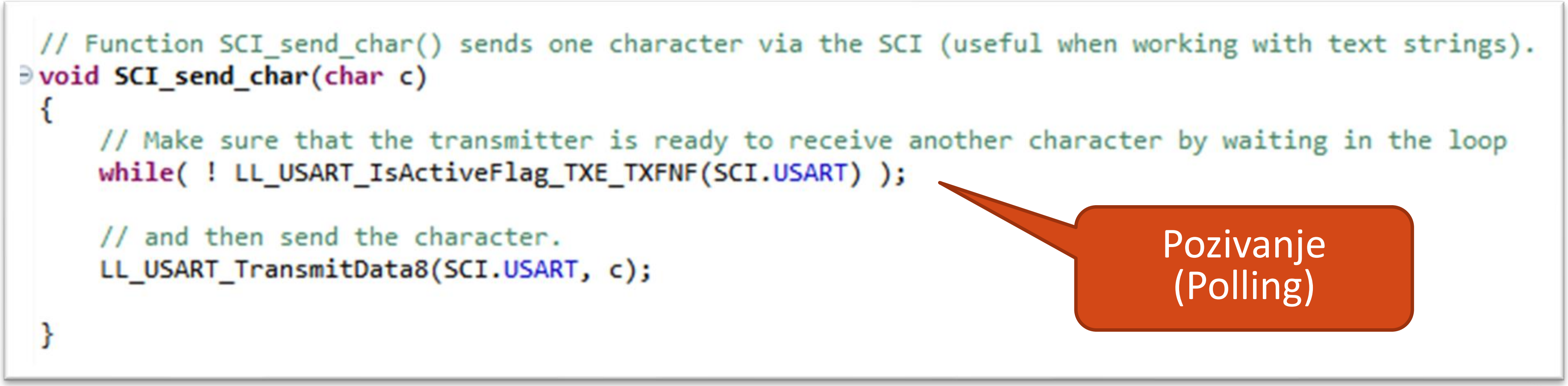
Osnove mikroprocesorske elektronike

Marko Jankovec

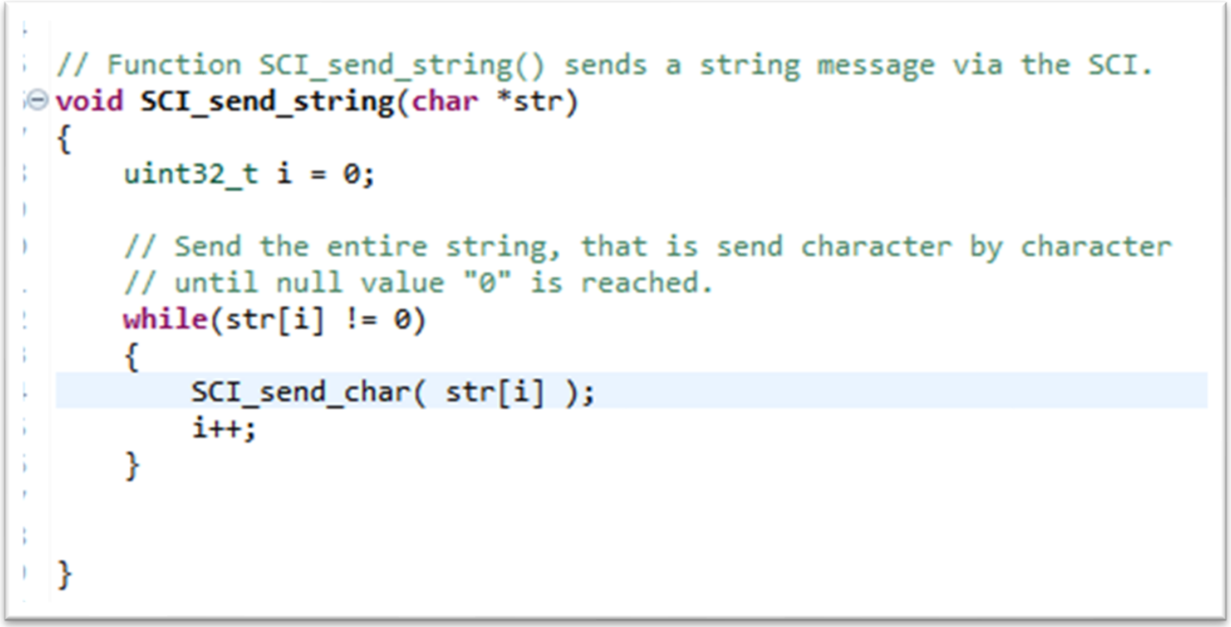
Primeri prekinitev pri STM32

Zakaj prekinitve?

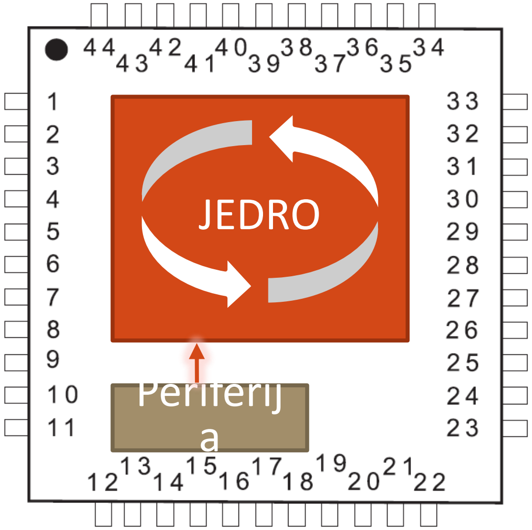


```
// Function SCI_send_char() sends one character via the SCI (useful when working with text strings).
void SCI_send_char(char c)
{
    // Make sure that the transmitter is ready to receive another character by waiting in the loop
    while( ! LL_USART_IsActiveFlag_TXE_TXFNF(SCI.USART) );

    // and then send the character.
    LL_USART_TransmitData8(SCI.USART, c);

}
```

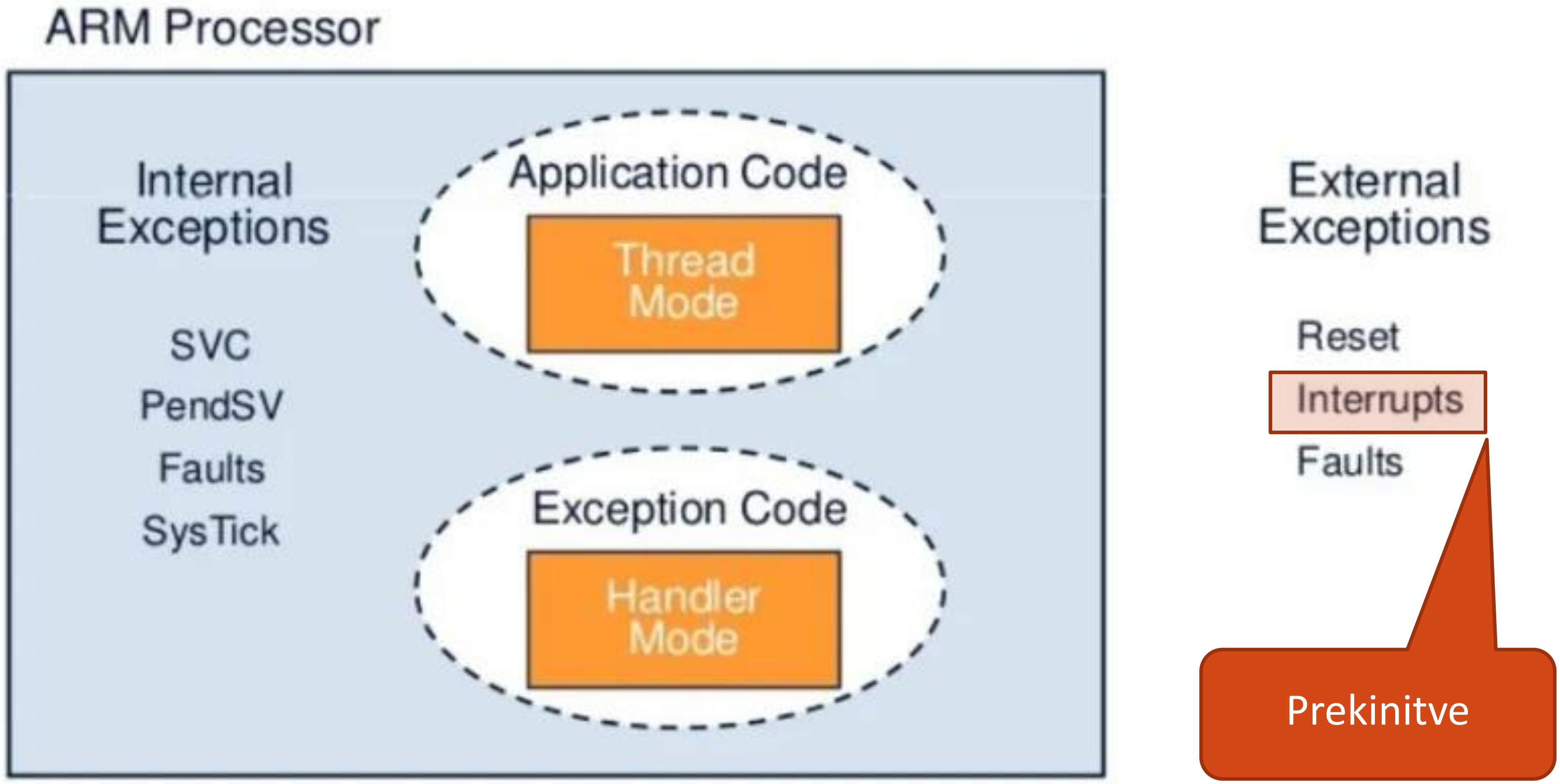
Pozivanje (Polling)



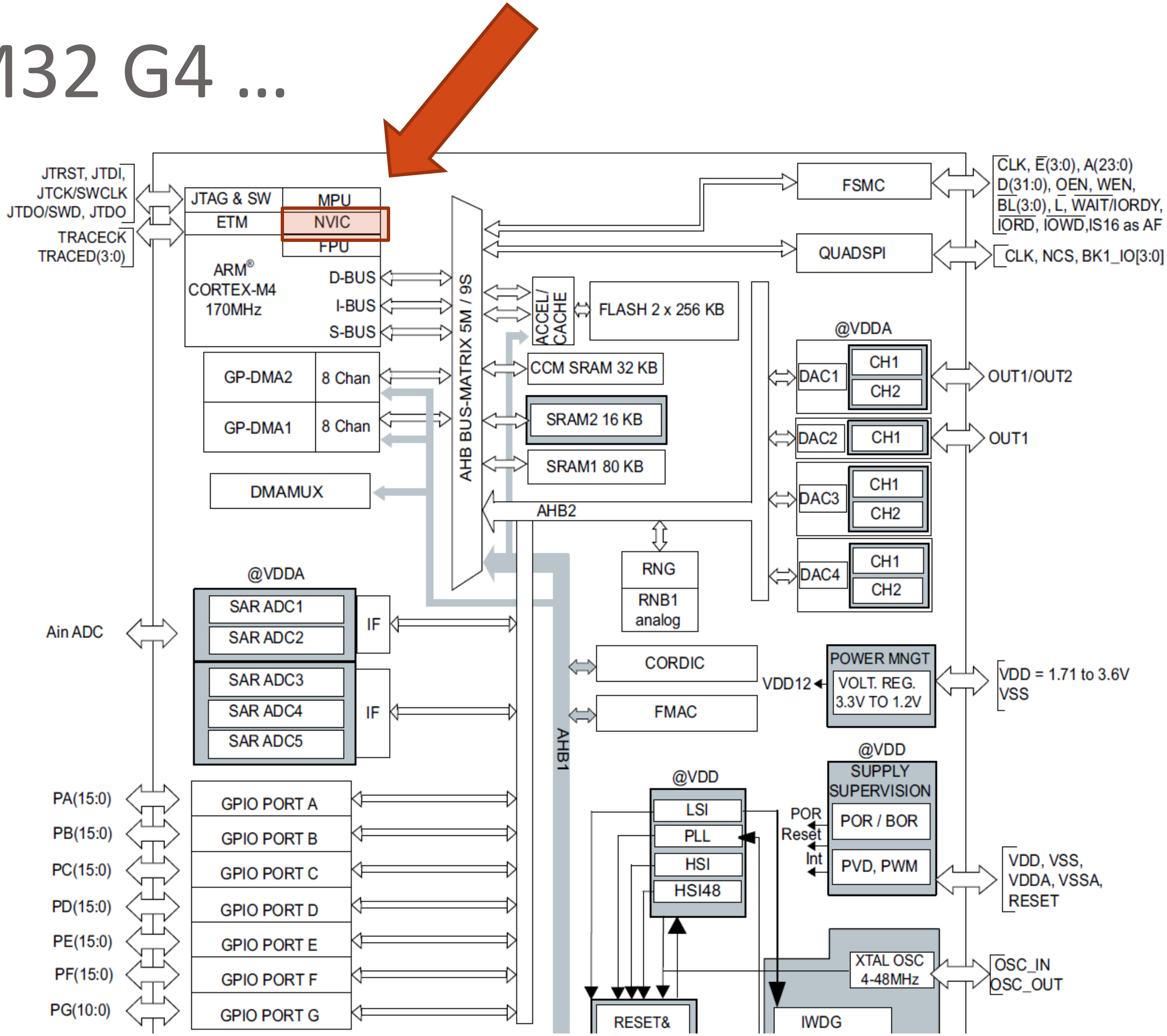
```
// Function SCI_send_string() sends a string message via the SCI.
void SCI_send_string(char *str)
{
    uint32_t i = 0;

    // Send the entire string, that is send character by character
    // until null value "0" is reached.
    while(str[i] != 0)
    {
        SCI_send_char( str[i] );
        i++;
    }


}
```



