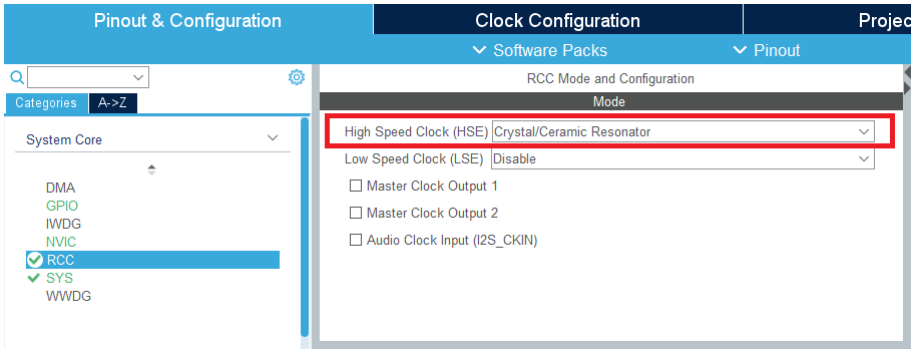
Vežbe 1

Primer 1: LED blinking i spoljašnji prekid

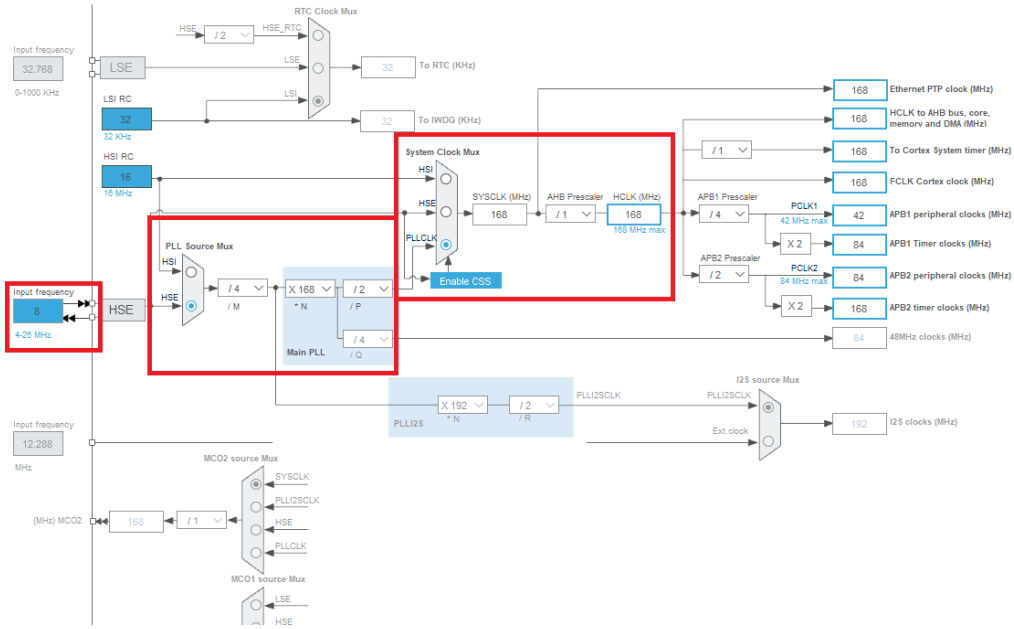
Konfiguracija sistema

- **Podešavanje oscilatora**
  Pod sekcijom *Pinout & Configuration - System Core - RCC* za *High Speed Clock (HSE)* odabrati *Crystal/Ceramic Resonator* (slika 1).



Slika 1: RCC podešavanja

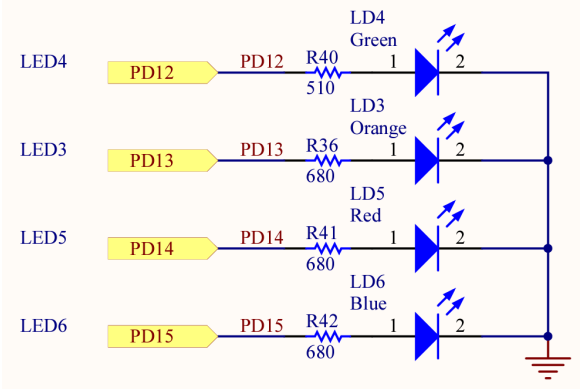
U *Clock Configuration* prozoru podesiti ulaznu frekvenciju (*Input frequency*) na 8 MHz sa eksternog oscilatora *HSE*. Na *PLL Source Mux* odabrati *HSE*, na *System Clock Mux* podesiti *PLLCLK*. Kao željenu frekvenciju u polje *HCLK* uneti 168 MHz (slika 2).



