

ARM

*Praktični projekt za STM32H7 vgrajen sistem
(informativno, dodatno gradivo)*

Vsota števil in utripanje LED diode

CubeIDE (VSCode)

STM32H750B-DK Discovery razvojni sistem

- 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

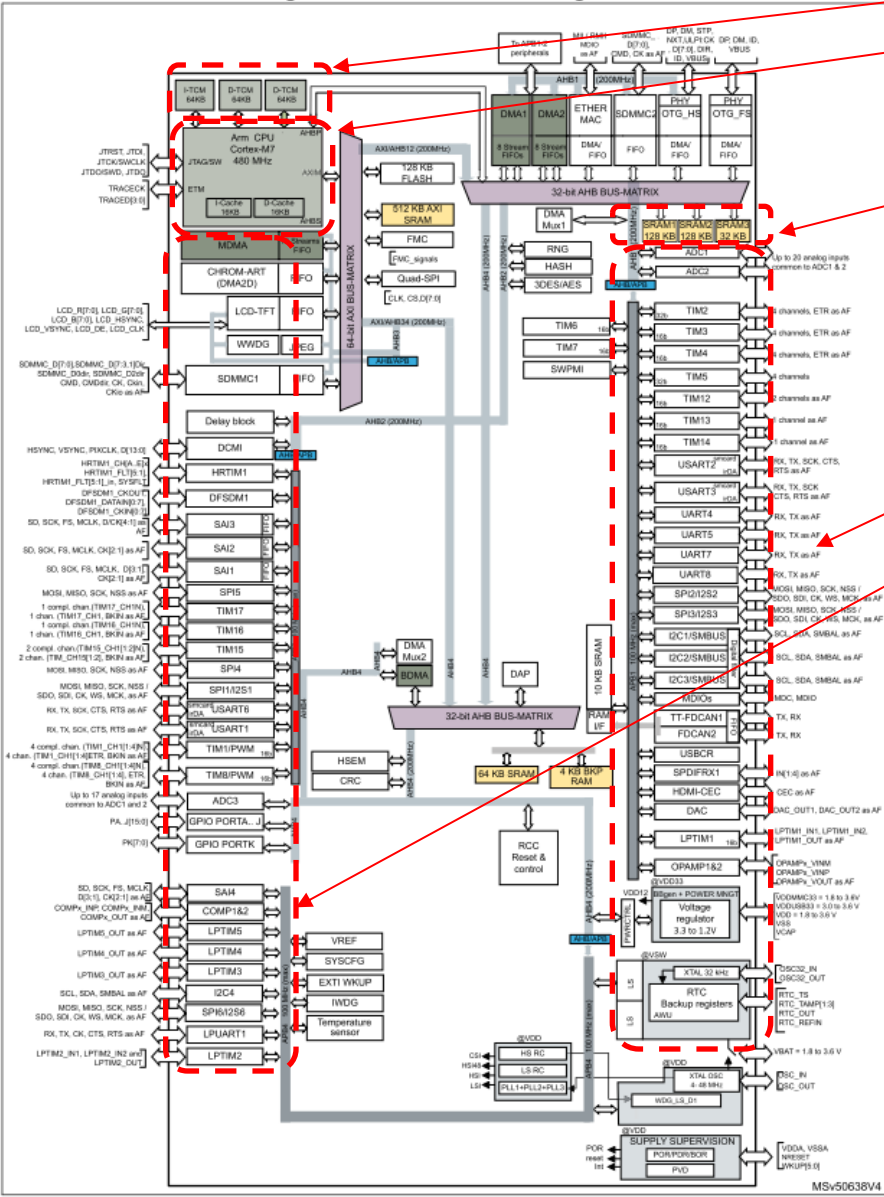
- Board connectors:

- 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+

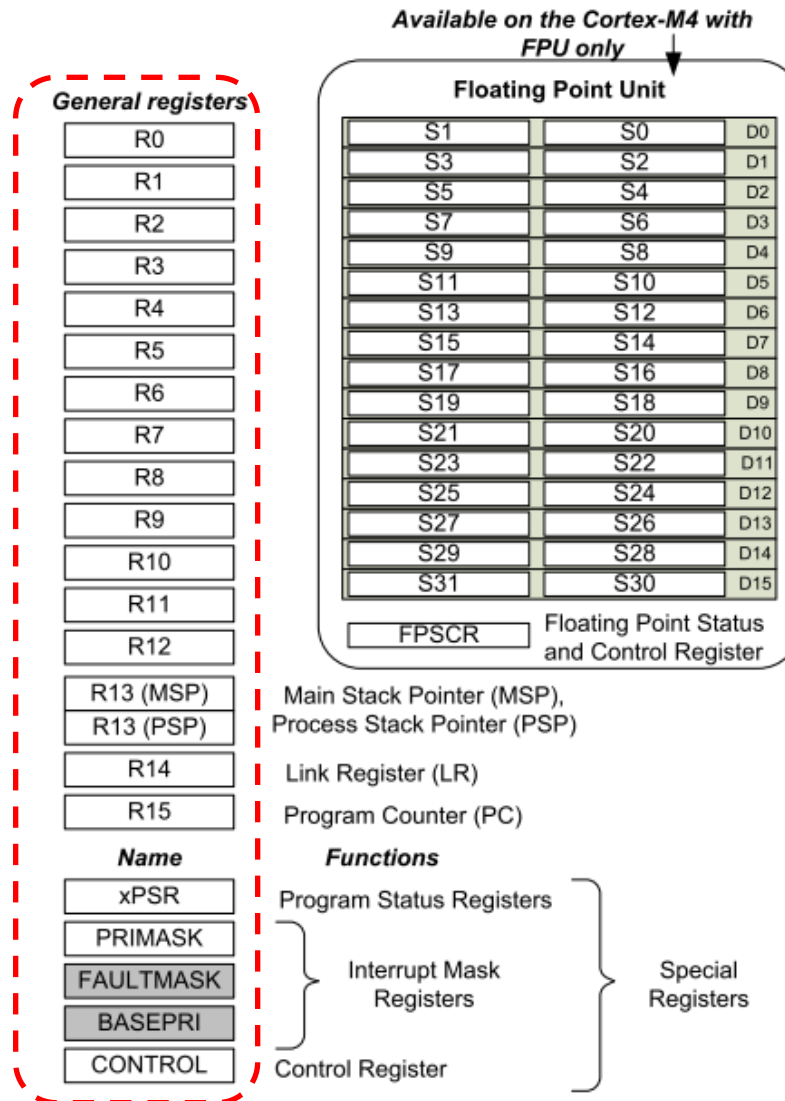


<https://www.st.com/en/evaluation-tools/stm32h750b-dk.html>

STM32H750XB

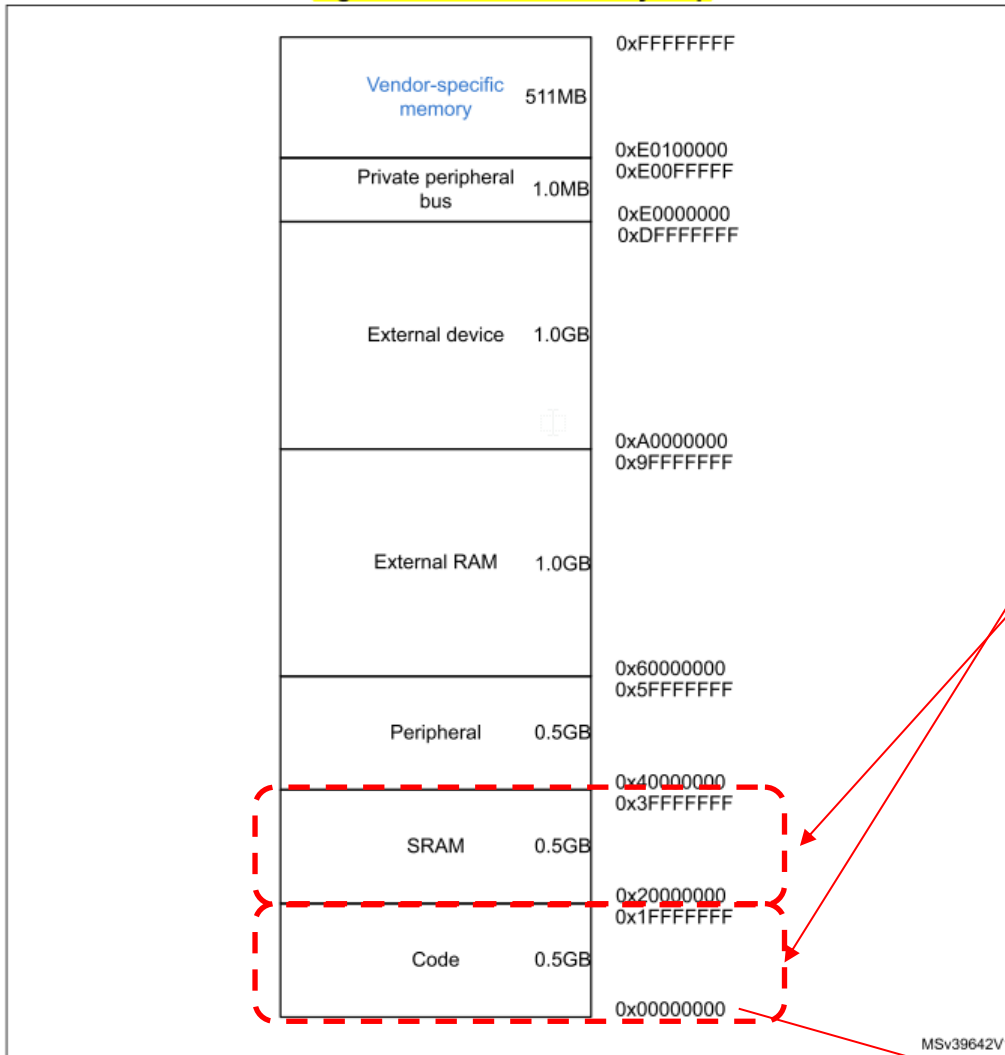


ARM Cortex M – Programski model



ARM Cortex M – Pomnilniški (naslovni) prostor

Figure 8. Processor memory map



Opis razporeda pomnilnikov – linker (povezovalnik)

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
}
    
```

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

ARM Cortex M – Vektorska tabela – začetek delovanja

