# PRIMENA MIKROKONTROLERA-MS1PMK

## 4. deo

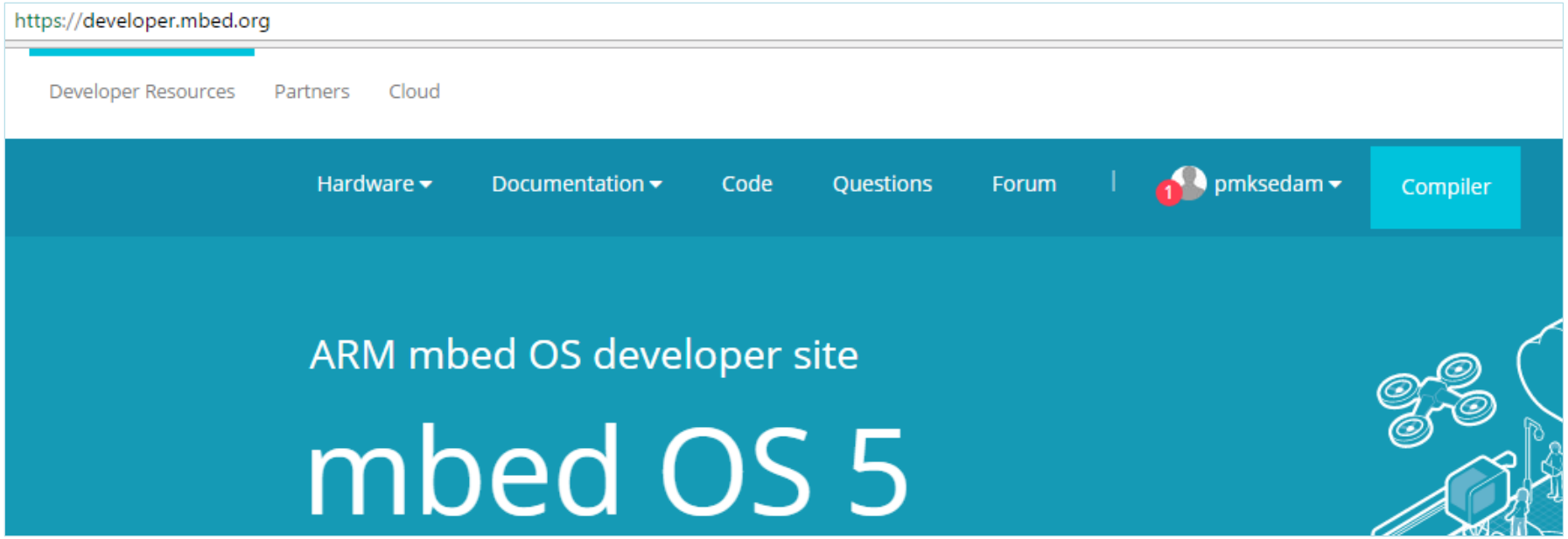
2018

Nenad Jovičić

Marija Janković

# Programiranje – MBED

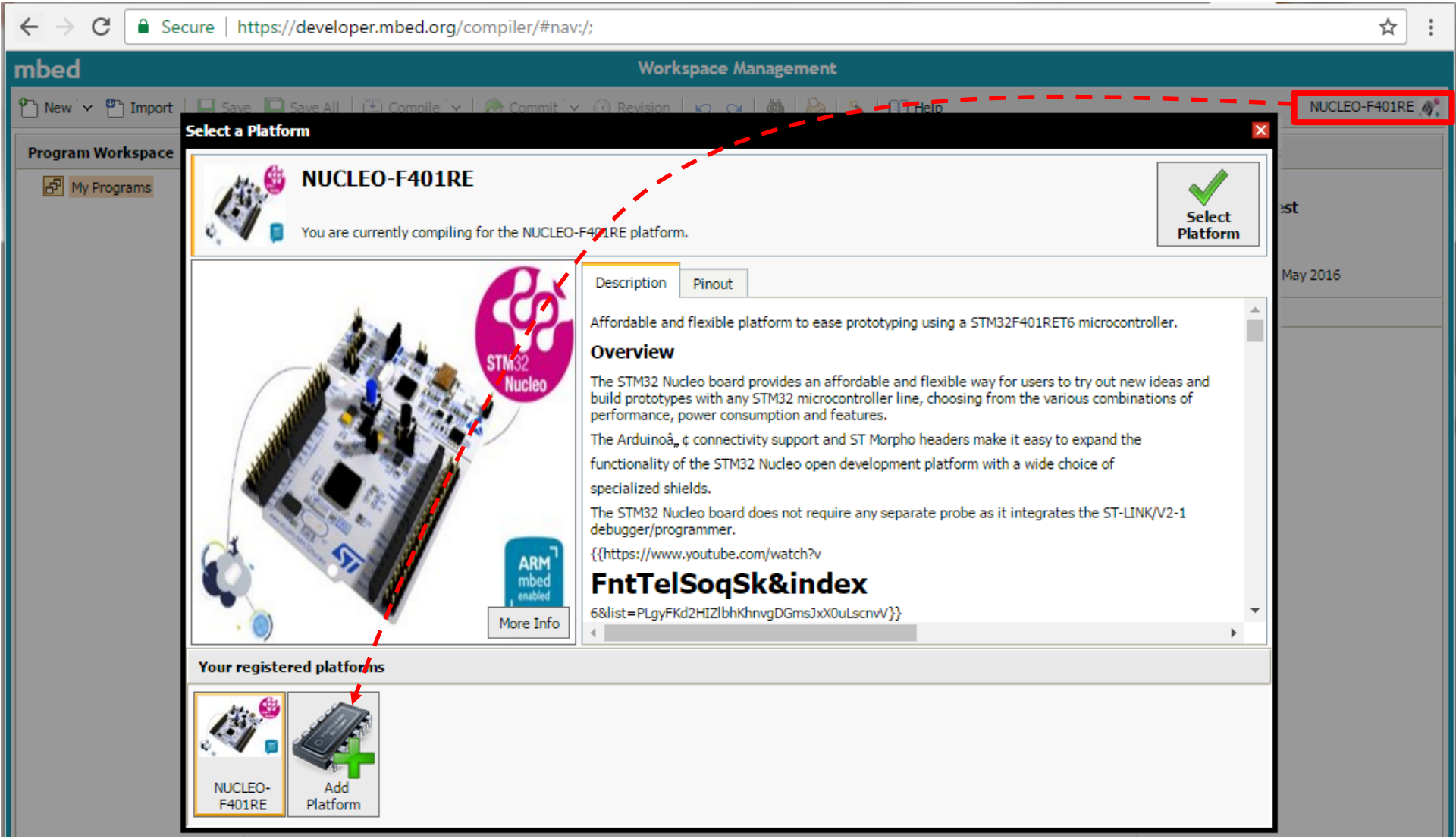
- Jedan od načina da se napiše program za MBED je korišćenje MBED online kompajlera. Potrebno je ulogovati se na sajtu developer.mbed.org i kliknuti na link **Compiler**.
- Kompajler zna da bude spor i nekomforan za editovanje, ali prevođenje radi brzo i nema ograničenja u veličini fajla.



# Logovanje na mbed.org

- developer.mbed.org -> Log In
- Username
  - pmkjedan
  - pmkdva
  - …
  - pmkdeset
- Password: ms1pmk
- Kreirati lični nalog

# Dodavanje nove platforme



# Dodavanje nove platforme

## NUCLEO-L476RG

Affordable and flexible platform to ease prototyping using a STM32L476RGT6 microcontroller.


To compile a program for this board, use **nucleo_l476rg** as the target name.

Board Partner

life.augmented

ST

A world leader in providing the semiconductor solutions that make a positive contribution to people's lives, both today and in the future.

Add to your mbed Compiler

Buy Now

### Overview

The STM32 Nucleo board provides an affordable and flexible way for users to try out new ideas and build prototypes with any STM32 microcontroller line, choosing from the various combinations of performance, power consumption and features.

The Arduino™ connectivity support and ST Morpho headers make it easy to expand the functionality of the STM32 Nucleo open development platform with a wide choice of specialized shields.

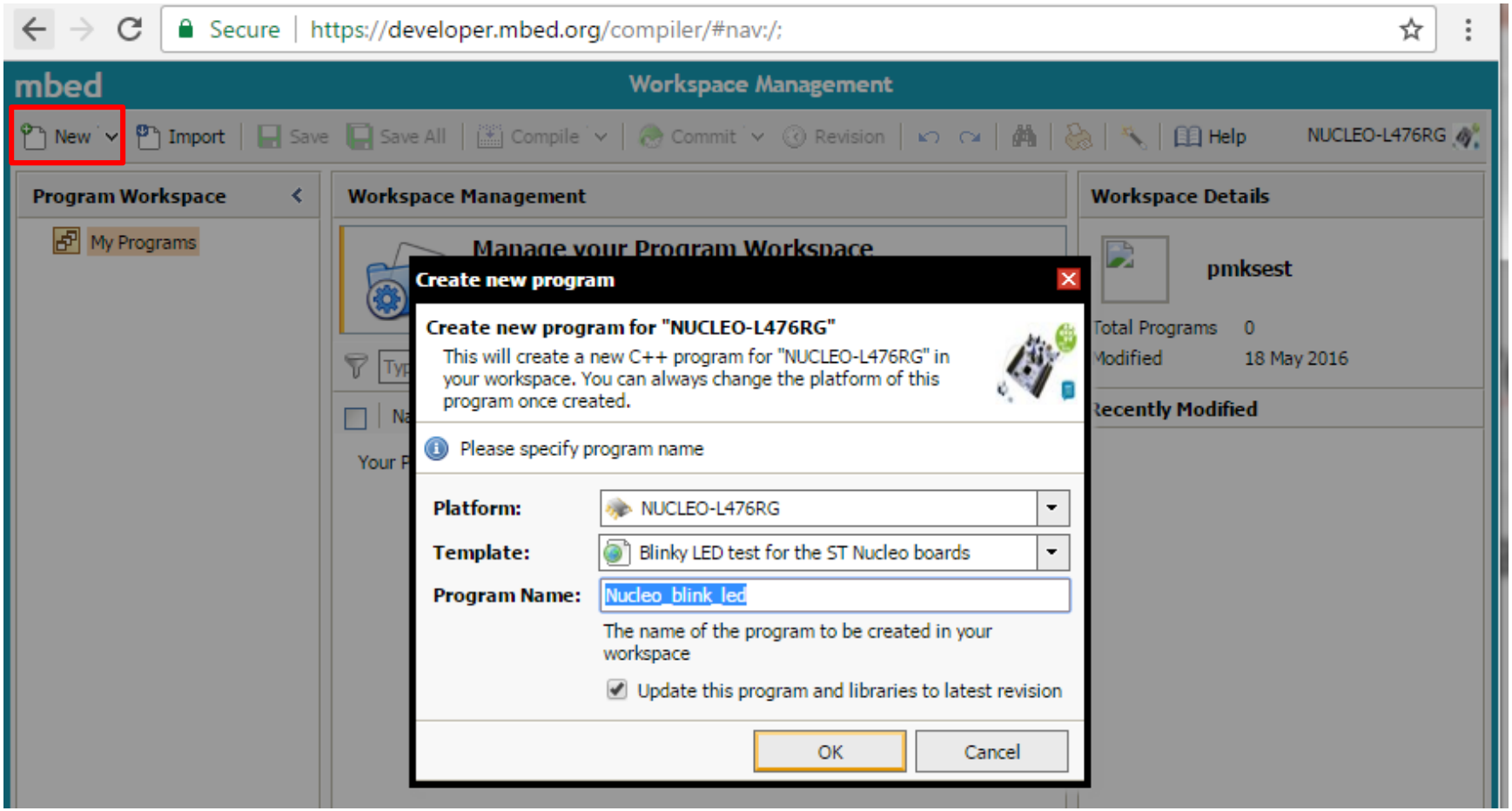
The STM32 Nucleo board does not require any separate probe as it integrates the ST-LINK/V2-1

Table of Contents

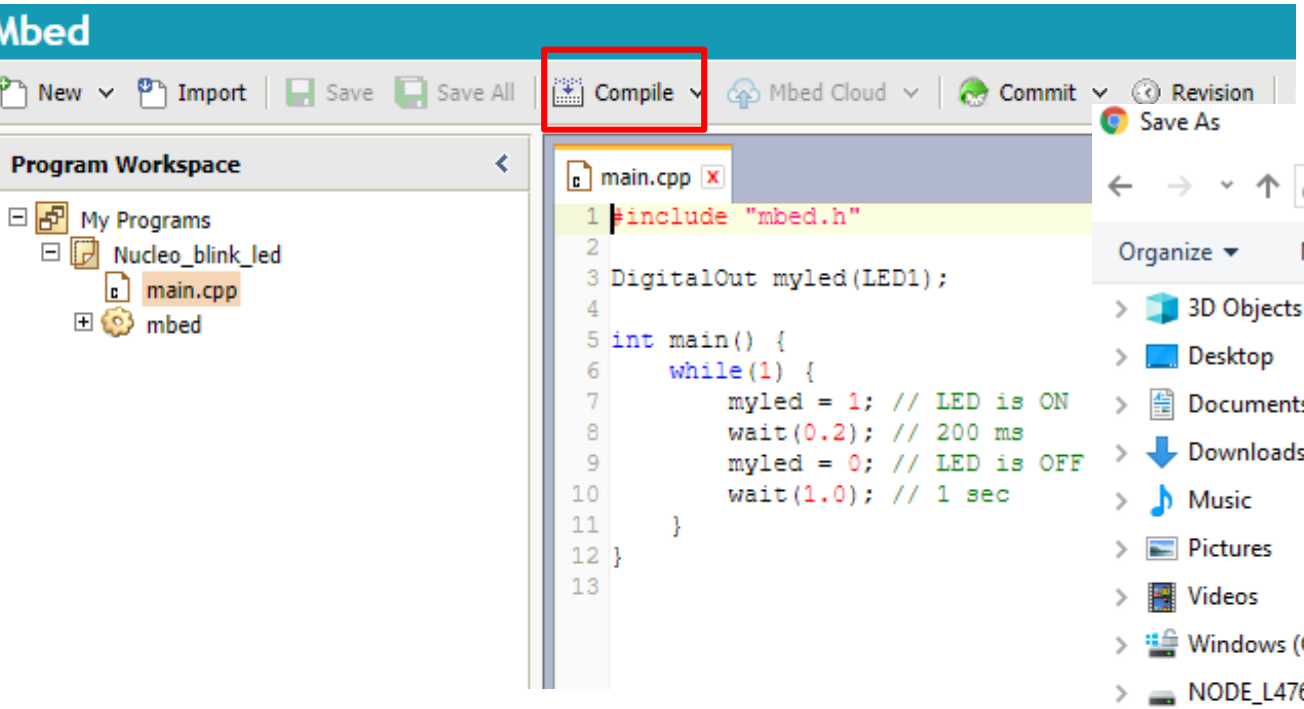


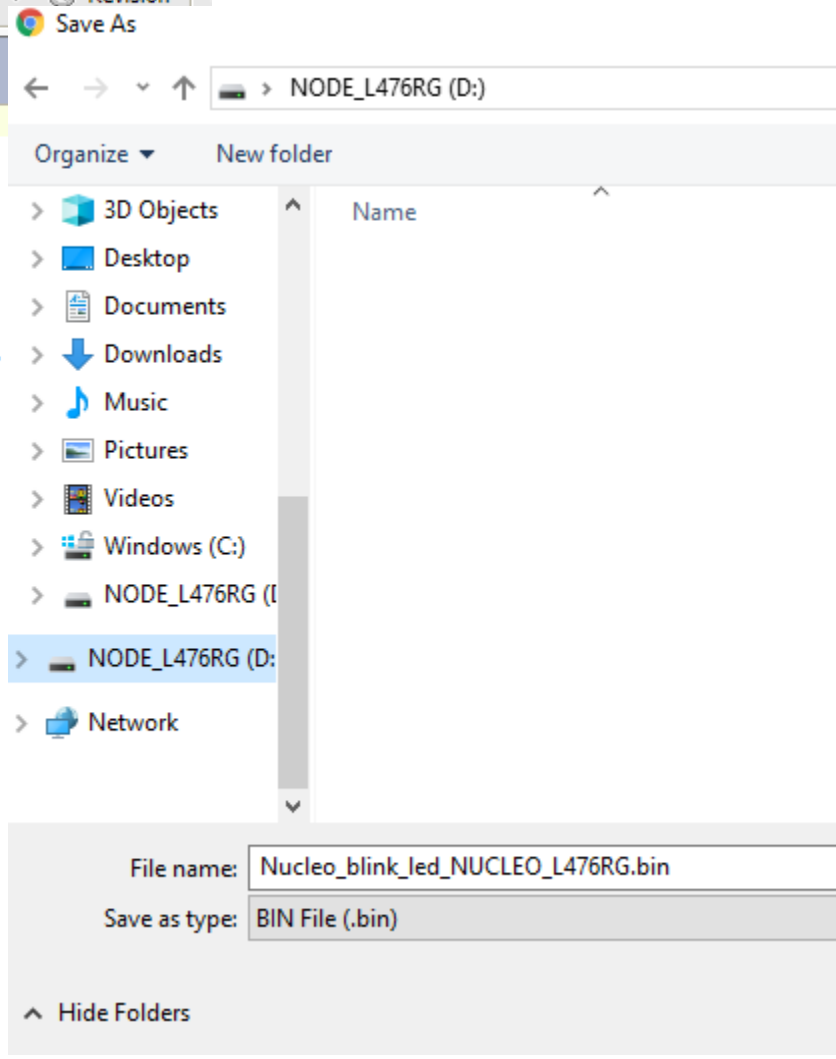
- vratimo se nazad u compiler i selektujemo dodatu platformu

# Kreiranje novog projekta



# Kompajliranje i spuštanje na ploču



# Blinky – C++

```
#include "mbed.h"

DigitalOut myled(LED1);

int main() {
    while(1) {
        myled = 1; // LED is ON
        wait(0.2); // 200 ms
        myled = 0; // LED is OFF
        wait(1.0); // 1 sec
    }
}
```

**naziv tipa** (klasa)

**myled objekat** tipa DigitalOut, poziv konstruktora

**PA5** – GPIOA Pin 5 (definisano u PinNames.h)

Upotreba DigitalOut biblioteke

- Nema adresiranja porta
- Nema bit operacija
- Nema direktnog pristupa registru

**operator=**

DigitalOut& operator= (int value)

# Digitalni izlaz – klasa Digital Out

http://mbed.org/handbook/DigitalOut

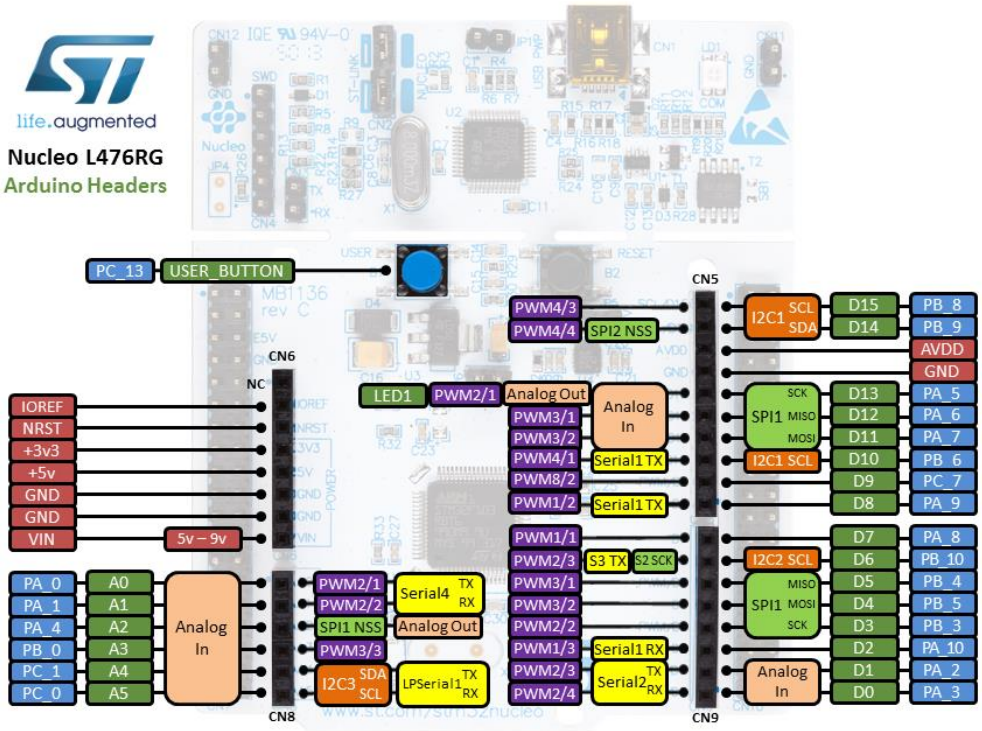
Objekat ove klase je jedan digitalni izlaz, jedan pin mikrokontrolera.

