Organizacija računalnikov

Laboratorijske vaje

R. Rozman 2024

Vsebina vaj

2 tematska sklopa :

1. **Programiranje v zbirnem jeziku ARM (4 vaje)**
   - Ponovitev RA, razširitev, nadgradnja:
     - Delo z biti, sklad, podprogrami

Primeri in osnove tudi v jeziku C (neobvezno)

2. **Sistemske naprave v zbirniku (7 vaj) - *(STM32H750B-DK Discovery, STM32F4 Discovery, FRI-SMS)***
   - Paralelni vhod/izhod: (G)PIO
   - Časovniki: TIM, SysTick, TC
   - Serijske povezave: U(S)ART, USB VCOM port, DBGU
   - Prekinitve, prekinitveni krmilnik, Mini RTOS (AIC)
3. **MiMo lab. vaja (vmes)**

2 obvezni in 2 neobvezni domači nalogi

1. MiMo, osnovna (obv.), MiMo dodatno delo (neobv.)
2. ARM,STM32 - aplikacija (obv.), ARM,STM-Dodatna (razširitve, aplikacija, senzorji) (neobv.)

- Vsako dodatno delo šteje !!!

Vsebina vaj

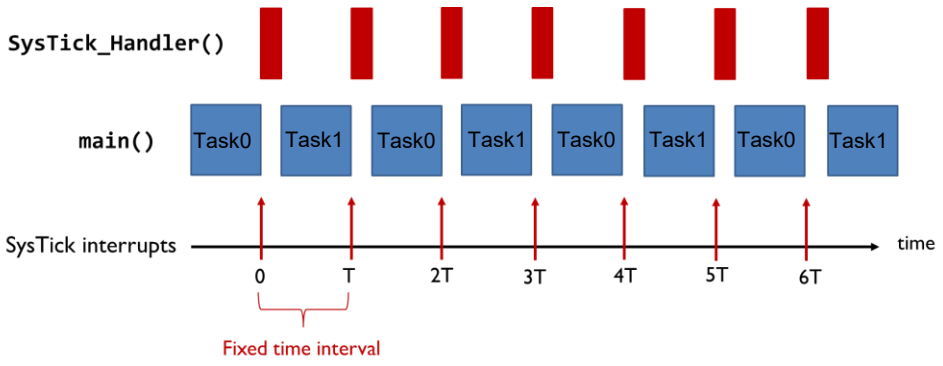
Primer zadnje vaje – Mini RTOS :

1. Mini RTOS

- Preprost operacijski sistem za več procesov
  - 1000x v sekundi preklopi med procesoma
- Task0: Blink LED1
- Task1: Blink LED2

```
TASK0_Start:

bl LED1_ON
mov r0,#0
mov r1,#'1'
bl ITM_Send
mov r0,#'0'
bl SEND_UART

mov r0,#500
bl DELAY // Zakasnitev SW Delay: r0 x 1msec

bl LED1_OFF
mov r0,#0
mov r1,#'0'
bl ITM_Send

mov r0,#500
bl DELAY // Zakasnitev SW Delay: r0 x 1msec

b TASK0_Start
```

```
TASK1_Start:

bl LED2_OFF
mov r0,#1
mov r1,#'+'
bl ITM_Send
mov r0,#'1'
bl SEND_UART

mov r0,#500
bl DELAY // Zakasnitev SW Delay: r0 x 1msec

bl LED2_ON
mov r0,#1
mov r1,#'-'
bl ITM_Send

mov r0,#500
bl DELAY // Zakasnitev SW Delay: r0 x 1msec

b TASK1_Start
```



Ocenjevanje OR

- Vaje prispevajo 50% h končni oceni in morajo biti opravljene:
  - Pozitivne domače naloge (obvezni del),
  - Dodatne domače naloge (neobvezni del – višja ocena).
    - Se prišteje obveznemu delu