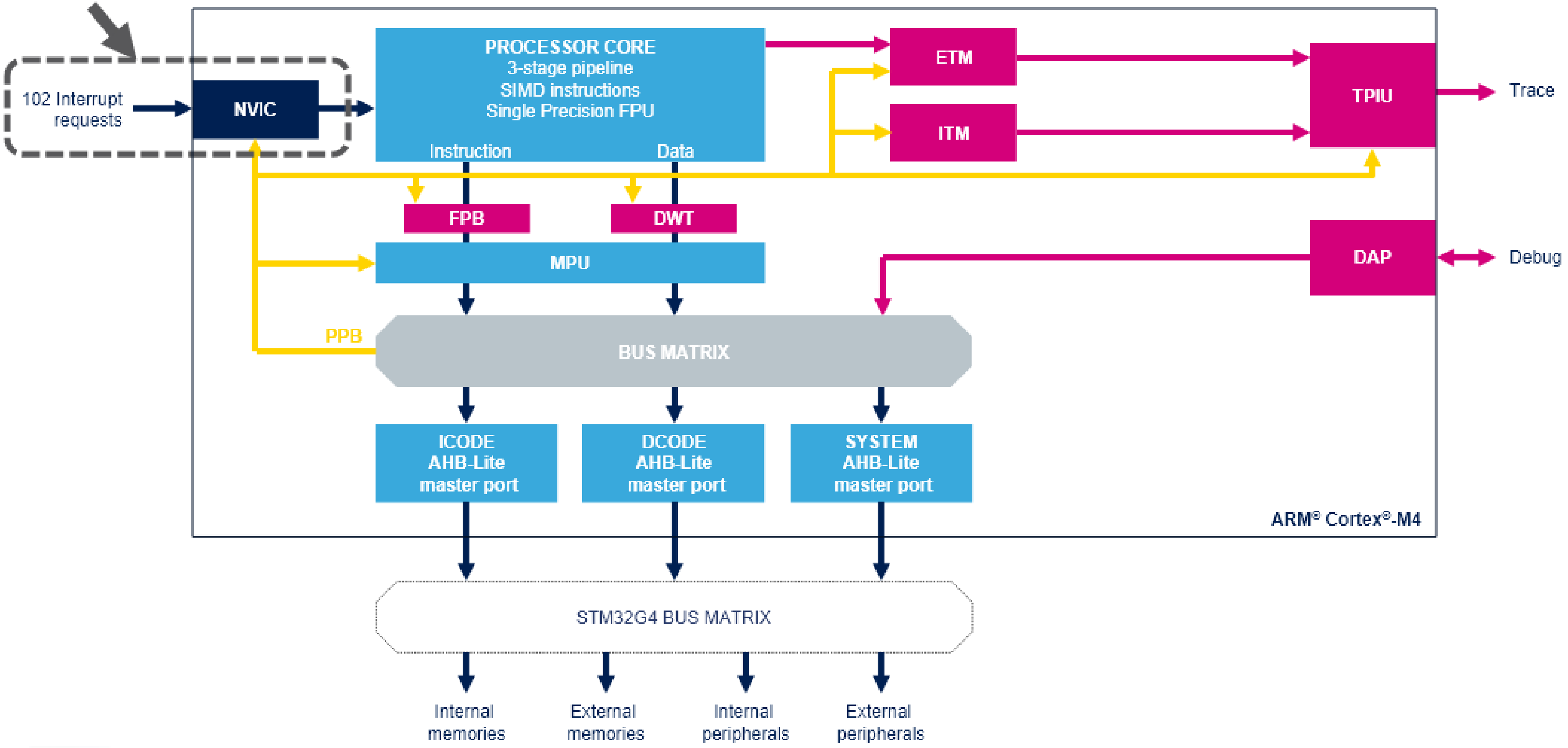
Izjeme in prekinitve pri STM32



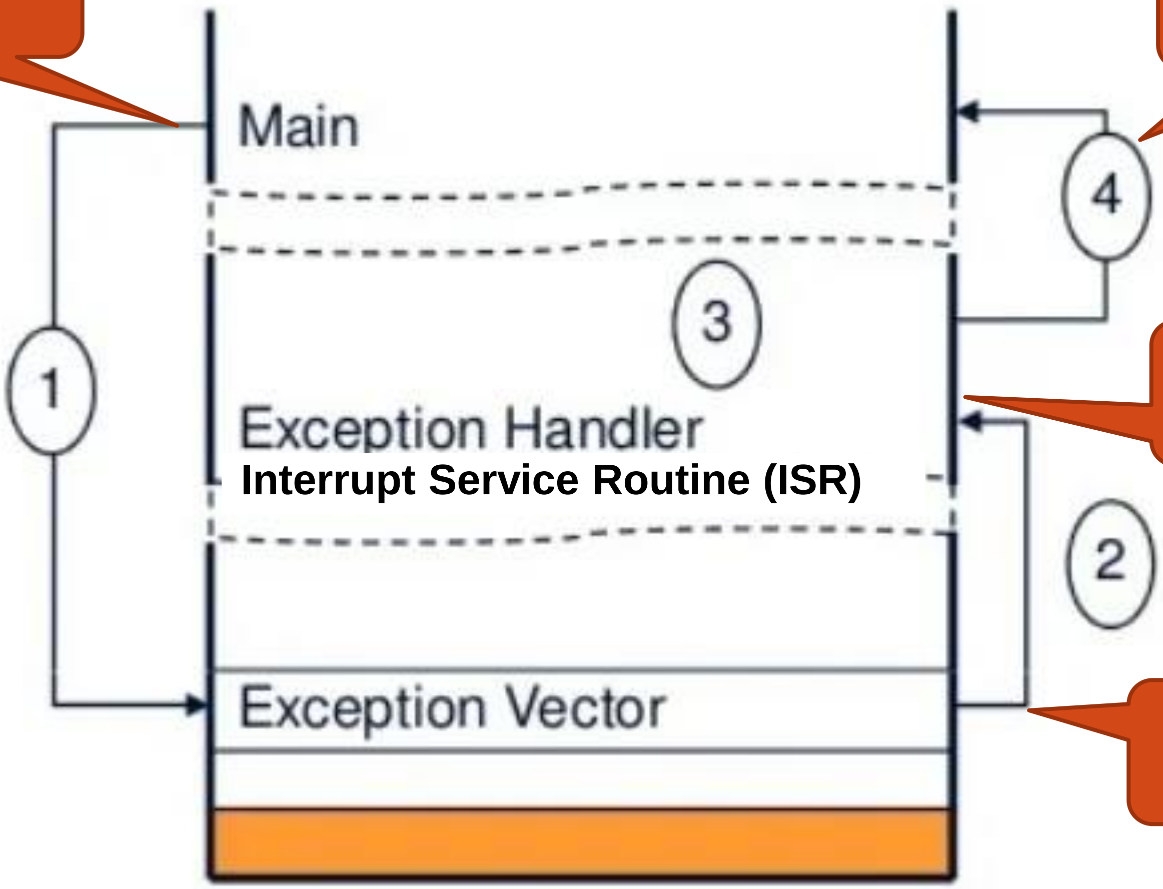
STM32 G4 …


JTRST, JTDI,
JTCK/SWCLK
JTDO/SWD, JTDO
TRACECK
TRACED(3:0)
JTAG & SW
MPU
ETM
NVIC
FPU
ARM®
CORTEX-M4
170MHz
D-BUS
I-BUS
S-BUS
GP-DMA2
8 Chan
GP-DMA1
8 Chan
DMAMUX
AHB BUS-MATRIX 5M / 9S
ACCEL/
CACHE
FLASH 2 x 256 KB
CCM SRAM 32 KB
SRAM2 16 KB
SRAM1 80 KB
AHB2
RNG
RNB1
analog
FSMC
QUADSPI
CLK, E(3:0), A(23:0)
D(31:0), OEN, WEN,
BL(3:0), L, WAIT/IORDY,
IORD, IOWD,IS16 as AF
CLK, NCS, BK1_IO[3:0]
@VDDA
DAC1
CH1
CH2
OUT1/OUT2
DAC2
CH1
OUT1
DAC3
CH1
CH2
DAC4
CH1
CH2
@VDDA
SAR ADC1
SAR ADC2
IF
SAR ADC3
SAR ADC4
SAR ADC5
IF
Ain ADC
CORDIC
FMAC
AHB1
VDD12
POWER MNGT
VOLT. REG.
3.3V TO 1.2V
VDD = 1.71 to 3.6V
VSS
@VDD
SUPPLY
SUPERVISION
POR / BOR
PVD, PWM
POR
Reset
Int
VDD, VSS,
VDDA, VSSA,
RESET
@VDD
LSI
PLL
HSI
HSI48
XTAL OSC
4-48MHz
OSC_IN
OSC_OUT
RESET&
IWDG
PA(15:0)
PB(15:0)
PC(15:0)
PD(15:0)
PE(15:0)
PF(15:0)
PG(10:0)
GPIO PORT A
GPIO PORT B
GPIO PORT C
GPIO PORT D
GPIO PORT E
GPIO PORT F
GPIO PORT G

NVIC – Nested Vector Interrupt Controller



Izvajanje prekinitev (Interrupt Service)



Zahteva za prekinitev

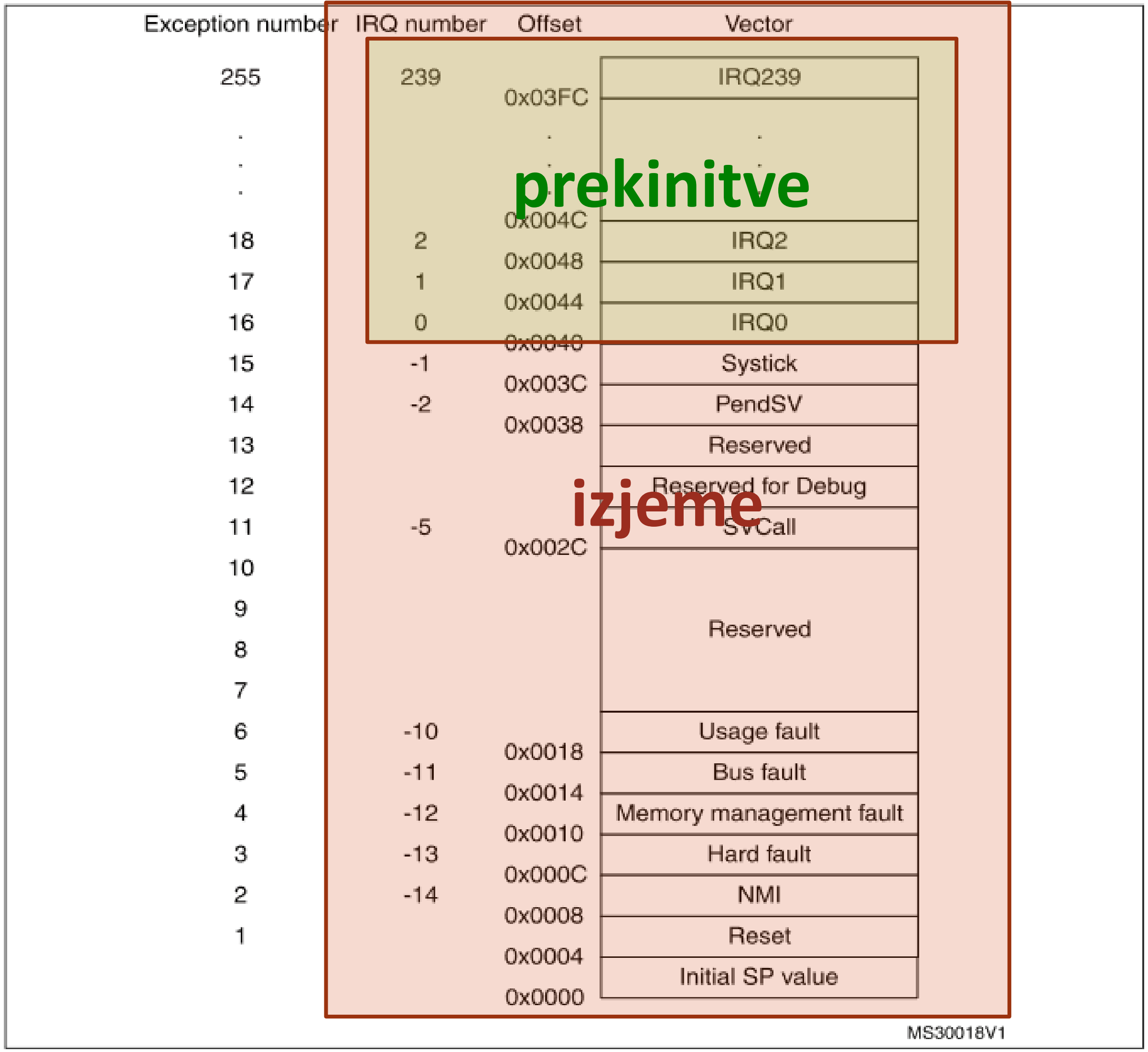
Skok na glavni program

Izvajanje ISR

Skok na ISR

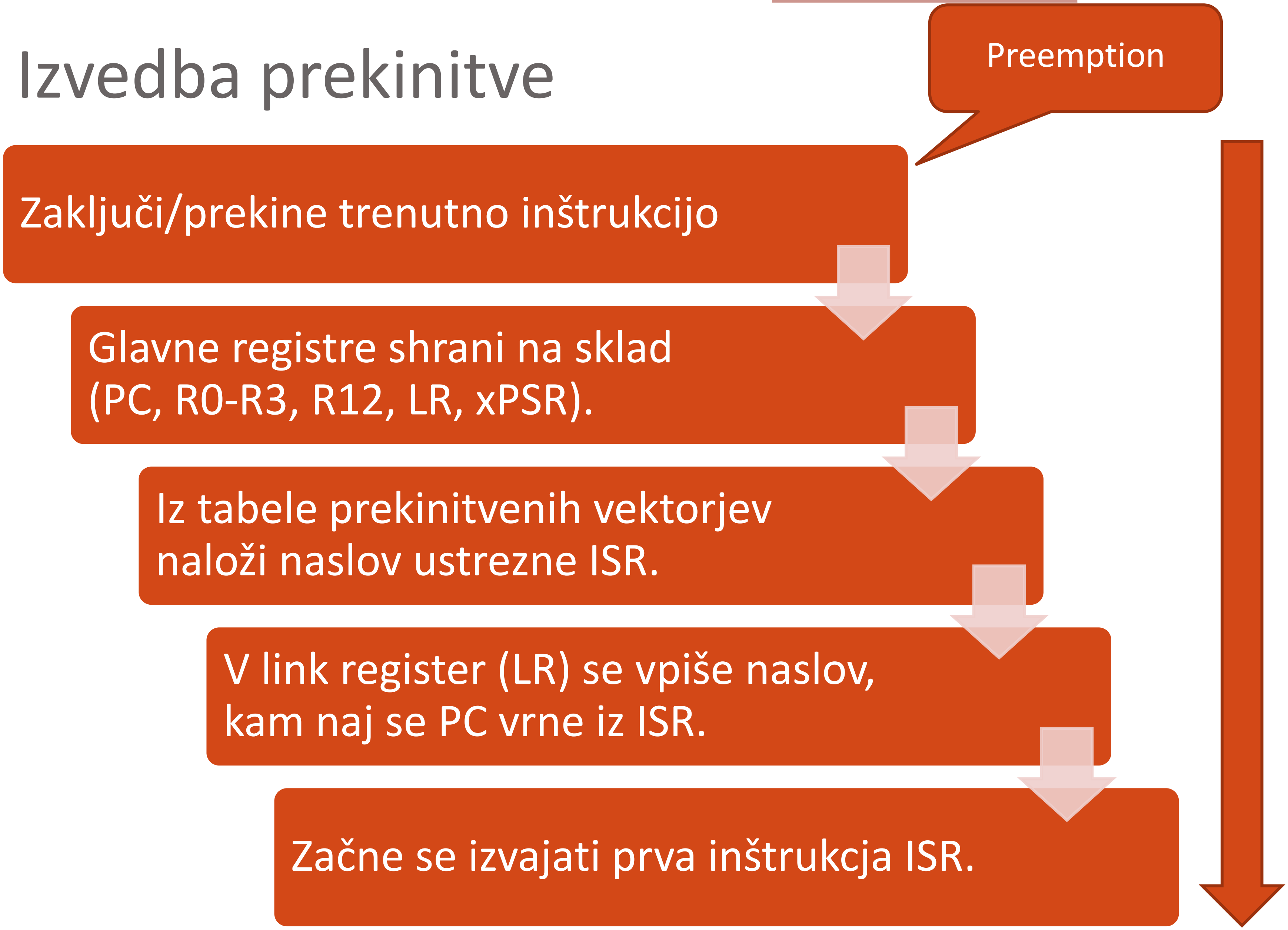
Tabela izjem/prekinitvenih vektorjev

Figure 11. Vector table


prekinitve
izjeme
MS30018V1

Exception number	IRQ number	Offset	Vector
255	239	0x03FC	IRQ239
. . .		. . 0x004C	. .
18	2	0x0048	IRQ2
17	1	0x0044	IRQ1
16	0	0x0040	IRQ0
15	-1	0x003C	Systick
14	-2	0x0038	PendSV
13			Reserved
12			Reserved for Debug
11	-5	0x002C	SVCall
10 9 8 7			Reserved
6	-10	0x0018	Usage fault
5	-11	0x0014	Bus fault
4	-12	0x0010	Memory management fault
3	-13	0x000C	Hard fault
2	-14	0x0008	NMI
1		0x0004	Reset
		0x0000	Initial SP value

Izvedba prekinitve



Preemption

Zaključi/prekine trenutno inštrukcijo

Glavne registre shrani na sklad
(PC, R0-R3, R12, LR, xPSR).

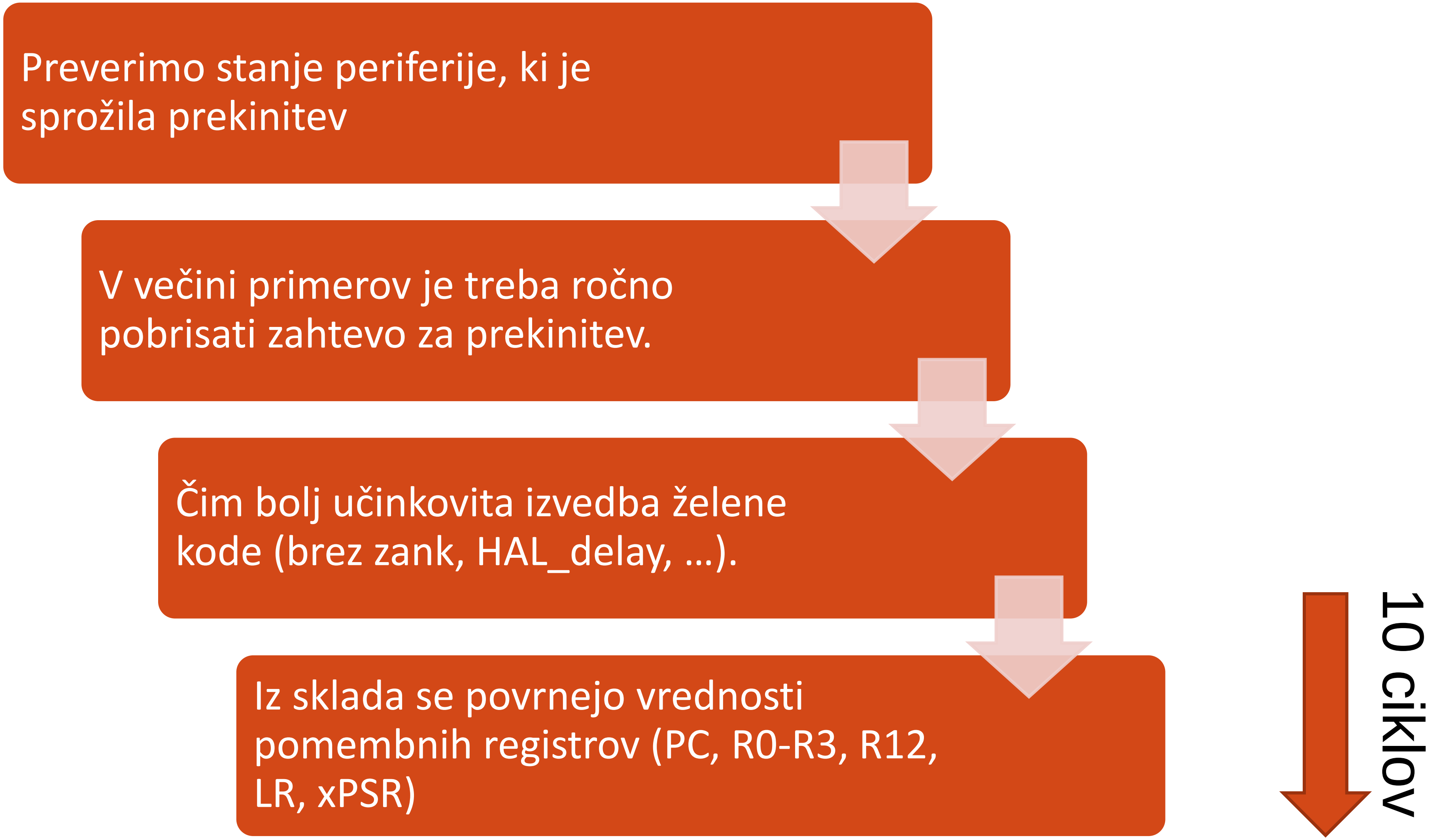
Iz tabele prekinitvenih vektorjev
naloži naslov ustrezne ISR.

V link register (LR) se vpiše naslov,
kam naj se PC vrne iz ISR.

Začne se izvajati prva inštrukcja ISR.

