Vhodno izhodne naprave

Laboratorijska vaja 4 - VP 3
STM32-CubeIDE projekt, VComPort, GPIO (LED, tipka)

© Rozman,Škraba, FRI

VIN projekt - VP3: STM32-CubeIDE projekt, VComPort, GPIO (LED, tipka)

- STM32F4 Discovery board in ostale platforme
- STM32 CubeIDE
- Osnovni projekt CubeIDE in CubeMX
  - CubeMX
  - USB VComPort
  - GPIO – nivoji programiranja
  - GPIO – tipka, LED diode
  - GPIO – PWM, LED diode

© Rozman,Škraba, FRI

VIN projekt - VP3: STM32-CubeIDE projekt, VComPort, GPIO (LED, tipka)

- STM32F4 Discovery board in ostale platforme

- STM32 CubeIDE

- Osnovni projekt CubeIDE in CubeMX
  - CubeMX
  - USB VComPort
  - GPIO – nivoji programiranja
  - GPIO – tipka, LED diode
  - GPIO – PWM, LED diode

© Rozman,Škraba, FRI

VIN Projekt – Osnovna platforma

STM32F407 ST Discovery

STM Discovery F4 (Cortex M4)

•STM32F407VGT6 microcontroller featuring 32-bit Arm® Cortex®-M4 with FPU core, 1-Mbyte Flash memory and 192-Kbyte RAM in an LQFP100 package

•USB OTG FS

•ST MEMS 3-axis accelerometer

•ST-MEMS audio sensor omni-directional digital microphone

•Audio DAC with integrated class D speaker driver

•User and reset push-buttons

•Eight LEDs:

- •LD1 (red/green) for USB communication
- •LD2 (red) for 3.3 V power on
- •Four user LEDs, LD3 (orange), LD4 (green), LD5 (red) and LD6 (blue)

•Board connectors:

- •USB with Micro-AB
- •Stereo headphone output jack
- •2.54 mm pitch extension header for all LQFP100 I/Os for quick connection to prototyping board and easy probing

•External application power supply: 3 V and 5 V

STM32

 © Rozman,Škraba, FRI

VIN Projekt – Ostale platforme

STM32F769 ST Discovery

STM Discovery F7 (Cortex M7)

•STM32F769NIH6 microcontroller featuring 2 Mbytes of Flash memory and 512+16+4 Kbytes of RAM, in BGA216 package

•On-board ST-LINK/V2-1 supporting USB reenumeration capability

•USB ST-LINK functions: virtual COM port, mass storage, debug port

•4" capacitive touch LCD display with MIPI® DSI connector (on STM32F769I-DISCO only)

•SAI audio codec

•Two audio line jacks, one for input and one for output

•Stereo speaker outputs

•Four ST MEMS microphones on DFSDM inputs

•Two SPDIF RCA input and output connectors

•Two push-buttons (user and reset)

•512-Mbit Quad-SPI Flash memory

•128-Mbit SDRAM

•Connector for microSD card

•Wi-Fi or Ext-EEP daughterboard connector

•USB OTG HS with Micro-AB connector

•Ethernet connector compliant with IEEE-802.3-2002

•Arduino™Uno V3 connectors

STM32

aws qualified device Internet of things

arm MBED Enabled

https://www.st.com/en/evaluation-tools/32f769idiscovery.html

© Rozman,Škraba, FRI

ST Discovery STM32MP157C

STM Discovery MP1 (2xCortex A7 + 1xCortex M4)

•STM32MP157 Arm®-based ***dual Cortex®-A7 32 bits + Cortex®-M4 32 bits MPU*** in TFBGA361 package

•4-Gbit DDR3L, 16 bits, 533 MHz

•1-Gbps Ethernet (RGMII) compliant with IEEE-802.3ab

•USB OTG HS

•Audio codec

•4 user LEDs

•2 user and reset push-buttons, 1 wake-up button

•5 V / 3 A USB Type-CTM power supply input (not provided

•Board connectors:Ethernet RJ454 × USB Host Type-AUS Type-CTM DRPMIPI DSISMHDMI®Stereo headset jack including analog microphone inputmicroSDTM cardGPIO expansion connector (Raspberry Pi® shields capability)

•ARDUINO® Uno V3 expansion connectors

•STM32CubeMP1 and full mainline open-source Linux® STM32 MPU OpenSTLinux Distribution (such as STM32MP1Starter) software and examples

•4" TFT 480 × 800 pixels with LED backlight, MIPI DSISM interface, and capacitive touch panel

