Organizacija računalnikov

Laboratorijske vaje

R. Rozman 2023

Vsebina vaj

2 tematska sklopa :

Primeri in osnove tudi v jeziku C (neobvezno)

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

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

- **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 !!!

Ocenjevanje*

- 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

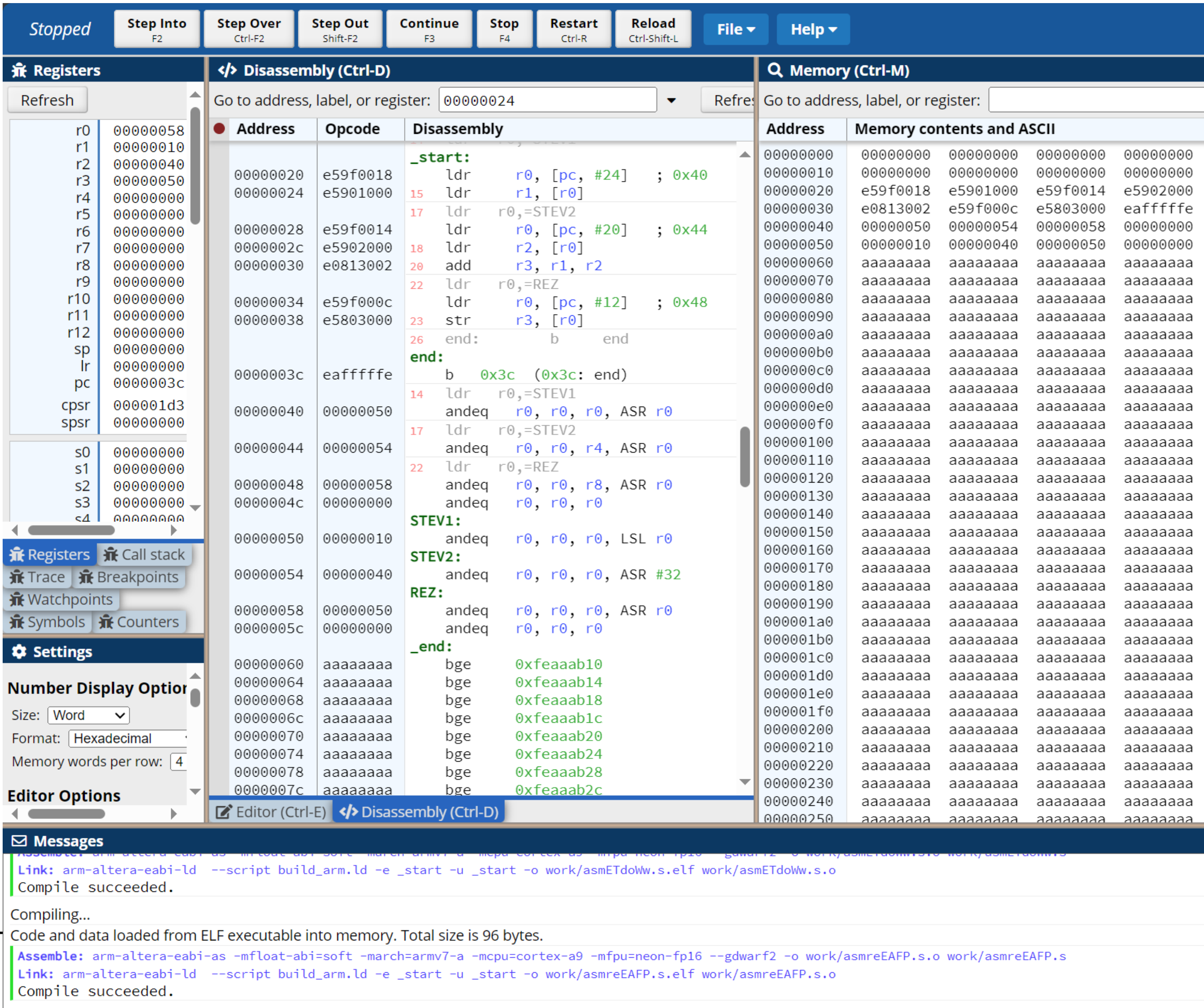
  - *2022: Vzporedno uvajanje STM32F4, STM32H7 Discovery*
  - *2023: Vzporedno: STM32H7, STM32F4, FRI-SMS*

** Zaradi Covid situacije se lahko še prilagodi*

Spletni simulator cpulator – 1.sklop

- **https://cpulator.01xz.net/?sys=arm**
- **začetni projekt OR:**
  - https://cpulator.01xz.net/?sys=arm&loadasm=share/sN7suQe.s

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

        .text
        .org 0x20
        .align

        .global _start
_start:

              ldr    r0,=STEV1
              ldr    r1,[r0]

              ldr    r0,=STEV2
              ldr    r2,[r0]

              add    r3,r1,r2

              ldr    r0,=REZ
              str    r3,[r0]


end:          b      end
```



```
r0    00000058
r1    00000010
r2    00000040
r3    00000050
r4    00000000
r5    00000000
r6    00000000
r7    00000000
r8    00000000
r9    00000000
r10   00000000
r11   00000000
r12   00000000
sp    00000000
lr    00000000
pc    0000003c
cpsr  000001d3
spsr  00000000

s0    00000000
s1    00000000
s2    00000000
s3    00000000
```

Number Display Options
Size: Word
Format: Hexadecimal
Memory words per row: 4

Editor Options

Go to address, label, or register: 00000024

