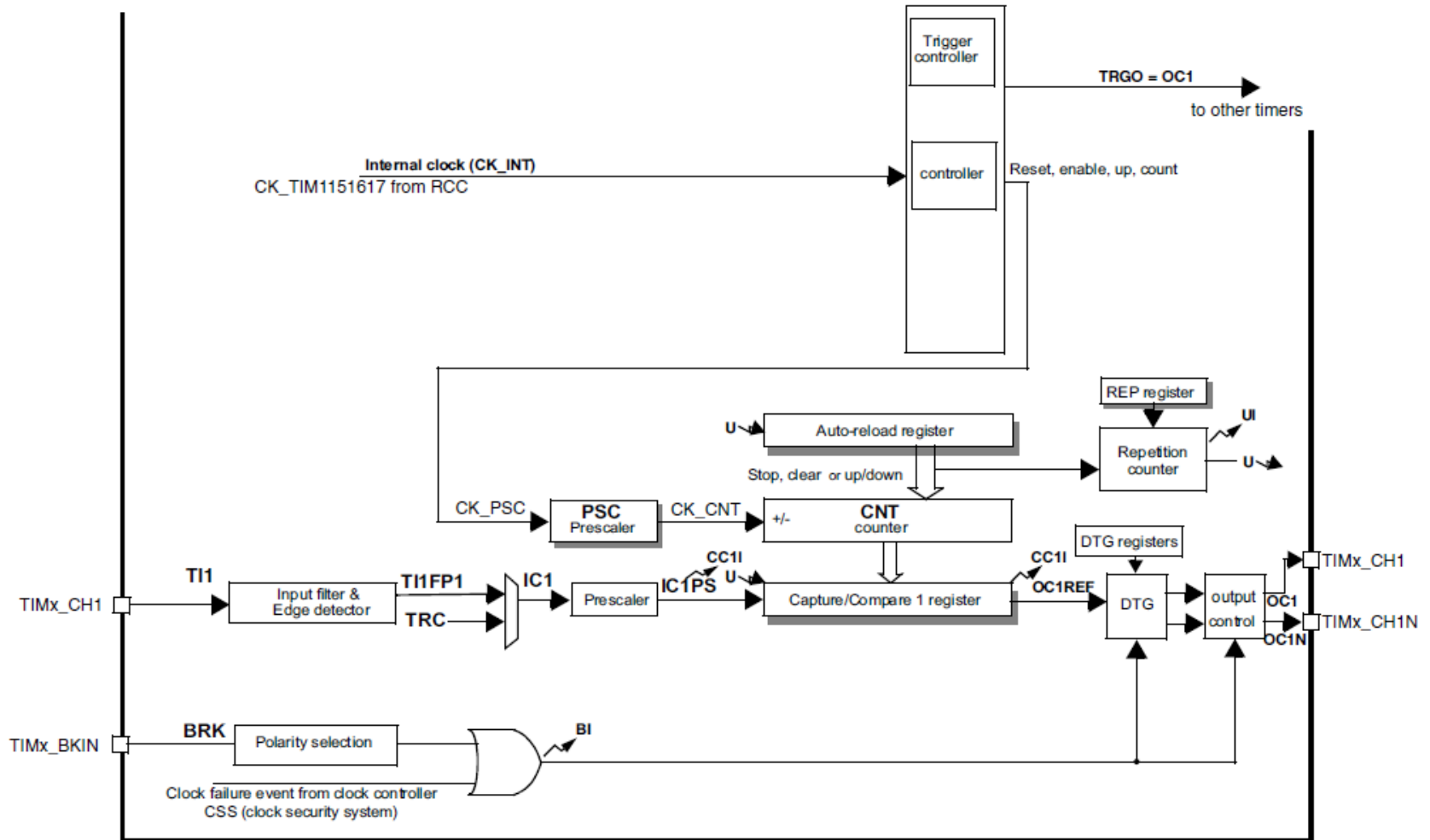
Tajmeri TIM15, TIM16, TIM17

- 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 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