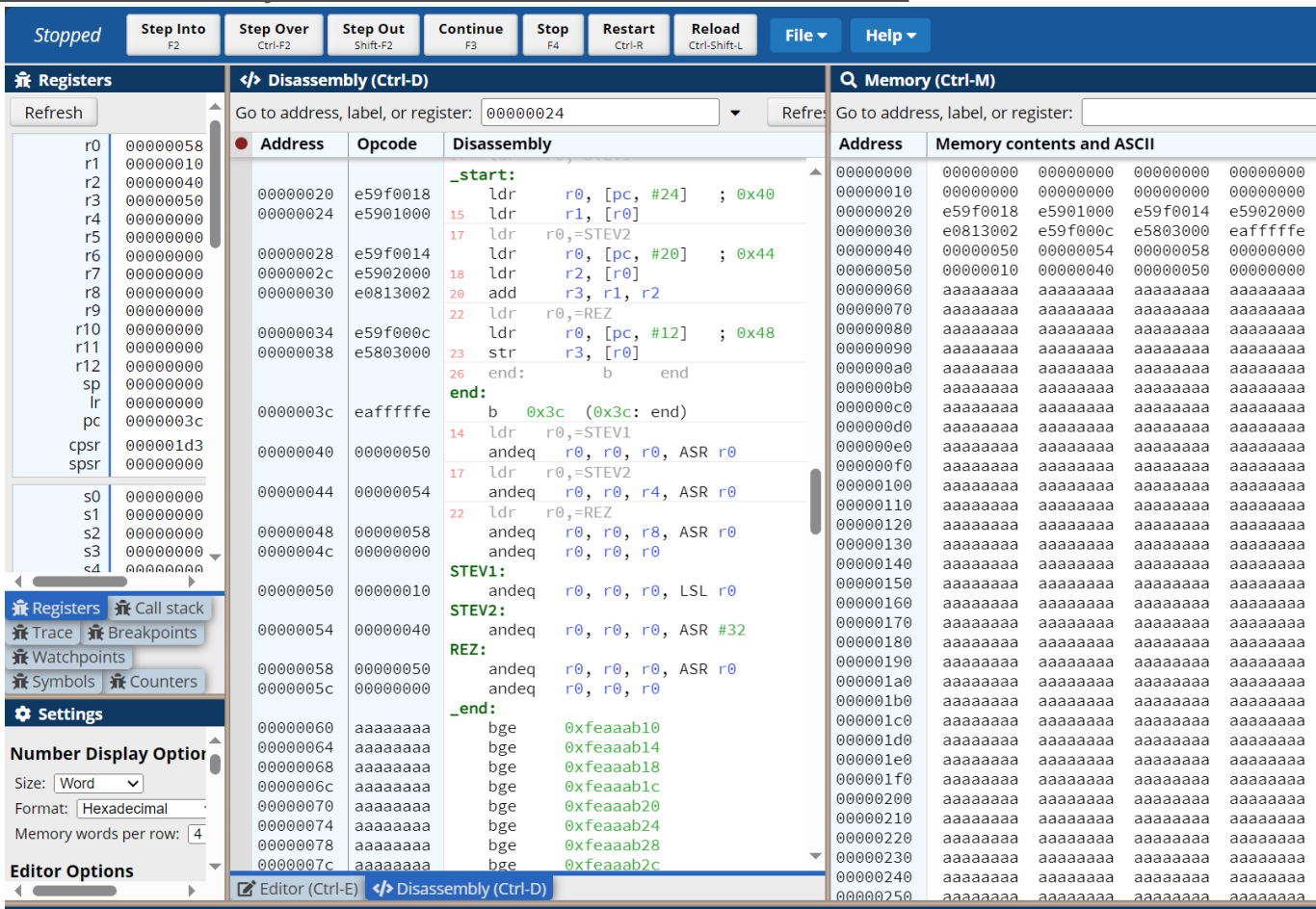
  - *2022: Vzporedno uvajanje STM32F4, STM32H7 Discovery*
  - *2023: Vzporedno: STM32H7, STM32F4, FRI-SMS*
  - *2024: Osrednja platforma: STM32H7, (STM32F4, FRI-SMS)*

- Ustni izpit 50%

Spletni simulator cpulator – 1.sklop

- **https://cpulator.01xz.net/?sys=arm**
- **začetni projekt OR:**
  - https://cpulator.01xz.net/?sys=arm&loadasm=share/s8zU3xx.s

```
        .text
        .org 0x20

@spremenljivke
STEV1:  .word   0x10
STEV2:  .word   0x40
REZ:    .word   0

        .align

        .global _start
_start:

        adr     r0,STEV1
        ldr     r1,[r0]

        adr     r0,STEV2
        ldr     r2,[r0]

        add     r3,r1,r2

        adr     r0,REZ
        str     r3,[r0]


end:    b       end
```