```
Address   Opcode    Disassembly
                    _start:
00000020  e59f0018      ldr     r0, [pc, #24]     ; 0x40
00000024  e5901000      ldr     r1, [r0]
                        ldr   r0,=STEV2
00000028  e59f0014      ldr     r0, [pc, #20]     ; 0x44
0000002c  e5902000      ldr     r2, [r0]
00000030  e0813002      add     r3, r1, r2
                        ldr   r0,=REZ
00000034  e59f000c      ldr     r0, [pc, #12]     ; 0x48
00000038  e5803000      str     r3, [r0]
                        end:          b       end
                    end:
0000003c  eafffffe      b    0x3c    (0x3c: end)
                        ldr   r0,=STEV1
00000040  00000050      andeq   r0, r0, r0, ASR r0
                        ldr   r0,=STEV2
00000044  00000054      andeq   r0, r0, r4, ASR r0
                        ldr   r0,=REZ
00000048  00000058      andeq   r0, r0, r8, ASR r0
0000004c  00000000      andeq   r0, r0, r0
                    STEV1:
00000050  00000010      andeq   r0, r0, r0, LSL r0
                    STEV2:
00000054  00000040      andeq   r0, r0, r0, ASR #32
                    REZ:
00000058  00000050      andeq   r0, r0, r0, ASR r0
0000005c  00000000      andeq   r0, r0, r0
                    _end:
00000060  aaaaaaaa      bge     0xfeaaab10
00000064  aaaaaaaa      bge     0xfeaaab14
00000068  aaaaaaaa      bge     0xfeaaab18
0000006c  aaaaaaaa      bge     0xfeaaab1c
00000070  aaaaaaaa      bge     0xfeaaab20
00000074  aaaaaaaa      bge     0xfeaaab24
00000078  aaaaaaaa      bge     0xfeaaab28
0000007c  aaaaaaaa      bge     0xfeaaab2c
```

Go to address, label, or register:

```
Address   Memory contents and ASCII
00000000  00000000  00000000  00000000  00000000
00000010  00000000  00000000  00000000  00000000
00000020  e59f0018  e5901000  e59f0014  e5902000
00000030  e0813002  e59f000c  e5803000  eafffffe
00000040  00000050  00000054  00000058  00000000
00000050  00000010  00000040  00000050  00000000
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
00000170  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa
00000180  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa
00000190  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa
000001a0  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa
000001b0  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa
000001c0  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa
000001d0  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa
000001e0  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa
000001f0  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa
00000200  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa
00000210  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa
00000220  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa
00000230  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa
00000240  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa
00000250  aaaaaaaa  aaaaaaaa  aaaaaaaa  aaaaaaaa
```

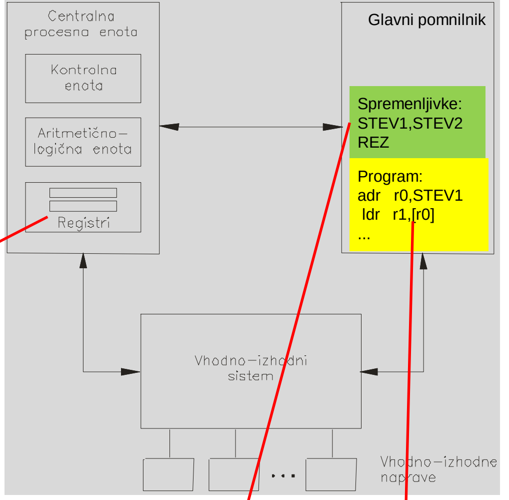
```
Link: arm-altera-eabi-ld  --script build_arm.ld -e _start -u _start -o work/asmETdoWw.s.elf work/asmETdoWw.s.o
Compile succeeded.

