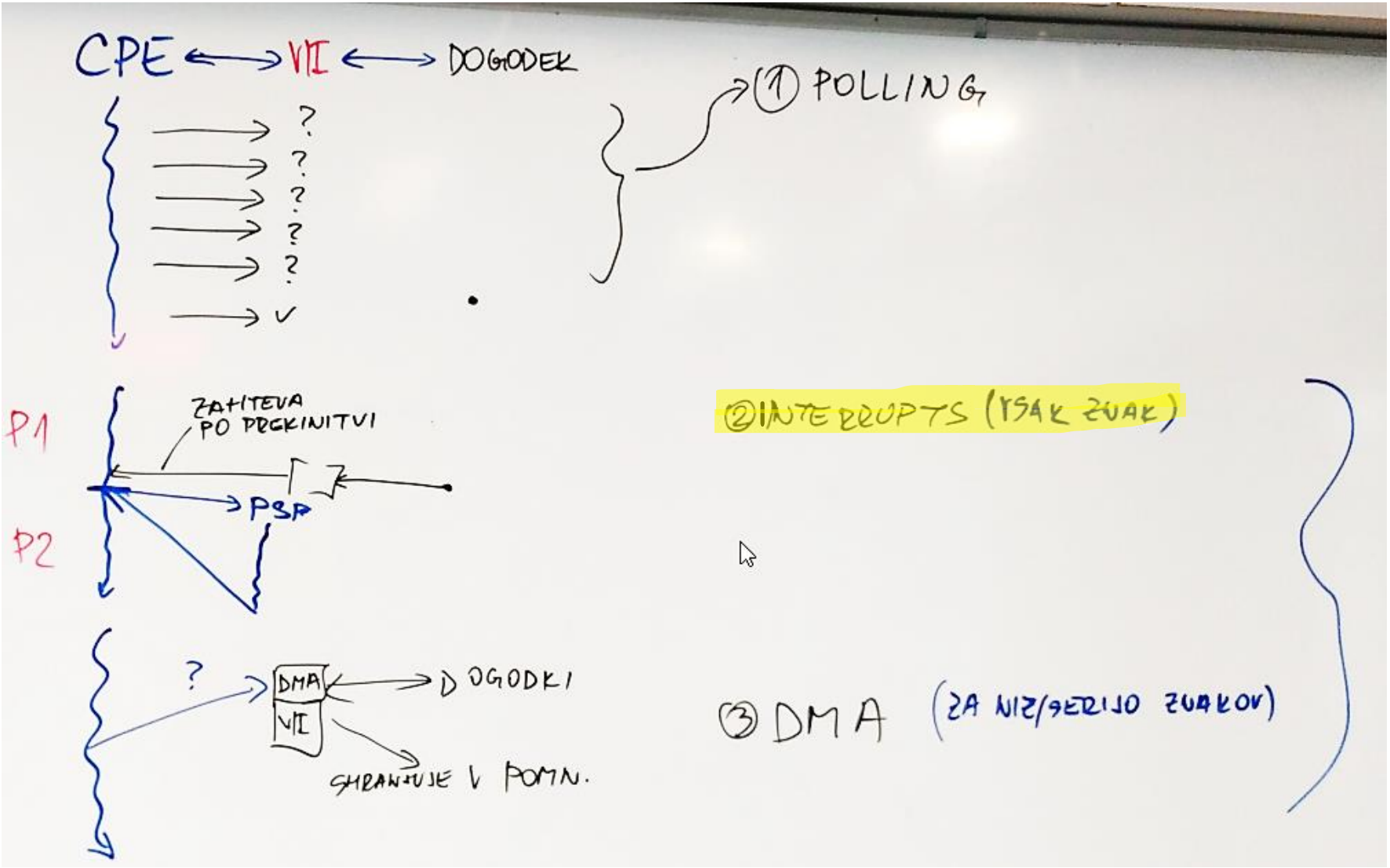
STM32H7

Vhodno / izhodne naprave

Prekinitve
+
SysTick Časovnik

Prekinitve – Zakaj ?