Slika 2: Clock Configuration podešavanja

- **LED diode STM32F407Discovery razvojne ploče**
  U *Pinout view* podesiti pinove PD12, PD13, PD14 i PD15, na koje su povezane odgovarajuće LED na STM32F407Discovery razvojnoj ploči, levim klikom na pin i odabirom *GPIO Output* funkcije (slika 3).

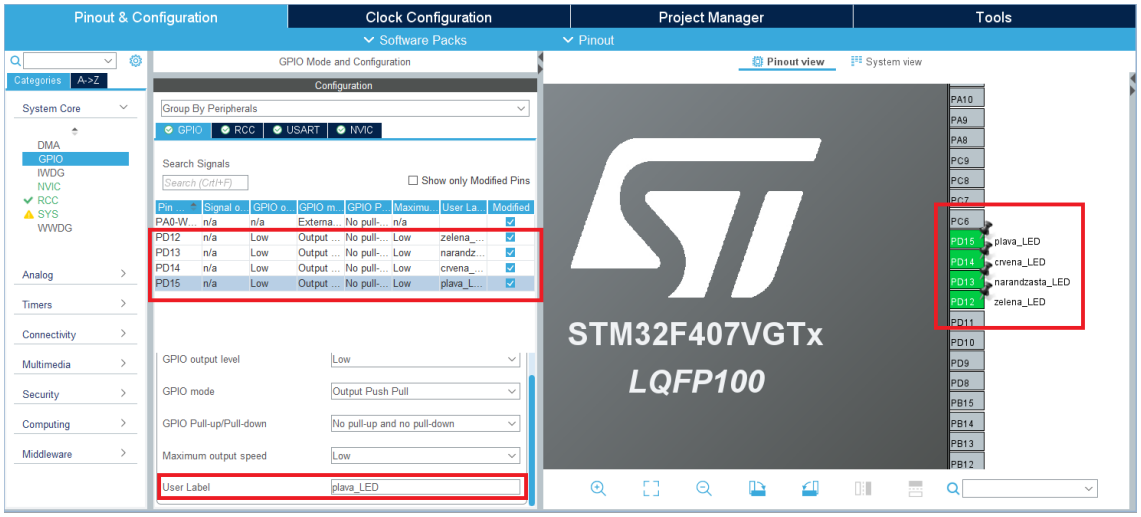
Slika 3: Pinout view - podešavanja funkcionalnosti pina



Slika 4: LED diode STM32F407Discovery razvojne ploče

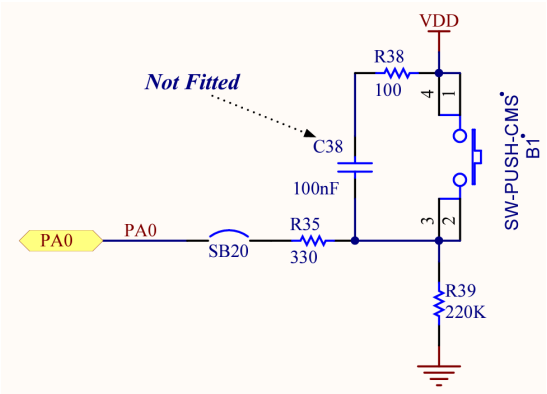
Dodatno, svakom od navedena četiri pina dodeliti naziv prema boji LED koja pripada datom pinu (slika 4) - u polju *System Core - GPIO - User Label* (slika 5):

- PD12 - zelena_LED
- PD13 - narandžasta_LED
- PD14 - crvena_LED
- PD15 - plava_LED

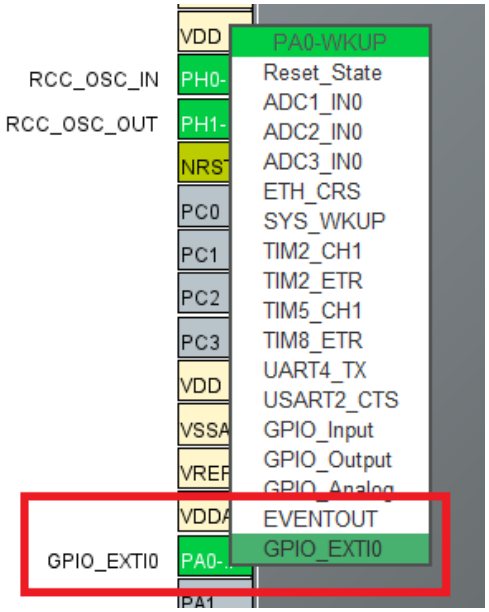


Slika 5: GPIO podešavanja

- **User taster STM32F407Discovery razvojne ploče**
  User taster STM32F407Discovery razvojne ploče povezan je na pin PA0 mikrokontrolera (slika 6). U cilju upotrebe spoljašnjih prekida sa ovog pina, PA0 podesiti u okviru *Pinout view* kao GPIO_EXTI0 (slika 7).