## DigitalOut Class Reference

```
#include <DigitalOut.h>
```

**Public Member Functions**

| | |
|---|---|
| | DigitalOut (PinName pin)<br>Create a **DigitalOut** connected to the specified pin. |
| void | write (int value)<br>Set the output, specified as 0 or 1 (int) |
| int | read ()<br>Return the output setting, represented as 0 or 1 (int) |
| **DigitalOut** & | operator= (int value)<br>A shorthand for write()<br>operator int ()<br>A shorthand for read() |



# Digitalni ulaz – klasa DigitalIn

http://mbed.org/handbook/DigitalIn

## DigitalIn Class Reference

```
#include <DigitalIn.h>
```

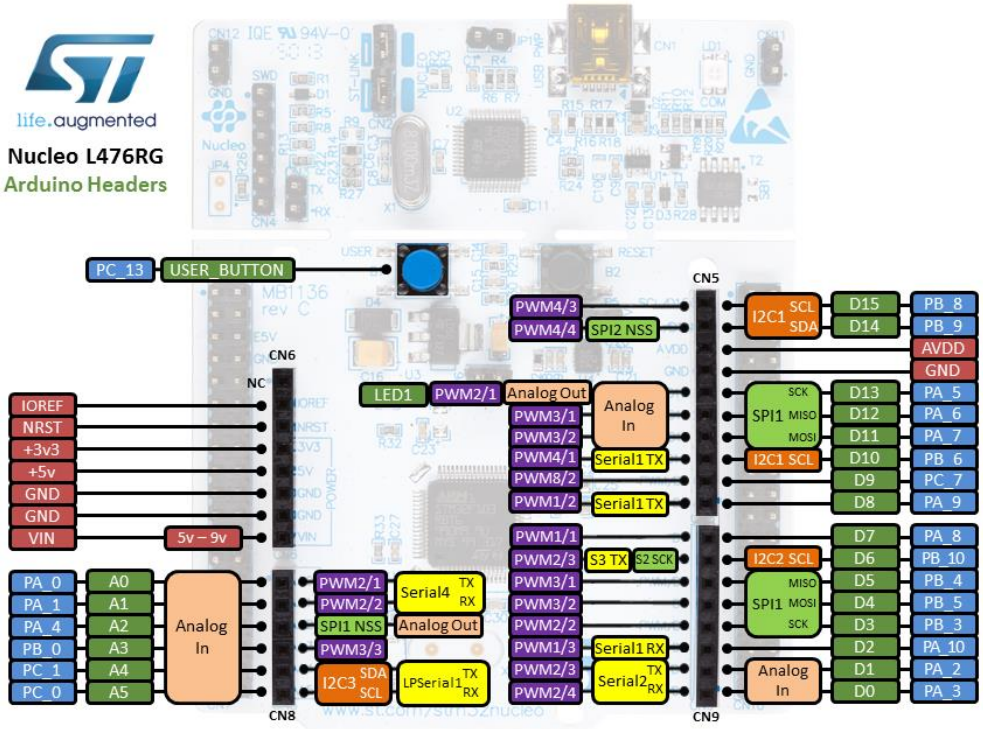
**Public Member Functions**

DigitalIn (PinName pin)
Create a **DigitalIn** connected to the specified pin.

int read ()
Read the input, represented as 0 or 1 (int)

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

operator int ()
An operator shorthand for read()



# DigitalIn

Objekat ove klase je jedan digitalni ulaz, jedan pin mikrokontrolera.

```
#include "mbed.h"

DigitalOut myled(LED1);
DigitalIn mybutton(USER_BUTTON);

int main() {
    while(1) {

        myled = mybutton.read();      // Read button state and
                                      // transfer the value to LED.

    }
}
```

**PC13** – GPIOC Pin 13 (definisano u PinNames.h)

Promeniti kod tako da je dioda uključena kada je pritisnut User taster. Iskoristiti operator int().

# Bidirekcioni digitalni pin – klasa DigitalInOut

http://mbed.org/handbook/DigitalInOut

- Objekat ove klase je jedan pin mikrokontrolera tj. MBED-a koji može da menja smer u toku izvršavanja programa. Ovo je interesantno kod interfejsa koji zahteva promenu smera kao što su na primer I2C ili one-wire magistrala.

```
#include "mbed.h"
DigitalInOut pin(LED2);

int main() {
    while (1) {
        pin.output();
        pin = !int(pin);
        wait_us(500);
        pin.input();
        wait_us(500);
    }
}
```

## DigitalInOut Class Reference

```
#include <DigitalInOut.h>
```

**Public Member Functions**

| | |
|---|---|
| | DigitalInOut (PinName pin)<br>Create a **DigitalInOut** connected to the specified pin. |
| void | write (int value)<br>Set the output, specified as 0 or 1 (int) |
| int | read ()<br>Return the output setting, represented as 0 or 1 (int) |
| void | output ()<br>Set as an output. |
| void | input ()<br>Set as an input. |
| void | mode (PinMode pull)<br>Set the input pin mode. |
| **DigitalInOut** & | operator= (int value)<br>A shorthand for write() |
| | operator int ()<br>A shorthand for read() |

# 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.

# ZADACI

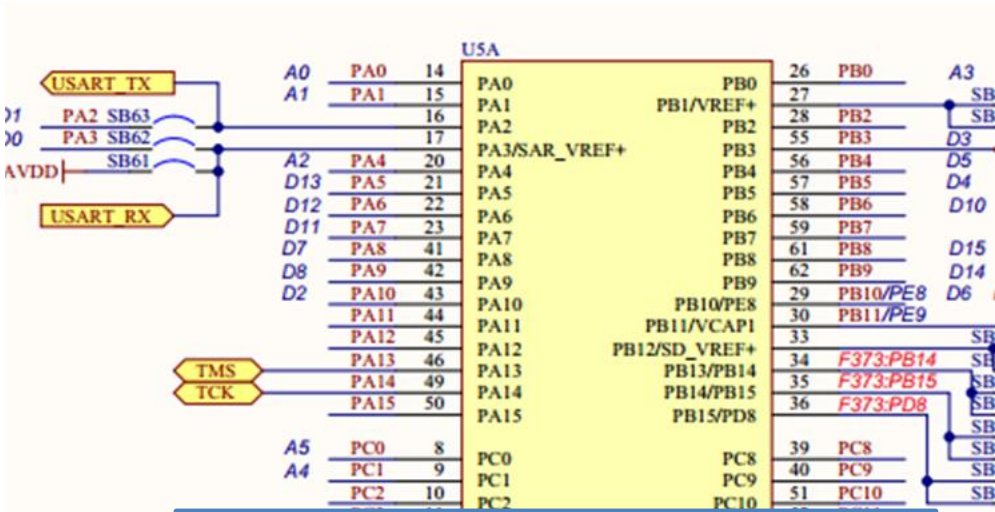
1. Napisati program koji obezbeđuje da se na pritisak tastera menja stanje LE diode.
2. Napisati program koji obezbeđuje da se na pritisak tastera inkrementira broj u opsegu 0-9. Inkrementiranje se vrši po modulu 10.
   - Obezbediti uključivanje LE diode kada je vrednost brojača deljiva sa 3.
   - http://www.cplusplus.com/reference/clibrary/

# Portovi

http://mbed.org/handbook/PortIn (PortOut, PortInOut)

- Port je osnovni način grupisanja digitalnih ulaza tj. izlaza u arhitekturi svih mikrokontrolera pa tako i LPC1768.
- Klase MBED-a koje obezbeđuju pristup originalnim portovima mikrokontrolera su PortIn, PortOut i PortInOut.

```
// Switch on an LED if any of mbed pins GPIOC 12-15 is high
#include "mbed.h"

PortIn p(PortC, 0x0000F000); // pC15-pC12
DigitalOut ind(LED1);

int main() {
    while(1) {
        int pins = p.read();
        if !(pins) {
            ind = 1;
        }
        else {
            ind = 0;
        }
    }
}
```




## PortIn Class Reference

```
#include <PortIn.h>
```

**Public Member Functions**

PortIn (PortName port, int mask=0xFFFFFFFF)
Create an **PortIn**, connected to the specified port.

int read ()
Read the value currently output on the port.

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

operator int ()
A shorthand for read()

# Magistrale

http://mbed.org/handbook/BusIn (BusOut, BusInOut)

- Klase BUSxx su zgodne za definisanje korisničkih portova koji sadrže digitalne ulaze i izlaze koji nisu grupisani u portove u samom mikrokontroleru.

```
#include "mbed.h"
BusOut myleds(PA_7, PA_6, PA_5, PA_4);
int main() {
        while(1) {
                for(int i=0; i<16; i++) {
                        myleds = i;
                        wait(0.25);
                }
        }
}
```

- Ovakva klasa se može iskoristiti na primer za uključivanje sedmosegmentnog LED modula povezanog na MBED.

## BusIn Class Reference

```
#include <BusIn.h>
```

**Public Member Functions**

BusIn (PinName p0, PinName p1=NC, PinName p2=NC, PinName p3=NC, PinName p4=NC, PinName p5=NC, PinName p6=NC, PinName p7=NC, PinName p8=NC, PinName p9=NC, PinName p10=NC, PinName p11=NC, PinName p12=NC, PinName p13=NC, PinName p14=NC, PinName p15=NC)

Create an **BusIn**, connected to the specified pins.

int read ()

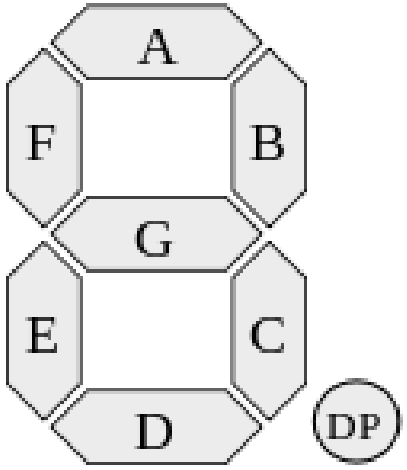
Read the value of the input bus.

operator int ()

A shorthand for read()

# ZADACI

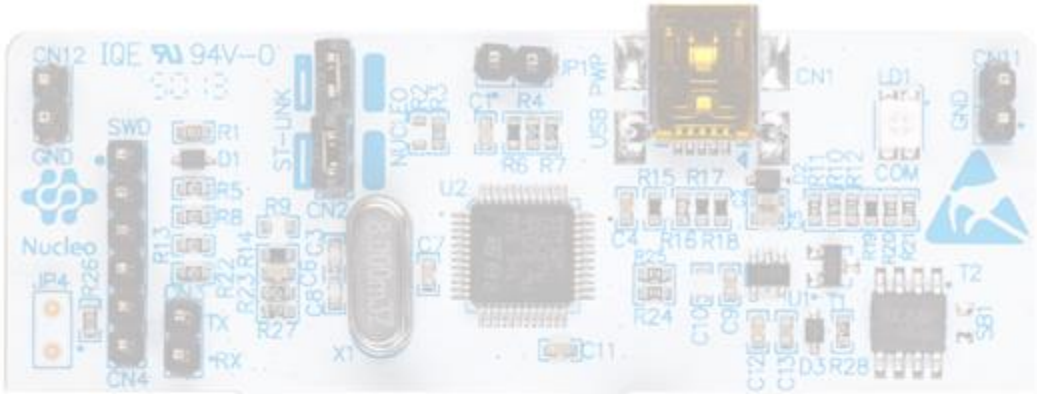
3. Napisati program koji trenutnu vrednost brojača iz tačke 2 prikazuje na sedmosegmentnom LED displeju.

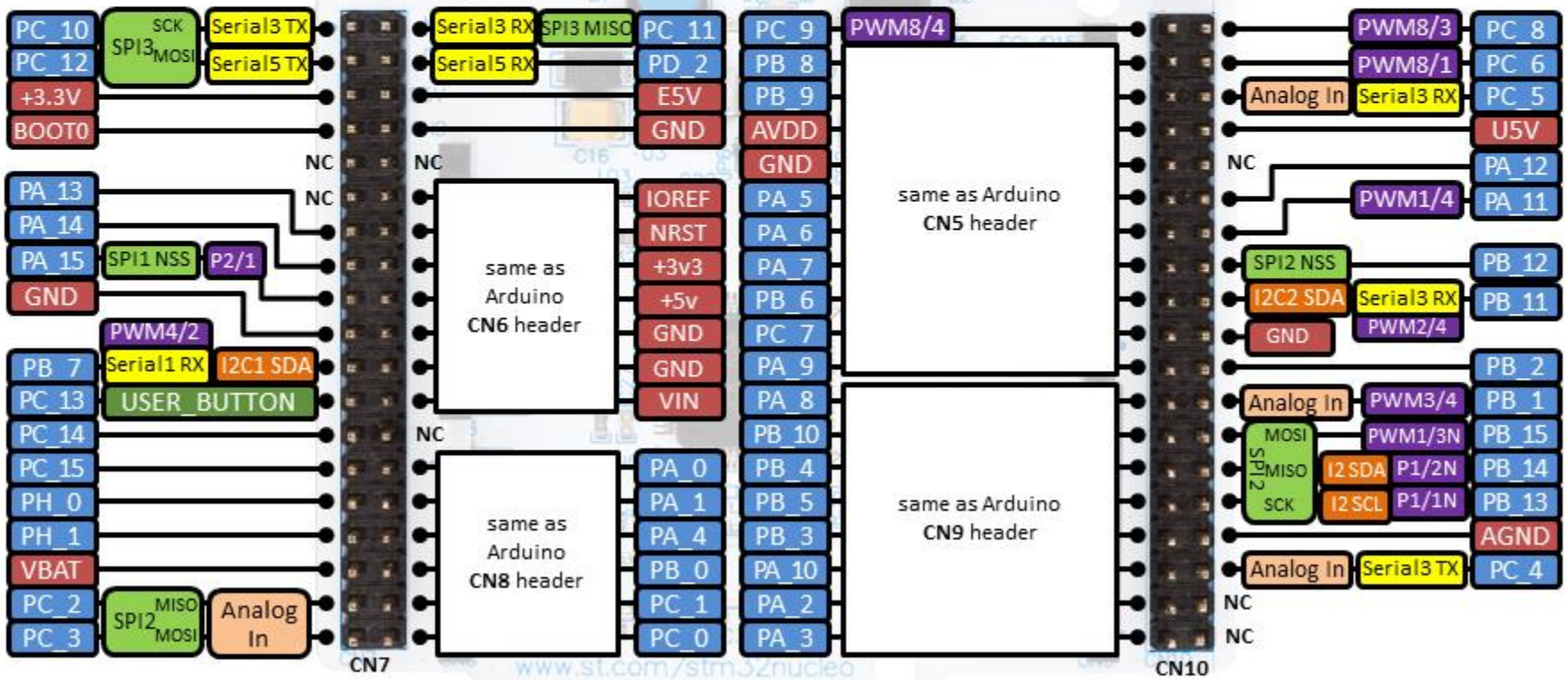


life.augmented

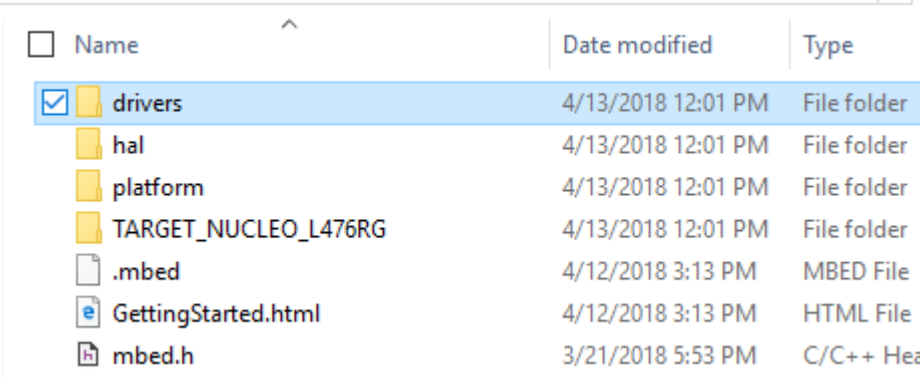
# Nucleo L476RG

## Morpho Headers





# MBED - source code




- MBED github
- MBED SDK – kompletan source kode je free i može da se download-uje.
- Osnovni fajl je mbed.h i on kao hijerarhijski najviši uključuje sve ostale heder fajlove.
- U drivers (api) folderu se nalaze heder fajlovi u kojima su definisani svi MBED objekti. Na primer DigitalOut.h, BusOut.h...
- Neki objekti, kao što je BusOut.h, nemaju fizički ekvivalent u hardveru već se dobijaju kombinovanjem funkcionalnosti osnovnih objekata.
- U hal folderu se nalaze heder fajlovi u kojima su definisani svi osnovni objekti kojima se predstavljaju fizičke hardverske komponente. Primer gpio_api.h
- U folderu targets se nalaze svi fajlovi koji definišu specifične implementacije osnovnih hardverskih objekata deklarisanih u folderu hal.