Compiling...
Code and data loaded from ELF executable into memory. Total size is 96 bytes.
Assemble: arm-altera-eabi-as -mfloat-abi=soft -march=armv7-a -mcpu=cortex-a9 -mfpu=neon-fp16 --gdwarf2 -o work/asmreEAFP.s.o work/asmreEAFP.s
Link: arm-altera-eabi-ld  --script build_arm.ld -e _start -u _start -o work/asmreEAFP.s.elf work/asmreEAFP.s.o
Compile succeeded.
```

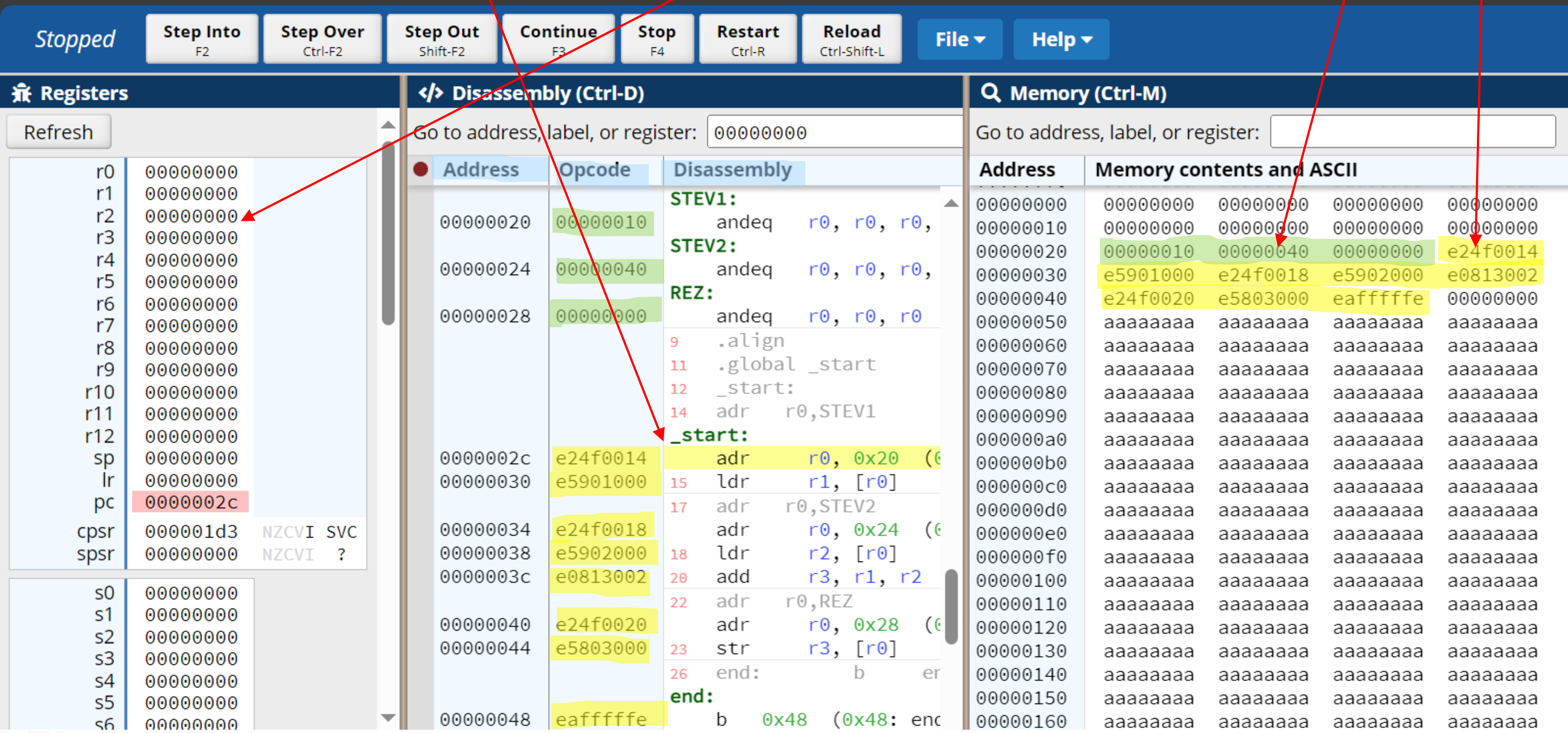
RA - Praktično delo: vsota dveh števil

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





Go to address, label, or register: 00000000

```
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
```

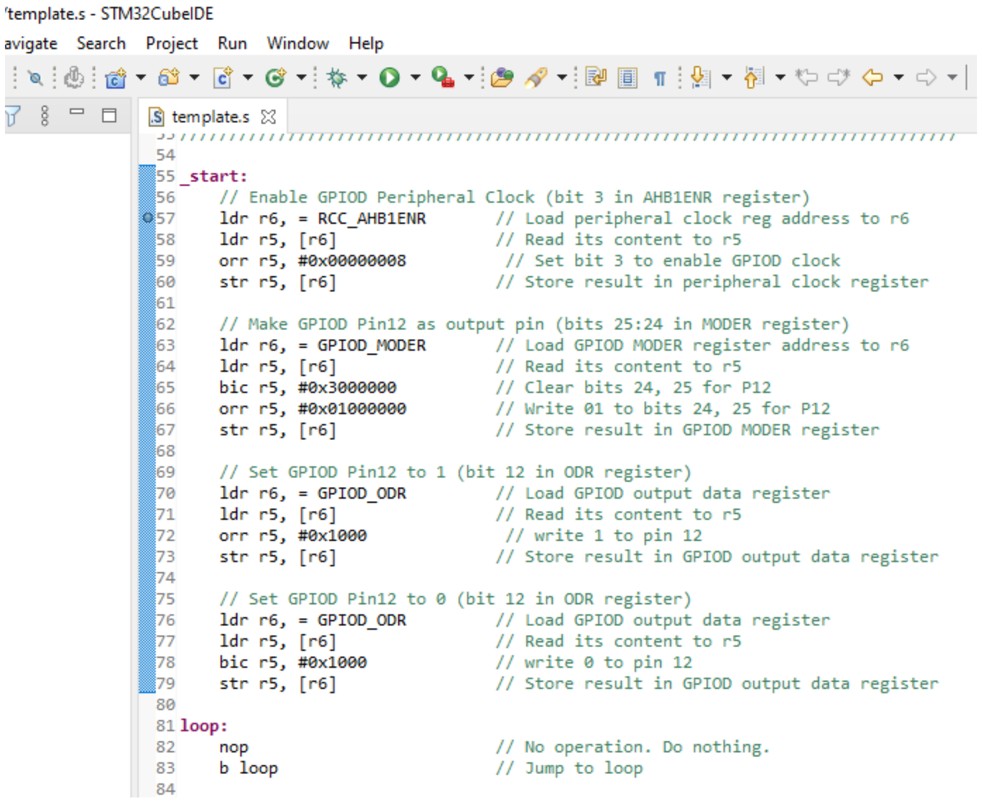
```
Address    Opcode     Disassembly
                      STEV1:
00000020   00000010     andeq    r0, r0, r0,
                      STEV2:
00000024   00000040     andeq    r0, r0, r0,
                      REZ:
00000028   00000000     andeq    r0, r0, r0
                        .align
            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

