

PRIMENA MIKROKONTROLERA- MS1PMK

4. deo

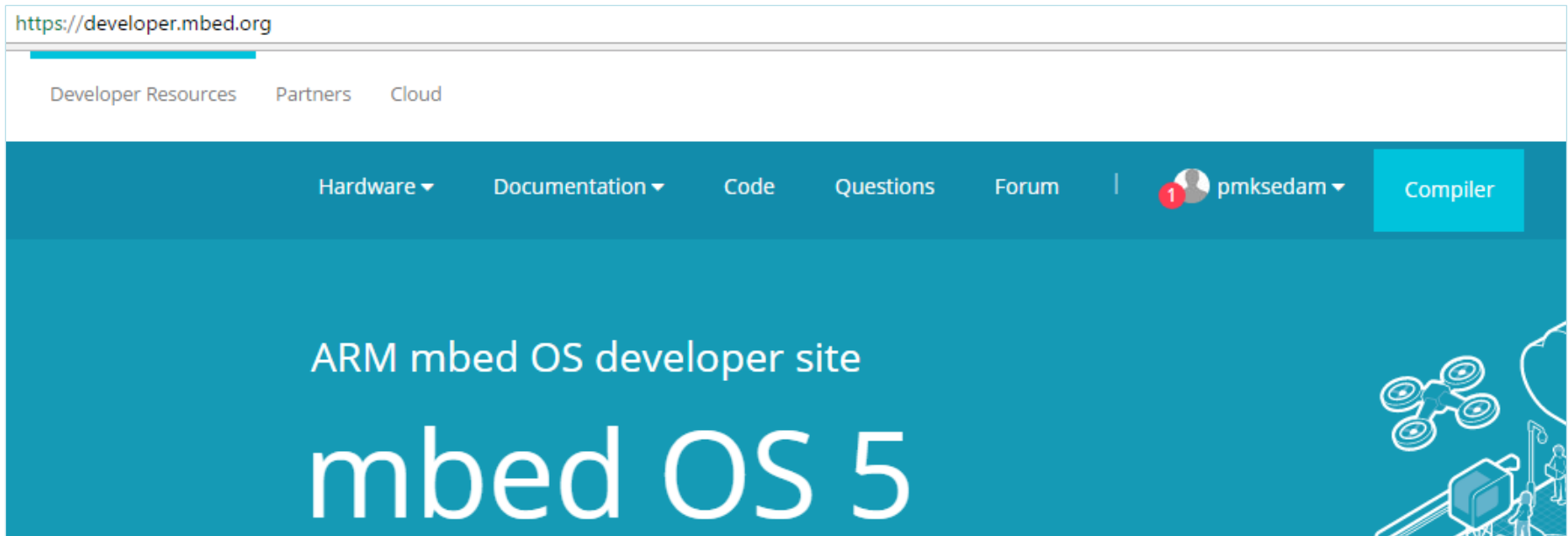
2018

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

The screenshot shows the mbed workspace management interface. At the top, the browser address bar displays "Secure | https://developer.mbed.org/compiler/#nav/;". The main header includes "mbed" and "Workspace Management". Below the header is a toolbar with icons for "New", "Import", "Save", "Save All", "Compile", "Commit", "Revision", and "Help".

The central focus is a "Select a Platform" dialog box. The title bar of the dialog reads "Select a Platform". Inside the dialog, the "NUCLEO-F401RE" platform is selected, indicated by a green checkmark and the text "You are currently compiling for the NUCLEO-F401RE platform." A "Select Platform" button is visible in the top right corner of the dialog.

The dialog displays an image of the NUCLEO-F401RE board with a red dashed line pointing to it. To the right of the image is a "More Info" button. The "Description" tab is active, showing the following text:

Description | Pinout

Affordable and flexible platform to ease prototyping using a STM32F401RET6 microcontroller.

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®, Ꞥ 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 debugger/programmer.

<https://www.youtube.com/watch?v=68&list=PLgyFKd2HIZlBhKhngvDGmsJxX0uLscnV>

FntTelSoqSk&index

68&list=PLgyFKd2HIZlBhKhngvDGmsJxX0uLscnV}}

At the bottom of the dialog, there is a section titled "Your registered platforms" which contains two items: "NUCLEO-F401RE" and "Add Platform". A red dashed line points from the "Add Platform" button to the "NUCLEO-F401RE" platform in the "Your registered platforms" section.

In the top right corner of the workspace, a red box highlights the "NUCLEO-F401RE" platform name.

Dodavanje nove platforme

NUCLEO-L476RG

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



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-J INK/V2.1

Table of Contents

1. Overview
2. Microcontroller features
3. Nucleo features
4. Nucleo pinout
5. Supported shields
6. Getting started



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

Board Partner



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](#)

- vratimo se nazad u compiler i selektujemo dodatu platformu

Kreiranje novog projekta

The screenshot shows the mbed workspace management interface. The browser address bar displays the URL <https://developer.mbed.org/compiler/#nav:/>. The main interface has a teal header with the mbed logo and 'Workspace Management'. A toolbar contains buttons for 'New', 'Import', 'Save', 'Save All', 'Compile', 'Commit', 'Revision', and 'Help'. The 'New' button is highlighted with a red box. Below the toolbar, the interface is divided into three panels: 'Program Workspace' (left), 'Workspace Management' (center), and 'Workspace Details' (right). The 'Workspace Management' panel shows 'Manage your Program Workspace' and a 'Create new program' dialog box. The dialog box has a title bar 'Create new program' and a close button. The main text reads: 'Create new program for "NUCLEO-L476RG"'. Below this, it says: 'This will create a new C++ program for "NUCLEO-L476RG" in your workspace. You can always change the platform of this program once created.' An information icon and the text 'Please specify program name' are also present. The dialog contains three dropdown menus: 'Platform:' set to 'NUCLEO-L476RG', 'Template:' set to 'Blinky LED test for the ST Nucleo boards', and 'Program Name:' with the text 'Nucleo blink led' entered. Below the 'Program Name' field, there is a note: 'The name of the program to be created in your workspace' and a checked checkbox labeled 'Update this program and libraries to latest revision'. At the bottom of the dialog are 'OK' and 'Cancel' buttons.

Secure | <https://developer.mbed.org/compiler/#nav:/>

mbed Workspace Management

New Import Save Save All Compile Commit Revision Help NUCLEO-L476RG

Program Workspace < My Programs

Workspace Management

Manage your Program Workspace

Create new program

Create new program for "NUCLEO-L476RG"

This will create a new C++ program for "NUCLEO-L476RG" in your workspace. You can always change the platform of this program once created.

Please specify program name

Platform: NUCLEO-L476RG

Template: Blinky LED test for the ST Nucleo boards

Program Name: Nucleo blink led

The name of the program to be created in your workspace

Update this program and libraries to latest revision

OK Cancel

Workspace Details

pmksest

Total Programs 0

Modified 18 May 2016

Recently Modified

Kompajliranje i spuštanje na ploču

The screenshot displays the Mbed IDE interface. The top menu bar includes options like 'New', 'Import', 'Save', 'Save All', 'Compile', 'Mbed Cloud', 'Commit', 'Revision', and 'Save As'. The 'Compile' button is highlighted with a red box. The 'Program Workspace' on the left shows a project named 'Nucleo_blink_led' with a file 'main.cpp'. The code editor shows the following code:

```
1 #include "mbed.h"
2
3 DigitalOut myled(LED1);
4
5 int main() {
6     while(1) {
7         myled = 1; // LED is ON
8         wait(0.2); // 200 ms
9         myled = 0; // LED is OFF
10        wait(1.0); // 1 sec
11    }
12 }
13
```

A file explorer window is open, showing the save location 'NODE_L476RG (D:)' and the file name 'Nucleo_blink_led_NUCLEO_L476RG.bin'. The 'Save as type' is set to 'BIN File (.bin)'.

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)`

Create a **DigitalOut** connected to the specified pin.

void `write (int value)`

Set the output, specified as 0 or 1 (int)

int `read ()`

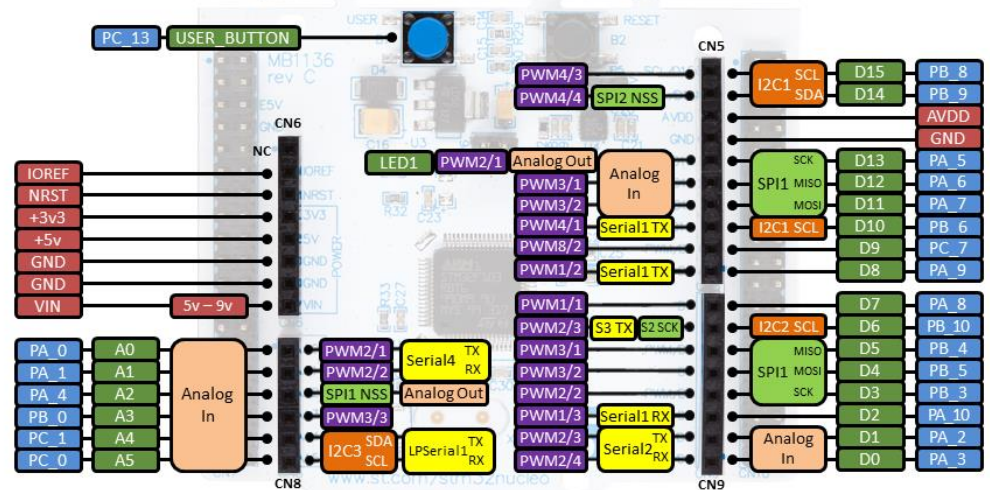
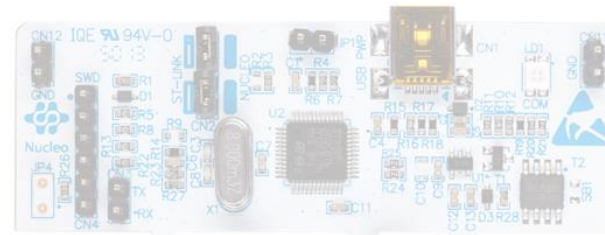
Return the output setting, represented as 0 or 1 (int)

DigitalOut & `operator= (int value)`

A shorthand for `write()`

`operator int ()`

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](#) ()

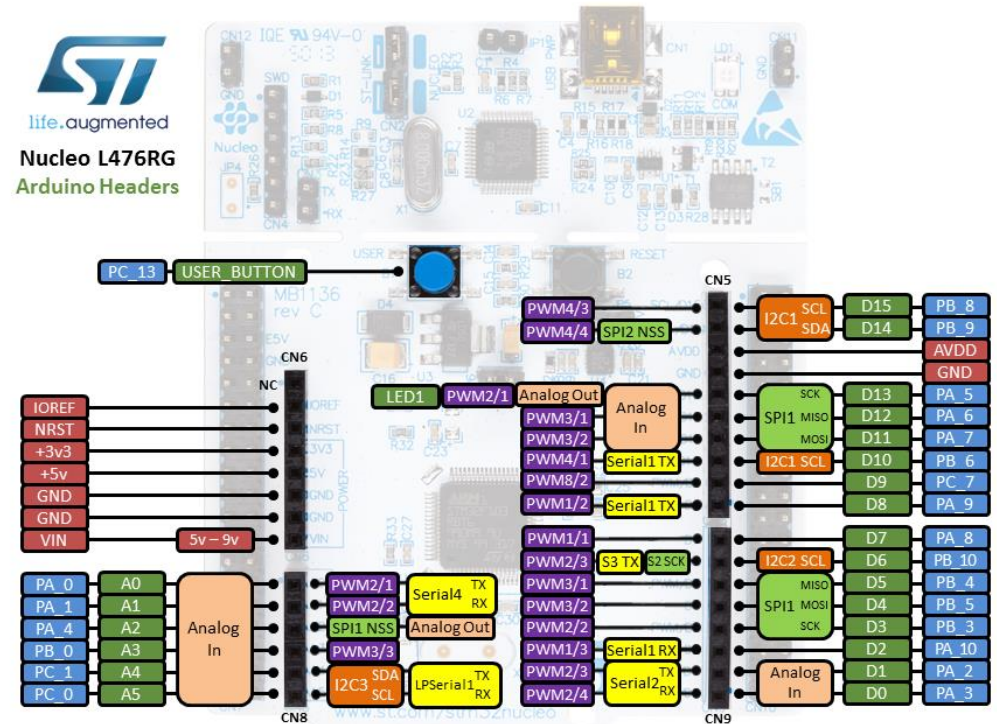
Read the input, represented as 0 or 1 (int)

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"
```

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

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