•Wi-Fi® 802.11b/g/n

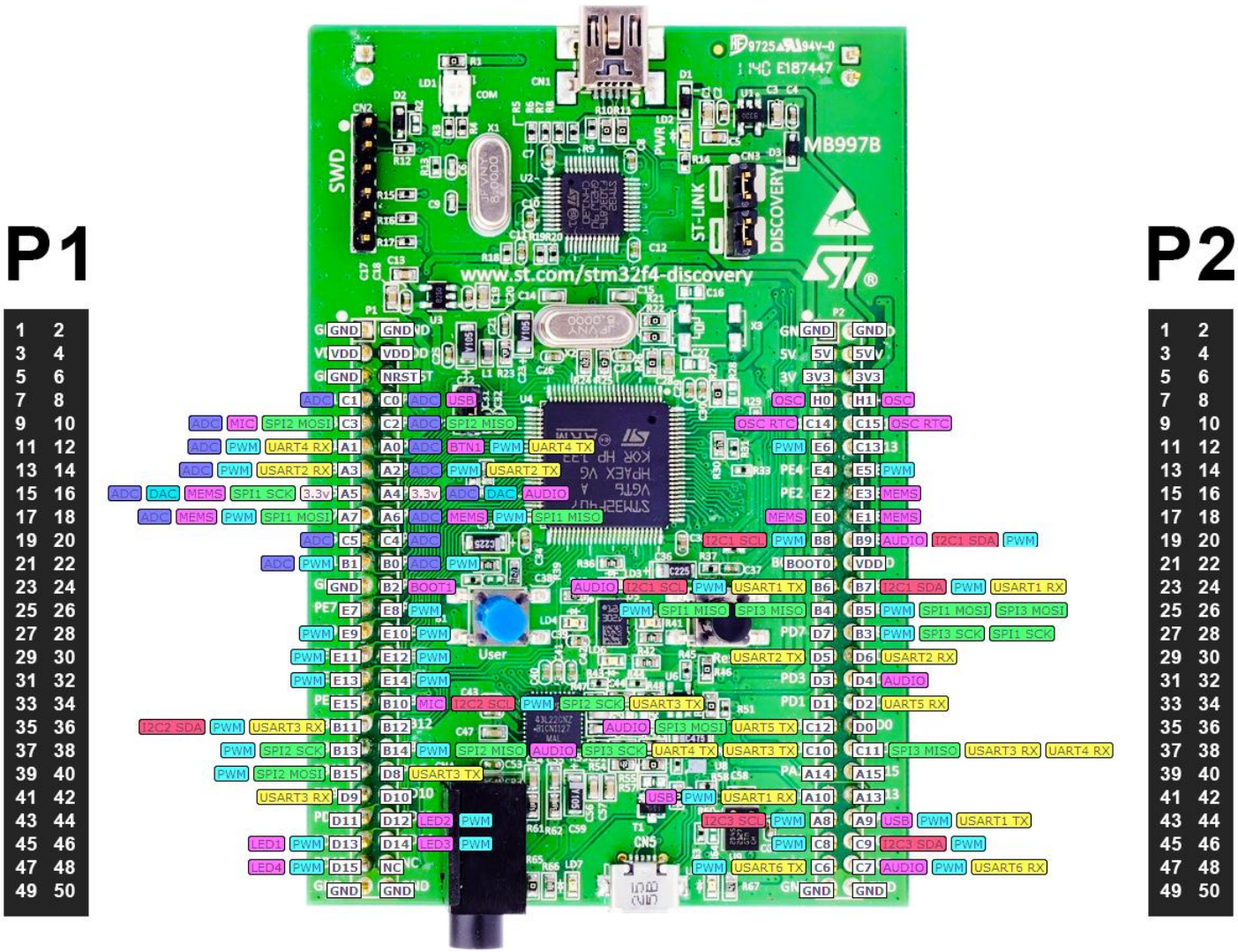
•Bluetooth® Low Energy 4.1

STM32MP1

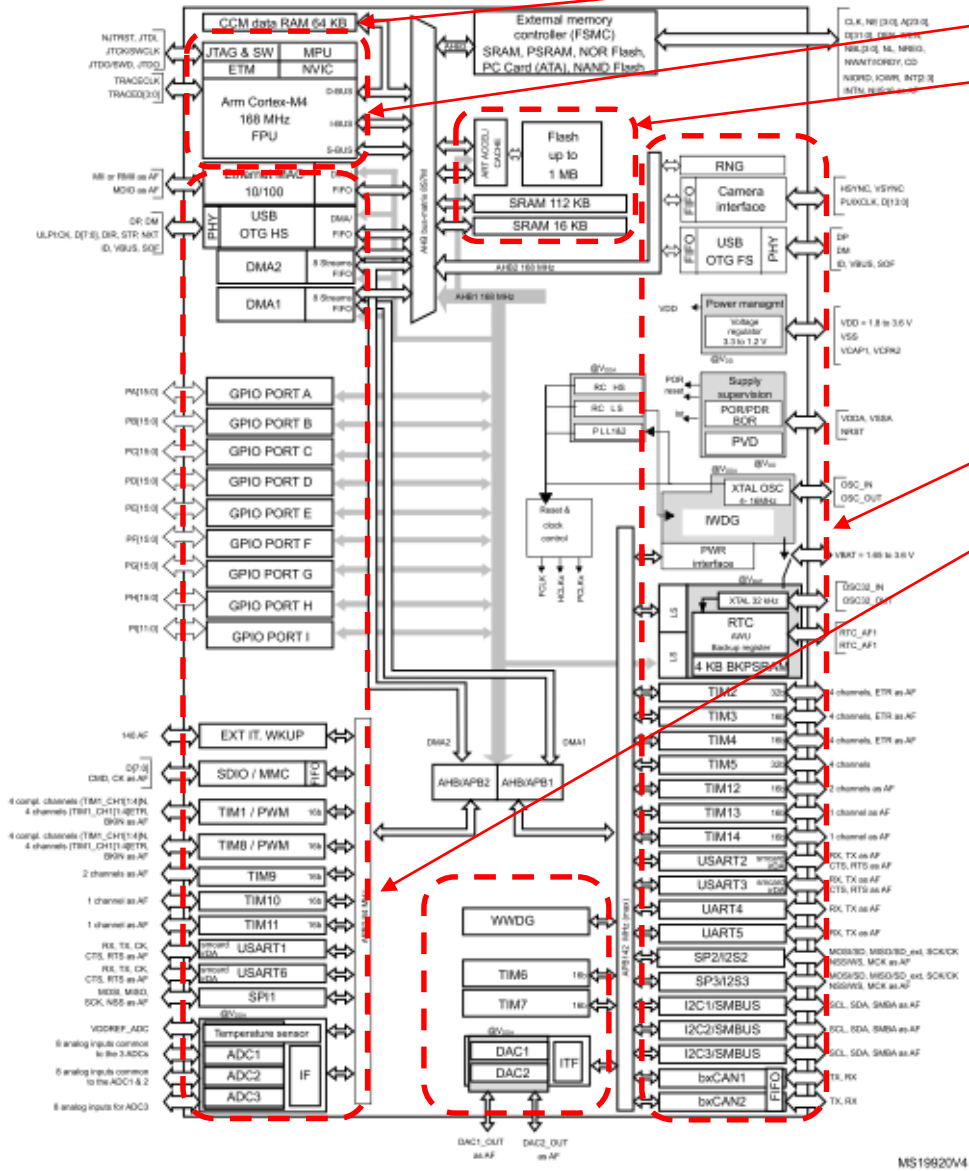
STM32

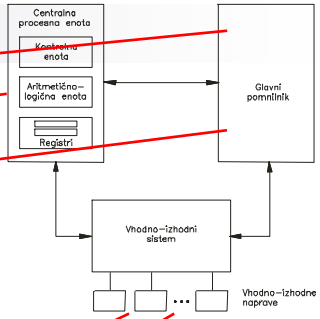
STM32F4DISCOVERY

3.3V !!!



STM32F407VG





© Rozman,Škraba, FRI

ARM Cortex M – ISA

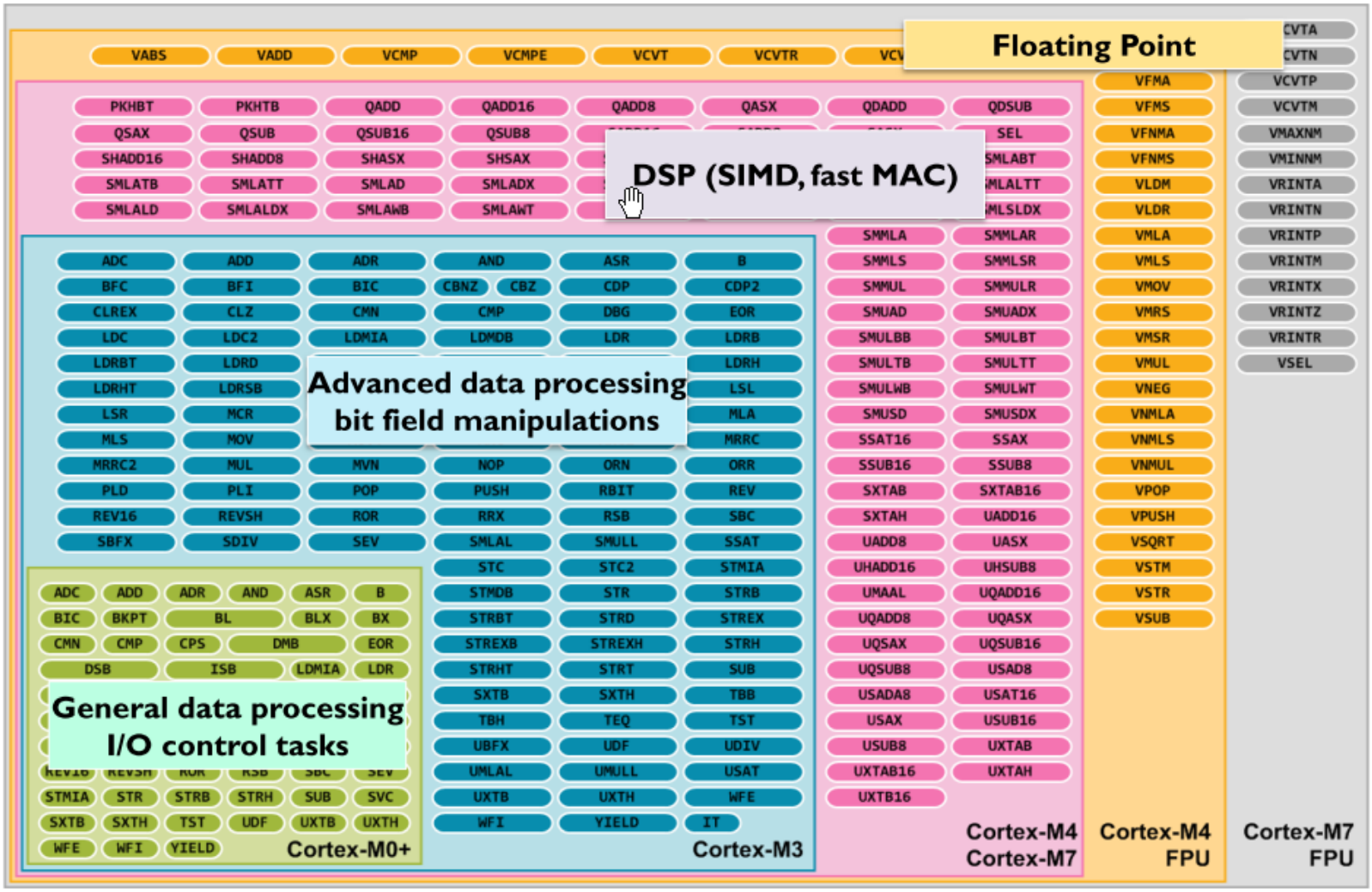
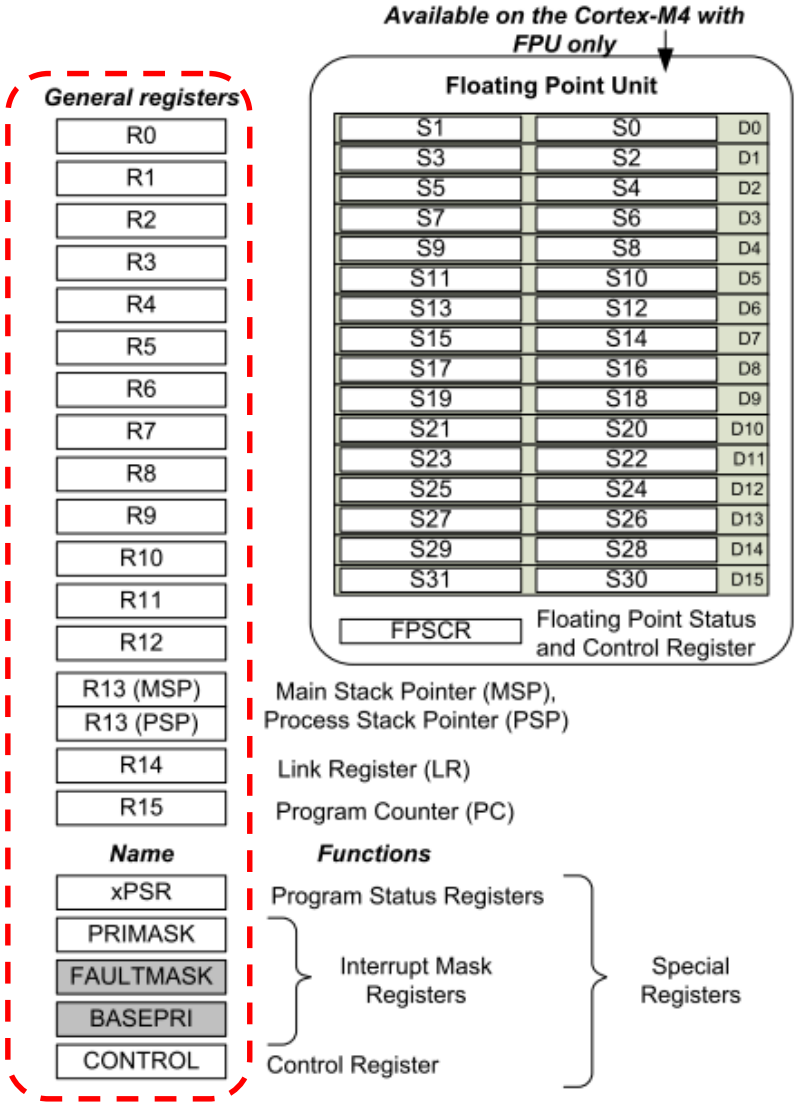


Figure 3: Instruction Set support in the Cortex-M processors

ARM Cortex M – Programski model



Delo na STM32F4 razvojnem sistemu

Lastni viri :

https://github.com/LAPSyLAB/STM32F4_Docs_and_Examples

https://github.com/LAPSyLAB/ORLab-STM32

https://github.com/LAPSyLAB/STM32F4_Discovery_VIN_Projects

VIN projekt - VP3: STM32-CubeIDE projekt, VComPort, GPIO (LED, tipka)

- STM32F4 Discovery board in ostale platforme
- STM32 CubeIDE
- Osnovni projekt CubeIDE in CubeMX
  - CubeMX
  - USB VComPort
  - GPIO – nivoji programiranja
  - GPIO – tipka, LED diode
  - GPIO – PWM, LED diode