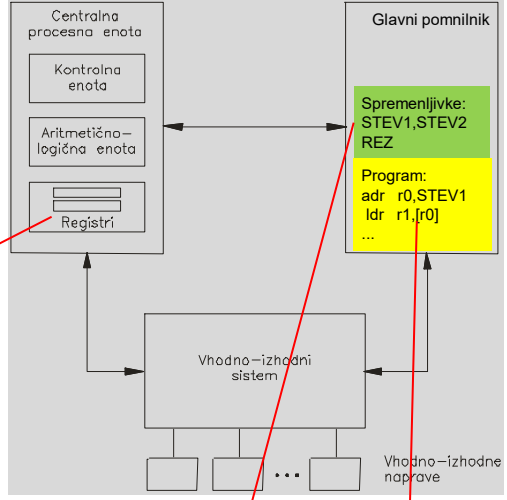


RA - Praktično delo: vsota dveh števil

https://cpulator.01xz.net/?sys=arm&loadasm=share/s8zU3xx.s

Zbirni jezik	Opis ukaza	Strojni jezik
adr r0, stev1	R0 ← nasl. stev1	0xE24F0014
ldr r1, [r0]	R1 ← M[R0]	0xE5901000
adr r0, stev2	R0 ← nasl. stev2	0xE24F0018
ldr r2, [r0]	R2 ← M[R0]	0xE5902000
add r3, r2, r1	R3 ← R1 + R2	0xE0823001
adr r0, rez	R0 ← nasl. rez	0xE24F0020
str r3, [r0]	M[R0] ← R3	0xE5803000

RA - Ponovitev



Stopped | Step Into F2 | Step Over Ctrl-F2 | Step Out Shift-F2 | Continue F3 | Stop F4 | Restart Ctrl-R | Reload Ctrl-Shift-L | File | Help

Registers

Refresh

r0	00000000	
r1	00000000	
r2	00000000	
r3	00000000	
r4	00000000	
r5	00000000	
r6	00000000	
r7	00000000	
r8	00000000	
r9	00000000	
r10	00000000	
r11	00000000	
r12	00000000	
sp	00000000	
lr	00000000	
pc	0000002c	
cpsr	000001d3	NZCVI SVC
spsr	00000000	NZCVI ?

s0	00000000
s1	00000000
s2	00000000
s3	00000000
s4	00000000
s5	00000000
s6	00000000

Disassembly (Ctrl-D)

Go to address, label, or register: 00000000

Address	Opcode	Disassembly
		STEV1:
00000020	00000010	andeq r0, r0, r0,
		STEV2:
00000024	00000040	andeq r0, r0, r0,
		REZ:
00000028	00000000	andeq r0, r0, r0
		9 .align
		11 .global _start
		12 _start:
		14 adr r0,STEV1
		_start:
0000002c	e24f0014	adr r0, 0x20 (0
00000030	e5901000	15 ldr r1, [r0]
		17 adr r0,STEV2
00000034	e24f0018	adr r0, 0x24 (0
00000038	e5902000	18 ldr r2, [r0]
0000003c	e0813002	20 add r3, r1, r2
		22 adr r0,REZ
00000040	e24f0020	adr r0, 0x28 (0
00000044	e5803000	23 str r3, [r0]
		26 end: b er
		end:
00000048	eafffffe	b 0x48 (0x48: enc

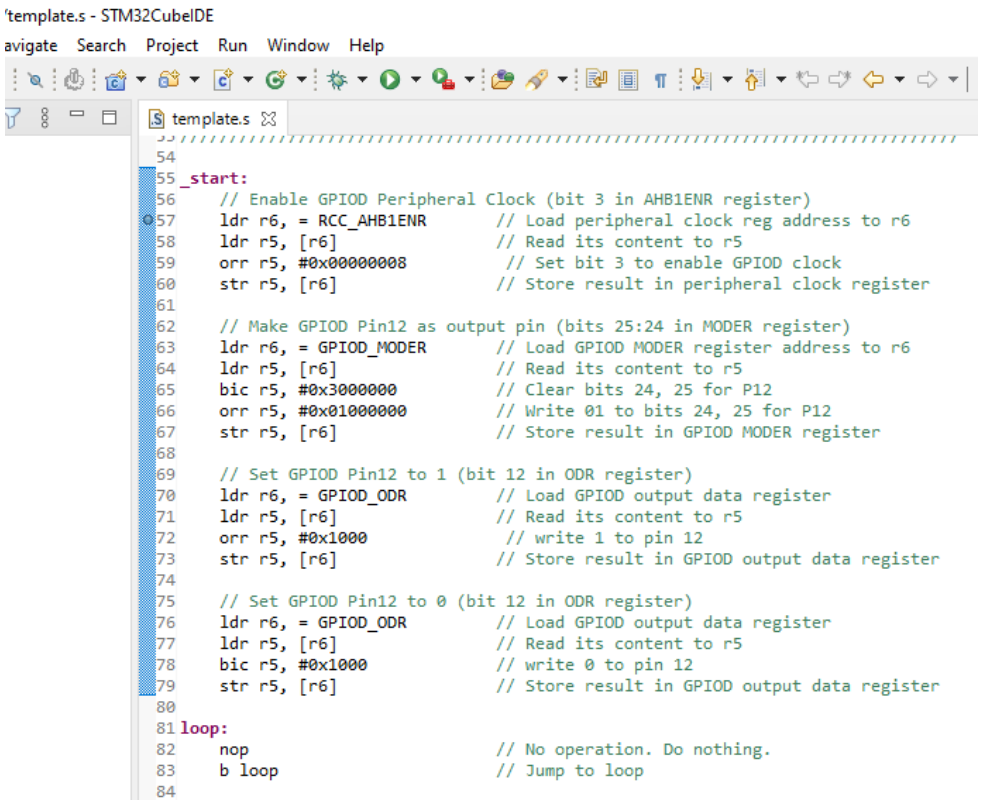
Memory (Ctrl-M)