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

Naslov 4 („Reset vector“) vsebuje naslov 1. ukaza „inicializacijskega“ dela, torej začetna koda se nahaja v podprogramu `Reset_Handler`.

```

.section .text.Reset_Handler
    .weak Reset_Handler
    .type Reset_Handler, %function

Reset_Handler:
    ...

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

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

    bl    main

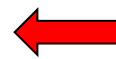
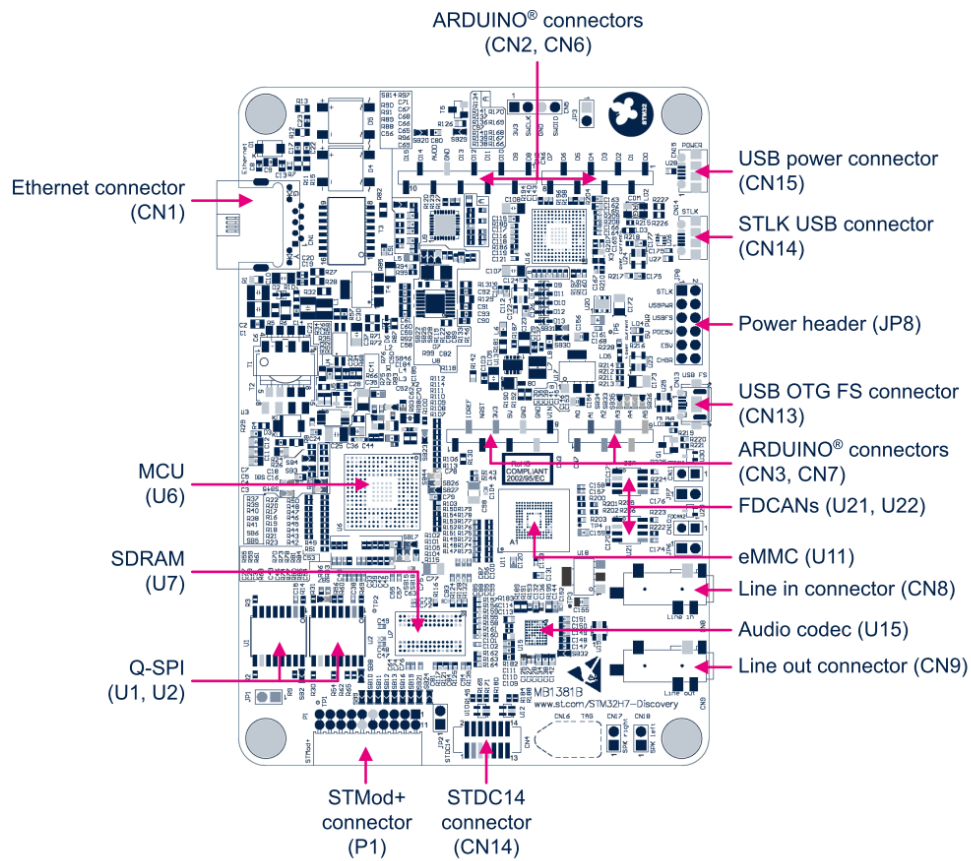
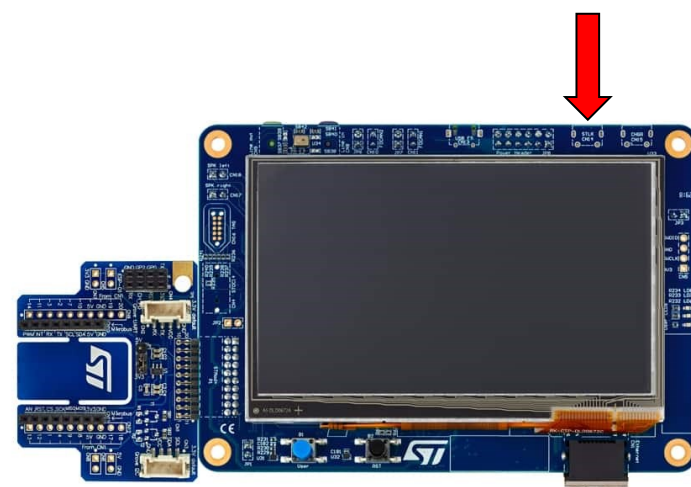
    bx   lr
    