12 ciklov

Prekinitvena rutina (ISR)



Preverimo stanje periferije, ki je sprožila prekinitev

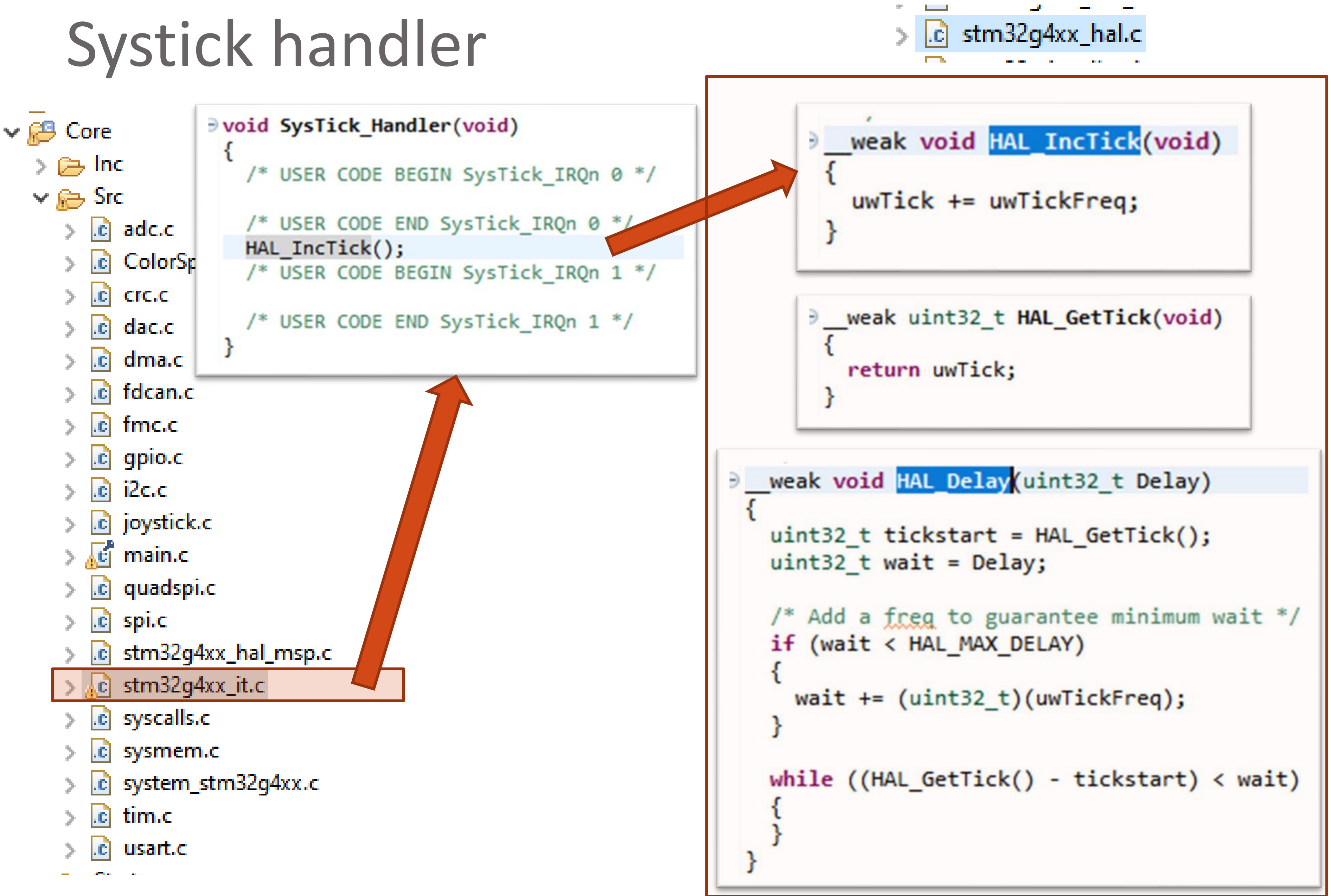
V večini primerov je treba ročno pobrisati zahtevo za prekinitev.

Čim bolj učinkovita izvedba želene kode (brez zank, HAL_delay, …).

Iz sklada se povrnejo vrednosti pomembnih registrov (PC, R0-R3, R12, LR, xPSR)

10 ciklov

Systick handler



- Core
  - Inc
  - Src
    - adc.c
    - ColorSp
    - crc.c
    - dac.c
    - dma.c
    - fdcan.c
    - fmc.c
    - gpio.c
    - i2c.c
    - joystick.c
    - main.c
    - quadspi.c
    - spi.c
    - stm32g4xx_hal_msp.c
    - stm32g4xx_it.c
    - syscalls.c
    - sysmem.c
    - system_stm32g4xx.c
    - tim.c
    - usart.c

```c
void SysTick_Handler(void)
{
  /* USER CODE BEGIN SysTick_IRQn 0 */

  /* USER CODE END SysTick_IRQn 0 */
  HAL_IncTick();
  /* USER CODE BEGIN SysTick_IRQn 1 */

  /* USER CODE END SysTick_IRQn 1 */
}
```

stm32g4xx_hal.c

```c
__weak void HAL_IncTick(void)
{
  uwTick += uwTickFreq;
}
```

```c
__weak uint32_t HAL_GetTick(void)
{
  return uwTick;
}
```

```c
__weak void HAL_Delay(uint32_t Delay)
{
  uint32_t tickstart = HAL_GetTick();
  uint32_t wait = Delay;

  /* Add a freq to guarantee minimum wait */
  if (wait < HAL_MAX_DELAY)
  {
    wait += (uint32_t)(uwTickFreq);
  }

  while ((HAL_GetTick() - tickstart) < wait)
  {
  }
}
```

Stanja prekinitev

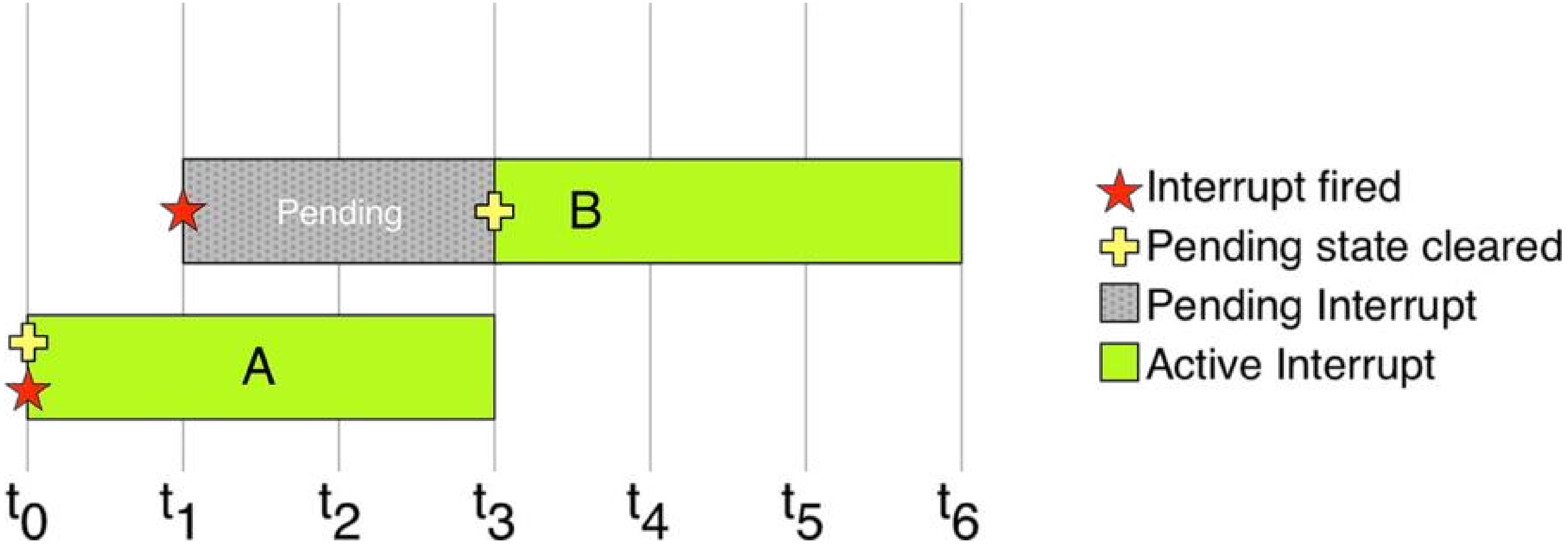
Interrupt mask registri

- Prekinitev je lahko
  - Onemogočena (Disabled)
  - Na čakanju da se izvede (Pending, waiting to be served)
  - Aktivna (active, being served)
  - Neaktivna (inactive, on hold)

- Če je prekinitev onemogočena
  - Še vedno gre na čakanje (pending)
  - Ko jo omogočimo, postane aktivna

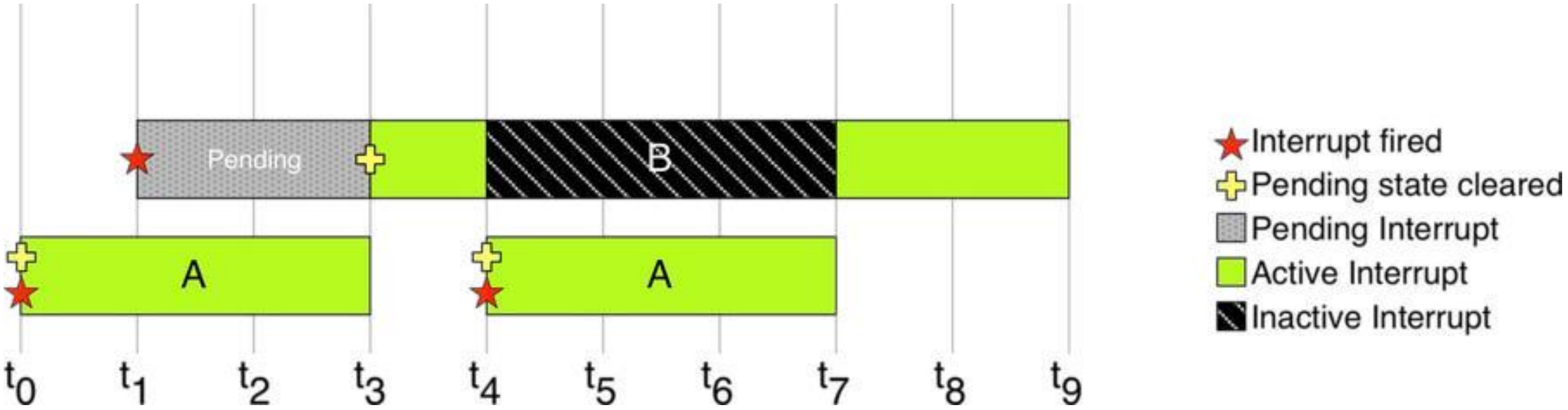
Čakajoča prekinitev

Interrupt source	Priority level
IRQ_A	0
IRQ_B	1

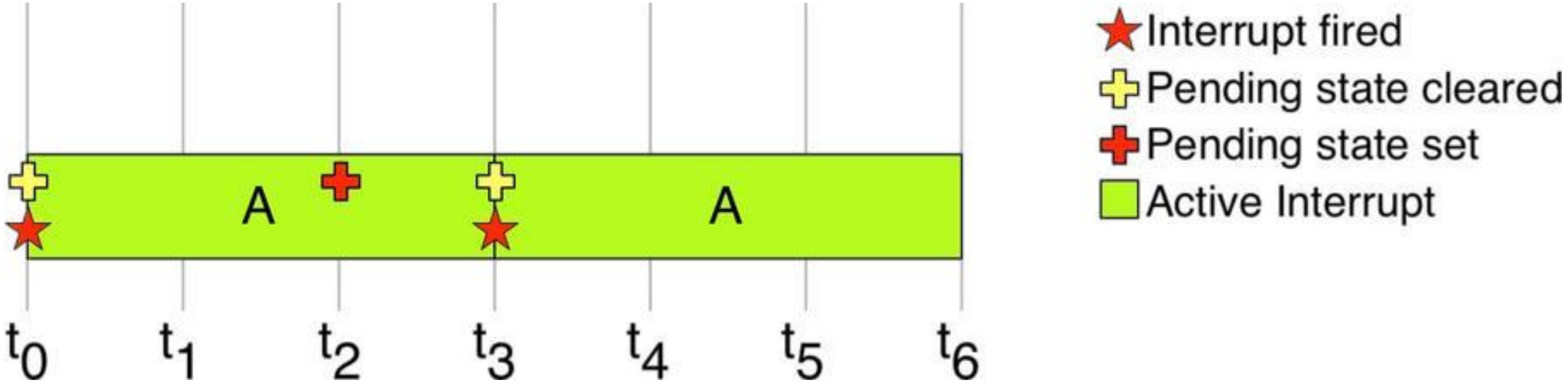


Neaktivna prekinitev

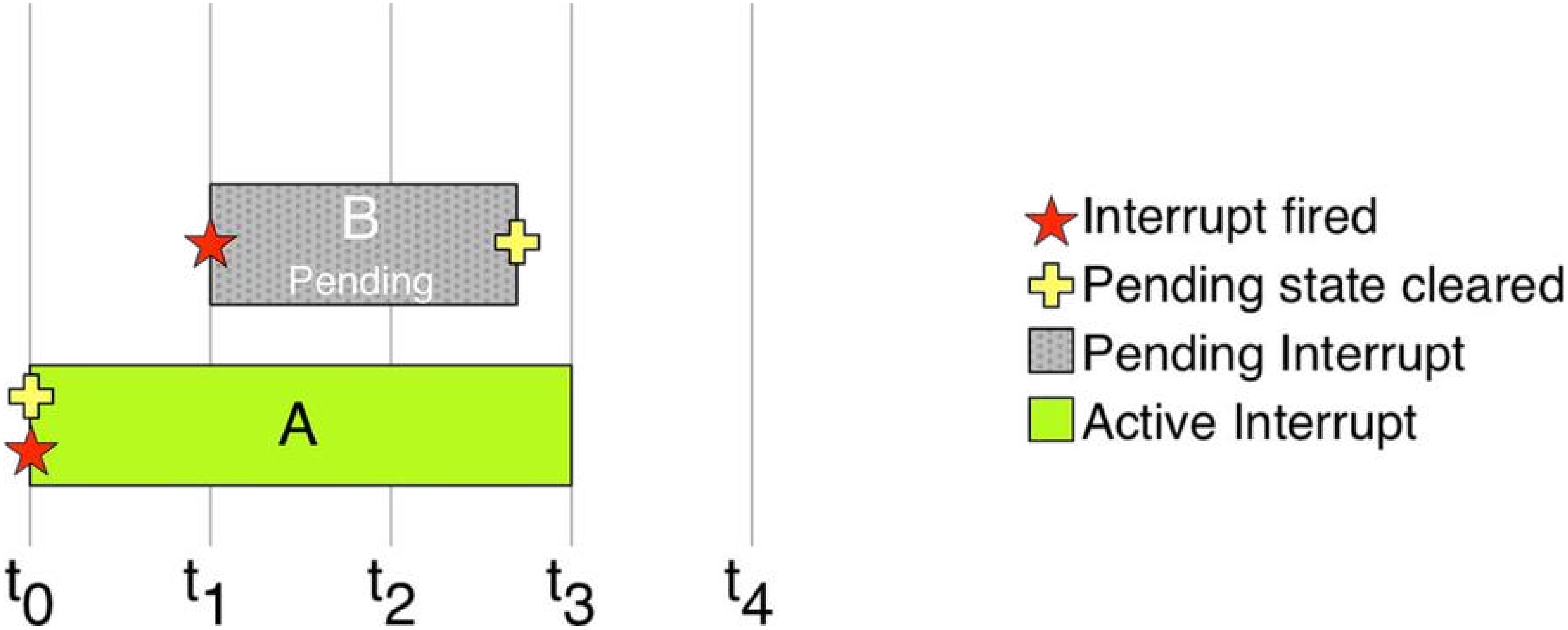
Interrupt source	Priority level
IRQ_A	0
IRQ_B	1