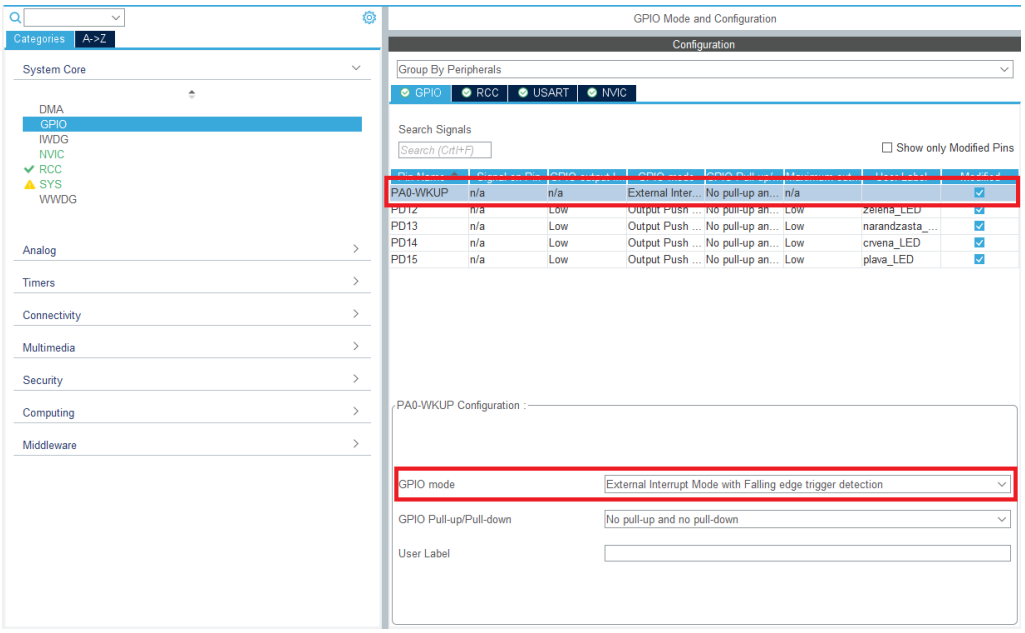


Slika 6: User taster STM32F407Discovery razvojne ploče



Slika 7: GPIO input podešavanja

Pod sekcijom *Pinout & Configuration - GPIO - PA0-WKUP Configuration* za *GPIO mode* PA0 podesiti *External Interrupt Mode with Falling edge trigger detection* (slika 8).



Slika 8: EXTI0 podešavanja

Kod

U okviru listinga 1 dat je kod kojim se realizuje uključivanje crvene, plave, zelene ili narandžaste LED u odnosu na pritisak User tastera. Svakim pritiskom tastera, sa pina PA0 generiše se eksterni interrupt, koji se obrađuje i izvršava se *HAL_GPIO_EXTI_Callback* funkcija, pri čemu se inkrementira *cnt* promenljiva. U odnosu na vrednost ostatka pri deljenju vrednosti *cnt* i broja 4, u okviru beskonačne petlje glavnog koda vrši se uključivanje jedne od četiri diode.

Listing 1: LED blinking i spoljašnji prekid

```

#include "main.h"

UART_HandleTypeDef huart2;

/* USER CODE BEGIN PV */
uint32_t cnt=0;
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();
  /* Configure the system clock */
  SystemClock_Config();
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART2_UART_Init();
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
          switch(cnt%4)
          {
                case 0:
                        HAL_GPIO_WritePin(zelena_LED_GPIO_Port, zelena_LED_Pin,
                        GPIO_PIN_SET);
                        HAL_GPIO_WritePin(GPIOD, crvena_LED_Pin|plava_LED_Pin|
                        narandzasta_LED_Pin, GPIO_PIN_RESET);
                        break;
                case 1:
                        HAL_GPIO_WritePin(plava_LED_GPIO_Port, plava_LED_Pin,
                        GPIO_PIN_SET);
                        HAL_GPIO_WritePin(GPIOD, crvena_LED_Pin|zelena_LED_Pin|
                        narandzasta_LED_Pin, GPIO_PIN_RESET);
                        break;
                case 2:
                        HAL_GPIO_WritePin(crvena_LED_GPIO_Port, crvena_LED_Pin,
                        GPIO_PIN_SET);
                        HAL_GPIO_WritePin(GPIOD, zelena_LED_Pin|plava_LED_Pin|
                        narandzasta_LED_Pin, GPIO_PIN_RESET);
```