```

Delo na plošči STM32H750B-DK

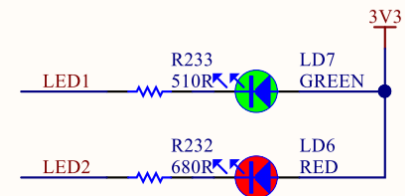
Priključitev :

- **Micro USB** prikllop nad zaslonom (**puščica!**), zasveti zaslon

Figure 5. STM32H745I-DISCO and STM32H750B-DK Discovery board bottom layout



LEDs



U6B STM32H750XBH6			
T8	PG0	PI0	A16 LCD G5
U8	PG1	PI1	A15 LCD G6
H16	PG2	PI2	B15 ARD D12
H15	PG3	PI3	C14 STMOD#8-MOSIs
H14	PG4	PI4	A4 SAI2 MCLKA
G14	PG5	PI5	A3 SAI2 SCKA
G15	PG6	PI6	A2 SAI2 SDA
F16	PG7	PI7	B3 SAI2 FSA
F15	PG8	PI8	E4 ARD D7
A10	PG9	PI9	E2 LCD VSYNC
A9	PG10	PI10	F3 MII RX ER
B9	PG11	PI11	F4 STMOD#18
C9	PG12	PI12	H1 LCD HSYNC
D9	PG13	PI13	H2 LED2
D8	PG14	PI14	H3 LCD CLK
D6	PG15	PI15	P5 LCD R0

LED: rdeča PI13, zelena PJ2

Delo na plošči STM32H750B-DK

Poseben projekt za STM32H750B-DK (e-učilnica) :

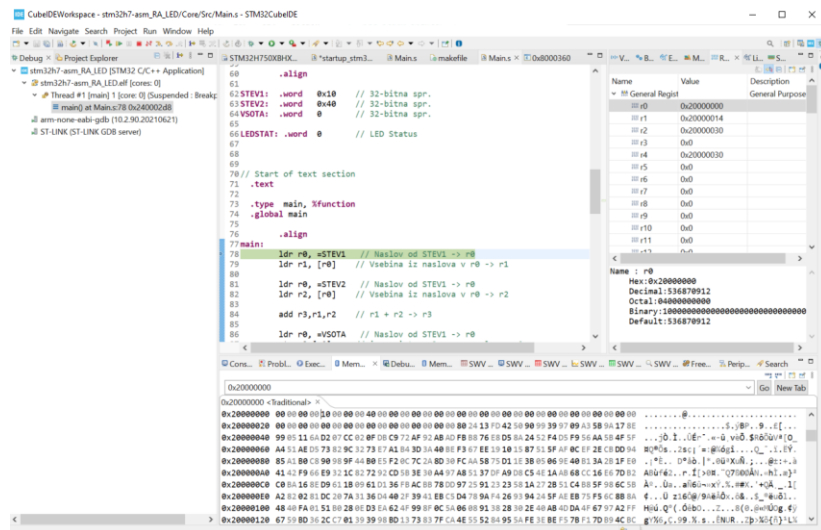
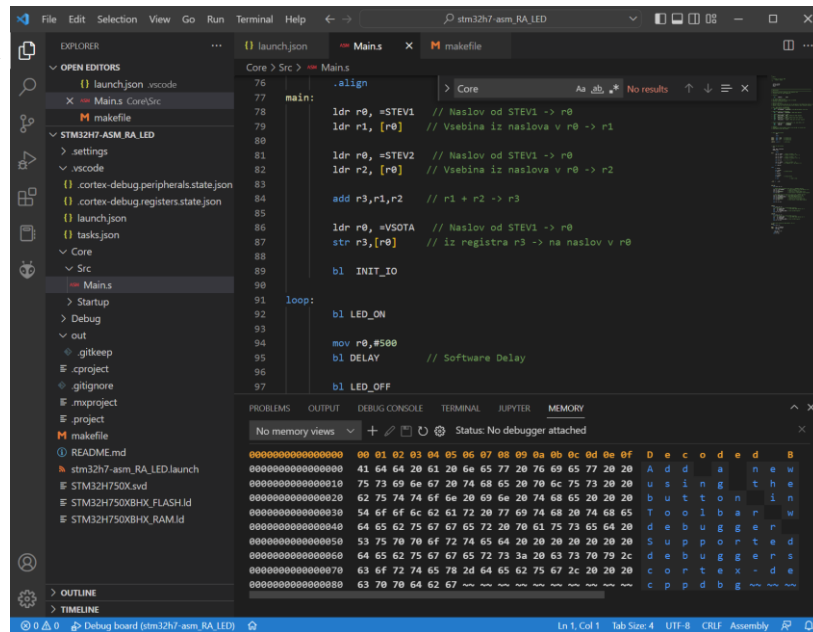
- mogoči IDE okolji (obe „cross-platform“)

- VSCoDe (MS):

- Uporaben tudi za druge jezike, projekte
- Zunanji prevajalniki
- Vse nastavljivo (konfiguracije)
- Dodati vtičnik „STM32 VS Code Extension“

- CubeIDE (ST):

- Specializiran za ST plošče
- Orodja za generiranje (pripravo) kode, knjižnic - CubeMX
- Nadgradnja Eclipse IDE
- Enostavna instalacija



Določitev spremenljivk

