

Компоненте и кола снаге

ЛАБОРАТОРИЈСКЕ ВЕЖБЕ, 2022/23.

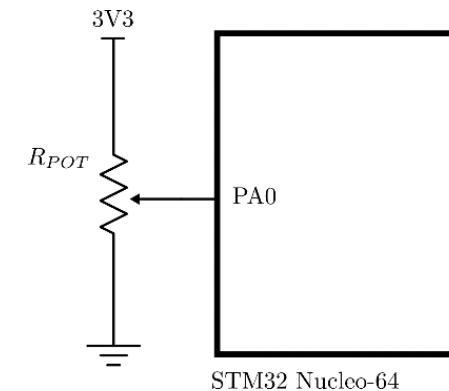
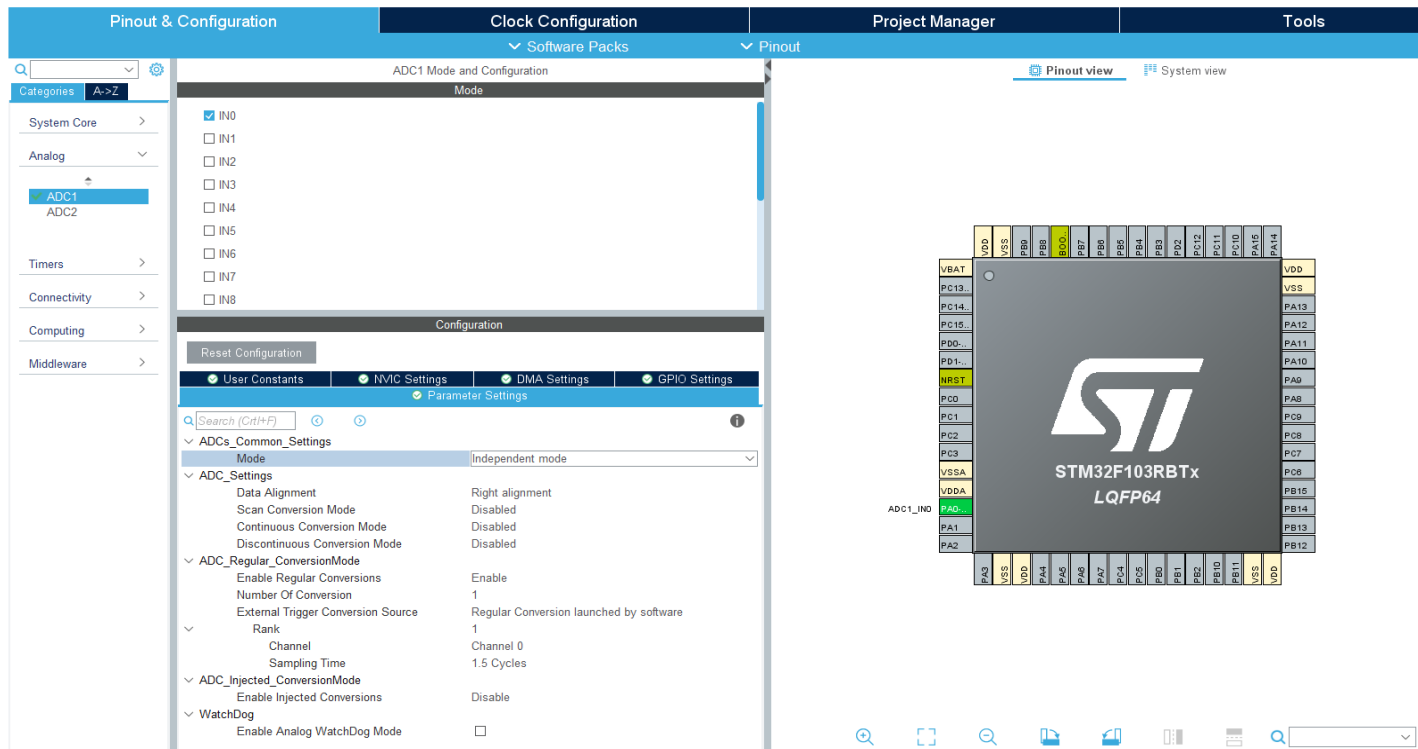
МИЛОШ МАРЈАНОВИЋ

МИЛАН СТОЈАНОВИЋ

ТРЕЋИ ТЕРМИН

A/D конверзија (*Single Conversion mode*)

- STM32F103RB има 2 A/D конвертора, у временском мултиплексу могуће је очитати вредности са највише 16 пинова.
- Резолуција A/D конвертора није подесива код овог микроконтролера, износи 12 битова.