# mbed.h

# platform.h

mbed_config.h | mbed.h | platform.h | device.h | objects.h

```
35
36  #define MBED_VERSION MBED_ENCODE_VERSION(MBED_MAJOR_VE
37
38  #if MBED_CONF_RTOS_PRESENT
39  #include "rtos/rtos.h"
40  #endif
41
42  #if MBED_CONF_NSAPI_PRESENT
43  #include "netsocket/nsapi.h"
44  #endif
45
46  #if MBED_CONF_EVENTS_PRESENT
47  #include "events/mbed_events.h"
48  #endif
49
50  #if MBED_CONF_FILESYSTEM_PRESENT
51  #include "filesystem/mbed_filesystem.h"
52  #endif
53
54  #include "platform/mbed_toolchain.h"
55  #include "platform/platform.h"
56  #include "platform/mbed_application.h"
57
58  // Useful C libraries
59  #include <math.h>
60  #include <time.h>
61
62  // mbed Debug libraries
63  #include "platform/mbed_error.h"
64  #include "platform/mbed_interface.h"
65  #include "platform/mbed_assert.h"
66
67  // mbed Peripheral components
68  #include "drivers/DigitalIn.h"
69  #include "drivers/DigitalOut.h"
70  #include "drivers/DigitalInOut.h"
71  #include "drivers/BusIn.h"
```

mbed_config.h | mbed.h | platform.h | device.h

```
1
2   /** \addtogroup platform */
3   /** @{*/
4   /* mbed Microcontroller Library
19  #ifndef MBED_PLATFORM_H
20  #define MBED_PLATFORM_H
21
22  #include <cstddef>
23  #include <cstdlib>
24  #include <cstdio>
25  #include <cstring>
26
27  #include "platform/mbed_retarget.h"
28  #include "platform/mbed_toolchain.h"
29  #include "device.h"
30  #include "PinNames.h"
31  #include "PeripheralNames.h"
32
33  #endif
34
35  /** @}*/
36
```

- Device.h
- U ovom fajlu su se nekad definisali resursi konkretnog hardvera, to se u novijim verzijama nalazi u samim opcijama projekta
- Target Options -> C/C++ -> Misc Controls

gpio.c | gpio_api.h | device.h | DigitalOut.h | stm32f10x_gpio.c | gpio_api.c | BusOut.cpp

```
17  #define MBED_DEVICE_H
18
19  #define DEVICE_PORTIN           1
20  #define DEVICE_PORTOUT          1
21  #define DEVICE_PORTINOUT        1
22
23  #define DEVICE_INTERRUPTIN      1
24
25  #define DEVICE_ANALOGIN         1
26  #define DEVICE_ANALOGOUT        1
27
28  #define DEVICE_SERIAL           1
29  #define DEVICE_SERIAL_FC        1
30
31  #define DEVICE_I2C              1
32  #define DEVICE_I2CSLAVE         1
33
34  #define DEVICE_SPI              1
35  #define DEVICE_SPISLAVE         1
36
37  #define DEVICE_CAN              1
38
39  #define DEVICE_RTC              1
40
41  #define DEVICE_ETHERNET         1
42
43  #define DEVICE_PWMOUT           1
44
45  #define DEVICE_SEMIHOST         1
46  #define DEVICE_LOCALFILESYSTEM  1
47  #define DEVICE_ID_LENGTH       32
48  #define DEVICE_MAC_OFFSET      20
49
50  #define DEVICE_SLEEP            1
51
52  #define DEVICE_DEBUG_AWARENESS  1
53
54  #define DEVICE_STDIO_MESSAGES   1
```

- Device.h fajl za STM32F401RE

new 2 | 24.03.2016_23.49.21.zgz | device.h

```
#define DEVICE_PORTIN           1
#define DEVICE_PORTOUT          1
#define DEVICE_PORTINOUT        1

#define DEVICE_INTERRUPTIN      1

#define DEVICE_ANALOGIN         1
#define DEVICE_ANALOGOUT        1

#define DEVICE_SERIAL           1

#define DEVICE_I2C              1
#define DEVICE_I2CSLAVE         1

#define DEVICE_SPI              1
#define DEVICE_SPISLAVE         1

#define DEVICE_RTC              1

#define DEVICE_PWMOUT           1

#define DEVICE_SLEEP            1

//========================================
```

gpio.c | AnalogOut.h | gpio_api.h | device.h | PinNames.h | DigitalOut.h | stm32f10x_gpio.c | gp

```
15   */
16  #ifndef MBED_ANALOGOUT_H
17  #define MBED_ANALOGOUT_H
18
19  #include "platform.h"
20
21  #if DEVICE_ANALOGOUT
22
23  #include "analogout_api.h"
24
25  namespace mbed {
26
27  /** An analog output, used for setting the voltage on a pin
28   *
29   * Example:
30   * @code
31   * // Make a sawtooth output
32   *
33   * #include "mbed.h"
34   *
35   * AnalogOut tri(p18);
36   * int main() {
37   *     while(1) {
38   *         tri = tri + 0.01;
39   *         wait_us(1);
40   *         if(tri == 1) {
41   *             tri = 0;
42   *         }
43   *     }
44   * }
45   * @endcode
46   */
47  class AnalogOut {
48
```

Ako je u Device.h definisano postojanje objekta isti se uključuje u projekat

# PinNames.h

- Ovde se definišu simboličke oznake pinova koje odgovaraju odgovarajućem target-u.
- Originalne oznake su u fajlu cmsis.h, tj. stm32l476xx.h

PinNames.h

```
162         LED1          = PA_5,
163         LED2          = PA_5,
164         LED3          = PA_5,
165         LED4          = PA_5,
166         USER_BUTTON = PC_13,
167         SERIAL_TX     = PA_2,
168         SERIAL_RX     = PA_3,
169         USBTX         = PA_2,
170         USBRX         = PA_3,
171         I2C_SCL       = PB_8,
172         I2C_SDA       = PB_9,
173         SPI_MOSI      = PA_7,
174         SPI_MISO      = PA_6,
175         SPI_SCK       = PA_5,
176         SPI_CS        = PB_6,
177         PWM_OUT       = PB_3,
178
179         // Not connected
180         NC = (int)0xFFFFFFFF
181     } PinName;
182
183     typedef enum {
184         PullNone  = 0,
185         PullUp    = 1,
186         PullDown  = 2,
187         OpenDrain = 3,
188         PullDefault = PullNone
189     } PinMode;
190
```

# Peripheral Names.h

ames.h | PeripheralNames.h

```
typedef enum {
    ADC_1 = (int)ADC1_BASE,
    ADC_2 = (int)ADC2_BASE,
    ADC_3 = (int)ADC3_BASE
} ADCName;

typedef enum {
    DAC_1 = (int)DAC_BASE
} DACName;

typedef enum {
    UART_1 = (int)USART1_BASE,
    UART_2 = (int)USART2_BASE,
    UART_3 = (int)USART3_BASE,
    UART_4 = (int)UART4_BASE,
    UART_5 = (int)UART5_BASE,
    LPUART_1 = (int)LPUART1_BASE
} UARTName;

#define STDIO_UART_TX  PA_2
#define STDIO_UART_RX  PA_3
#define STDIO_UART     UART_2

typedef enum {
    SPI_1 = (int)SPI1_BASE,
    SPI_2 = (int)SPI2_BASE,
    SPI_3 = (int)SPI3_BASE
} SPIName;
```

# STM32L476RG – kopamo dublje

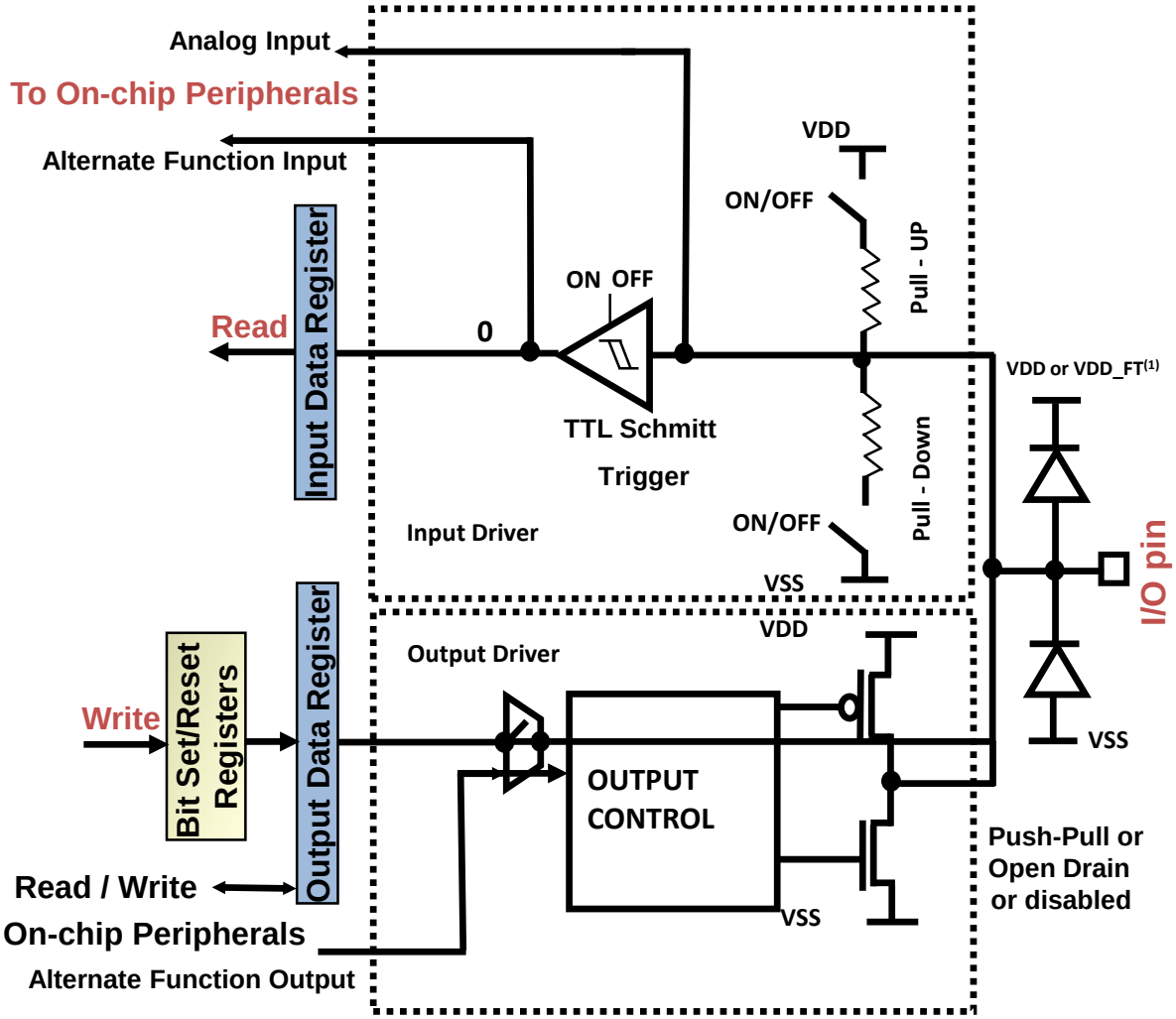
- STM32 Nucleo-64 board – User Manual
  - Podaci vezani za pločicu: pinout, soldering bridges, pull up, pull down, tasteri, diode
- STM32L476xx – MCU Datasheet
  - Provera karakteristika pri izboru mikrokontrolera
  - Alternate pin function
  - Additional pin function
- STM32L4x5 and STM32L4x6 advanced ARM®-based 32-bit MCUs - Reference Manual
  - Detaljan opis periferija, registara, programiranja.
  - Sve što ste hteli da znate ali vas mrzi da čitate

# STM Cortex-M - GPIO - Portovi

- 5V tolerantni ulazi
- Kapacitet po pinu 25mA
- 18 MHz učestanost toglovanja
- Konfigurabilna izlazna brzina do 50 MHz
- Do 16 analognih ulaza
- Alternativne funkcije (USARTx, TIMx, I2Cx, SPIx,…)
- Svaki pin može da generiše spoljašnji prekid
- Jedan pin može da se koristi za buđenje iz STANDBY moda (PA.00)
- Jedan pin može da bude Tamper Pin (za watcdog) (PC.13)
- Pinovi grupisani u 5 16-bitnih portova (GPIOA..GPIOE)
- Mehanizam zaključavanja konfiguracije

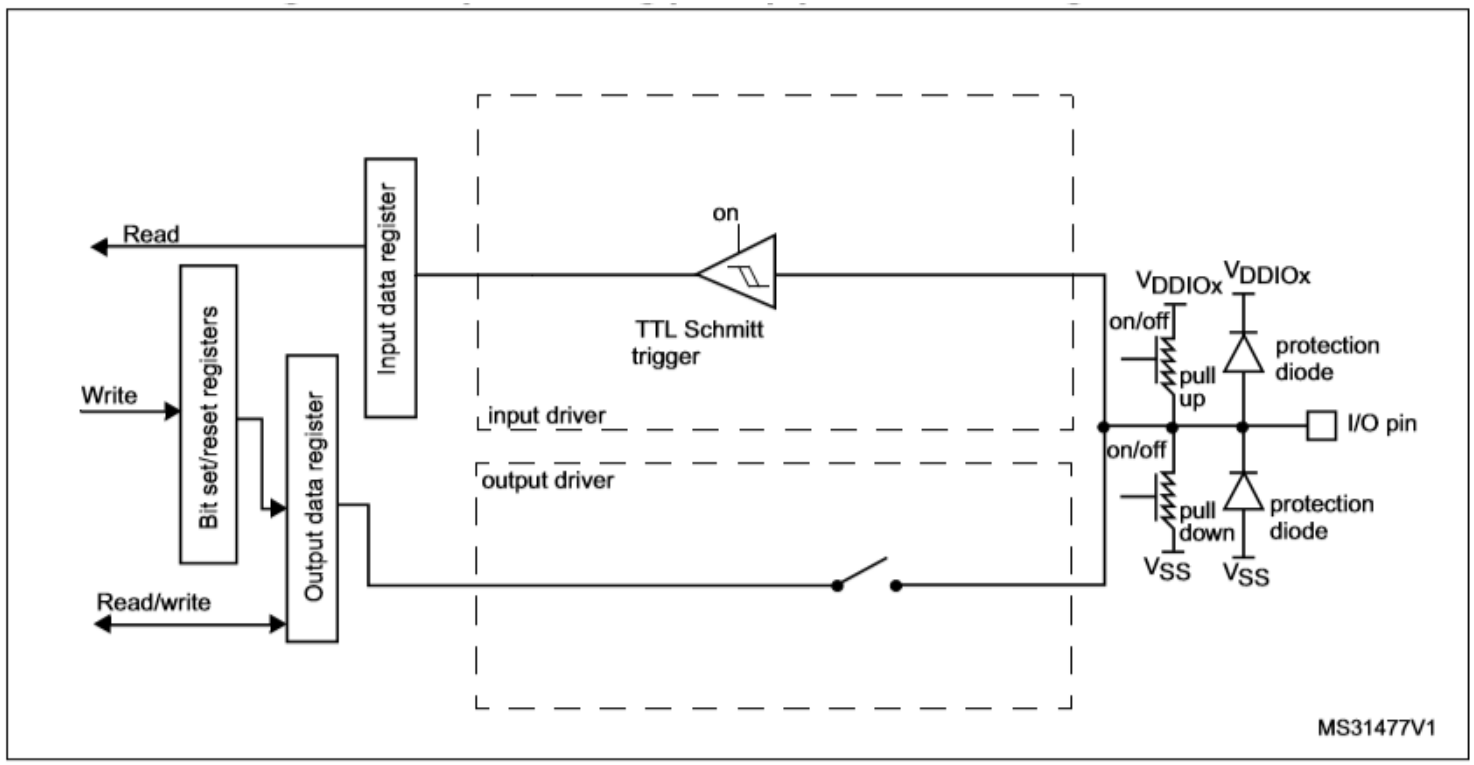
# GPIO Konfiguracije

| Configuration Mode | CNF1 | CNF0 | MOD1 | MOD0 |
|---|---|---|---|---|
| Analog Input/Output **(Reset state)** | 0 | 0 | 00 | |
| Input Floating | 0 | 1 | 01 | |
| Input Pull-Up(2) | 1 | 0 | | |
| Input Pull-Down(2) | | | | |
| Output Push-Pull | 0 | 0 | 10 | |
| Output Open-Drain | 0 | 1 | | |
| AF Push-Pull | 1 | 0 | 11 | |
| AF Open-Drain | 1 | 1 | | |



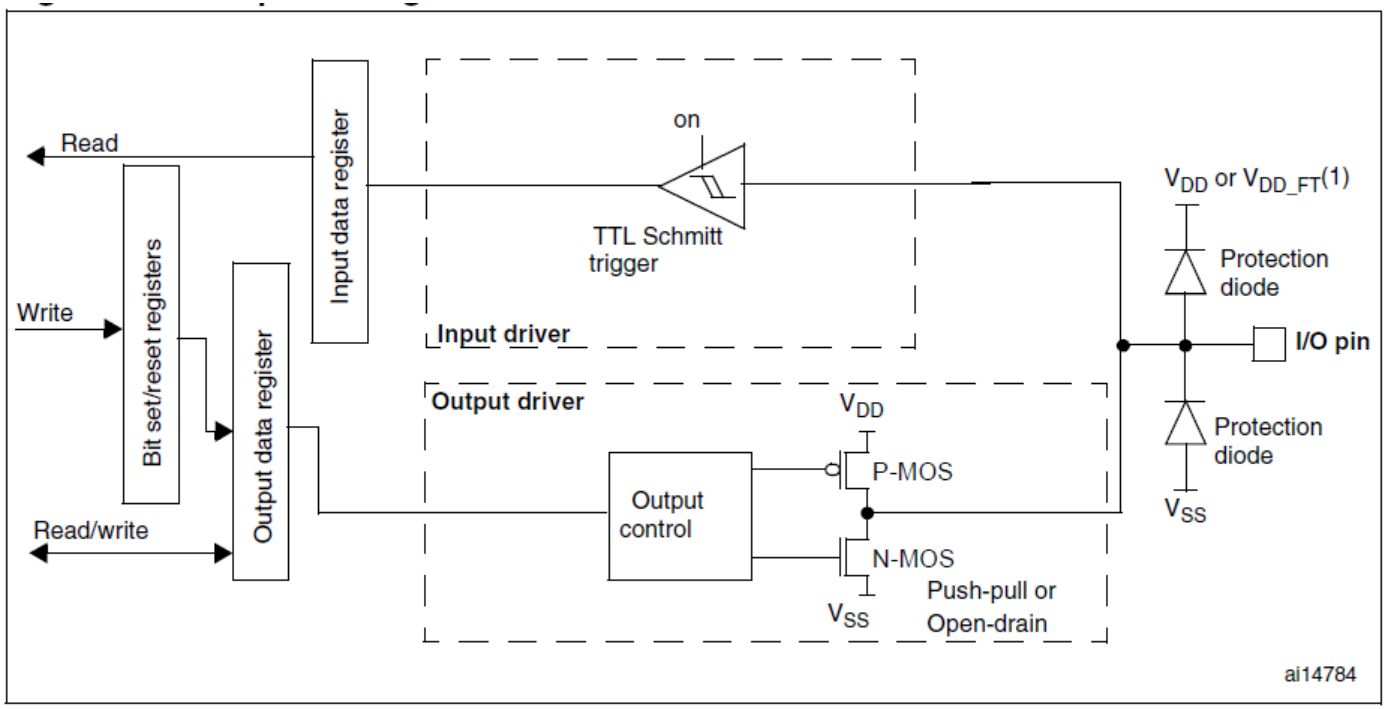
# Ulazna konfiguracija

- **Izlazni bafer** je **isključen**
- **Schmitt trigger** kolo je **aktivno**
- *Pull-up* i *pull-down* otpornici su aktivni u zavisnosti od sadržaja GPIOx_PUPDR registra
- Vrednosti na ulazima pinova se očitavaju i upisuju u ulazni registar na svaki takt AHB magistrale
- Čitanjem ulaznog registra (*Input data register*) dobijaju se digitalna stanja ulaza