© Rozman,Škraba, FRI

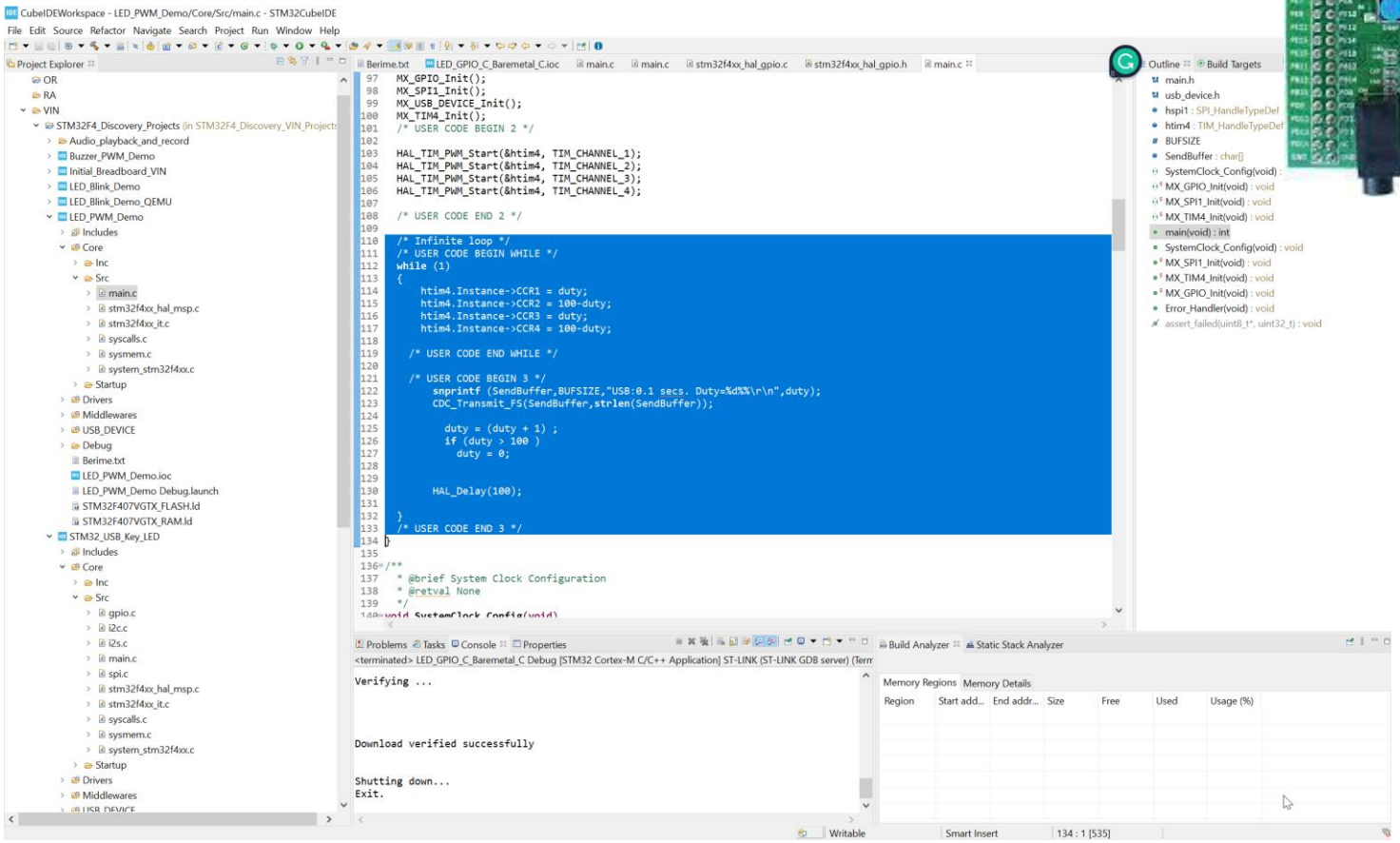
Delo na STM32F4 razvojnem sistemu

Priključitev :

- **Mini USB** priklop na **krajši stranici**, svetita rdeči **LED** diodi

STM32 CubeIDE

- https://www.st.com/en/development-tools/stm32cubeide.html

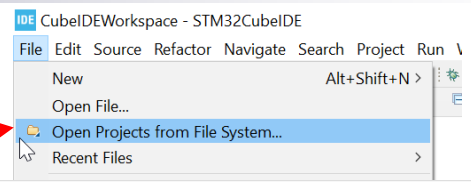


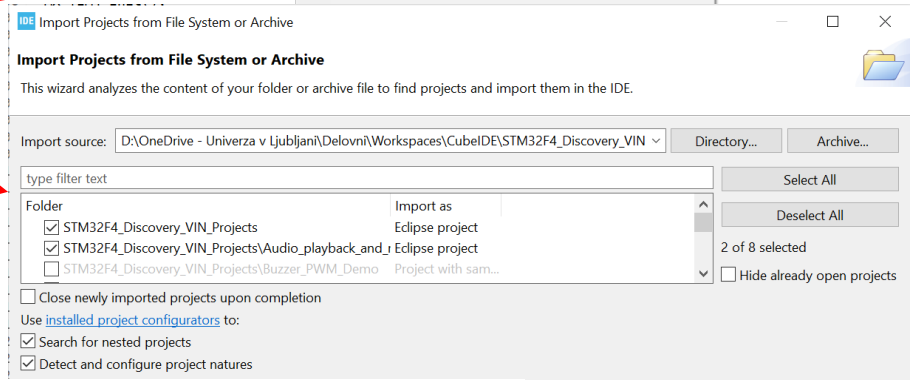
© Rozman,Škraba, FRI

Delo na STM32F4 razvojnem sistemu

Vzpostavitev začetnega projekta :

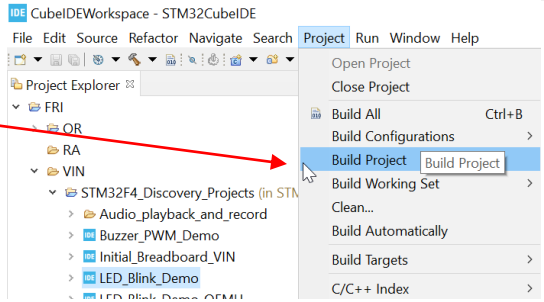
- **Uvoz obstoječega**
  - Open projects from File System
  - Select project(s)
- **Nov projekt Cube MX ->**
  (v nadaljevanju)

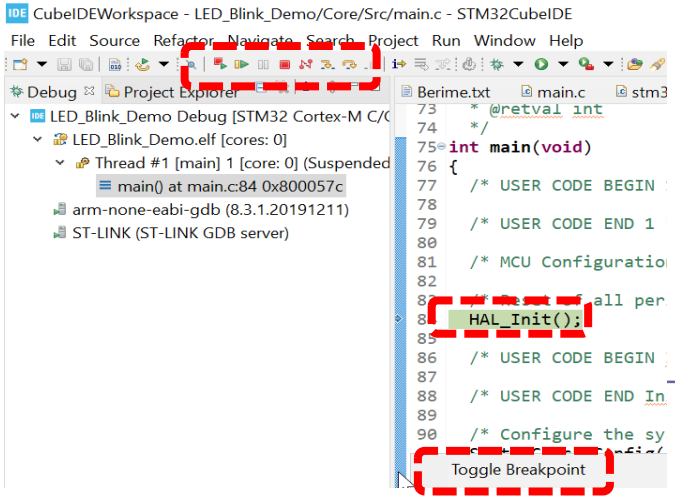


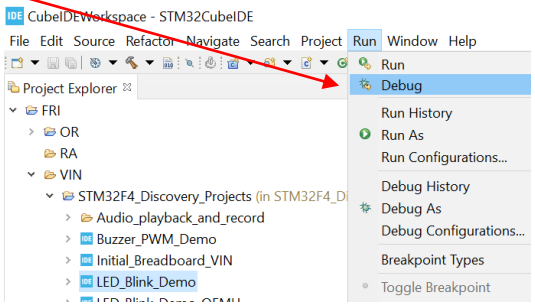


Prevajanje, zagon :

- Project -> Build Project
- Run -> Debug
- Step (Into,Over), Breakpoints







© Rozman,Škraba, FRI

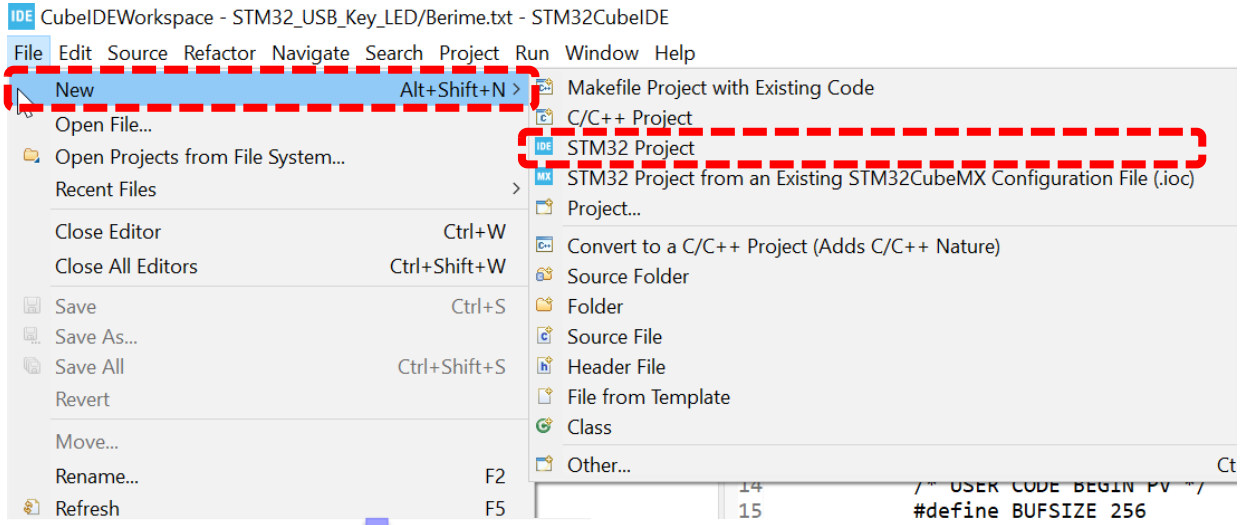
VIN projekt - VP3: STM32-CubeIDE projekt, VComPort, GPIO (LED, tipka)

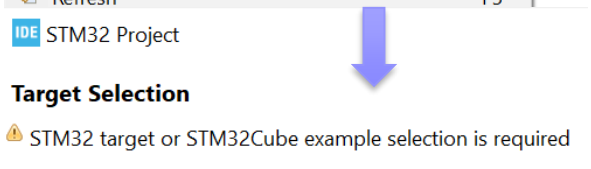
- STM32F4 Discovery board in ostale platforme
- STM32 CubeIDE
- Osnovni projekt CubeIDE in CubeMX
  - CubeMX
  - USB VComPort
  - GPIO – nivoji programiranja
  - GPIO – tipka, LED diode
  - GPIO – PWM, LED diode

