Konfiguracija takta

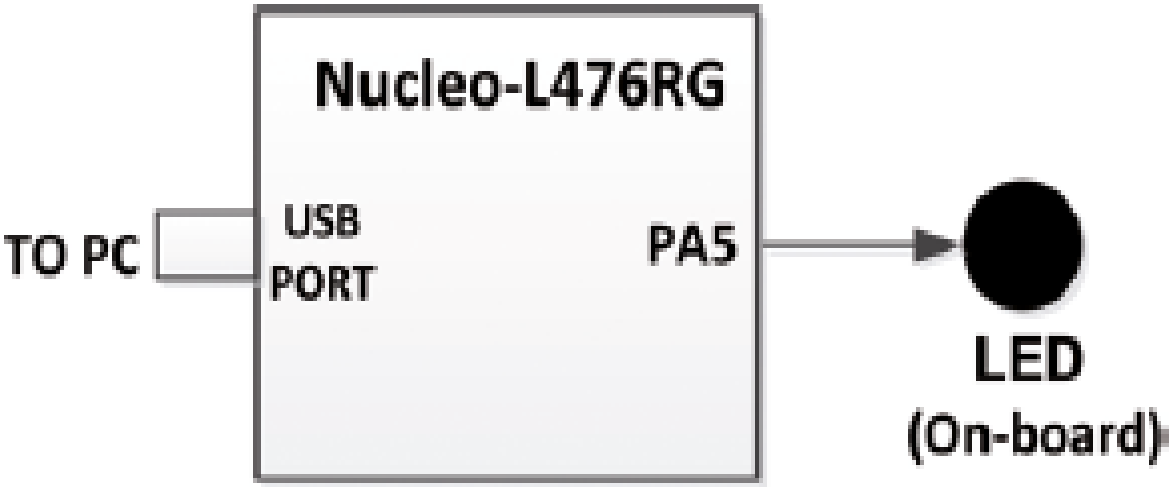


Figure 5.1: Block diagram of the project.

Konfiguracija takta

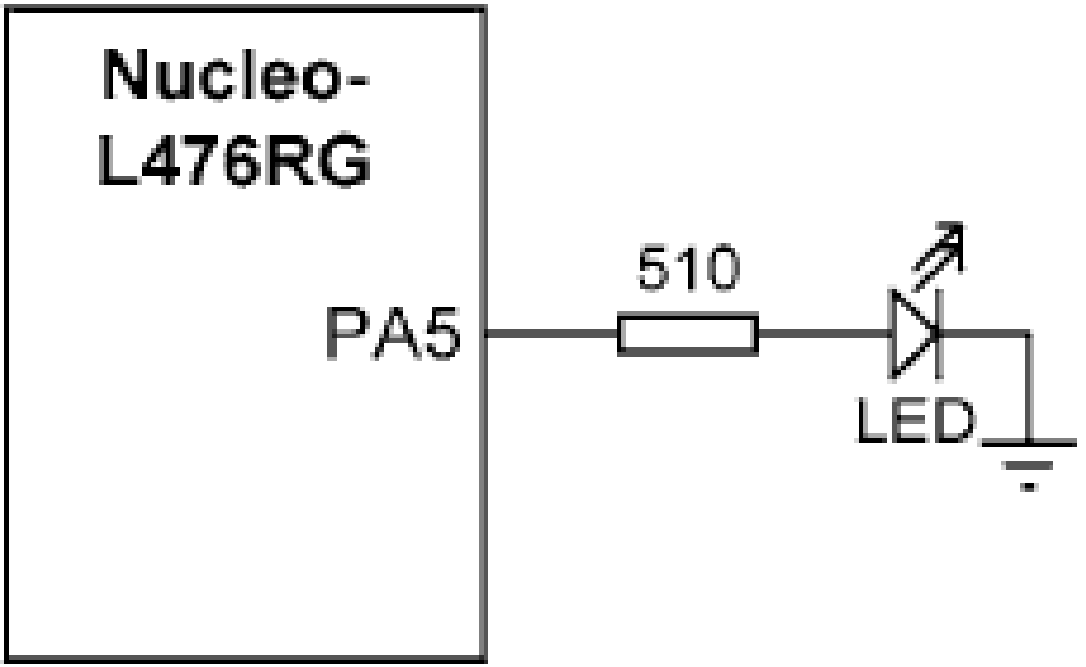


Figure 5.2: Circuit diagram of the project.