Go to address, label, or register:

Address	Memory contents and ASCII			
00000000	00000000	00000000	00000000	00000000
00000010	00000000	00000000	00000000	00000000
00000020	00000010	00000040	00000000	e24f0014
00000030	e5901000	e24f0018	e5902000	e0813002
00000040	e24f0020	e5803000	eafffffe	00000000
00000050	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa
00000060	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa
00000070	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa
00000080	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa
00000090	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa
000000a0	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa
000000b0	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa
000000c0	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa
000000d0	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa
000000e0	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa
000000f0	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa
00000100	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa
00000110	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa
00000120	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa
00000130	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa
00000140	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa
00000150	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa
00000160	aaaaaaaa	aaaaaaaa	aaaaaaaa	aaaaaaaa

Razvojno okolja 2. sklop: CubeIDE

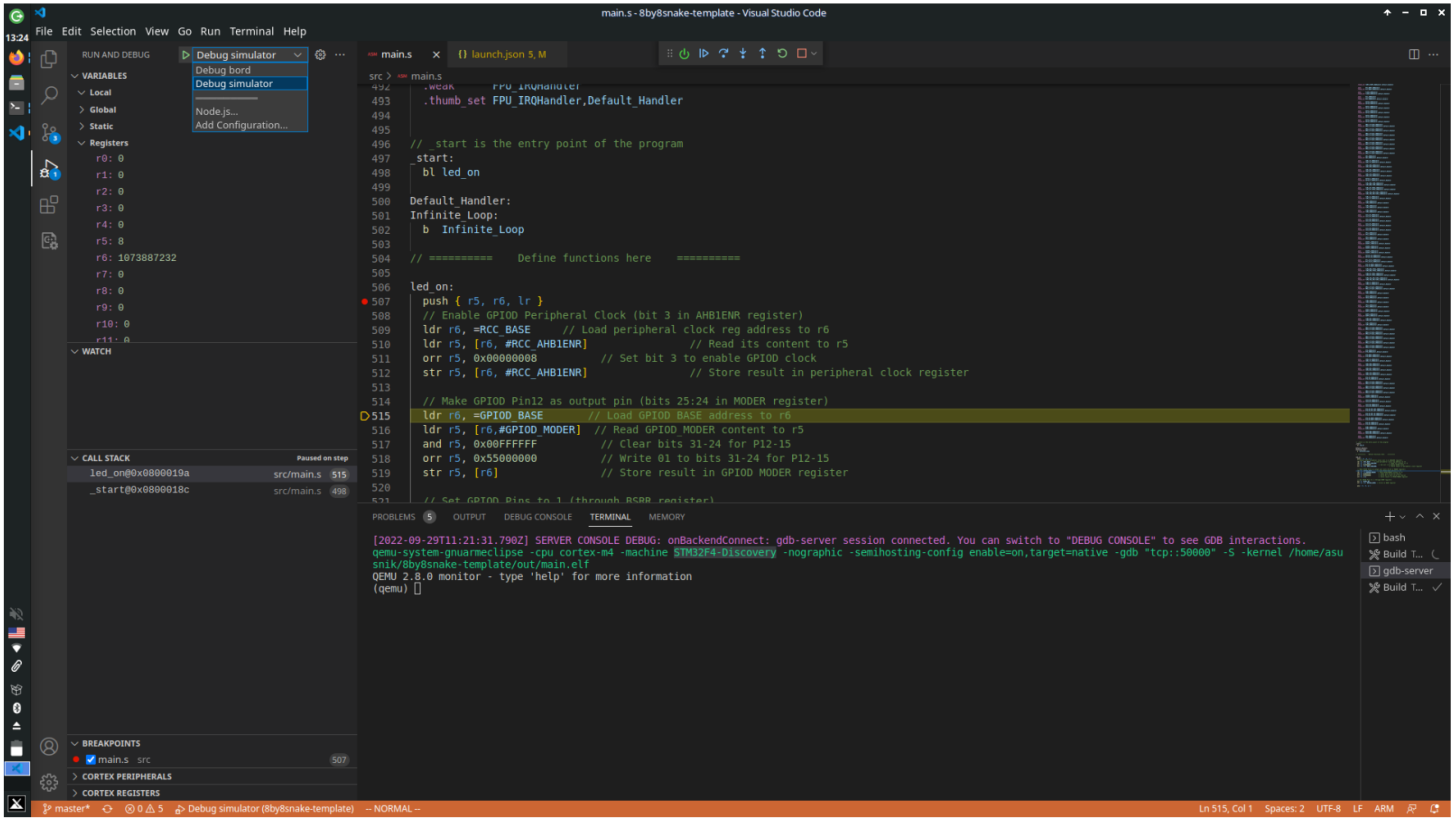
Delo s ploščami: STM32H7, STM32F4



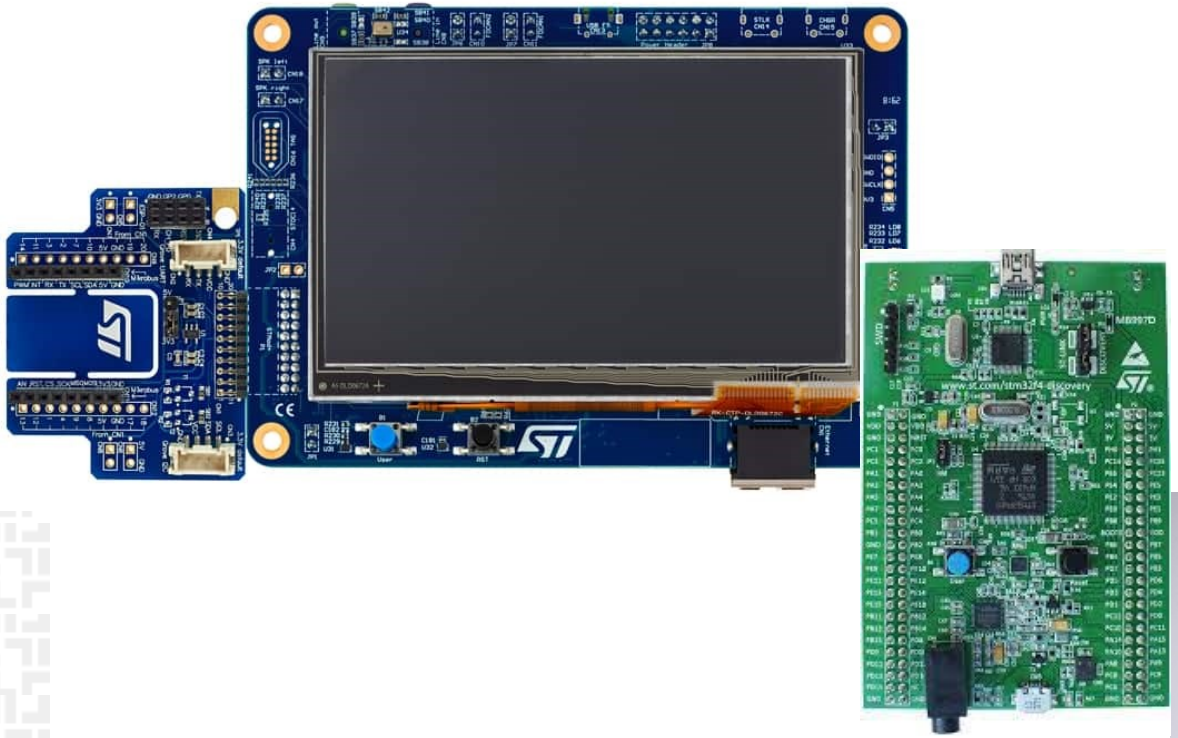

```
54
55 _start:
56     // Enable GPIOD Peripheral Clock (bit 3 in AHB1ENR register)
57     ldr r6, = RCC_AHB1ENR        // Load peripheral clock reg address to r6
58     ldr r5, [r6]                 // Read its content to r5
59     orr r5, #0x00000008           // Set bit 3 to enable GPIOD clock
60     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
```