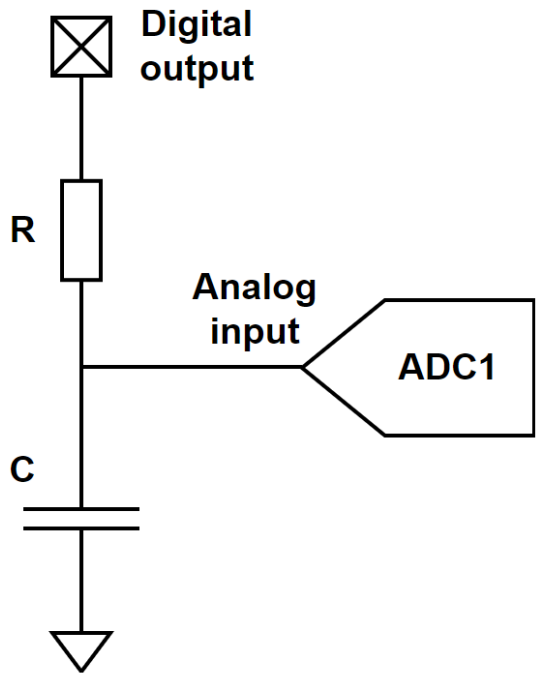
```
                        break;
                case 3:
                        HAL_GPIO_WritePin(narandzasta_LED_GPIO_Port,
                        narandzasta_LED_Pin, GPIO_PIN_SET);
                        HAL_GPIO_WritePin(GPIOD, crvena_LED_Pin|plava_LED_Pin|
                        zelena_LED_Pin, GPIO_PIN_RESET);
                        break;
            }
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = 4;
  RCC_OscInitStruct.PLL.PLLN = 168;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief USART2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART2_UART_Init(void)
{
  huart2.Instance = USART2;
  huart2.Init.BaudRate = 115200;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
```

```
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOD, zelena_LED_Pin|narandzasta_LED_Pin|
                    crvena_LED_Pin|plava_LED_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin : PA0 */
  GPIO_InitStruct.Pin = GPIO_PIN_0;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pins : zelena_LED_Pin narandzasta_LED_Pin crvena_LED_Pin plava_LED_Pi
  GPIO_InitStruct.Pin = zelena_LED_Pin|narandzasta_LED_Pin|
                        crvena_LED_Pin|plava_LED_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

  /* EXTI interrupt init*/
  HAL_NVIC_SetPriority(EXTI0_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(EXTI0_IRQn);

}

/* USER CODE BEGIN 4 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
        cnt++;
}
/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
```

