

ARM

Projekt za STM32H7 vgrajen sistem II. del

CubeIDE

STM32H750B-DK Discovery razvojni sistem

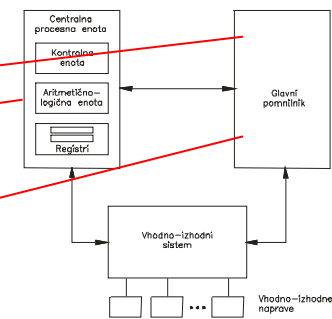
I. del

- Arm® Cortex® core-based microcontroller with 128 Kbytes (STM32H750XBH6) of Flash memory and 1 Mbyte of RAM, in TFBGA240+25 package
- 4.3" RGB interface LCD with touch panel connector
- Ethernet compliant with IEEE-802.3-2002, and POE
- USB OTG FS with Micro-AB connector
- SAI audio codec
- One ST-MEMS digital microphone
- 2 x 512-Mbit Quad-SPI NOR Flash memory
- 128-Mbit SDRAM
- 4-Gbyte on-board eMMC
- 1 user and reset push-button
- Fanout daughterboard
- 2 x FDCANs
- Board connectors:
 - USB FS Micro-AB connectors
 - ST-LINK Micro-B USB connector
 - USB power Micro-B connector
 - Ethernet RJ45
 - Stereo headset jack including analog microphone input
 - Audio header for external speakers
 - Arduino™ Uno V3 expansion connectors
 - STMod+



<https://www.st.com/en/evaluation-tools/stm32h750b-dk.html>

I. del



Delo na STM32H7 razvojnem sistemu

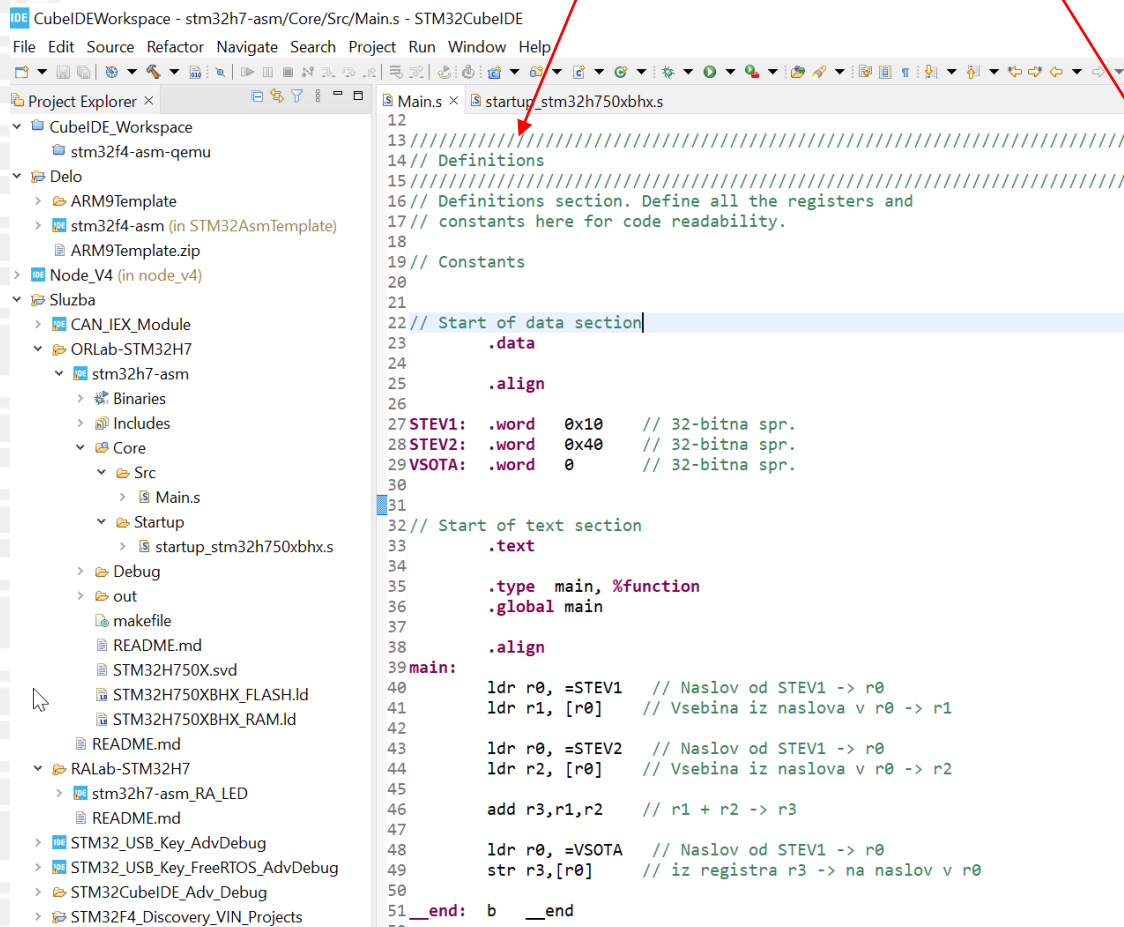
I. del

Priključitev :

- **Mikro USB** prikllop na **daljši stranici** (nad LCD, srednji !!!)

Poseben začetni projekt (github) in info za STM32H7 (e-učilnica):

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



----- Razvojni sistem STM32H750-DK -----

- STM32H750B-DK Discovery kit with STM32H750XB MCU
- ORLab-STM32H7 - GitHub repozitorij
- User Manual Discovery kit stm32h750xb Uploaded 11/11/22, 10.15
- DataSheet_stm32h750xb Uploaded 11/11/22, 10.16
- Reference Manual rm0433-stm32h750xb Uploaded 11/11/22, 10.17
- Programming_Manual_pm0253-stm32h750xb Uploaded 11/11/22, 10.17
- Errata_es0396-stm32h750xb Uploaded 11/11/22, 10.19

Delo na STM32H7 razvojnem sistemu

I. del

Priključitev :

- **Mikro USB** prikllop na **daljši stranici (nad LCD, srednji !!!)**

Poseben začetni projekt (github) in info za STM32H7 (e-učilnica):

- **začetne nastavitve** (startup_stm32h750xbhx.s) :
 - Pustimo default nastavitve:
 - 64MHz frekvenca urinega signala
 - (višja poveča porabo!)
 - izklop predpomnilnikov
 - **inicializacija sklada** oz. SP – kazalca na sklad
- **dodajanje vsebine** ([Main.s](#)):
 - podatki/operandi:
 - dodamo v .data sekcijo, končamo z .align
 - program (dodamo v .text sekcijo) :
 - **dodamo** od oznake **main:** naprej
 - na koncu programa je **mrtva zanka** (**__end:** b __end)
 - **podprograme** dodamo za mrtvo zanko