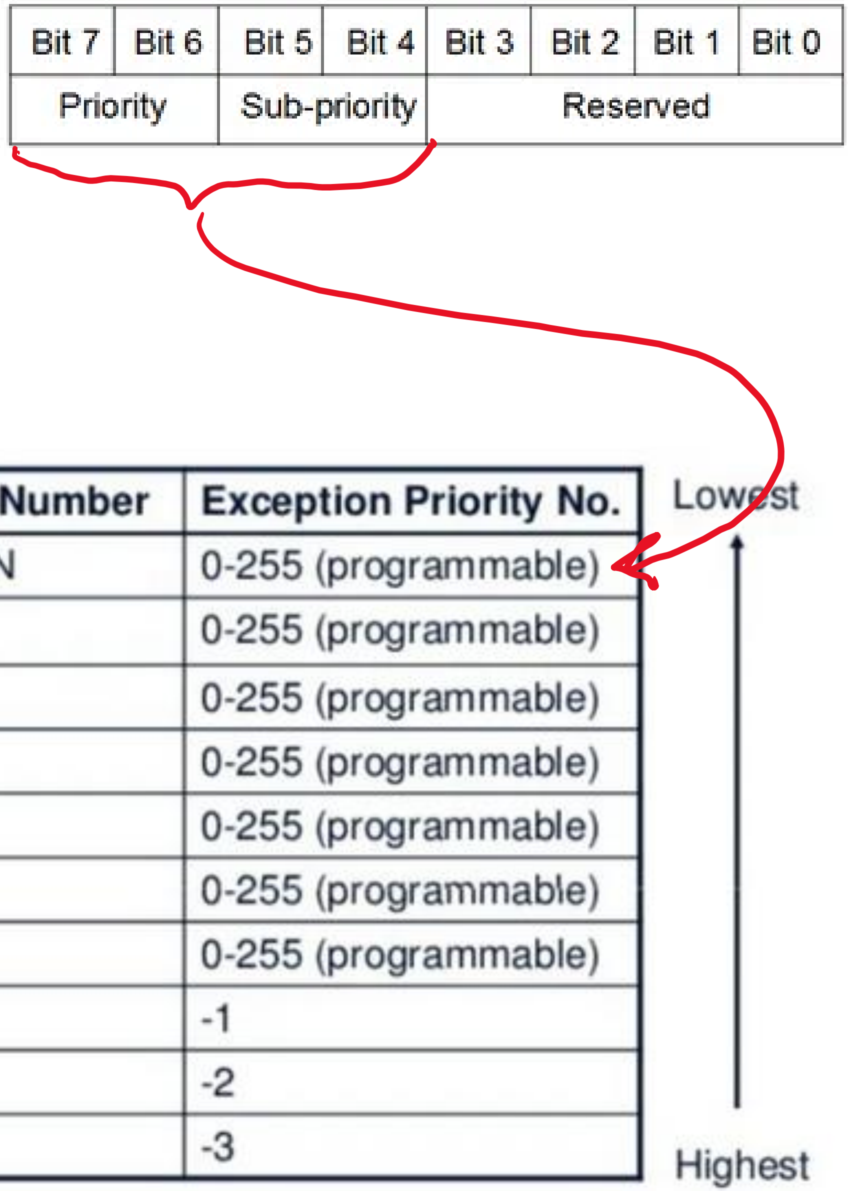
Ponovna zahteva po prekinitvi



Preklic prekinitve



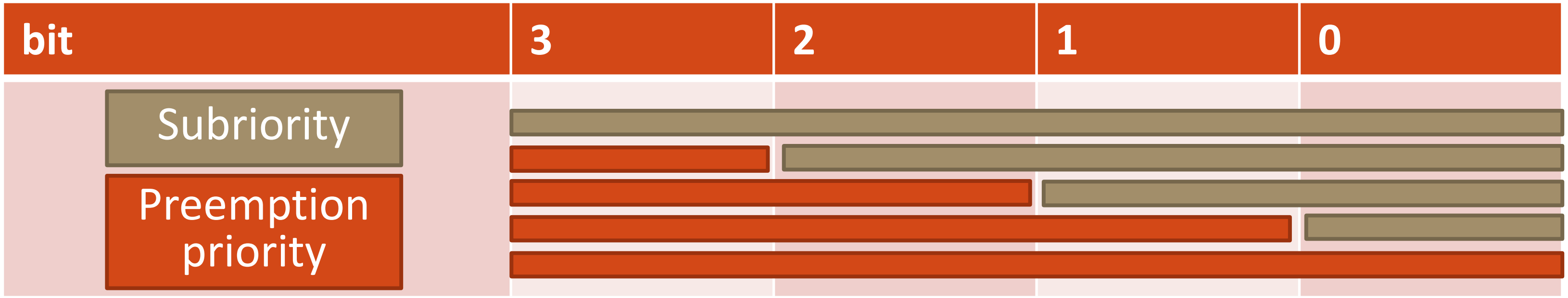
Prioritete prekinitev

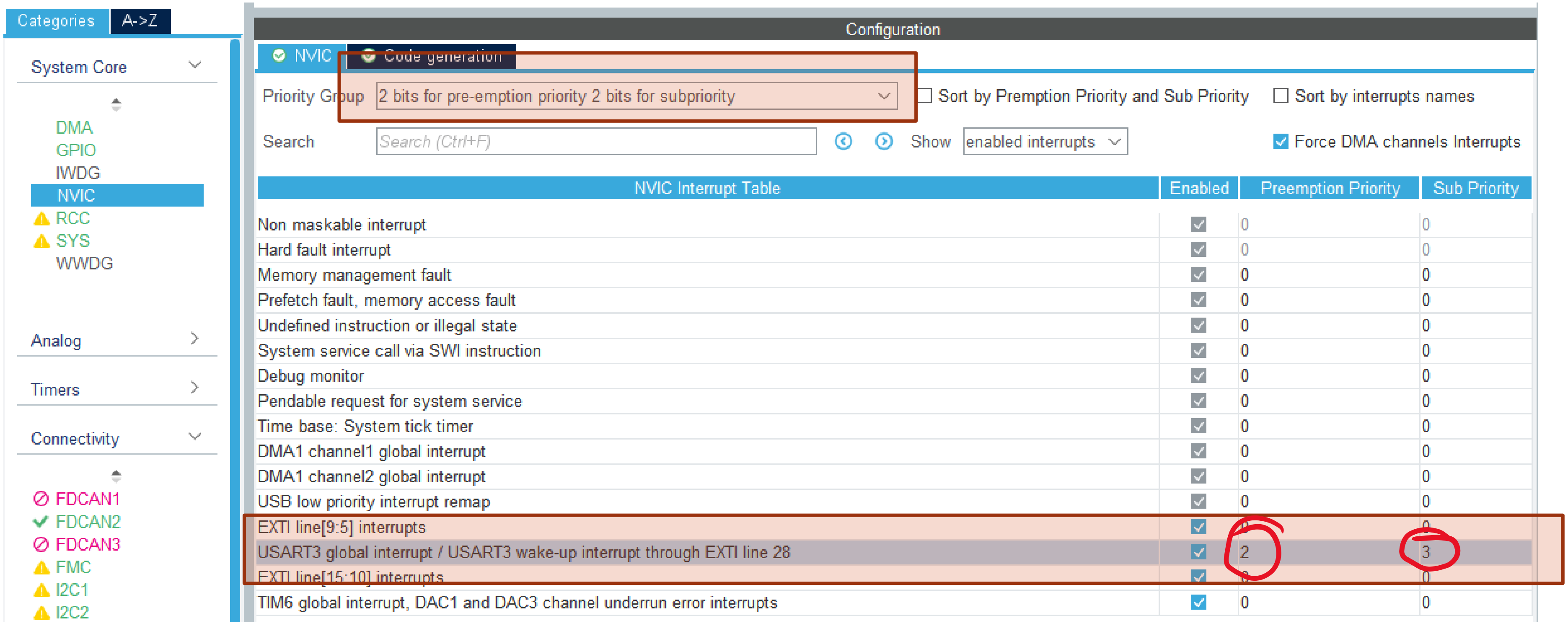


Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Priority		Sub-priority		Reserved			

Name	Exception Number	Exception Priority No.
Interrupts #0 - #495 (N interrupts)	16 to 16 + N	0-255 (programmable)
SysTick	15	0-255 (programmable)
PendSV	14	0-255 (programmable)
SVCall	11	0-255 (programmable)
Usage Fault	6	0-255 (programmable)
Bus Fault	5	0-255 (programmable)
Memory Management Fault	4	0-255 (programmable)
Hard Fault	3	-1
Non Maskable Interrupt (NMI)	2	-2
Reset	1	-3

Prioritetne skupine





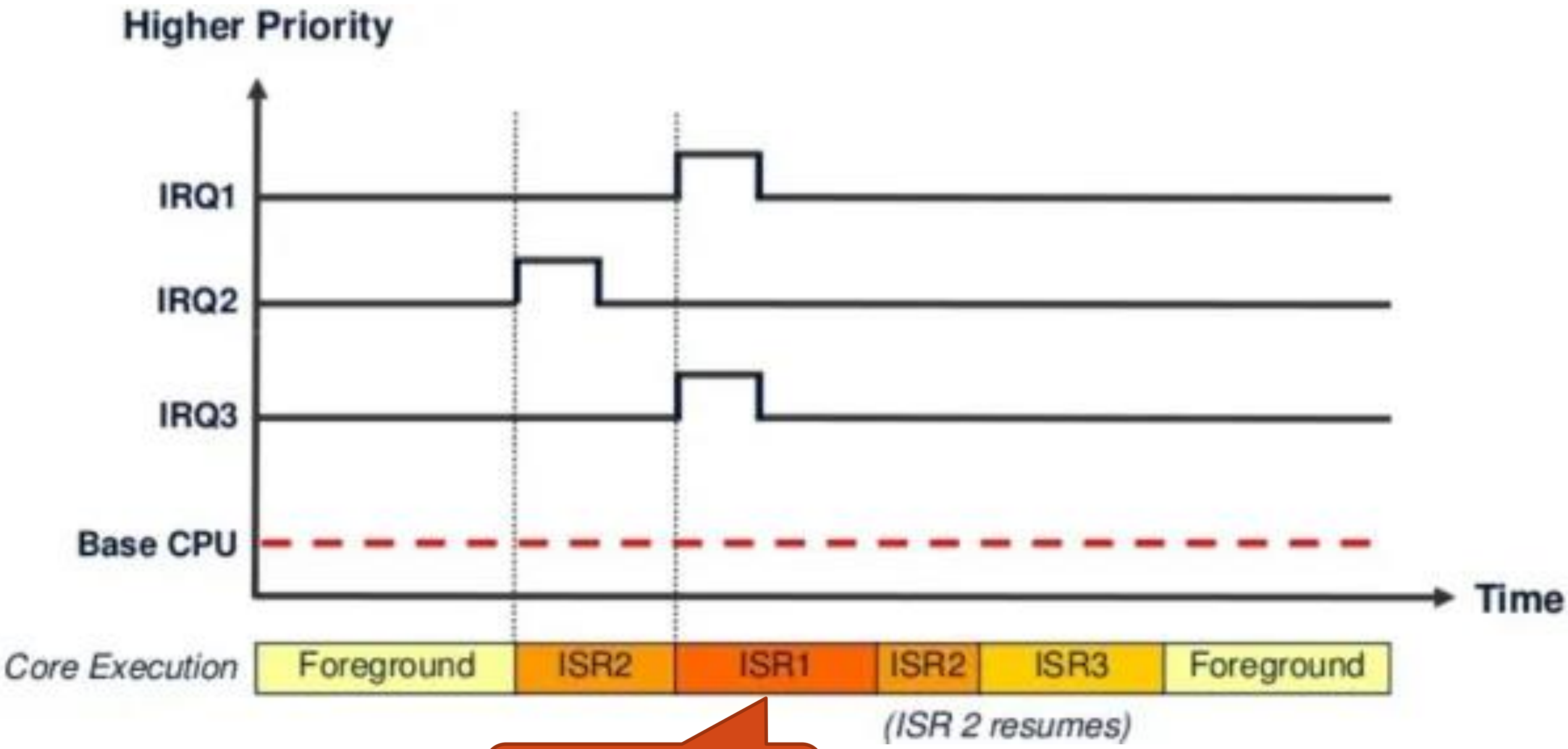
Priority Group: 2 bits for pre-emption priority 2 bits for subpriority

□ Sort by Premption Priority and Sub Priority   □ Sort by interrupts names

Search: Search (Ctrl+F)   Show: enabled interrupts   ☑ Force DMA channels Interrupts

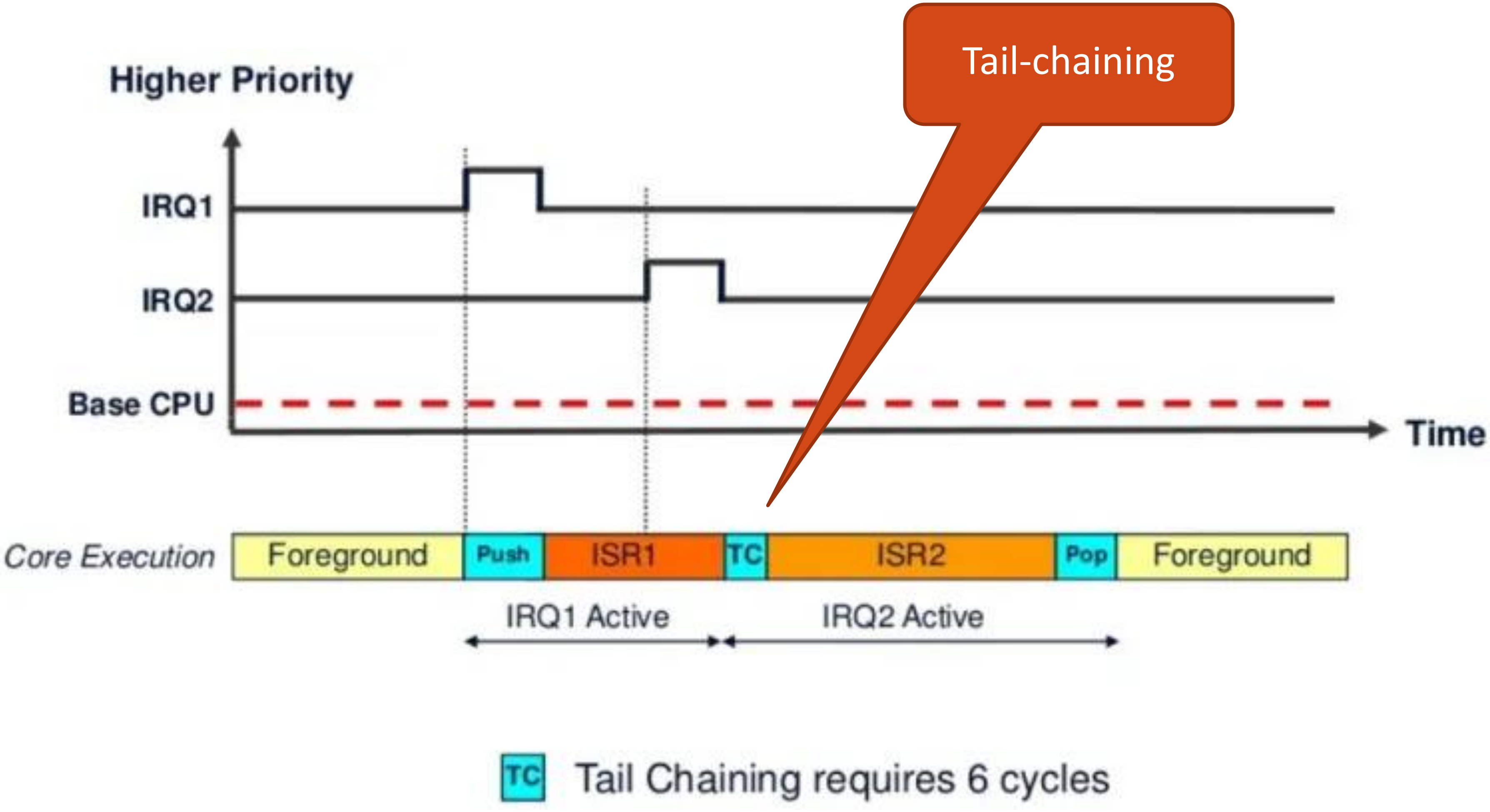
NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	☑	0	0
Hard fault interrupt	☑	0	0
Memory management fault	☑	0	0
Prefetch fault, memory access fault	☑	0	0
Undefined instruction or illegal state	☑	0	0
System service call via SWI instruction	☑	0	0
Debug monitor	☑	0	0
Pendable request for system service	☑	0	0
Time base: System tick timer	☑	0	0
DMA1 channel1 global interrupt	☑	0	0
DMA1 channel2 global interrupt	☑	0	0
USB low priority interrupt remap	☑	0	0
EXTI line[9:5] interrupts	☑	0	0
USART3 global interrupt / USART3 wake-up interrupt through EXTI line 28	☑	2	3
EXTI line[15:10] interrupts	☑	0	0
TIM6 global interrupt, DAC1 and DAC3 channel underrun error interrupts	☑	0	0

Gnezdenje prekinitev (Nesting)

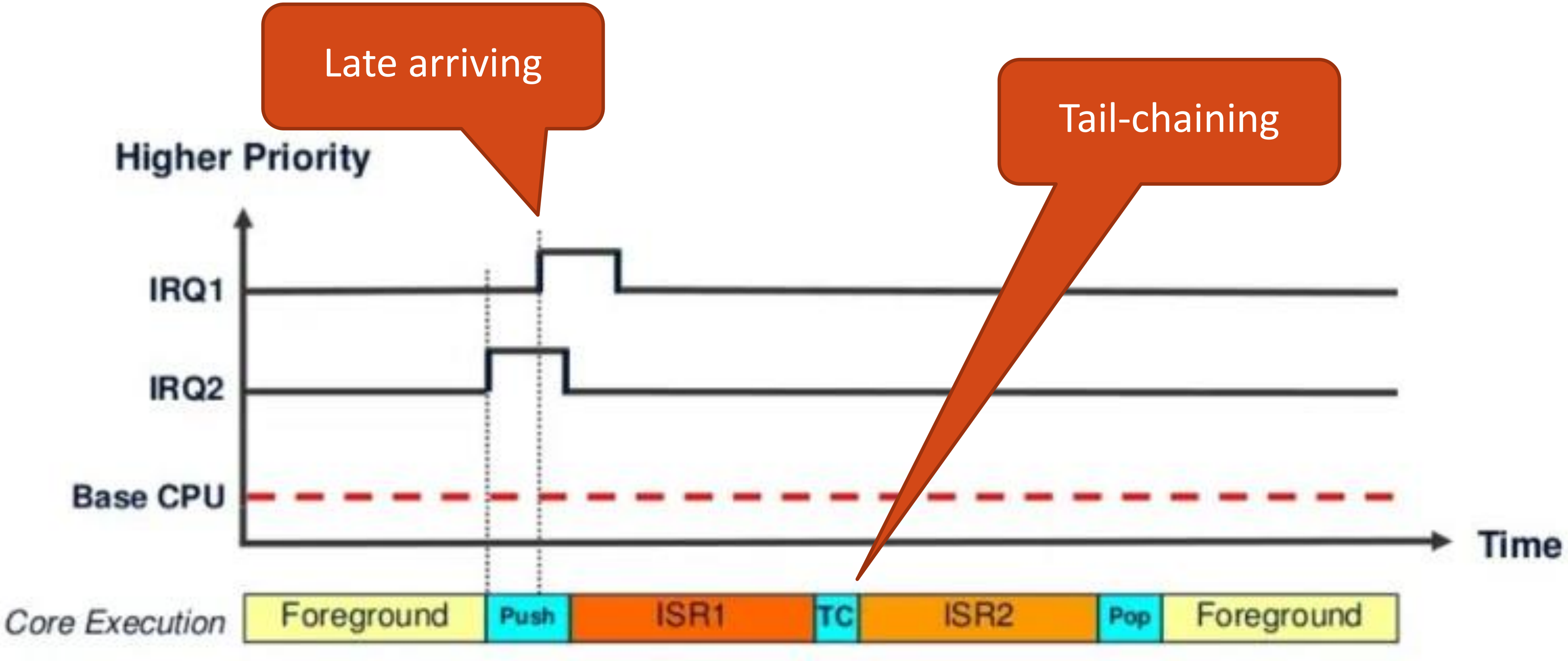


Nesting

Veriženje prekinitev (Tail-chaining)