```
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

Primer 2: Merenje kapacitivnosti kondenzatora

U ovom primeru potrebno je realizovati merenje kapacitivnosti kondenzatora na osnovu očitavanja napona sa kondenzatora pri njegovom punjenju, odnosno, pražnjenju preko otpornika otpornosti R. Ilustracija realizacije sistema prikazana je na slici 9. Pomoću GPIO pina mikrokontrolera, koji je konfigurisan kao digitalni izlaz, na RC član dovodi se napon napajanja (logička 1) ili masa (logička 0), čime se inicira punjenje, odnosno, pražnjenje kondenzatora, preko otpornika.



Slika 9: Kontrola punjenja i pražnjenja kondenzatora putem GPIO izlaza i očitavanje vrednosti napona sa A/D konvertora

Ukoliko je poznat vremenski interval punjenja kondenzatora, napona napajanja Vcc i otpornost R, moguće je odrediti kapacitivnost kondenzatora C. Naime, ukoliko je u trenutku T1 vrednost napona na kondenzatoru V_{C1}, jednačina punjenja kondenzatora na datu vrednost napona je (slika 10):

$$V_{C1} = Vcc \cdot (1 - e^{-\frac{T1}{RC}}) \tag{1}$$

Odakle je vreme T1:

$$\frac{V_{C1}}{Vcc} = 1 - e^{-\frac{T1}{RC}} \tag{2}$$

$$1 - \frac{V_{C1}}{Vcc} = e^{-\frac{T1}{RC}} \tag{3}$$

$$ln(1 - \frac{V_{C1}}{Vcc}) = -\frac{T1}{RC} \tag{4}$$

$$T1 = -RC \cdot ln(1 - \frac{V_{C1}}{Vcc}) \tag{5}$$

Ukoliko se punjenje kondenzatora nastavi i vrednost napona na kondenzatoru se očita u trenutku T2 (slika 10) jednačina punjenja kondenzatora glasi:

$$V_{C2} = Vcc \cdot (1 - e^{-\frac{T2}{RC}}) \tag{6}$$

Odakle je vreme T1:

$$T2 = -RC \cdot ln(1 - \frac{V_{C2}}{Vcc}) \tag{7}$$