Inicializacija sistema – začetno stanje

startup_stm32h750xbhx.s :

g_pfnVectors:

```

.word _estack
.word Reset_Handler
.word NMI_Handler
.word HardFault_Handler
.word MemManage_Handler
.word BusFault_Handler
.word UsageFault_Handler
.word 0
.word 0
.word 0
.word 0
.word SVC_Handler
.word DebugMon_Handler
.word 0
.word PendSV_Handler
.word SysTick_Handler

```

ARM Cortex M – Vektorska tabela

Vector Table	Vector address (initial)
Interrupt#239 vector 1	0x000003FC
Interrupt#31 vector 1	0x000000BC
Interrupt#1 vector 1	0x00000044
Interrupt#0 vector 1	0x00000040
SysTick vector 1	0x0000003C
PendSV vector 1	0x00000038
Not used	0x00000034
Debug Monitor vector 1	0x00000030
SVC vector 1	0x0000002C
Not used	0x00000028
Not used	0x00000024
Not used	0x00000020
SecureFault (ARMv8-M Mainline) 1	0x0000001C
Usage Fault vector 1	0x00000018
Bus Fault vector 1	0x00000014
MemManage vector 1	0x00000010
HardFault vector 1	0x0000000C
NMI vector 1	0x00000008
Reset vector 1	0x00000004
MSP initial value	0x00000000

Inicializacija sistema – začetno stanje

startup_stm32h750xbhx.s :

Reset_Handler:

```
ldr    sp, =_estack    /* set stack pointer */
```

```
/* Copy the data segment initializers from flash to SRAM */
```

CopyDataInit: ...

FillZerobss: ...

```
// Initialize DWT counters - added for cycle measurements
```

```
...
```

```
/* Call the application's entry point.*/
```

```
[ bl    main ]  
bx    lr
```

Povezovalna skripta („linker script“)

II. Del
(v nadaljevanju)

Introduction to the STM32 Memory Model

The memory of STM32 microcontrollers has two main areas :

1. Flash Memory :

- ☐ Read-Only
- ☐ Begins at address 0x08000000
- ☐ Size depends on specific STM32 microcontroller
- ☐ Program code is stored here.
- ☐ Contains a vector table at address 0x08000004

MEMORY

```
{  
    FLASH(rx):ORIGIN =0x08000000,LENGTH =512K  
    SRAM(rwx):ORIGIN =0x20000000,LENGTH =128K  
}
```

2. SRAM Memory :

- ☐ Read and Write
- ☐ Begins at 0x20000000
- ☐ Size depends on specific STM32 microcontroller
- ☐ Variables and Stack are stored here

SECTIONS

```
{  
    .text :  
    {  
        *(.text)      /*merge all .text sections of input files */  
    }> FLASH  
}
```

<https://go.embeddedexpert.io/article1635692200758>

Povezovalna skripta („linker script“) - CubeIDE

Linker skripta določi razpored po pomnilnikih

Introduction to STM32 Memory Model and Linker Scripts

Introduction to the STM32 Memory Model

The memory of STM32 microcontrollers has two main areas :

1. Flash Memory :

- Read-Only
 - Begins at address 0x08000000
 - Size depends on specific STM32 microcontroller
 - Program code is stored here.
-
- Contains a vector table at address 0x08000004

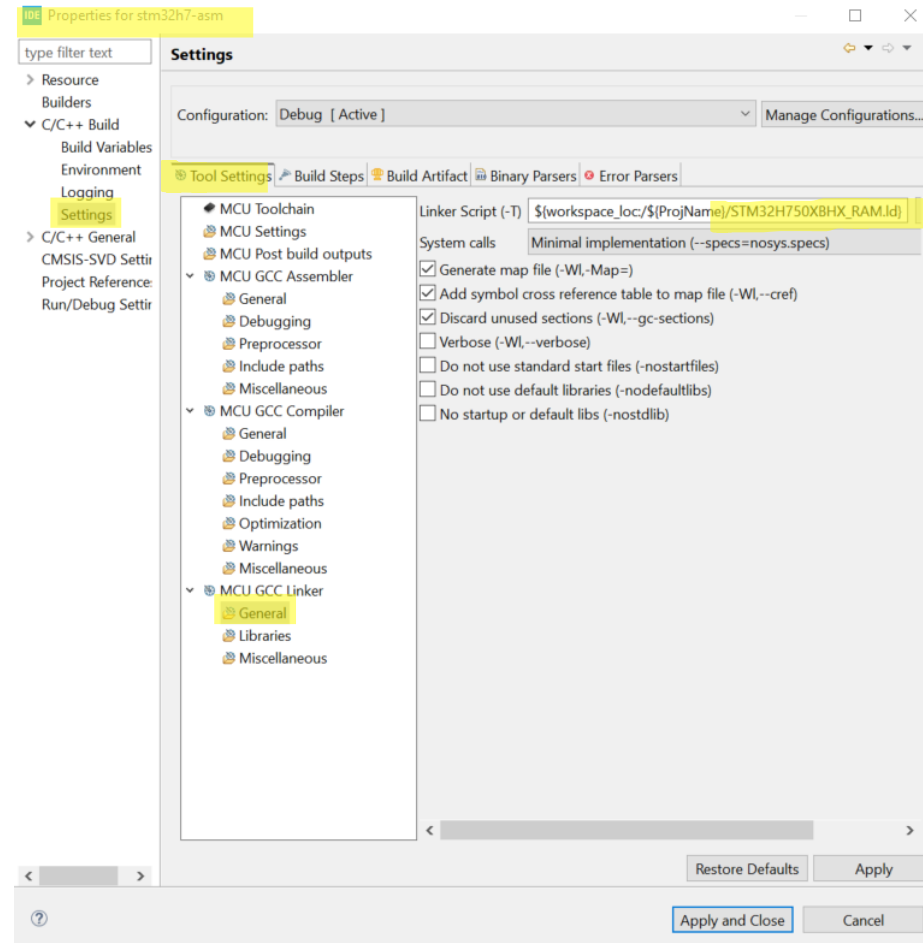
2. SRAM Memory :

- Read and Write
- Begins at 0x20000000
- Size depends on specific STM32 microcontroller
- Variables and Stack are stored here

The 4 features of a linker script :

- Memory Layout** : Available memory types
- Section Definitions** : Placing specific parts at specific locations
- Options** : Commands e.g. ENTRY POINT
- Symbols** : Variables to inject into program at link time

<https://go.embeddedexpert.io/article1635692200758>>



Povezovalna skripta („linker script“) - CubeIDE

STM32H750XBHX FLASH.Id :

/* Specify the memory areas */

MEMORY

```
{  
  FLASH (rx) : ORIGIN = 0x08000000, LENGTH = 128K  
  DTCMRAM (xrw) : ORIGIN = 0x20000000, LENGTH = 128K  
  RAM_D1 (xrw) : ORIGIN = 0x24000000, LENGTH = 512K  
  RAM_D2 (xrw) : ORIGIN = 0x30000000, LENGTH = 288K  
  RAM_D3 (xrw) : ORIGIN = 0x38000000, LENGTH = 64K  
  ITCMRAM (xrw) : ORIGIN = 0x00000000, LENGTH = 64K  
}
```

...