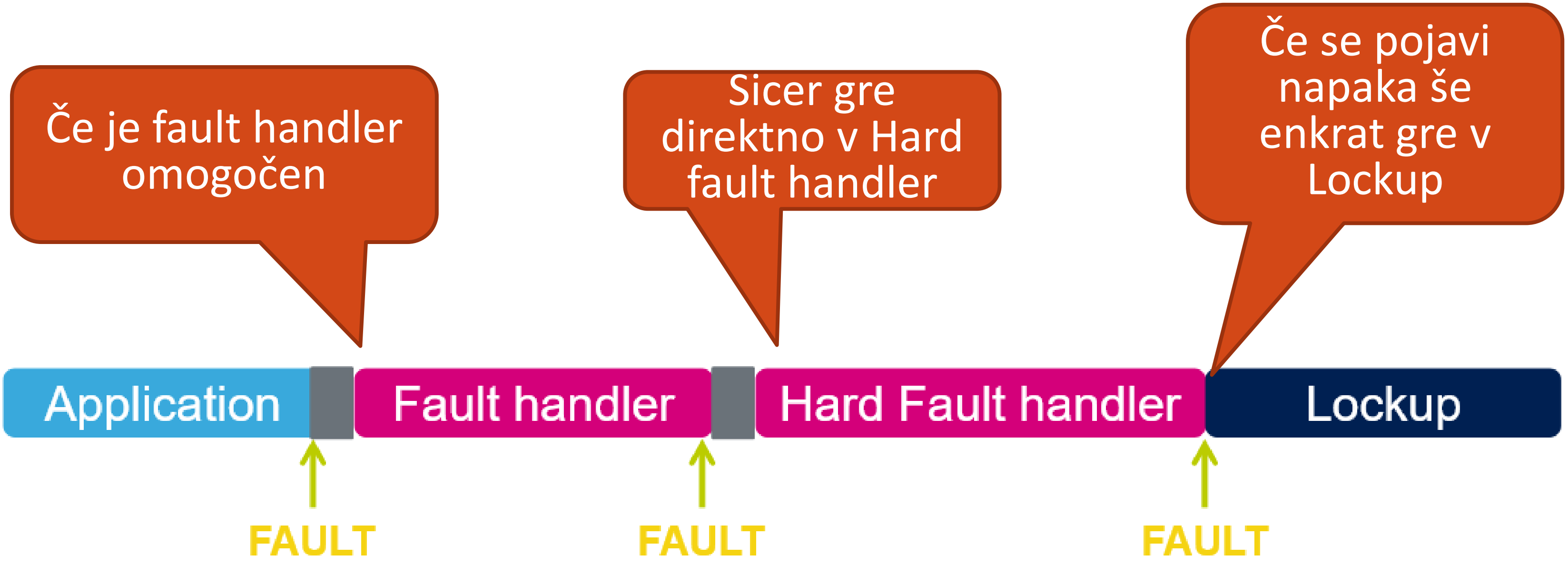


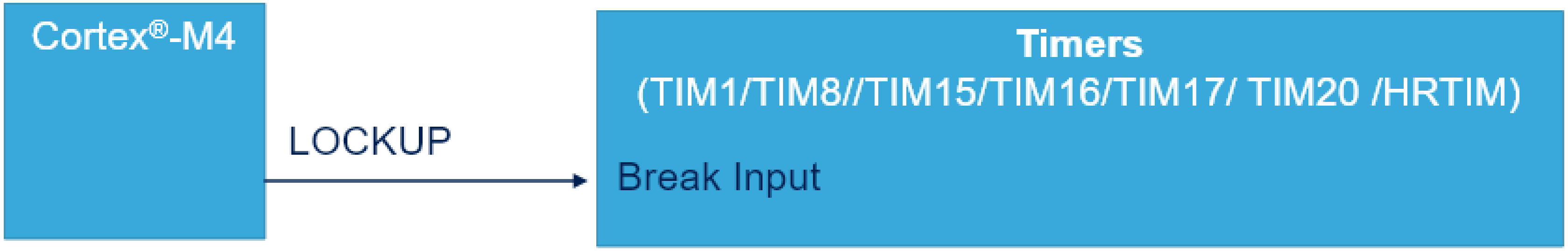
Prekinitev “tik pred zdajci” (Late arriving)


Late arriving

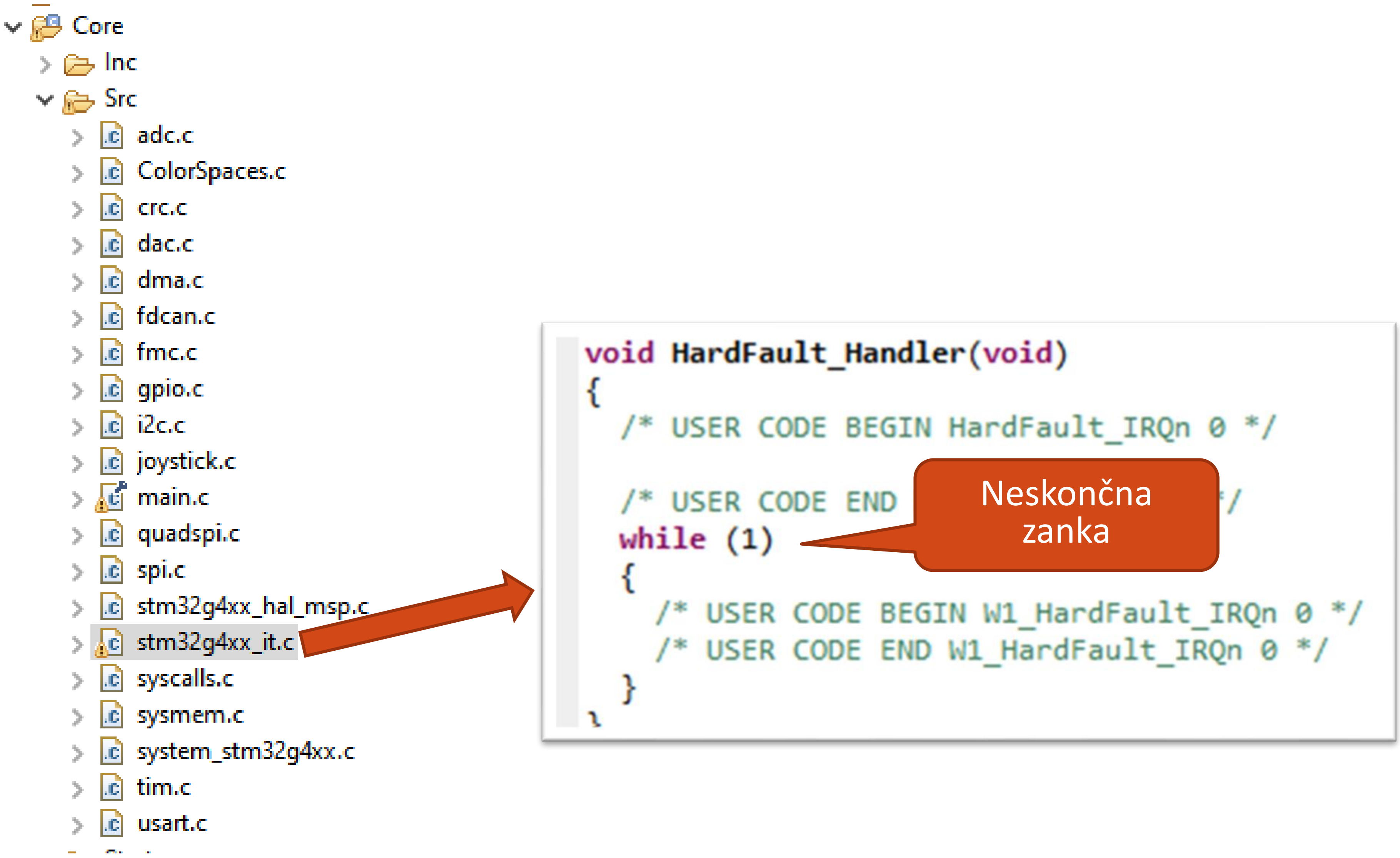
Tail-chaining

Upravljanje z napakami


Če je fault handler omogočen
Sicer gre direktno v Hard fault handler
Če se pojavi napaka še enkrat gre v Lockup
Application
Fault handler
Hard Fault handler
Lockup
FAULT
FAULT
FAULT



ISR za upravljanje z napakami

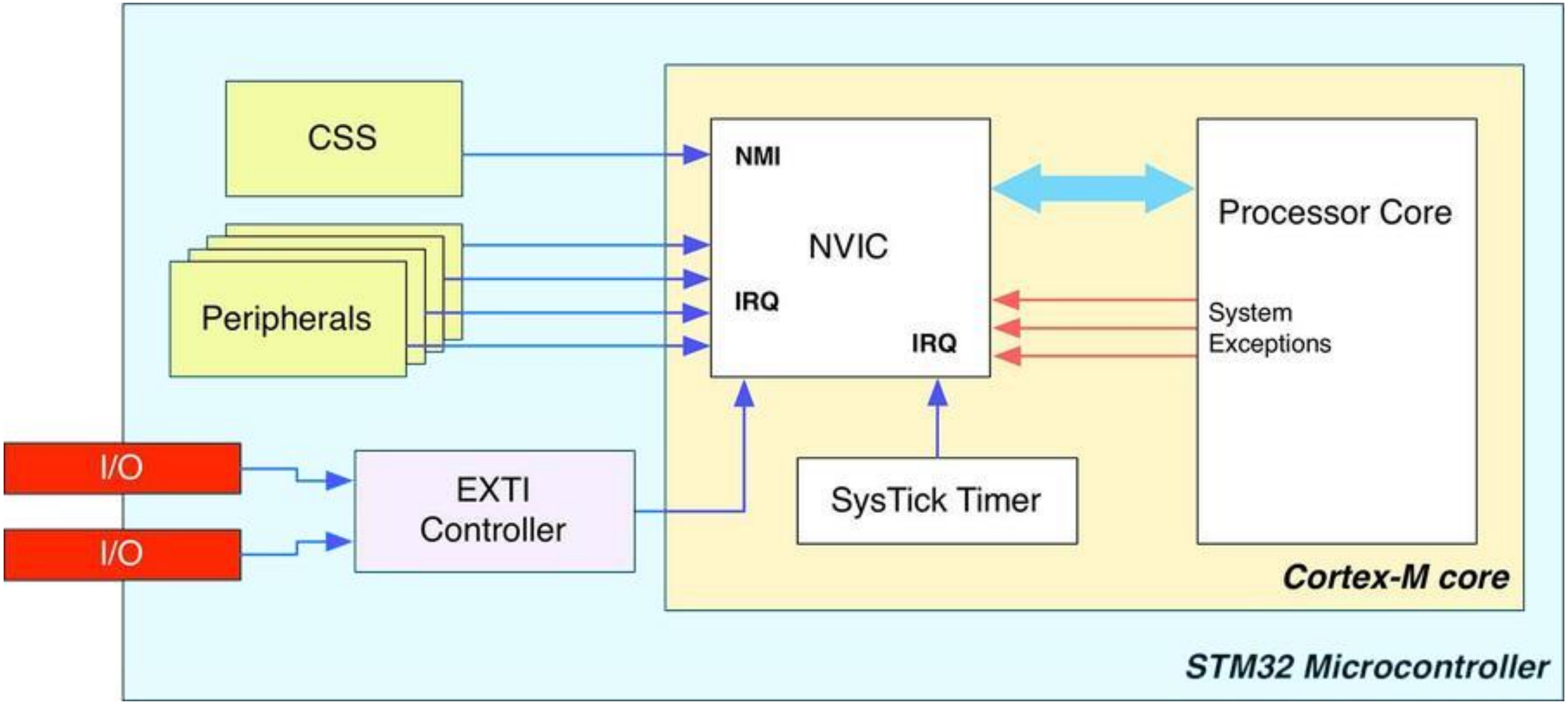


- Core
  - Inc
  - Src
    - adc.c
    - ColorSpaces.c
    - crc.c
    - dac.c
    - dma.c
    - fdcan.c
    - fmc.c
    - gpio.c
    - i2c.c
    - joystick.c
    - main.c
    - quadspi.c
    - spi.c
    - stm32g4xx_hal_msp.c
    - stm32g4xx_it.c
    - syscalls.c
    - sysmem.c
    - system_stm32g4xx.c
    - tim.c
    - usart.c

```
void HardFault_Handler(void)
{
  /* USER CODE BEGIN HardFault_IRQn 0 */

  /* USER CODE END                     */
  while (1)
  {
    /* USER CODE BEGIN W1_HardFault_IRQn 0 */
    /* USER CODE END W1_HardFault_IRQn 0 */
  }
}
```

Neskončna zanka

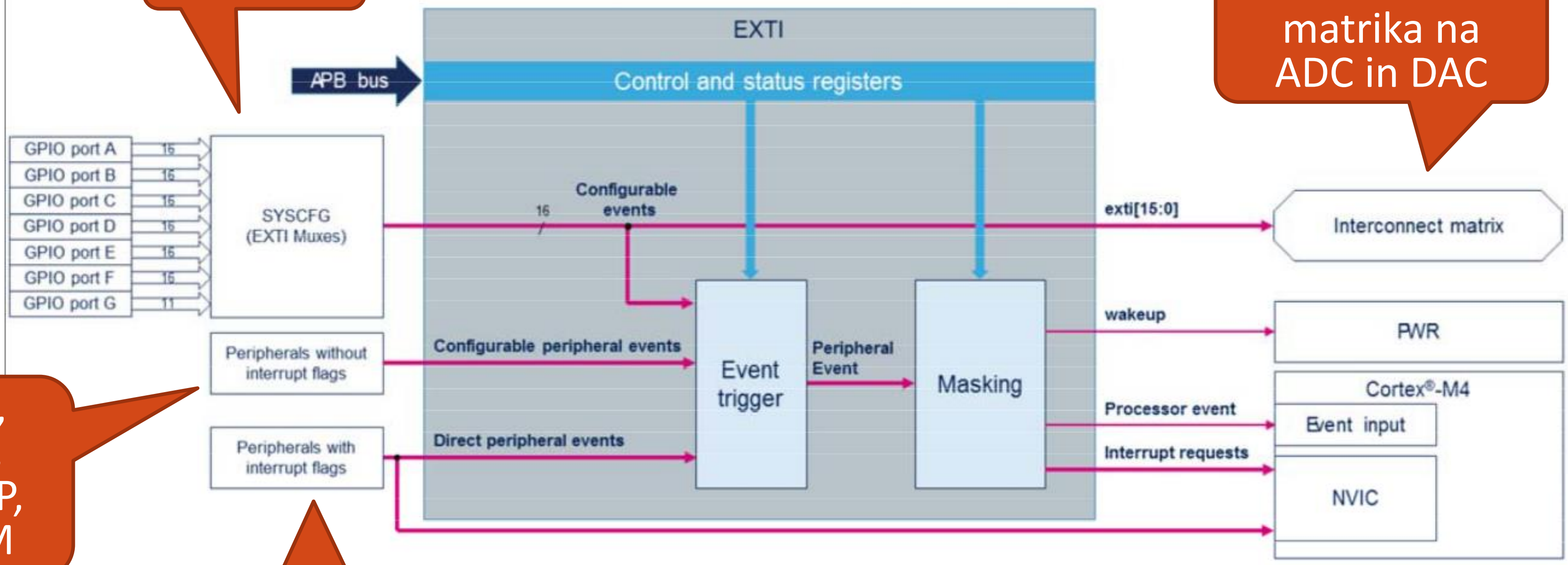
Zunanje prekinitve in dogodki (EXTI)



Zunanje prekinitve in dogodki (EXTI)