Vremenski interval punjenja kondenzatora od vrednosti V_{C1} do vrednosti V_{C2} iznosi:

$$\Delta t = T2 - T1 \tag{8}$$

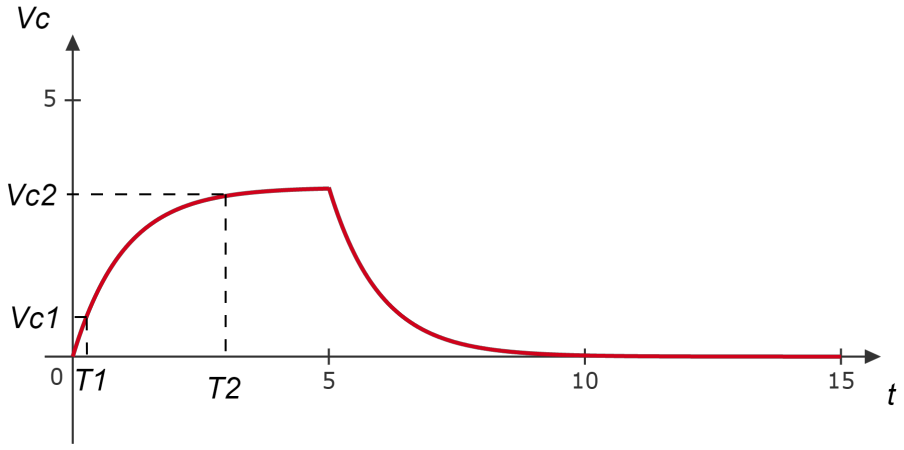
$$\Delta t = -RC(ln(1 - \frac{V_{C2}}{Vcc}) - ln(1 - \frac{V_{C1}}{Vcc})) \tag{9}$$

$$\Delta t = -RC(ln(\frac{\frac{Vcc - V_{C2}}{Vcc}}{\frac{Vcc - V_{C1}}{Vcc}}) \tag{10}$$

$$\Delta t = -RC \cdot ln(\frac{Vcc - V_{C2}}{Vcc - V_{C1}}) \tag{11}$$

Iz jednačine (11) moguće je izraziti kapacitivnost kondenzatora C kao:

$$C = \frac{-\Delta t}{R \cdot ln(\frac{Vcc - V_{C2}}{Vcc - V_{C1}})} \tag{12}$$



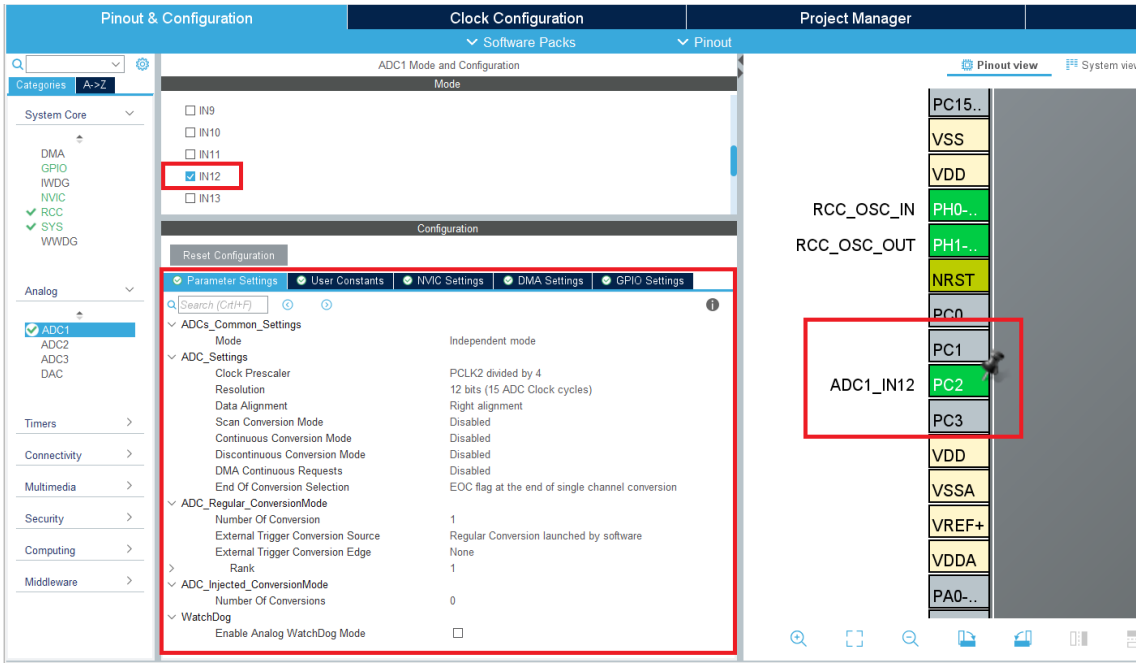
Slika 10: Promena napona na kondenzatoru pri punjenju i pražnjenju

U datom zadatku kao digitalni izlazni pin uzima se pin PE13, dok se kao analogni ulazni pin uzima pin PC2. Dakle, postavljanjem logičke nule na pin PE13 ostvaruje se pražnjenje kondenzatora. Za pražnjenje kondenzatora pravi se pauza od 10 s, a zatim se očitava vrednost na izlazu A/D konvertora koja odgovara naponu na kondenzatoru u datom trenutku T1. Zatim se na pin PE13 postavlja stanje logičke 1, čime se na otpornik dovodi napon napajanja Vcc i postiže se punjenje kondenzatora, za koje se pravi pauza od 5 s. Nakon isteka 5 s očitava se vrednost na izlazu A/D konvertora, koja odgovara naponu na kondenzatoru u trenutku T2. Na osnovu datih vrednosti vrši se proračun kapacitivnosti kondenzatora prema jednačini (12).
Opisani postupak merenja kapacitivnosti kondenzatora implementiran je u okviru koda datog u listingu 2.

Konfiguracija sistema

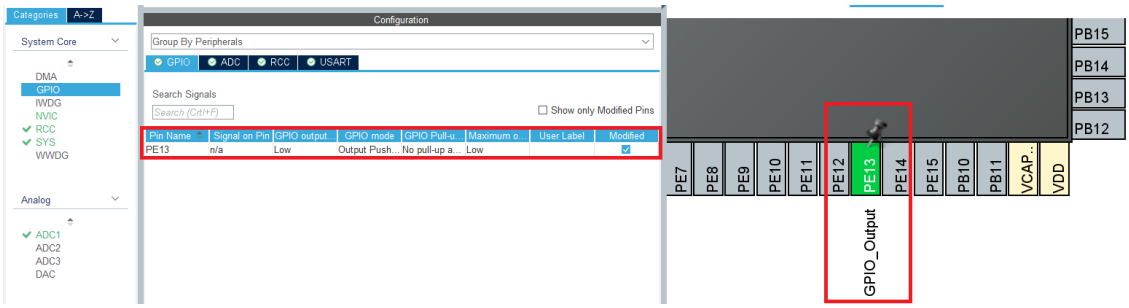