SECTIONS

```
{  
  /* The startup code goes first into FLASH */  
  .isr_vector :  
  {  
    ...  
  } >FLASH  
  /* The program code and other data goes into FLASH */  
  .text :  
  {  
    ...  
  } >FLASH  
  
  /* Initialized data sections goes into RAM, load LMA copy*/  
  .data :  
  {  
    ...  
  } >RAM_D1 AT> FLASH  
}
```

STM32H750XBHX RAM.Id :

/* Specify the memory areas */

MEMORY

```
{  
  RAM_EXEC (xrw) : ORIGIN = 0x24000000, LENGTH = 128K  
  DTCMRAM (xrw) : ORIGIN = 0x20000000, LENGTH = 128K  
  RAM_D2 (xrw) : ORIGIN = 0x30000000, LENGTH = 288K  
  RAM_D3 (xrw) : ORIGIN = 0x38000000, LENGTH = 64K  
  ITCMRAM (xrw) : ORIGIN = 0x00000000, LENGTH = 64K  
}  
...
```

SECTIONS

```
{  
  /* The startup code goes first into FLASH */  
  .isr_vector :  
  {  
    ...  
  } >RAM_EXEC  
  /* The program code and other data goes into FLASH */  
  .text :  
  {  
    ...  
  } >RAM_EXEC  
  
  /* Initialized data sections goes into RAM, load LMA copy*/  
  .data :  
  {  
    ...  
  } >DTCMRAM AT> RAM_EXEC  
}
```

CubeIDE – Izvedba programa - RAM

STM32H750XBHX RAM.ld:

```
/* Define output sections */
SECTIONS
{
/* The startup code goes first into RAM_EXEC (D1) */
.isr_vector :
{
...
} >RAM_EXEC

/* The program code and other data goes into RAM_EXEC(D1)*/
.text :
{
...
} >RAM_EXEC

/* Initialized data sections go into DTCMRAM from RAM_EXEC*/
.data :
{
...
} >DTCMRAM AT> RAM_EXEC

/* Uninitialized data section goes into DTCMRAM */
.bss :
{
...
} >DTCMRAM

MEMORY /* Specify the memory areas */
{
RAM_EXEC (xrw) : ORIGIN = 0x24000000, LENGTH = 128K
DTCMRAM (xrw) : ORIGIN = 0x20000000, LENGTH = 128K
RAM_D2 (xrw) : ORIGIN = 0x30000000, LENGTH = 288K
RAM_D3 (xrw) : ORIGIN = 0x38000000, LENGTH = 64K
ITCMRAM (xrw) : ORIGIN = 0x00000000, LENGTH = 64K
}
```

Region	Start address	End address	Size	Free	Used	Usage (%)
RAM_EXEC	0x24000000	0x2401ffff	128 KB	126,9 KB	1,1 KB	0.86%
DTCMRAM	0x20000000	0x2001ffff	128 KB	126,45 KB	1,55 KB	1.21%
RAM_D2	0x30000000	0x30047fff	288 KB	288 KB	0 B	0.00%
RAM_D3	0x38000000	0x3800ffff	64 KB	64 KB	0 B	0.00%
ITCMRAM	0x00000000	0x0000ffff	64 KB	64 KB	0 B	0.00%

Name	Run address ...	Load address (...)	Size
ITCMRAM	0x00000000		64 KB
DTCMRAM	0x20000000		128 KB
.data	0x20000000	0x24000454	16 B
.bss	0x20000010		28 B
.user_heap_stack	0x2000002c		1,5 KB
RAM_EXEC	0x24000000		128 KB
.data	0x20000000	0x24000454	16 B
isr_vector	0x24000000	0x24000000	664 B
g_pfnVectors	0x24000000	0x24000000	0 B
.text	0x24000298	0x24000298	436 B
.rodata	0x2400044c	0x2400044c	0 B
.init_array	0x2400044c	0x2400044c	4 B
.fini_array	0x24000450	0x24000450	4 B
RAM_D2	0x30000000		288 KB
RAM_D3	0x38000000		64 KB

Potrebno spremeniti naslov vektorske tabele !!!
//Set Vector table addr. to 0x24000000
ldr r1, =VTOR
ldr r0, =0x24000000
str r0, [r1]

CubeIDE – Izvedba programa - Flash

STM32H750XBHX FLASH.ld:

```
/* Define output sections */
SECTIONS
{
/* The startup code goes first into FLASH */
.isr_vector :
{
...
} >FLASH

/* The program code and other data goes into FLASH */
.text :
{
...
} >FLASH

/* Initialized data sections go into RAM_D1 from FLASH*/
.data :
{
...
} >RAM_D1 AT> FLASH

/* Uninitialized data section */
.bss :
{
...
} >RAM_D1
```

```
MEMORY /* Specify the memory areas */
{
FLASH (rx) : ORIGIN = 0x08000000, LENGTH = 128K
DTCMRAM (xrw) : ORIGIN = 0x20000000, LENGTH = 128K
RAM_D1 (xrw) : ORIGIN = 0x24000000, LENGTH = 512K
RAM_D2 (xrw) : ORIGIN = 0x30000000, LENGTH = 288K
RAM_D3 (xrw) : ORIGIN = 0x38000000, LENGTH = 64K
ITCMRAM (xrw) : ORIGIN = 0x00000000, LENGTH = 64K
}
```

Build Analyzer × Static Stack Analyzer Search Disassembly

stm32h7-asm.elf - /stm32h7-asm/Debug - Dec 11, 2023, 10:57:40 AM

Region	Start address	End address	Size	Free	Used	Usage (%)
FLASH	0x08000000	0x0801ffff	128 KB	126,9 KB	1,1 KB	0.86%
DTCMRAM	0x20000000	0x2001ffff	128 KB	128 KB	0 B	0.00%
RAM_D1	0x24000000	0x2407ffff	512 KB	510,45 KB	1,55 KB	0.30%
RAM_D2	0x30000000	0x30047fff	288 KB	288 KB	0 B	0.00%
RAM_D3	0x38000000	0x3800ffff	64 KB	64 KB	0 B	0.00%
ITCMRAM	0x00000000	0x0000ffff	64 KB	64 KB	0 B	0.00%