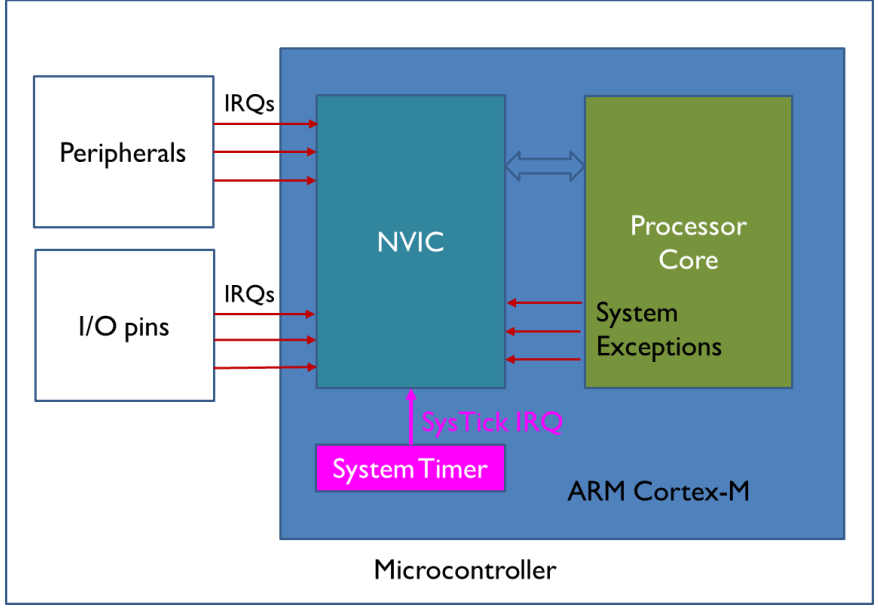


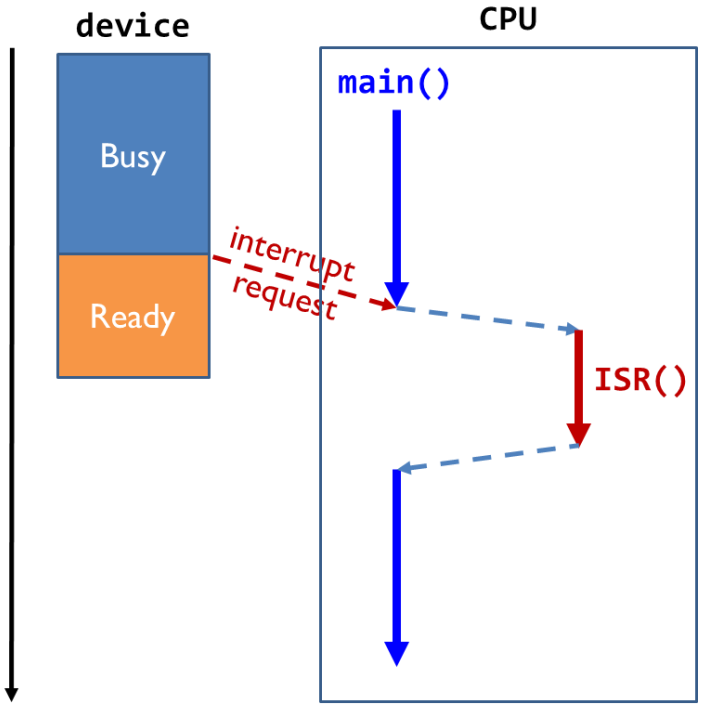
Prekinitve

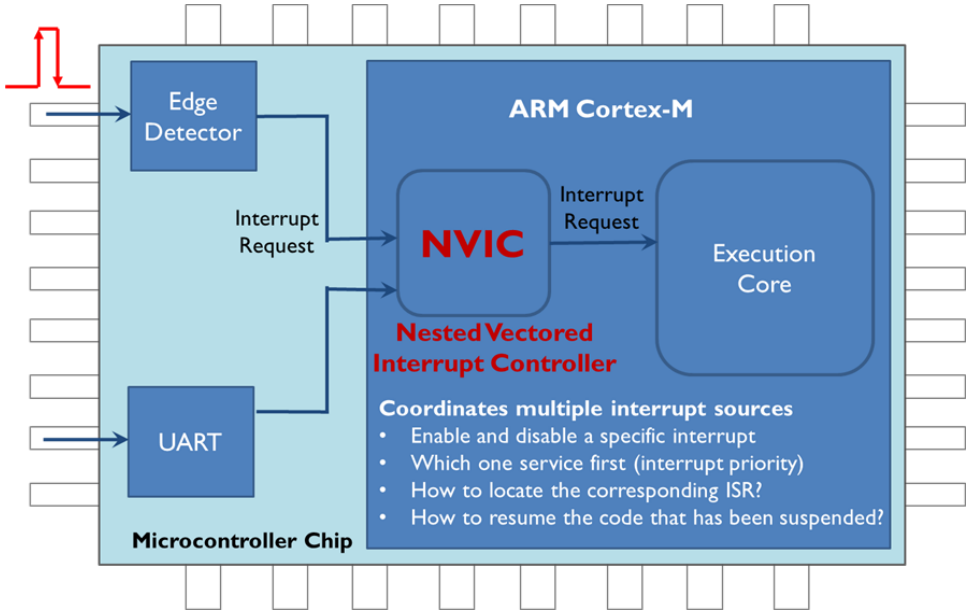
Optimizacija programske opreme za ustrezne reakcije na dogodke v realnem času:

- dogodkovno vodene reakcije
- ciklične prekinitve s časovniki

Prožijo se ustrezni Prekinitveno servisni podprogrami – PSP (ang. ISR)

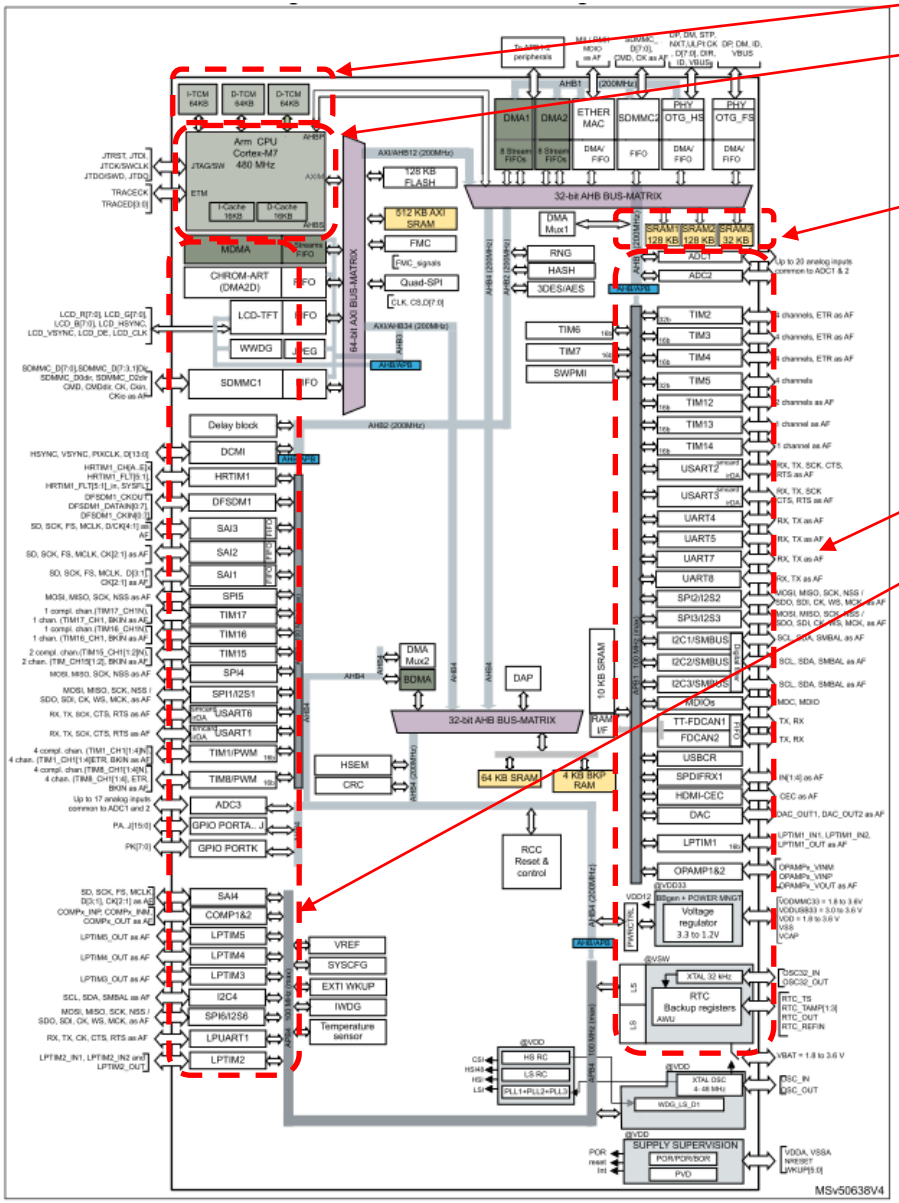


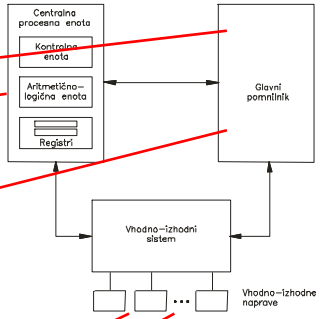




Veliko virov (naprav) povezanik na krmilnik (NVIC), ki naprej sporoča zahteve na CPE