- **Podešavanje oscilatora**
  Oscilator i frekvenciju takta sistema podesiti kao u prethodnom primeru.

- **A/D konvertor**
  U polju *Analog* uključiti ADC1 modul (slika 11). Za ulazni analogni pin odabrati IN12, koji odgovara PC2 pinu. PC2 pin predstavlja AN pin mikroBUS socket-a 2, na koji će biti povezana RC komponenta. Jedan kraj (-) kondenzatora povezati na GND mikroBUS socket-a 2, a drugi kraj u spoju sa otpornikom od $100k\Omega$ povezati na AN pin mikroBUS socket-a 2. Drugi kraj otpornika povezati na PWM pin mikroBUS socket-a 2, odnosno, na PE13 pin mikrokontrolera.



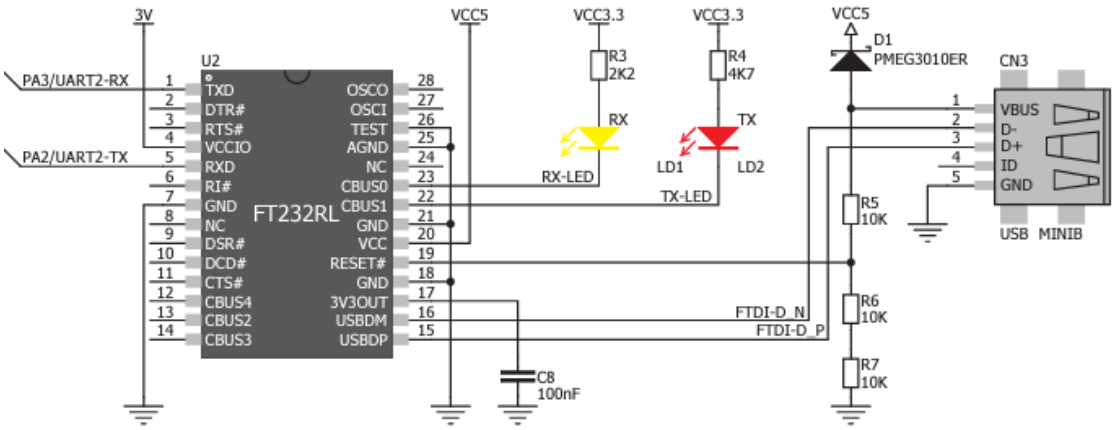
Slika 11: ADC1 podešavanja

- **GPIO output**
  Pin PE13 postaviti kao GPIO izlaz. Dovođenjem stanja logičke 1 vrši se punjenje kondenzatora preko otpornika na vrednost Vcc, dok se dovođenjem logičke 0 na PE13 kondenzator prazni (slika 13).



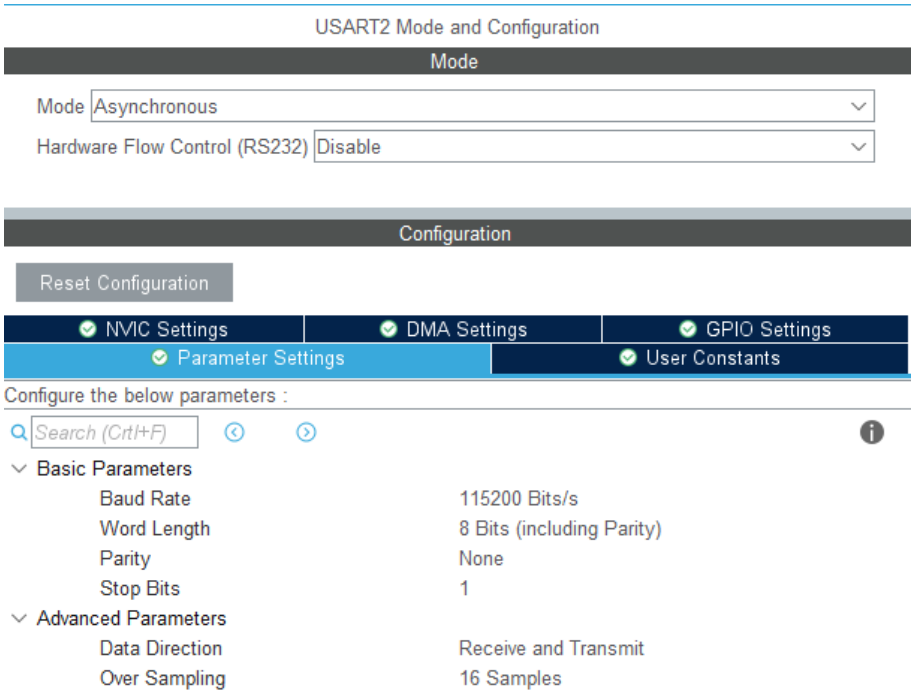
Slika 12: PE13 - GPIO output

- **Podešavanja USART2 modula**
  Uključiti USART2 modul (pinovi PA2 - Tx, PA3 - Rx) za USB-UART komunikaciju sa računarom. Pinovi PA2 i PA3 povezani su na Rx i Tx pinove FT232RL čipa na STM32F4 Discovery Shield ploči (slika 13), čime je omogućena USB-UART konverzija i komunikacija računara i STM32F407VGT mikrokontrolera.



Slika 13: Deo šeme STM32F4 Discovery Shield ploče za USB-UART komunikaciju

Podesiti asinhroni mod rada, brzinu prenosa podataka 115 200 bps, 8 bitova podataka, bez bita pariteta i 1 stop bit (slika 14).



Slika 14: USART2 podešavanja

Kod

Listing 2: Merenje kapacitivnosti kondenzatora