```
int main() {  
    while(1) {
```

```
        myled = mybutton.read();
```

```
        // Read button state and  
        // transfer the value to LED.
```

```
    }  
}
```

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

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

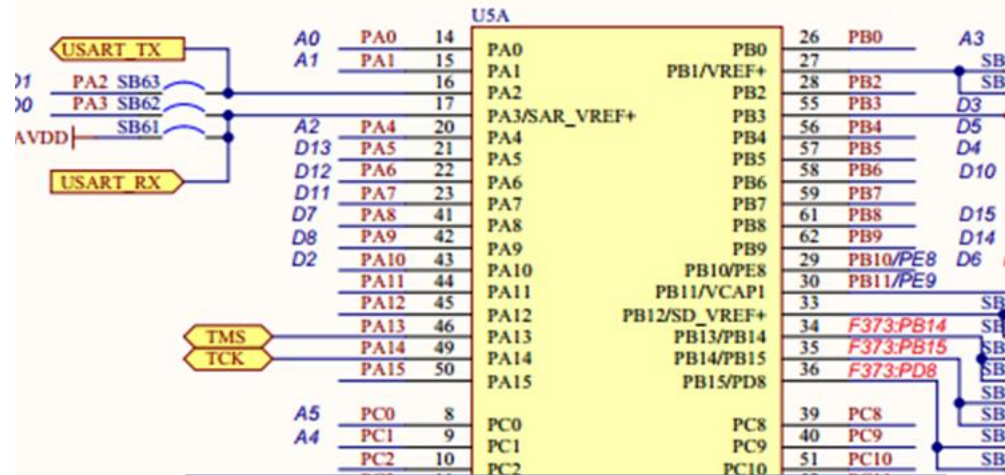
<http://mbed.org/handbook/PortIn> (PortOut, PortInOut)

- 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](#) ()

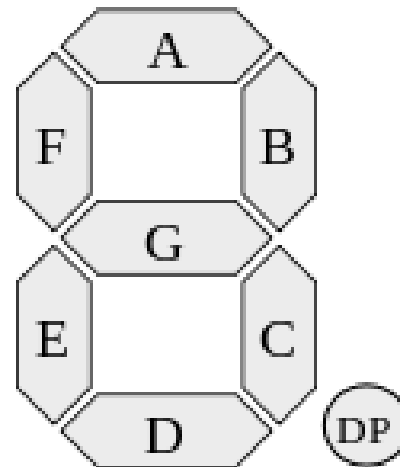
Read the value of the input bus.

[operator int](#) ()

A shorthand for [read\(\)](#)

ZADACI

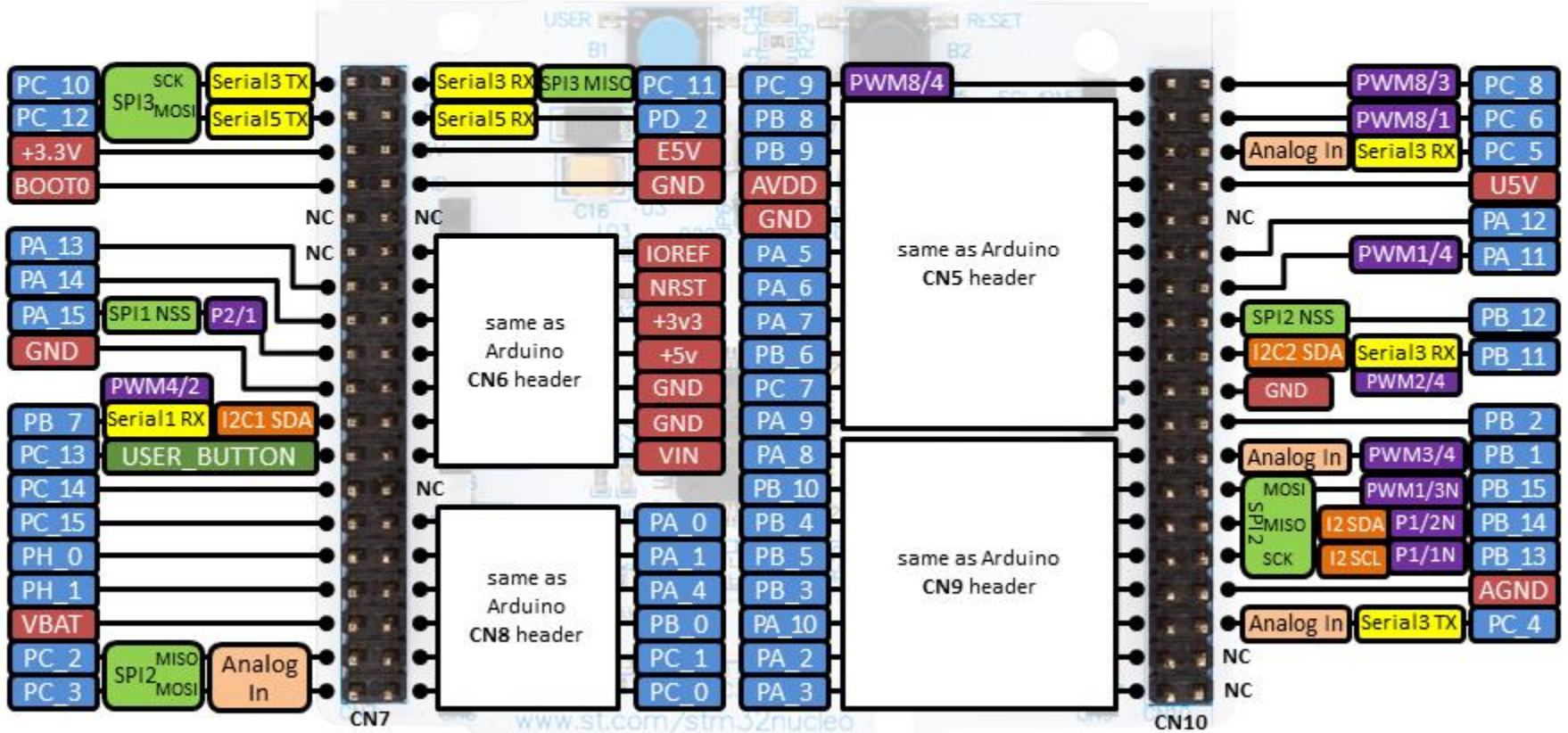
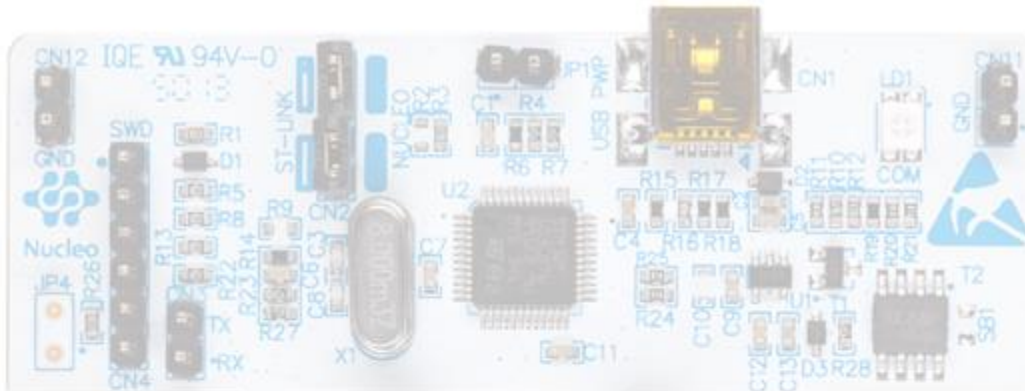
3. Napisati program koji trenutnu vrednost brojača iz tačke 2 prikazuje na sedmosegmentnom LED displeju.



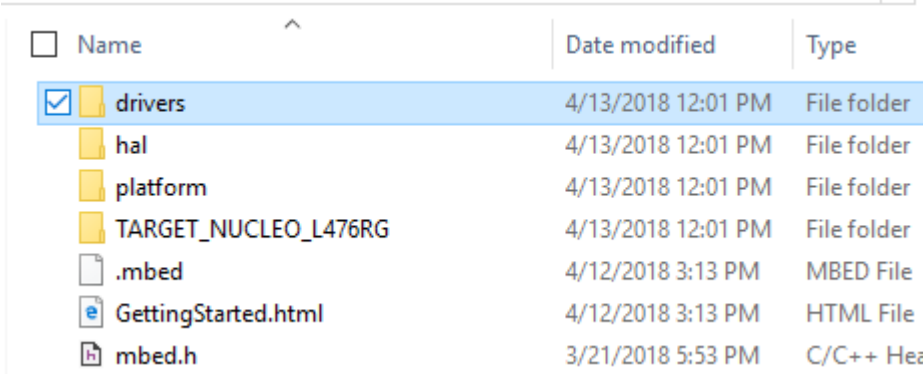


life.augmented

Nucleo L476RG Morpho Headers



MBED - source code



Name	Date modified	Type
<input checked="" type="checkbox"/> drivers	4/13/2018 12:01 PM	File folder
hal	4/13/2018 12:01 PM	File folder
platform	4/13/2018 12:01 PM	File folder
TARGET_NUCLEO_L476RG	4/13/2018 12:01 PM	File folder
.mbed	4/12/2018 3:13 PM	MBED File
GettingStarted.html	4/12/2018 3:13 PM	HTML File
mbed.h	3/21/2018 5:53 PM	C/C++ Hea

- 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 deklariranih u folderu **hal**.

mbed.h

platform.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"
```

```
1
2 /** \addtogroup platform */
3 /** @{*/
4 /* mbed Microcontroller Library
19 #ifndef MBED_PLATFORM_H
20 #define MBED_PLATFORM_H
21
22 #include <stddef>
23 #include <stdlib>
24 #include <stdio>
25 #include <string>
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 x  gpio_api.h x  device.h x  DigitalOut.h x  stm32f10x_gpio.c x  gpio_api.c x  BusOut.cpp x
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 x 24.03.2016_23.49.21.zgz x device.h x
#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 x AnalogOut.h x gpio_api.h x device.h x PinNames.h x DigitalOut.h x stm32f10x_gpio.c x 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 x PeripheralNames.h x  
- 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

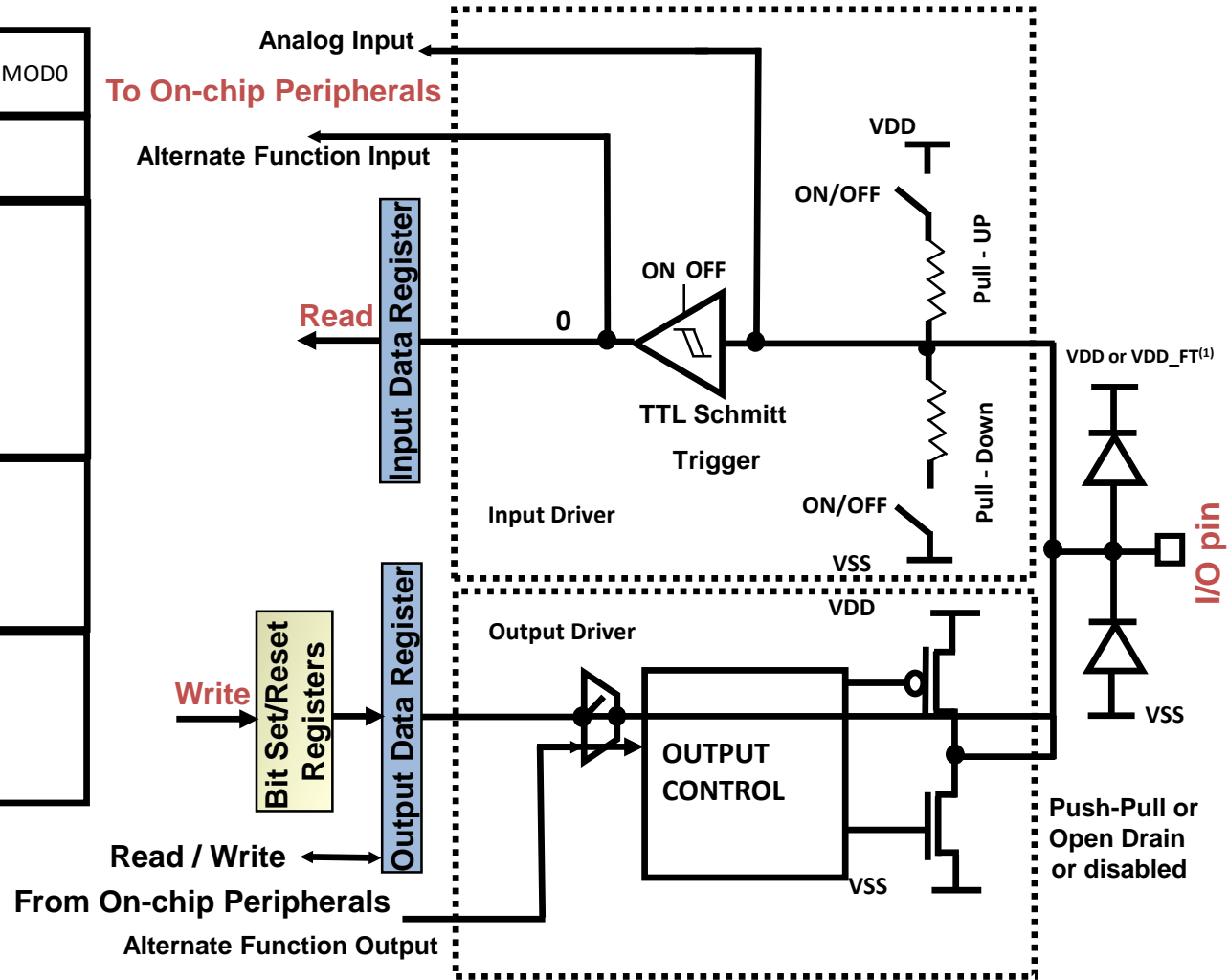
- STM32 Nucleo-64 board – [User Manual](#)
 - Podaci vezani za pločicu: pinout, soldering bridges, pull up, pull down, tasteri, diode
- STM32L476xx – MCU [Datasheet](#)
 - Provera karakteristika pri izboru mikrokontrolera
 - Alternate pin function
 - Additional pin function
- STM32L4x5 and STM32L4x6 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

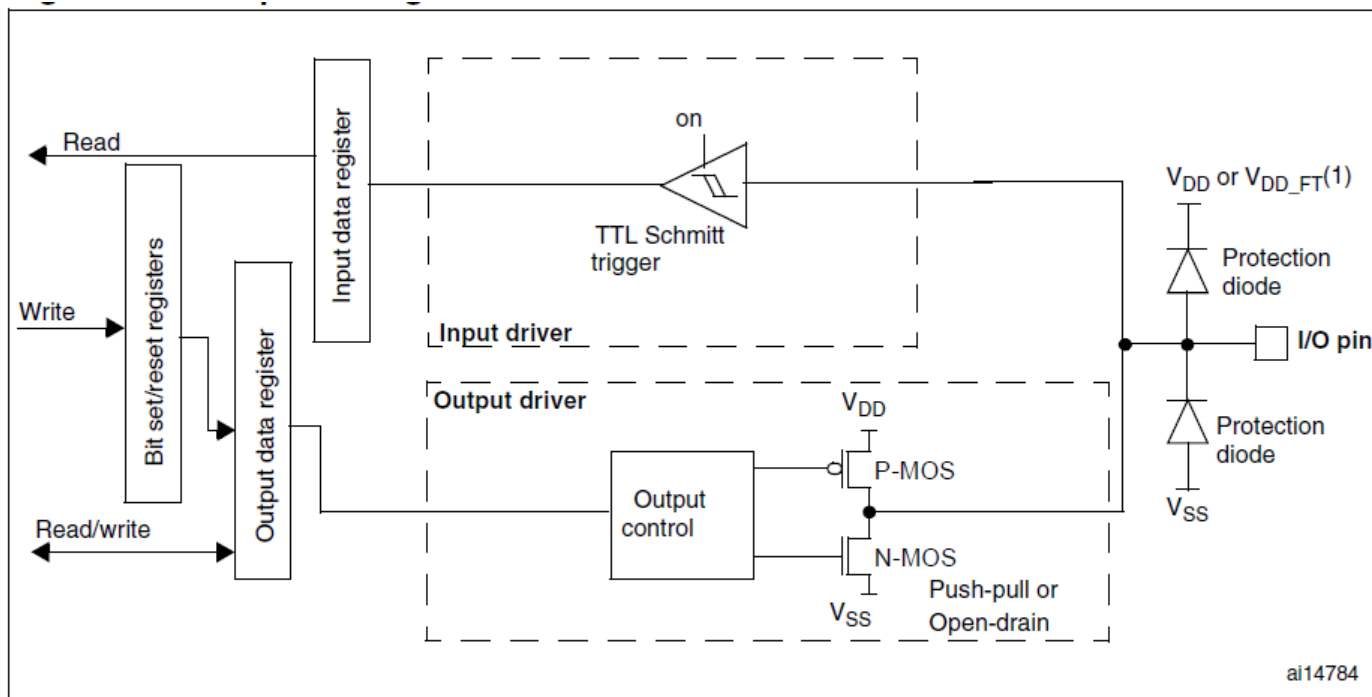
GPIO Konfiguracije

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		



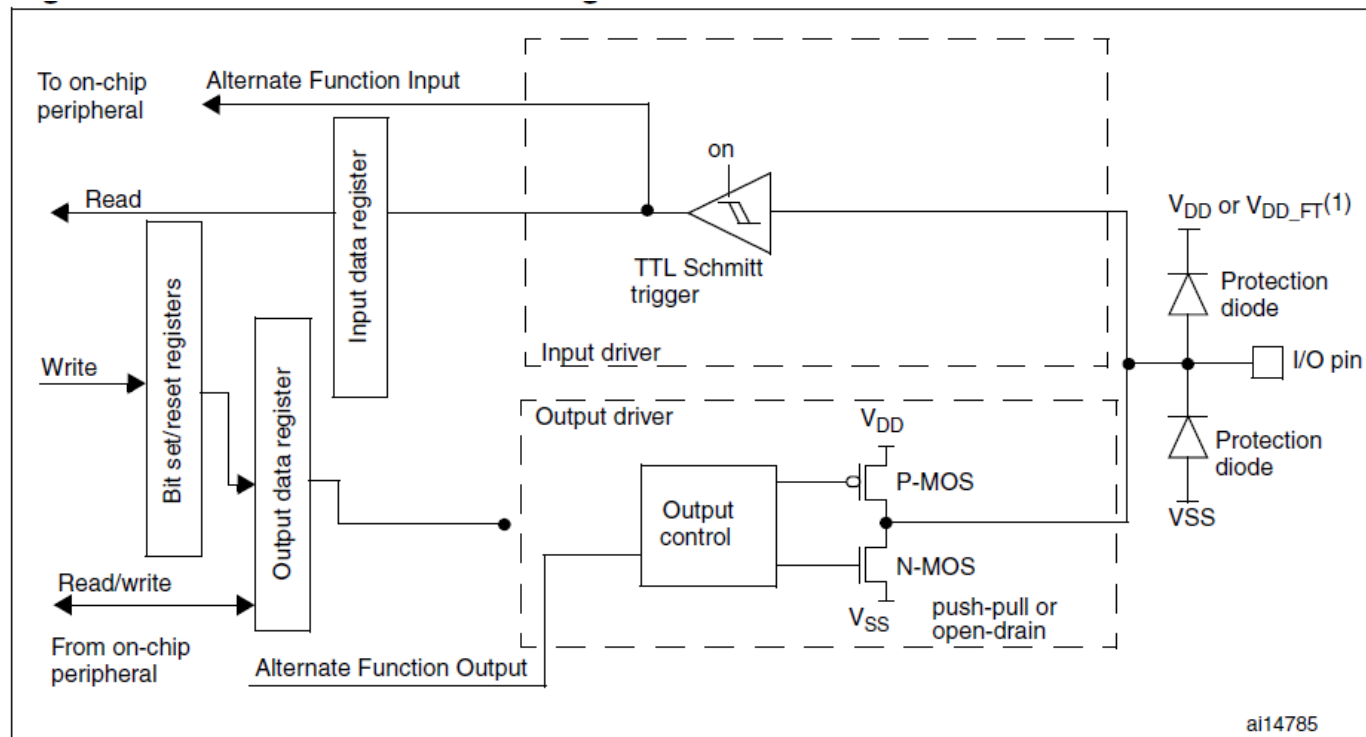
Izlazna konfiguracija

- **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 tranzistor se nikada ne aktivira).
- **Push-pull mod:** "0" u *Output Data* registru aktivira N-MOS, dok "1" u *Output Data* registru aktivira P-MOS.
- **Schmitt Trigger** kolo je **uključeno**
- Interni **pull-up** i **pull-down** 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



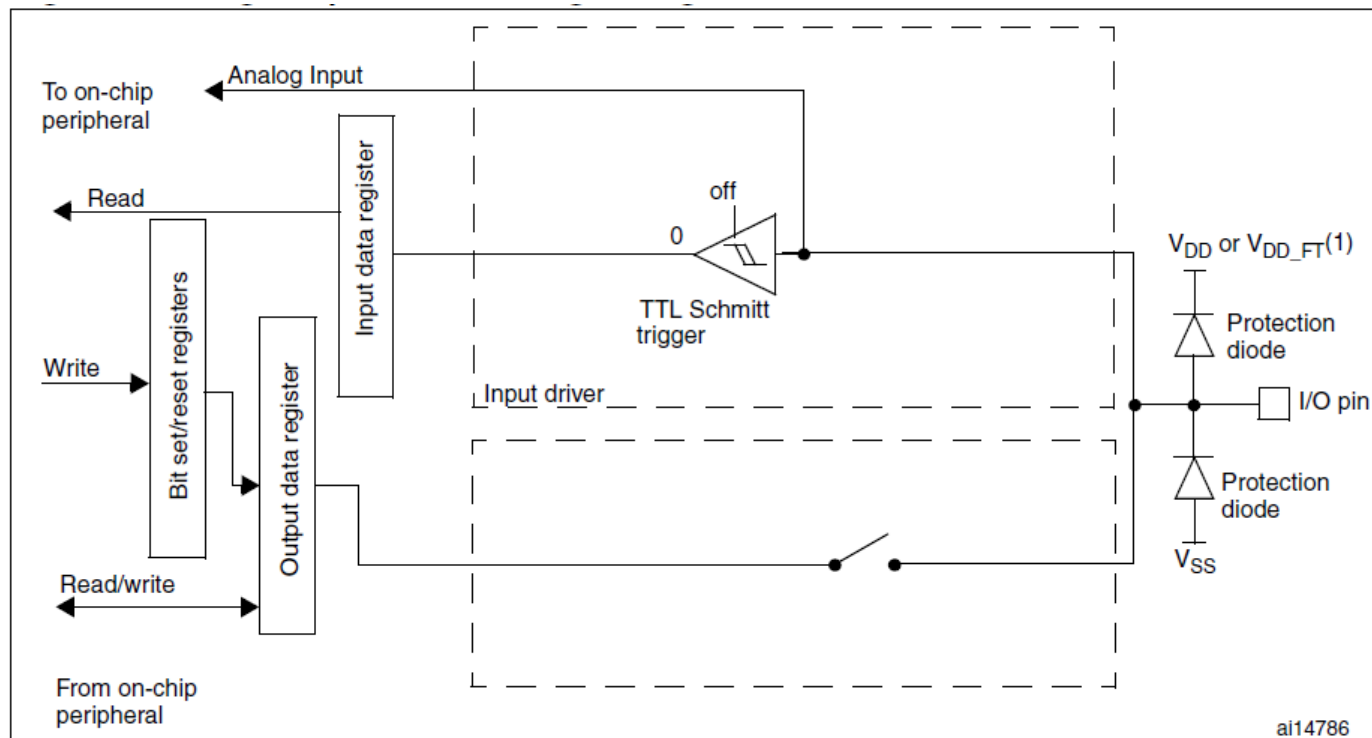
Konfiguracija alternativne funkcije