Konfiguracija takta

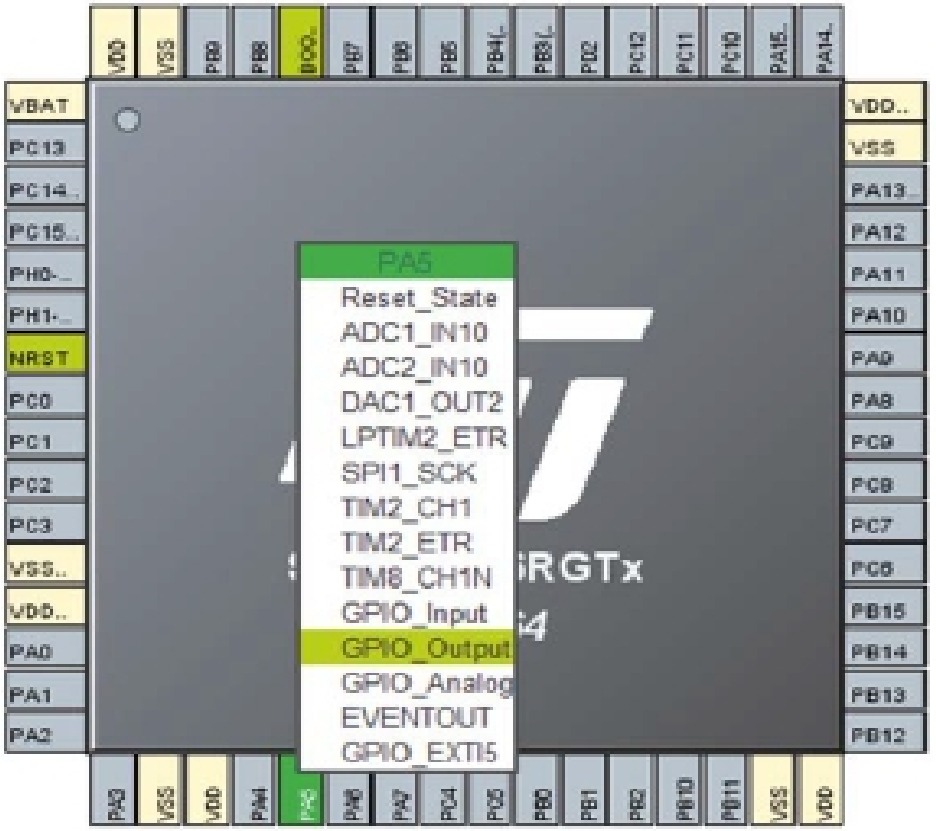


Figure 5.8: Configuring PA5 as an output.