STM32H750XB





Delo na STM32H7 razvojnem sistemu

Priključitev :

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

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

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

CubeIDEWorkspace - stm32h7-asm/Core/Src/Main.s - STM32CubeIDE

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer

- CubeIDE_Workspace
  - stm32f4-asm-qemu
- Delo
  - ARM9Template
  - stm32f4-asm (in STM32AsmTemplate)
  - ARM9Template.zip
- Node_V4 (in node_v4)
- Sluzba
  - CAN_IEX_Module
  - ORLab-STM32H7
    - stm32h7-asm
      - Binaries
      - Includes
      - Core
        - Src
          - Main.s
        - Startup
          - startup_stm32h750xbhx.s
      - Debug
      - out
      - makefile
      - README.md
      - STM32H750X.svd
      - STM32H750XBHX_FLASH.ld
      - STM32H750XBHX_RAM.ld
    - README.md
  - RALab-STM32H7
    - stm32h7-asm_RA_LED
    - README.md
  - STM32_USB_Key_AdvDebug
  - STM32_USB_Key_FreeRTOS_AdvDebug
  - STM32CubeIDE_Adv_Debug
  - STM32F4_Discovery_VIN_Projects

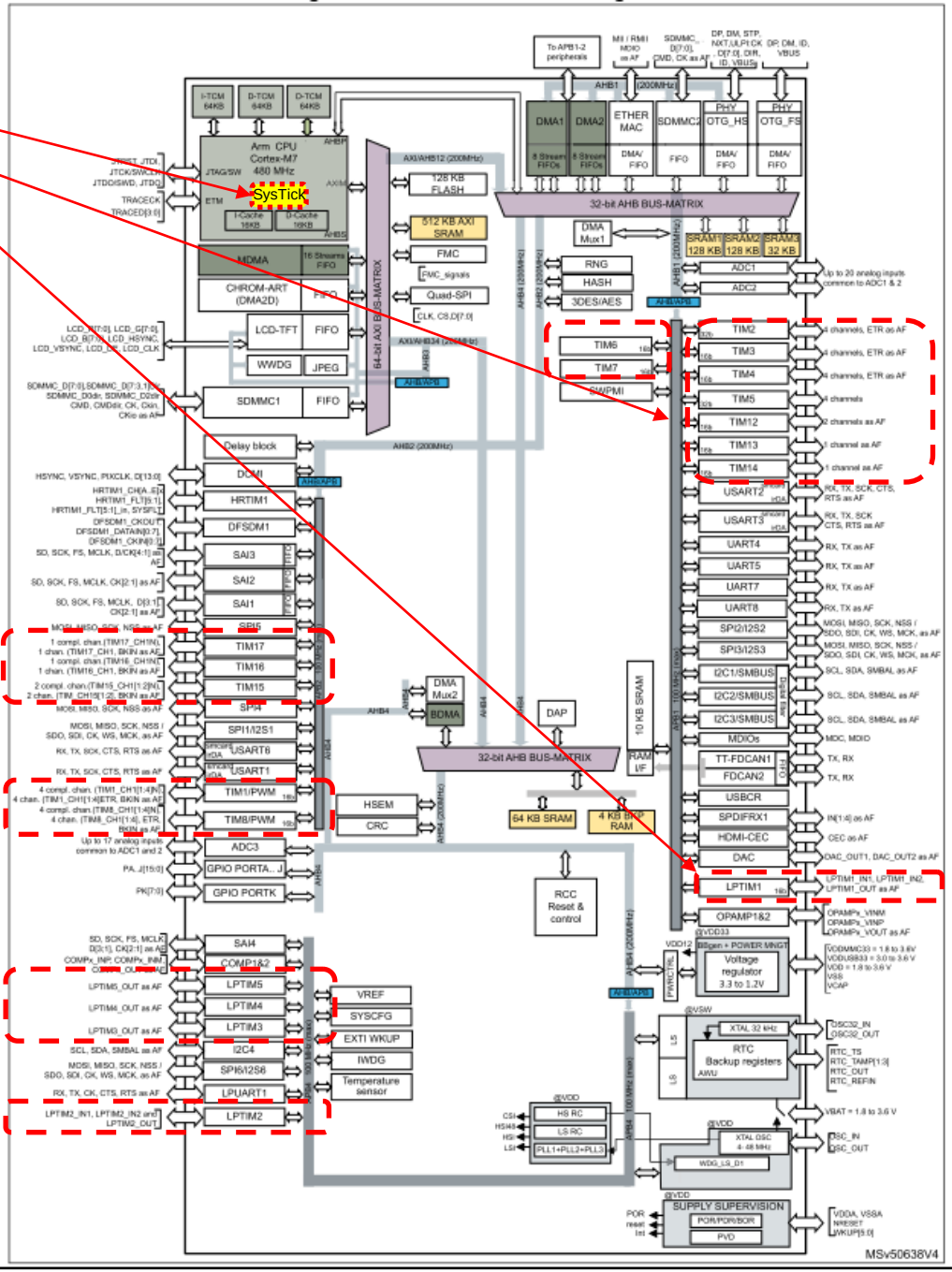
Main.s × startup_stm32h750xbhx.s

```
12
13 ////////////////////////////////////////////////////////////////////////////////
14 // Definitions
15 ////////////////////////////////////////////////////////////////////////////////
16 // Definitions section. Define all the registers and
17 // constants here for code readability.
18
19 // Constants
20
21
22 // Start of data section
23         .data
24
25         .align
26
27 STEV1:  .word   0x10    // 32-bitna spr.
28 STEV2:  .word   0x40    // 32-bitna spr.
29 VSOTA:  .word   0       // 32-bitna spr.
30
31
32 // Start of text section
33         .text
34
35         .type  main, %function
36         .global main
37
38         .align
39 main:
40         ldr r0, =STEV1   // Naslov od STEV1 -> r0
41         ldr r1, [r0]     // Vsebina iz naslova v r0 -> r1
42
43         ldr r0, =STEV2   // Naslov od STEV1 -> r0
44         ldr r2, [r0]     // Vsebina iz naslova v r0 -> r2
45
46         add r3,r1,r2     // r1 + r2 -> r3
47
48         ldr r0, =VSOTA   // Naslov od STEV1 -> r0
49         str r3,[r0]      // iz registra r3 -> na naslov v r0
50
51 __end:  b    __end
52
```