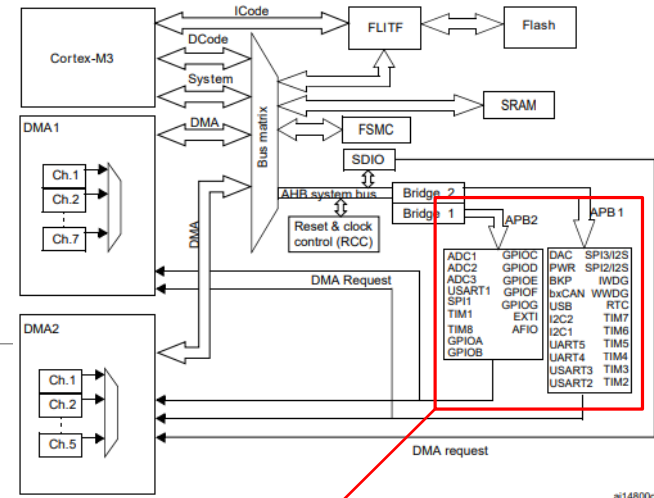


ADC *Single Conversion*

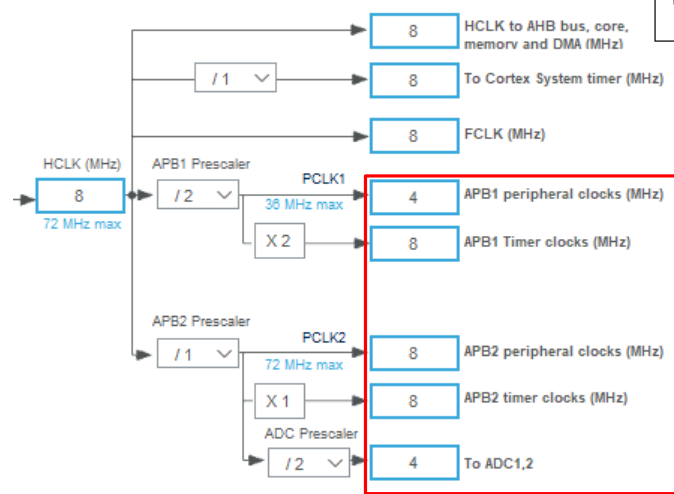
```
/* USER CODE BEGIN Includes */
#include "string.h"
/* USER CODE END Includes */
***
/* Private user code -----*/
/* USER CODE BEGIN 0 */
uint32_t adcVal0;
char bafer[20];
/* USER CODE END 0 */
***
HAL_ADC_Start(&hadc1);
if(HAL_ADC_PollForConversion(&hadc1,5) == HAL_OK)
{
    adcVal0 = HAL_ADC_GetValue(&hadc1);
    sprintf(bafer, " %d \r\n",adcVal0);
    HAL_UART_Transmit(&huart2, (uint8_t*)bafer, strlen(bafer), 10);
}
HAL_Delay(50);
```

ADC Continuous mode

Користи се када је потребно вршити аквизицију података најбрже могуће.