Konfiguracija takta

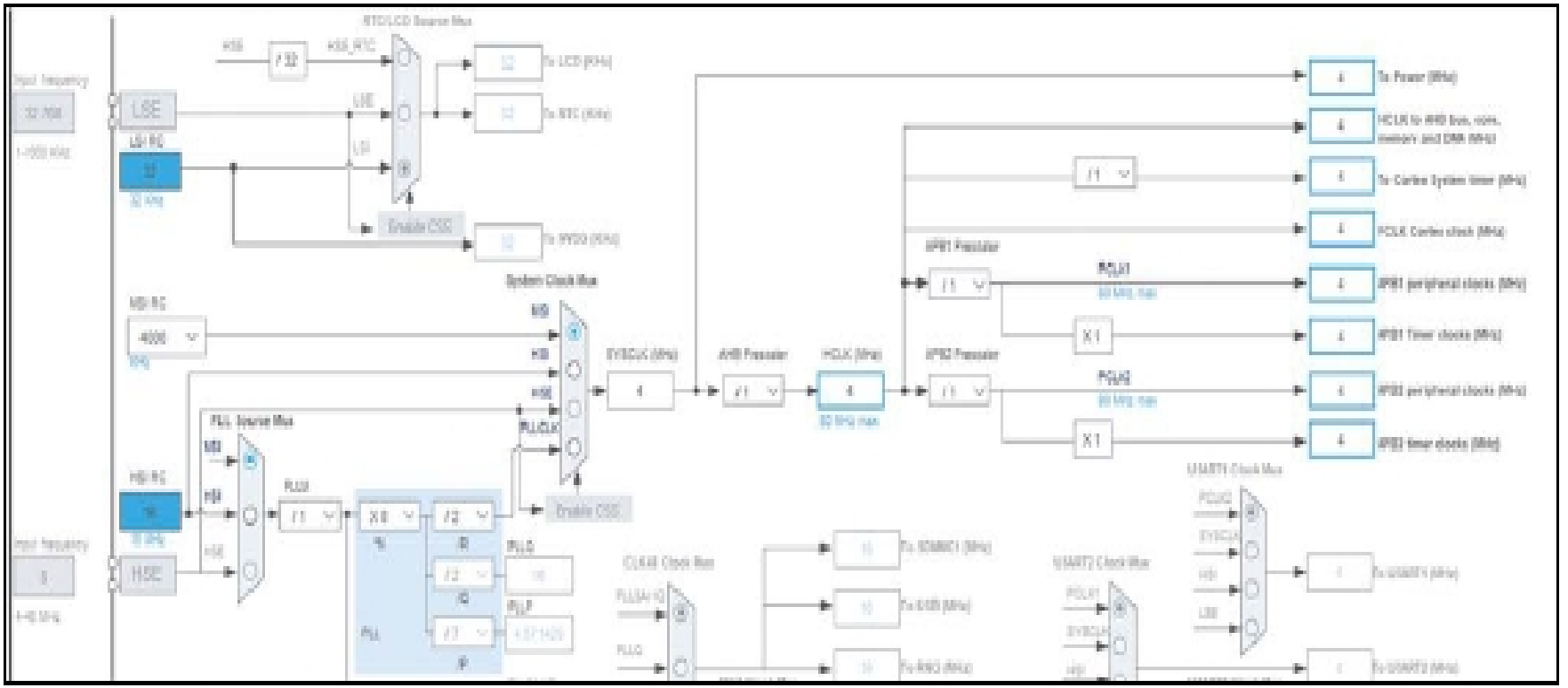


Figure 5.9: Clock configuration screen (part of the screen is shown).

Konfiguracija takta

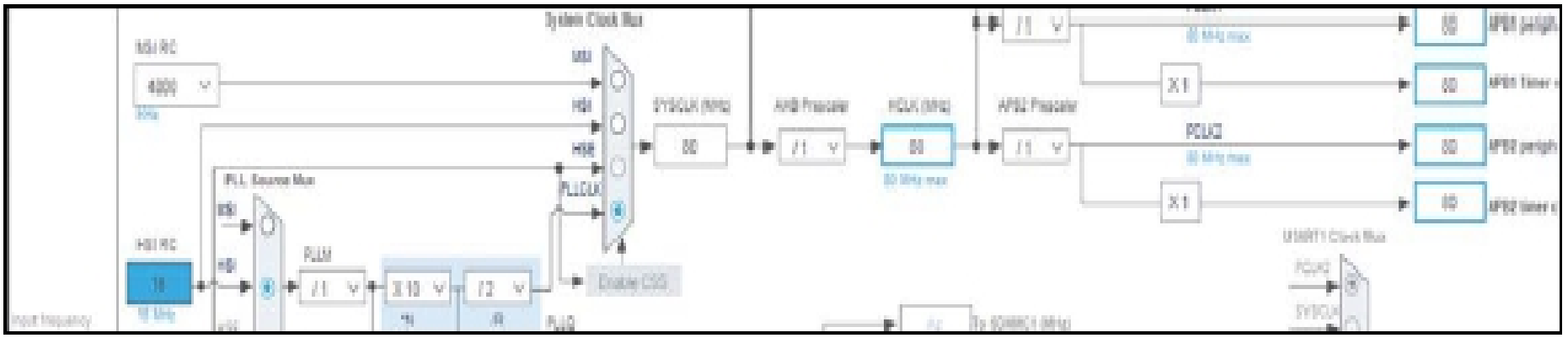


Figure 5.10: Configuring 80 MHz clock.

Podesiti takt za max frekvenciju stm32f103!

Blinkanje na pritisak tastera EXTI

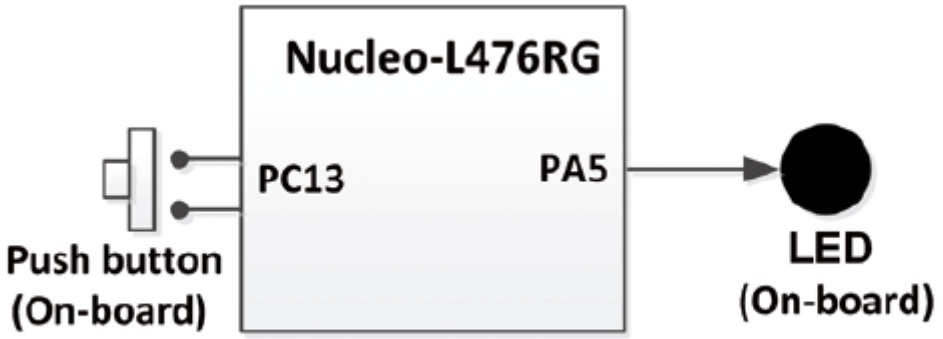


Figure 5.34: Block diagram of the project.

Blinkanje na pritisak tastera EXTI

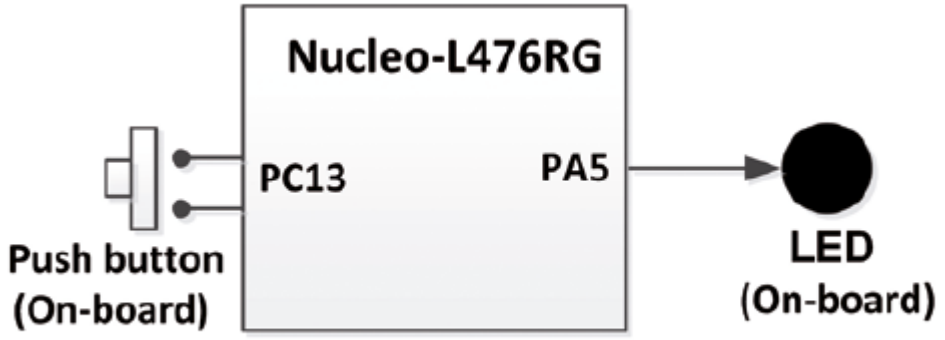


Figure 5.34: Block diagram of the project.