--------- 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

Časovniki


SysTick
Arm CPU Cortex-M7 480 MHz
32-bit AHB BUS-MATRIX
64-bit AXI BUS-MATRIX
TIM1/PWM
TIM8/PWM
TIM2
TIM3
TIM4
TIM5
TIM12
TIM13
TIM14
TIM6
TIM7
TIM15
TIM16
TIM17
LPTIM1
LPTIM2
LPTIM3
LPTIM4
LPTIM5
MSv50638V4

Vira: Reference & Programming manuals

life.augmented

PM0253
Programming manual

STM32F7 Series and STM32H7 Series Cortex®-M7 processor programming manual

4 Cortex-M7 peripherals

4.1 About the Cortex-M7 peripherals

The address map of the *Private peripheral bus* (PPB)

Core peripherals PM0214

4.5 SysTick timer (STK)

0xE000E100-0xE000E4EF	Nested Vectored Interrupt Controller	Table 40 on page 184
0xE000ED00-0xE000ED3F	System control block	Table 50 on page 192
0xE000ED78- 0xE000ED84	Processor features	Table 77 on page 217
0xE000ED90-0xE000EDB8	Memory Protection Unit	Table 84 on page 222
0xE000EF00-0xE000EF03	Nested Vectored Interrupt Controller	Table 40 on page 184
0xE000EF30-0xE000EF44	Floating-Point Unit	Table 94 on page 233
0xE000EF50-0xE000EF78	Cache maintenance operations	Table 100 on page 240
0xE000EF90-0xE000EFA8	Access control	Table 104 on page 245

Table 71. System timer registers summary

Address	Name	Type	Required privilege	Reset value	Description
0xE000E010	SYST_CSR	RW	Privileged	0x00000004	*SysTick control and status register*
0xE000E014	SYST_RVR	RW	Privileged	UNKNOWN	*SysTick reload value register*
0xE000E018	SYST_CVR	RW	Privileged	UNKNOWN	*SysTick current value register*
0xE000E01C	SYST_CALIB	RO	Privileged	0xC0000000	*SysTick calibration value register*

life.augmented

RM0433
Reference manual

STM32H742, STM32H743/753 and STM32H750 Value line advanced Arm®-based 32-bit MCUs

SysTick časovnik + prek. – stanje , nastavitve

Bazni naslov za registre SysTick je 0xE000E010

Figure 40. SysTick SYST_CSR bit assignments

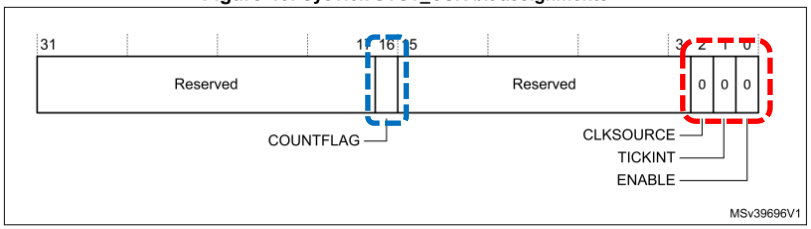


Figure 41. SYST_RVR bit assignments

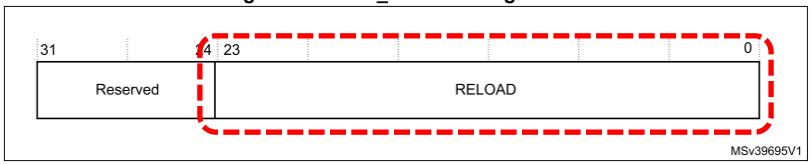
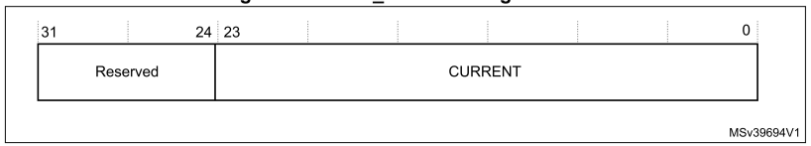


Figure 42. SYST_CVR bit assignments:



Osnovni registri za delovanje SysTick časovnika:

SYST_CSR : vklop časovnika

CLKSOURCE=1, TICKINT=1, ENABLE=1

COUNTERFLAG=1, ko prešteje do 0 (postavi na SYST_RVR in nadaljuje)

SYST_RVR : zač. vrednost štetja (šteje proti 0)

SYST_RVR = število period

SYST_CVR : trenutna vrednost števca

SYST_CVR = nekje med SYST_RVR in 0

SysTick časovnik *(Registri za nastavitve delovanja)*

Figure 40. SysTick SYST_CSR bit assignments

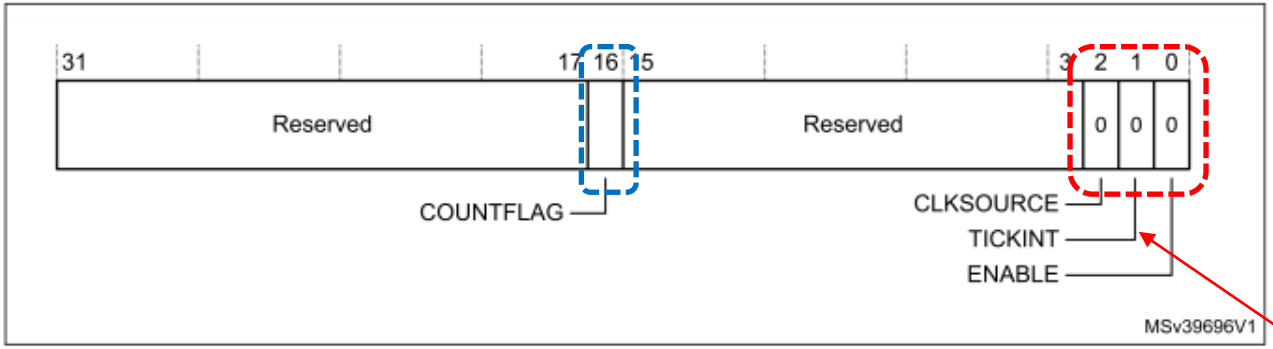


Table 72. SysTick SYST_CSR bit assignments

Bits	Name	Function
[31:17]	-	Reserved.
[16]	COUNTFLAG	Returns 1 if timer counted to 0 since last time this was read.
[15:3]	-	Reserved.
[2]	CLKSOURCE	Indicates the clock source: – 0: External clock. – 1: Processor clock.
[1]	TICKINT	Enables SysTick exception request: 0: Counting down to zero does not assert the SysTick exception request. 1: Counting down to zero asserts the SysTick exception request. Software can use COUNTFLAG to determine if SysTick has ever counted to zero.
[0]	ENABLE	Enables the counter: 0: Counter disabled. 1: Counter enabled.