- 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 sempljuje 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 **isključeno** 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
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

Bits 15:0 **OT_y**: 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]	
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
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]	
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 2y+1:2y **PUPD_y[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]	
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
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]	
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits $2y+1:2y$ **OSPEED y [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
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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 Write Only mode.

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
Bits 31:17 Reserved, must be kept at reset value.															
15	14	13	12	11	10	9	8	7	Bit 16 LCKK : Lock key						
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	This bit can be read any time. It can only be modified using the lock key write sequence.						
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	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.

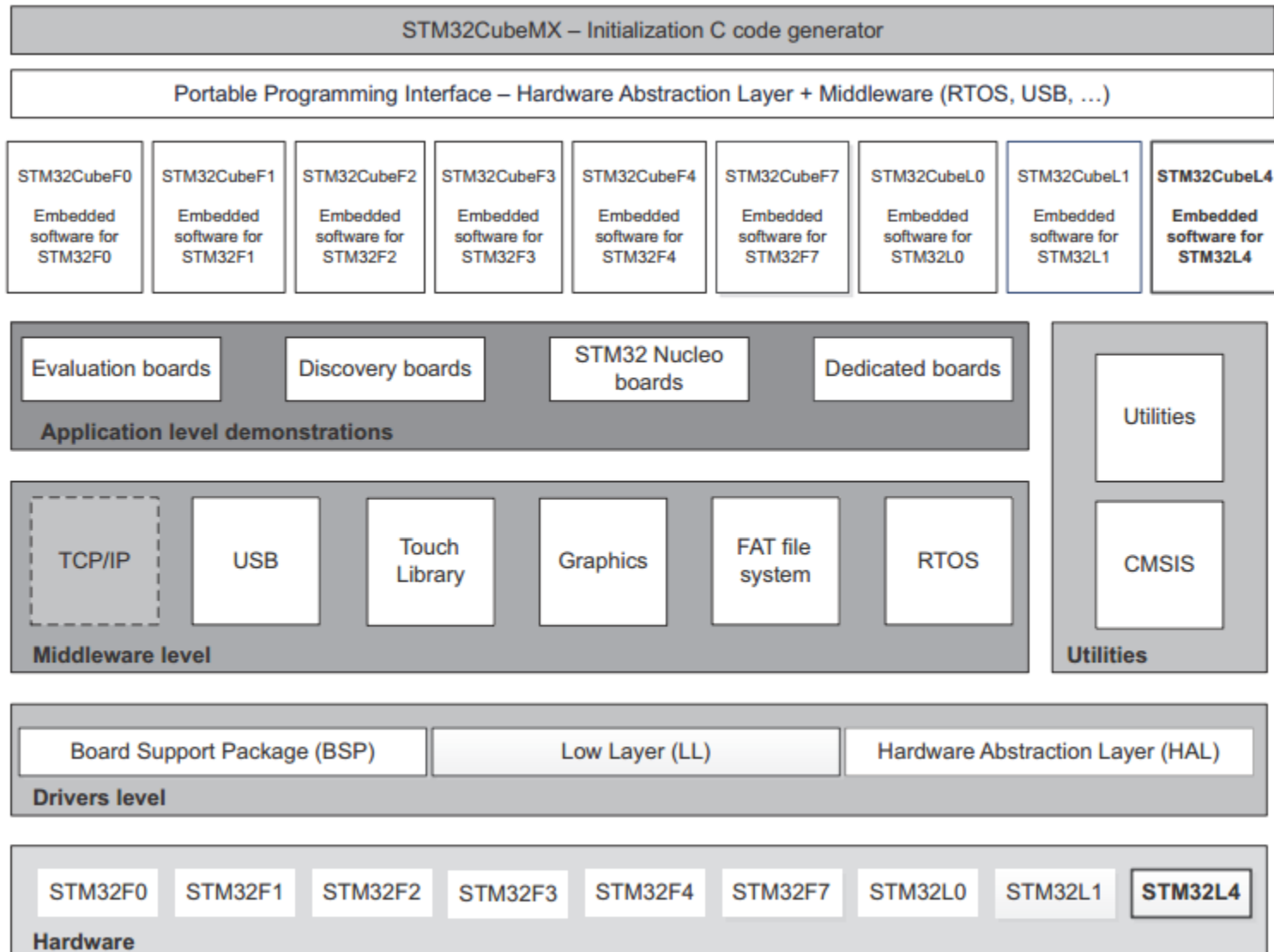
Bits 15:0 **LCKy**: Port x lock bit y (y= 0..15)

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

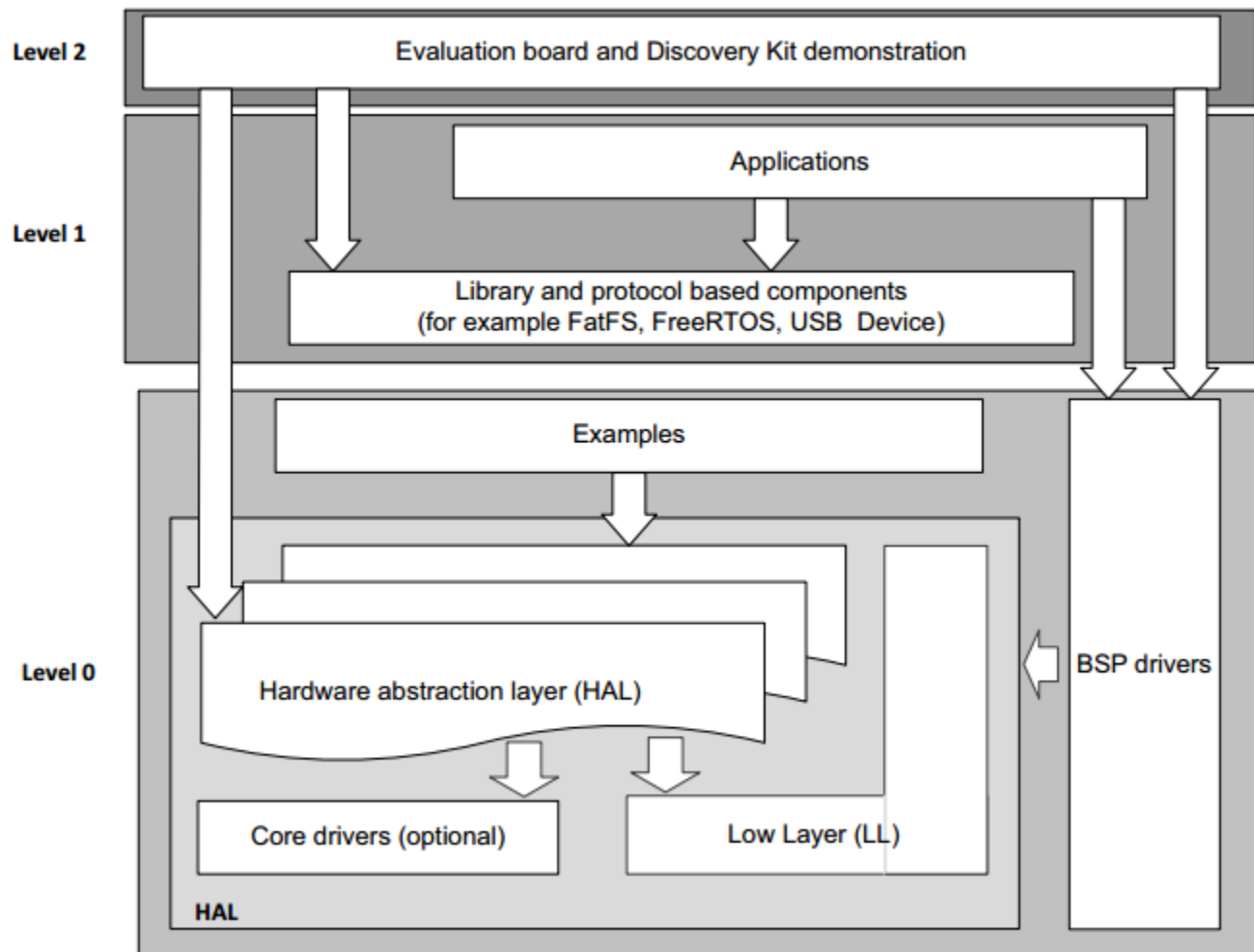
0: Port configuration not locked

1: Port configuration locked

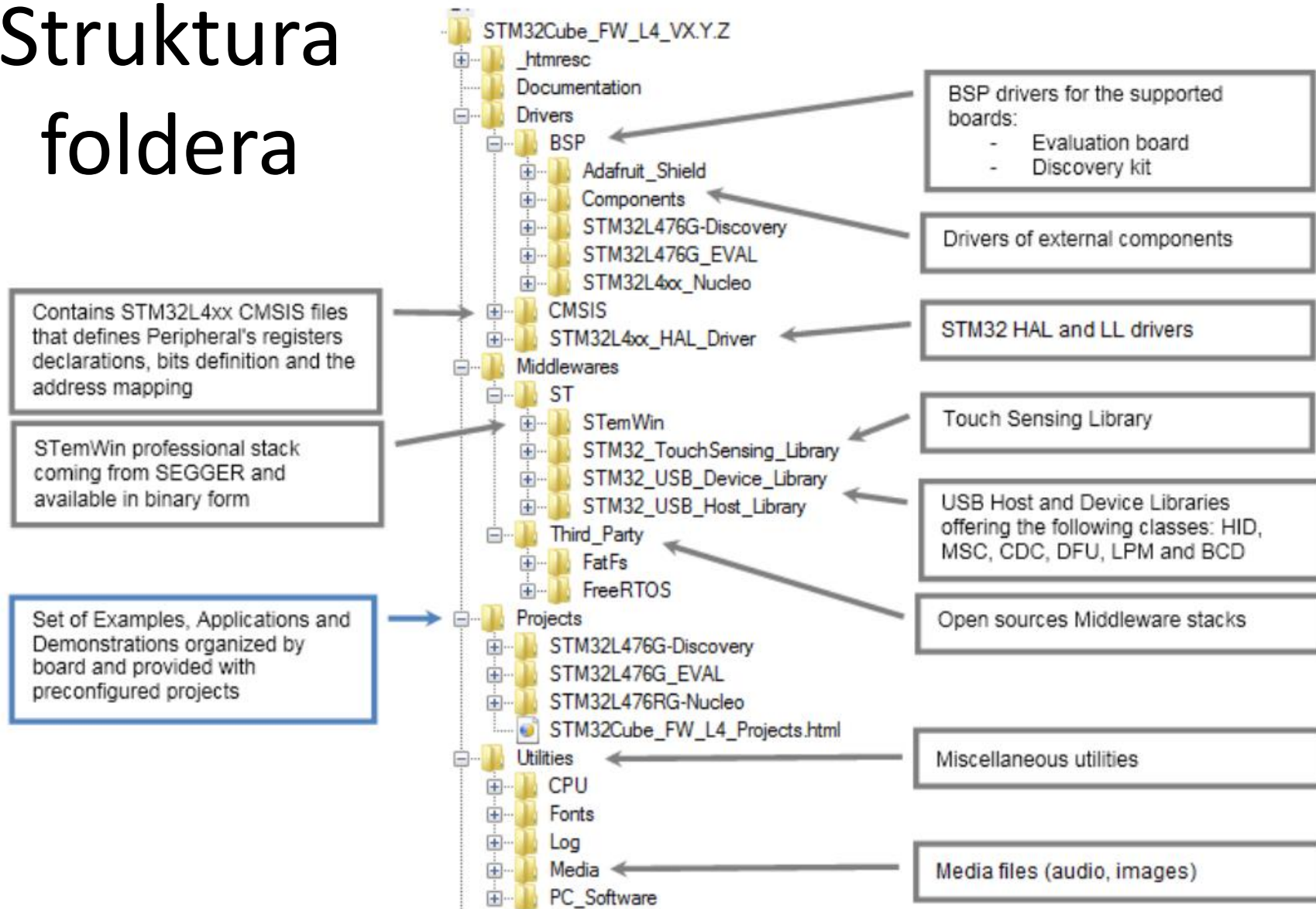
STM32CubeL4 komponente




STM32CubeL4 arhitektura



Struktura foldera



 Library files, to not be modified by user

 User modifiable files

Primeri – GPIO IOToggle

The image shows a Windows File Explorer window displaying the directory structure of a project named 'stm32cube4'. The left pane shows a tree view where the 'GPIO' folder under the 'Examples' subdirectory is selected. A red arrow points from this 'GPIO' folder to the 'GPIO_IOToggle' folder in the right pane. The right pane shows the contents of the selected folder, including 'MDK-ARM' (highlighted), 'Src', 'SW4STM32', and 'TrueSTUDIO'. A third pane on the right shows a list of files: 'Project.uvoptx', 'Project.uvprojx' (highlighted), and 'startup_stm32l476xx.s'.

- stm32cube4
 - STM32Cube_FW_L4_V1.4.0
 - _htmresc
 - Documentation
 - Drivers
 - Middlewares
 - Projects
 - STM32L432KC-Nucleo
 - STM32L476G_EVAL
 - STM32L476G-Discovery
 - STM32L476RG-Nucleo
 - Applications
 - Demonstrations
 - Examples
 - COMP
 - Cortex
 - CRC
 - DAC
 - FIREWALL
 - FLASH
 - GPIO**
 - HAL
 - I2C
 - IWDG
 - LPTIM

- FIREWALL
- FLASH
- GPIO
- GPIO_EXTI
- GPIO_IOToggle
- EWARM
- Inc
- MDK-ARM
- Src
- SW4STM32
- TrueSTUDIO
- HAL
- I2C

Name

- Project.uvoptx
- Project.uvprojx
- startup_stm32l476xx.s

Projekat GPIO_IOToggle

Project: Project

STM32L476RG_NUCLEO

- Drivers/CMSIS
- Doc
- readme.txt
- Drivers/BSP/STM32L4xx_Nuc
- stm32l4xx_nucleo.c
- Example/User
- main.c
- stm32l4xx_it.c
- Drivers/STM32L4xx_HAL_Drv
- stm32l4xx_hal_gpio.c
- stm32l4xx_hal.c
- stm32l4xx_hal_rcc.c
- stm32l4xx_hal_cortex.c
- stm32l4xx_hal_pwr.c
- stm32l4xx_hal_pwr_ex.c
- Example/MDK-ARM
- startup_stm32l476xx.s

Disassembly