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

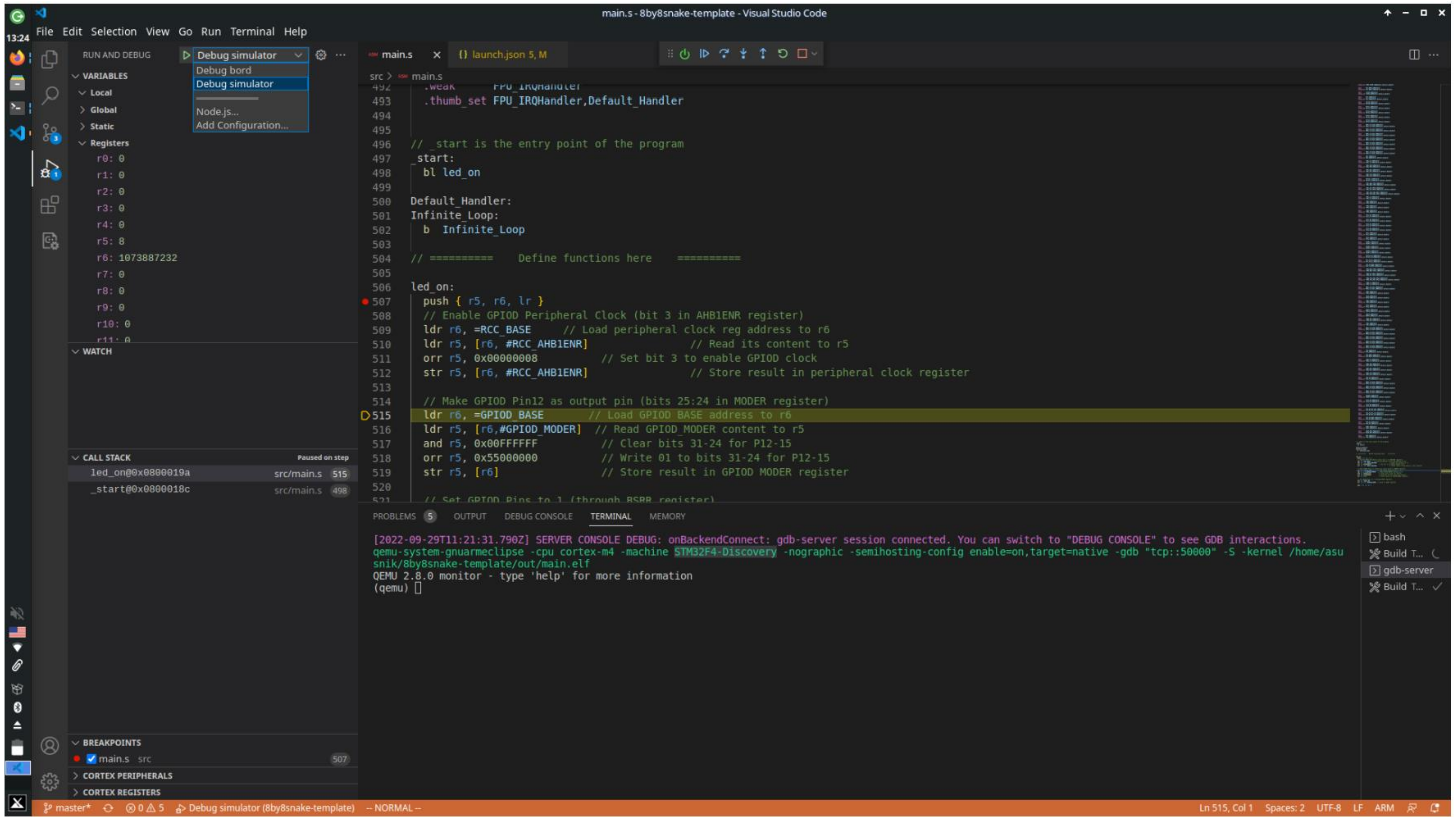
Delo s ploščami: STM32H7, STM32F4


'template.s - STM32CubeIDE
avigate Search Project Run Window Help
template.s

```
////////////////////////////////////////////////////////////////////////////////

_start:
    // 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_MODER       // Load GPIOD MODER register address to r6
    ldr r5, [r6]                // Read its content to r5
    bic r5, #0x3000000          // Clear bits 24, 25 for P12
    orr r5, #0x01000000         // Write 01 to bits 24, 25 for P12
    str r5, [r6]                // Store result in GPIOD MODER register

    // Set GPIOD Pin12 to 1 (bit 12 in ODR register)
    ldr r6, = GPIOD_ODR         // Load GPIOD output data register
    ldr r5, [r6]                // Read its content to r5
    orr r5, #0x1000              // write 1 to pin 12
    str r5, [r6]                // Store result in GPIOD output data register

    // Set GPIOD Pin12 to 0 (bit 12 in ODR register)
    ldr r6, = GPIOD_ODR         // Load GPIOD output data register
    ldr r5, [r6]                // Read its content to r5
    bic r5, #0x1000             // write 0 to pin 12
    str r5, [r6]                // Store result in GPIOD output data register

loop:
    nop                         // No operation. Do nothing.
    b loop                      // Jump to loop