Razvojno okolja 2. sklop: VSCode

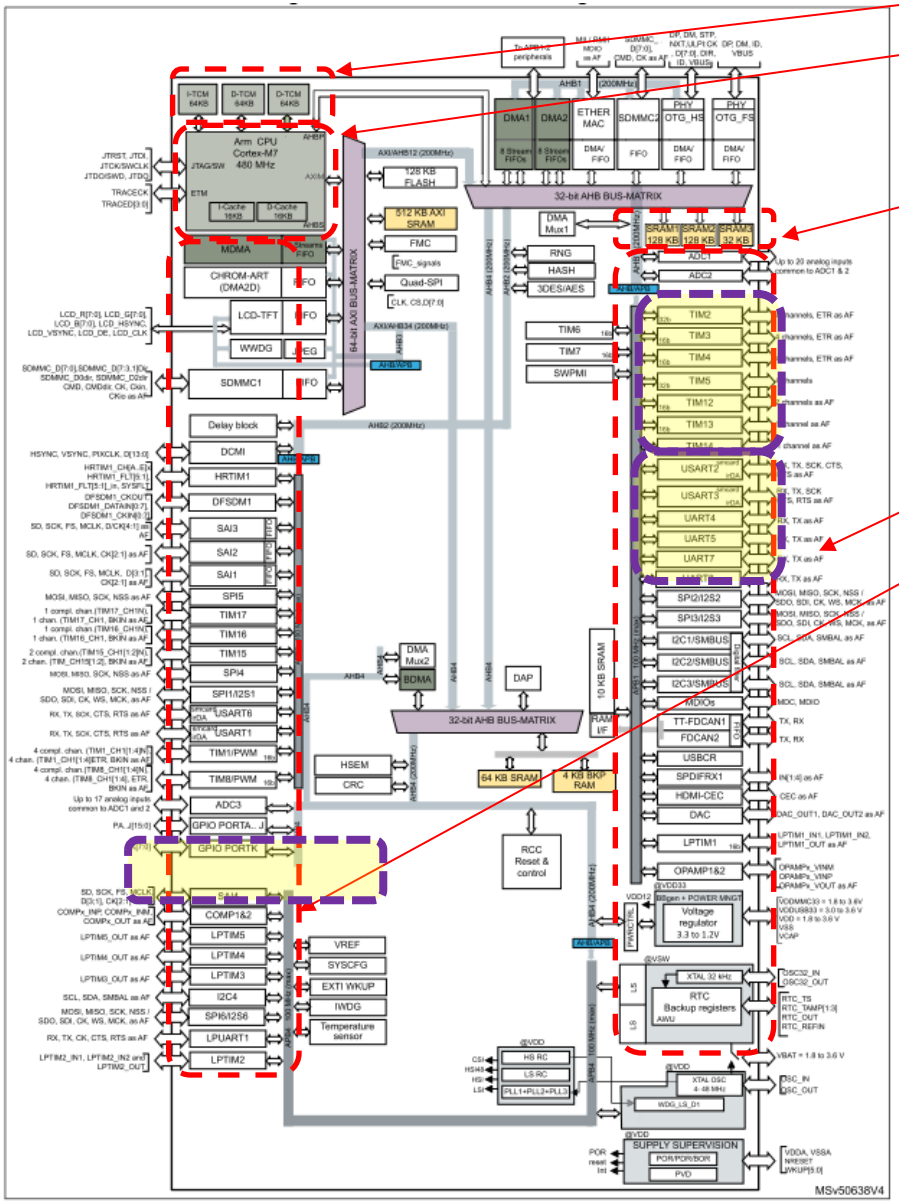
Delo s ploščami: STM32H7, STM32F4
(ni „uradno“ podprt, je pa zanimiv)