The screenshot shows the STM32CubeMX configuration tool. The 'ADC1 Mode and Configuration' window is open, showing the 'Mode' and 'Configuration' sections. The 'Continuous Conversion Mode' is set to 'Enabled', and the 'Sampling Time' is set to '239.5 Cycles'. The 'Parameter Settings' section is also visible, showing various ADC settings.



$$T_{conv}(\text{cycles}) = \text{Sampling time} + 12.5$$

$$T_{conv}(\text{cycles}) = 239.5 + 12.5 = 252$$

$$f_{adc} = 4\text{MHz}$$

$$T_{conv}(s) = \frac{1}{4 \cdot 10^6} \cdot 252 = 63\mu\text{s}$$

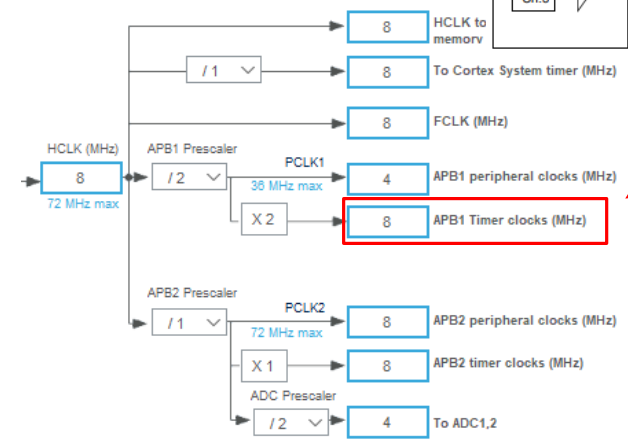
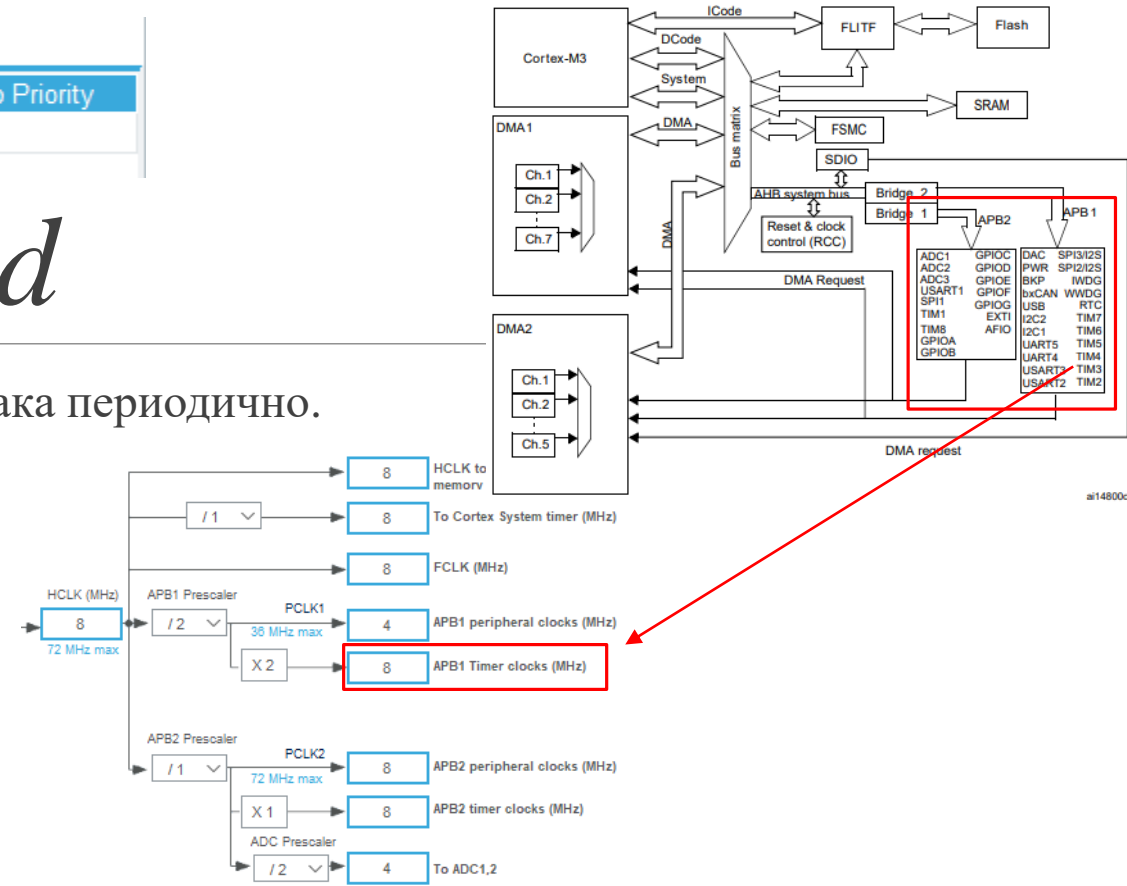
ADC *Continuous mode*

```
/* USER CODE BEGIN 0 */
uint32_t adcVrednost;
char bafer[20];
/* USER CODE END 0 */
***
/* USER CODE BEGIN 2 */
HAL_ADC_Start(&hadc1); //једном се покрене, A/D конверзија се врши континуално!
/* USER CODE END 2 */
***
adcVrednost=HAL_ADC_GetValue(&hadc1);
sprintf(bafer,"Pot = %d \r\n",adcVrednost);
HAL_UART_Transmit(&huart2, (uint8_t*)bafer,strlen(bafer),10);
```