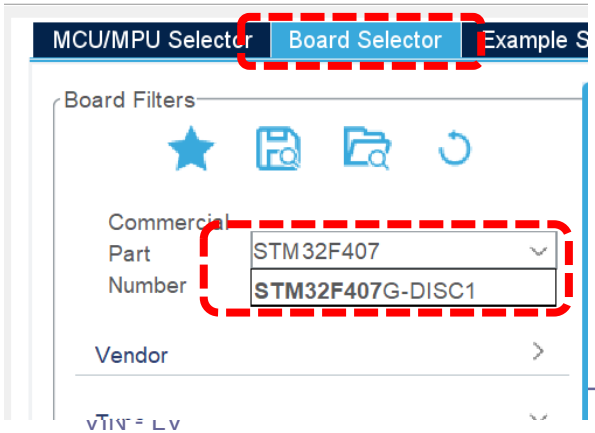
© Rozman,Škraba, FRI

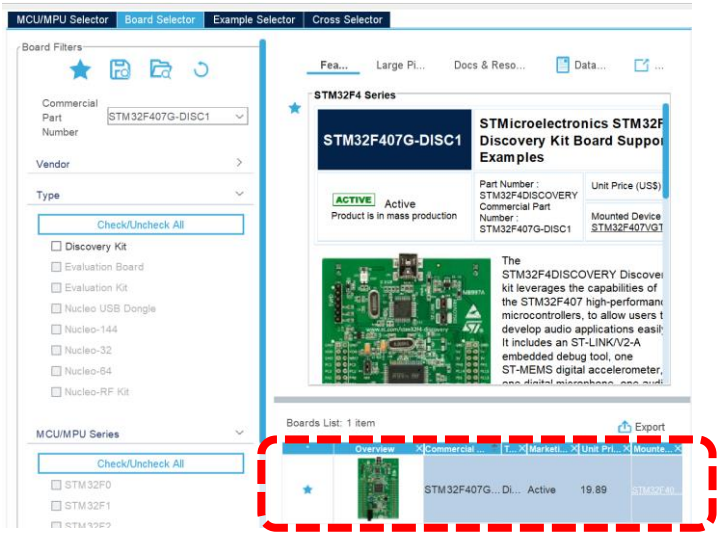
Osnovni projekt CubeIDE – CubeMX

Nov projekt :