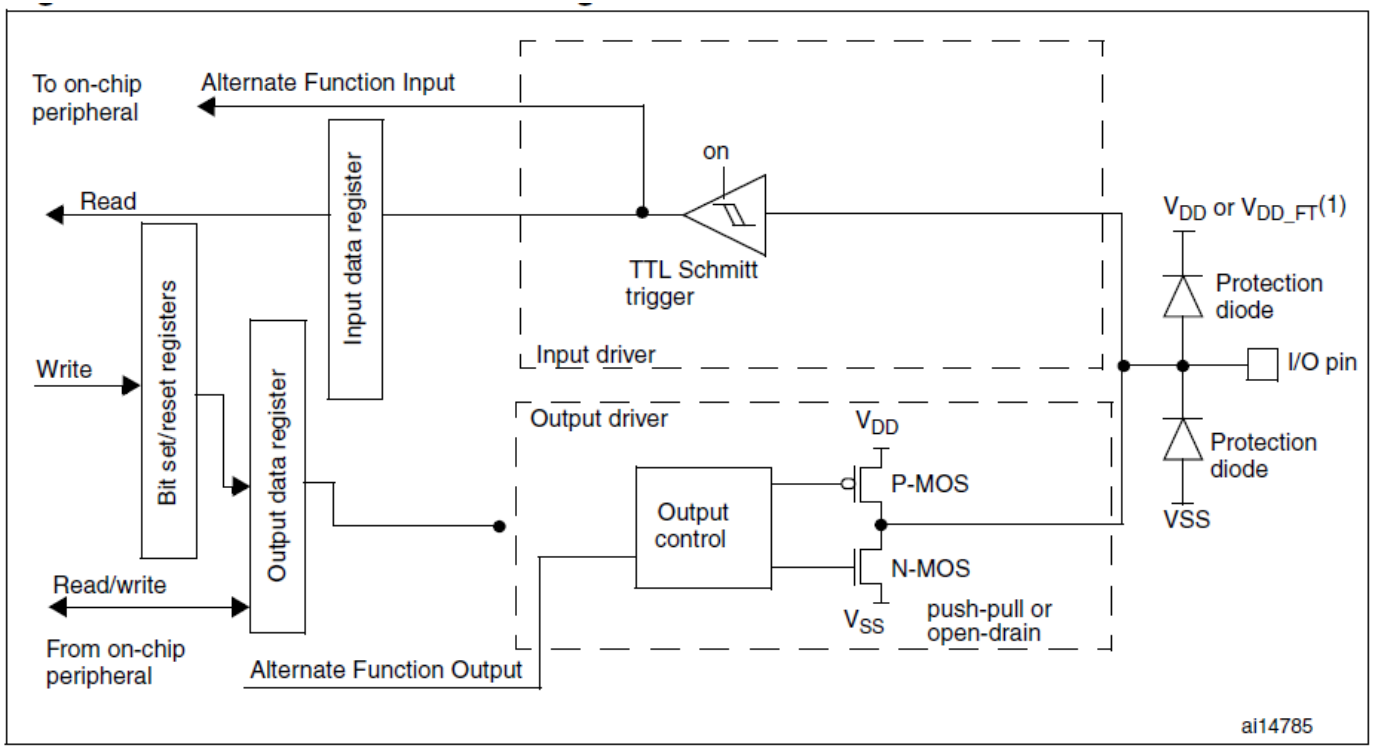
# Izlazna konfiguracija

- ***Open-Drain* mod**: “0” u *Output Data* registru aktivira N-MOS tranzistor, dok “1” u *Output Data* registru ostavlja pin u stanju visoke impedanse (P-MOS transistor se nikada ne aktivira).
- ***Push-pull* mod**: “0” u *Output Data* registru aktivira N-MOS, dok “1” u *Output Data* registru aktivira P-MOStor.
- **Schmitt Trigger** kolo je **uključeno**
- Interni **pull-up i pull-dow**n otpornici su **isključeni**
- Stanje koje je na pinu se sempluje i upisuje u *Input Data* registar na svaki APB2 klok
- Čitanjem *Input Data* registra očitava se stanje pina kada je konfigurisan u *Open-Drain* ili *Push-pull* modu



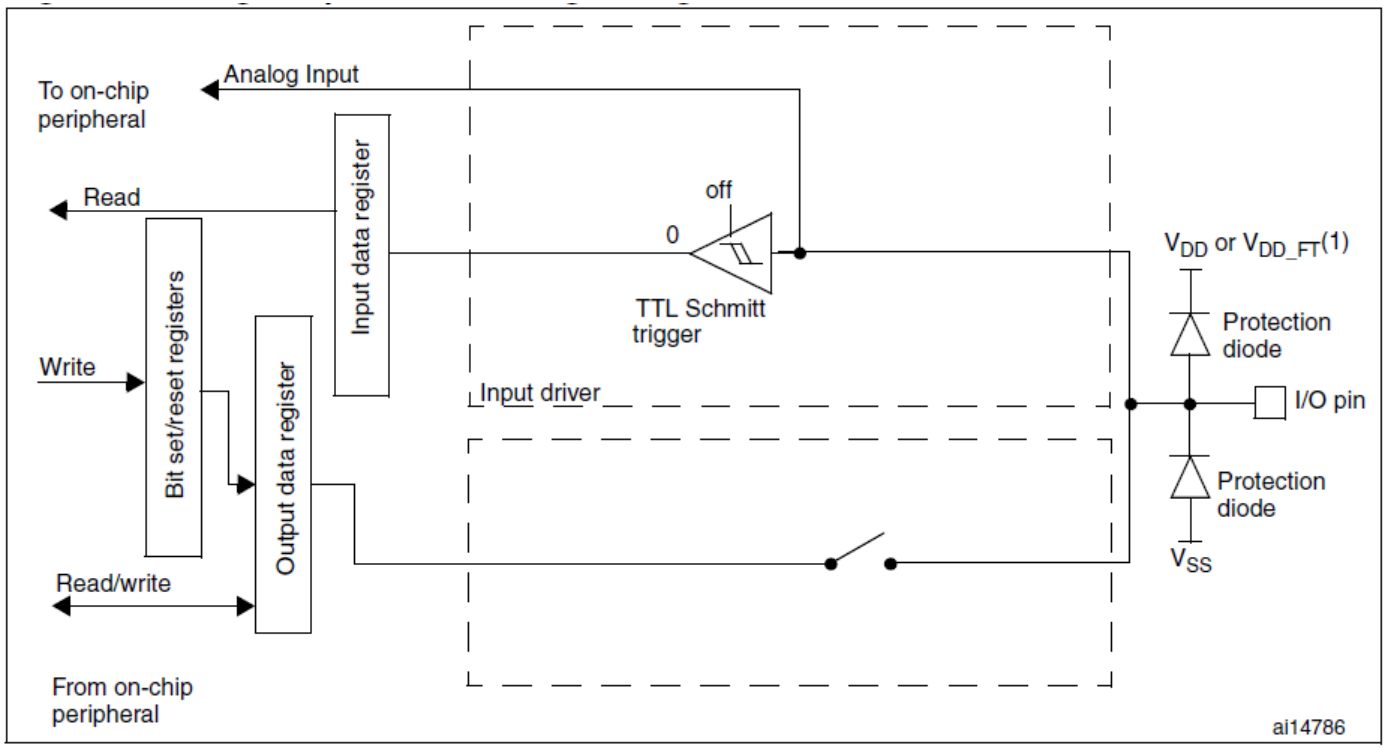
# Konfiguracija alternativne funkcije

- Izlazni dajver je u **Open Drain ili Push-Pull** konfiguraciji
- **Schmitt Trigger** ulaz je **aktiviran**
- **Pull-up i pull-down** otpornici su **deaktivirani**
- Podatak na pinu se sempluje u ulazni registar na svaku ivicu APB2 takta
- Ako je izlaz u open drain modu očitavanje ulaznog registra daje stanje na pinu



# Konfiguracija analognog ulaza

- **Izlazni bafer je isključen** (otkačen)
- **Schmitt Trigger** kolo je **isključen**o zbog smanjenja potrošnje.
- **Pull-up i pull-down** otpornici su **isključeni**
- Očitavanje ulaznog registra uvek daje “0”



# GPIO registri

- Četiri 32-bitna ***konfiguraciona*** registra (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR)
- Dva 32-bitna ***data*** registra (GPIOx_IDR, GPIOx_ODR)
- Jedan  32-bitni ***set/reset*** registar (GPIOx_BSRR)
- Jedan 16-bitni ***reset*** registar (GPIOx_BRR)
- Jedan 32-bitni ***locking*** registar (GPIOx_LCKR)
- Dva 32-bitna ***alternate function*** selekciona registra (GPIOx_AFRH, GPIOx_AFRL)

# Port mode register (GPIOx_MODER) (x=A..I)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MODE15[1:0] | | MODE14[1:0] | | MODE13[1:0] | | MODE12[1:0] | | MODE11[1:0] | | MODE10[1:0] | | MODE9[1:0] | | MODE8[1:0] | |
| 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 |
| MODE7[1:0] | | MODE6[1:0] | | MODE5[1:0] | | MODE4[1:0] | | MODE3[1:0] | | MODE2[1:0] | | MODE1[1:0] | | MODE0[1:0] | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 2y+1:2y **MODEy[1:0]:** Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O mode.

00: Input mode

01: General purpose output mode

10: Alternate function mode

11: Analog mode (reset state)

# Port output type register (GPIOx_OTYPER) (x=A..I)

| 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 |
| OT15 | OT14 | OT13 | OT12 | OT11 | OT10 | OT9 | OT8 | OT7 | OT6 | OT5 | OT4 | OT3 | OT2 | OT1 | OT0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OTy:** Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O output type.

0: Output push-pull (reset state)

1: Output open-drain

# Port pull-up/pull-down register (GPIOx_PUPDR) (x=A..I)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PUPD15[1:0] | | PUPD14[1:0] | | PUPD13[1:0] | | PUPD12[1:0] | | PUPD11[1:0] | | PUPD10[1:0] | | PUPD9[1:0] | | PUPD8[1:0] | |
| 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 |
| PUPD7[1:0] | | PUPD6[1:0] | | PUPD5[1:0] | | PUPD4[1:0] | | PUPD3[1:0] | | PUPD2[1:0] | | PUPD1[1:0] | | PUPD0[1:0] | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 2y+1:2y **PUPDy[1:0]:** Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O pull-up or pull-down

00: No pull-up, pull-down

01: Pull-up

10: Pull-down

11: Reserved

# Port speed register (GPIOx_OSPEEDR) (x=A..I)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OSPEED15 [1:0] | | OSPEED14 [1:0] | | OSPEED13 [1:0] | | OSPEED12 [1:0] | | OSPEED11 [1:0] | | OSPEED10 [1:0] | | OSPEED9 [1:0] | | OSPEED8 [1:0] | |
| 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 |
| OSPEED7 [1:0] | | OSPEED6 [1:0] | | OSPEED5 [1:0] | | OSPEED4 [1:0] | | OSPEED3 [1:0] | | OSPEED2 [1:0] | | OSPEED1 [1:0] | | OSPEED0 [1:0] | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 2y+1:2y **OSPEEDy[1:0]**: Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O output speed.

00: Low speed

01: Medium speed

10: High speed

11: Very high speed

*Note: Refer to the device datasheet for the frequency specifications and the power supply and load conditions for each speed.*

# Port input data register (GPIOx_IDR) (x=A..I)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IDR15 | IDR14 | IDR13 | IDR12 | IDR11 | IDR10 | IDR9 | IDR8 | IDR7 | IDR6 | IDR5 | IDR4 | IDR3 | IDR2 | IDR1 | IDR0 |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bits 31:16 Reserved, always read as 0.

Bits 15:0 **IDRy[15:0]:** Port input data (y= 0 .. 15)

These bits are read only and can be accessed in Word mode only. They contain the input value of the corresponding I/O port.

# Port output data register (GPIOx_ODR) (x=A..I)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ODR15 | ODR14 | ODR13 | ODR12 | ODR11 | ODR10 | ODR9 | ODR8 | ODR7 | ODR6 | ODR5 | ODR4 | ODR3 | ODR2 | ODR1 | ODR0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:16 Reserved, always read as 0.

Bits 15:0 **ODRy[15:0]:** Port output data (y= 0 .. 15)

These bits can be read and written by software and can be accessed in Word mode only.

*Note: For atomic bit set/reset, the ODR bits can be individually set and cleared by writing to the GPIOx_BSRR register (x = A .. E).*

# Port bit set/reset register (GPIOx_BSRR) (x=A..G)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BR15 | BR14 | BR13 | BR12 | BR11 | BR10 | BR9 | BR8 | BR7 | BR6 | BR5 | BR4 | BR3 | BR2 | BR1 | BR0 |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BS15 | BS14 | BS13 | BS12 | BS11 | BS10 | BS9 | BS8 | BS7 | BS6 | BS5 | BS4 | BS3 | BS2 | BS1 | BS0 |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

Bits 31:16 **BRy:** Port x Reset *bit y (y= 0 .. 15)*

These bits are write-only and can be accessed in Word mode only.

0: No action on the corresponding ODRx bit

1: Reset the corresponding ODRx bit

*Note: If both BSx and BRx are set, BSx has priority.*

Bits 15:0 **BSy:** Port x *Set bit y (y= 0 .. 15)*

These bits are write-only and can be accessed in Word mode only.

0: No action on the corresponding ODRx bit

1: Set the corresponding ODRx bit

# Port bit reset register (GPIOx_BRR) (x=A..G)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BR15 | BR14 | BR13 | BR12 | BR11 | BR10 | BR9 | BR8 | BR7 | BR6 | BR5 | BR4 | BR3 | BR2 | BR1 | BR0 |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

Bits 31:16 Reserved

Bits 15:0 **BRy:** Port x Reset bit y (y= 0 .. 15)

These bits are write-only and can be accessed in Wor

0: No action on the corresponding ODRx bit

1: Reset the corresponding ODRx bit

Da li se stanje pina u mbed-u postavlja upisom u ODR ili BSRR/BRR?

Izvršiti toggle-ovanje LE diode u mbed-u direktnim upisivanjem u BSR/BR registar.

# MBED – direktan upis u registre

```
#include "mbed.h"
// This program will blink LED2 using register names

DigitalOut myled1(LED2);

int main() {
    int value = 0;
    uint32_t myled2_bit_mask=0;

    while (1) {
        //control LED2 using C/C++ hardware I/O register names
        //from STM32L476RG reference manual the LED2 pin goes to GPIOA port bit 5
        myled2_bit_mask = 0x00000020; // 0x00000020 = 1<<5 all "0"s with a "1" in bit 5
        if (value==0) {
            GPIOA->BRR = GPIOA->BRR | myled2_bit_mask; //Write to register that clears bits
        } else {
            GPIOA->BSRR = GPIOA->BSRR | myled2_bit_mask; //Write to register that sets bits
        }
        // flip value and wait
        value = !value;
        wait(0.5);
    }
}
```

Obezbediti istu funkcionalnost menjajući sadržaj samo BSR registra.

# Port configuration lock register (GPIOx_LCKR) (x=A..G)

*Note:* *A specific write sequence is used to write to the GPIOx_LCKR register. Only word access (32-bit long) is allowed during this locking sequence.*

Each lock bit freezes a specific configuration register (control and alternate function registers).

Address offset: 0x1C

Reset value: 0x0000 0000

| 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. | LCKK |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 |
|---|---|---|---|---|---|---|---|---|
| LCK15 | LCK14 | LCK13 | LCK12 | LCK11 | LCK10 | LCK9 | LCK8 | LCK7 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **LCKK:** Lock key

This bit can be read any time. It can only be modified using the lock key write sequence.

0: Port configuration lock key not active

1: Port configuration lock key active. The GPIOx_LCKR register is locked until the next MCU reset or peripheral reset.

LOCK key write sequence:

WR LCKR[16] = '1' + LCKR[15:0]

WR LCKR[16] = '0' + LCKR[15:0]

WR LCKR[16] = '1' + LCKR[15:0]

RD LCKR

RD LCKR[16] = '1' (this read operation is optional but it confirms that the lock is active)

*Note:* *During the LOCK key write sequence, the value of LCK[15:0] must not change.*

*Any error in the lock sequence aborts the lock.*

*After the first lock sequence on any bit of the port, any read access on the LCKK bit will return '1' until the next MCU reset or peripheral reset.*

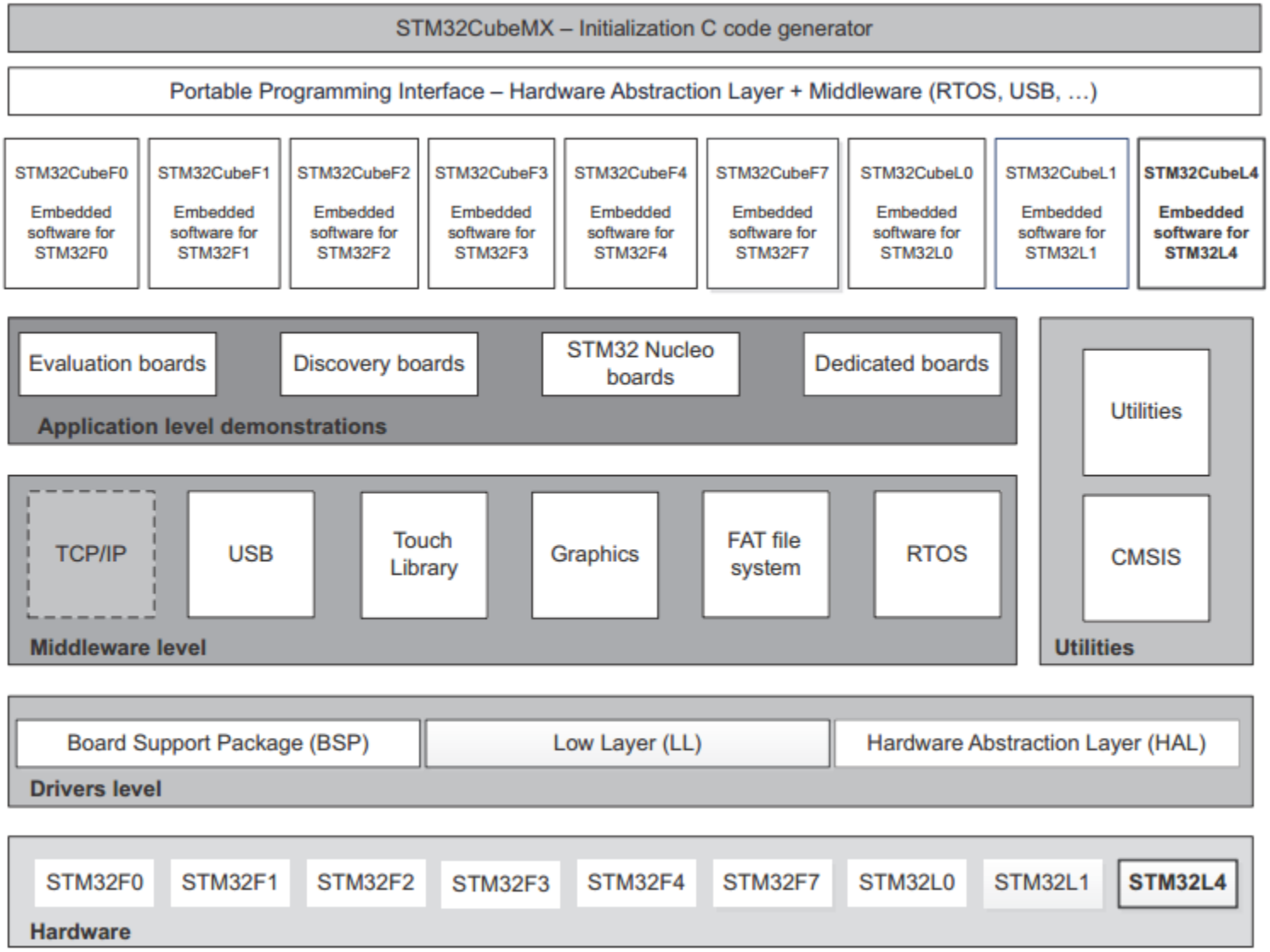
Bits 15:0 **LCKy:** Port x lock bit y (y= 0..15)

These bits are read/write but can only be written when the LCKK bit is '0.