GPIO

Povezovalna matrika na ADC in DAC

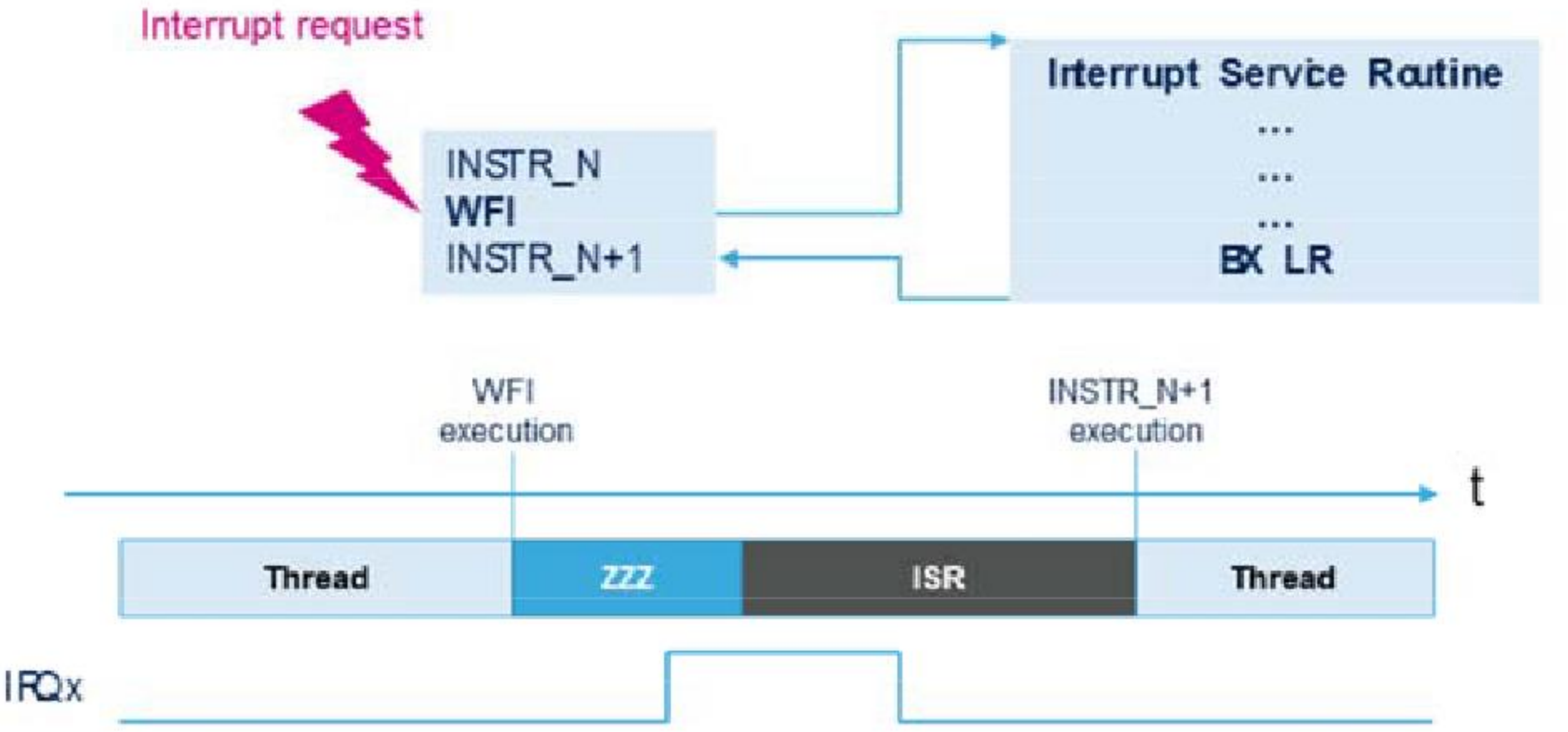


PVD, RTC, COMP, PWM

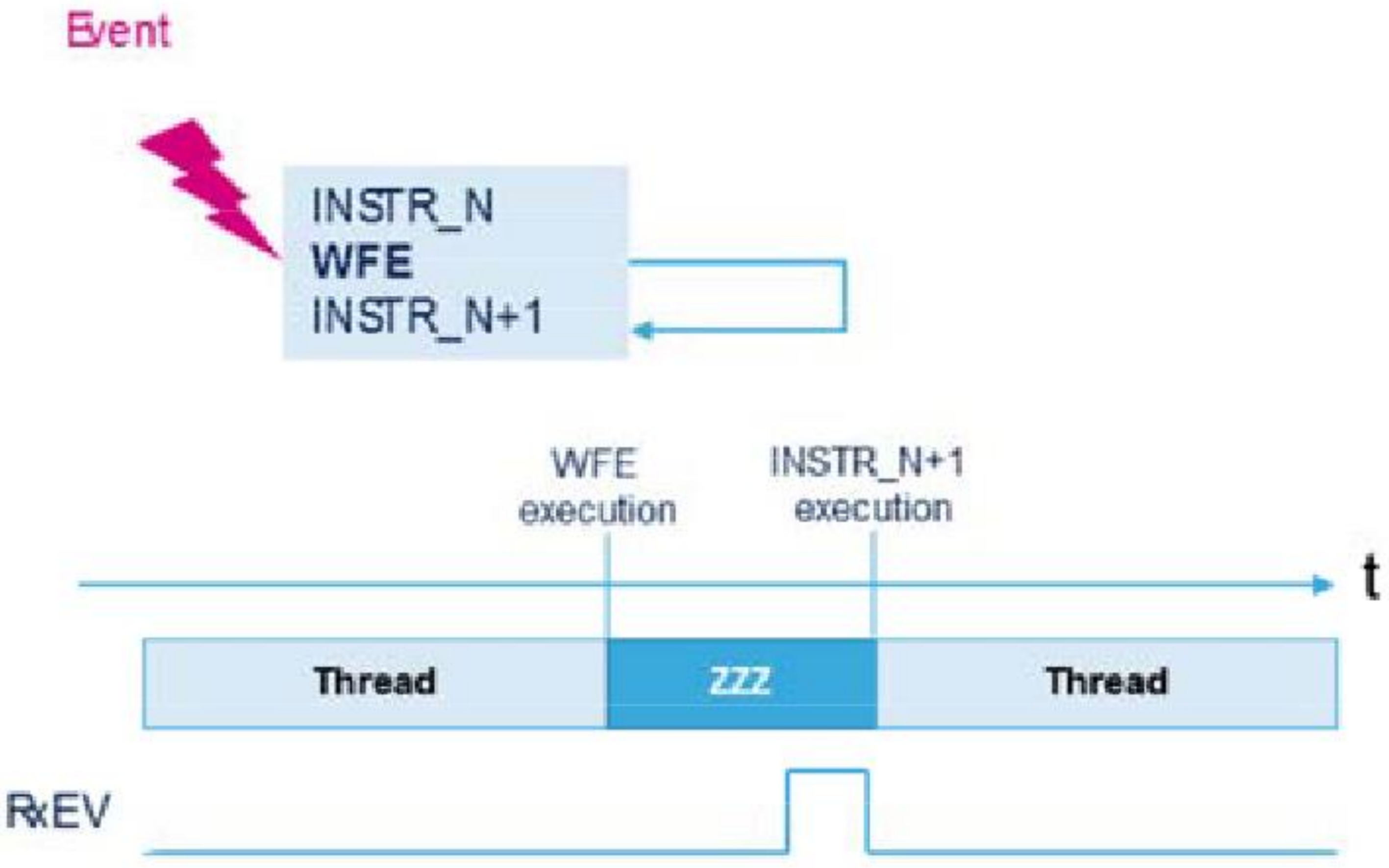
USB, I2C, U(S)ART, …

Namenjeno bujenju

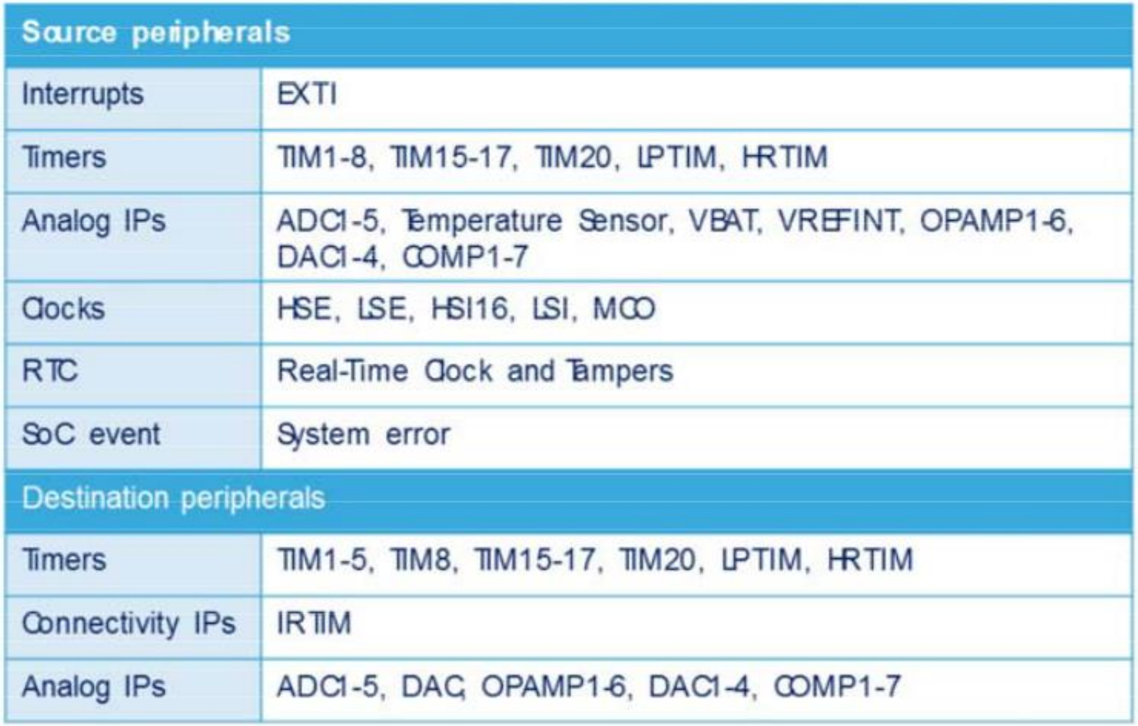
Prekinitev


Interrupt request
INSTR_N
WFI
INSTR_N+1
Interrupt Service Routine
...
...
...
BX LR
WFI execution
INSTR_N+1 execution
t
Thread
ZZZ
ISR
Thread
IRQx

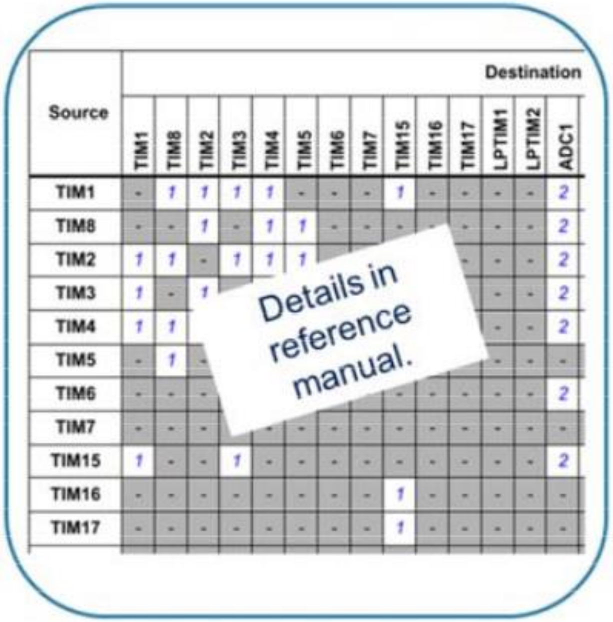
Dogodek


Event
INSTR_N
WFE
INSTR_N+1
WFE
execution
INSTR_N+1
execution
t
Thread
zzz
Thread
RxEV

(Povezovalna matrika) Interconnection matrix



Source peripherals	
Interrupts	EXTI
Timers	TIM1-8, TIM15-17, TIM20, LPTIM, HRTIM
Analog IPs	ADC1-5, Temperature Sensor, VBAT, VREFINT, OPAMP1-6, DAC1-4, COMP1-7
Clocks	HSE, LSE, HSI16, LSI, MCO
RTC	Real-Time Clock and Tampers
SoC event	System error
Destination peripherals	
Timers	TIM1-5, TIM8, TIM15-17, TIM20, LPTIM, HRTIM
Connectivity IPs	IRTIM
Analog IPs	ADC1-5, DAC, OPAMP1-6, DAC1-4, COMP1-7

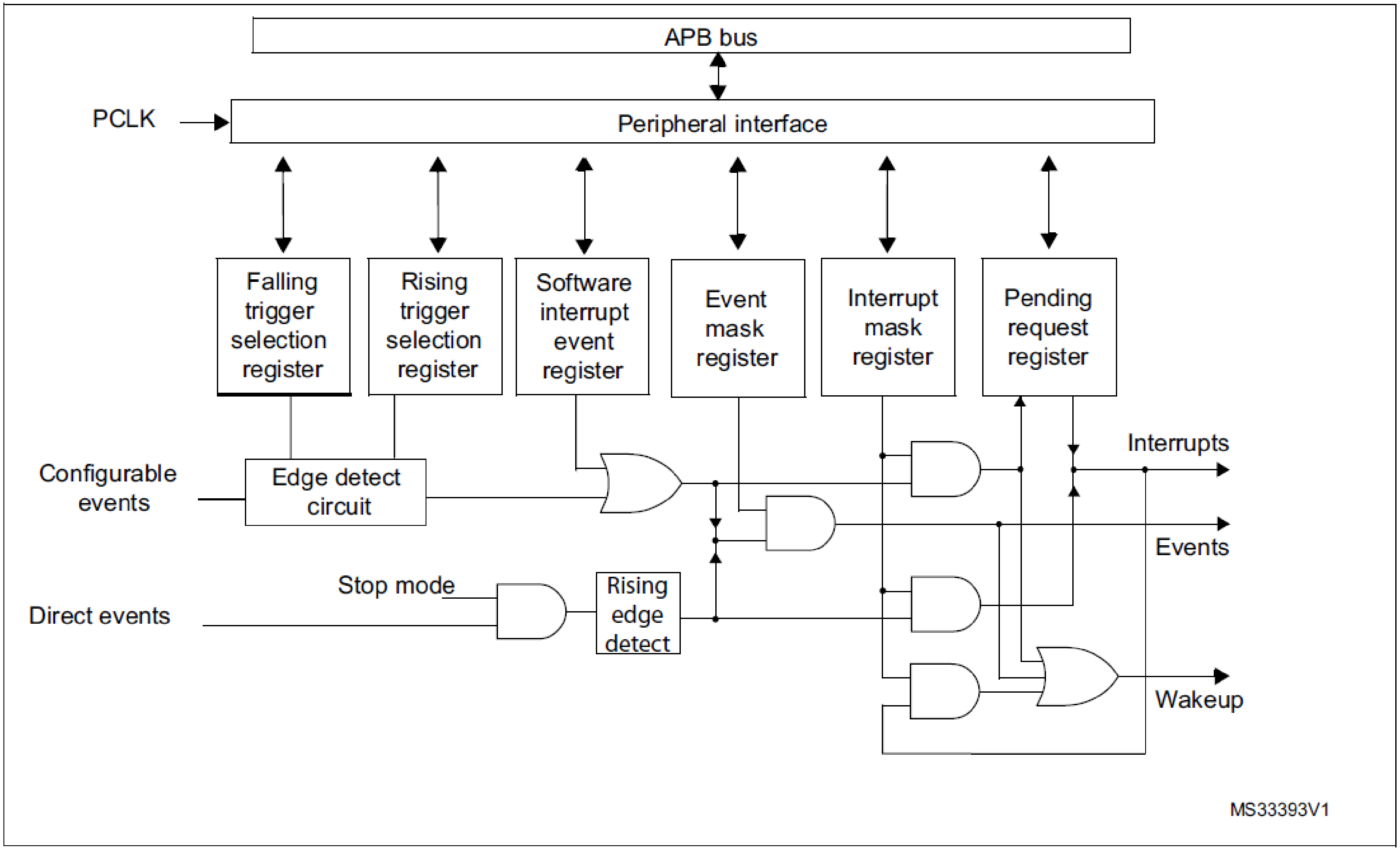


Source	Destination													
	TIM1	TIM8	TIM2	TIM3	TIM4	TIM5	TIM6	TIM7	TIM15	TIM16	TIM17	LPTIM1	LPTIM2	ADC1
TIM1	-	1	1	1	1	-	-	-	1	-	-	-	-	2
TIM8	-	-	1	-	1	1	-	-	-	-	-	-	-	2
TIM2	1	1	-	1	1	1	[illegible]	[illegible]	[illegible]	[illegible]	-	-	-	2
TIM3	1	-	1	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-	-	2
TIM4	1	1	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-	-	2
TIM5	-	1	-	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-	-	-
TIM6	-	-	-	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-	-	-	-	-	2
TIM7	-	-	-	[illegible]	-	-	-	-	-	-	-	-	-	-
TIM15	1	-	-	1	-	-	-	-	-	-	-	-	-	2
TIM16	-	-	-	-	-	-	-	-	1	-	-	-	-	-
TIM17	-	-	-	-	-	-	-	-	1	-	-	-	-	-

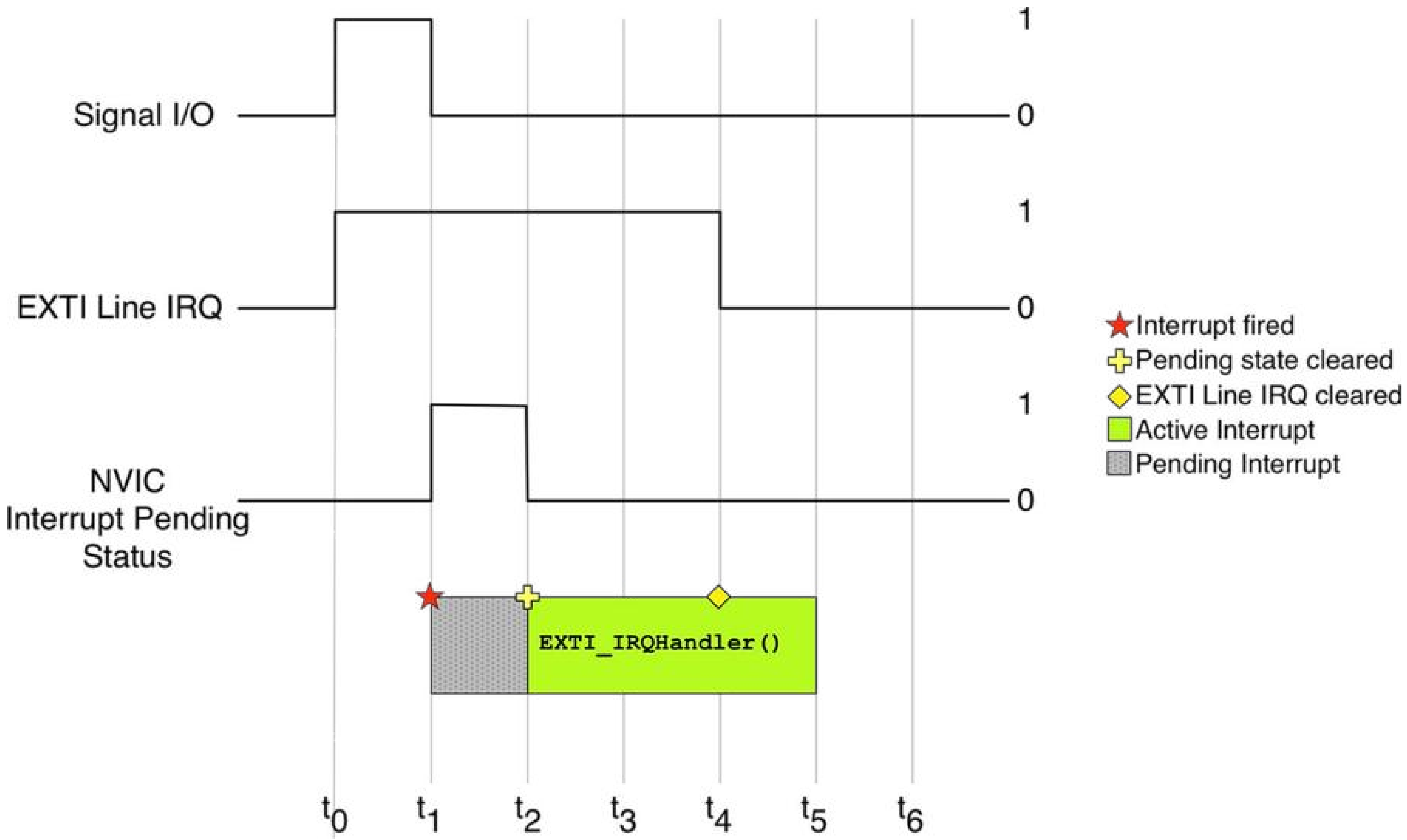
Details in reference manual.