```

Razvojno okolja 2. sklop: VSCode

Delo s ploščami: STM32H7, STM32F4
(ni uradno podprt)



```
r0: 0
r1: 0
r2: 0
r3: 0
r4: 0
r5: 8
r6: 1073887232
r7: 0
r8: 0
r9: 0
r10: 0
r11: 0
```

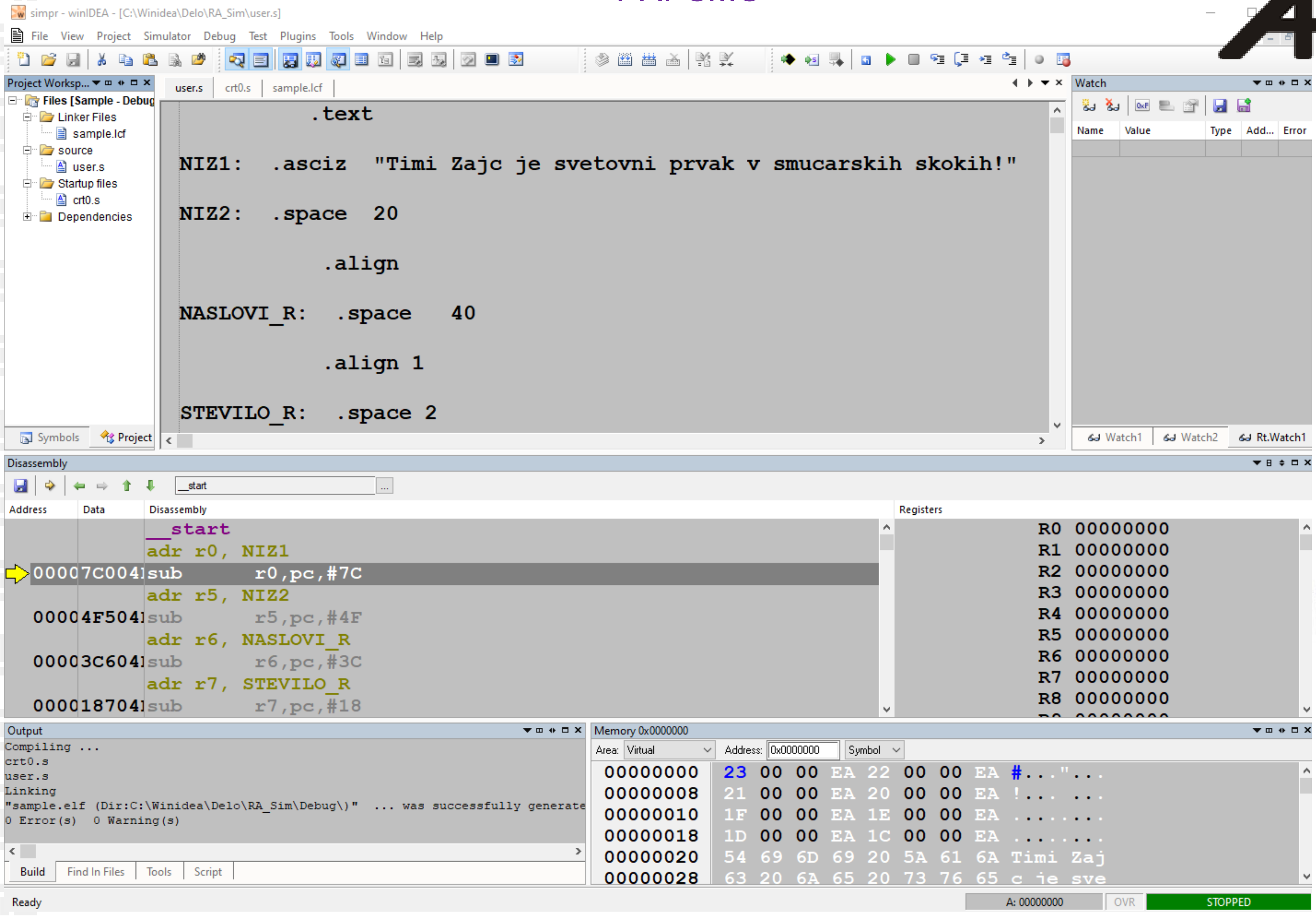
```
led_on@0x0800019a     src/main.s 515
_start@0x0800018c     src/main.s 498
```

```
  .weak      FPU_IRQHandler
  .thumb_set FPU_IRQHandler,Default_Handler


// _start is the entry point of the program
_start:
  bl led_on

Default_Handler:
Infinite_Loop:
  b  Infinite_Loop

// ==========    Define functions here    ==========

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

  // Set GPIOD Pins to 1 (through BSRR register)
```

```
[2022-09-29T11:21:31.790Z] SERVER CONSOLE DEBUG: onBackendConnect: gdb-server session connected. You can switch to "DEBUG CONSOLE" to see GDB interactions.
qemu-system-gnuarmeclipse -cpu cortex-m4 -machine STM32F4-Discovery -nographic -semihosting-config enable=on,target=native -gdb "tcp::50000" -S -kernel /home/asu
snik/8by8snake-template/out/main.elf
QEMU 2.8.0 monitor - type 'help' for more information
(qemu) 
```

Razvojno okolja 2. sklop: WinIDEA

FRI-SMS



```
            .text

NIZ1:   .asciz  "Timi Zajc je svetovni prvak v smucarskih skokih!"

NIZ2:   .space  20

             .align

NASLOVI_R:   .space   40

             .align 1

STEVILO_R:   .space 2
```

```
         __start
         adr r0, NIZ1
00007C004 sub        r0,pc,#7C
         adr r5, NIZ2
00004F504 sub        r5,pc,#4F
         adr r6, NASLOVI_R
00003C604 sub        r6,pc,#3C
         adr r7, STEVILO_R
00001870 4 sub        r7,pc,#18
```