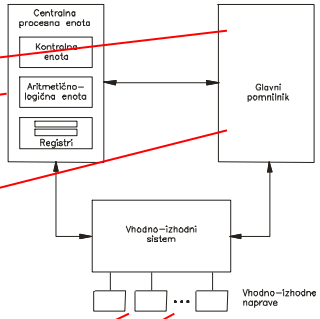


V 2. sklopu – delo s sistemi STM32H7, STM32F4 , FRI SMS

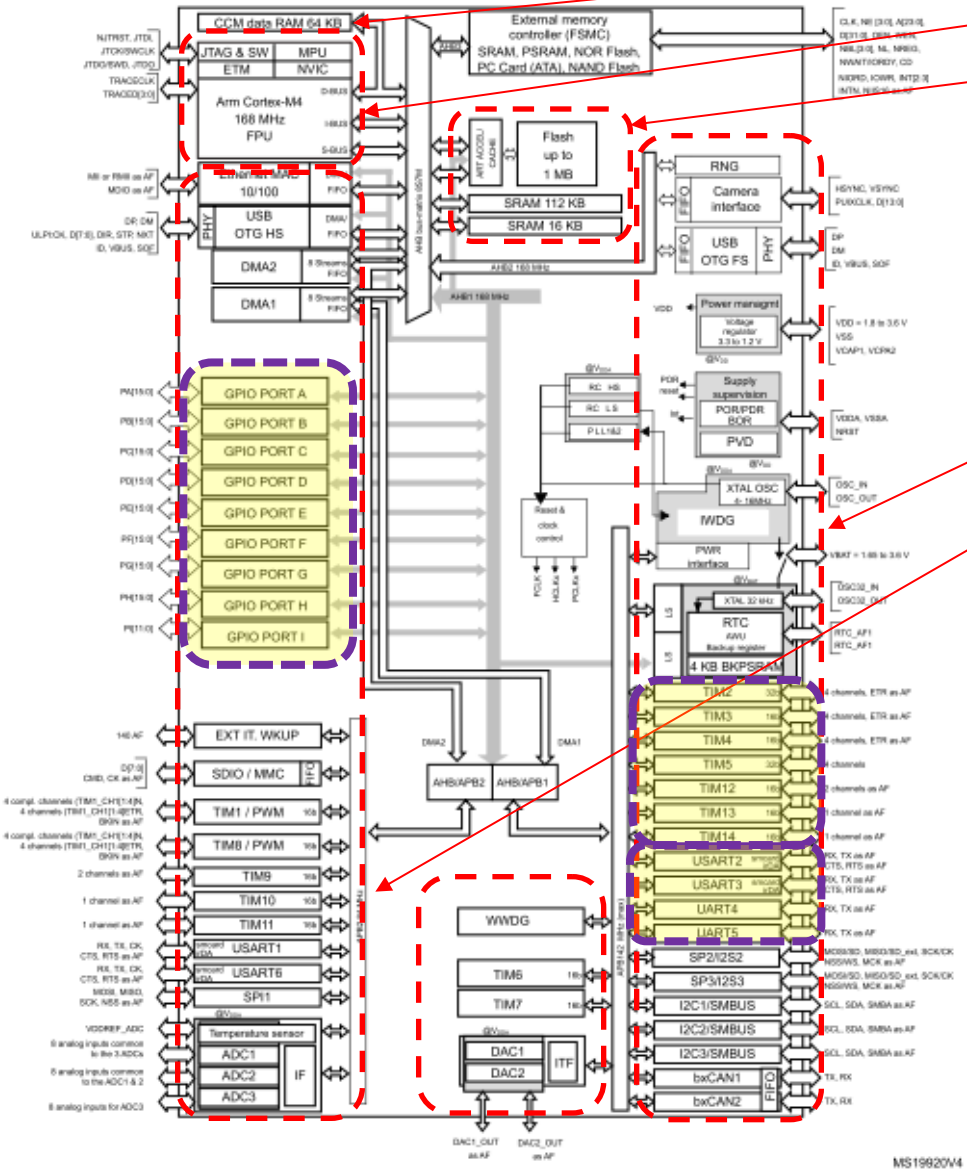


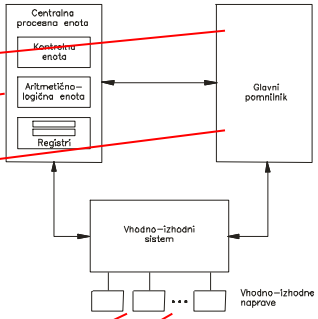
STM32H750: GPIO, TIMer, USART



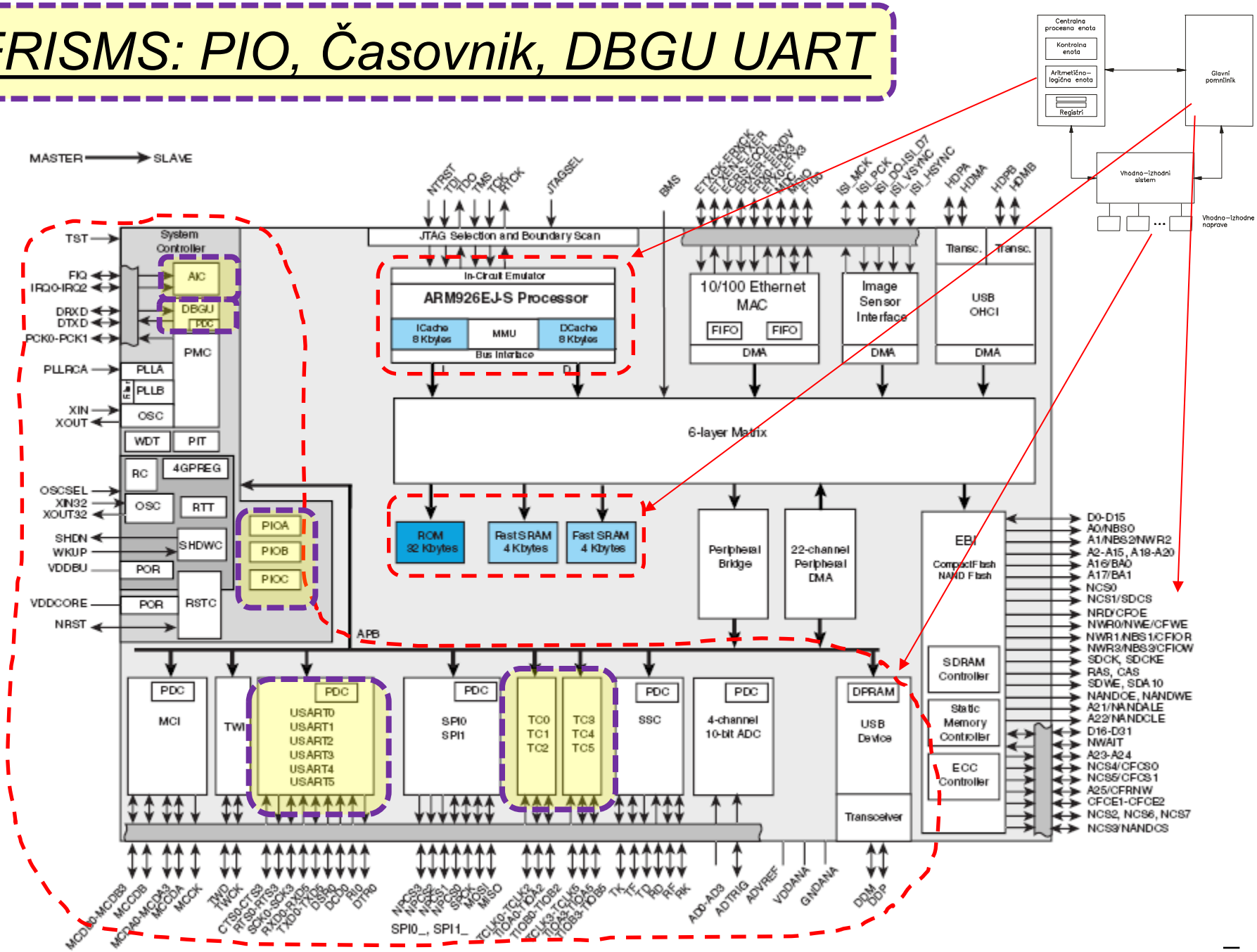


STM32F407VG: GPIO, TIMer, USART





FRISMS: PIO, Časovnik, DBGU UART



Iščete vsebine vaj, dodatne izzive,
pomoč, ideje za projekte ?

Dobrodošli:

- e-učilnica

- MS Teams
  - Komunikacija (govorilne ure, pogovor, pomoč),
  - OneNote zvezek z zapiski predavanj

vx9k5no
OR VSP 2024/25

- Discord(https://discord.gg/nmzjQU7me7)

ty5qjm9

LAPSy Embedded Academy

- LEA - Lapsy Embedded Academy
  - Projekt spletnega izobraževalnega portala z vsebinami RA, OR in VIN (ter ostalimi projekti, diplomami, tečaji, …)
  - https://unilj.sharepoint.com/sites/LAPSYEmbeddedAcademy/