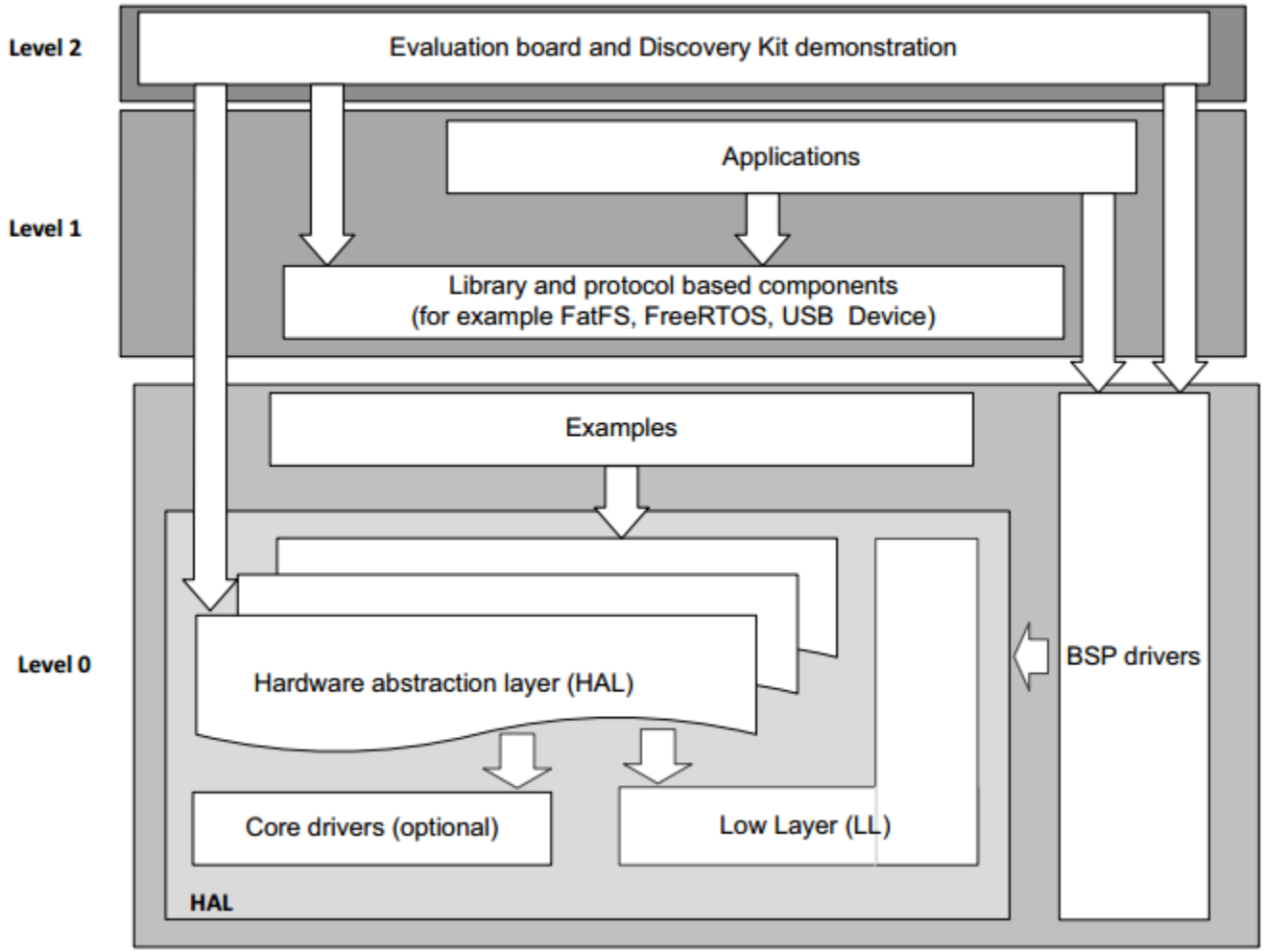
0: Port configuration not locked

1: Port configuration locked

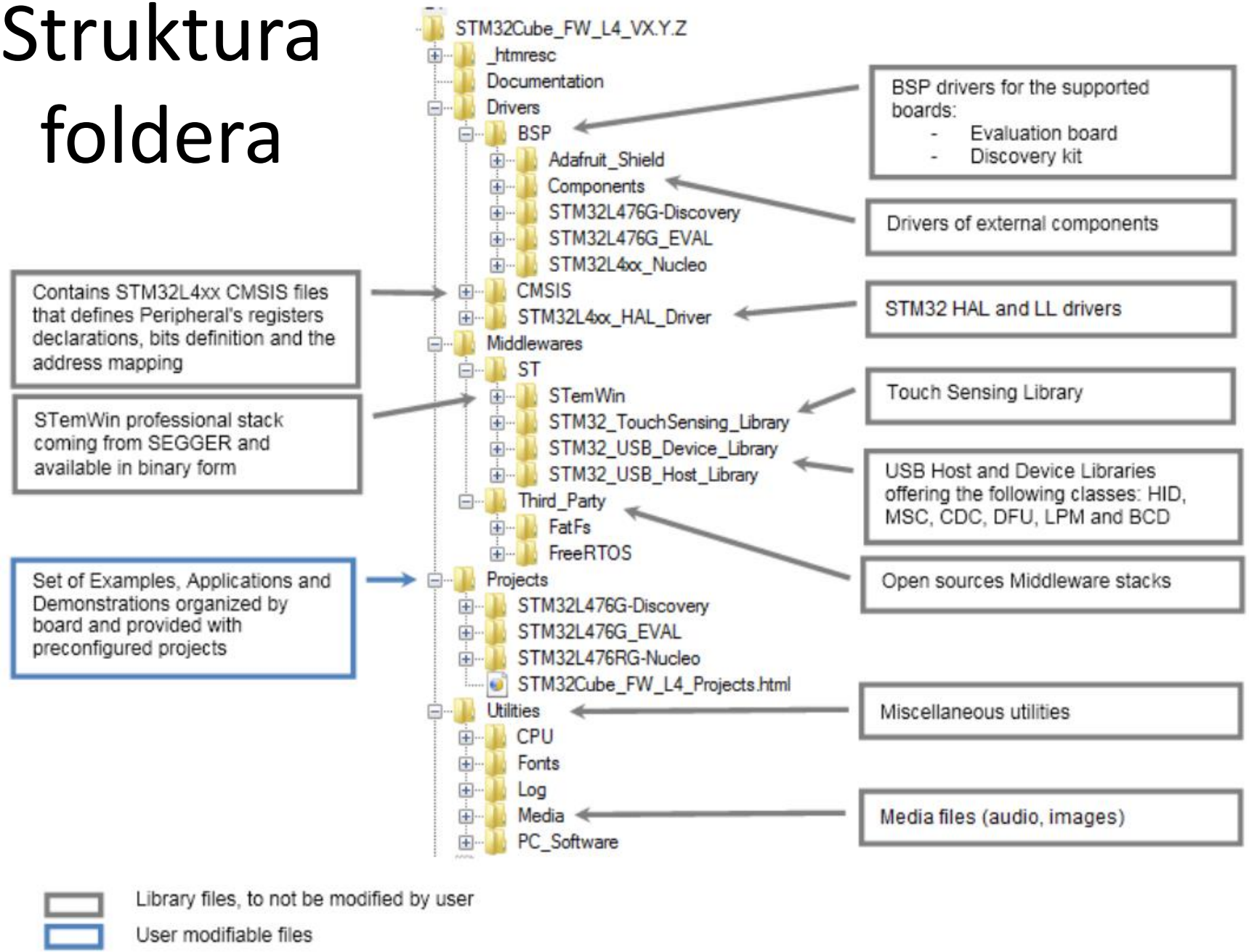
# STM32CubeL4 komponente



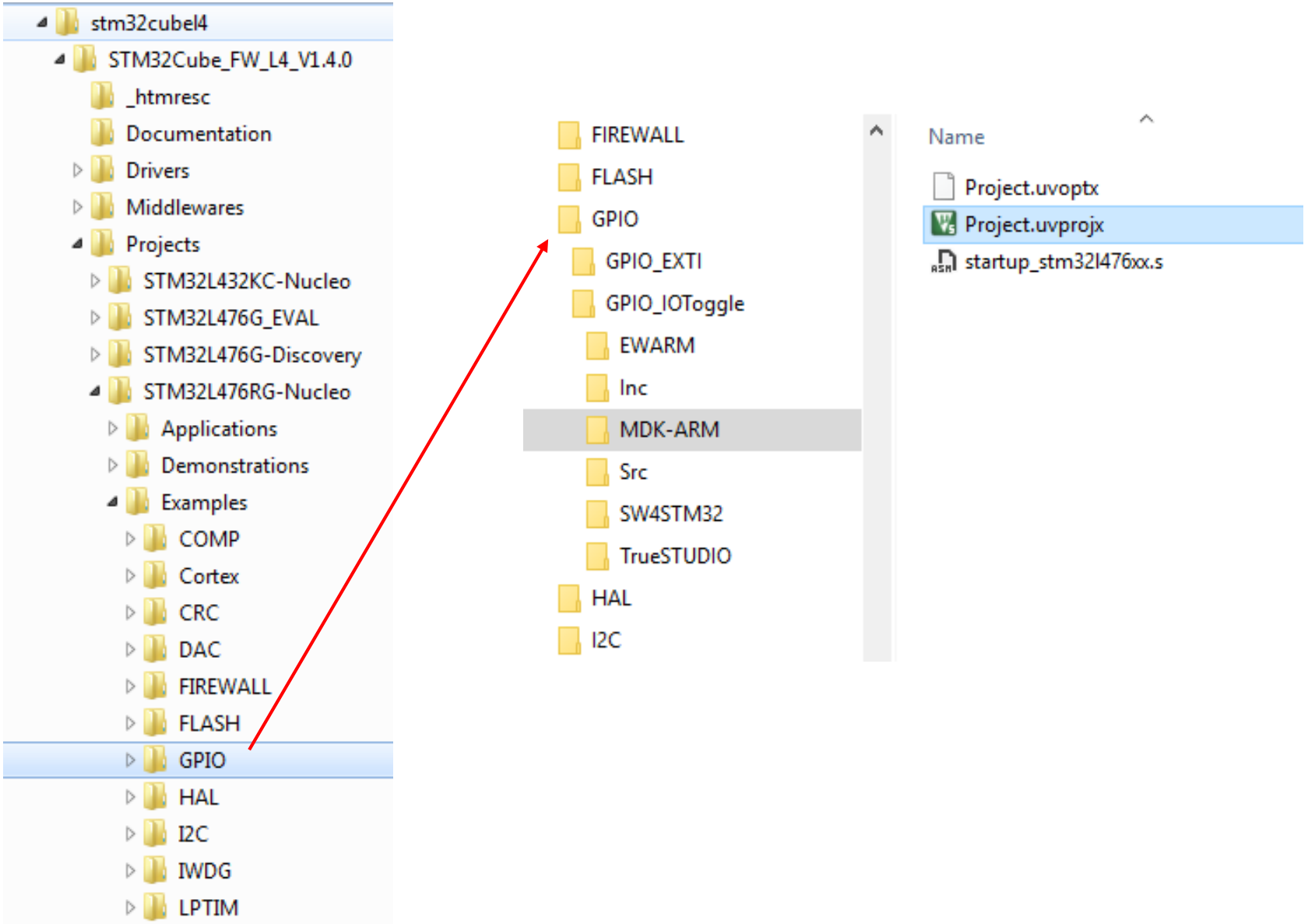
# STM32CubeL4 arhitektura



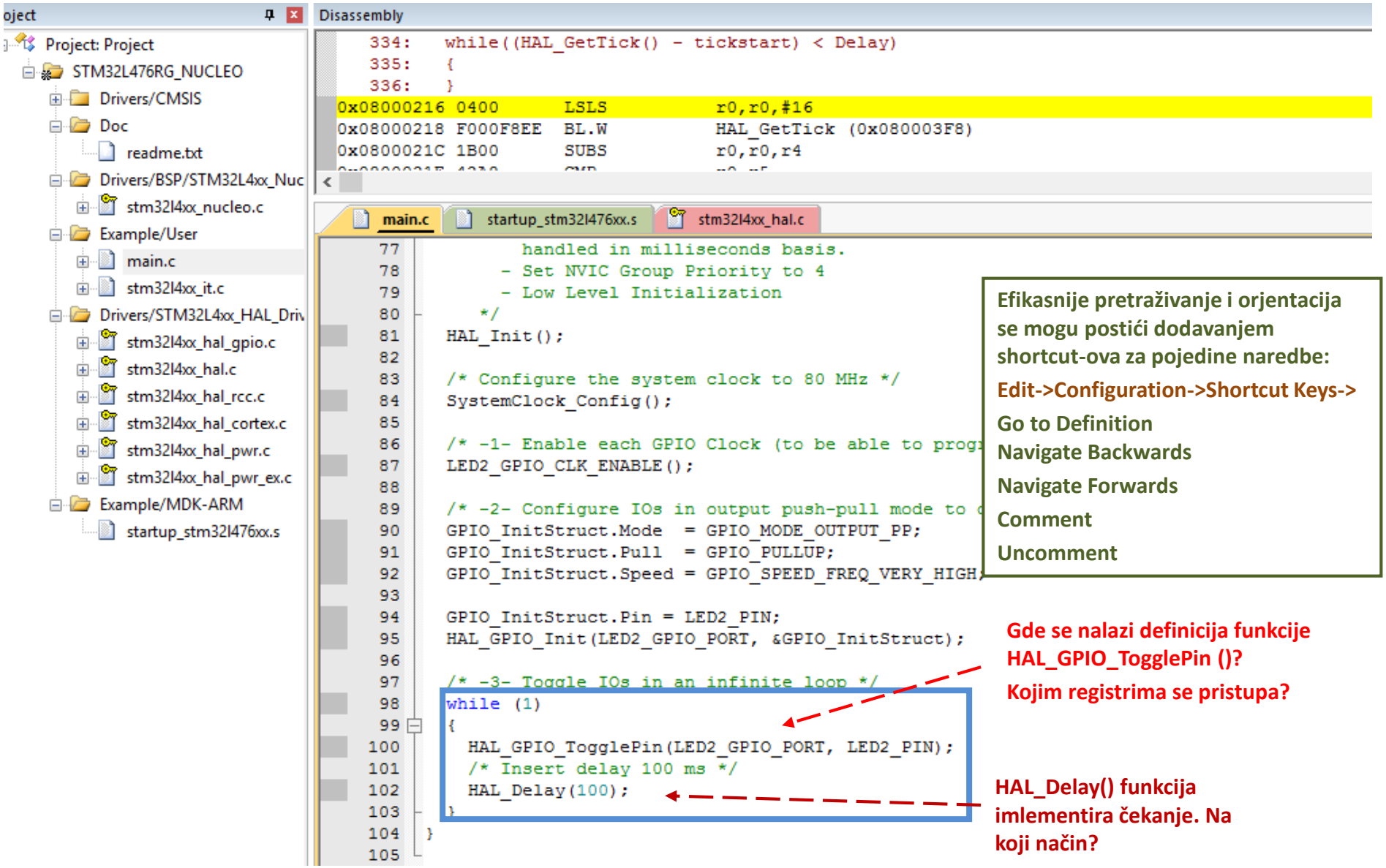
# Struktura foldera



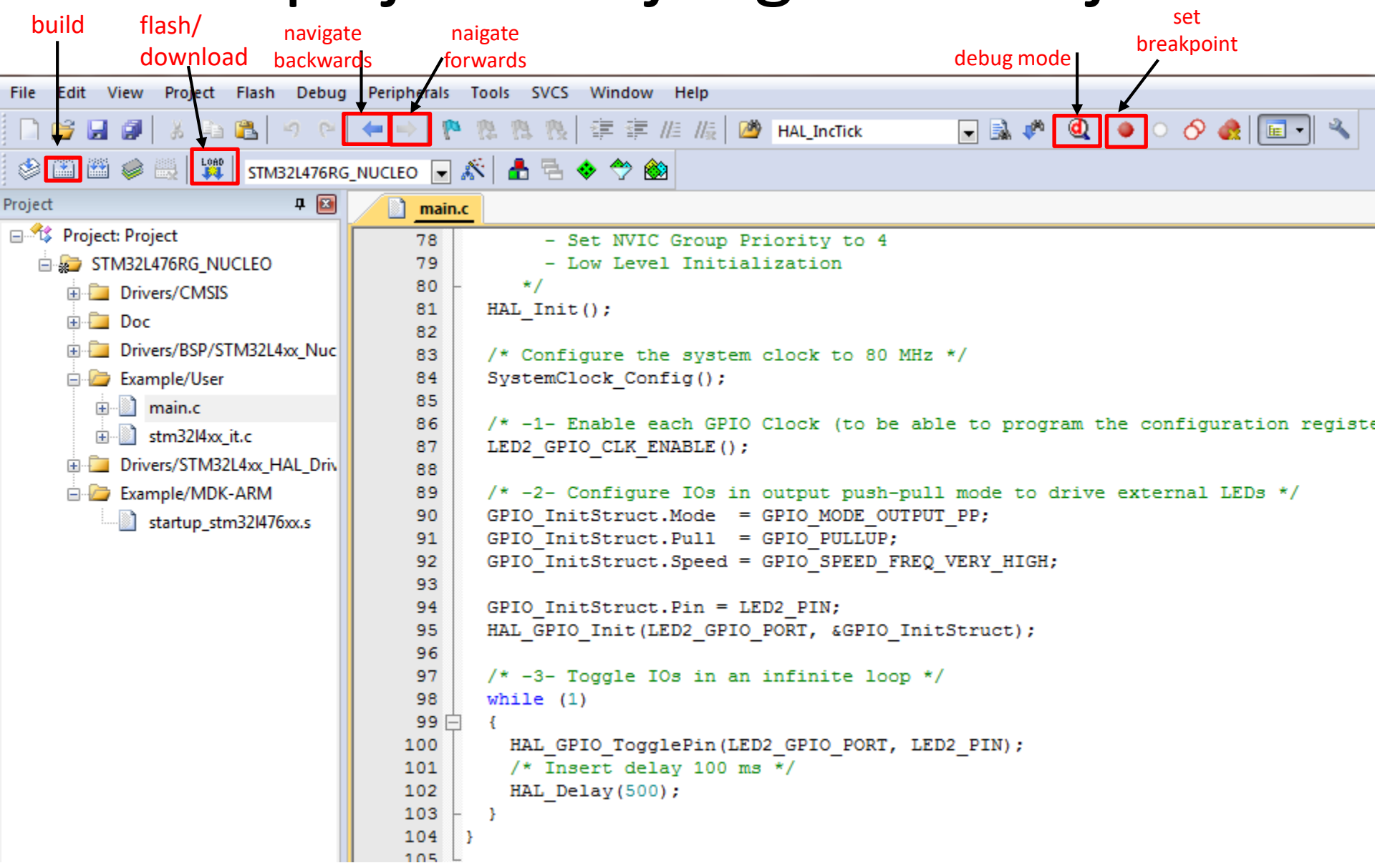
# Primeri – GPIO IOToggle



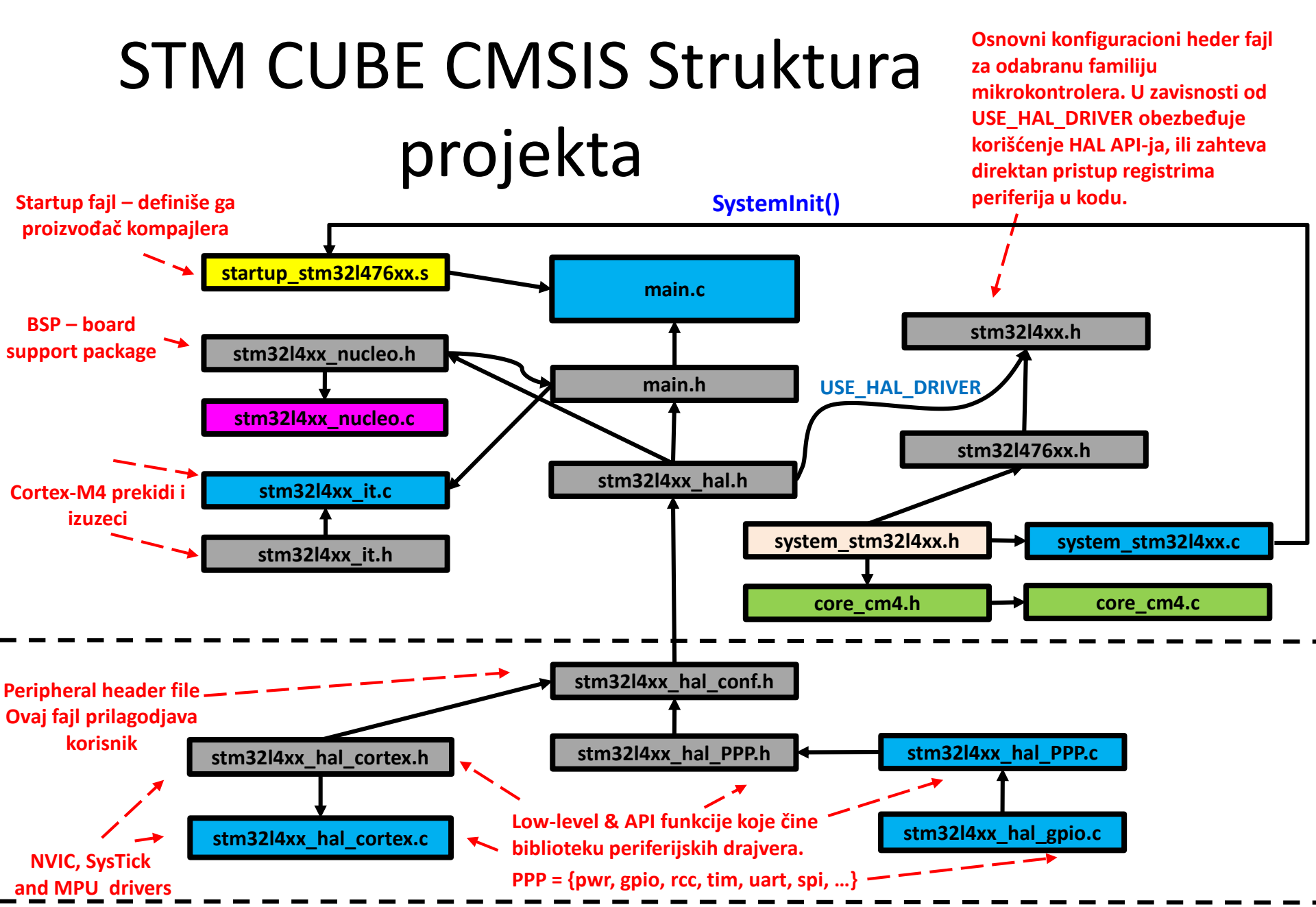
# Projekat GPIO_IOToggle



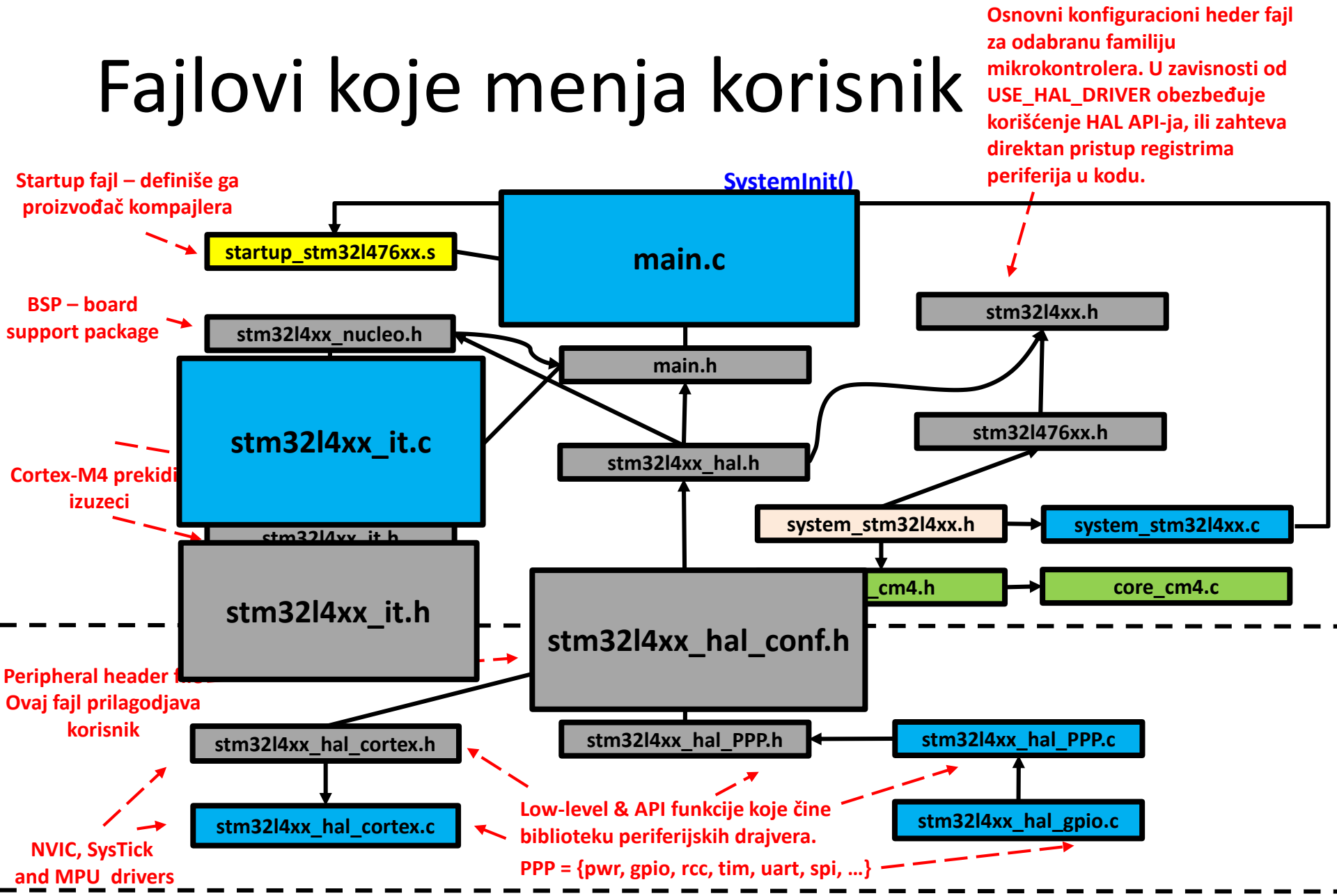
# Opcije razvojnog okruženja



# STM CUBE CMSIS Struktura projekta



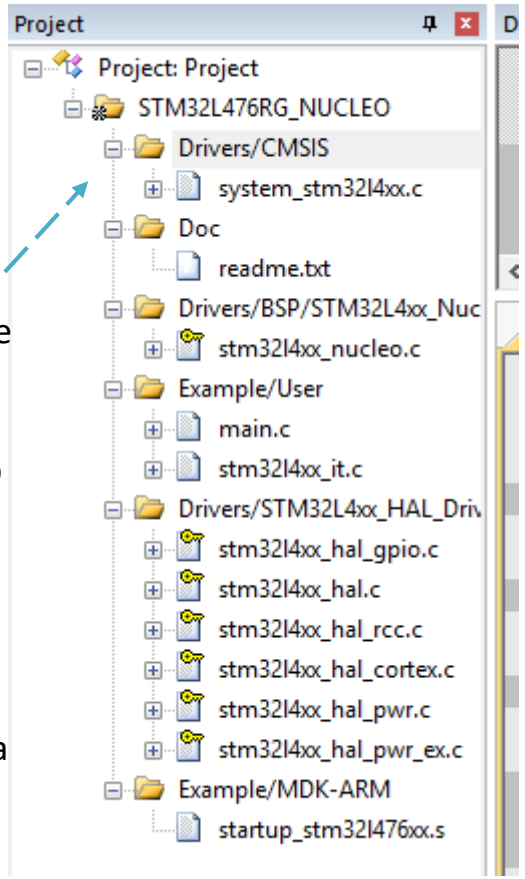
# Fajlovi koje menja korisnik



# CMSIS - fajlovi

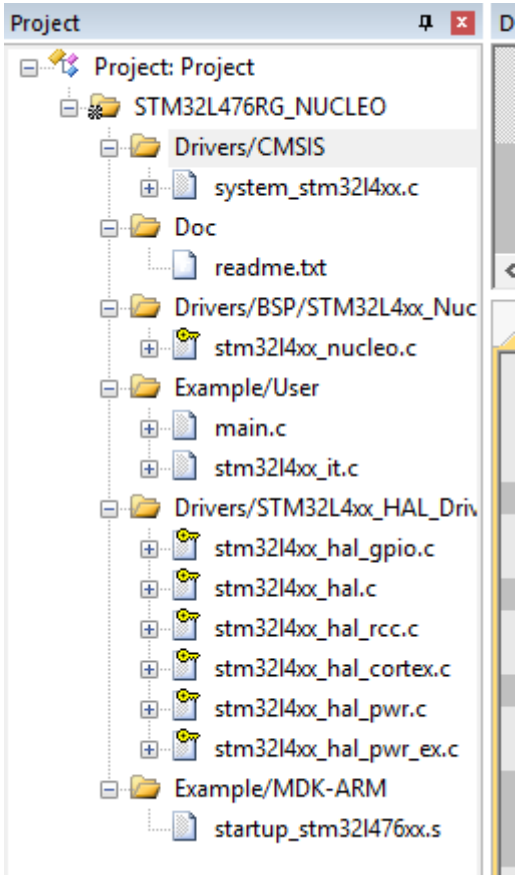
- **Fajlovi koje definiše isključivo ARM:**
- core_cm4.c - *Core Peripheral Access Layer Source File*
  - Ovaj fajl sadži implementaciju CMSIS *intrinsic* funkcija koje su za razliku od *intrinsic* funkcija u oviru C kompajlera nezavisne od vrste kompajlera. stm32l4xx_hal_cortex sadrži dummy f-je koji pozivaju one definisane u core_cm4.c fajlu – NVIC, MPU, SysTick
- core_cm4.h - *Core Peripheral Access Layer Header File*
  - Ovaj fajl sadrži definicije registara periferija jezgra i funkcije za pristup tim registrima, kao i deklaraciju *intrinsic funkcija*

- **CMSIS layer *Device Peripheral Access Layer* čine sledeći fajlovi:**
- system_stm32l4xx.c - *Device Peripheral Access Layer Source File*
  - *O*vaj fajl sadži definiciju funkcije *SystemInit* koja vrši inicijalizaciju dela mikrokontrolera zaduženog za generisanje klok signala i koja se poziva u okviru *startup koda*
- system_stm32l4xx.h - *Device Peripheral Access Layer Header File*
  - Ovaj fajl sadži deklaraciju funkcije *SystemInit*
- stm32l4xx.h - *Device Peripheral Access Layer Header File*
  - Ovaj fajl sadrži definiciju registara periferija mikrokontrolera kao i definiciju bitova u okviru svakog definisanog registra, bazne adrese registara i numeraciju vektora prekida



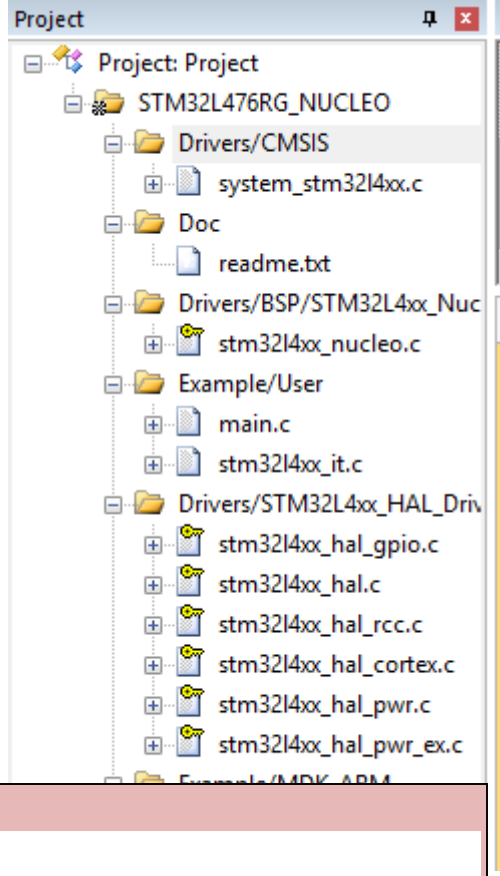