Build Analyzer × Static Stack Analyzer Search Disassembly

stm32h7-asm.elf - /stm32h7-asm/Debug - Dec 11, 2023, 10:57:40 AM

Memory Regions Memory Details

Name	Run address ...	Load address (...)	Size
ITCMRAM	0x00000000		64 KB
FLASH	0x08000000		128 KB
> .isr_vector	0x08000000	0x08000000	664 B
> .text	0x08000298	0x08000298	436 B
> .rodata	0x0800044c	0x0800044c	0 B
> .init_array	0x0800044c	0x0800044c	4 B
> .fini_array	0x08000450	0x08000450	4 B
> .data	0x24000000	0x08000454	16 B
DTCMRAM	0x20000000		128 KB
RAM_D1	0x24000000		512 KB
> .data	0x24000000	0x08000454	16 B
.bss	0x24000010		28 B
.user_heap_stack	0x2400002c		1,5 KB
RAM_D2	0x30000000		288 KB
RAM_D3	0x38000000		64 KB

GitHub repozitorij ORLab-STM32H7

LAPSYLAB / ORLab-STM32H7

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

ORLab-STM32H7 Public

main 1 Branch 0 Tags Go to file

LAPSYLAB	USART3 solution added	95129
DWT_Cycles_Measurements	Minor	
Docs	Docs added	
STM32H750B-DK_BSP_C_Basic	STM32H750B-DK_BSP_C_Basic added	
STM32H750B-DK_C_Basic	Basic project in C added	
asm_sol_GPIO_LEDs	Minor	
asm_sol_SysTick_LEDs	Minor	
asm_sol_USART3	USART3 solution added	
stm32h7-asm	Minor	
.gitignore	Initial commit	
README.md	Minor	

Projekti:

- stm32h7-asm
 - Osnovni projekt za zbirnik
- STM32H750B-DK BSP C Basic
 - Osnovni projekt z BSP (brez CubeMx)
- STM32H750B-DK C Basic
 - Osnovni projekt s CubeMx
- DWT Cycles Measurements
 - Main.s za meritve hitrosti izvajanja delov kode in preučevanje delovanja cevovoda

Docs:

- Dokumentacija

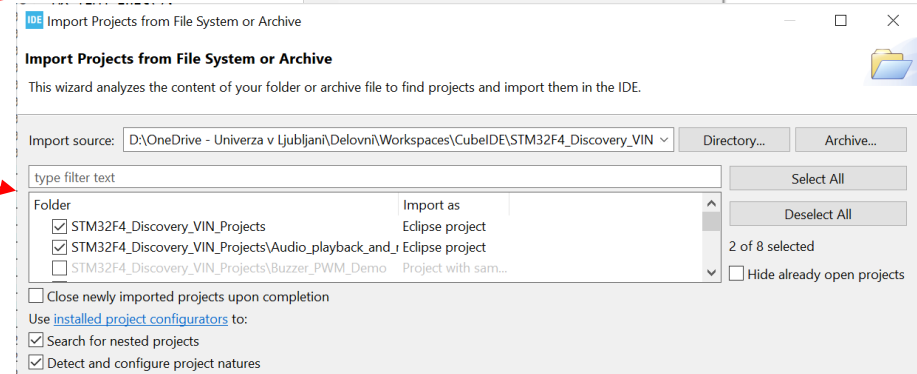
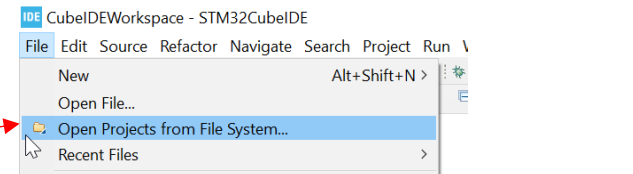
Rešitve:

- asm_sol...
 - Main.s datoteke za rešitev naloge

CubeIDE: Uvoz in delo na projektu

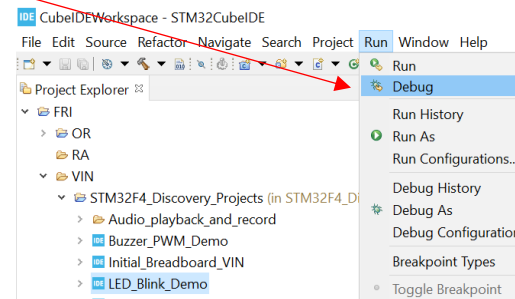
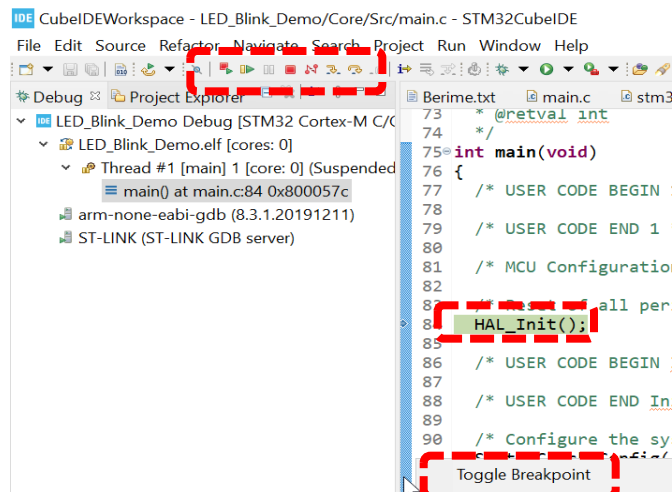
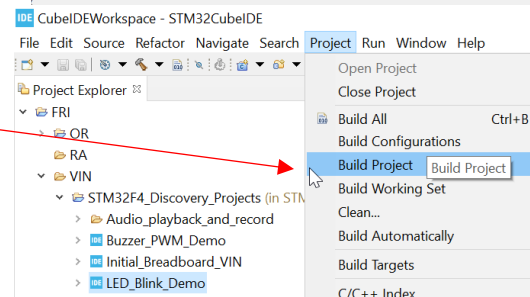
Vzpostavitev začetnega projekta :

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



Prevajanje, zagon :