```
Main.s :      // Start of data section
                .data                // začetek sekcije s spremenljivkami    (v RAM pomnilnik)

                .align

                STEV1: .word 0x10    // 32-bitna spr.
                STEV2: .word 0x40    // 32-bitna spr.
                VSOTA: .word 0       // 32-bitna spr.
```

Razlike pri delu s ploščo ali CPULator simulatorjem :

- **Simulator (CPULator):**
 - Spremenljivke so kar neposredno pred programom
 - Nimamo posebne .data sekcije
 - Lahko uporabljamo ukaze za nastavitev naslova v register, npr. :
 - **adr r0, STEV1**
- **Plošča (STM32H750):**
 - Spremenljivke so po naslovih precej „oddaljene“ od programa
 - spremenljivke v RAM pomnilniku, program v FLASH pomnilniku
 - Potrebni drugačni ukazi na nastavitev naslova ali poljubne 32 bitne vrednosti v register, npr. :
 - **ldr r0, =STEV1**

Določitev programske kode

Main.s :

Razlike pri delu s ploščo ali CPULator simulatorjem :

- Simulator (CPULator):
 - *Napišemo samo program, brez „inicializacijskega“ dela*
- Plošča (STM32H750):
 - *Glavni program je v bistvu podprogram z imenom **main***
 - *kliče se iz „inicializacijskega dela projekta (startup_stm32h750xbhx.s)*
 - *„inicializacijski“ del projekta se požene vedno ob zagonu sistema ali pritisku na reset tipko*
 - *Pri intenzivnem razvoju pogosto tudi program prenašamo v RAM pomnilnik (varčujemo s številom brisanj FLASH pomnilnika)*
 - *Po koncu intenzivnega razvoja se program seveda shrani v FLASH pomnilnik.*

```
// Start of text section
.text // začetek sekcije s
programom (v FLASH pomnilnik)
.type main, %function
.global main

.align

main:
ldr r0, =STEVI // Naslov od STEVI -> r0
ldr r1, [r0] // Vsebina iz naslova v r0 -> r1

ldr r0, =STEVI // Naslov od STEVI -> r0
ldr r2, [r0] // Vsebina iz naslova v r0 -> r2

add r3,r1,r2 // r1 + r2 -> r3

ldr r0, =VSOTA // Naslov od STEVI -> r0
str r3,[r0] // iz registra r3 -> na naslov v r0
```

stm32h7-asm_RA_LED.elf - /stm32h7-asm_RA_LED/Debug - 2. nov. 2022 08:23:53

Region	Start address	End addr...	Size	Free	Used	Usage (%)
RAM_EXEC	0x24000000	0x24020...	128 KB	127,03 KB	992 B	0,76%
DTCMRAM	0x20000000	0x20020...	128 KB	126,45 KB	1,55 KB	1,21%

Priprava sistema za delovanje („inicializacija“)

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  SVC_Handler
    .word  DebugMon_Handler
    .word  0
    .word  PendSV_Handler
    .word  SysTick_Handler
    
```

```

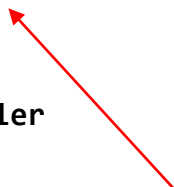
.section .text.Reset_Handler
.weak Reset_Handler
.type Reset_Handler, %function

Reset_Handler:
    ldr sp, =_estack /* set stack pointer */

    /* Call the application's entry point.*/
    bl main

    bx lr
    
```