```
334:  while((HAL_GetTick() - tickstart) < Delay)
335:  {
336:  }
0x08000216 0400 LSLS      r0,r0,#16
0x08000218 F000F8EE BL.W     HAL_GetTick (0x080003F8)
0x0800021C 1B00 SUBS    r0,r0,r4
0x0800021E 4310 CMP     r0,r5
```

main.c

```
77      handled in milliseconds basis.
78      - Set NVIC Group Priority to 4
79      - Low Level Initialization
80      */
81  HAL_Init();
82
83  /* Configure the system clock to 80 MHz */
84  SystemClock_Config();
85
86  /* -1- Enable each GPIO Clock (to be able to program
87  LED2_GPIO_CLK_ENABLE());
88
89  /* -2- Configure IOs in output push-pull mode to
90  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
91  GPIO_InitStruct.Pull = GPIO_PULLUP;
92  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
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 100 ms */
102     HAL_Delay(100);
103  }
```

stm32l4xx_hal.c

Effikasnije pretraživanje i orijentacija se mogu postići dodavanjem shortcut-ova za pojedine naredbe:

- Edit->Configuration->Shortcut Keys->
- Go to Definition
- Navigate Backwards
- Navigate Forwards
- Comment
- Uncomment

Gde se nalazi definicija funkcije HAL_GPIO_TogglePin ()?
Kojim registrima se pristupa?

HAL_Delay() funkcija implementira čekanje. Na koji način?

Opcije razvojnog okruženja

The image shows a screenshot of an IDE interface for STM32 development. The main window displays a C source file named `main.c` with the following code:

```
78     - Set NVIC Group Priority to 4
79     - Low Level Initialization
80     */
81 HAL_Init();
82
83 /* Configure the system clock to 80 MHz */
84 SystemClock_Config();
85
86 /* -1- Enable each GPIO Clock (to be able to program the configuration registers)
87 LED2_GPIO_CLK_ENABLE();
88
89 /* -2- Configure IOs in output push-pull mode to drive external LEDs */
90 GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
91 GPIO_InitStruct.Pull = GPIO_PULLUP;
92 GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
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 100 ms */
102     HAL_Delay(500);
103 }
104 }
105
```

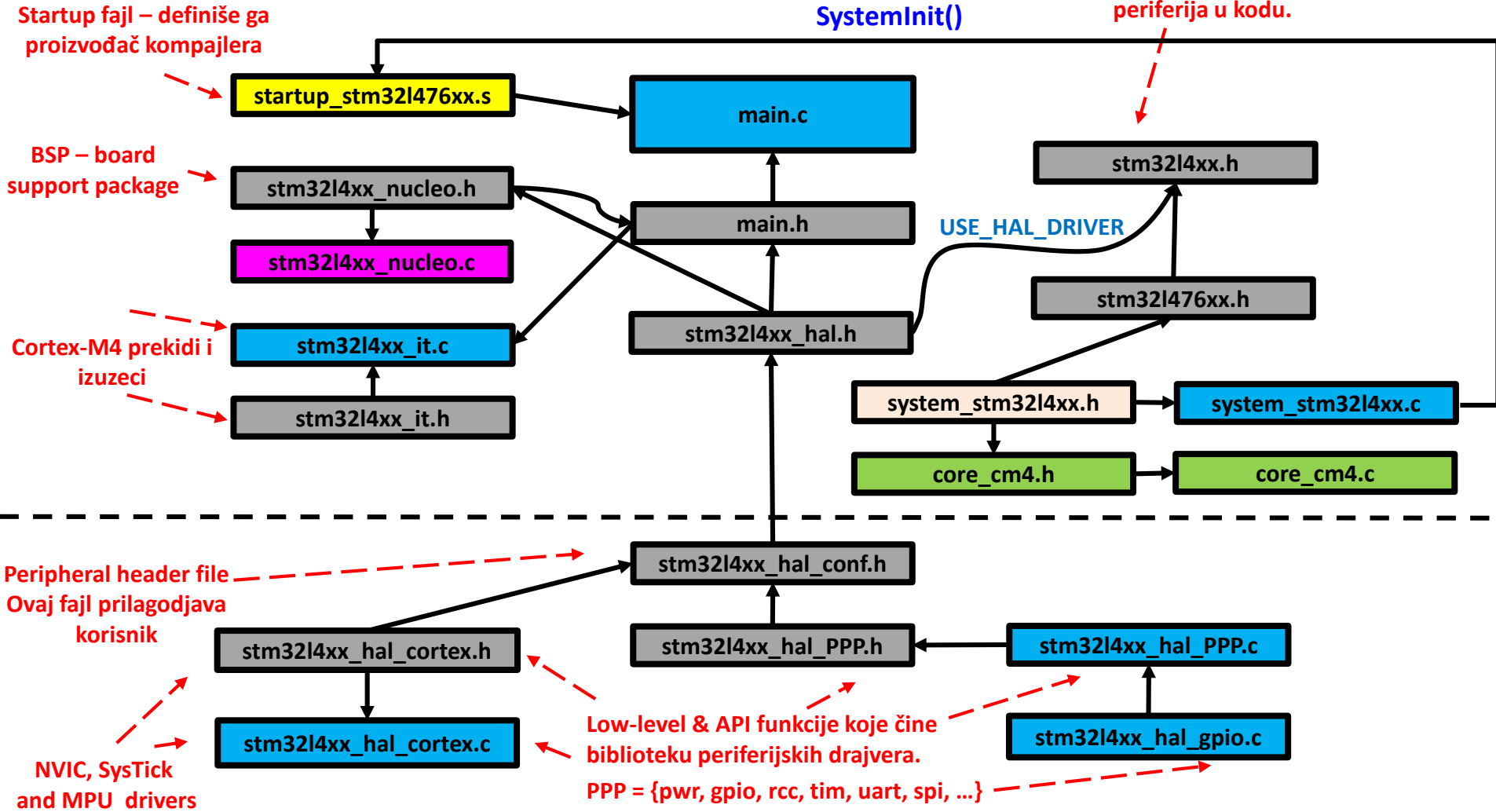
Annotations with red arrows point to specific IDE features:

- build**: Points to the Build icon in the toolbar.
- flash/download**: Points to the Load icon in the toolbar.
- navigate backwards**: Points to the left arrow icon in the toolbar.
- naigate forwards**: Points to the right arrow icon in the toolbar.
- debug mode**: Points to the Debug icon in the toolbar.
- set breakpoint**: Points to the Breakpoint icon in the toolbar.

The Project Explorer on the left shows the project structure for `STM32L476RG_NUCLEO`, including folders like `Drivers/CMSIS`, `Doc`, `Drivers/BSP/STM32L4xx_Nuc`, `Example/User`, `main.c`, `stm32l4xx_it.c`, `Drivers/STM32L4xx_HAL_Drv`, and `Example/MDK-ARM`.

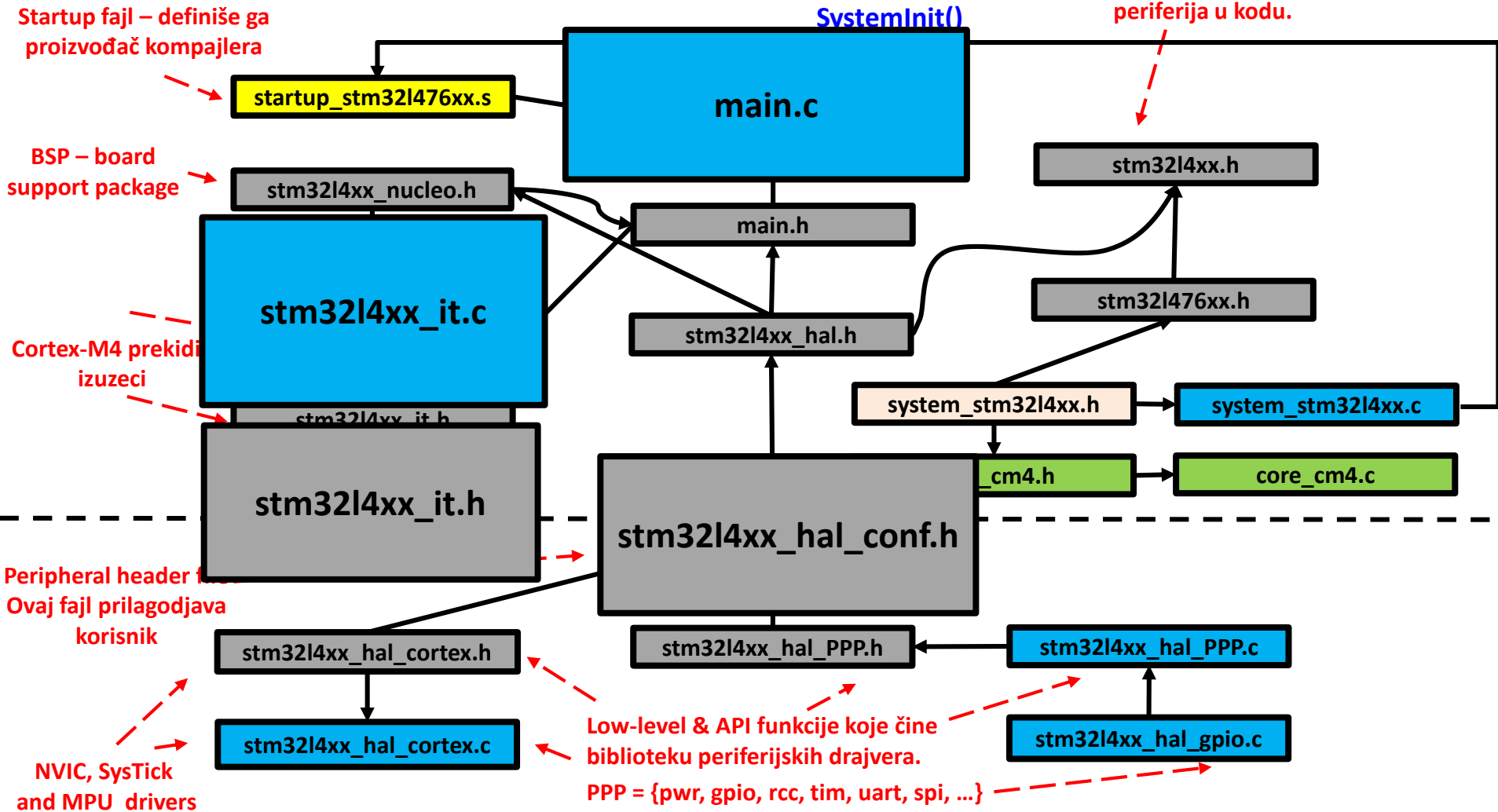
STM CUBE CMSIS Struktura projekta

Osnovni konfiguracioni header fajl za odabranu familiju mikrokontrolera. U zavisnosti od USE_HAL_DRIVER obezbeduje korišćenje HAL API-ja, ili zahteva direktan pristup registima periferija u kodu.



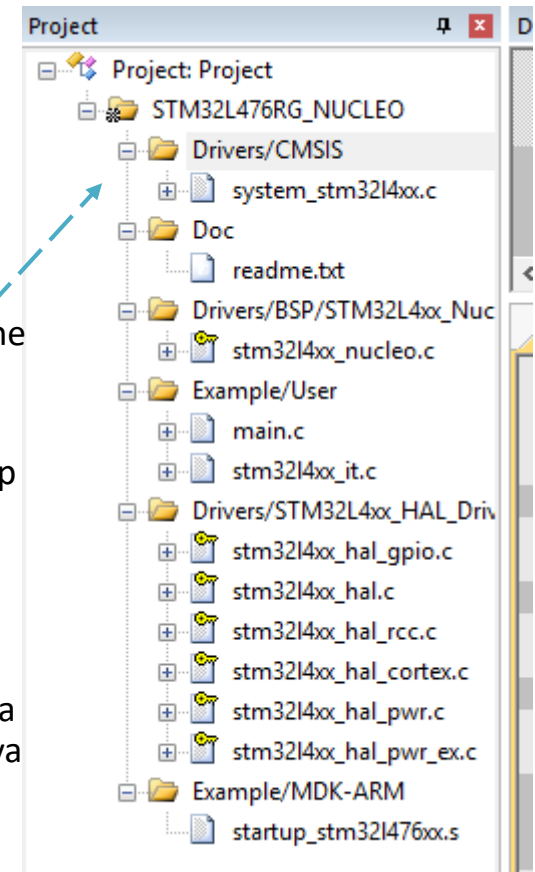
Fajlovi koje menja korisnik

Osnovni konfiguracioni header fajl za odabranu familiju mikrokontrolera. U zavisnosti od USE_HAL_DRIVER obezbeđuje korišćenje HAL API-ja, ili zahteva direktan pristup registrima periferija u kodu.



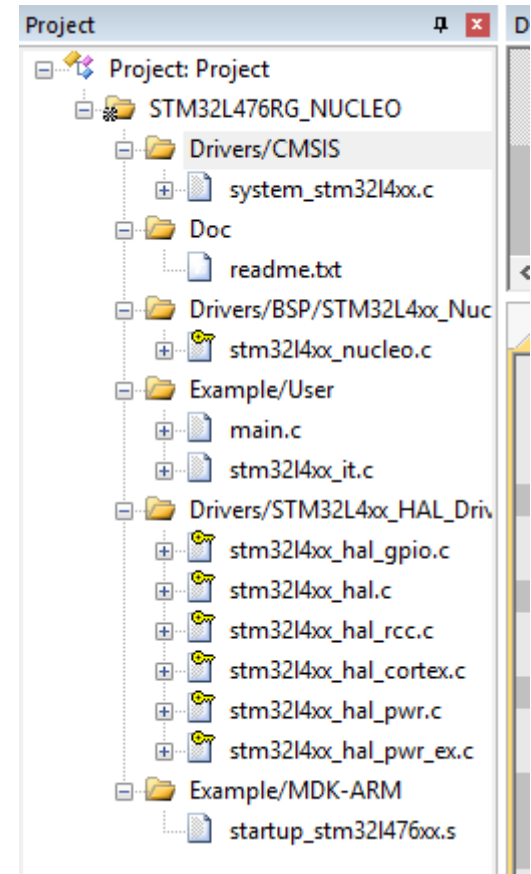
CMSIS - fajlovi

- **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 okviru 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*
 - Ovaj 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

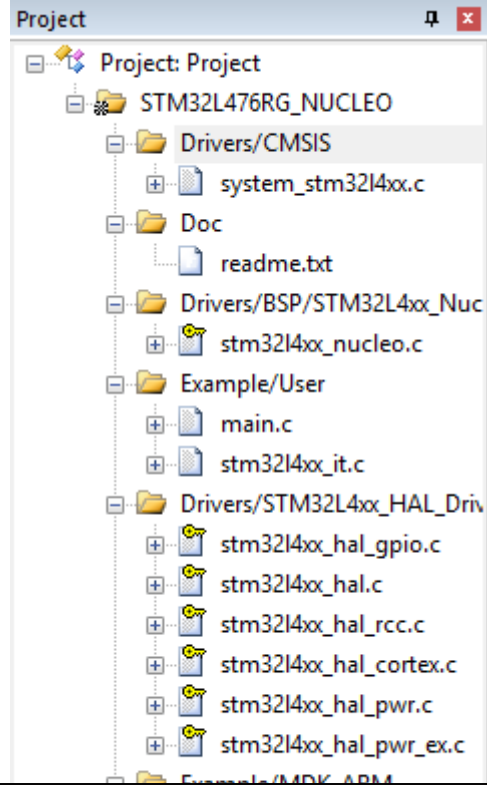


CMSIS - fajlovi

- `stm32l4xx_hal_conf.h` - *Device Peripheral Access Layer Configuration Header File*
 - Ovaj fajl kontroliše korišćenje perifernih 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 klock signala i pozicioniranje PC na labelu `__iar_program_start` gde se nalazi kod za inicijalizaciju promenljivih, nakon čega se poziva *main* funkcija
 - 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 */
```