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



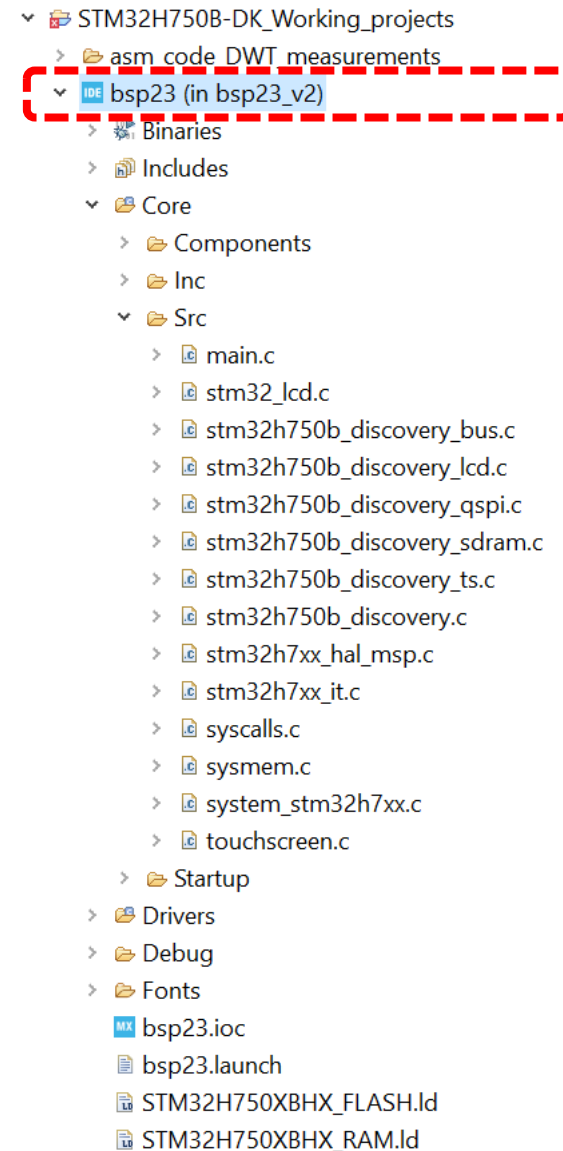
CubeIDE: kopiranje projekta

•Kopiranje CubeIDE projekta z CubeMX .ioc datoteko

- 1) Edit > Copy (obstoječi projekt).
- 2) Edit > Paste (nova lokacija).
- 3) Preimenuj .ioc datoteko.
- 4) Zbriši Debug.launch datoteko.
- 5) Project > Clean.
- 6) Generiraj kodo s CubeMX.
- 7) Project > Build Project.
- 8) Debug As Stm32 Application.
- 9) Debug aplikacije.
-

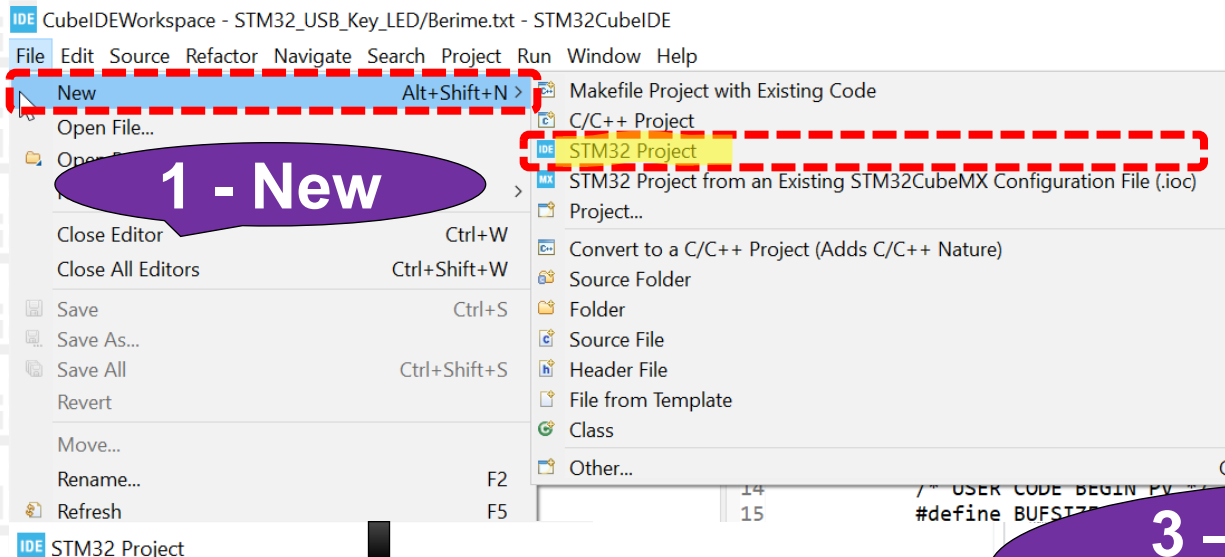
•Kopiranje osnovnih CubeIDE asm,BSP C projekta

- 1) Edit > Copy (obstoječi projekt).
- 2) Edit > Paste (nova lokacija).
- 3) Delete the Debug.launch file.
- 4) Project > Clean.
- 5) Project > Build Project.
- 6) Debug As Stm32 Application.
- 7) And debug the application
- 8) Add breakpoint on first instruction if necessary



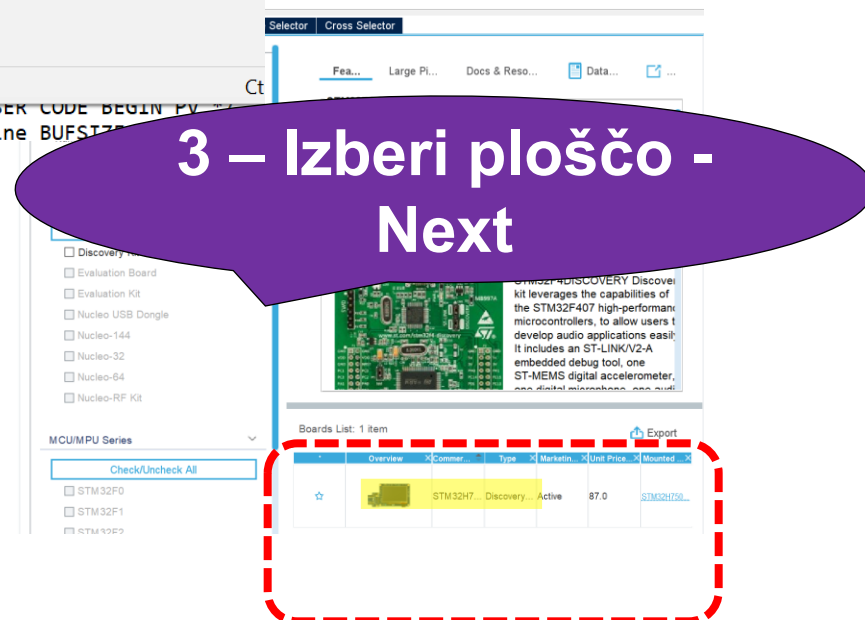
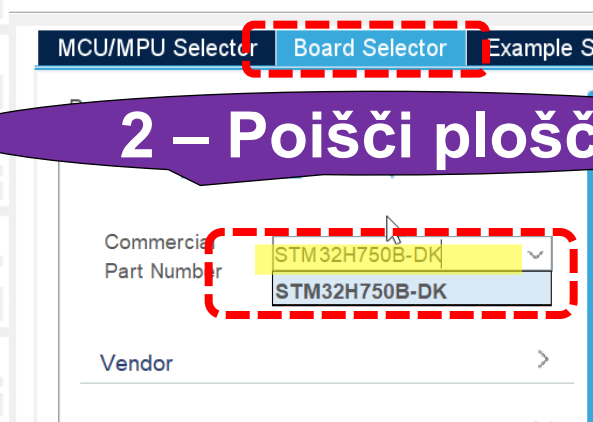
CubeIDE – Vzpostavitev novega projekta s CubeMX