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



Glavni program – vsota dveh števil STM32H750

Main.s :

main:

```
ldr r0, =STEVI1 // Naslov od STEVI1 -> r0 (CPU simulator: adr r0,STEVI1)
ldr r1, [r0] // Vsebina iz naslova v r0 -> r1

ldr r0, =STEVI2 // Naslov od STEVI1 -> r0 (CPU simulator: adr r0,STEVI2)
ldr r2, [r0] // Vsebina iz naslova v r0 -> r2

add r3,r1,r2 // r1 + r2 -> r3

ldr r0, =VSOTA // Naslov od STEVI1 -> r0 (CPU simulator: adr r0,VSOTA)
str r3,[r0] // iz registra r3 -> na naslov v r0
```

Glavni program – utripanje LED diod in PA3

Main.s :

```
bl INIT // Priprava V/I in sistemskih naprav za kontrolo LED diod in PA3

ldr r1,=LED1
ldr r2,=LED2
ldr r3,=PA3

mov r4,#0xff // LED(Pin) On value
mov r5,#0 // LED(Pin) Off value

loop:
str r4,[r1] // Vklop LED1 diode
str r5,[r2] // Izklop LED2 diode
str r4,[r3] // Vklop PA3
bl WRITEOUT // Prenesi na prikljucke

@ delay half cycle

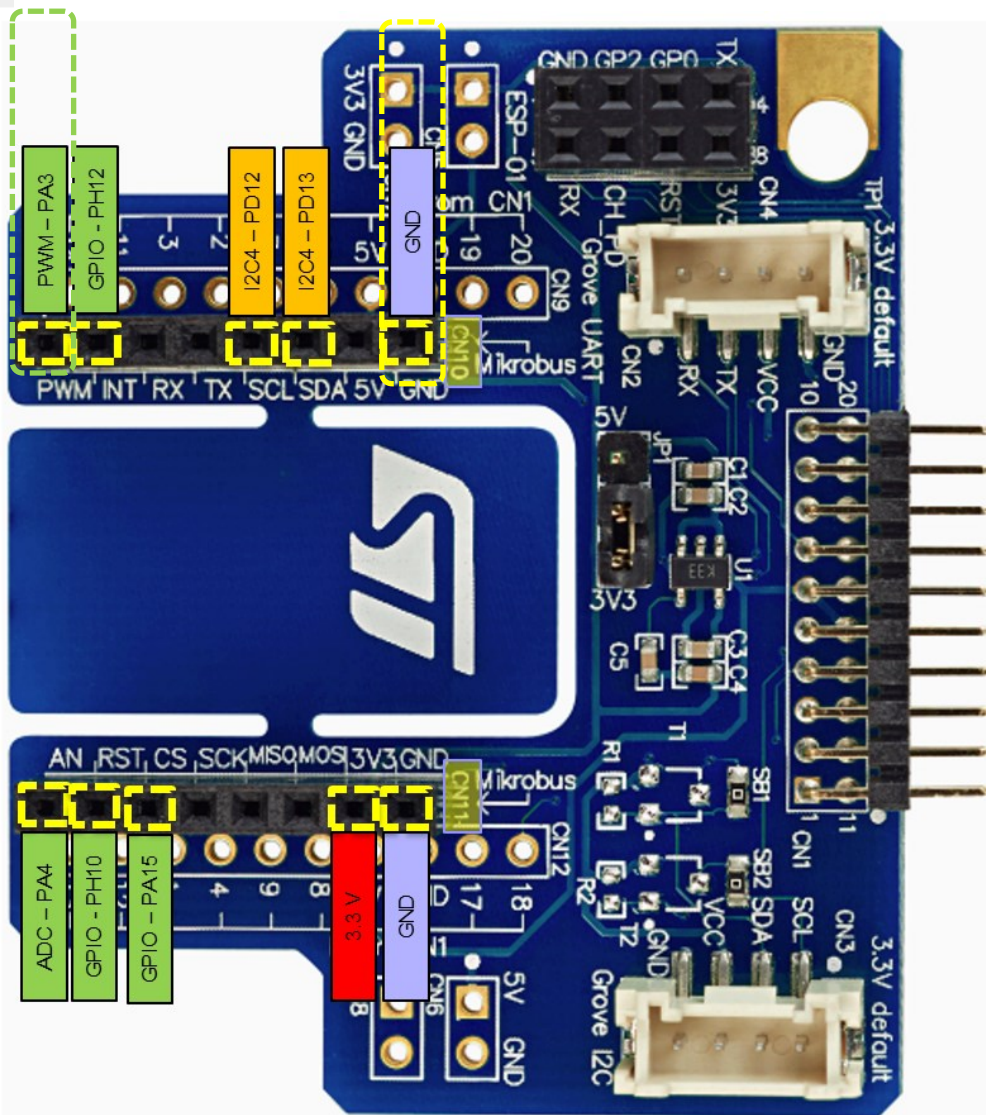
str r5,[r1] // Izklop LED1 diode
str r4,[r2] // Vklop LED2 diode
str r5,[r3] // Izklop PA3
bl WRITEOUT // Prenesi na prikljucke

@ delay half cycle

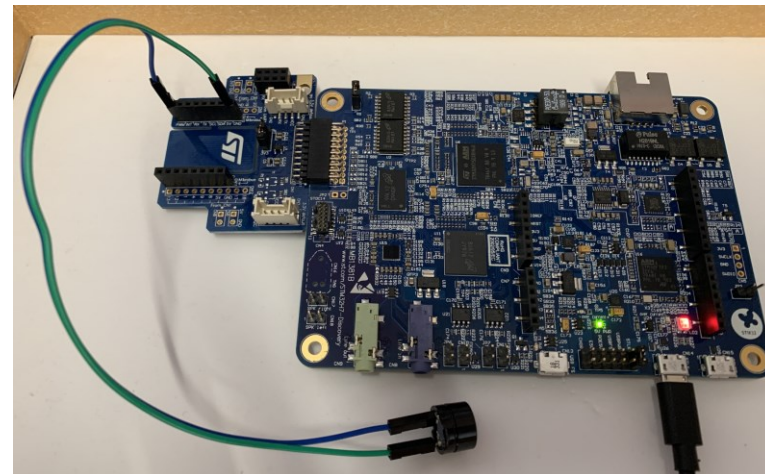
b loop // skok na vrstico loop:

__end: b __end
```