# CMSIS - fajlovi

- stm32l4xx_hal_conf.h - *Device Peripheral Access Layer Configuration Header File*
  - Ovaj fajl kontroliše korišćenje periferijskih biblioteka koje definiše isključivo proizvođač mikrokontrolera.
- stm32l4xx_hal_cortex.c stm32l4xx_hal_cortex.h
  - Ovi fajlovi definišu specifičnosti kontrole prekida, sistemskog takta i reset-a. Formu propisuje ARM, ali ga neznatno redefiniše svaki proizvođač.
- stm32l4xx_hal_PPP.x
  - Ovi fajlovi predstavljaju drajvere za odgovarajuće periferije i razlikuju se od proizvođača do proizvođača.




- **Specifičnosti različitih razvojnih okruženja se definišu sledećim fajlom:**
- startup_stm32l476xx.s - *STM32L476xx Devices vector table for MDK-ARM toolchain.*
  - Ovaj fajl sadrži vektor tabelu i *Reset_Handler* ISR u kojoj se vrši setovanje SP, inicijalizacija dela mikrokontrolera zaduženog za generisanje klok signala i pozicioniranje PC na labelu __iar_program_start gde se nalazi kod za inicijalizaciju promenljivih, nakon čega se poziva *main f*unkcija
  - Napomena: *Ovaj fajl se po pravilu razlikuje od okruženja do okruženja!*

# Fajlovi koje menja korisnik




**stm32l4xx_hal_conf.h**

```
57  #define HAL_CORTEX_MODULE_ENABLED
58  /* #define HAL_CRC_MODULE_ENABLED */
59  /* #define HAL_DAC_MODULE_ENABLED */
60  /* #define HAL_DMA_MODULE_ENABLED */
61  /* #define HAL_ETH_MODULE_ENABLED */
62  #define HAL_FLASH_MODULE_ENABLED
63  #define HAL_GPIO_MODULE_ENABLED
64  /* #define HAL_HCD_MODULE_ENABLED */
```

**main_X.c**

```
102   while (1)
103   {
104     HAL_GPIO_TogglePin(LED3_GPIO_PORT, LED3_PIN);
105     /* Insert delay 100 ms */
106     HAL_Delay(100);
107     HAL_GPIO_TogglePin(LED4_GPIO_PORT, LED4_PIN);
108     /* Insert delay 100 ms */
109     HAL_Delay(100);
110   }
```

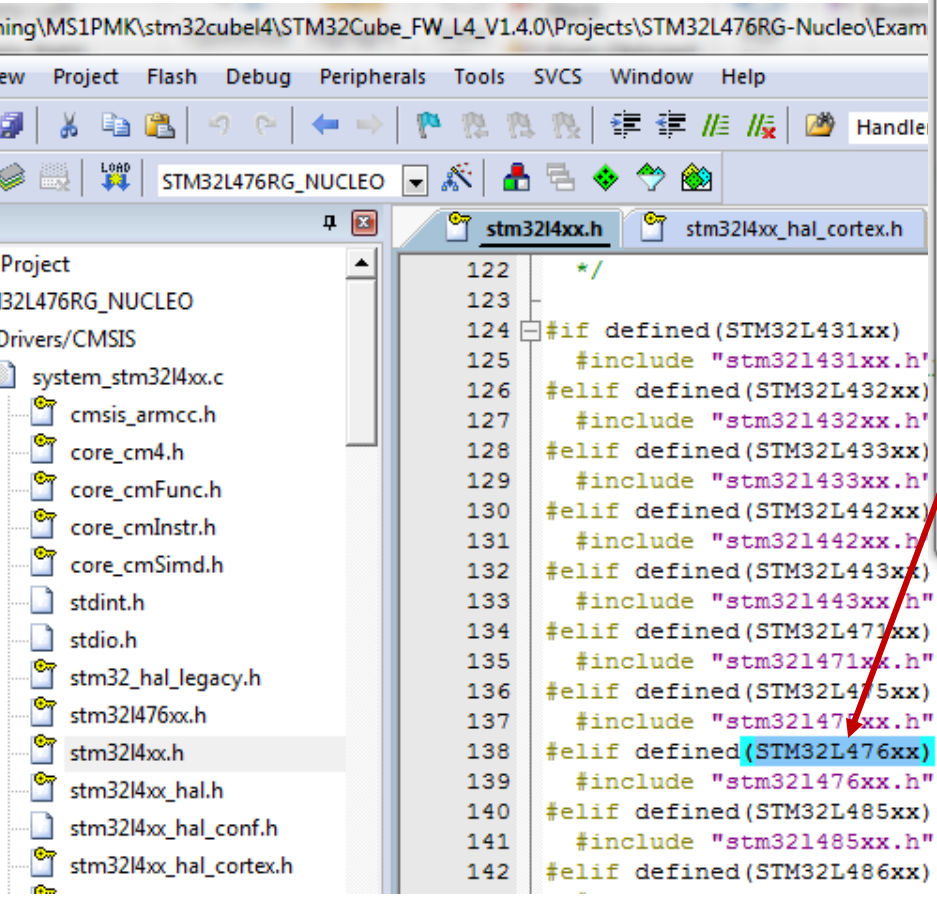
**stm32l4xx_it.c**

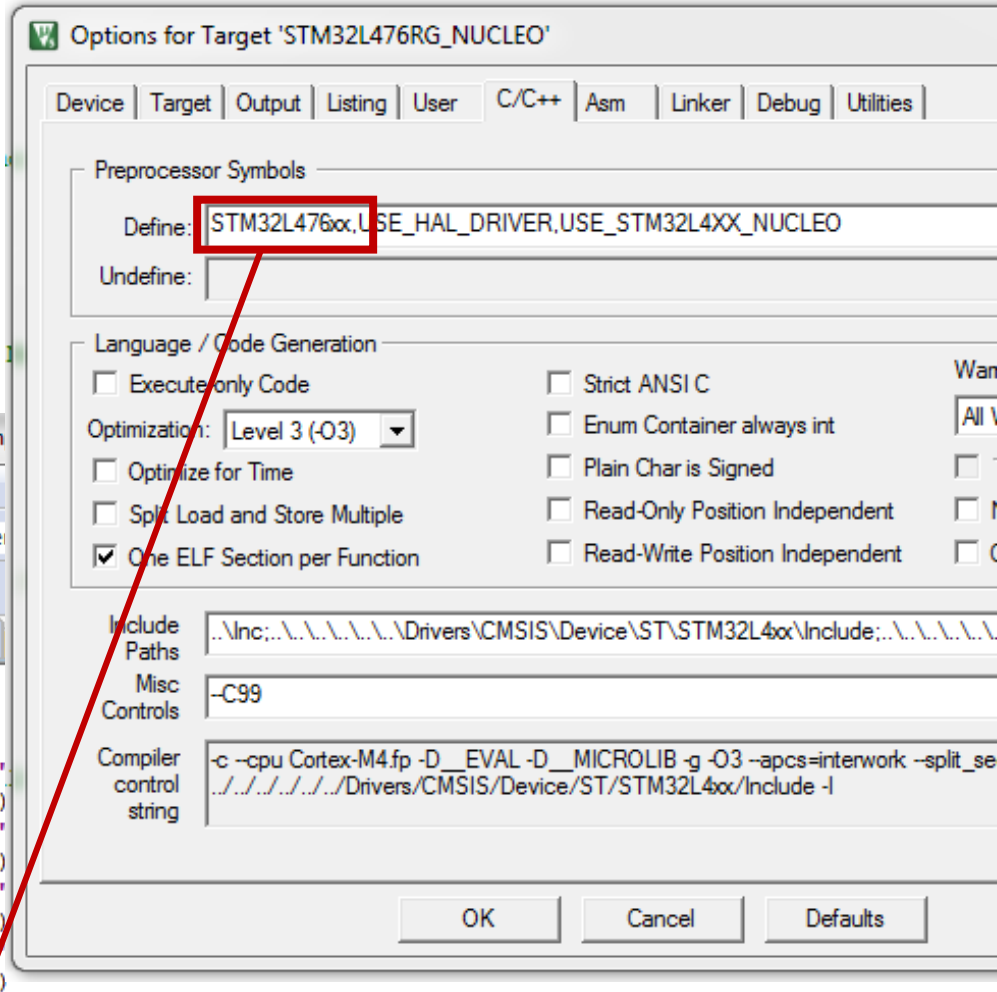
```
152 /**
153   * @brief  This function handles SysTick Handler.
154   * @param  None
155   * @retval None
156   */
157 void SysTick_Handler(void)
158 {
159   HAL_IncTick();
160 }
161
```