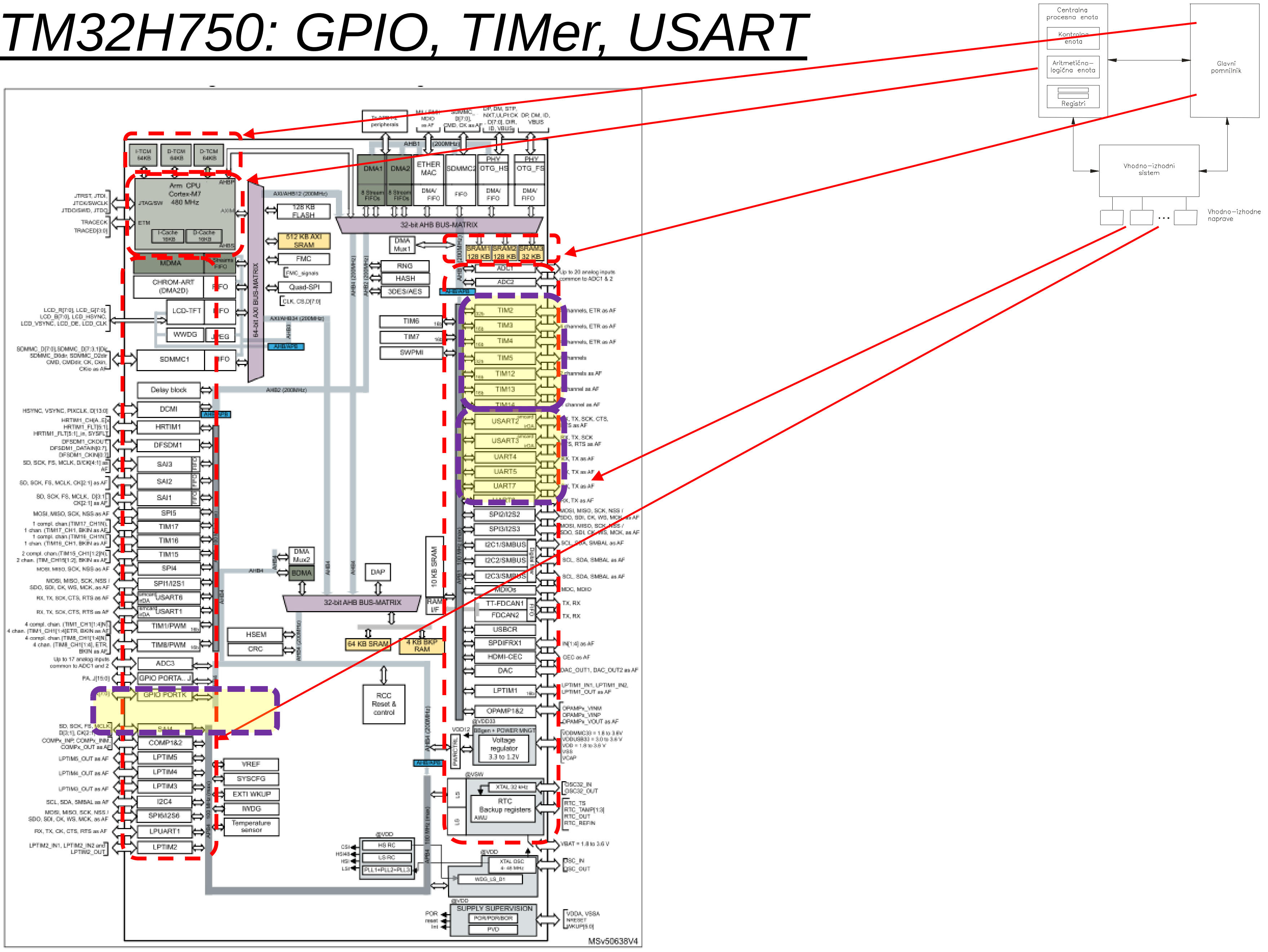
```
R0 00000000
R1 00000000
R2 00000000
R3 00000000
R4 00000000
R5 00000000
R6 00000000
R7 00000000
R8 00000000
```

```
Compiling ...
crt0.s
user.s
Linking
"sample.elf (Dir:C:\Winidea\Delo\RA_Sim\Debug\)"  ... was successfully generate
0 Error(s)  0 Warning(s)
```