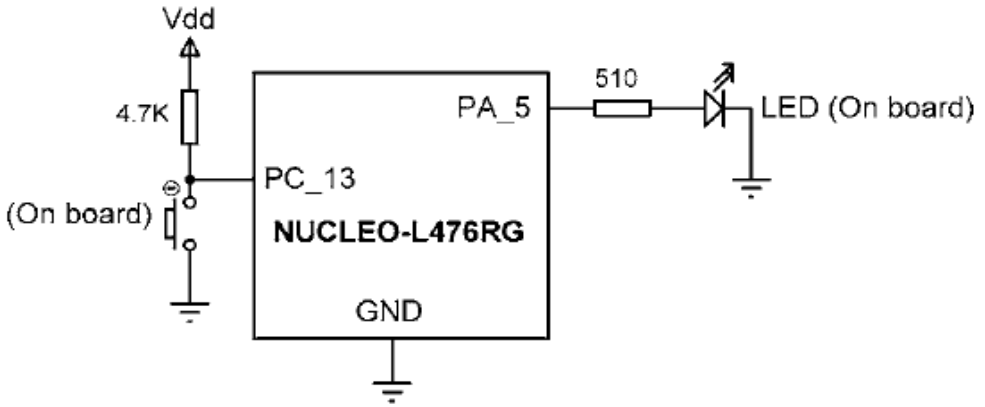


Figure 5.35: Circuit diagram of the project.

Blinkanje na pritisak tastera EXTI

- U zavisnosti od vrste kontrolera, postoji vise eksternih linija prekida, na primer 16, sa 16 multipleksera i posebnim adresama vektora prekida, koje su povezane sa GPIO pinovima.
- Ove linije su imenovane kao EXTI0, EXTI1, itd., do EXTI15. GPIO pinovi sa istim redosledom su grupisani zajedno i povezani na jednu EXTI liniju.
- Na primer, EXTI2 je povezan sa PA2, PB2, PC2 i tako dalje.
- Zbog toga, u bilo kom trenutku možemo imati eksterni prekid samo na jednom od povezanih GPIO pinova tog EXTI multipleksera

Blinkanje na pritisak tastera EXTI

- Na primer, kada zelimo da koristimo EXTI3,Možemo koristiti PA3, PB3, PC3, itd., ali ne i PA2, PB2, itd. istovremeno.
- Tako se ceo GPIO port ili pinovi sa različitih GPIO portova mogu konfigurisati kao eksterni prekidi.
- Takođe možemo odlučiti kada će se eksterni prekid detektovati — na uzlaznim ili silaznim ivicama, ili na obe.
- EXTI linije od 0 do 4 su pojedinačno i direktno povezane sa NVIC interfejsom, dok su preostale višeg reda EXTI linije grupisane u dve grupe — jedna obuhvata EXTI5 do EXTI9, a druga EXTI10 do EXTI15.
- Zbog toga gornji EXTI pinovi nisu pojedinačno i direktno povezani sa NVIC-om.
- Zbog ovoga, EXTI0–EXTI4 imaju zasebne jedinstvene adrese vektora prekida, dok ostatak koristi dve odvojene adrese vektora prekida.

Blinkanje na pritisak tastera EXTI

Sledeći koraci su potrebni da bi se MCU konfigurisao za prihvatanje prekida na port pinu PC13:

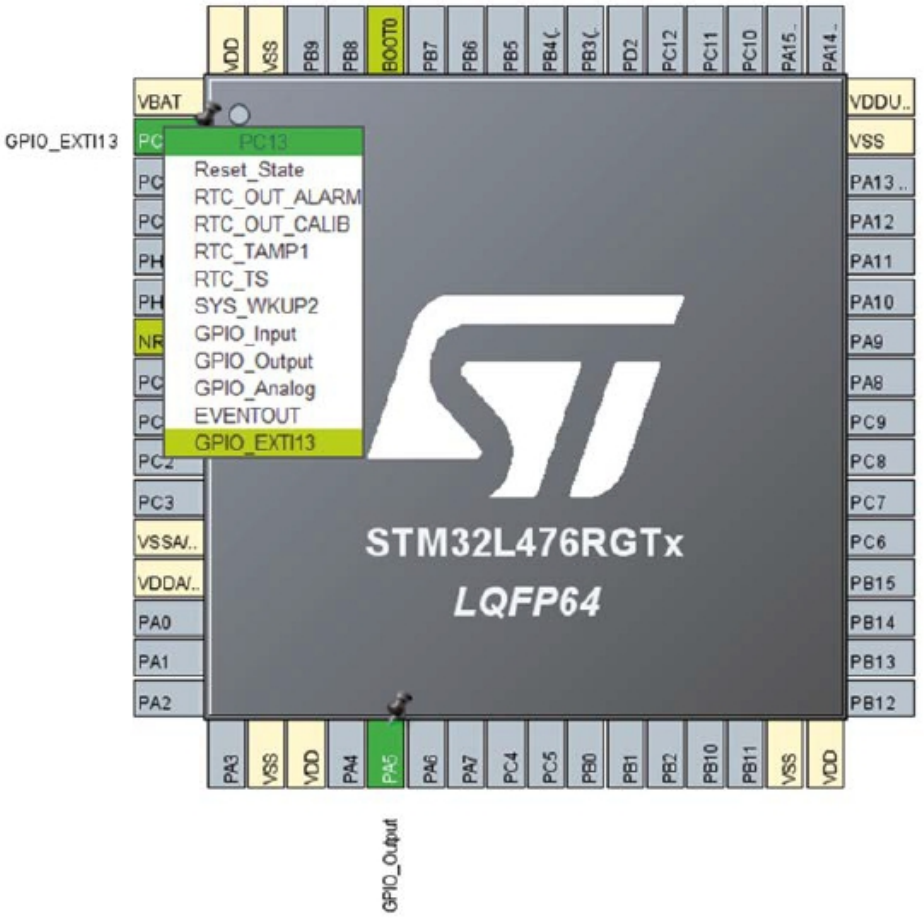
- Konfigurisati GPIO mod da prihvata prekide na uzlaznim (od niskog ka visokom) ili silaznim (od visokog ka niskom) ivicama;
- Omogućiti prekid EXTI linije koja je povezana sa pinom PC13, odnosno **EXTI15_10_IRQn**;
- Definisati funkciju **void EXTI15_10_IRQHandler(void)**, koja predstavlja ISR rutinu povezanu sa IRQ za liniju EXTI15_10 unutar vektorske tabele

Blinkanje na pritisak tastera EXTI

U ovom primer ćemo menjati stanje LED-a unutar ISR-a (Interrupt Service Routine). Koraci za razvoj programa su sledeći:

- Pokrenuti STM32CubeIDE.
- Kreirati novi workspace.
- Kliknuti na opciju za kreiranje novog STM32 projekta.
- Dodeliti ime programu: EXTERNALINT.
- Kliknuti na PA5 i konfigurirati ga kao digitalni izlaz (Digital Output).
- Kliknuti na PC13 i odabrati GPIO_EXTI13 .

Blinkanje na pritisak tastera EXTI



Blinkanje na pritisak tastera EXTI

- Konfigurisati takt MCU-a da radi na 72 MHz.
- Kliknuti na System Core, zatim na GPIO, i potom odabrati PC13.
- Podesiti GPIO mod na External interrupt mode with falling edge trigger detection (Eksterni prekid sa detekcijom padajuće ivice), kao što je prikazano na slici.

Blinkanje na pritisak tastera EXTI

- Konfigurisati takt MCU-a da radi na 72 MHz.
- Kliknuti na System Core, zatim na GPIO, i potom odabrati PC13.
- Podesiti GPIO mod na External interrupt mode with falling edge trigger detection (Eksterni prekid sa detekcijom padajuće ivice), kao što je prikazano na slici.