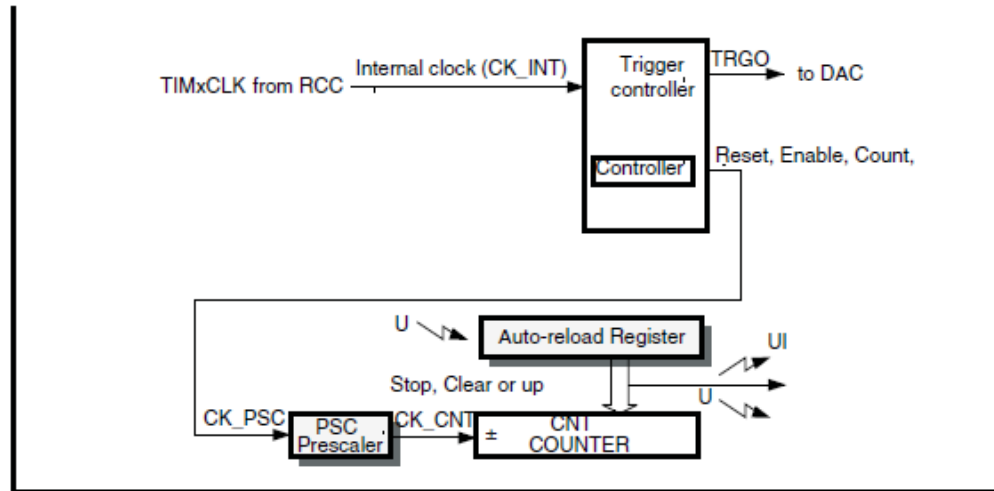
TIM16, TIM17



Osnovni tajmeri (Basic Timer)

TIM6 i TIM7

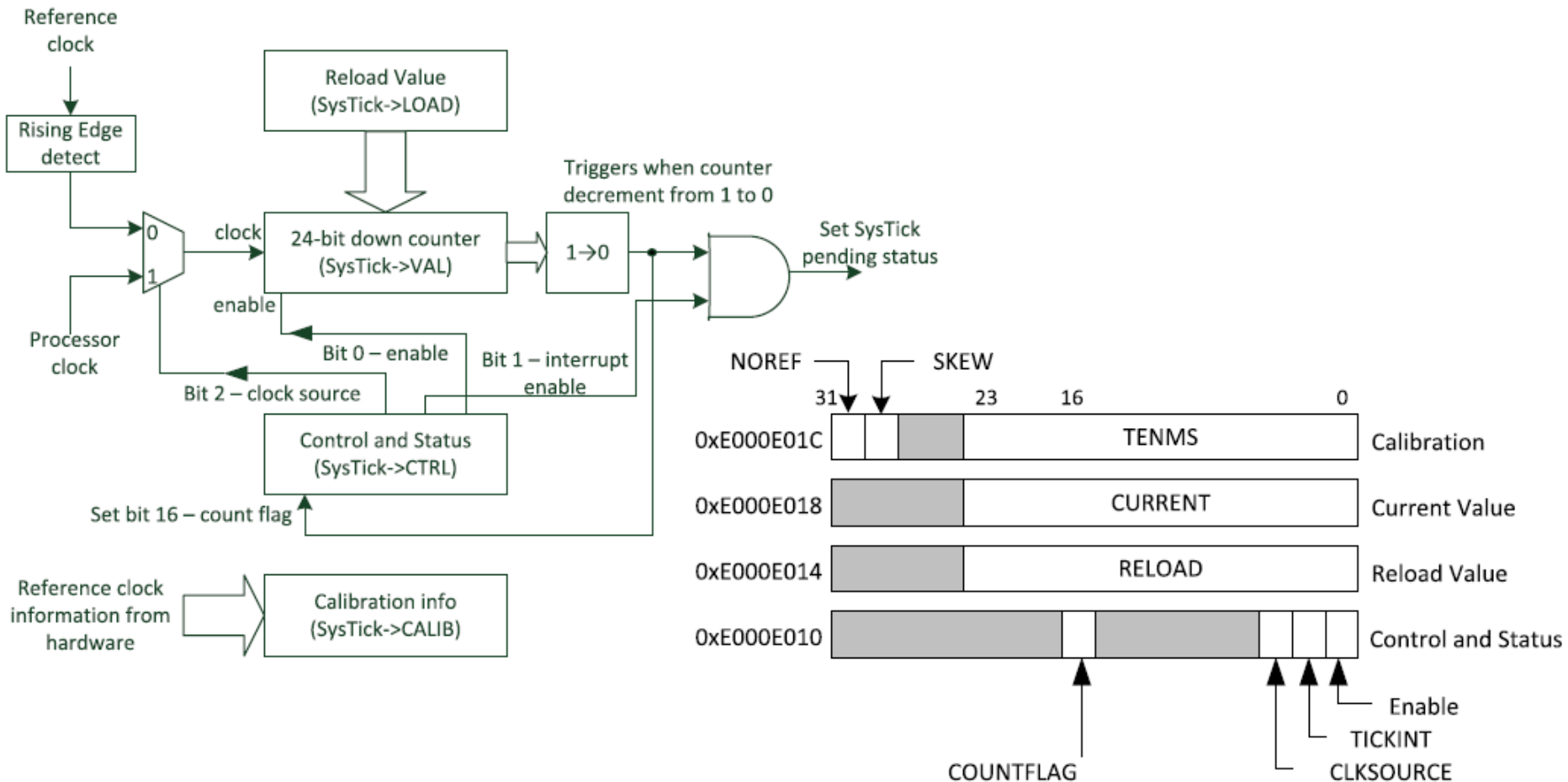
- 16-bitni brojač na gore.
- 16-bitni preskaler za ulazni takt.
- Mogućnost startovanja DAC-a.
- Prekid i DMA zahtev overflow događaj.
- Podrжан je DMA prenos.
- Uvek postoji jedan komplementarni izlaz.
- Brojač ponavljanja.