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
```

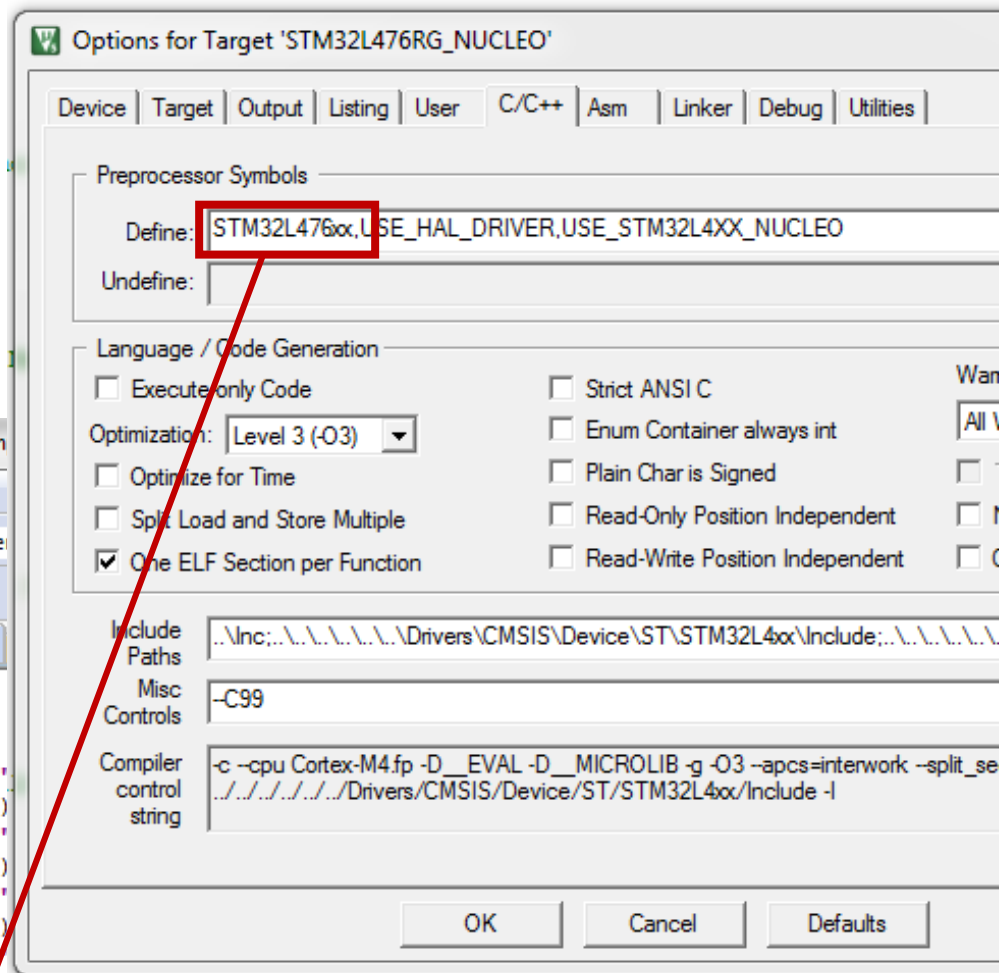
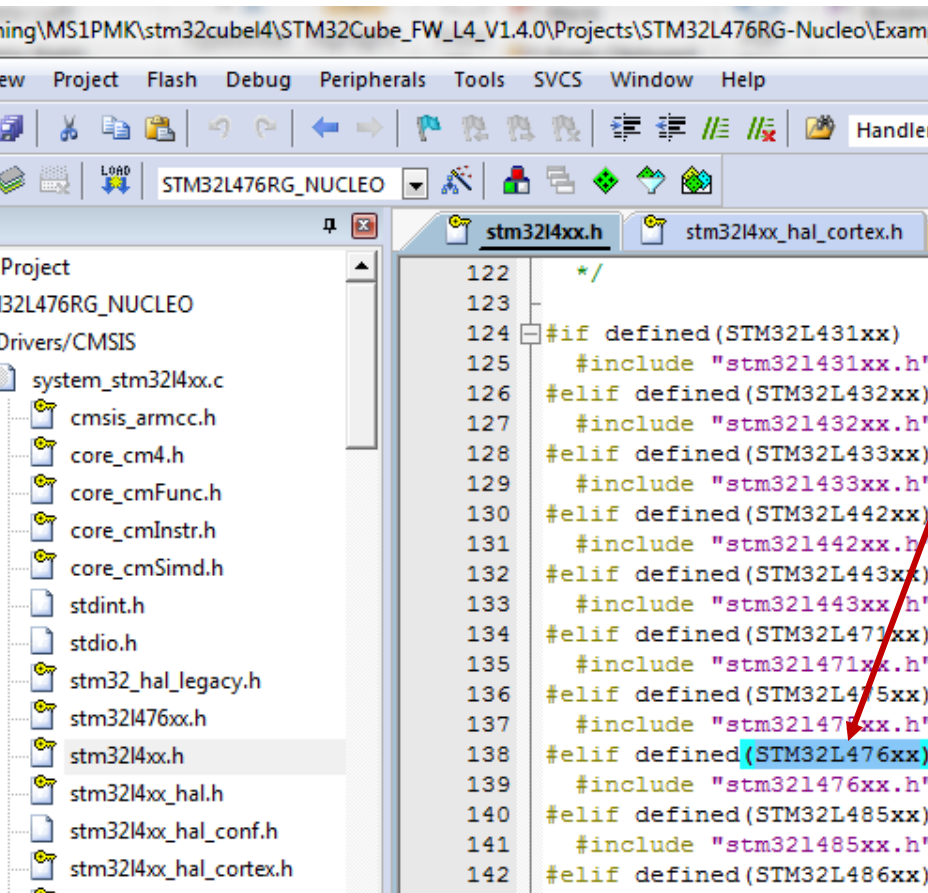
main_X.c

```
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.h

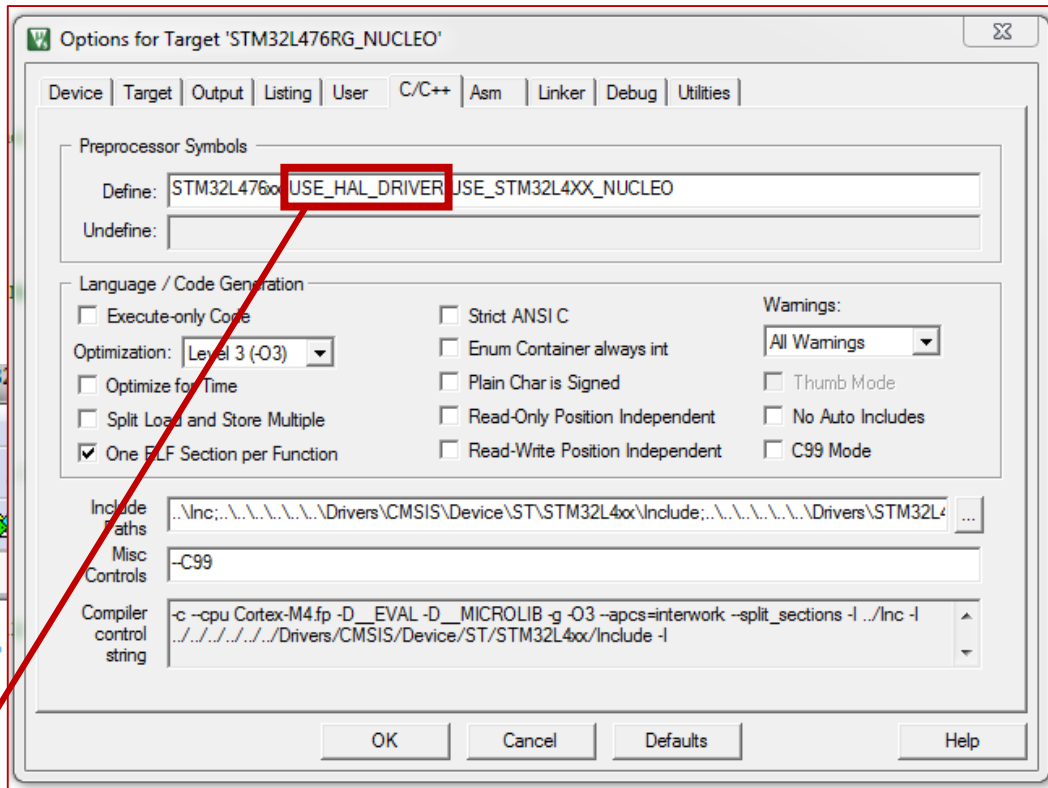
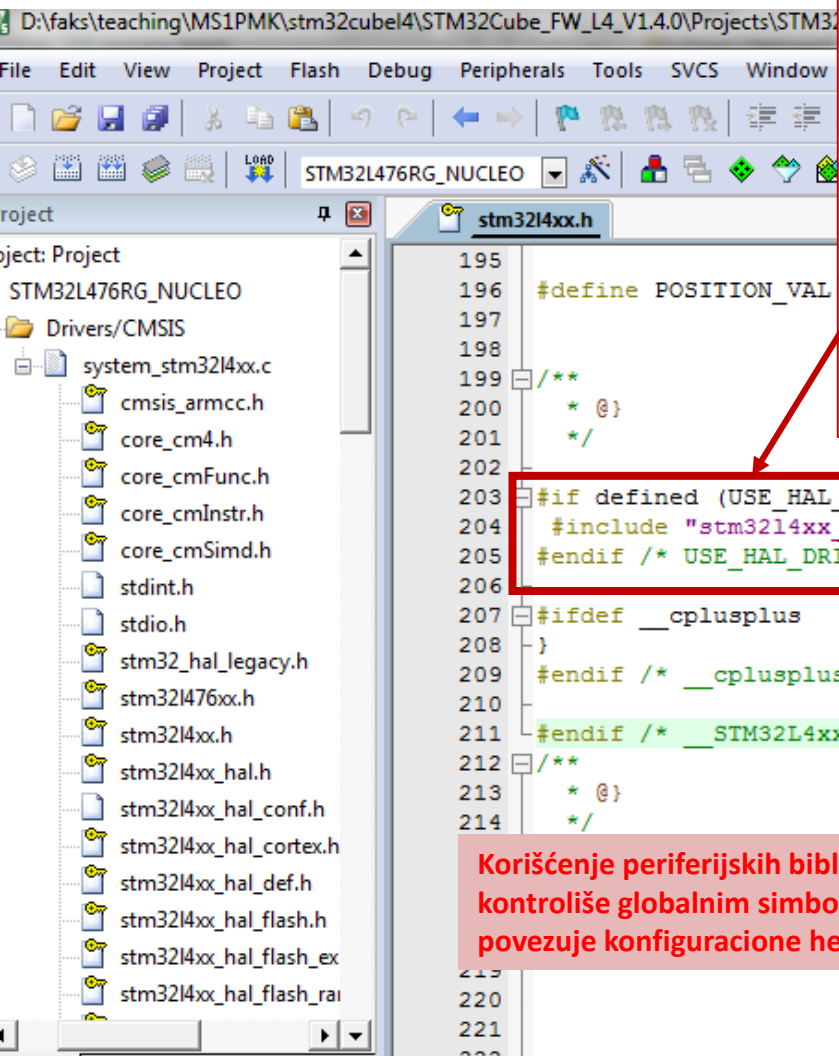
```
57 void SVC_Handler(void);
58 void DebugMon_Handler(void);
59 void PendSV_Handler(void);
60 void SysTick_Handler(void);
```