Vklop prekinitve

SysTick časovnik *(Registri za nastavitve delovanja)*

4.4.2 SysTick reload value register

The SYST_RVR register specifies the start value to load into the SYST_CVR register. See the register summary in *Table 71 on page 213* for its attributes. The bit assignments are:

Figure 41. SYST_RVR bit assignments

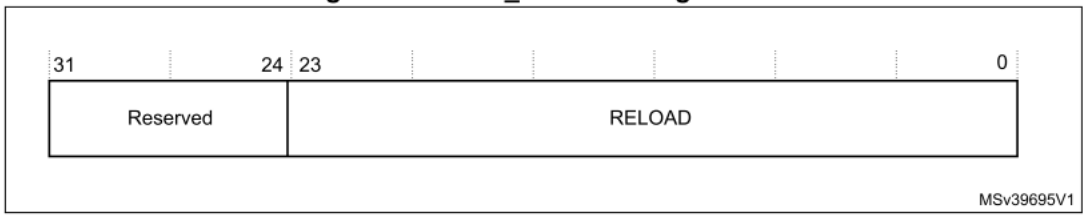


Table 73. SYST_RVR bit assignments

Bits	Name	Function
[31:24]	-	Reserved.
[23:0]	RELOAD	Value to load into the SYST_CVR register when the counter is enabled and when it reaches 0, see *Calculating the RELOAD value*.

Calculating the RELOAD value

The RELOAD value can be any value in the range 0x00000001-0x00FFFFFF. A start value of 0 is possible, but has no effect because the SysTick exception request and COUNTFLAG are activated when counting from 1 to 0.

The RELOAD value is calculated according to its use. For example, to generate a multi-shot timer with a period of N processor clock cycles, use a RELOAD value of N-1. If the SysTick interrupt is required every 100 clock pulses, set RELOAD to 99.

SysTick časovnik *(Registri za nastavitve delovanja)*

4.4.3 SysTick current value register

The SYST_CVR register contains the current value of the SysTick counter. See the register summary in *Table 71 on page 213* for its attributes. The bit assignments are

Figure 42. SYST_CVR bit assignments:

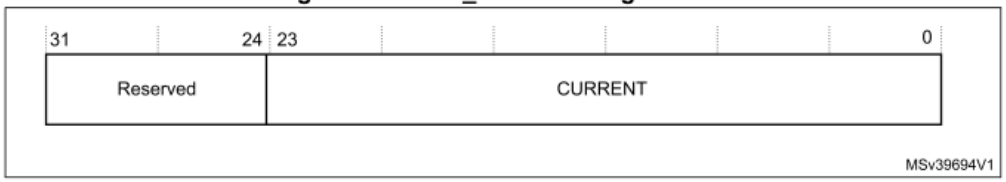
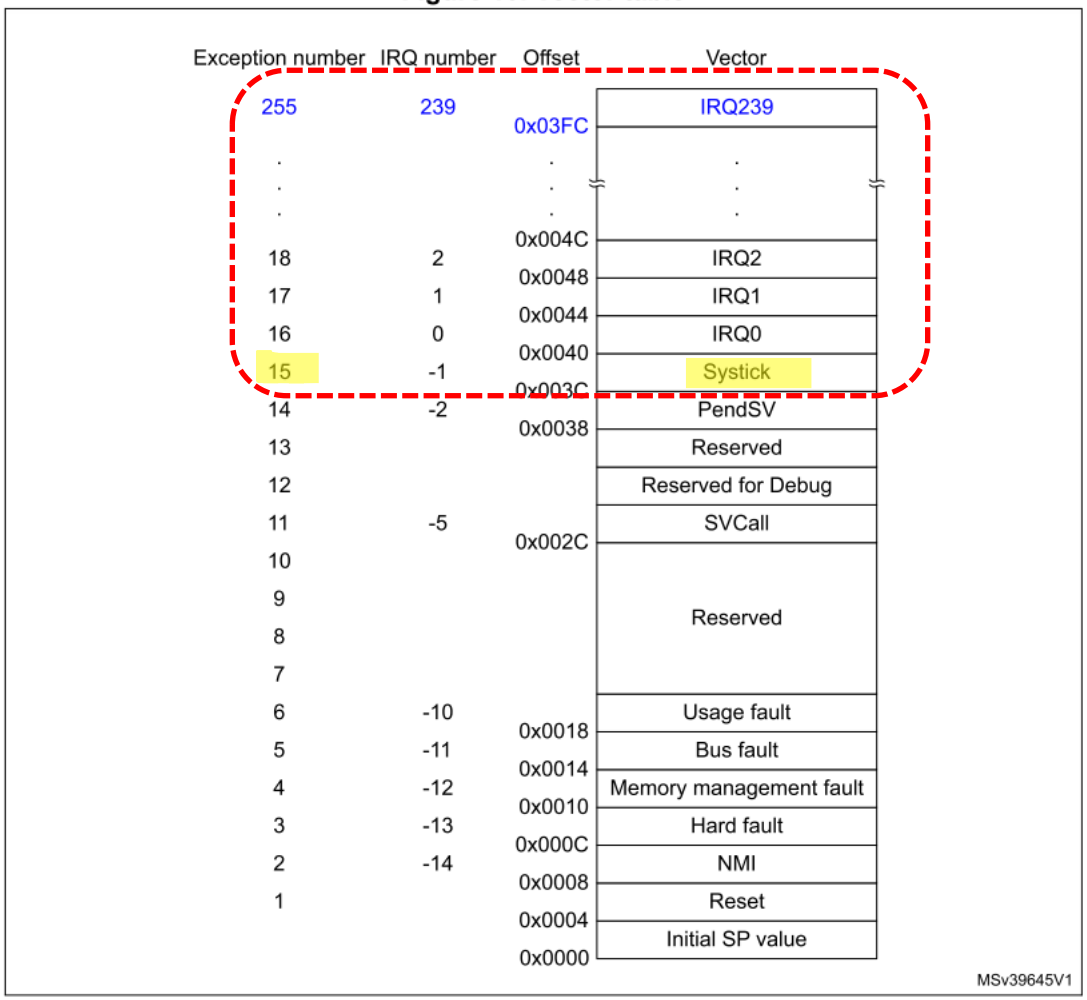


Table 74. SYST_CVR bit assignments

Bits	Name	Function
[31:24]	-	Reserved.
[23:0]	CURRENT	Reads return the current value of the SysTick counter. A write of any value clears the field to 0, and also clears the SYST_CSR COUNTFLAG bit to 0.