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.

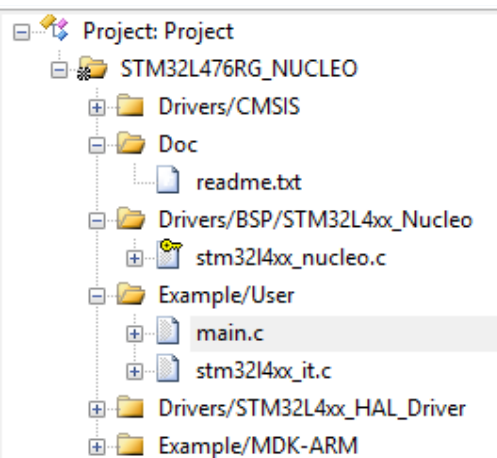
Systick timer



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_Handler function. The function is defined as 'void SysTick_Handler(void)' and contains a call to 'HAL_IncTick();'. This function is highlighted with a red box. A red dashed arrow points from a red text annotation to this function. The bottom window, titled 'stm32l4xx_hal.c', shows the implementation of the HAL_IncTick function. It is defined as '__weak void HAL_IncTick(void)' and contains the statement 'uwTick++;'. This function is highlighted with a blue box. A blue dashed arrow points from a blue text annotation to this function. The left sidebar shows a project tree with folders like 'Drivers/CMSIS', 'Doc', 'Drivers/BSP/STM32L4xx_Nucleo', 'Example/User', and 'Drivers/STM32L4xx_HAL_Driver'.

```
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 __weak void HAL_IncTick(void)
305 {
306     uwTick++;
307 }
308
```

Prekidna rutina SysTick_Handler() realizuje odbrojavanje vremena pozivom HAL funkcije

Jedna od osnovnih HAL drajver funkcija

Inicijalizacija SysTick tajmera

The screenshot displays the initialization of the SysTick timer in three files within an IDE. The left pane shows the project structure for STM32L476RG_NUCLEO, including folders for Drivers, Doc, Example/User, and Example/MDK-ARM, and files like main.c and stm32l4xx_it.c.

main.c (lines 74-78):

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

stm32l4xx_hal.c (lines 256-265):

```
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 (lines 267-272):

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

core_cm4.h (lines 1829-1846):

```
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 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