Parameter Settings	User Constants	NVIC Settings	DMA Settings	
NVIC Interrupt Table		Enabled	Preemption Priority	Sub Priority
TIM3 global interrupt		<input checked="" type="checkbox"/>	0	0

ADC TIMER triggered

Користи се када је потребно вршити аквизицију података периодично.



$$f_{TIM3} = 8MHz$$

$$f_{TIM3(prescaler)} = \frac{f_{TIM3}}{Prescaler} = \frac{8MHz}{8000 - 1} = 1kHz$$

$$t_{ISR} = \frac{Counter\ Period}{f_{TIM3(prescaler)}} = \frac{1000 - 1}{1kHz} = 1s$$

User Constants	NVIC Settings	DMA Settings	GPIO Settings
Parameter Settings			
NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
ADC1 and ADC2 global interrupts	<input checked="" type="checkbox"/>	0	0

ADC TIMER *triggered*

The screenshot shows the configuration for ADC1. The 'External Trigger Conversion Source' is set to 'Timer 3 Trigger Out event', which is highlighted with a red box. Other settings include: Mode: Independent mode; Data Alignment: Right alignment; Scan Conversion Mode: Disabled; Continuous Conversion Mode: Disabled; Discontinuous Conversion Mode: Disabled; Enable Regular Conversions: Enable; Number Of Conversion: 1; Rank: 1; Channel: Channel 0; Sampling Time: 1.5 Cycles; Enable Injected Conversions: Disable; Enable Analog WatchDog Mode: .

$$T_{conv}(\text{cycles}) = \text{Sampling time} + 12.5$$

$$T_{conv}(\text{cycles}) = 1.5 + 12.5 = 14$$

$$f_{adc} = 4\text{MHz}$$

$$T_{conv}(s) = \frac{1}{4 \cdot 10^6} \cdot 14 = 3.5\mu\text{s}$$

Сваке секунде TIM3 генерише *interrupt*, који стартује ADC, када је конверзија завршена – генерише се *interrupt*, тада преузимамо податке.

ADC TIMER *triggered*

```
/* USER CODE BEGIN Includes */
#include "string.h"
/* USER CODE END Includes */
/* USER CODE BEGIN 0 */
uint32_t adcVrednost;
char bafer[20];
/* USER CODE END 0 */
/* USER CODE BEGIN 2 */
    HAL_TIM_Base_Start(&htim3);           //стартовање тајмера
    HAL_ADC_Start_IT(&hadc1);             //стартовање A/D конвертора са interrupt-ом
/* USER CODE END 2 */
/* USER CODE BEGIN 4 */
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
    /* Prevent unused argument(s) compilation warning */
    UNUSED(hadc);
    /* NOTE : This function should not be modified. When the callback is needed,
       function HAL_ADC_ConvCpltCallback must be implemented in the user file.
    */

    adcVrednost=HAL_ADC_GetValue(&hadc1);
    sprintf(bafer,"Pot = %d \r\n",adcVrednost);
    HAL_UART_Transmit(&huart2,(uint8_t*)bafer,strlen(bafer),10);
}

```