Prekinitve - vektorska tabela

Figure 10. Vector table



Prekinitveni vektorji

```
// Start of text section
.section .text
///////////////////////////////////////////////////////////////////////////
// Vectors
///////////////////////////////////////////////////////////////////////////
// Vector table start
// Add all other processor specific exceptions/interrupts in order here
.long    __StackTop                  // Top of the stack. from linker script
.long    _start +1                   // reset location, +1 for thumb mode
  .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

  /* External Interrupts */
  .word     WWDG_IRQHandler                 /* Window WatchDog              */
  .word     PVD_IRQHandler                  /* PVD through EXTI Line detection */
  .word     TAMP_STAMP_IRQHandler           /* Tamper and TimeStamps through the EXTI line */
  .word     RTC_WKUP_IRQHandler             /* RTC Wakeup through the EXTI line */
```

Prekinitveni vektorji – začetne nastavitve

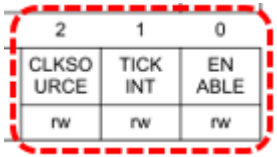
```
/*******************************************************************************
*
* Provide weak aliases for each Exception handler to the Default_Handler.
* As they are weak aliases, any function with the same name will override
* this definition.
*
*******************************************************************************/
    .weak      NMI_Handler
    .thumb_set NMI_Handler,Default_Handler

    .weak      HardFault_Handler
    .thumb_set HardFault_Handler,Default_Handler

    .weak      MemManage_Handler
    .thumb_set MemManage_Handler,Default_Handler

    .weak      BusFault_Handler
    .thumb_set BusFault_Handler,Default_Handler

    .weak      UsageFault_Handler
    .thumb_set UsageFault_Handler,Default_Handler

    .weak      SVC_Handler
    .thumb_set SVC_Handler,Default_Handler

    .weak      DebugMon_Handler
    .thumb_set DebugMon_Handler,Default_Handler

    .weak      PendSV_Handler
    .thumb_set PendSV_Handler,Default_Handler

    .weak      SysTick_Handler
    .thumb_set SysTick_Handler,Default_Handler
```

```
/**
 * @brief  This is the code that gets called when the
processor receives an
 *         unexpected interrupt.  This simply enters
an infinite loop, preserving
 *         the system state for examination by a
debugger.
 * @param  None
 * @retval None
*/
    .section  .text.Default_Handler,"ax",%progbits
Default_Handler:
Infinite_Loop:
  b  Infinite_Loop
  .size  Default_Handler, .-Default_Handler
```

SysTick Časovnik – krmiljenje

Potrebni koraki za krmiljenje časovnika SysTick:

1. **SYST_RVR (Reload Value Register): Value** SYSTICK_RELOAD_1MS
2. **SYST_CVR (Current Value Register): 0, reset to zero**
3. **SYST_CSR (Control/Status Register): 0b111 : Proc. Clock, TickInt, Enable**
   -> Start SysTick



4. **Delovanje:**
   Proženje SysTick_Handler vsako 1 ms

SysTick_Handler :

```
.global SysTick_Handler

.section  .text.SysTick_Handler,"ax",%progbits
.type  SysTick_Handler, %function


SysTick_Handler:
    push {r3, r4, r5, lr}
    …
RET: pop {r3, r4, r5, pc}
```

Števec v r8 šteje 500 ms
Zastavica v r7 pove stanje LED diod
Vse se dogaja v PSP !

Prekinitve *(Register za nastavitve vektorske tabele v RAM pomnilniku)*

4.3.4 Vector table offset register

The VTOR indicates the offset of the vector table base address from memory address 0x00000000. See the register summary in *Table 50 on page 192* for its attributes. The bit assignments are:

Figure 27. VTOR bit assignments

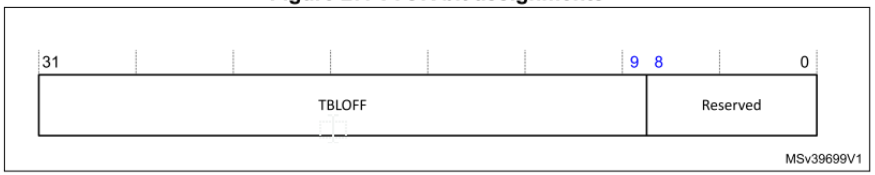


Table 54. VTOR bit assignments

Bits	Name	Function
[31:9]	TBLOFF	Vector table base offset field. It contains bits [29:7] of the offset of the table base from the bottom of the memory map.
[8:0]	-	Reserved.

When setting TBLOFF, the user must align the offset to the number of exception entries in the vector table.

The table alignment requirements mean that bits [8:0] of the table offset are always zero.

4.3 System control block

The *System Control Block* (SCB) provides system implementation information, and system control. This includes configuration, control, and reporting of the system exceptions. The system control block registers are:

Table 50. Summary of the system control block registers

Address	Name	Type	Required privilege	Reset value	Description
0xE000E008	ACTLR	RW	Privileged	0x00000000	*Auxiliary control register on page 193*
0xE000ED00	CPUID	RO	Privileged	0x410FC270	*CPUID base register on page 194*
0xE000ED04	ICSR	RW(1)	Privileged	0x00000000	*Interrupt control and state register on page 194*
0xE000ED08	VTOR	RW	Privileged	Unknown	*Vector table offset register on page 197*
0xE000ED0C	AIRCR	RW(1)	Privileged	0xFA050000	*Application interrupt and reset control register on page 197*

Dodamo samo, če poganjamo kodo iz RAM pomnilnika !!!
Sicer je naslov vektorske tabele 0x08000000 (Flash)

```
.equ VTOR,0xE000ED08

ldr r1, =VTOR
ldr r0, =0x24000000
str r0, [r1] ldr
```

Register	Address	Value
Control		
ACTLR	0xe000e008	0x1000
ICSR	0xe000ed04	0x0
VTOR	0xe000ed08	0x8000000
TBLOFF	[7:25]	0x100000
TBLBASE	[29:1]	0x0

Register	Address	Value
Control		
ACTLR	0xe000e008	0x1000
ICSR	0xe000ed04	0x0
VTOR	0xe000ed08	0x24000000
TBLOFF	[7:25]	0x480000
TBLBASE	[29:1]	0x1

SysTick Časovnik s prekinitvami – osnova

Ostala koda :

```
// Vector table offset register definition
// Important for relocated Vector table on running from RAM
        .equ VTOR,0xE000ED08

// Start of data section
  .data
LEDSTAT: .word   0   // LED state
MSECCNT: .word   0   //MSecs counter for SysTick_Handler
MSECMAX: .word   500 //MSecs interval for SysTick_Handler
…
// Start of text section
  .text

  .type  main, %function
  .global main
  .align
main:

    bl INIT_IO     // Priprava za kontrolo LED diode

    ldr r1, =VTOR // Set Vector table addr. to 0x24000000
    ldr r0, =0x24000000
    str r0, [r1]

    bl INIT_TC_PSP // Priprava SysTick časovnika s prek.

__end: b __end
```