**stm32l4xx_it.h**

```
57  void SVC_Handler(void);
58  void DebugMon_Handler(void);
59  void PendSV_Handler(void);
60  void SysTick_Handler(void);
```

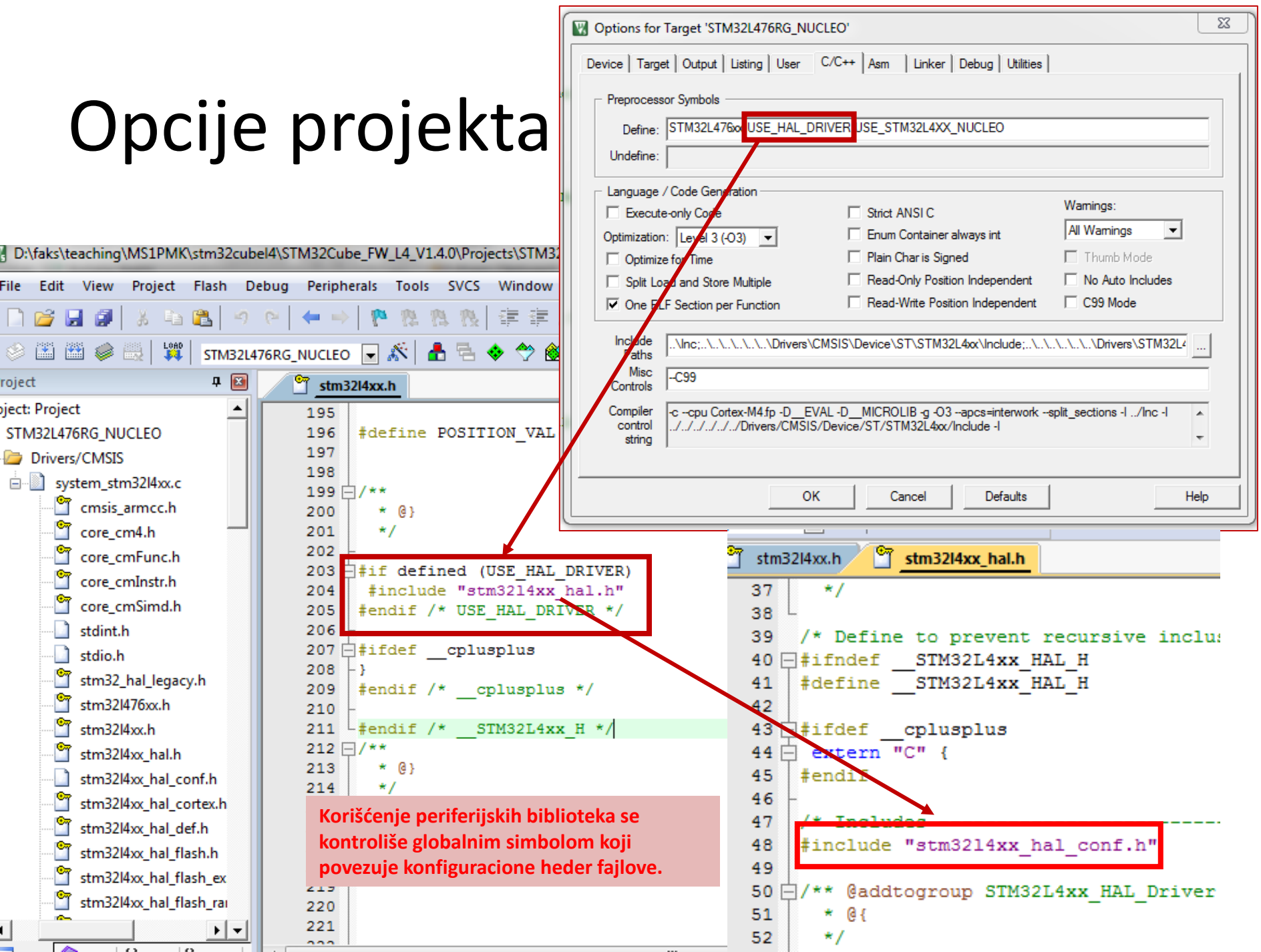
# Opcije projekta



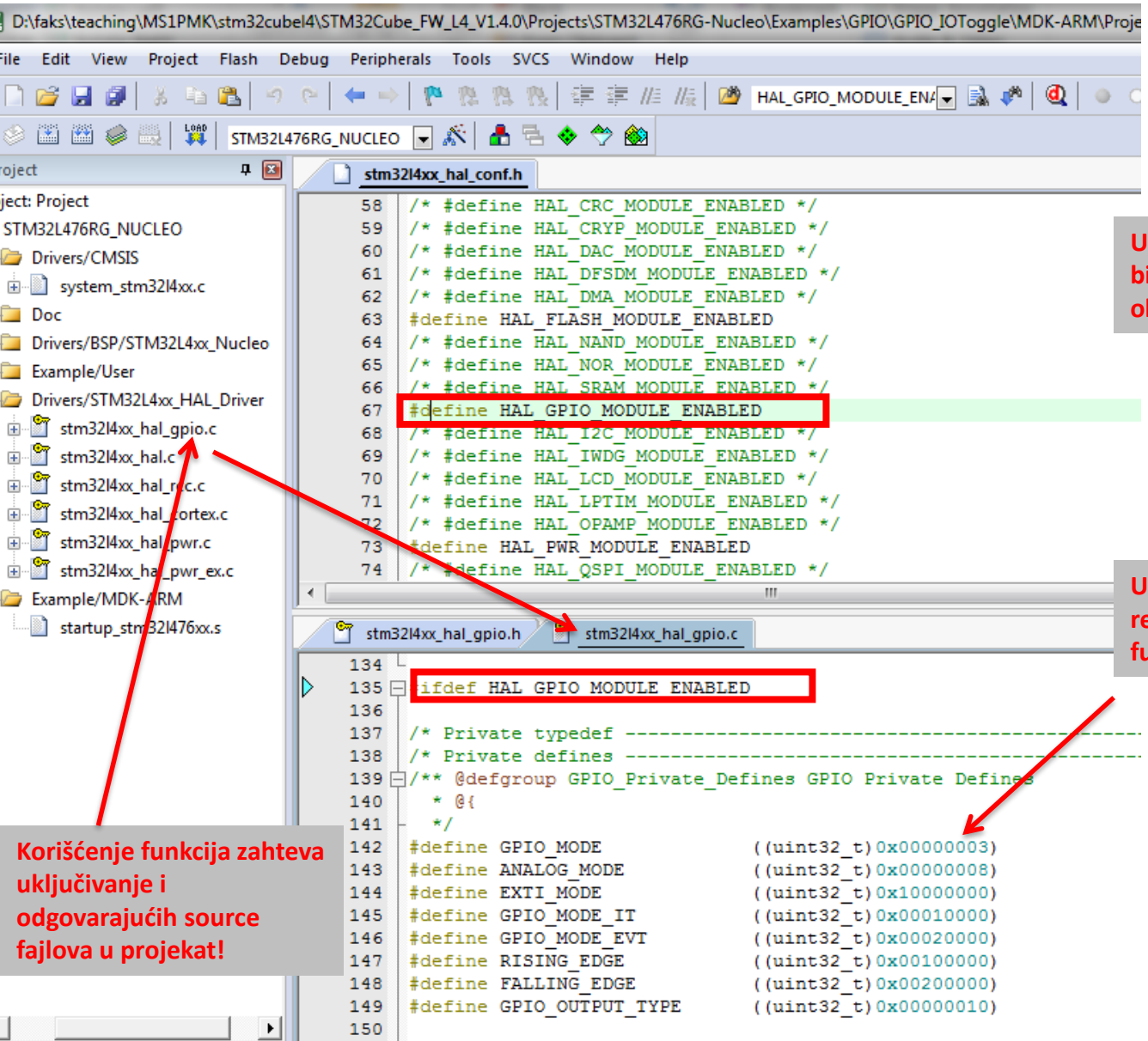




**Konretan mikrokontroler se definiše preko globalnog simbola što utiče na deklaracije u glavnom heder fajlu**

# Opcije projekta



# Periferijske biblioteke



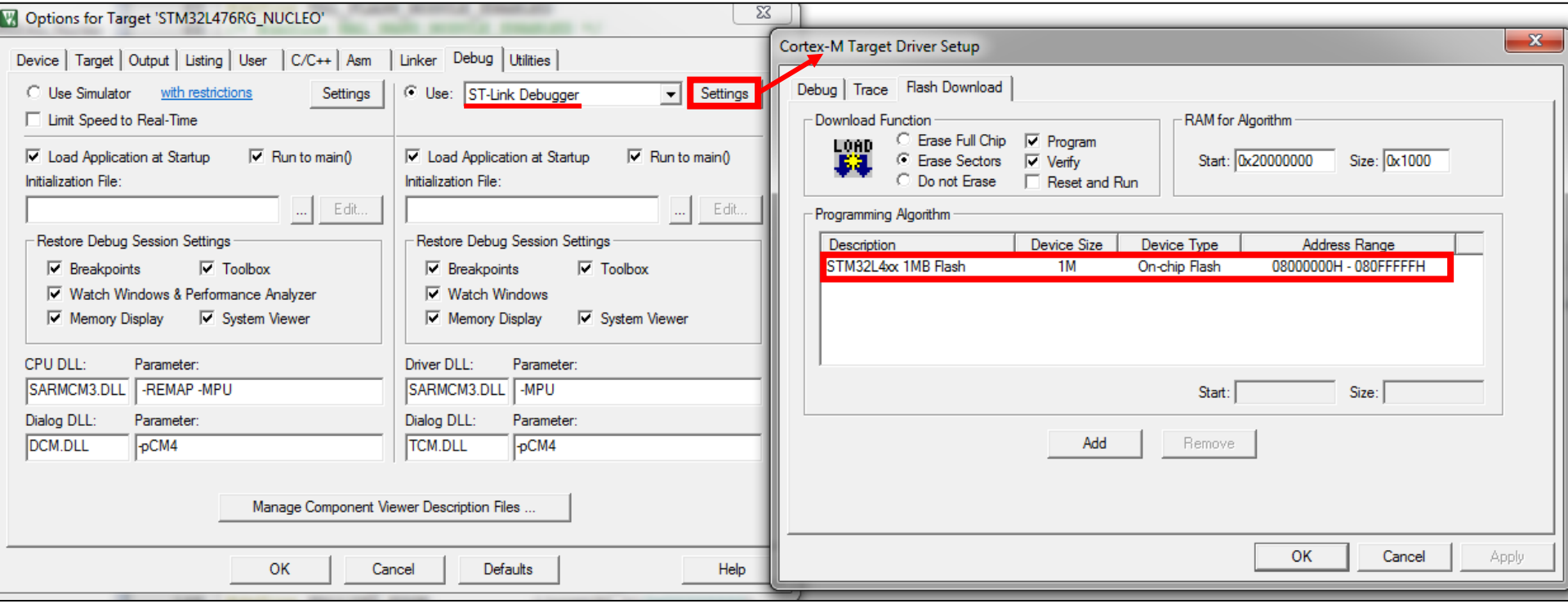

**U konfiguracionom fajlu se uključuju biblioteke vendor-specific periferija koje obezbeđuje proizvođač mikrokontrolera.**

**U tim hederima se nalaze deklaracije registara, ali i prototipovi specifičnih funkcija za pristup konkretnoj periferiji.**

**Korišćenje funkcija zahteva uključivanje i odgovarajućih source fajlova u projekat!**

# Opcije projekta - debug

- Programming Algorithm - ovaj algoritam definiše proizvođač kompajlera (KEIL).
- Bez ovog algoritma neće biti moguće spustiti program na uC.



# Generisanje takta

## Sistemski takt:

Four different clock sources can be used to drive the system clock (SYSCLK):

- HSI16 (high speed internal)16 MHz RC oscillator clock
- MSI (multispeed internal) RC oscillator clock
- HSE oscillator clock, from 4 to 48 MHz
- PLL clock

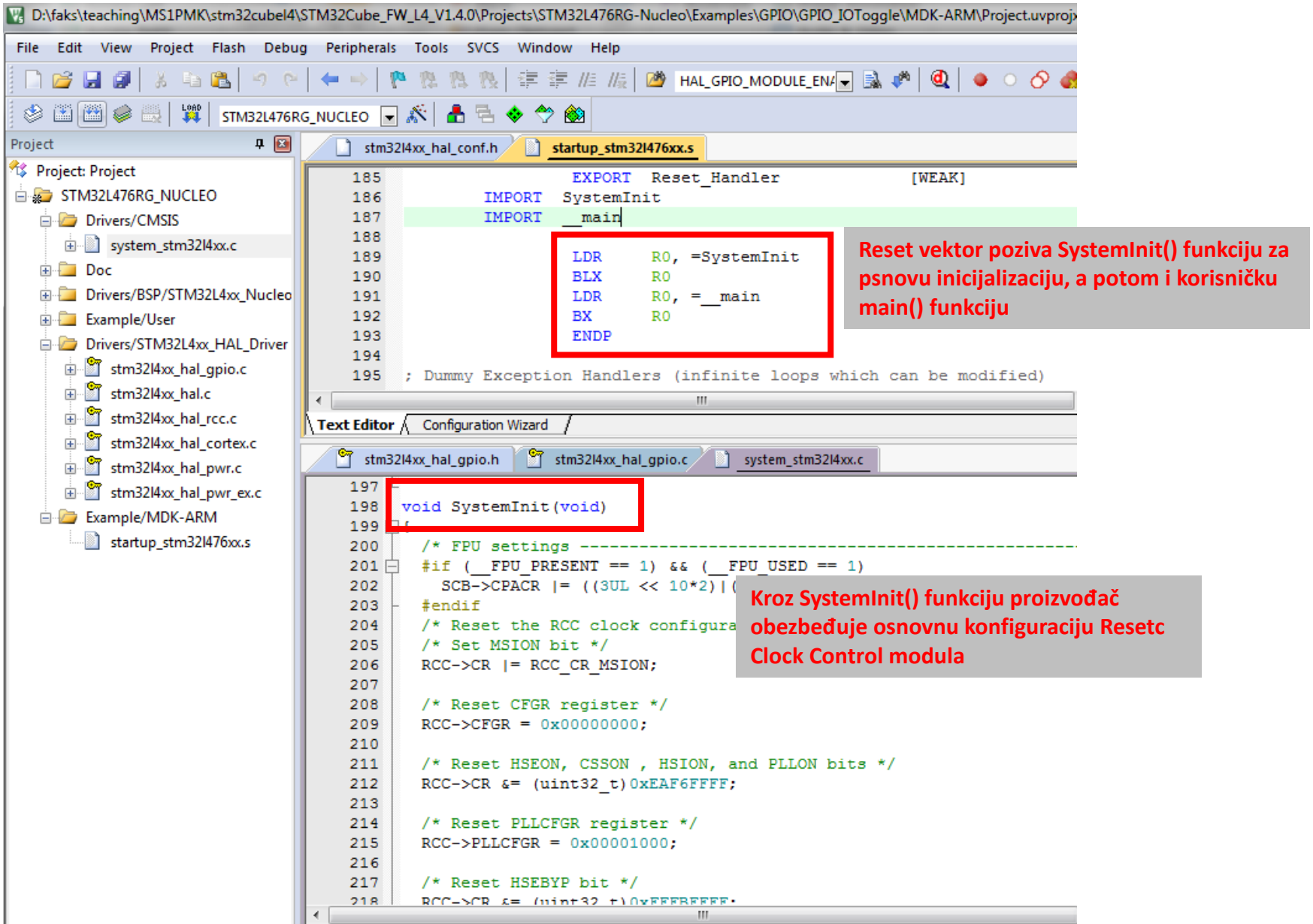
## Dodatni taktovi:

The devices have the following additional clock sources:

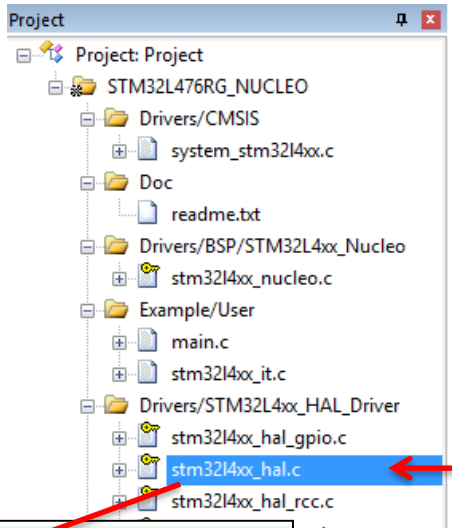
- 32 kHz low speed internal RC (LSI RC) which drives the independent watchdog and optionally the RTC used for Auto-wakeup from Stop and Standby modes.
- 32.768 kHz low speed external crystal (LSE crystal) which optionally drives the realtime clock (RTCCLK).
- RC 48 MHz internal clock sources (HSI48) to potentially drive the USB FS, the SDMMC and the RNG (only for STM32L496xx/4A6xx devices).