Opcije projekta

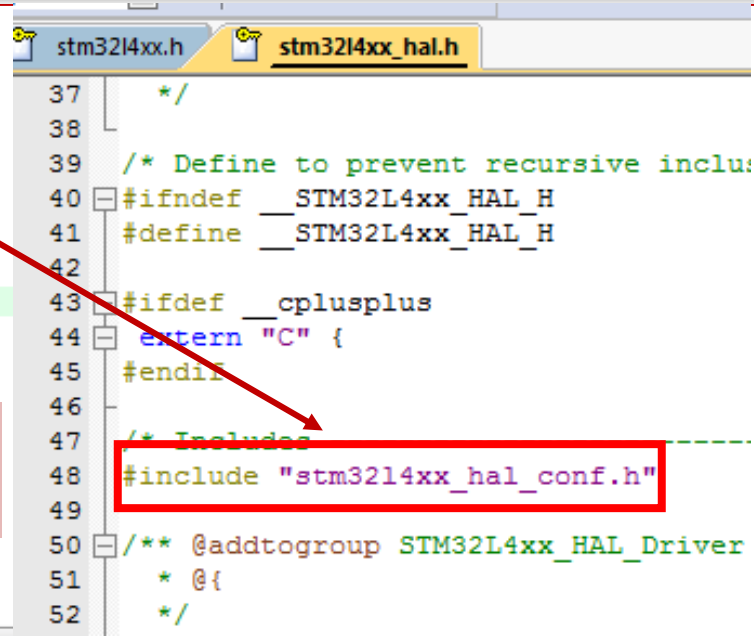


Konretan mikrokontroler se definiše preko globalnog simbola što utiče na deklaracije u glavnom heder fajlu

Opcije projekta



Korišćenje periferijskih biblioteka se kontroliše globalnim simbolom koji povezuje konfiguracione heder fajlove.



Periferijske biblioteke

```
D:\faks\teaching\MS1PMK\stm32cubel4\STM32Cube_FW_L4_V1.4.0\Projects\STM32L476RG-Nucleo\Examples\GPIO\GPIO_IOToggle\MDK-ARM\Proje

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

STM32L476RG_NUCLEO

project

stm324xx_hal_conf.h

58 /* #define HAL_CRC_MODULE_ENABLED */
59 /* #define HAL_Cryp_MODULE_ENABLED */
60 /* #define HAL_DAC_MODULE_ENABLED */
61 /* #define HAL_DFSDM_MODULE_ENABLED */
62 /* #define HAL_DMA_MODULE_ENABLED */
63 #define HAL_FLASH_MODULE_ENABLED
64 /* #define HAL_NAND_MODULE_ENABLED */
65 /* #define HAL_NOR_MODULE_ENABLED */
66 /* #define HAL_SRAM_MODULE_ENABLED */
67 #define HAL_GPIO_MODULE_ENABLED
68 /* #define HAL_I2C_MODULE_ENABLED */
69 /* #define HAL_IWDG_MODULE_ENABLED */
70 /* #define HAL_LCD_MODULE_ENABLED */
71 /* #define HAL_LPTIM_MODULE_ENABLED */
72 /* #define HAL_OPAMP_MODULE_ENABLED */
73 #define HAL_PWR_MODULE_ENABLED
74 /* #define HAL_QSPI_MODULE_ENABLED */

stm324xx_hal_gpio.h

134
135 #ifdef HAL_GPIO_MODULE_ENABLED
136
137 /* Private typedef -----
138 /* Private defines -----
139 /** @defgroup GPIO_Private_Defines GPIO Private Defines
140 * @{
141 */
142 #define GPIO_MODE ((uint32_t)0x00000003)
143 #define ANALOG_MODE ((uint32_t)0x00000008)
144 #define EXTI_MODE ((uint32_t)0x10000000)
145 #define GPIO_MODE_IT ((uint32_t)0x00010000)
146 #define GPIO_MODE_EVT ((uint32_t)0x00020000)
147 #define RISING_EDGE ((uint32_t)0x00100000)
148 #define FALLING_EDGE ((uint32_t)0x00200000)
149 #define GPIO_OUTPUT_TYPE ((uint32_t)0x00000010)
150
```

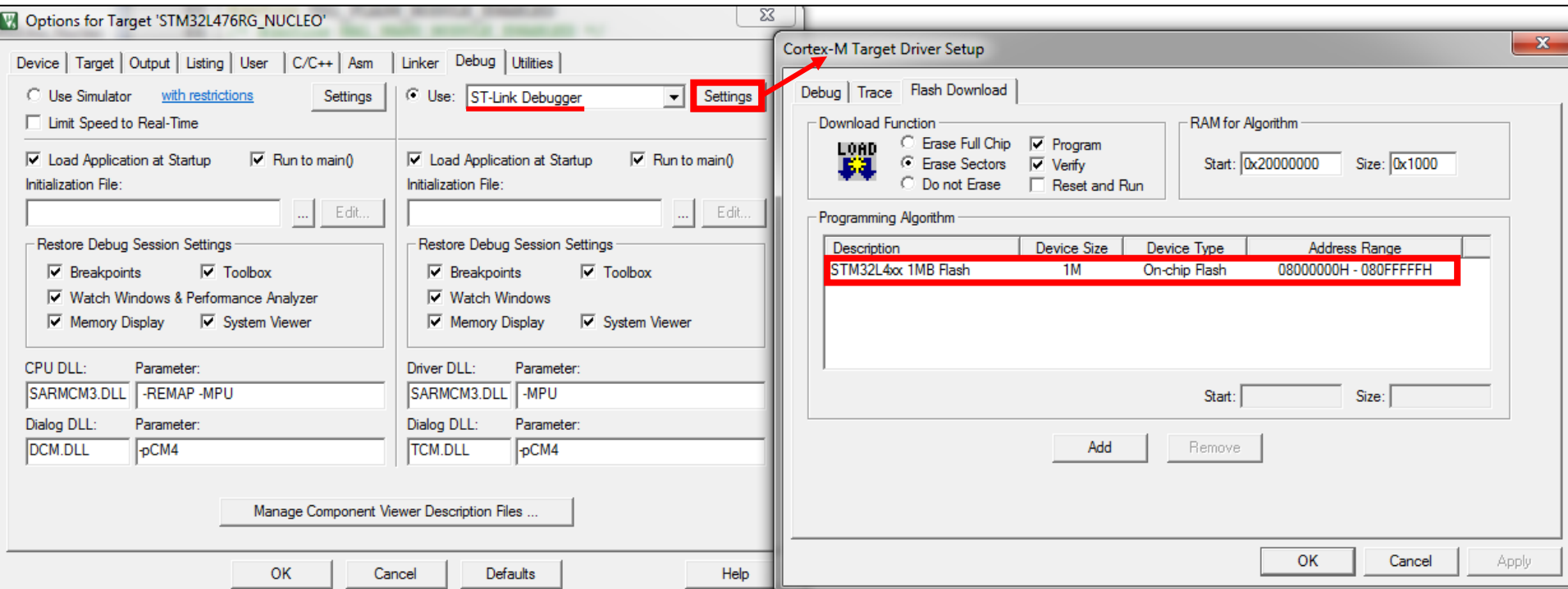
U konfiguracionom fajlu se uključuju biblioteke vendor-specific periferija koje obezbeđuje proizvođač mikrokontrolera.

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

Sistemiški 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

The screenshot shows an IDE window with the following content:

- Project Explorer:** Shows a project named 'STM32L476RG_NUCLEO' with various driver and example files.
- Text Editor (Top):** Displays the assembly code for 'startup_stm32l476xx.s'. Lines 185-195 are highlighted. A red box highlights the assembly code:

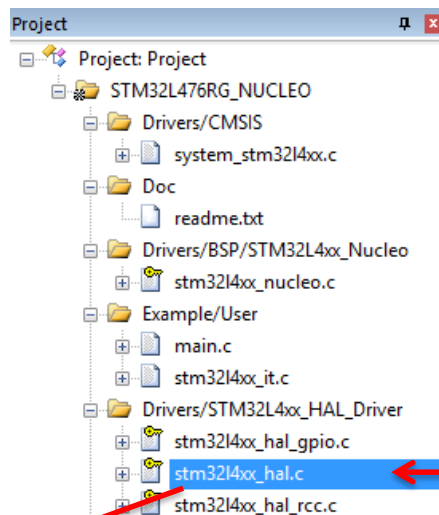
```
185     EXPORT Reset_Handler [WEAK]
186     IMPORT SystemInit
187     IMPORT __main
188
189     LDR R0, =SystemInit
190     BLX R0
191     LDR R0, =__main
192     BX R0
193     ENDP
194
195 ; Dummy Exception Handlers (infinite loops which can be modified)
```
- Text Editor (Bottom):** Displays the C code for 'system_stm32l4xx.c'. Line 198 is highlighted with a red box:

```
197
198 void SystemInit(void)
199 {
200     /* FPU settings -----
201     #if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
202         SCB->CPACR |= ((3UL << 10*2) | (3UL << 11*2) | (3UL << 12*2) | (3UL << 13*2));
203     #endif
204     /* Reset the RCC clock configuration
205     /* Set MSION bit */
206     RCC->CR |= RCC_CR_MSION;
207
208     /* Reset CFGR register */
209     RCC->CFGR = 0x00000000;
210
211     /* Reset HSEON, CSSON , HSION, and PLLON bits */
212     RCC->CR &= (uint32_t)0xEAF6FFFF;
213
214     /* Reset PLLCFGR register */
215     RCC->PLLCFGR = 0x00001000;
216
217     /* Reset HSEBYP bit */
218     RCC->CR &= (uint32_t)0xFFBF0000;
```

Reset vektor poziva SystemInit() funkciju za osnovnu inicijalizaciju, a potom i korisničku main() funkciju

Kroz SystemInit() funkciju proizvođač obezbeđuje osnovnu konfiguraciju Reset Clock Control modula

Tipičan main()



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

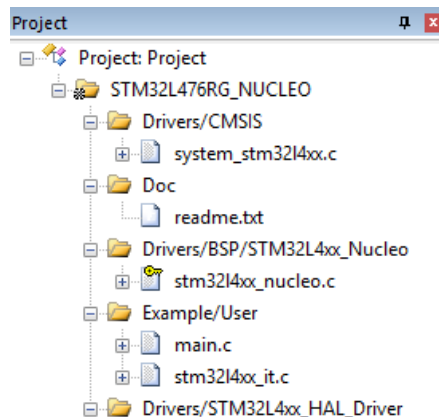
    /* Init the low level hardware */
    HAL_MspInit();

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

```
main.c startup_stm32l476xx.s stm32l4xx_hal.c
65 /*
66 int main(void)
67 {
68 /* This sample code shows how to use GPIO HAL API to toggle LE
69 in an infinite loop. */
70
71 /* STM32L4xx HAL library initialization:
72 - Configure the Flash prefetch
73 - SysTick timer is configured by default as source of tim
74 can eventually implement his proper time base source (a
75 timer for example or other time source), keeping in min
76 duration should be kept 1ms since PPP_TIMEOUT_VALUES ar
77 handled in milliseconds basis.
78 - Set NVIC Group Priority to 4
79 - Low Level Initialization
80 */
81 HAL_Init();
82
83 /* Configure the system clock to 80 MHz */
84 SystemClock_Config();
85
86 /* -1- Enable each GPIO Clock (to be able to program the confi
87 LED2_GPIO_CLK_ENABLE();
88
89 /* -2- Configure IOs in output push-pull mode to drive externe
90 GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
91 GPIO_InitStruct.Pull = GPIO_PULLUP;
92 GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
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 100 ms */
102 HAL_Delay(100);
103 }
104 }
```

Ovo je praktično inicijalizacija Cortex
Periferija

Tipičan main()



```
main.c startup_stm32l476xx.s stm32l4xx_hal.c
123  * @retval None
124  */
125  void SystemClock_Config(void)
126  {
127      RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
128      RCC_OscInitTypeDef RCC_OscInitStruct = {0};
129
130      /* MSI is enabled after System reset, activate PLL with MSI as source
131      RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_MSI;
132      RCC_OscInitStruct.MSIState = RCC_MSI_ON;
133      RCC_OscInitStruct.MSIClockRange = RCC_MSIRANGE_6;
134      RCC_OscInitStruct.MSICalibrationValue = RCC_MSICALIBRATION_DEFAULT;
135      RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
136      RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_MSI;
137      RCC_OscInitStruct.PLL.PLLM = 1;
138      RCC_OscInitStruct.PLL.PLLN = 40;
139      RCC_OscInitStruct.PLL.PLLR = 2;
140      RCC_OscInitStruct.PLL.PLLP = 7;
141      RCC_OscInitStruct.PLL.PLLQ = 4;
142      if(HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
143      {
144          /* Initialization Error */
145          while(1);
146      }
147
148      /* Select PLL as system clock source and configure the HCLK, PCLK1 and
149      clocks dividers */
150      RCC_ClkInitStruct.ClockType = (RCC_CLOCKTYPE_SYSCLK | RCC_CLOCKTYPE_PCLK1 |
151      RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
152      RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
153      RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
154      RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
155      if(HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_4) != HAL_OK)
156      {
157          /* Initialization Error */
158          while(1);
159      }
```