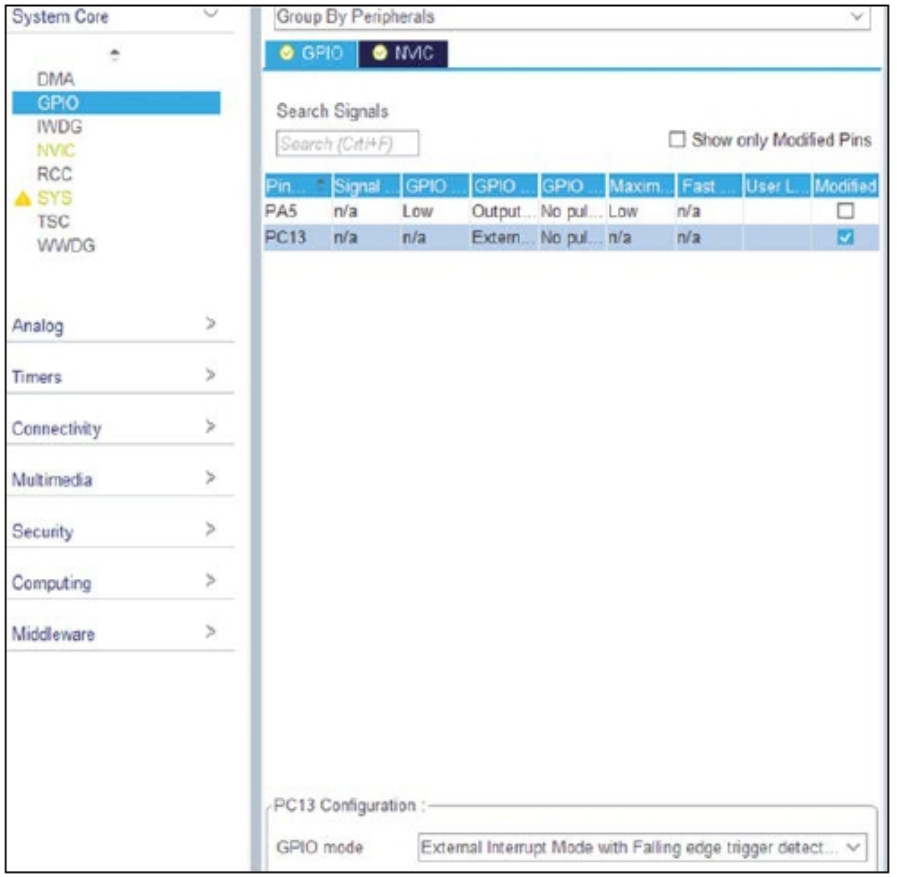


Figure 5.69 Select falling edge detection.

Blinkanje na pritisak tastera EXTI

- Kliknite na NVIC (Nested Vector Interrupt Controller).
- Omogućite EXTI liniju[15:10] tako što ćete je označiti kao aktivnu.
- Podesite Preemption Priority i Subpriority na 0, kao što je prikazano na slici.

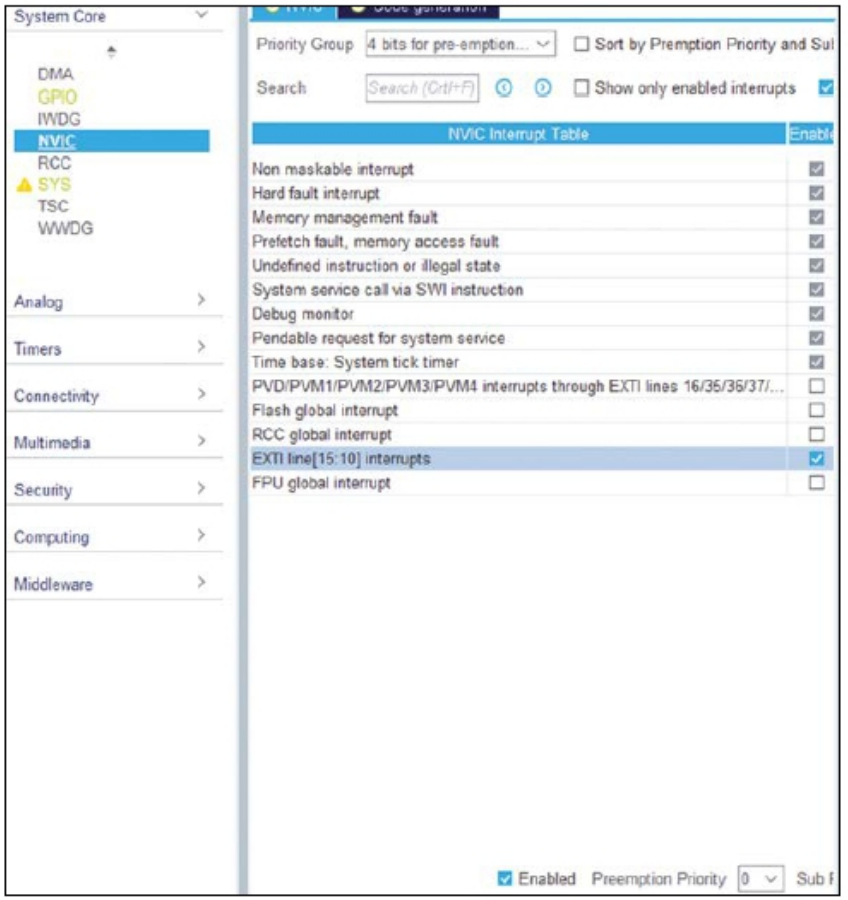


Figure 5.70: Enable EXTI line.

Blinkanje na pritisak tastera EXTI

- Kliknite na File, zatim na Save, i izaberite YES kako biste generisali kod.
- Unesite sledeću naredbu za LED:

  #define myled GPIO_PIN_5

- Obratite pažnju na sledeće dodatne naredbe koje se nalaze na kraju funkcije MX_GPIO_Init(void), gde se podešava prioritet prekida i omogućava eksterni prekid na liniji EXTI15_10:

```
HAL_NVIC_SetPriority(EXTI15_10_IRQn, 0, 0);
HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);
```

Blinkanje na pritisak tastera EXTI

- Zanimljivo je primetiti da se svaki put kada se dogodi eksterni prekid, aktivira funkcija pod nazivom EXTI15_10_IRQHandler u fajlu stm32l4xx_it.c, koji se nalazi u folderu src.
- Sadržaj ove funkcije izgleda ovako (komentari su uklonjeni radi preglednosti):

```
void EXTI15_10_IRQHandler(void)
{
   HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_13);
}
```

- Ova funkcija poziva HAL (Hardware Abstraction Layer) funkciju HAL_GPIO_EXTI_IRQHandler, koja obrađuje prekid vezan za pin GPIO_PIN_13 (u ovom slučaju PC13).