Nov projekt :

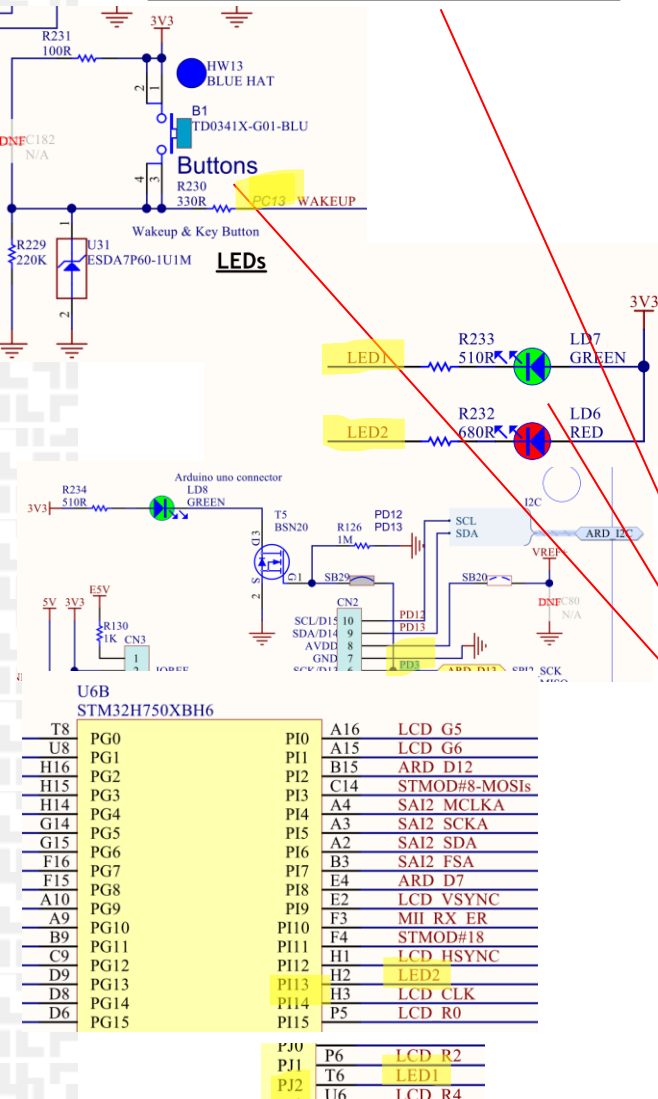


Target Selection

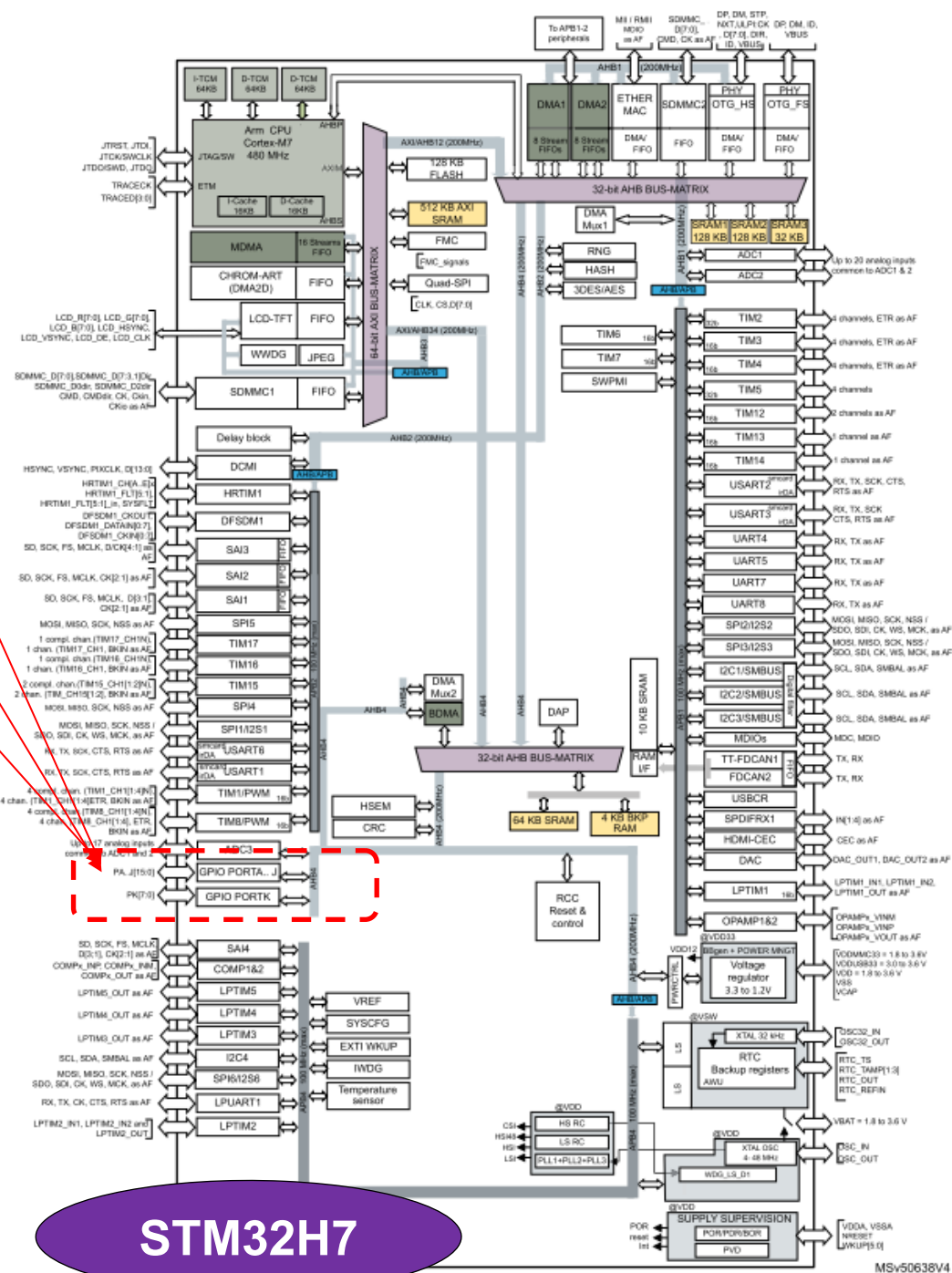
STM32 target or STM32Cube example selection is required