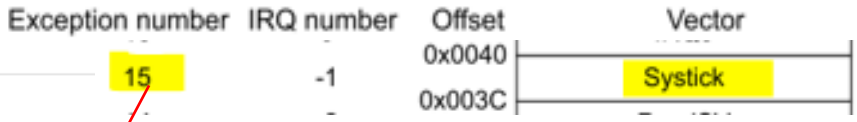
```
INIT_TC_PSP:

      ???

      pop {r0, r1, pc}
```

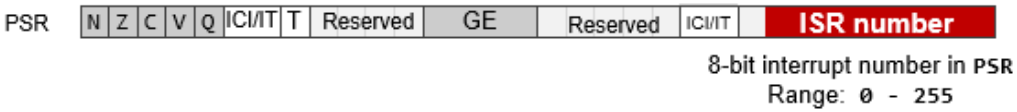
SysTick_Handler :

```
.global SysTick_Handler
.section  .text.SysTick_Handler,"ax",%progbits
.type  SysTick_Handler, %function

SysTick_Handler:
      push {r3, r4, r5, r6, lr}


      ???

RET:  pop {r3, r4, r5, r6, pc}
```

CubeIDE – Registers okno

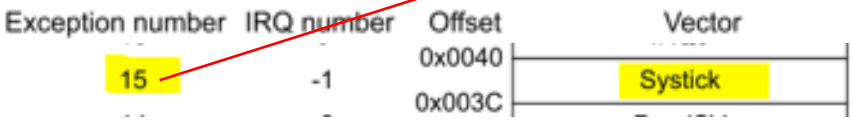
Name	Value
General Registers	
r0	603979776 (Decimal)
r1	0xe000ed08
r2	0x40000001
r3	0x0
r4	0x2000002c
r5	0x0
r6	0x0
r7	0x0
r8	0x0
r9	0x0
r10	0x0
r11	0x0
r12	0x0
sp	0x2001ffcc
lr	0xfffffff9
pc	0x24000386
xpsr	0x6100000f