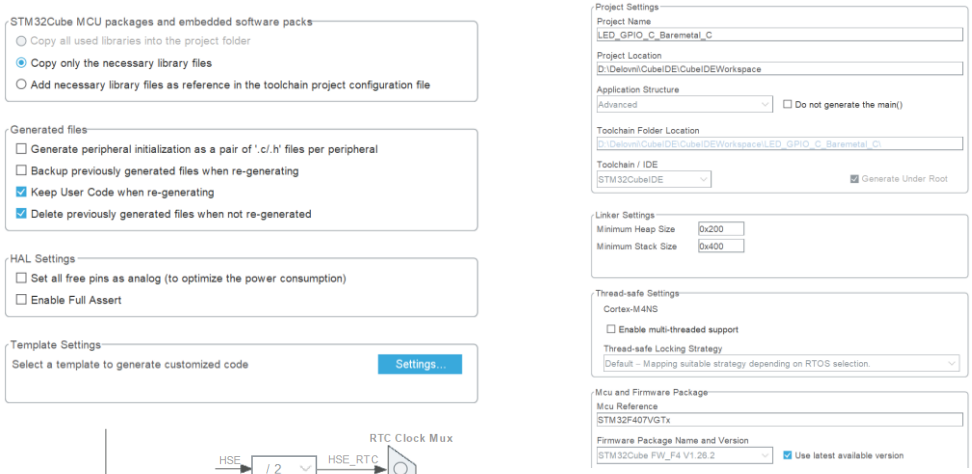


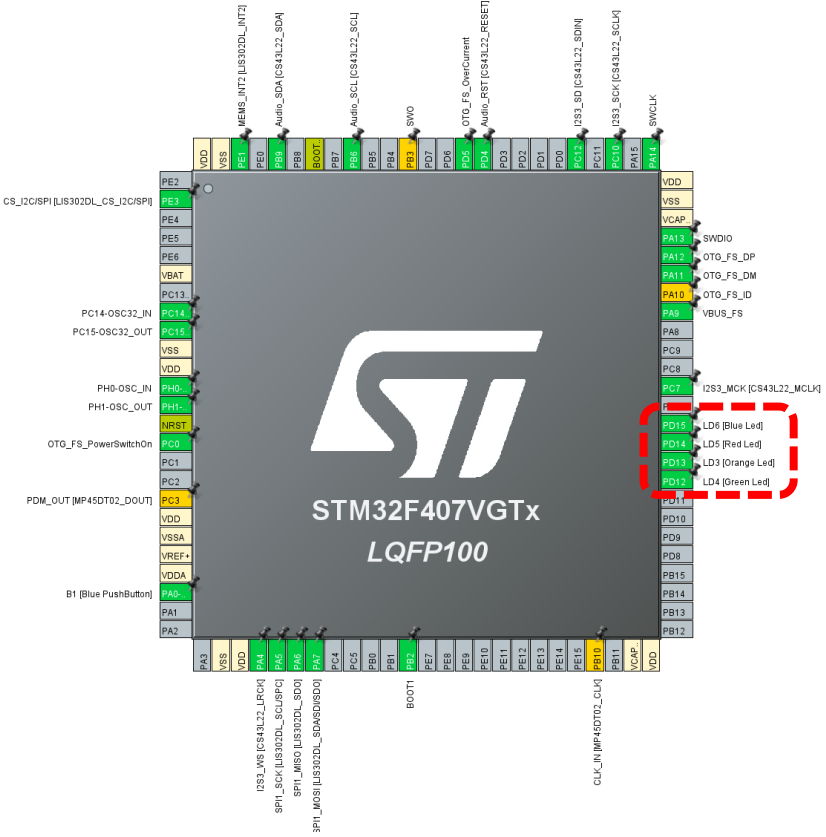


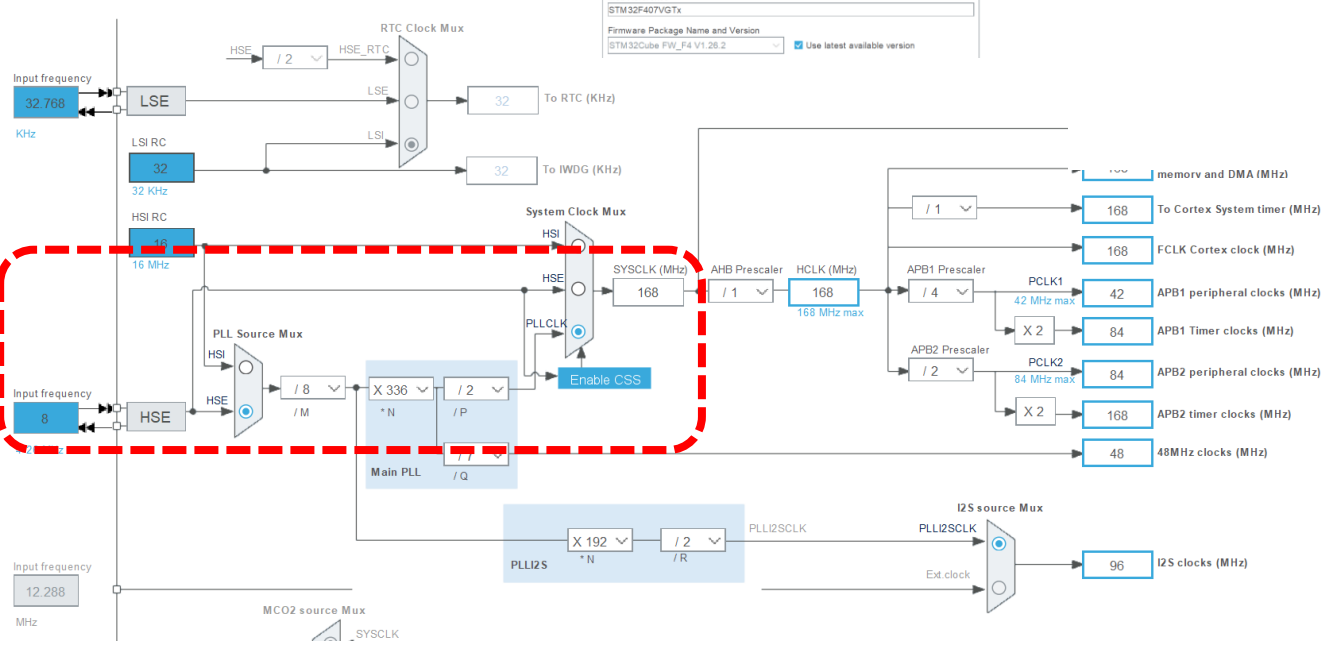
© Rozman,Škraba, FRI

Osnovni projekt CubeIDE – CubeMX

Konfiguracija : priključki, knjižnice



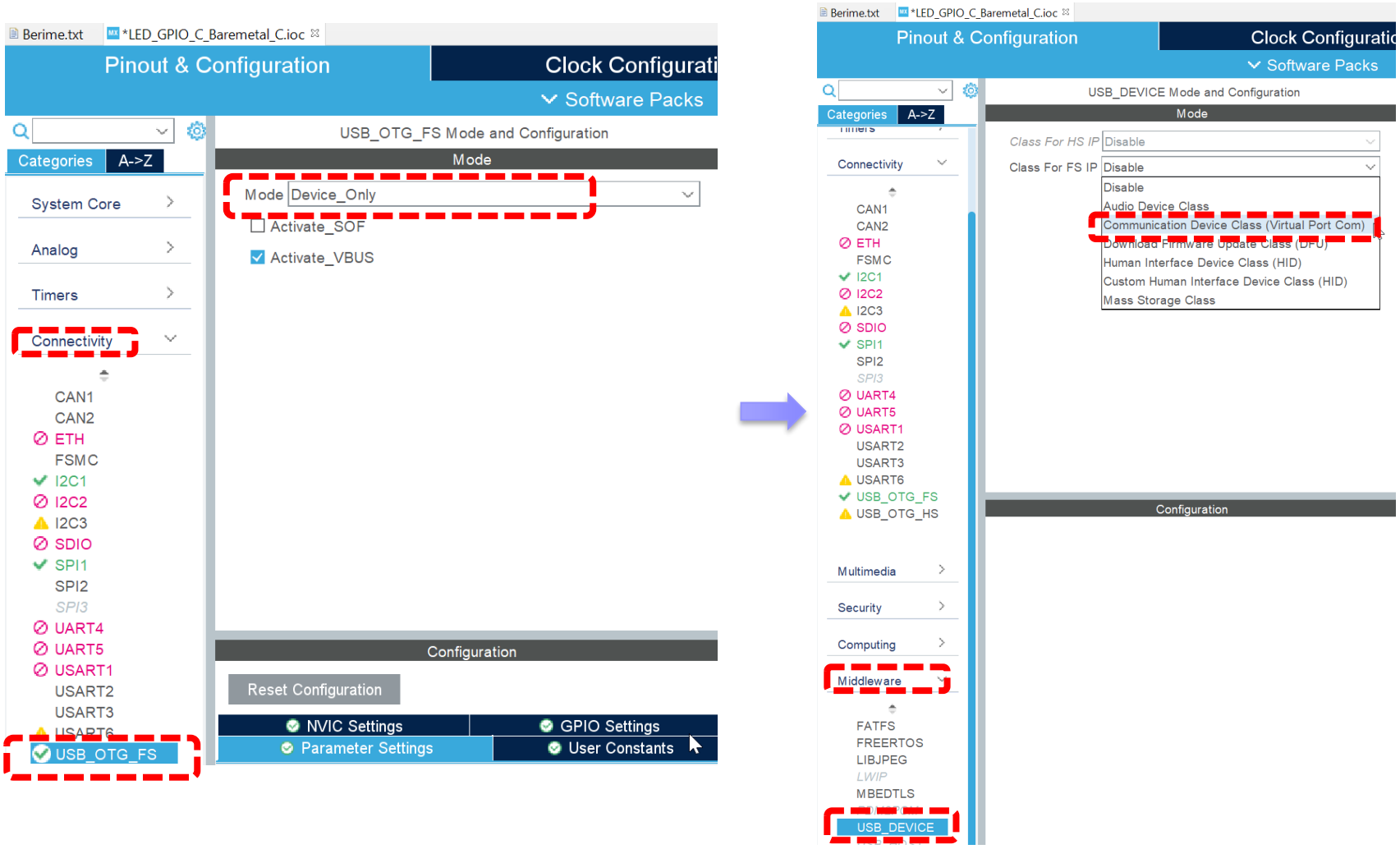




© Rozman,Škraba, FRI

Osnovni projekt CubeIDE – USB Virtual COM Port

Konfiguracija : USB Device,CDC Class = Virtual COM Port

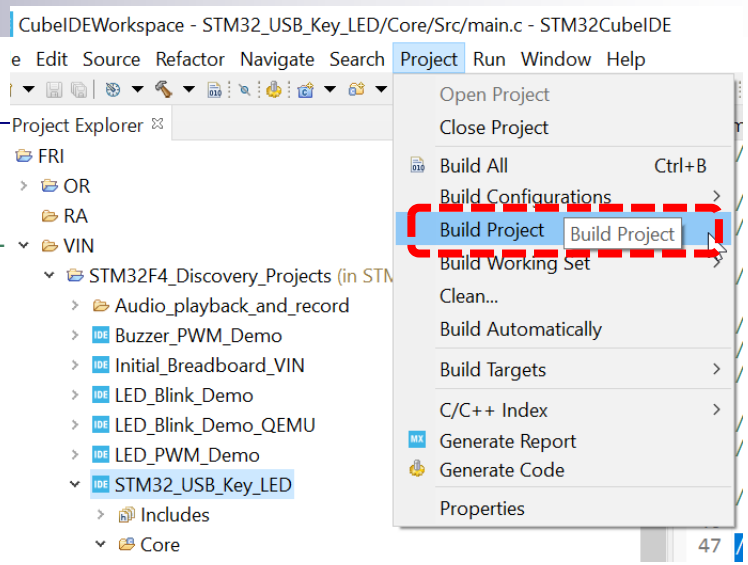


 © Rozman,Škraba, FRI

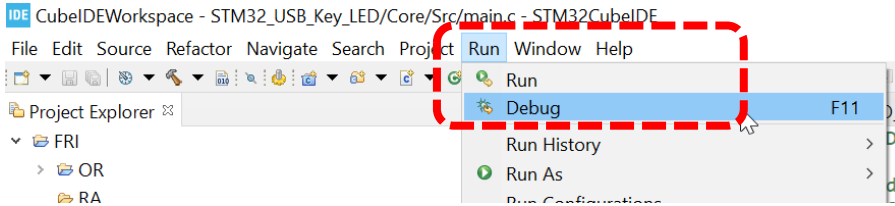
Osnovni projekt CubeIDE – USB Virtual COM Port

Program : za pošiljanje po USB Virtual COM Port

```
/* Private variables ---------------------------------------------------------*/

/* USER CODE BEGIN PV */
#define     BUFSIZE 256
char        SendBuffer[BUFSIZE];
int         Counter;
/* USER CODE END PV */
```

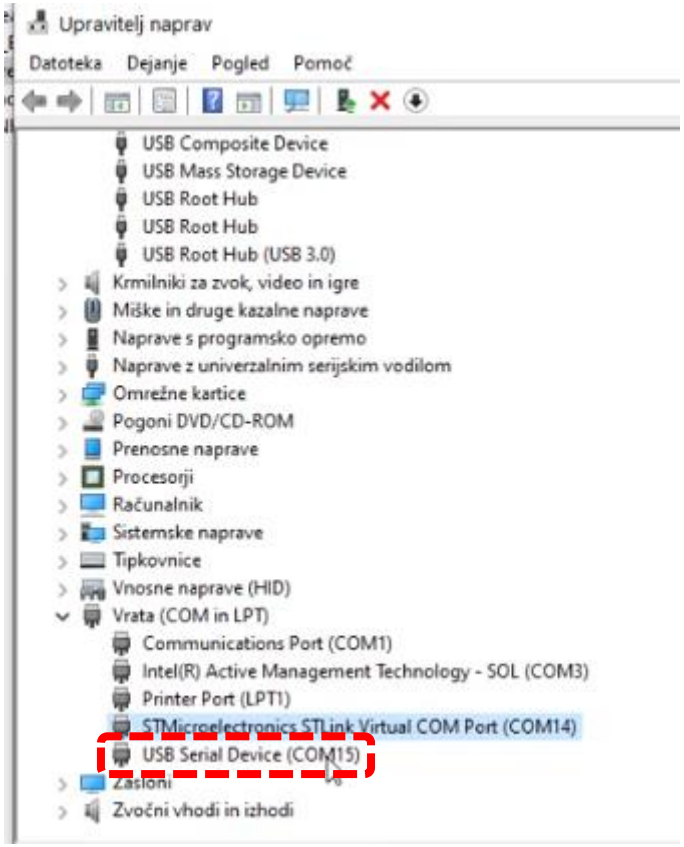


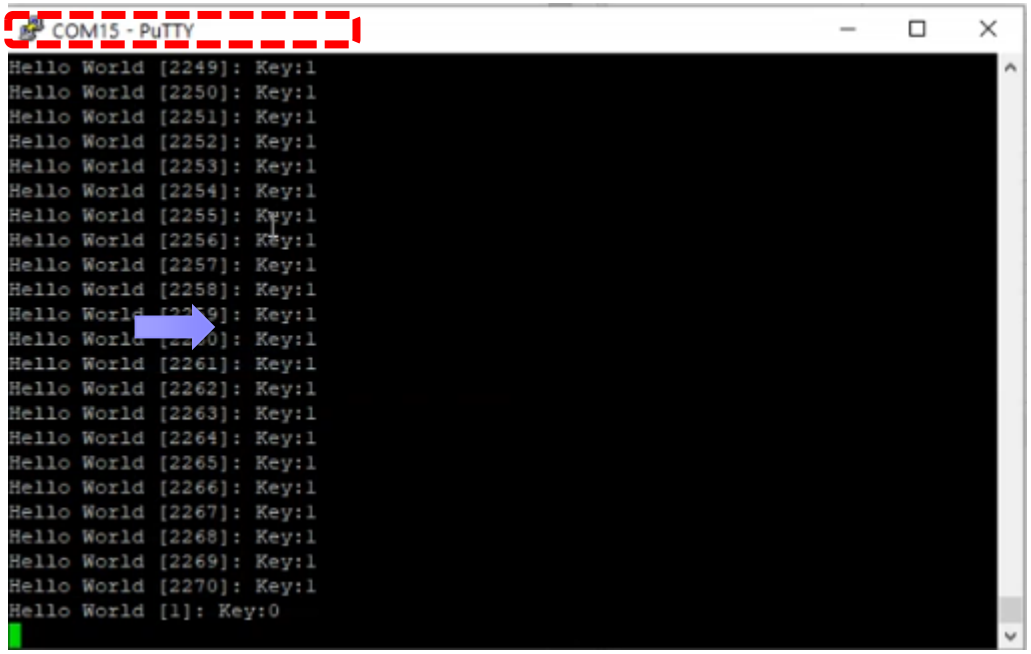

```
/* Infinite loop */
 /* USER CODE BEGIN WHILE */
 while (1)
 {
         snprintf(SendBuffer,BUFSIZE,"Hello World [%d]\r\n",Counter++);
         CDC_Transmit_FS(SendBuffer,strlen(SendBuffer));
 /* USER CODE END WHILE */

 /* USER CODE BEGIN 3 */
         HAL_Delay(1000);
 }
 /* USER CODE END 3 */
```



© Rozman,Škraba, FRI

Program : sprejem na PC strani (povezava z Micro-USB kablom)





Osnovni projekt CubeIDE – GPIO – nivoji programiranja

Baremetal - zbirnik

```
INIT_IO:
  push {r5, r6, lr}
    // Enable GPIOD Peripheral Clock (bit 3 in AHB1ENR register)
    ldr r6, = RCC_AHB1ENR       // Load peripheral clock reg address to r6
    ldr r5, [r6]                // Read its content to r5
    orr r5, 0x00000008          // Set bit 3 to enable GPIOD clock
    str r5, [r6]                // Store result in peripheral clock register

    // Make GPIOD Pin12 as output pin (bits 25:24 in MODER register)
    ldr r6, =GPIOD_BASE       // Load GPIOD BASE address to r6
    ldr r5, [r6,#GPIOD_MODER]  // Read GPIOD_MODER content to r5
    and r5, 0x00FFFFFF          // Clear bits 31-24 for P12-15
    orr r5, 0x55000000          // Write 01 to bits 31-24 for P12-15
    str r5, [r6]                // Store result in GPIOD MODER register
  pop {r5, r6, pc}


LED_ON:
    push {r5, r6, lr}
    // Set GPIOD Pins to 1 (through BSSR register)
    ldr     r6, =GPIOD_BASE       // Load GPIOD BASE address to r6
    mov     r5, #LEDs_ON
    str     r5, [r6,#GPIOD_BSSR] // Write to BSRR register
    pop {r5, r6, pc}

LED_OFF:
    push {r5, r6, lr}
    // Set GPIOD Pins to 0 (through BSSR register)
    ldr     r6, =GPIOD_BASE       // Load GPIOD BASE address to r6
    mov     r5, #LEDs_OFF
    str     r5, [r6,#GPIOD_BSSR] // Write to BSRR register
    pop {r5, r6, pc}
```

https://github.com/LAPSyLAB/ORLab-STM32/tree/main/GPIO_LEDs

Baremetal - C

```
  /* USER CODE BEGIN 2 */


  RCC->AHB1ENR |= 0x08;
// Enable clock for GPIOD
  GPIOD->MODER |= 0x01000000;        //
MODE Register: bit 12 == out

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    GPIOD->ODR ^= 0x1000;            //
Toggle PD12

/* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    for (int i=0; i<0x1000000; i++) {};
// waste some time
  }
  /* USER CODE END 3 */
```

https://github.com/LAPSyLAB/STM32F4_Discovery_VIN_Projects/tree/main/LED_GPIO_C_Baremetal_C