STM32H7 –Razširitvena plošča



Pravilna priključitev



Nepravilna priključitev



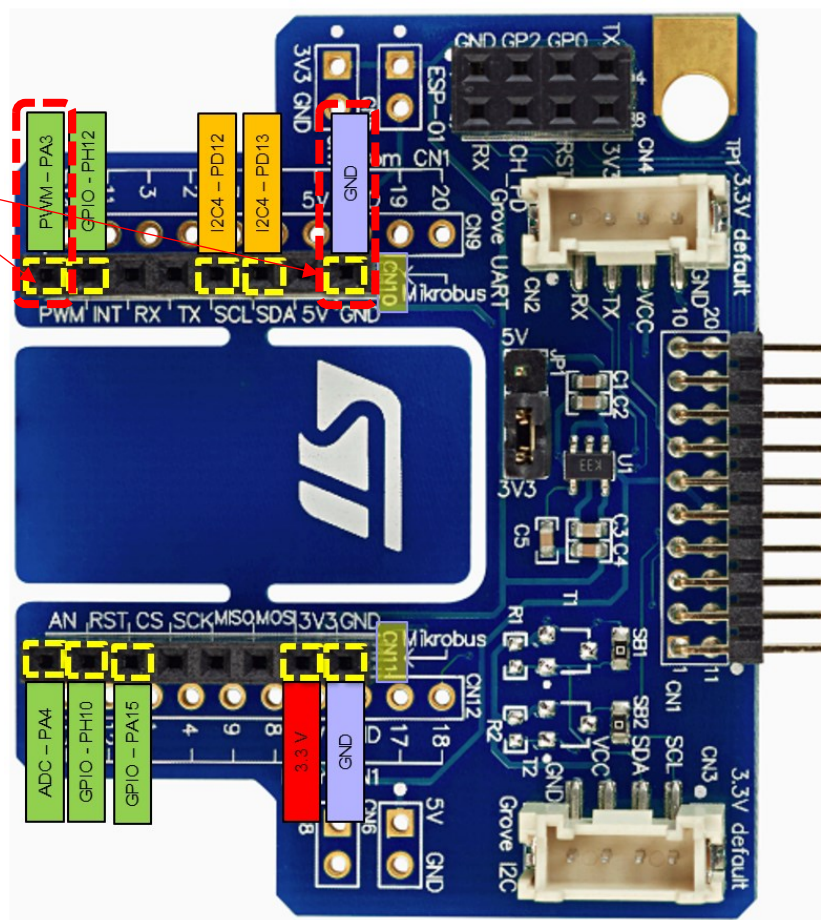
<https://www.st.com/en/evaluation-tools/stm32h750b-dk.html>

Razširitvena plošča (uporabimo priključek PA3 kot digitalni izhod)

Priključitev na STM32H7 : PA3 kot digitalni izhod (priključimo brenčač ali LED diodo z uporom)

Povezava brenčača - STM32H7:

GPIO	Vrsta	Povezava
PA3	Brenčač	+
GND	Brenčač	-



Skočni ukaz (dodatna razlaga)

Skok je ukaz tipa GOTO oznaka - pri skokih se sklicujemo na oznake. Naslov ukaza, ki stoji za oznako se zapiše v PC.

b (Branch)

```
zanka:      ...
            sub r1, r1, #1
            b zanka @ GOTO zanka
```

Zanka se bo ponavljala v nedogled. r1 se bo neprestano zmanjševal, ko bo prišel do 0, bo prišlo do prenosa, v r1 pa bo 0xffffffff.