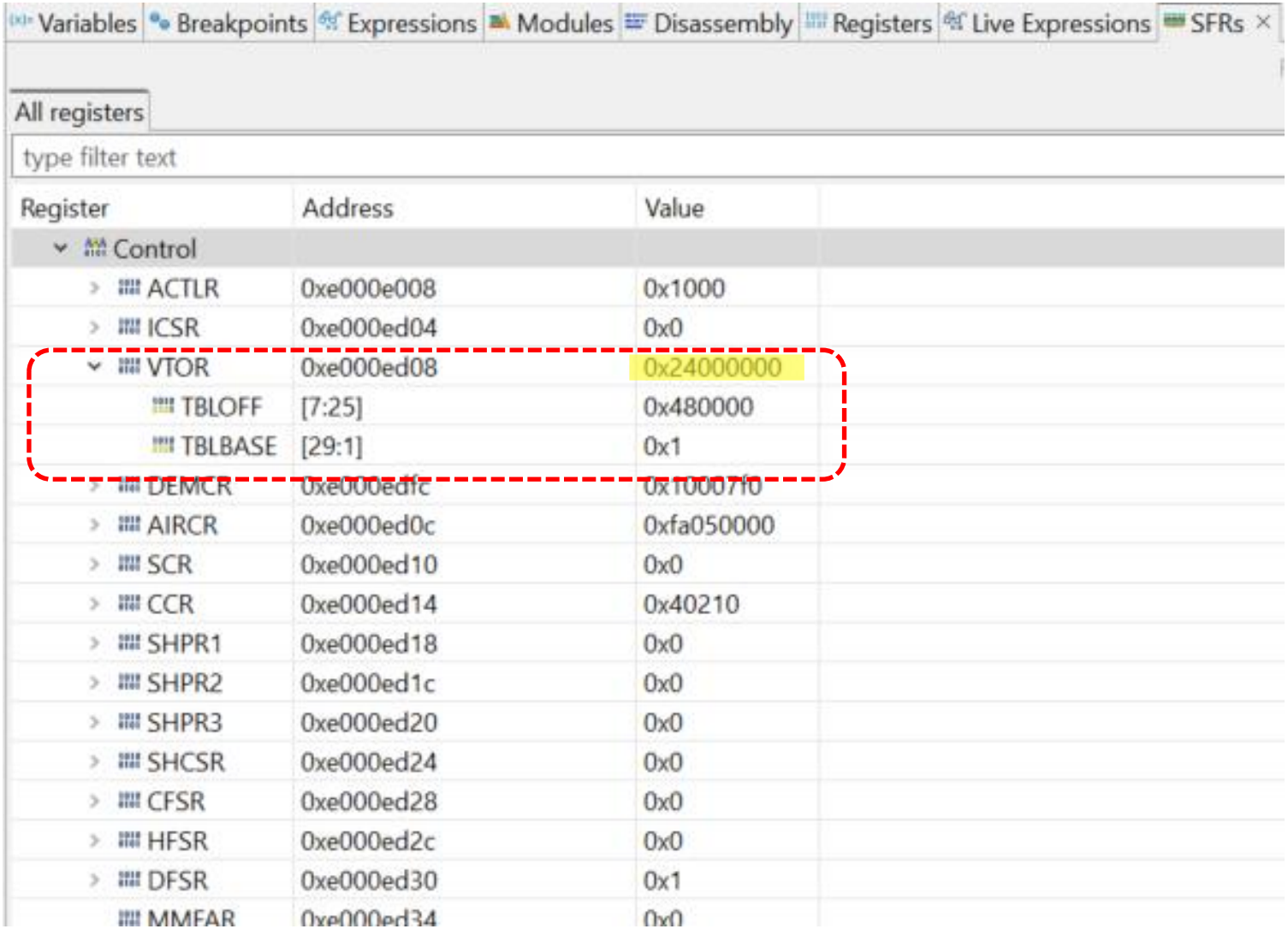


CubeIDE – SFR okno

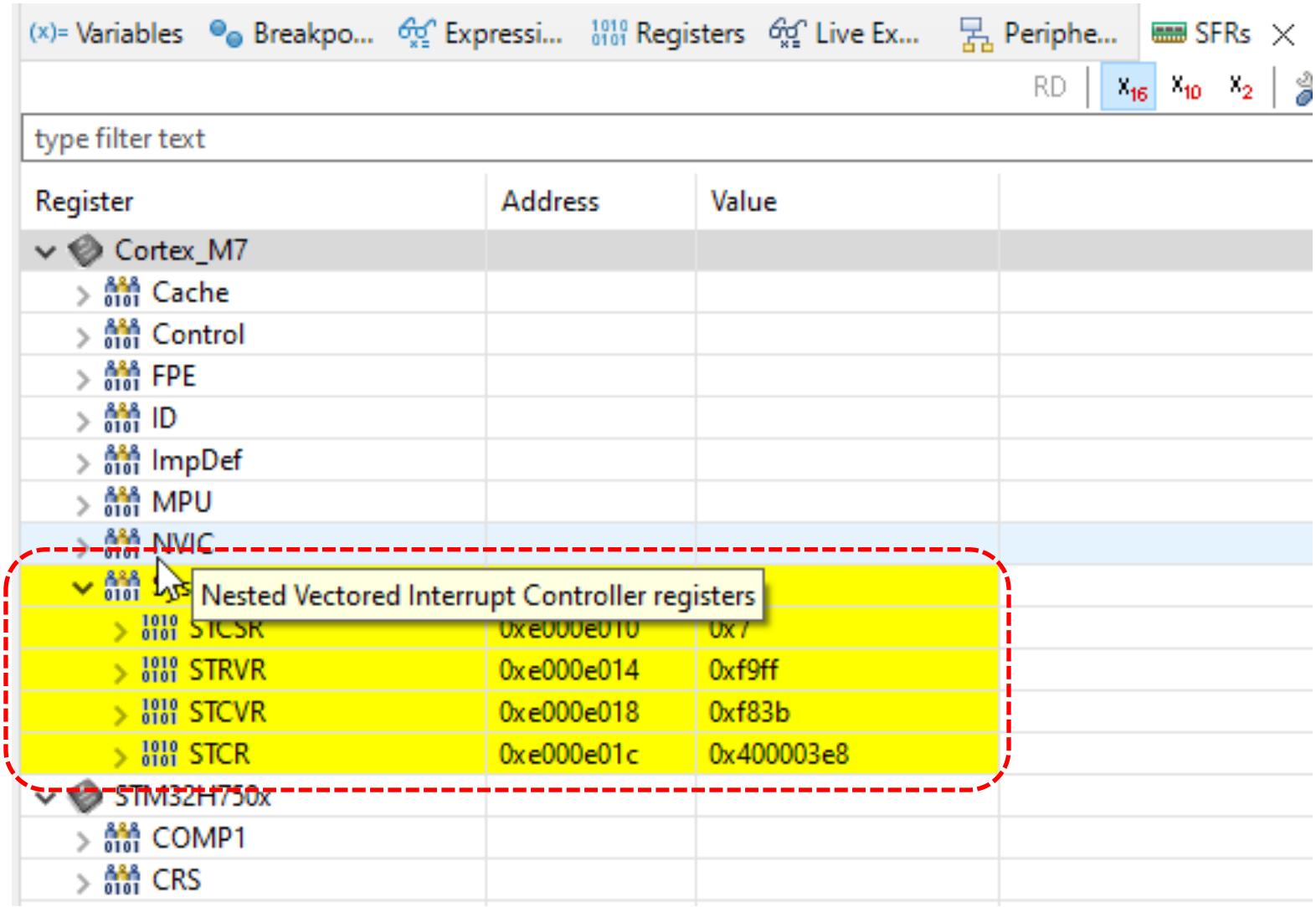
Register	Address	Value
Cortex_M7		
Cache		
Control		
ACTLR	0xe000e008	0x1000
ICSR	0xe000ed04	0x80f
NMIPENDSET	[31:1]	0x0
PENDSVSET	[28:1]	0x0
PENDSVCLR	[27:1]	0x0
PENDSTSET	[26:1]	0x0
PENDSTCLR	[25:1]	0x0
ISRPREEMPT	[23:1]	0x0
ISRPENDING	[22:1]	0x0
VECTPENDING	[12:9]	0x0
RETTOBASE	[11:1]	0x1
VECTACTIVE	[0:9]	0xf



CubeIDE – SFR okno



CubeIDE – SFR okno



SysTick Časovnik s prekinitvami – koda

Ostala koda :

```
// Vector table offset register definition
// Important for relocated Vector table on running from RAM
        .equ VTOR,0xE000ED08

// Start of data section
  .data
LEDSTAT: .word   0   // LED state
MSECCNT: .word   0   //MSecs counter for SysTick_Handler
MSECMAX: .word   500 //MSecs interval for SysTick_Handler
…
// Start of text section
  .text

  .type  main, %function
  .global main
  .align
main:

    bl INIT_IO     // Priprava za kontrolo LED diode

    ldr r1, =VTOR // Set Vector table addr. to 0x24000000
    ldr r0, =0x24000000
    str r0, [r1]

    bl INIT_TC_PSP // Priprava SysTick časovnika s prek.

__end: b __end
```

```
INIT_TC_PSP:

      push {r0, r1, lr}
      ldr r1, =SCS_BASE

      ldr r0, =SYSTICK_RELOAD_1MS
      str r0, [r1, #SCS_SYST_RVR]

      mov r0, #0
      str r0, [r1, #SCS_SYST_CVR]

      mov r0, #0b111 // TickINT Bit also set to 1
      str r0, [r1, #SCS_SYST_CSR]

      pop {r0, r1, pc}
```

SysTick Časovnik – PSP

SysTick_Handler :

```
.global SysTick_Handler
.section  .text.SysTick_Handler,"ax",%progbits
.type  SysTick_Handler, %function

SysTick_Handler:
      push {r3, r4, r5, r6, lr}


      ldr r3,=MSECMAX // Load MAX value
      ldr r5,[r3]
      ldr r3,=MSECCNT // Load MSecs Counter value
      ldr r4,[r3]
      add r4,r4,#1 // Increment (+1)
      str r4,[r3]
      cmp r4,r5 // End of interval ?
      blo RET


      // Reset counter state and check LED
      mov r4,#0
      str r4,[r3]


      ldr r3,=LEDSTAT
      ldr r4,[r3]
      cmp r4,#0


      beq LON
      …
```

```
      …
      mov r5, #LEDs_OFF
      mov r4,#0
      str r4,[r3]
      // mov r7,#0 // set LED status in r7


      b CONT

LON:  mov r5, #LEDs_ON
      mov r4,#0xff
      str r4,[r3]
      // mov r7,#0xff // set LED status in r7

CONT:
      // Set GPIOI Pins through BSSR register
      ldr r6, =GPIOI_BASE // Load GPIOD BASE address to r6
      str r5, [r6,#GPIOx_BSRR] // Write to BSRR register


RET:  pop {r3, r4, r5, r6, pc}
```