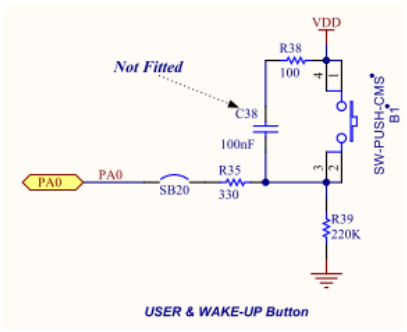
HAL - C

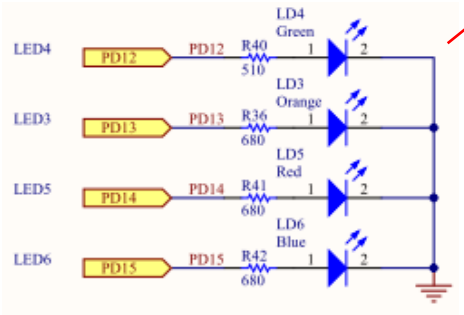
```
/* Infinite loop */
 /* USER CODE BEGIN WHILE */
 while (1)
 {

HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_12);

  /* USER CODE END WHILE */

   /* USER CODE BEGIN 3 */
 HAL_Delay(1000);
 }
 /* USER CODE END 3 */
```

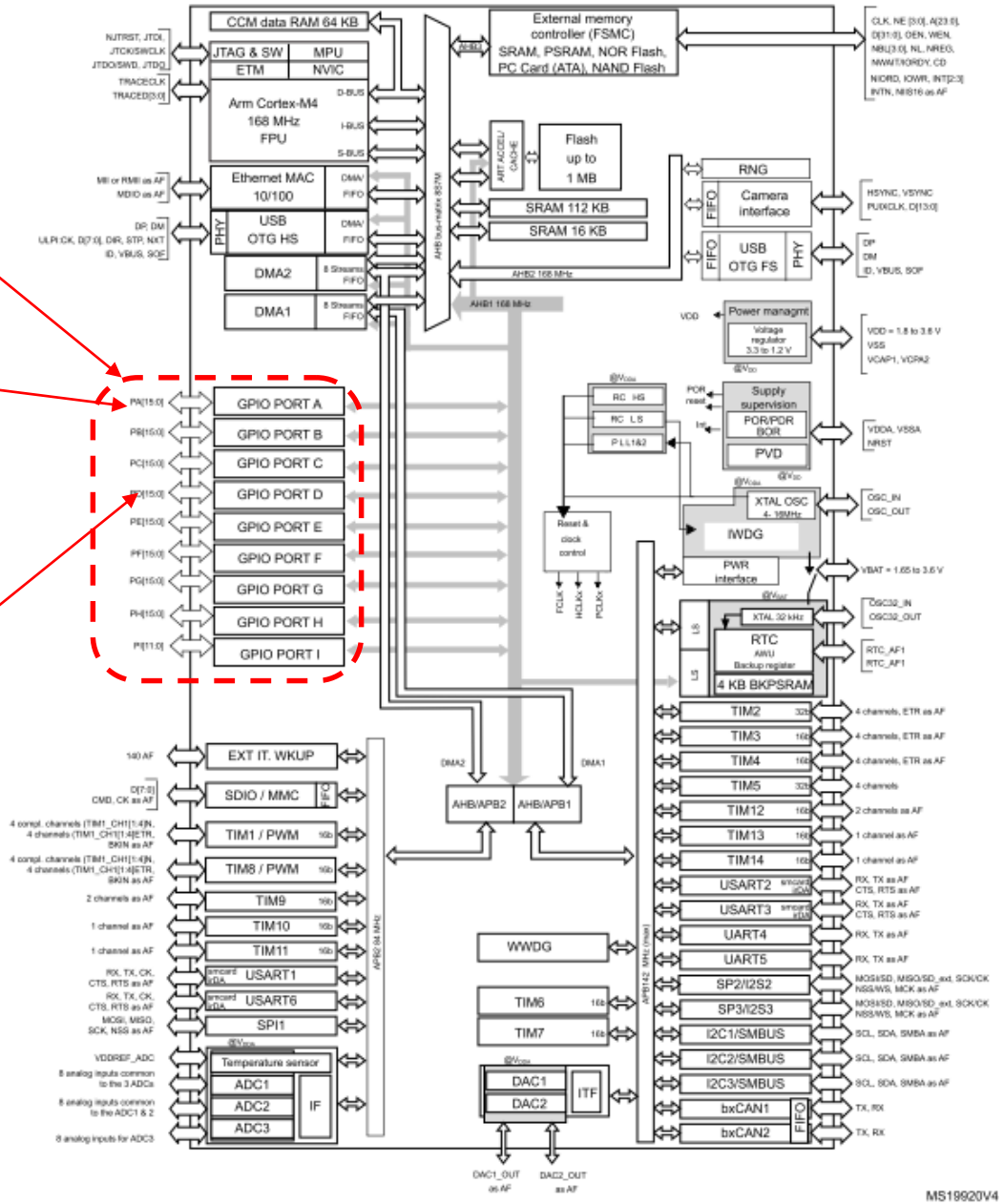
```
void HAL_GPIO_TogglePin(GPIO_TypeDef* GPIOx,
uint16_t GPIO_Pin)
{
  uint32_t odr;

  /* Check the parameters */
  assert_param(IS_GPIO_PIN(GPIO_Pin));

  /* get current Ouput Data Register value
*/
  odr = GPIOx->ODR;

  /* Set selected pins that were at low
level, and reset ones that were high */
  GPIOx->BSRR = ((odr & GPIO_Pin) <<
GPIO_NUMBER) | (~odr & GPIO_Pin);
}
```

https://github.com/LAPSyLAB/STM32F4_Discovery_VIN_Projects/tree/main/LED_Blink_Demo

GPIO Krmilnik







Delo na STM32F4 razvojnem sistemu

Baremetal pristop : zbirnik

Poseben začetni projekt za *STM32F4* (e-učilnica) :

- ***dodajanje vsebine (template.s):***

'template.s - STM32CubeIDE

avigate Search Project Run Window Help

template.s

```
54
55 _start:
56     // Enable GPIOD Peripheral Clock (bit 3 in AHB1ENR register)
57     ldr r6, = RCC_AHB1ENR        // Load peripheral clock reg address to r6
58     ldr r5, [r6]                 // Read its content to r5
59     orr r5, #0x00000008           // Set bit 3 to enable GPIOD clock
60     str r5, [r6]                 // Store result in peripheral clock register
61
62     // Make GPIOD Pin12 as output pin (bits 25:24 in MODER register)
63     ldr r6, = GPIOD_MODER        // Load GPIOD MODER register address to r6
64     ldr r5, [r6]                 // Read its content to r5
65     bic r5, #0x3000000           // Clear bits 24, 25 for P12
66     orr r5, #0x01000000          // Write 01 to bits 24, 25 for P12
67     str r5, [r6]                 // Store result in GPIOD MODER register
68
69     // Set GPIOD Pin12 to 1 (bit 12 in ODR register)
70     ldr r6, = GPIOD_ODR          // Load GPIOD output data register
71     ldr r5, [r6]                 // Read its content to r5
72     orr r5, #0x1000               // write 1 to pin 12
73     str r5, [r6]                 // Store result in GPIOD output data register
74
75     // Set GPIOD Pin12 to 0 (bit 12 in ODR register)
76     ldr r6, = GPIOD_ODR          // Load GPIOD output data register
77     ldr r5, [r6]                 // Read its content to r5
78     bic r5, #0x1000              // write 0 to pin 12
79     str r5, [r6]                 // Store result in GPIOD output data register
80
81 loop:
82     nop                          // No operation. Do nothing.
83     b loop                       // Jump to loop
84
```