Zunanje prekinitve in dogodki (EXTI)

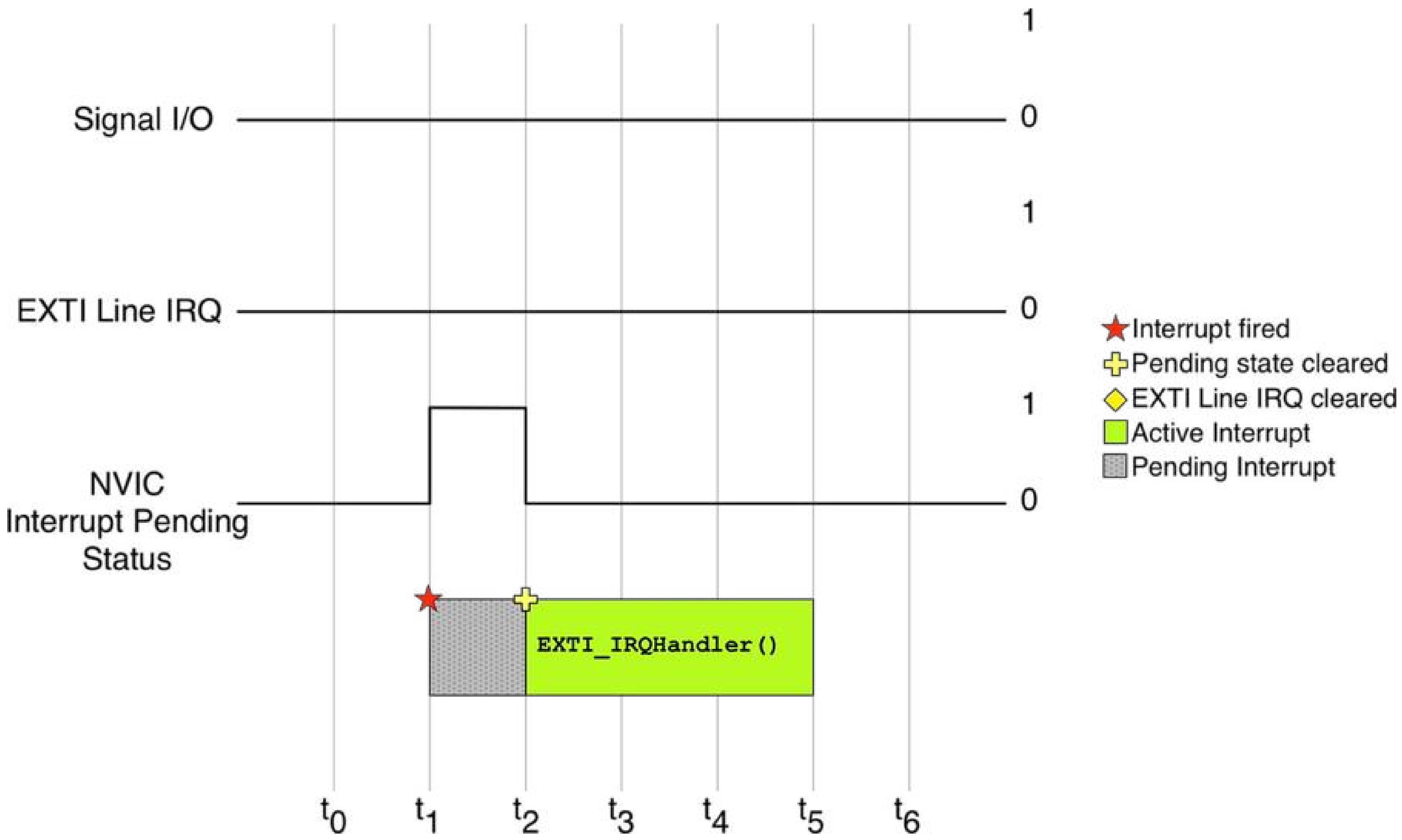
Figure 35. Configurable interrupt/event block diagram



Strojno proženje prekinitve

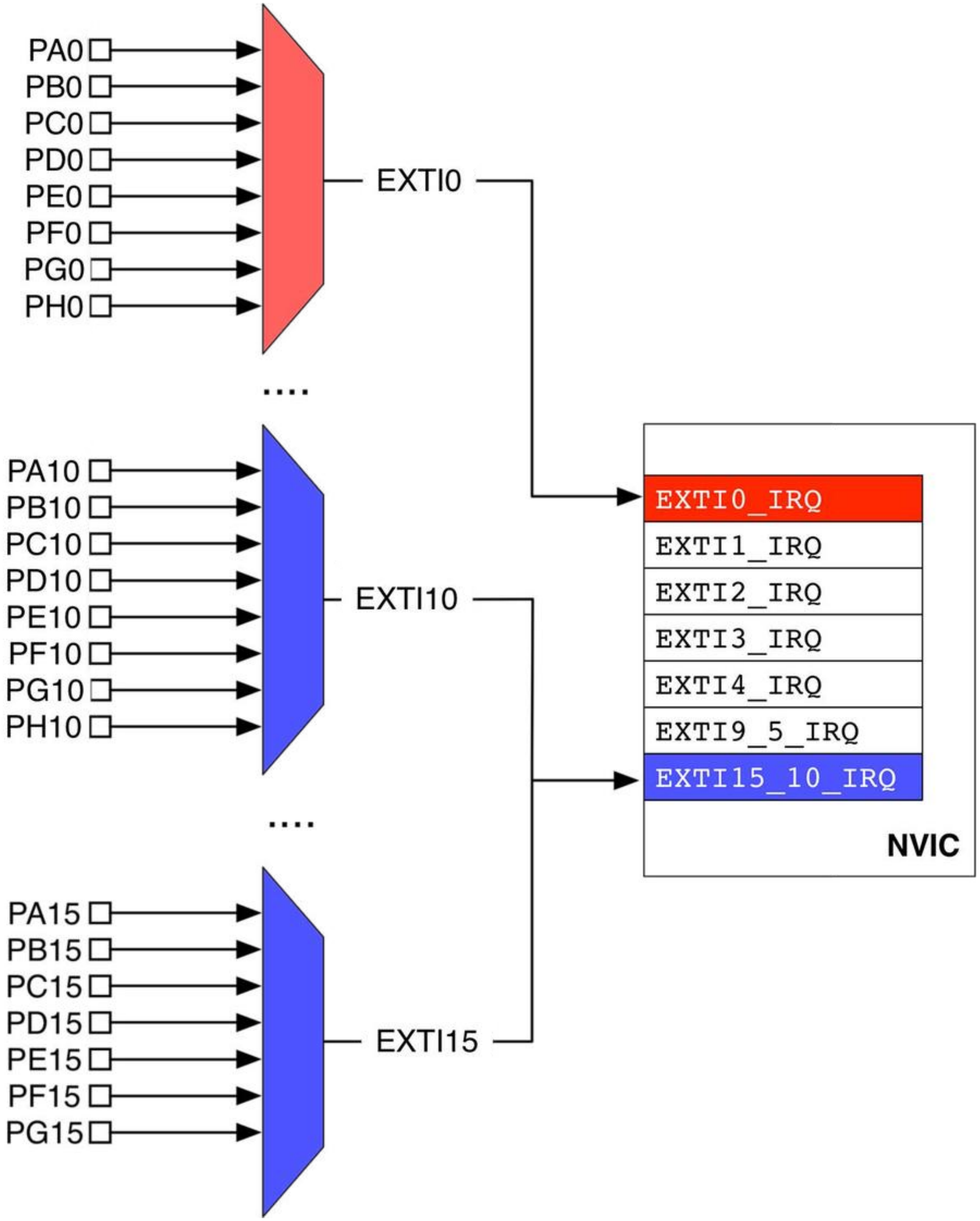


Programsko proženje prekinitve

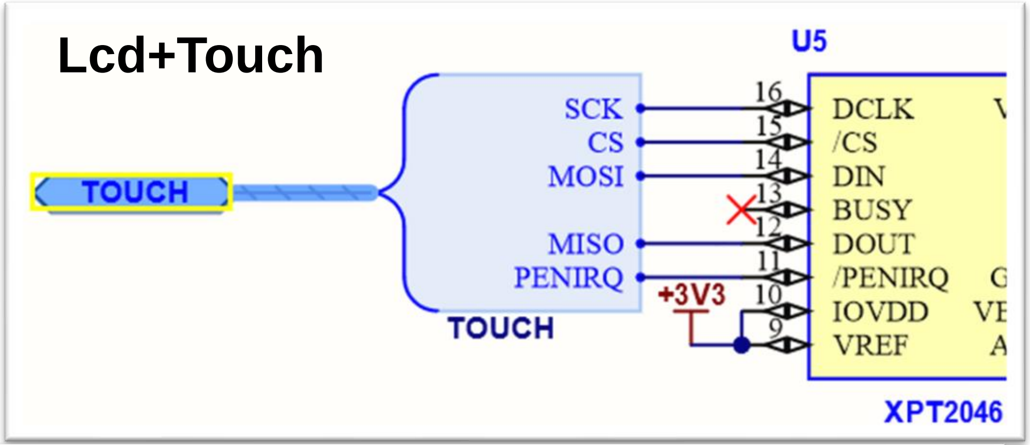


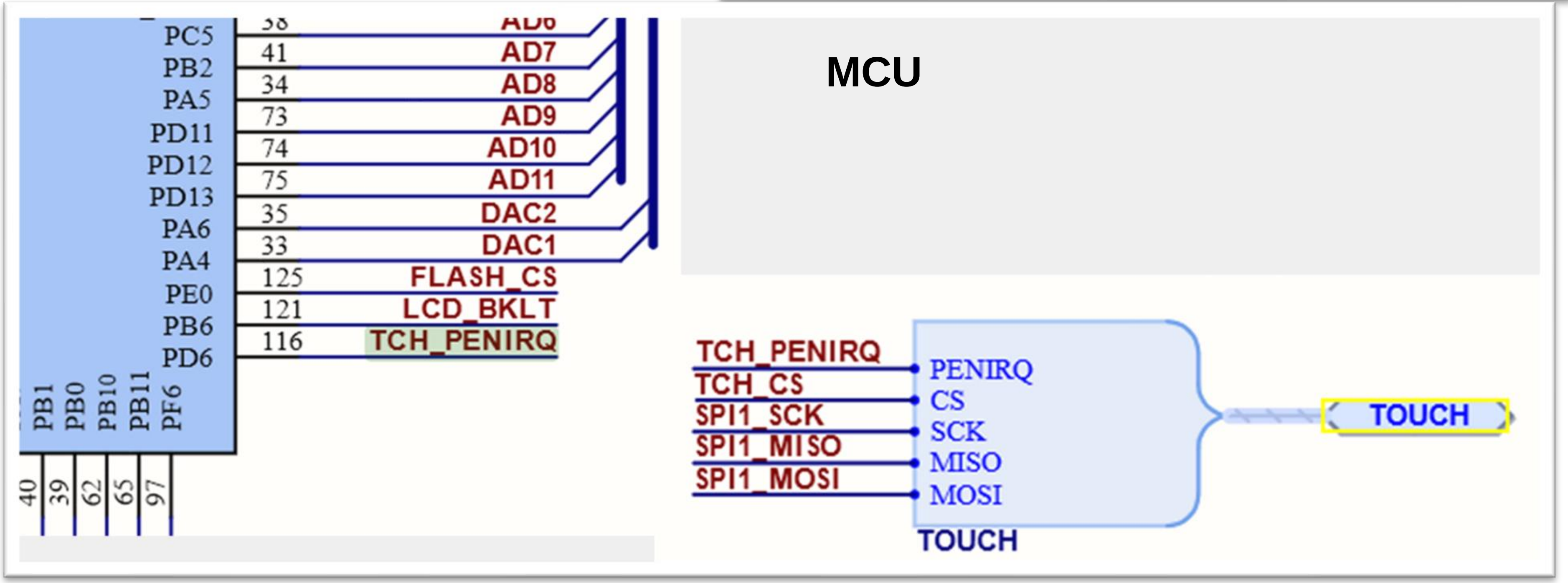
Izbira GPIO vhoda za EXTI

Samo eden naenkrat!!!

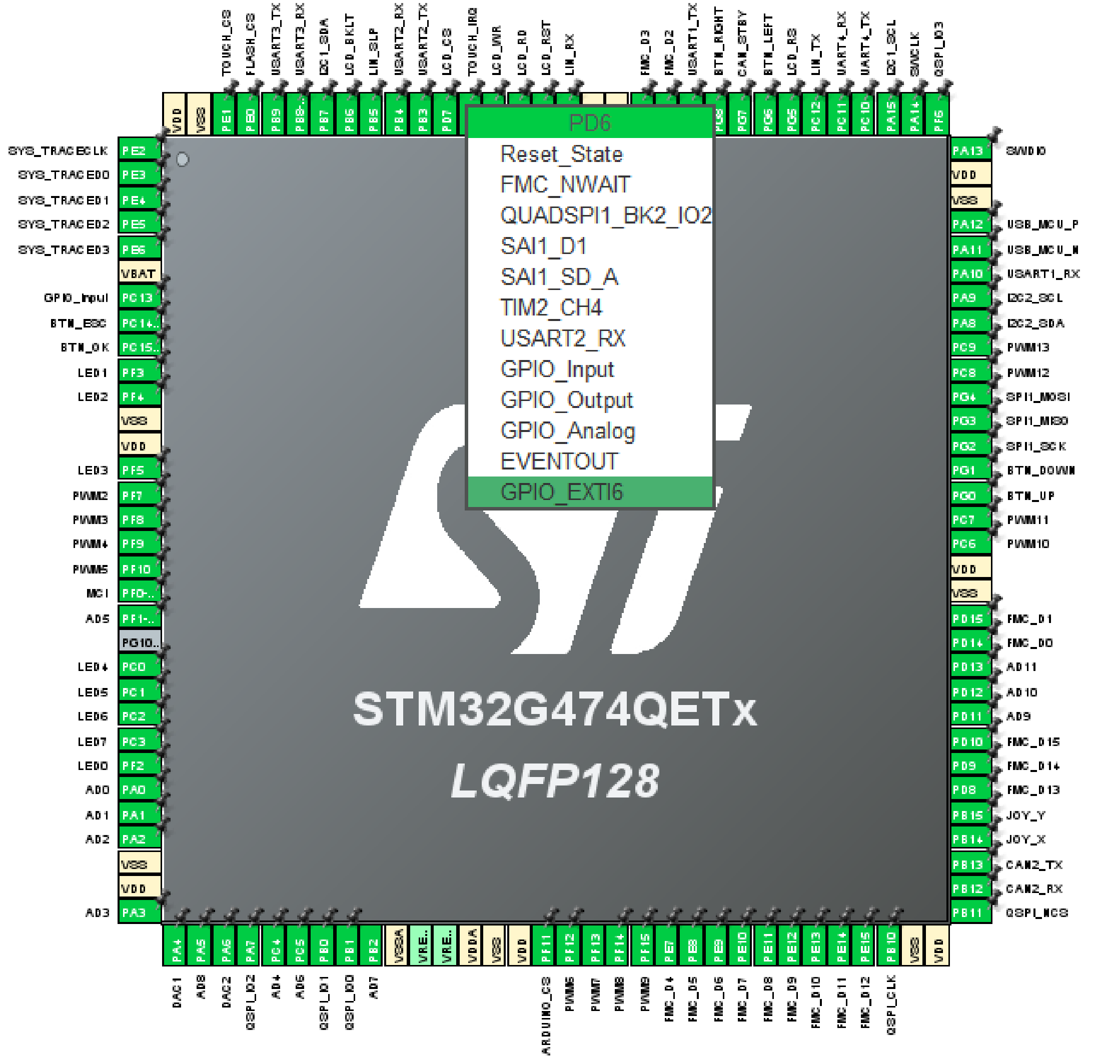


Primer detekcije pritiska na zaslon: TOUCH_IRQ

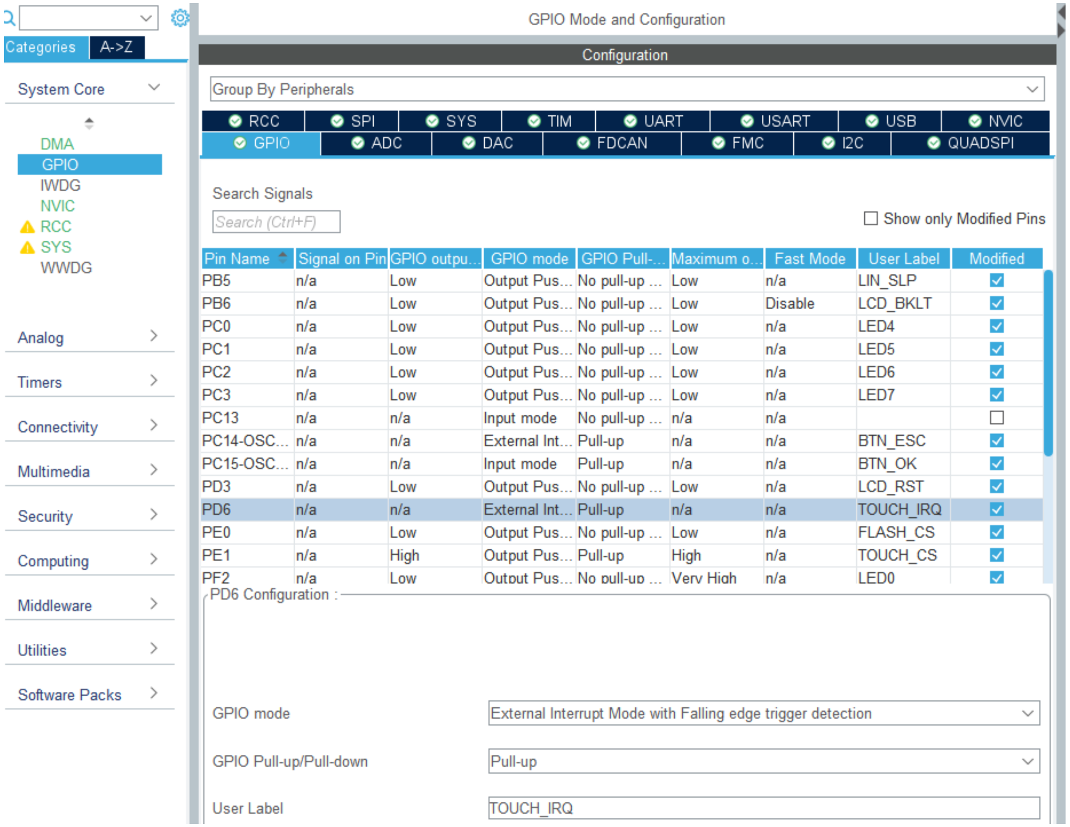




Primer detekcije pritiska na zaslon: TOUCH_IRQ



Primer detekcije pritiska na zaslon: TOUCH_IRQ



GPIO Mode and Configuration

Configuration

Group By Peripherals

Search Signals

Search (Ctrl+F)