Blinkanje na pritisak tastera EXTI

- Treba da kreiramo funkciju rutine prekida sa nazivom **HAL_GPIO_EXTI_Callback** i da unutar ove funkcije napišemo našu rutinu za obradu prekida.
- U ovom slučaju, jedini kod u rutini prekida menja stanje LED-a:

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    HAL_GPIO_TogglePin(GPIOA, myled);
}
```

- Ova funkcija se automatski poziva kada se dogodi prekid i kada je HAL_GPIO_EXTI_IRQHandler izvršen.
- U ovom primeru, LED koja je povezana na pin PA5 (definisana makroom myled) menja svoje stanje (uključuje/isključuje).

Blinkanje na pritisak tastera EXTI

```
#include «main.h»
#define myled GPIO_PIN_5
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
//
// Interrupt service routine, called when an external interrupt occurs
// i.e. when the button is pressed the LED is toggled by this function
//
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
HAL_GPIO_TogglePin(GPIOA, myled);
}
```

Obrada prekida

STM32 MCU obično ima 7 rukovaoca za prekide GPIO pinova, kao što je prikazano u Tabeli.

IRQ	Handler	Description
EXTI0_IRQn	EXTI0_IRQHandler	Handler for GPIO pins connected to line 0
EXTI1_IRQn	EXTI1_IRQHandler	Handler for GPIO pins connected to line 1
EXTI2_IRQn	EXTI2_IRQHandler	Handler for GPIO pins connected to line 2
EXTI3_IRQn	EXTI3_IRQHandler	Handler for GPIO pins connected to line 3
EXTI4_IRQn	EXTI4_IRQHandler	Handler for GPIO pins connected to line 4
EXTI9_5_IRQn	EXTI9_5_IRQHandler	Handler for GPIO pins connected to line 5 to 9
EXTI15_10_IRQn	EXTI15_10_IRQHandler	Handler for GPIO pins connected to line 10 to 15

Table 5.6: STM32 MCU interrupt handlers for GPIO.

- Tabela prikazuje koji IRQ treba postaviti za NVIC (prvi kolona) i imena funkcija koje obrađuju te prekide (drugi kolona).
- Važno je napomenuti da samo linije 0–4 imaju svoje vlastite rukovaoce prekidima.
- Linije 5–9 imaju isti rukovaoc prekida, kao i linije 10–15.
- Ako koristimo više prekida sa linija 5–9 ili sa linija 10–15, moramo u softveru proveriti koja linija je zapravo izazvala prekid.

Obrada prekida

- Prekidi se omogućavaju pomoću funkcije:

HAL_NVIC_EnableIRQ(IRQn_Type IRQn);

- Odgovarajuća funkcija za onemogućavanje IRQ-a je:

HAL_DisableIRQ(IRQn_Type IRQn);

- Cortex-M automatski obavlja većinu posla u vezi sa obradom prekida.

- Prekid može biti u jednom od tri stanja:

- Onemogućen ili omogućen
- Na čekanju ili nije na čekanju
- Aktivan ili neaktivan

Obrada prekida

- Kada se omogući prekid, on prelazi u stanje pending (na čekanju) dok procesor ne bude spreman da ga obradi.
- Ako trenutno ne postoji nijedan drugi prekid koji se obrađuje, stanje čekanja se briše, prekid postaje aktivan i procesor ga obrađuje.
- Funkcija **HAL_NVIC_GetPendingIRQ(IRQn_Type IRQn)** može se pozvati da proveri da li je prekid na čekanju, ali nije aktivan. Ako IRQ nije na čekanju, funkcija vraća 0, a ako je na čekanju, vraća 1.
- Bit čekanja za IRQ može se postaviti pozivom funkcije:

HAL_NVIC_SetPendingIRQ(IRQn_Type IRQn);

Obrada prekida

- Postavljanje bita čekanja stavlja prekid u stanje čekanja, gde može postati aktivan kada procesor ne obrađuje druge prekide.
- Bit čekanja za prekid može se očistiti pomoću funkcije:

HAL_NVIC_ClearPendingIRQ(IRQn_Type IRQn);

- Funkcija HAL_NVIC_GetActive(IRQn_Type IRQn) može se koristiti da proverite da li je prekid trenutno aktivan.

Obrada prekida

- Prioritet prekida se definiše putem 8-bitnog IPR registra, koji omogućava do 255 različitih nivoa prioriteta.
- Što je niži nivo prioriteta, to je veći stvarni prioritet prekida. U STM32 procesorima koristi se samo 4 gornja bita IPR registra, dok su niži bitovi svi nule.
- To znači da postoji samo 16 nivoa prioriteta: 0x00, 0x10, 0x20, 0x30, 0x40, 0x50, 0x60, 0x70, 0x80, 0x90, 0xA0, 0xB0, 0xC0, 0xD0, 0xE0, 0xF0.