STM32 CubeIDE, STM32F4 (izbrana dokumentacij

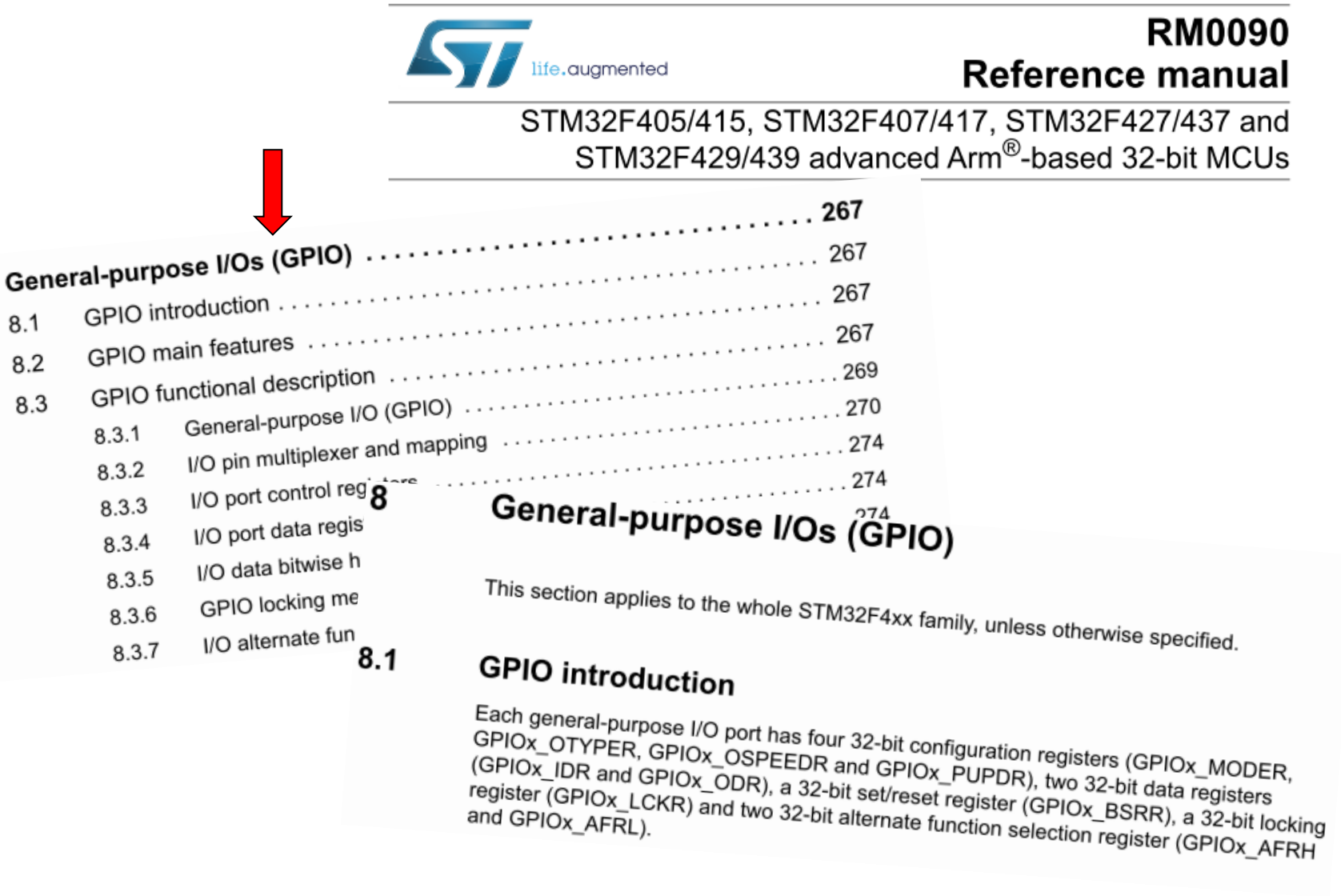
--------- Razvojni sistem ----------

- STM32 CubeIDE
- ORLab-STM32 - GitHub repozitorij
- User Manual Discovery kit stm32f407vg Uploaded 8/11/21, 12.58
- DataSheet_stm32f407vg Uploaded 8/11/21, 12.56
- Reference Manual rm0090-stm32f407417 Uploaded 8/11/21, 12.57
- Programming_Manual_pm0214-stm32-cortexm4-mcus-and-mpu
- Arm Cortex-M4 Processor Datasheet Short Uploaded 29/10/21, 15.00

--------- Cortex-M arhitektura, zbirnik ----------

- ARM Cortex-M for Beginners ARM 2017 Uploaded 29/10/21, 14.50

Vir: RM0090 Reference manual



life.augmented

RM0090
Reference manual

STM32F405/415, STM32F407/417, STM32F427/437 and STM32F429/439 advanced Arm®-based 32-bit MCUs

8 **General-purpose I/Os (GPIO)** **267**

8.1 GPIO introduction 267

8.2 GPIO main features 267

8.3 GPIO functional description 269

8.3.1 General-purpose I/O (GPIO) 270

8.3.2 I/O pin multiplexer and mapping 274

8.3.3 I/O port control reg 274

8.3.4 I/O port data regis

8.3.5 I/O data bitwise h

8.3.6 GPIO locking me

8.3.7 I/O alternate fun

8 General-purpose I/Os (GPIO)

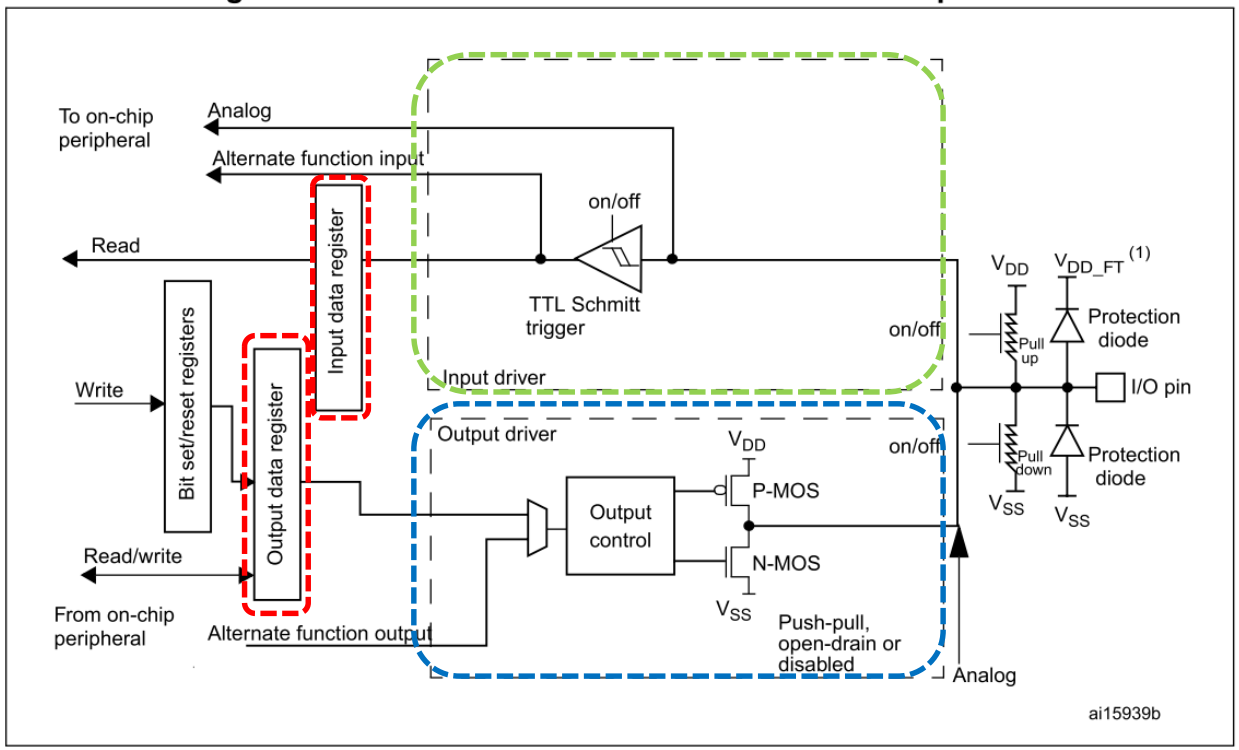
This section applies to the whole STM32F4xx family, unless otherwise specified.

8.1 GPIO introduction

Each general-purpose I/O port has four 32-bit configuration registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR and GPIOx_PUPDR), two 32-bit data registers (GPIOx_IDR and GPIOx_ODR), a 32-bit set/reset register (GPIOx_BSRR), a 32-bit locking register (GPIOx_LCKR) and two 32-bit alternate function selection register (GPIOx_AFRH and GPIOx_AFRL).

GPIO krmilnik – vhod/izhod

Figure 25. Basic structure of a five-volt tolerant I/O port bit



Osnovni registri za GPIO priključke :

RCC_AHBxENR : vklop urinega signala (enote) : Port D: RCC_AHB1ENR(b_3=1 .. Port D Enable)

MODER (Mode Register): 00: Input (reset) / 01: General purpose output mode

OTYPER (Output TYPE Register): 0: Output push-pull (reset) / 1: Output open-drain

OSPEEDR (Output SPEED Register): 00 – Low speed (reset) .. 11: Very high speed

PUPDR (Pull Up/Down Register): 00 – No pull (reset) .. 01: Pull-Up .. 10: Pull-Down

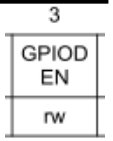
IDR (Input Data Register): stanje vhoda 1 / 0

ODR (Output Data Register): stanje izhoda 1 / 0

GPIO krmilnik – izhod *(Registri za nastavitve delovanja)*

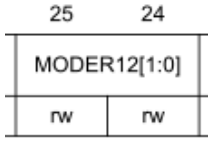
RCC_AHB1ENR(Peripheral Clock Register):

Bit 3 **GPIODEN:** IO port D clock enable
This bit is set and cleared by software.
0: IO port D clock disabled
1: IO port D clock enabled



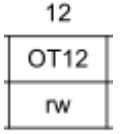
MODER (Mode Register):

Bits 2y:2y+1 **MODERy[1:0]:** Port x configuration bits (y = 0..15)
These bits are written by software to configure the I/O direction mode.
00: Input (reset state)
01: General purpose output mode
10: Alternate function mode
11: Analog mode



OTYPER (Output TYPE Register):

Bits 15:0 **OTy:** Port x configuration bits (y = 0..15)
These bits are written by software to configure the output type of the I/O port.
0: Output push-pull (reset state)
1: Output open-drain

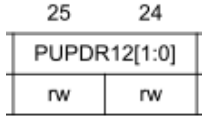


OSPEEDR (Output SPEED Register):

Bits 2y:2y+1 **OSPEEDRy[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

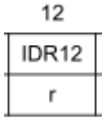
PUPDR (Pull Up/Down Register):

Bits 2y:2y+1 **PUPDRy[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



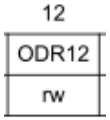
IDR (Input Data Register): stanje vhoda 1 / 0

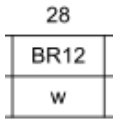
Bits 15:0 **IDRy**: Port input data (y = 0..15)
These bits are read-only and can be accessed in word mode only. They contain the input value of the corresponding I/O port.



ODR (Output Data Register): stanje izhoda 1 / 0

Bits 15:0 **ODRy**: Port output data (y = 0..15)
These bits can be read and written by software.
Note: For atomic bit set/reset, the ODR bits can be individually set and reset by writing to the GPIOx_BSRR register (x = A..I/J/K).

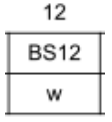




Bits 31:16 **BRy:** Port x reset bit y (y = 0..15)
1: Resets the corresponding ODRx bit

Bits 15:0 **BSy:** Port x set bit y (y= 0..15)
1: Sets the corresponding ODRx bit

0: No action on the corresponding ODRx bit



GPIO krmilnik – izhod *(Registri za določanje stanja izhodov)*

2 možnosti za določanje stanja izhodov:

1. Spreminjanje bitov v registru ODR
   Read-Modify-Write operacija na registru ODR
   ODR (Output Data Register): stanje izhoda 1 / 0

12
ODR12
rw

Bits 15:0 **ODRy**: Port output data (y = 0..15)
These bits can be read and written by software.
Note: For atomic bit set/reset, the ODR bits can be individually set and reset by writing to the GPIOx_BSRR register (x = A..I/J/K).