□ Show only Modified Pins

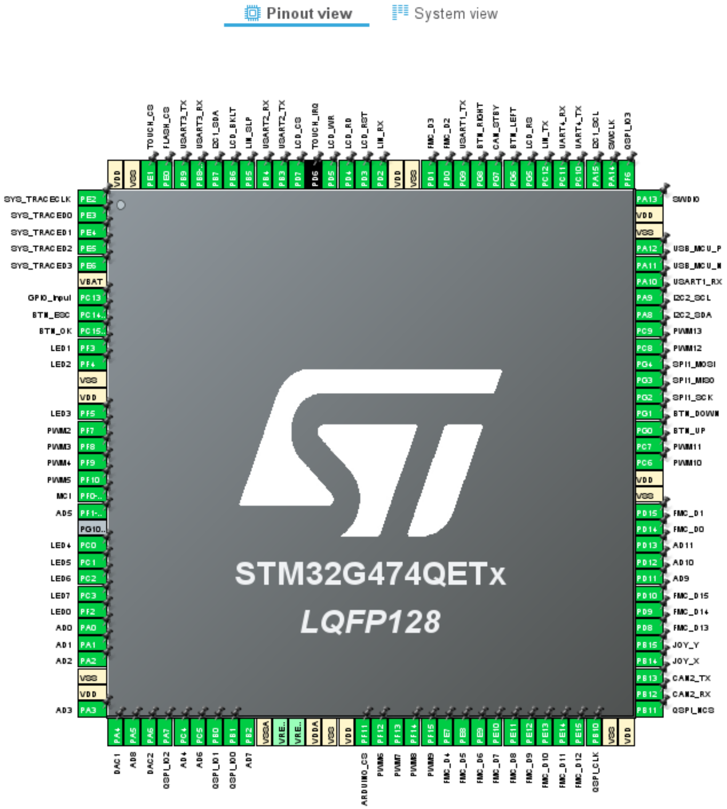
Pin Name	Signal on Pin	GPIO outpu...	GPIO mode	GPIO Pull-...	Maximum o...	Fast Mode	User Label	Modified
PB5	n/a	Low	Output Pus...	No pull-up ...	Low	n/a	LIN_SLP	☑
PB6	n/a	Low	Output Pus...	No pull-up ...	Low	Disable	LCD_BKLT	☑
PC0	n/a	Low	Output Pus...	No pull-up ...	Low	n/a	LED4	☑
PC1	n/a	Low	Output Pus...	No pull-up ...	Low	n/a	LED5	☑
PC2	n/a	Low	Output Pus...	No pull-up ...	Low	n/a	LED6	☑
PC3	n/a	Low	Output Pus...	No pull-up ...	Low	n/a	LED7	☑
PC13	n/a	n/a	Input mode	No pull-up ...	n/a	n/a		□
PC14-OSC...	n/a	n/a	External Int...	Pull-up	n/a	n/a	BTN_ESC	☑
PC15-OSC...	n/a	n/a	Input mode	Pull-up	n/a	n/a	BTN_OK	☑
PD3	n/a	Low	Output Pus...	No pull-up ...	Low	n/a	LCD_RST	☑
PD6	n/a	n/a	External Int...	Pull-up	n/a	n/a	TOUCH_IRQ	☑
PE0	n/a	Low	Output Pus...	No pull-up ...	Low	n/a	FLASH_CS	☑
PE1	n/a	High	Output Pus...	Pull-up	High	n/a	TOUCH_CS	☑
PF2	n/a	Low	Output Pus...	No pull-up ...	Very High	n/a	LED0	☑

PD6 Configuration :

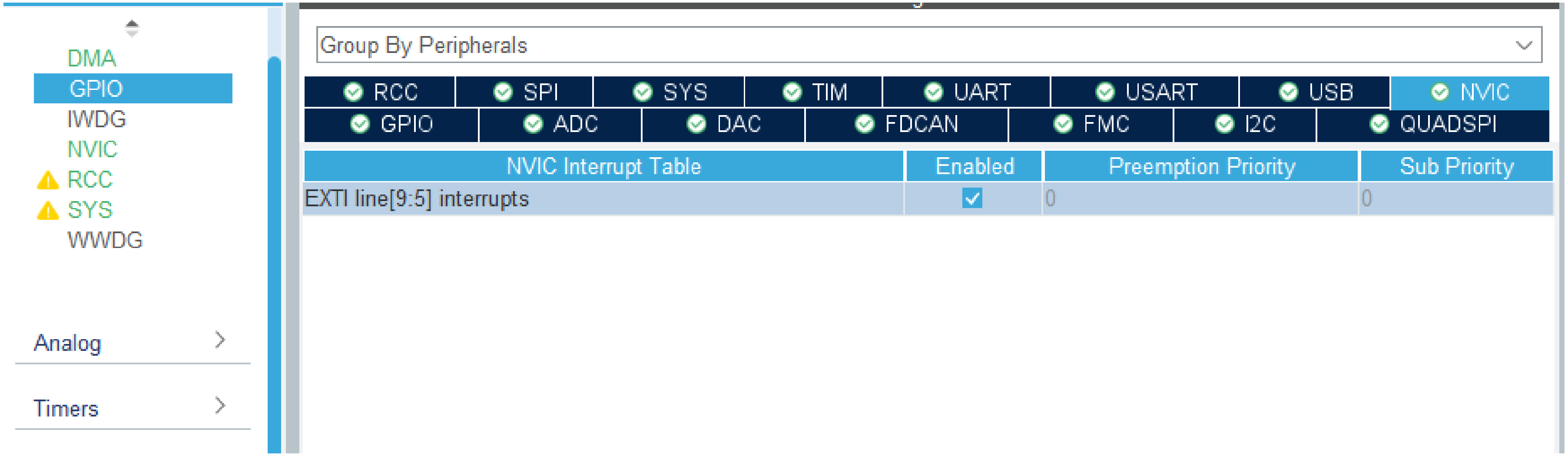
GPIO mode: External Interrupt Mode with Falling edge trigger detection

GPIO Pull-up/Pull-down: Pull-up

User Label: TOUCH_IRQ

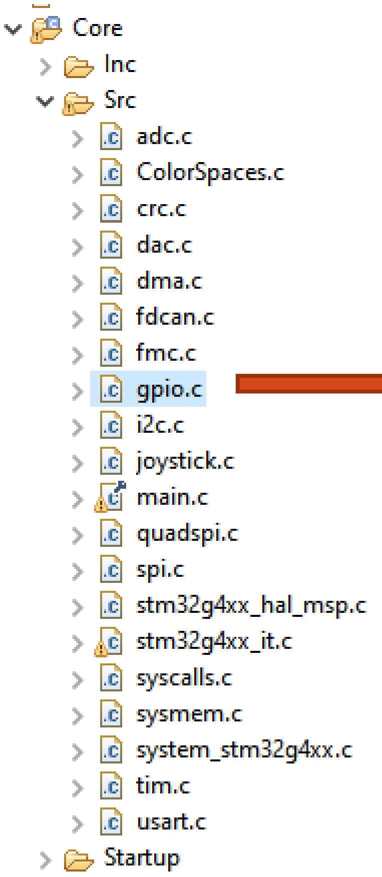


EXTI linija 9:5

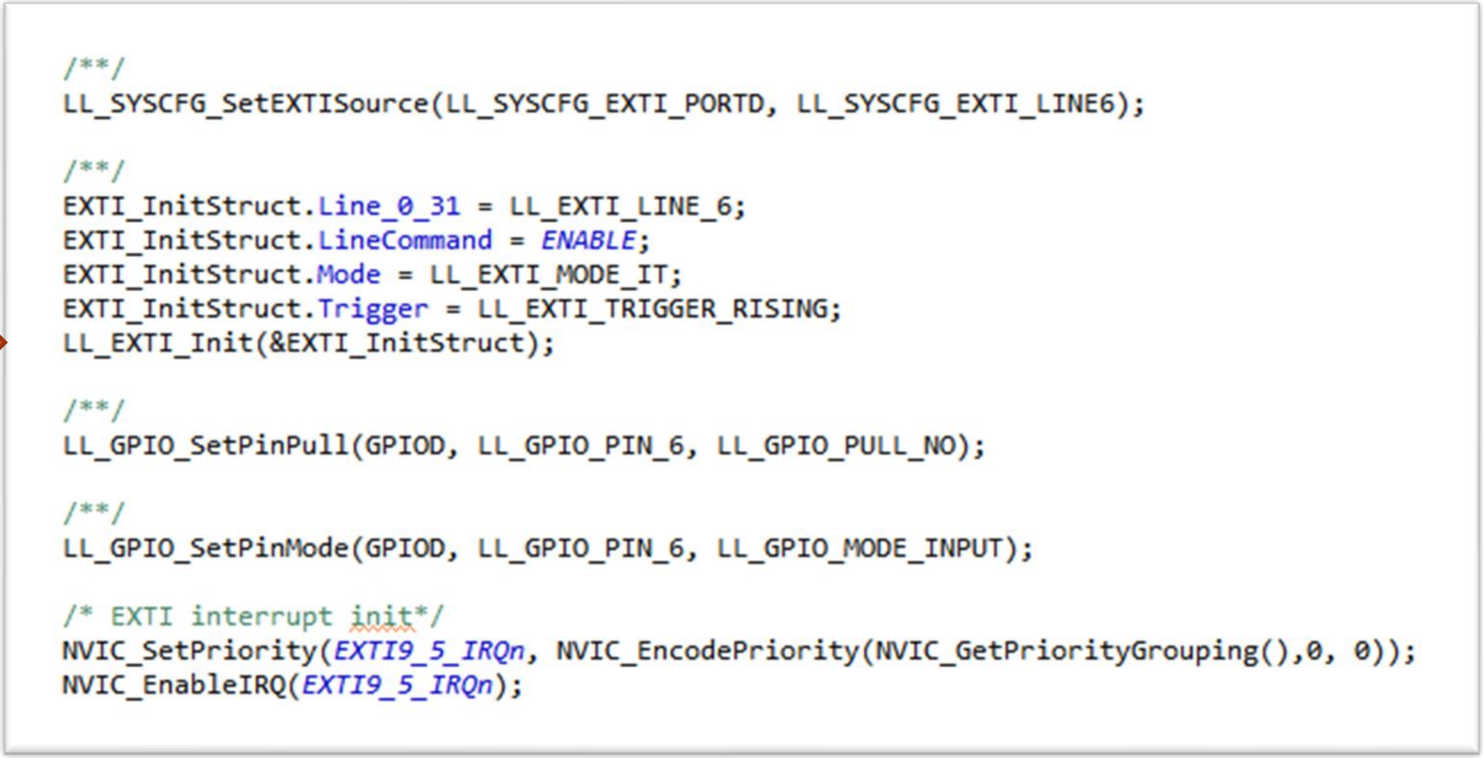


NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
EXTI line[9:5] interrupts	☑	0	0

Inicializacija EXTI[9:5] na PORTD, pin 6

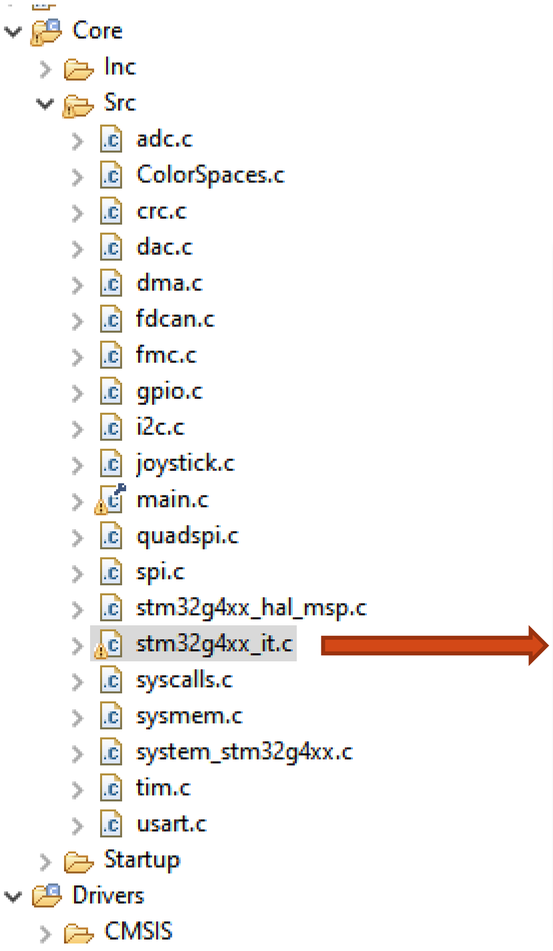


- Core
  - Inc
  - Src
    - adc.c
    - ColorSpaces.c
    - crc.c
    - dac.c
    - dma.c
    - fdcan.c
    - fmc.c
    - gpio.c
    - i2c.c
    - joystick.c
    - main.c
    - quadspi.c
    - spi.c
    - stm32g4xx_hal_msp.c
    - stm32g4xx_it.c
    - syscalls.c
    - sysmem.c
    - system_stm32g4xx.c
    - tim.c
    - usart.c
  - Startup

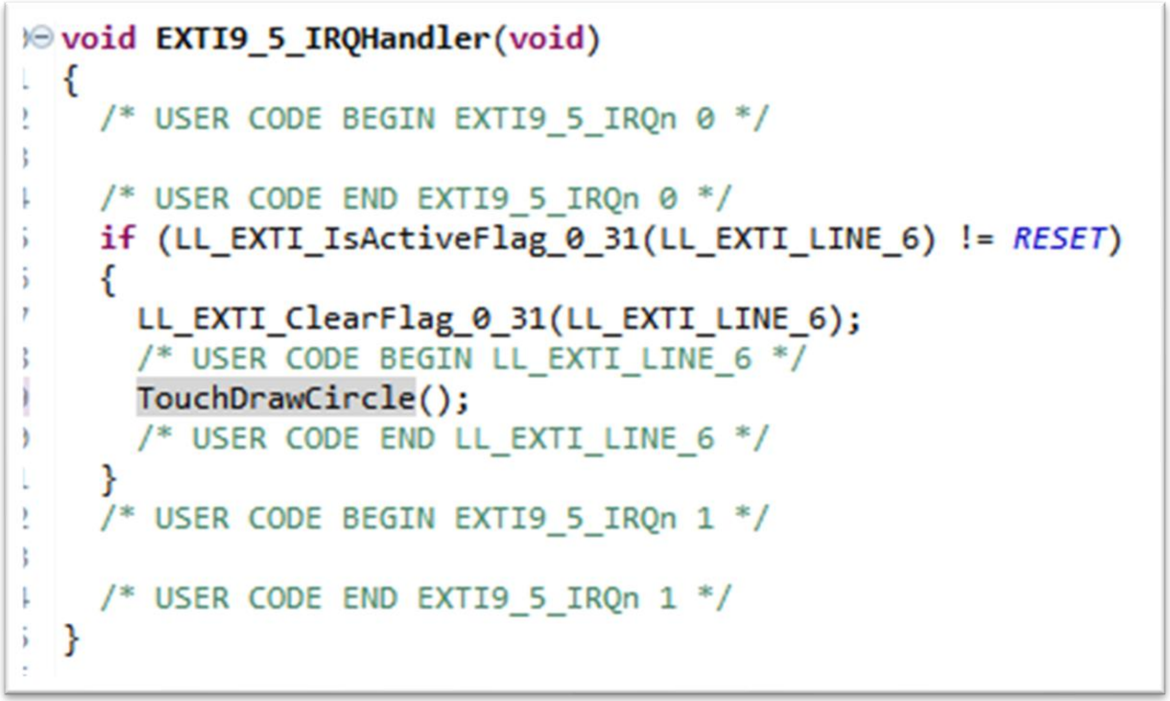


```
/**/
LL_SYSCFG_SetEXTISource(LL_SYSCFG_EXTI_PORTD, LL_SYSCFG_EXTI_LINE6);

/**/
EXTI_InitStruct.Line_0_31 = LL_EXTI_LINE_6;
EXTI_InitStruct.LineCommand = ENABLE;
EXTI_InitStruct.Mode = LL_EXTI_MODE_IT;
EXTI_InitStruct.Trigger = LL_EXTI_TRIGGER_RISING;
LL_EXTI_Init(&EXTI_InitStruct);

/**/
LL_GPIO_SetPinPull(GPIOD, LL_GPIO_PIN_6, LL_GPIO_PULL_NO);

/**/
LL_GPIO_SetPinMode(GPIOD, LL_GPIO_PIN_6, LL_GPIO_MODE_INPUT);

/* EXTI interrupt init*/
NVIC_SetPriority(EXTI9_5_IRQn, NVIC_EncodePriority(NVIC_GetPriorityGrouping(),0, 0));
NVIC_EnableIRQ(EXTI9_5_IRQn);
```

IRQ EXTI[5:9] Handler funkcija



```
Core
  Inc
  Src
    adc.c
    ColorSpaces.c
    crc.c
    dac.c
    dma.c
    fdcan.c
    fmc.c
    gpio.c
    i2c.c
    joystick.c
    main.c
    quadspi.c
    spi.c
    stm32g4xx_hal_msp.c
    stm32g4xx_it.c
    syscalls.c
    sysmem.c
    system_stm32g4xx.c
    tim.c
    usart.c
  Startup
Drivers
  CMSIS
```



```
void EXTI9_5_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI9_5_IRQn 0 */

  /* USER CODE END EXTI9_5_IRQn 0 */
  if (LL_EXTI_IsActiveFlag_0_31(LL_EXTI_LINE_6) != RESET)
  {
    LL_EXTI_ClearFlag_0_31(LL_EXTI_LINE_6);
    /* USER CODE BEGIN LL_EXTI_LINE_6 */
    TouchDrawCircle();
    /* USER CODE END LL_EXTI_LINE_6 */
  }
  /* USER CODE BEGIN EXTI9_5_IRQn 1 */

  /* USER CODE END EXTI9_5_IRQn 1 */
}
```