```
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include <string.h>
#include <stdio.h>
#include <math.h>
/* USER CODE END Includes */

/* Private variables ---------------------------------------------------------*/
ADC_HandleTypeDef hadc1;
UART_HandleTypeDef huart2;

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_ADC1_Init(void);
static void MX_USART2_UART_Init(void);

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */
        HAL_StatusTypeDef status;
        uint16_t adc_raw;
        float voltage1,voltage2;
        char adc_string[40];
        float c;
        const uint32_t R=100000;
        const int8_t t=5;
        const float vcc=3300;
  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();
  /* Configure the system clock */
  SystemClock_Config();
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_ADC1_Init();
  MX_USART2_UART_Init();
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    HAL_GPIO_WritePin(GPIOE, GPIO_PIN_13, GPIO_PIN_RESET);
    HAL_Delay(10000);
    HAL_ADC_Start(&hadc1);
    status = HAL_ADC_PollForConversion(&hadc1, 1000);

    if(status == HAL_OK)
    {
      adc_raw = HAL_ADC_GetValue(&hadc1);
      voltage1 = ((float)adc_raw)*3300.0/4096;
```

```
    }
    HAL_GPIO_WritePin(GPIOE, GPIO_PIN_13, GPIO_PIN_SET);
    HAL_Delay(5000);
    HAL_ADC_Start(&hadc1);
    status = HAL_ADC_PollForConversion(&hadc1, 1000);

    if(status == HAL_OK)
    {
      adc_raw = HAL_ADC_GetValue(&hadc1);
      voltage2 = ((float)adc_raw)*3300.0/4096;

      c = -t/(R*log((vcc-voltage2)/(vcc-voltage1)))*1000000;
      sprintf(adc_string, "Kapacitivnost je: %.2f uF\n", c);
      HAL_UART_Transmit(&huart2, (uint8_t *)adc_string, strlen(adc_string),
                        HAL_MAX_DELAY);
    }
    else
    {
      sprintf(adc_string, "Greska\n");
      HAL_UART_Transmit(&huart2, (uint8_t *)adc_string, strlen(adc_string),
                        HAL_MAX_DELAY);
    }
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = 4;
  RCC_OscInitStruct.PLL.PLLN = 168;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5) != HAL_OK)
  {
    Error_Handler();
  }
```

```
}

/**
  * @brief ADC1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_ADC1_Init(void)
{
  ADC_ChannelConfTypeDef sConfig = {0};

  /* USER CODE BEGIN ADC1_Init 1 */

  /* USER CODE END ADC1_Init 1 */
  /** Configure the global features of the ADC (Clock, Resolution, Data Alignment and nu
  */
  hadc1.Instance = ADC1;
  hadc1.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV4;
  hadc1.Init.Resolution = ADC_RESOLUTION_12B;
  hadc1.Init.ScanConvMode = DISABLE;
  hadc1.Init.ContinuousConvMode = DISABLE;
  hadc1.Init.DiscontinuousConvMode = DISABLE;
  hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
  hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
  hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc1.Init.NbrOfConversion = 1;
  hadc1.Init.DMAContinuousRequests = DISABLE;
  hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
  if (HAL_ADC_Init(&hadc1) != HAL_OK)
  {
    Error_Handler();
  }
  /** Configure for the selected ADC regular channel its corresponding rank in the seque
  */
  sConfig.Channel = ADC_CHANNEL_12;
  sConfig.Rank = 1;
  sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief USART2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART2_UART_Init(void)
{
  huart2.Instance = USART2;
  huart2.Init.BaudRate = 115200;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief GPIO Initialization Function
  * @param None
```

```
198   * @retval None
199   */
200 static void MX_GPIO_Init(void)
201 {
202   GPIO_InitTypeDef GPIO_InitStruct = {0};
203
204   /* GPIO Ports Clock Enable */
205   __HAL_RCC_GPIOH_CLK_ENABLE();
206   __HAL_RCC_GPIOC_CLK_ENABLE();
207   __HAL_RCC_GPIOA_CLK_ENABLE();
208   __HAL_RCC_GPIOE_CLK_ENABLE();
209
210   /*Configure GPIO pin Output Level */
211   HAL_GPIO_WritePin(GPIOE, GPIO_PIN_13, GPIO_PIN_RESET);
212
213   /*Configure GPIO pin : PE13 */
214   GPIO_InitStruct.Pin = GPIO_PIN_13;
215   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
216   GPIO_InitStruct.Pull = GPIO_NOPULL;
217   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
218   HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);
219
220 }
221 /**
222   * @brief  This function is executed in case of error occurrence.
223   * @retval None
224   */
225 void Error_Handler(void)
226 {
227   /* USER CODE BEGIN Error_Handler_Debug */
228   /* User can add his own implementation to report the HAL error return state */
229   __disable_irq();
230   while (1)
231   {
232   }
233   /* USER CODE END Error_Handler_Debug */
234 }
235
236 #ifdef  USE_FULL_ASSERT
237 /**
238   * @brief  Reports the name of the source file and the source line number
239   *         where the assert_param error has occurred.
240   * @param  file: pointer to the source file name
241   * @param  line: assert_param error line source number
242   * @retval None
243   */
244 void assert_failed(uint8_t *file, uint32_t line)
245 {
246   /* USER CODE BEGIN 6 */
247   /* User can add his own implementation to report the file name and line number,
248      ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
249   /* USER CODE END 6 */
250 }
251 #endif /* USE_FULL_ASSERT */
```

.....