Če želimo narediti zanko, ki se bo nehala ponavljati, ko bo r1 prišel do 0, potrebujemo ukaz tip **IF pogoj THEN GOTO oznaka**. Ukaz b se bo torej izvedel samo, če bo pogoj ustrezen.

Podprogrami (dodatna razlaga)

Pri klicanju podprogramov si je potrebno zapomniti povratni naslov.

Rešitev pri ARM CPE:

- pri klicu podprograma :
 - povratni naslov (pc) shrani v register r14 (link register = lr)
 - v **pc** vstavi naslov 1. ukaza podprograma
- pri vračanju iz podprograma je potrebno
 - povratni naslov iz r14 (lr) prepisati v r15 (pc)
 - Dosežemo s shranjevanjem lr na začetku (push) in vračanjem pc na koncu podprograma (pop) ->

Ukaz za klic podprograma:

BL PODPROGR

- Branch with Link (L = 1) - shrani povratni naslov v r14.

Podprogrami

Primeri klicev podprogramov

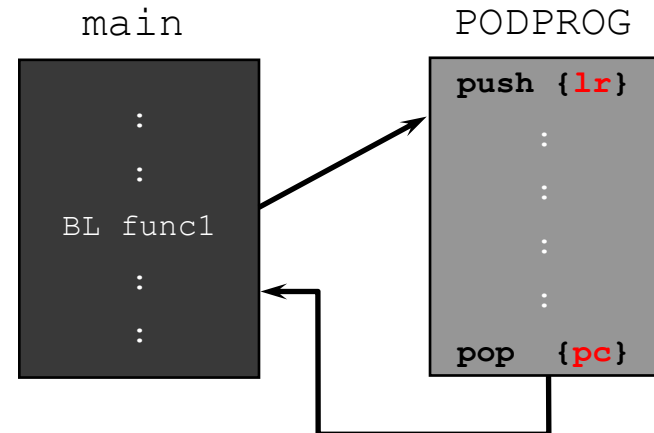
Zgled:

```

        bl  PODPROG
        ..
__end:  b  __end

PODPROG: push {r1, ..., lr}
        ..

        pop {r1, ..., pc}
    
```



Na sklad lahko začasno shranimo tudi „delovne“ registre (ki se spremenijo v podprogramu) in jih povrnemo v prvotno stanje

LED_ON:

```

    push {r5, r6, lr}
    // Set GPIOx Pins to 1 (through BSSR register)
    ldr  r6, =GPIOI_BASE      // Load GPIOI BASE address to r6
    mov  r5, #LEDS_ON
    str  r5, [r6,#GPIOx_BSSR] // Write to BSRR register
    pop {r5, r6, pc}
    
```

LED_OFF:

```

    push {r5, r6, lr}
    // Set GPIOx Pins to 0 (through BSSR register)
    ldr  r6, =GPIOI_BASE      // Load GPIOI BASE address to r6
    mov  r5, #LEDS_OFF
    str  r5, [r6,#GPIOx_BSSR] // Write to BSRR register
    pop {r5, r6, pc}
    
```

```

    // Delay with internal SW loop approx. r0 x ms
    
```

DELAY:

```

    push {r1, lr}
    
```

MSEC: ldr r1,=LEDELAY

```

    LOOP: subs r1, r1, #1
           bne LOOP
    
```

```

           subs r0, r0, #1
    
```

```

           bne MSEC
           pop {r1, pc}
    
```

Izzivi, ideje

- **Krajšanje časa vklopa/izklopa**
 - Notranji delay števec na usec (64000->64)

- **Spreminjanje razmerja časa vklopa/izklopa**
 - LED -> Dimmer
 - Vklop:izklop (10:90 - LED „temna“, 90:10 - LED „svetla“)
 - @ LED dimming (0-100% demo in 10% steps
 - TABLECNT: .hword 9
 - @ half blink period in microseconds
 - TABLE: .hword 90,80,70,60,50,40,30,20,10

- **Spreminjanje periode (enako razmerje časa vklopa/izklopa)**
 - Brenčoč -> melodije
 - @ Kuza Pazi Melody on buzzer demo
 - TABLECNT: .hword 15
 - @ Kuza pazi c-d-e-d-c (C=262Hz, D=290Hz, E=330Hz)
 - @ half note period in microseconds
 - TABLE: .hword
 - 1908,1908,1908,1908,1700,1700,1700,1700,1515,1515,1700,1700,1908,1908,1908
 - .equ REPEAT,100
 - .equ PAUSE,48000*20
 - @ first&second cycle counter is from TABLE, only period is varied at 50% duty cycle