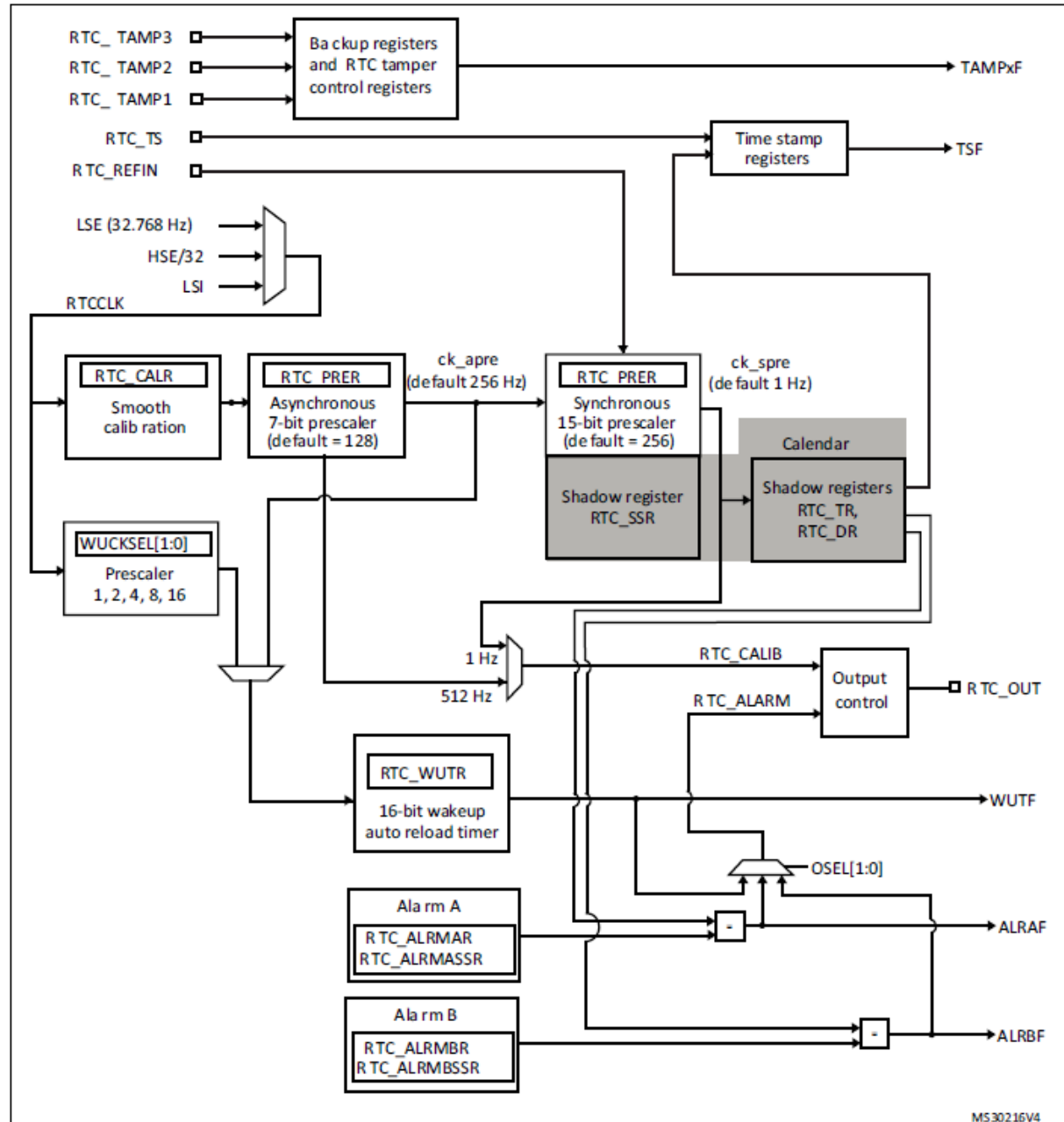
```
void start ()
    Start the timer.
void stop ()
    Stop the timer.
void reset ()
    Reset the timer to 0.
float read ()
    Get the time passed in seconds.
int read\_ms ()
    Get the time passed in milli-seconds.
int read\_us ()
    Get the time passed in micro-seconds.
```

Real Time Clock - RTC

- The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar with programmable alarm interrupts.
- Two 32-bit registers contain the seconds, minutes, hours (12- or 24-hour format), day (day of week), date (day of month), month, and year, expressed in binary coded decimal format (BCD). The sub-seconds value is also available in binary format.
- Compensations for 28-, 29- (leap year), 30-, and 31-day months are performed automatically. Daylight saving time compensation can also be performed.
- Additional 32-bit registers contain the programmable alarm subseconds, seconds, minutes, hours, day, and date.
- As long as the supply voltage remains in the operating range, the RTC never stops, regardless of the device status (Run mode, low-power mode or under reset).

RTC

- Two alarms
- Three tamper events
- 32 x 32-bit backup registers – The backup registers (RTC_BKPxR) are implemented in the RTC domain that remains powered-on by VBAT when the VDD power is switched off.



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

<u>time</u>	Get the current time
<u>set_time</u>	Set the current time
<u>mktime</u>	Converts a <code>tm</code> structure in to a timestamp
<u>localtime</u>	Converts a timestamp in to a <code>tm</code> structure
<u>ctime</u>	Converts a timestamp to a human-readable string
<u>strftime</u>	Converts a <code>tm</code> structure to a custom format human-readable string

RTC

- `#include "mbed.h"`
- `int main() {`
- `set_time(1493834400); // Set RTC time to Wed, 3 May 2017 18:00:00`
- `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);`
- `}`
- `}`