2. Spreminjanje bitov v registru ODR s pomočjo vpisa v register BSRR
   Write operacija v register BSRR -> sprememba bitov v registru ODR
   + atomična operacija
   BSRR (Port Set/Reset Register): stanje izhoda 1 / 0

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

28
BR12
w

Bits 31:16 **BRy:** Port x reset bit y (y = 0..15)
1: Resets the corresponding ODRx bit

Bits 15:0 **BSy:** Port x set bit y (y= 0..15)
1: Sets the corresponding ODRx bit

0: No action on the corresponding ODRx bit

12
BS12
w

GPIO krmilnik – krmiljenje izhodov

Potrebni koraki za krmiljenje izhoda:

1. RCC_AHB1ENR(Peripheral Clock Register): b_3=1 .. Port D Enable
2. **MODER (Mode Register): 01: General purpose output mode**
3. Default vrednosti že ustrezne v registrih :
   OTYPER (Output TYPE Register): 0: Output push-pull (reset state)
   OSPEEDR (Output SPEED Register): 00 – Low speed (reset state)
   PUPDR (Pull Up/Down Register): 00 – No pull (reset state)
4. določi stanje izhoda s pisanjem v ODR ali BSRR (nastavljamo na 1/0)

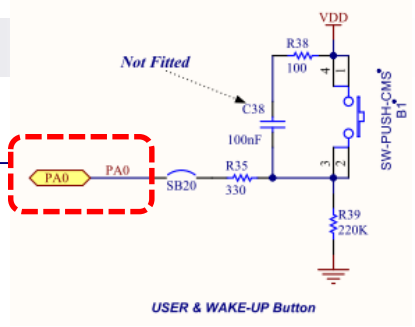
Naslovi registrov:

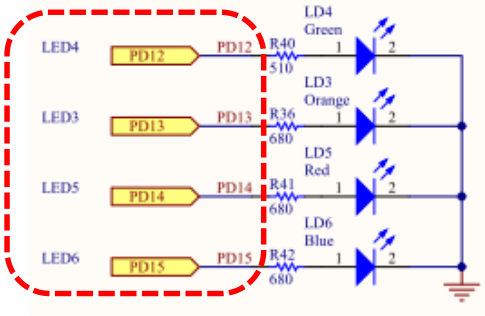
```
// RCC   base address is 0x40023800
//   AHB1ENR register offset is 0x30
.equ     RCC_AHB1ENR,   0x40023830 // RCC AHB1 peripheral clock reg (page 180)

// GPIOD base address is 0x40020C00
//   MODER register offset is 0x00
//   ODR   register offset is 0x14
//   BSRR  register offset is 0x18

.equ     GPIOD_MODER,   0x40020C00 // GPIOD port mode register (page 281)
.equ     GPIOD_ODR,     0x40020C14 // GPIOD output data register (page 283)
.equ     GPIOD_BSRR,    0x40020C18 // GPIOD port set/reset register (page 284)
```

CubeIDE – SFR okno

Register	Address	Value
GPIOF		
GPIOE		
GPIOD		
MODER	0x40020c00	0x1000000
OTYPER	0x40020c04	0x0
OSPEEDR	0x40020c08	0x0
PUPDR	0x40020c0c	0x0
IDR	0x40020c10	0x14ac
ODR	0x40020c14	0x1000
BSRR	0x40020c18	
LCKR	0x40020c1c	0x0
AFRL	0x40020c20	0x0
AFRH	0x40020c24	0x0
GPIOC		
GPIOJ		
GPIOK		
GPIOB		
GPIOA		
SYSCFG		
SPI1		
SPI2		
SPI3		

Osnovni projekt CubeIDE – GPIO – tipka, LED diode





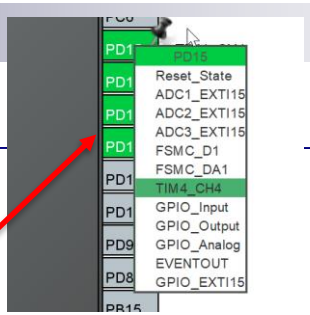
HAL - C

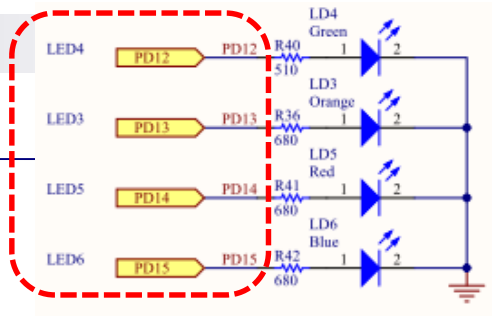
```
/* USER CODE BEGIN PV */
#define BUFSIZE 256
char SendBuffer[BUFSIZE];
int Counter;
int KeyState=0;

/* USER CODE END PV */



/* Infinite loop */
 /* USER CODE BEGIN WHILE */
 while (1)
 {

        HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_12);
        HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_13);
        HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_14);

        KeyState = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0);
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, KeyState);


         snprintf(SendBuffer,BUFSIZE,"Hello World [%d]: Key:%d\r\n",Counter++,KeyState);
         CDC_Transmit_FS(SendBuffer,strlen(SendBuffer));

   /* USER CODE END WHILE */

   /* USER CODE BEGIN 3 */
 HAL_Delay(1000);
 }
 /* USER CODE END 3 */
```

https://github.com/LAPSyLAB/STM32F4_Discovery_VIN_Projects/tree/main/STM32_USB_Key_LED

Osnovni projekt CubeIDE – GPIO – PWM, LED diode

HAL - C



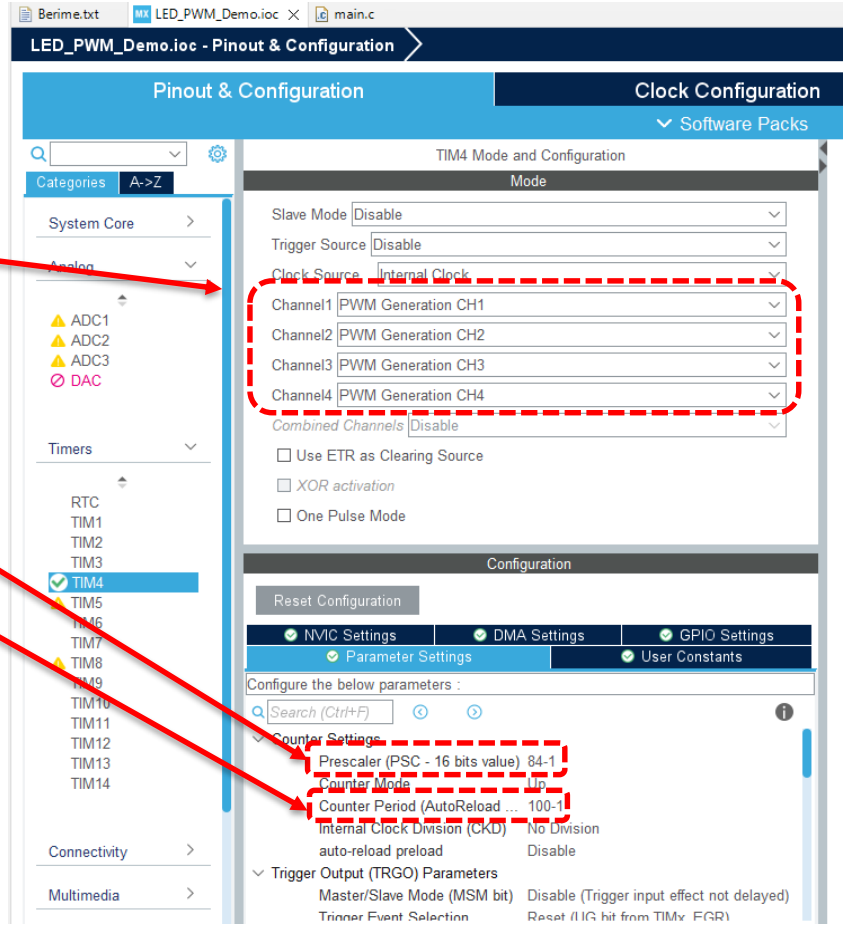


CubeMX :

1. New project -> STM32 Project -> Board -> 407DISC1
2. CubeMX: Spremeniti USB Host v USB Device :
   - Connectivity -> USB_OTG_FS -> Mode v Device Only
   - Middleware -> DEVICE_USB in Class Virtual Com Port

3. Spremeniti pine PD12-PD15 (LED diode) v TIM4_CH0-3
   - tim4 Vse kanale spremeniti na PWM Generation CH0-3

4. Clock :
   - Ura števca = 1 MHz
     - Prescaler (PSC - 16 bits value) Prescaler (PSC - 16 bits value) must be between 0 and 65 535 **= 84-1 = 83** (clock 1Mhz)
   - Perioda štetja je 100 (duty cycle pa lahko 0-100)
     - Counter Period (AutoReload Register - 16 bits value) Counter Period (AutoReload Register - 16 bits value) **= 100-1 = 99**



 © Rozman,Škraba, FRI

HAL - C

```
/* USER CODE BEGIN PV */
#define BUFSIZE 256
char SendBuffer[BUFSIZE];

/* USER CODE END PV */
/* USER CODE BEGIN 2 */

HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_1);
HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_2);
HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_3);
HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_4);

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
htim4.Instance->CCR1 = duty;
htim4.Instance->CCR2 = 100-duty;
htim4.Instance->CCR3 = duty;
htim4.Instance->CCR4 = 100-duty;

  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
  snprintf (SendBuffer,BUFSIZE,"USB:0.1 secs. Duty=%d%%\r\n",duty);
  CDC_Transmit_FS(SendBuffer,strlen(SendBuffer));

    duty = (duty + 1) ;
    if (duty > 100 )
      duty = 0;


  HAL_Delay(100);

}
/* USER CODE END 3 */
```

CubeMX :

1. New project -> STM32 Project -> Board -> 407DISC1
2. CubeMX: Spremeniti USB Host v USB Device :
   - Connectivity -> USB_OTG_FS -> Mode v Device Only
   - Middleware -> DEVICE_USB in Class Virtual Com Port
3. Spremeniti pine PD12-PD15 (LED diode) v TIM4_CH0-3
   - tim4 Vse kanale spremeniti na PWM Generation CH0-3
4. Clock :
   - Ura števca = 1 MHz
     - Prescaler (PSC - 16 bits value) Prescaler (PSC - 16 bits value) must be between 0 and 65 535 = 84-1 = 83 (clock 1Mhz)
   - Perioda štetja je 100 (duty cycle pa lahko 0-100)
     - Counter Period (AutoReload Register - 16 bits value) Counter Period (AutoReload Register - 16 bits value) = 100-1 = 99

https://github.com/LAPSyLAB/STM32F4_Discovery_VIN_Projects/tree/main/LED_PWM_Demo