GPIO Krmilnik



LED: rdeča PI13, zelena PJ2
zelena PD3



OR – Organizacija računa

Osnovni projekt CubeIDE – CubeMX

Konfiguracija : priključki, knjižnice STM32H7

STM32Cube MCU packages and embedded software packs

- ☐ Copy all used libraries into the project folder
- ☒ Copy only the necessary library files
- ☐ Add necessary library files as reference in the toolchain project configuration file

Generated files

- ☐ Generate peripheral initialization as a pair of '.c/.h' files per peripheral
- ☐ Backup previously generated files when re-generating
- ☒ Keep User Code when re-generating
- ☒ Delete previously generated files when not re-generated

HAL Settings

- ☐ Set all free pins as analog (to optimize the power consumption)
- ☐ Enable Full Assert

Template Settings

Select a template to generate customized code

Settings...

Project Settings

Project Name
LED_GPIO_C_Baremetal_C

Project Location
D:\Delovni\CubeIDE\CubeIDEWorkspace

Application Structure
Advanced ☐ Do not generate the main()

Toolchain Folder Location
D:\Delovni\CubeIDE\CubeIDEWorkspace\LED_GPIO_C_Baremetal_C

Toolchain / IDE
STM32CubeIDE ☒ Generate Under Root

Linker Settings

Minimum Heap Size
0x200

Minimum Stack Size
0x400

Thread-safe Settings

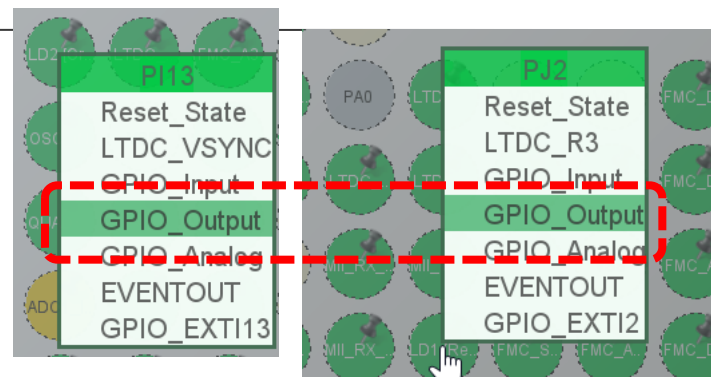
- ☐ Enable multi-threaded support
- Thread-safe Locking Strategy
Default - Mapping suitable strategy to

MCU and Firmware Package

MCU Reference
STM32F407VGTx

Firmware Package Name and Version
STM32Cube FW_F4 V1.28.2

Advanced Settings



Generated Function Calls

Generate Code	Rank	Function Name	Peripheral Instance Name	Do Not Generate Function Call
☒	1	SystemClock_Config	RCC	☐
☒	2	MX_GPIO_Init	GPIO	☐
☒	3	MX_ADC1_Init	ADC1	☐
☒	4	MX_ADC2_Init	ADC2	☐
☒	5	MX_ADC3_Init	ADC3	☐
☒	6	MX_ETH_Init	ETH	☐
☒	7	MX_FDCAN1_Init	FDCAN1	☐
☒	8	MX_FDCAN2_Init	FDCAN2	☐
☒	9	MX_FMC_Init	FMC	☐
☒	10	MX_LTDC_Init	LTDC	☐
☒	11	MX_QUADSPI_Init	QUADSPI	☐
☒	12	MX_RTC_Init	RTC	☐
☒	13	MX_SAI2_Init	SAI2	☐
☒	14	MX_SDMMC1_MMC_Init	SDMMC1	☒



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(GPIOI, GPIO_PIN_13);
```

```
    KeyState = HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13);
    HAL_GPIO_WritePin(GPIOJ, GPIO_PIN_2, KeyState);
```

```
    snprintf(SendBuffer,BUFSIZE,"Hello World [%d]: Key:%d\r\n",Counter++,KeyState);
    HAL_UART_Transmit(&huart3,SendBuffer,strlen(SendBuffer),100);
```