-
- Pošto manji broj prioriteta znači veći prioritet, na primer, IRQ sa prioritetom 0x20 ima viši prioritet od IRQ sa prioritetom 0x80.
- Prioritet je važna karakteristika MCU-a.
- Ako se dva prekida dogode u isto vreme, onda će onaj sa višim prioritetom biti obrađen prvi od strane MCU-a.
- Takođe, ako procesor obrađuje neki prekid, drugi prekid sa višim prioritetom će obustaviti trenutni prekid, a MCU će obraditi prekid sa višim prioritetom.
- Kada se prekid sa višim prioritetom završi, izvršavanje će se vratiti na prethodni prekid.

Obrada prekida

- IPR registar je logički podeljen na dva dela: broj bitova koji definišu
- Preemptive prioritet i broj bitova koji definišu
- Subpriority prioritet. Preemptive prioritet upravlja prioritetima pre-empcije između ISR-ova. Subpriority određuje koji će ISR biti izvršen prvi ako postoji više prekida na čekanju sa istim Preemptive prioritetima.
- Prekid sa višim Subpriority (tj. manjim brojem Subpriority) biće obrađen ako postoji više prekida sa istim Preemptive prioritetima na čekanju, a aktivan je prekid sa višim Preemptive prioritetom.
- Kada ovaj viši Preemptive prekid završi, on će omogućiti izvršavanje prekida sa višim Subpriority.
- Na primer, pretpostavimo da postoje tri izvora prekida pod nazivima A, B i C, svi sa Preemptive prioritetom 0.
- Takođe, pretpostavimo da izvor prekida B ima Subpriority 0, a izvor prekida C ima Subpriority 1.
- Sada, pretpostavimo da je prekid A aktivan. Kada prekid B dođe, on se stavlja u stanje čekanja.
- Takođe, kada prekid C dođe, on se takođe stavlja u stanje čekanja. Kada prekid A završi, sledeći prekid koji postaje aktivan je B, jer on ima viši Subpriority od C (tj. manji broj Subpriority).

Obrada prekida

- Prioritet prekida može se postaviti tokom faze konfiguracije, kao što je opisano u ovom projektu
- Alternativno, prioritet prekida možemo postaviti korišćenjem funkcije HAL_NVIC_SetPriority:

HAL_NVIC_SetPriority(IRQn_Type IRQn, PreemptPriority, SubPriority);

- PreemptPriority i SubPriority mogu biti konfigurirani u opsegu od 0 do 16 (ove vrednosti se pomeraju u 4 gornja bita IPR registra).
- Grupisanje prioriteta može se definisati korišćenjem funkcije HAL_NVIC_SetPriorityGrouping:

HAL_NVIC_SetPriorityGrouping(PriorityGroup);

- Ova funkcija omogućava definisanje prioriteta tako da se postavi željena kombinacija Preemptive i Subpriority nivoa.

Obrada prekida

- Prioritet prekida može se dobiti pozivanjem funkcije:

HAL_NVIC_GetPriority(IRQn_Type IRQn, uint32_t *PreemptPriority, uint32_t *SubPriority);

- Ova funkcija vraća prioritet prekida za određeni IRQ, gde PreemptPriority i SubPriority predstavljaju pre-empcioni prioritet i subprioritet prekida.
- Takođe, trenutna grupacija prioriteta može se dobiti korišćenjem funkcije:

HAL_NVIC_GetPriorityGrouping(void);

- Ova funkcija vraća vrednost koja označava trenutnu grupaciju prioriteta, koja određuje broj bita dodeljenih za PreemptPriority i SubPriority u IPR registru.

Obrada prekida

- Latencija prekida je važan parametar koji treba uzeti u obzir u aplikacijama sa kritičnim vremenima odgovora.
- Ovaj pojam se odnosi na broj taktova procesora koji su potrebni da MCU odgovori na zahtev za prekid.
- Latencija prekida obično se meri brojem taktova između aktivacije zahteva za prekidom i tačkom kada će prva instrukcija rukovaoca prekidima (ISR) biti izvršena.
- Latencija prekida na Cortex-M procesorima je veoma niska. Cortex-M3 i Cortex-M4 procesori imaju latenciju od 12 taktova.
- Ove vrednosti pretpostavljaju da sistem memorije ima nula čekanja, da usluga prekida nije blokirana drugim trenutno aktivnim prekidom i da trenutna izvršna instrukcija ne vrši neporavnate transfere, što može dodati jedan dodatni ciklus transfera.