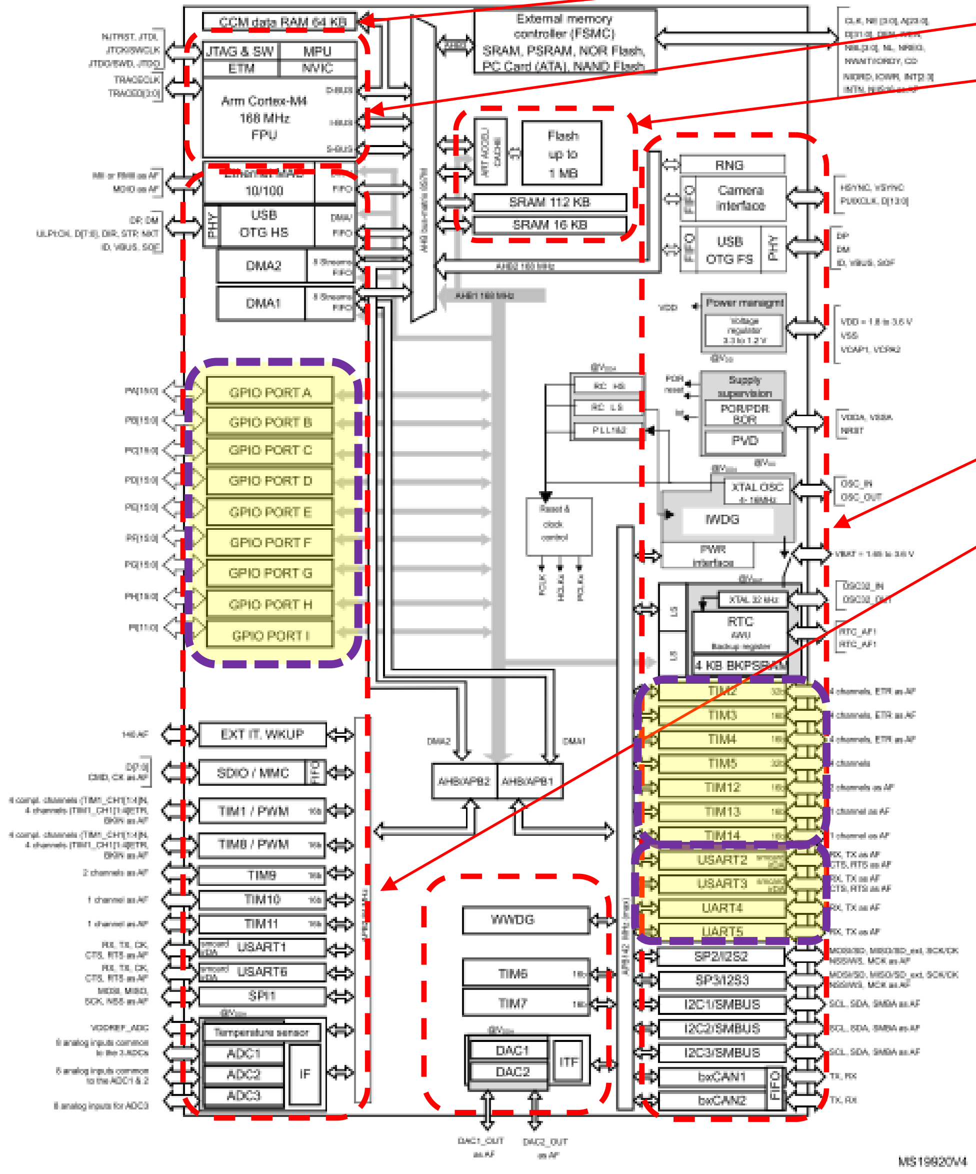
```
00000000  23 00 00 EA 22 00 00 EA #..."...
00000008  21 00 00 EA 20 00 00 EA !... ...
00000010  1F 00 00 EA 1E 00 00 EA ........
00000018  1D 00 00 EA 1C 00 00 EA ........
00000020  54 69 6D 69 20 5A 61 6A Timi Zaj
00000028  63 20 6A 65 20 73 76 65 c je sve
```

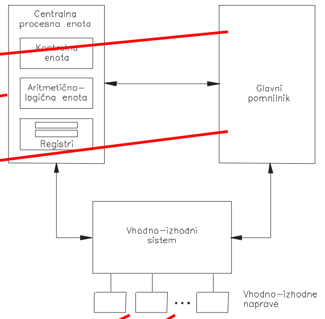
V 2. sklopu – delo s sistemi STM32H7, STM32F4 , FRI SMS