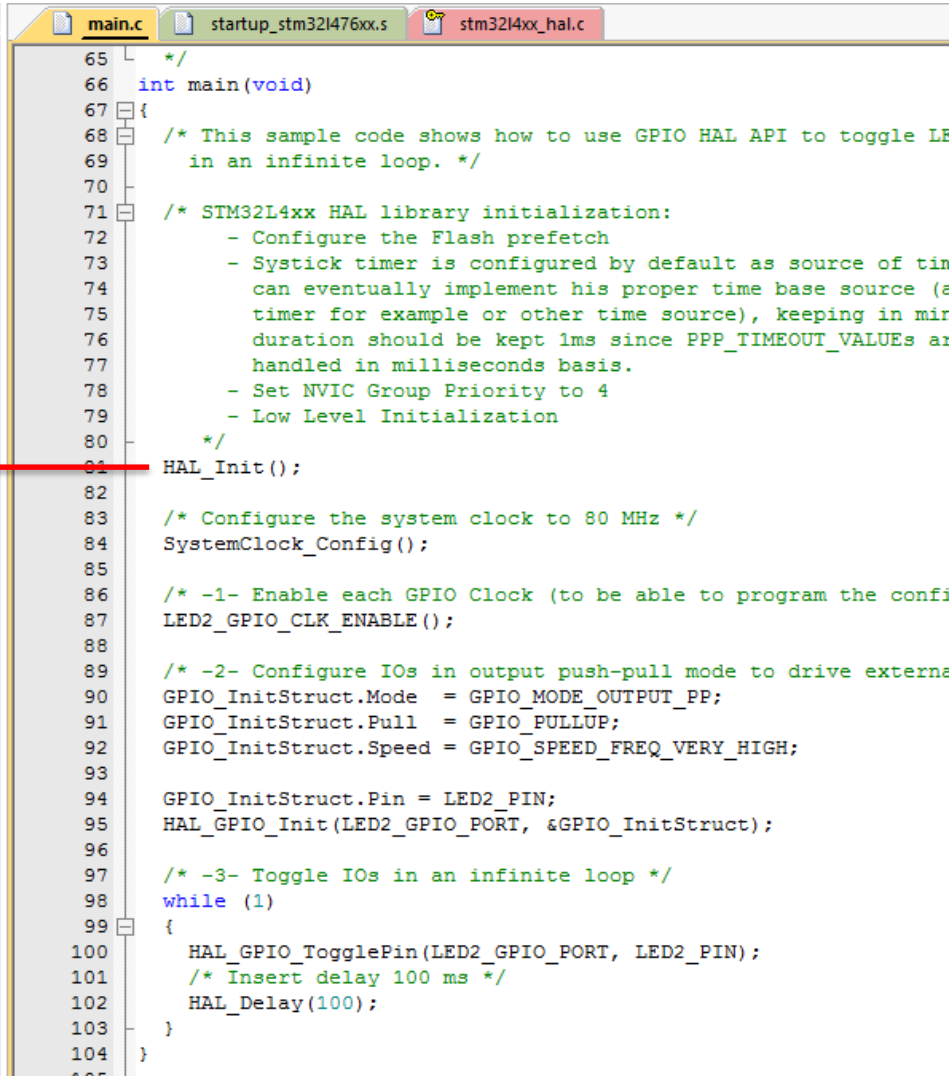
- ❑ Each clock source can be switched on or off independently when it is not used, to optimize power consumption.
- ❑ Several prescalers can be used to configure the AHB frequency, the high speed APB (APB2) and the low speed APB (APB1) domains. The maximum frequency of the AHB, the APB1 and the APB2 domains is 80 MHz.

# SystemInit() funkcija



# Tipičan main()

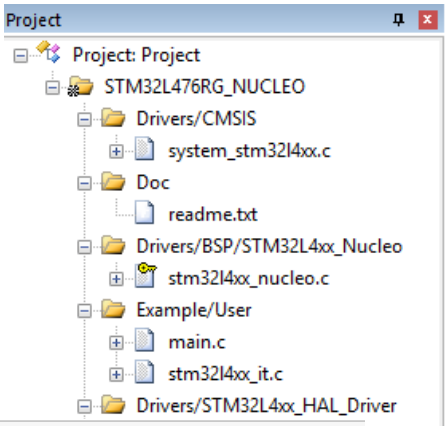



```
HAL_StatusTypeDef HAL_Init(void)
{
  /* Configure Flash prefetch */
#if (PREFETCH_ENABLE != 0)
#if defined(STM32F101x6) || defined(STM32F101xB) || d
    defined(STM32F102x6) || defined(STM32F102xB) || \
    defined(STM32F103x6) || defined(STM32F103xB) || d
    defined(STM32F105xC) || defined(STM32F107xC)

  /* Prefetch buffer is not available on value line d
  __HAL_FLASH_PREFETCH_BUFFER_ENABLE();
#endif
#endif /* PREFETCH_ENABLE */

  /* Set Interrupt Group Priority */
  HAL_NVIC_SetPriorityGrouping(NVIC_PRIORITYGROUP_4);

  /* Use systick as time base source and configure 1m
  HAL_InitTick(TICK_INT_PRIORITY);

  /* Init the low level hardware */
  HAL_MspInit();

  /* Return function statu
  return HAL_OK;
}
```

**Ovo je praktično inicijalizacija Cortex Periferija**

# Tipičan main()




```c
/**
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};

  /* MSI is enabled after System reset, activate PLL with MSI as source
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_MSI;
  RCC_OscInitStruct.MSIState = RCC_MSI_ON;
  RCC_OscInitStruct.MSIClockRange = RCC_MSIRANGE_6;
  RCC_OscInitStruct.MSICalibrationValue = RCC_MSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_MSI;
  RCC_OscInitStruct.PLL.PLLM = 1;
  RCC_OscInitStruct.PLL.PLLN = 40;
  RCC_OscInitStruct.PLL.PLLR = 2;
  RCC_OscInitStruct.PLL.PLLP = 7;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  if(HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    /* Initialization Error */
    while(1);
  }

  /* Select PLL as system clock source and configure the HCLK, PCLK1 ar
     clocks dividers */
  RCC_ClkInitStruct.ClockType = (RCC_CLOCKTYPE_SYSCLK | RCC_CLOCKTYPE_F
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
  if(HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_4) != HAL_OK
  {
    /* Initialization Error */
    while(1);
  }
```

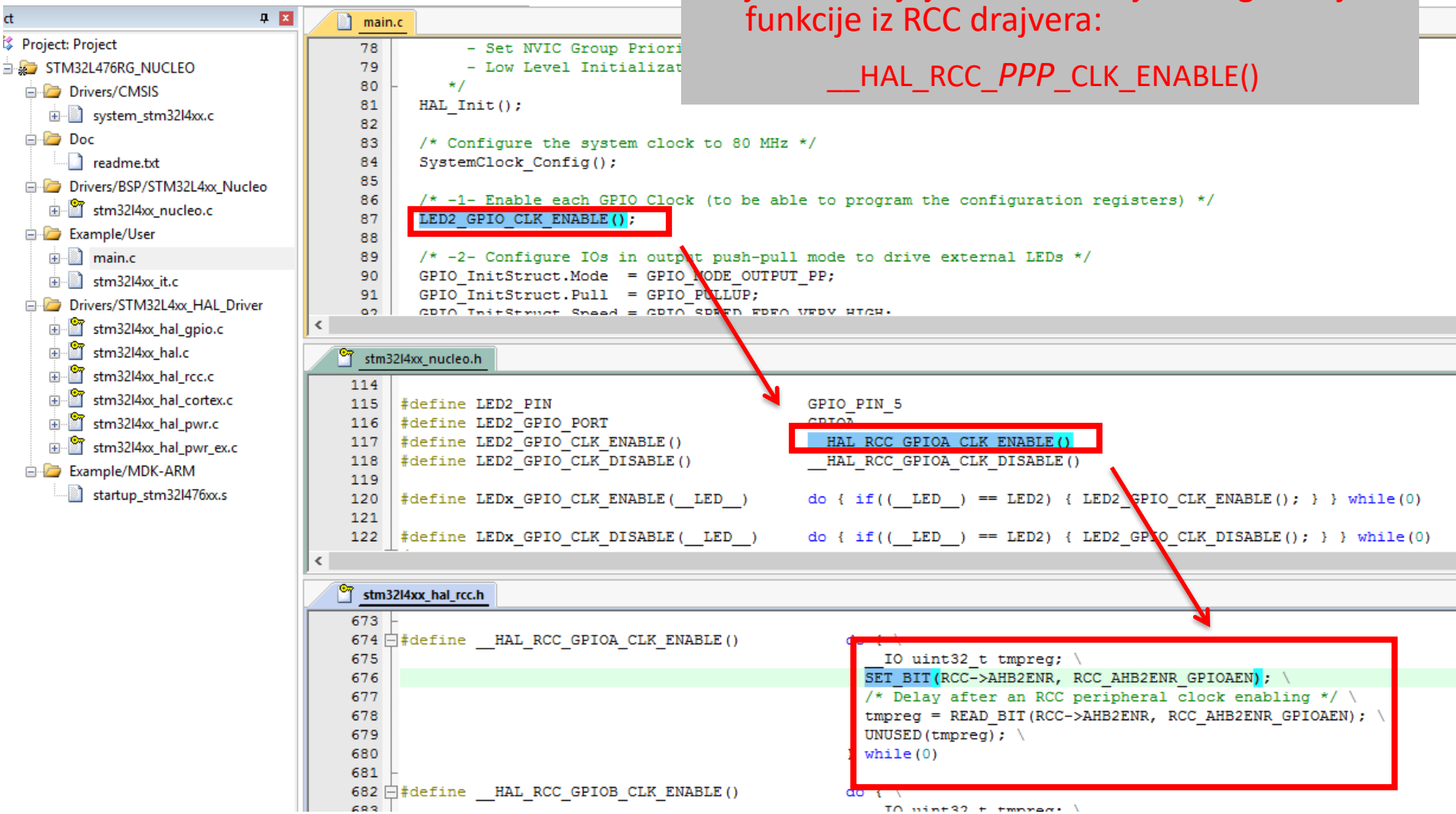
Definiše korisnik

```c
*/
int main(void)
{
  /* This sample code shows how to use GPIO HAL API to toggle LE
    in an infinite loop. */

  /* STM32L4xx HAL library initialization:
       - Configure the Flash prefetch
       - Systick timer is configured by default as source of tim
         can eventually implement his proper time base source (a
         timer for example or other time source), keeping in min
         duration should be kept 1ms since PPP_TIMEOUT_VALUEs ar
         handled in milliseconds basis.
       - Set NVIC Group Priority to 4
       - Low Level Initialization
     */
  HAL_Init();

  /* Configure the system clock to 80 MHz */
  SystemClock_Config();

  /* -1- Enable each GPIO Clock (to be able to program the confi
  LED2_GPIO_CLK_ENABLE();

  /* -2- Configure IOs in output push-pull mode to drive externa
  GPIO_InitStruct.Mode  = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull  = GPIO_PULLUP;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;

  GPIO_InitStruct.Pin = LED2_PIN;
  HAL_GPIO_Init(LED2_GPIO_PORT, &GPIO_InitStruct);

  /* -3- Toggle IOs in an infinite loop */
  while (1)
  {
    HAL_GPIO_TogglePin(LED2_GPIO_PORT, LED2_PIN);
    /* Insert delay 100 ms */
    HAL_Delay(100);
  }
}
```

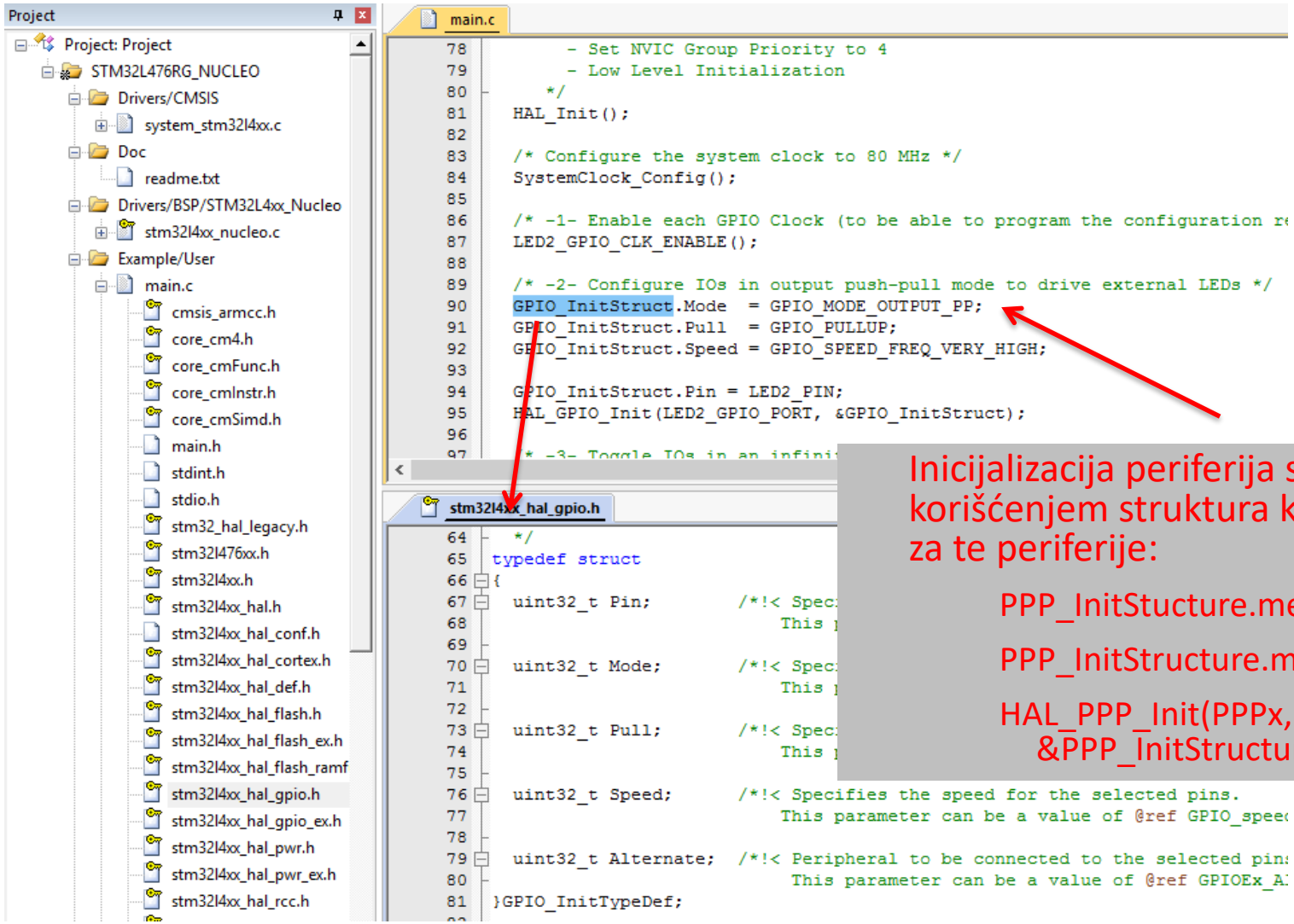
# Inicijalizacija periferija I

Pre konfigurisanja bilo koje periferije potrebno je dovesti joj takt korišćenjem odgovarajuće funkcije iz RCC drajvera:

__HAL_RCC_*PPP*_CLK_ENABLE()



# Inicijalizacija periferija II




Inicijalizacija periferija se uvek izvododi korišćenjem struktura karakterističnih za te periferije:

PPP_InitStucture.memberX = valX;

PPP_InitStructure.memberY = valY;

HAL_PPP_Init(PPPx, &PPP_InitStructure);

# Kako se togluje LED2

```
main.c
93
94    GPIO_InitStruct.Pin = LED2_PIN;
95    HAL_GPIO_Init(LED2_GPIO_PORT, &GPIO_InitStruct);
96
97    /* -3- Toggle IOs in an infinite loop */
98    while (1)
99    {
100     HAL_GPIO_TogglePin(LED2_GPIO_PORT, LED2_PIN);
101     /* Insert delay
102     HAL_Delay(100);
```

```
stm32l4xx_hal_gpio.c
  * @param  GPIO_Pin: specifies the pin to be toggled.
  * @retval None
  */
void HAL_GPIO_TogglePin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
{
  /* Check the parameters */
  assert_param(IS_GPIO_PIN(GPIO_Pin));

  GPIOx->ODR ^= GPIO_Pin;
```

```
stm32l4xx_nucleo.h
#define LEDn                                1

#define LED2_PIN                            GPIO_PIN_5
#define LED2_GPIO_PORT                      GPIOA
#define LED2_GPIO_CLK_ENABLE()              __HAL_RCC_GPIO
#define LED2_GPIO_CLK_DISABLE()             __HAL_RCC_
```

```
stm32l4xx_hal_gpio.h
#define GPIO_PIN_0                 ((uint16_t)0x0001)
#define GPIO_PIN_1                 ((uint16_t)0x0002)
#define GPIO_PIN_2                 ((uint16_t)0x0004)
#define GPIO_PIN_3                 ((uint16_t)0x0008)
#define GPIO_PIN_4                 ((uint16_t)0x0010)
#define GPIO_PIN_5                 ((uint16_t)0x0020)
```

Podatak će biti 0x0020

```
stm32l4xx_hal_gpio.c  stm32l4xx_nucleo.h  stm32l476xx.h
#define GPIOA                           (GPIO_TypeDef *) GPIOA_BASE)
#define GPIOB                           (GPIO_TypeDef *) GPIOB_BASE)
#define GPIOC                           (GPIO_TypeDef *) GPIOC_BASE)
#define GPIOD                           (GPIO_TypeDef *) GPIOD_BASE)
#define GPIOE                           (GPIO_TypeDef *) GPIOE_BASE)
```

```
stm32l476xx.h
typedef struct
{
  __IO uint32_t MODER;
  __IO uint32_t OTYPER;
  __IO uint32_t OSPEEDR;
  __IO uint32_t PUPDR;
  __IO uint32_t IDR;
  __IO uint32_t ODR;
  __IO uint32_t BSRR;
  __IO uint32_t LCKR;
  __IO uint32_t AFR[2];
  __IO uint32_t BRR;
  __IO uint32_t ASCR;

} GPIO_TypeDef;
```

```
stm32l4xx_hal_gpio.c  stm32l4xx_nucleo.h  stm32l476xx.h
#define GPIOA_BASE              (AHB2PERIPH_BASE + 0x0000U)
#define GPIOB_BASE              (AHB2PERIPH_BASE + 0x0400U)
#define GPIOC_BASE              (AHB2PERIPH_BASE + 0x0800U)
```

```
stm32l476xx.h  stm32l4xx_nucleo.h
  */
#define FLASH_BASE              ((uint32_t)0x08000000U) /*!< FLASH(up to 1 MB) base
#define SRAM1_BASE              ((uint32_t)0x20000000U) /*!< SRAM1(up to 96 KB) bas
#define PERIPH_BASE             ((uint32_t)0x40000000U) /*!< Peripheral base addres
#define FMC_BASE                ((uint32_t)0x60000000U) /*!< FMC base address */
#define SRAM2_BASE              ((uint32_t)0x10000000U) /*!< SRAM2(32 KB) base addr
/*!< Peripheral memory map */
#define APB1PERIPH_BASE          PERIPH_BASE
#define APB2PERIPH_BASE         (PERIPH_BASE + 0x00010000U)
#define AHB1PERIPH_BASE         (PERIPH_BASE + 0x00020000U)
#define AHB2PERIPH_BASE         (PERIPH_BASE + 0x08000000U)
```

**Na dresu 0x48000000 dodaje offset za ODR koji iznosi 0x14 i dobija se adresa 0x48000014**

# LED Toggle - BSRR

- Napisati funkciju
  `GPIO_TogglePin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)`
  koja vrši izmenu stanja izlaza pina upisujući u
  - BSR registar GPIO porta
  - BSR i BR registar GPIO porta
- Testirati