Definiše korisnik

```
main.c startup_stm32l476xx.s stm32l4xx_hal.c
65  */
66  int main(void)
67  {
68      /* This sample code shows how to use GPIO HAL API to toggle LED2
69      in an infinite loop. */
70
71      /* STM32L4xx HAL library initialization:
72      - Configure the Flash prefetch
73      - SysTick timer is configured by default as source of time base
74      can eventually implement his proper time base source (e.g. IWDG or
75      timer for example or other time source), keeping in mind that the
76      duration should be kept 1ms since PPP_TIMEOUT_VALUEs are
77      handled in milliseconds basis.
78      - Set NVIC Group Priority to 4
79      - Low Level Initialization
80      */
81      HAL_Init();
82
83      /* Configure the system clock to 80 MHz */
84      SystemClock_Config();
85
86      /* -1- Enable each GPIO Clock (to be able to program the configuration
87      LED2_GPIO_CLK_ENABLE();
88
89      /* -2- Configure IOs in output push-pull mode to drive external LEDs
90      GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
91      GPIO_InitStruct.Pull = GPIO_PULLUP;
92      GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
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 100 ms */
102          HAL_Delay(100);
103      }
104  }
```

Inicijalizacija periferija I

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

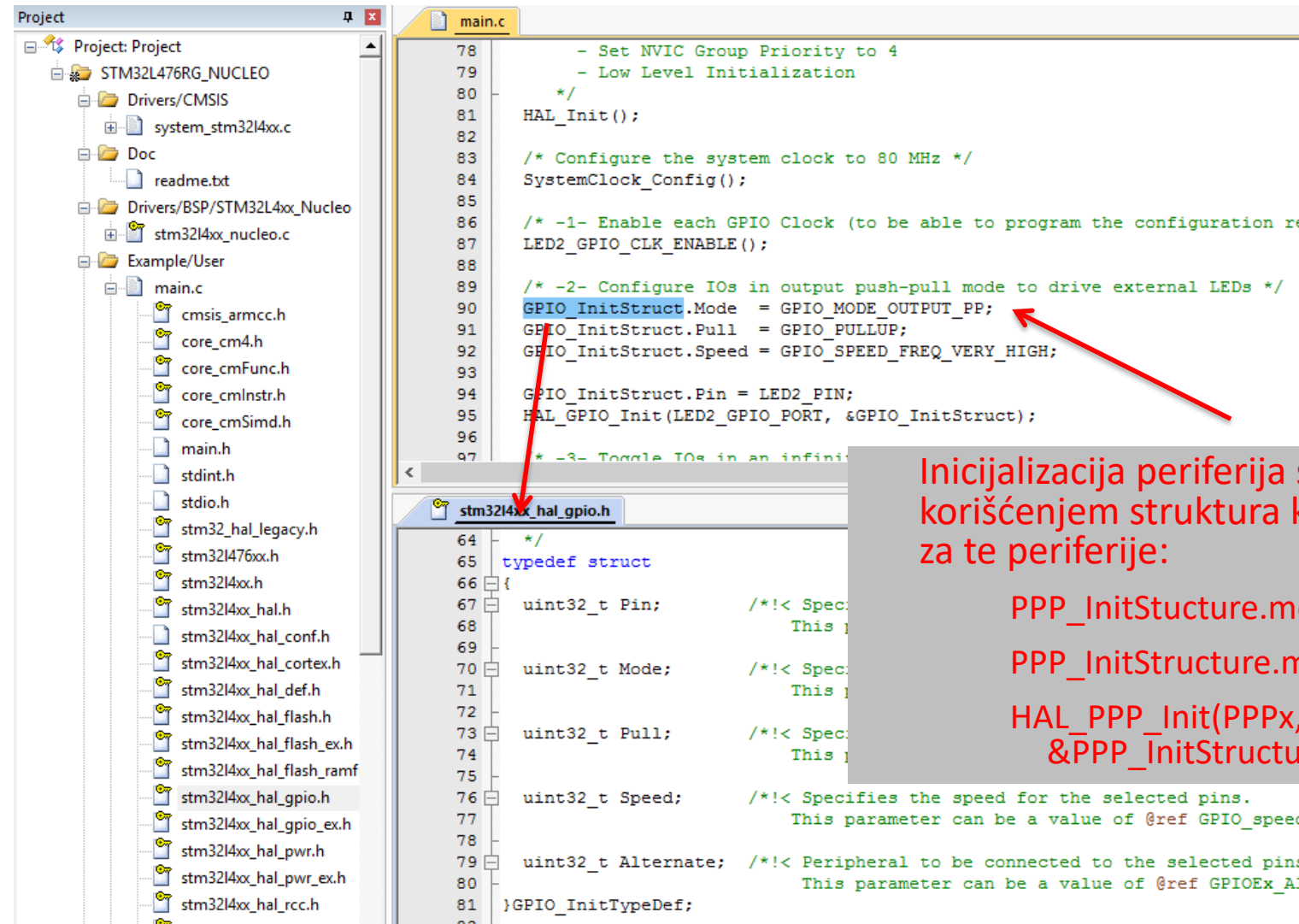
`__HAL_RCC_PPP_CLK_ENABLE()`

```
78     - Set NVIC Group Prioriti
79     - Low Level Initializat
80     */
81     HAL_Init();
82
83     /* Configure the system clock to 80 MHz */
84     SystemClock_Config();
85
86     /* -1- Enable each GPIO Clock (to be able to program the configuration registers) */
87     LED2_GPIO_CLK_ENABLE();
88
89     /* -2- Configure IOs in output push-pull mode to drive external LEDs */
90     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
91     GPIO_InitStruct.Pull = GPIO_PULLUP;
92     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
```

```
114
115 #define LED2_PIN                GPIO_PIN_5
116 #define LED2_GPIO_PORT          GPIOA
117 #define LED2_GPIO_CLK_ENABLE()  HAL_RCC_GPIOA_CLK_ENABLE()
118 #define LED2_GPIO_CLK_DISABLE() __HAL_RCC_GPIOA_CLK_DISABLE()
119
120 #define LEDx_GPIO_CLK_ENABLE(__LED__) do { if((__LED__) == LED2) { LED2_GPIO_CLK_ENABLE(); } } while(0)
121
122 #define LEDx_GPIO_CLK_DISABLE(__LED__) do { if((__LED__) == LED2) { LED2_GPIO_CLK_DISABLE(); } } while(0)
```

```
673
674 #define __HAL_RCC_GPIOA_CLK_ENABLE() do { \
675     __IO uint32_t tmpreg; \
676     SET_BIT(RCC->AHB2ENR, RCC_AHB2ENR_GPIOAEN); \
677     /* Delay after an RCC peripheral clock enabling */ \
678     tmpreg = READ_BIT(RCC->AHB2ENR, RCC_AHB2ENR_GPIOAEN); \
679     UNUSED(tmpreg); \
680     while(0)
681
682 #define __HAL_RCC_GPIOB_CLK_ENABLE() do { \
683     __IO uint32_t tmpreg; \
```

Inicijalizacija periferija II



```
Project
├── Project
│   ├── STM32L476RG_NUCLEO
│   │   ├── Drivers/CMSIS
│   │   │   └── system_stm32l4xx.c
│   │   ├── Doc
│   │   │   └── readme.txt
│   │   ├── Drivers/BSP/STM32L4xx_Nucleo
│   │   │   └── stm32l4xx_nucleo.c
│   │   └── Example/User
│   │       └── main.c
│   │           ├── cmsis_armcc.h
│   │           ├── core_cm4.h
│   │           ├── core_cmFunc.h
│   │           ├── core_cmInstr.h
│   │           ├── core_cmSimd.h
│   │           ├── main.h
│   │           ├── stdint.h
│   │           ├── stdio.h
│   │           ├── stm32_hal_legacy.h
│   │           ├── stm32l476xx.h
│   │           ├── stm32l4xx.h
│   │           ├── stm32l4xx_hal.h
│   │           ├── stm32l4xx_hal_conf.h
│   │           ├── stm32l4xx_hal_cortex.h
│   │           ├── stm32l4xx_hal_def.h
│   │           ├── stm32l4xx_hal_flash.h
│   │           ├── stm32l4xx_hal_flash_ex.h
│   │           ├── stm32l4xx_hal_flash_ramif.h
│   │           ├── stm32l4xx_hal_gpio.h
│   │           ├── stm32l4xx_hal_gpio_ex.h
│   │           ├── stm32l4xx_hal_pwr.h
│   │           ├── stm32l4xx_hal_pwr_ex.h
│   │           └── stm32l4xx_hal_rcc.h
└── main.c
    ├── 78     - Set NVIC Group Priority to 4
    ├── 79     - Low Level Initialization
    ├── 80     */
    ├── 81     HAL_Init();
    ├── 82
    ├── 83     /* Configure the system clock to 80 MHz */
    ├── 84     SystemClock_Config();
    ├── 85
    ├── 86     /* -1- Enable each GPIO Clock (to be able to program the configuration r
    ├── 87     LED2_GPIO_CLK_ENABLE();
    ├── 88
    ├── 89     /* -2- Configure IOs in output push-pull mode to drive external LEDs */
    ├── 90     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    ├── 91     GPIO_InitStruct.Pull = GPIO_PULLUP;
    ├── 92     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    ├── 93
    ├── 94     GPIO_InitStruct.Pin = LED2_PIN;
    ├── 95     HAL_GPIO_Init(LED2_GPIO_PORT, &GPIO_InitStruct);
    ├── 96
    ├── 97     * -3- Toggle IOs in an infinit
    └── stm32l4xx_hal_gpio.h
        ├── 64     */
        ├── 65     typedef struct
        ├── 66     {
        ├── 67     uint32_t Pin;           /*!< Spec
        ├── 68     This p
        ├── 69
        ├── 70     uint32_t Mode;      /*!< Spec
        ├── 71     This p
        ├── 72
        ├── 73     uint32_t Pull;        /*!< Spec
        ├── 74     This p
        ├── 75
        ├── 76     uint32_t Speed;      /*!< Specifies the speed for the selected pins.
        ├── 77     This parameter can be a value of @ref GPIO_speed
        ├── 78
        ├── 79     uint32_t Alternate;    /*!< Peripheral to be connected to the selected pin:
        ├── 80     This parameter can be a value of @ref GPIOEx_A
        ├── 81     }GPIO_InitTypeDef;
```

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

```
PPP_InitStructure.memberX = valX;
PPP_InitStructure.memberY = valY;
HAL_PPP_Init(PPPx,
&PPP_InitStructure);
```

Kako se togluje LED2

```
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);

```

```
3  #define LEDn 1
4
5  #define LED2_PIN GPIO_PIN_5
6  #define LED2_GPIO_PORT GPIOA
7  #define LED2_GPIO_CLK_ENABLE() HAL_RCC_GPIOA
8  #define LED2_GPIO_CLK_DISABLE() HAL_RCC_GPIOA

```

```
2  #define GPIOA ((GPIO_TypeDef *) GPIOA_BASE)
3  #define GPIOB ((GPIO_TypeDef *) GPIOB_BASE)
4  #define GPIOC ((GPIO_TypeDef *) GPIOC_BASE)
5  #define GPIOD ((GPIO_TypeDef *) GPIOD_BASE)
6  #define GPIOE ((GPIO_TypeDef *) GPIOE_BASE)

```

```
102 #define GPIO_PIN_0 ((uint16_t)0x0001)
103 #define GPIO_PIN_1 ((uint16_t)0x0002)
104 #define GPIO_PIN_2 ((uint16_t)0x0004)
105 #define GPIO_PIN_3 ((uint16_t)0x0008)
106 #define GPIO_PIN_4 ((uint16_t)0x0010)
107 #define GPIO_PIN_5 ((uint16_t)0x0020)

```

Podatak će biti 0x0020

```
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;

```

```
#define GPIOA_BASE (AHB2PERIPH_BASE + 0x0000U)
#define GPIOB_BASE (AHB2PERIPH_BASE + 0x0400U)
#define GPIOC_BASE (AHB2PERIPH_BASE + 0x0800U)

```

```
#define FLASH_BASE ((uint32_t)0x08000000U) /*!< FLASH (up to 1 MB) base address */
#define SRAM1_BASE ((uint32_t)0x20000000U) /*!< SRAM1 (up to 96 KB) base address */
#define PERIPH_BASE ((uint32_t)0x40000000U) /*!< Peripheral base address */
#define FMC_BASE ((uint32_t)0x60000000U) /*!< FMC base address */
#define SRAM2_BASE ((uint32_t)0x10000000U) /*!< SRAM2 (32 KB) base address */

```

```
/*!< 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