STM32H750: GPIO, TIMer, USART

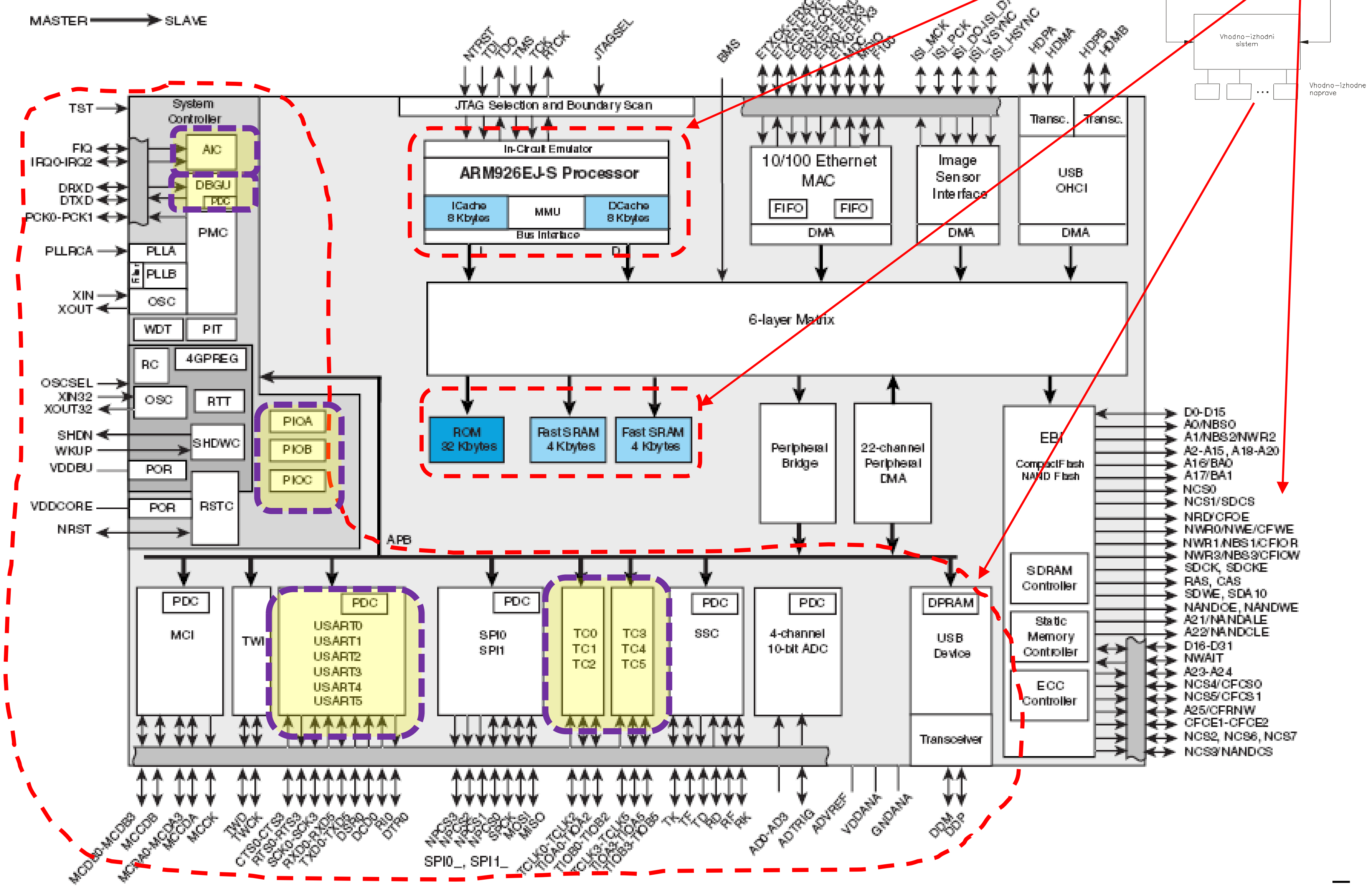


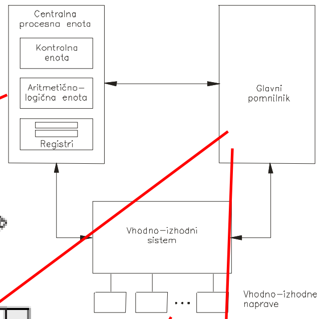
STM32F407VG: GPIO, TIMer, USART



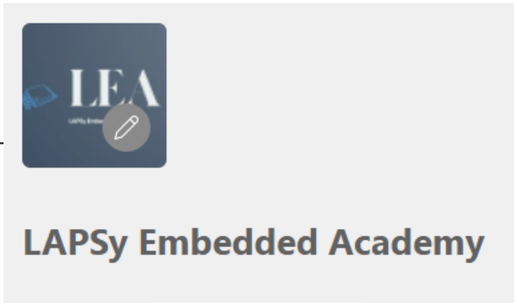


FRISMS: PIO, Časovnik, DBGU UART





Iščete dodatne izzive ?

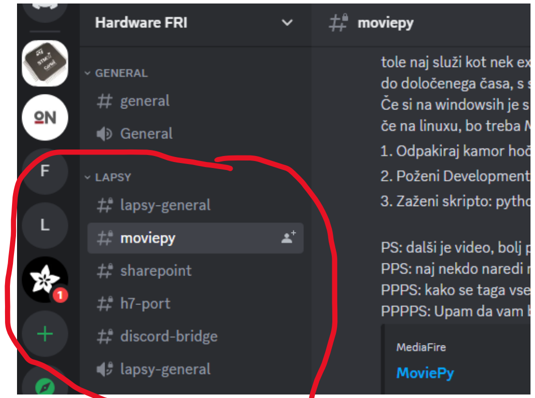


ty5qjm9

Dobrodošli:

- LEA - Lapsy Embedded Academy
  - Projekt spletnega izobraževalnega portala z vsebinami RA, OR in VIN (ter ostalimi)
  - Rok: 1.1.2024, vabljeni !

- LAPSy Embedded LAB
  - Skupina navdušencev nad vgrajenimi sistemi in nizkonivojskim programiranjem


Hardware FRI

GENERAL
- general
- General

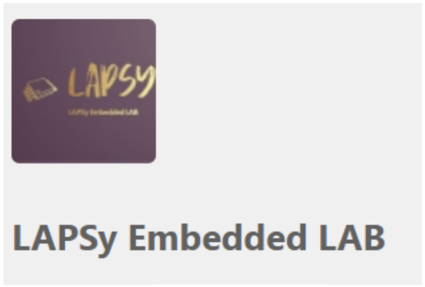
LAPSY
- lapsy-general
- moviepy
- sharepoint
- h7-port
- discord-bridge
- lapsy-general

moviepy

tole naj služi kot nek ex
do določenega časa, s
Če si na windowsih je s
če na linuxu, bo treba
1. Odpakiraj kamor hoč
2. Poženi Development
3. Zaženi skripto: pytho

PS: další je video, bolj
PPS: naj nekdo naredi
PPPS: kako se taga vse
PPPPS: Upam da vam

MediaFire
MoviePy



jxhtzj8