```
    HAL_Delay(1000);
```

```
/* USER CODE END WHILE */
```

```
/* USER CODE BEGIN 3 */
```

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

UM2217

User manual

Description of STM32H7 HAL and low-layer drivers

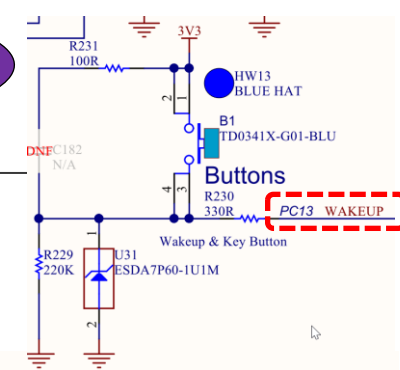
35.2.4

IO operation functions

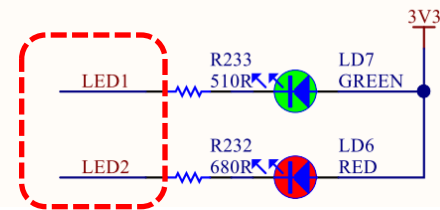
This section contains the following APIs:

- HAL_GPIO_ReadPin()
- HAL_GPIO_WritePin()
- HAL_GPIO_TogglePin()
- HAL_GPIO_LockPin()
- HAL_GPIO_EXTI_IRQHandler()
- HAL_GPIO_EXTI_Callback()

5 – GPIO

6 – USART
COM Port

LEDs



P111	H1	LCD HSYNC
P112	H2	LED2
P113	H3	LCD CLK
P114	P5	LCD R0
P115		

P30	P6	LCD R2
PJ1	T6	LED1
PJ2	T6	LCD R4

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;
int KeyState=0;

/* USER CODE END PV */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_13);

    KeyState = HAL_GPIO_ReadPin(GPIOD, GPIO_PIN_13);
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, KeyState);

    snprintf(SendBuffer, BUFSIZE, "Hello World [%d]: Key:%d\r\n", Counter++, KeyState);
    HAL_UART_Transmit(&huart3, SendBuffer, strlen(SendBuffer), 100);

    HAL_Delay(1000);
/* USER CODE END WHILE */

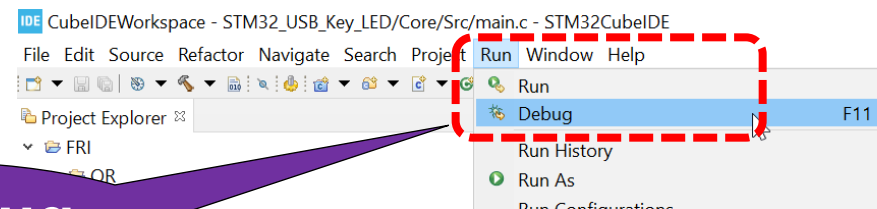
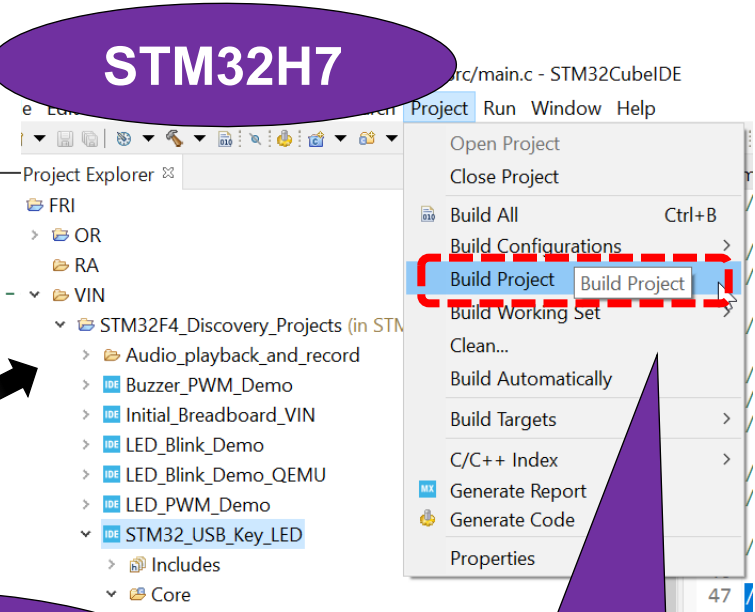
/* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */

```

7 – Delay

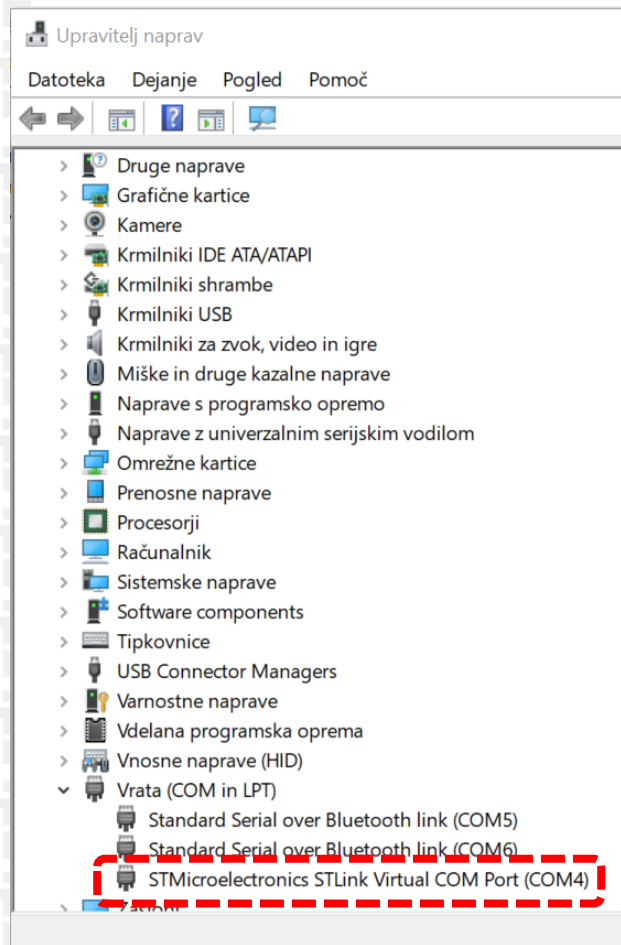
8 – Build project

9 – Debug project

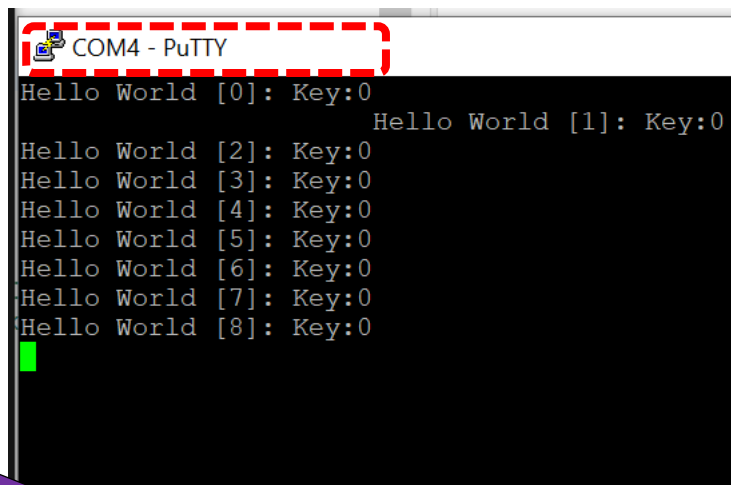
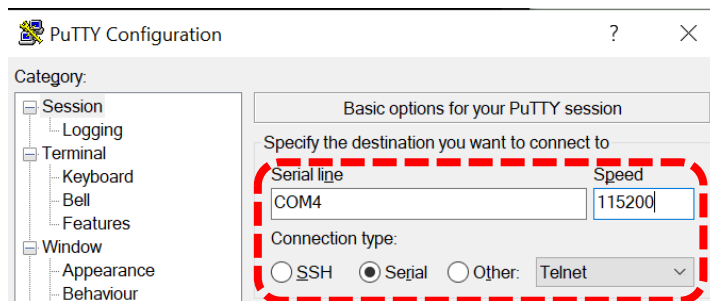


Osnovni projekt CubeIDE – USB Virtual COM Port (USART3 na STM strani)

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



<https://the.earth.li/~sgtatham/putty/latest/w64/putty.exe>



10 – Test project