



Vhodno-izhodne naprave (VIN)

Predavanja

9. Programiranje in povezovanje V/I naprav

Robert Rozman

rozman@fri.uni-lj.si



Teškoča obvestila

Vsebina

Uvod

1. V/I sistem: naprave in prenosne poti
 1. Programski prenosi V/I (ang. Programmed I/O – „PIO“)
 2. Neposreden dostop do pomnilnika (ang. Direct Memory Access – DMA)
 3. Krmilniki V/I naprav
2. Prekinitve, izjeme, pasti
3. Komunikacija: računalnik <-> V/I naprave

Uvod

□ Vhodno/izhodni sistem (V/I sistem) sestavljajo

▪ V/I naprave

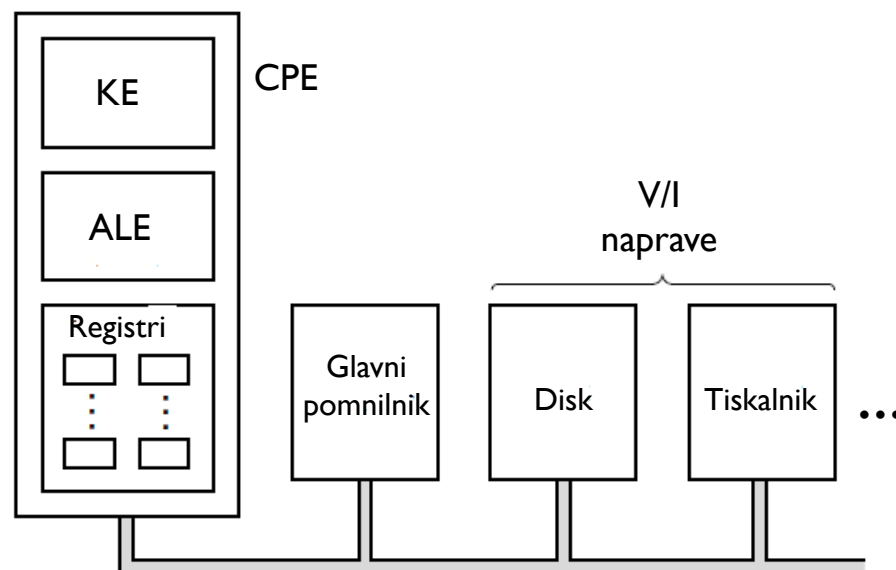
- Uporaba: vhod – enkratno branje, izhod – samo pisanje, zunanji pomnilniki – večkratno branje in pisanje
- Povezava: V/I naprava je povezana s človekom, V/I naprava je povezana z računalnikom, procesorjem
- Hitrost: največja hitrost (b/s) s katero se prenašajo podatki med V/I napravo in procesorjem ali pomnilnikom

▪ Prenosne poti: vodilo, točka-v-točko

▪ Izvajanje V/I operacij

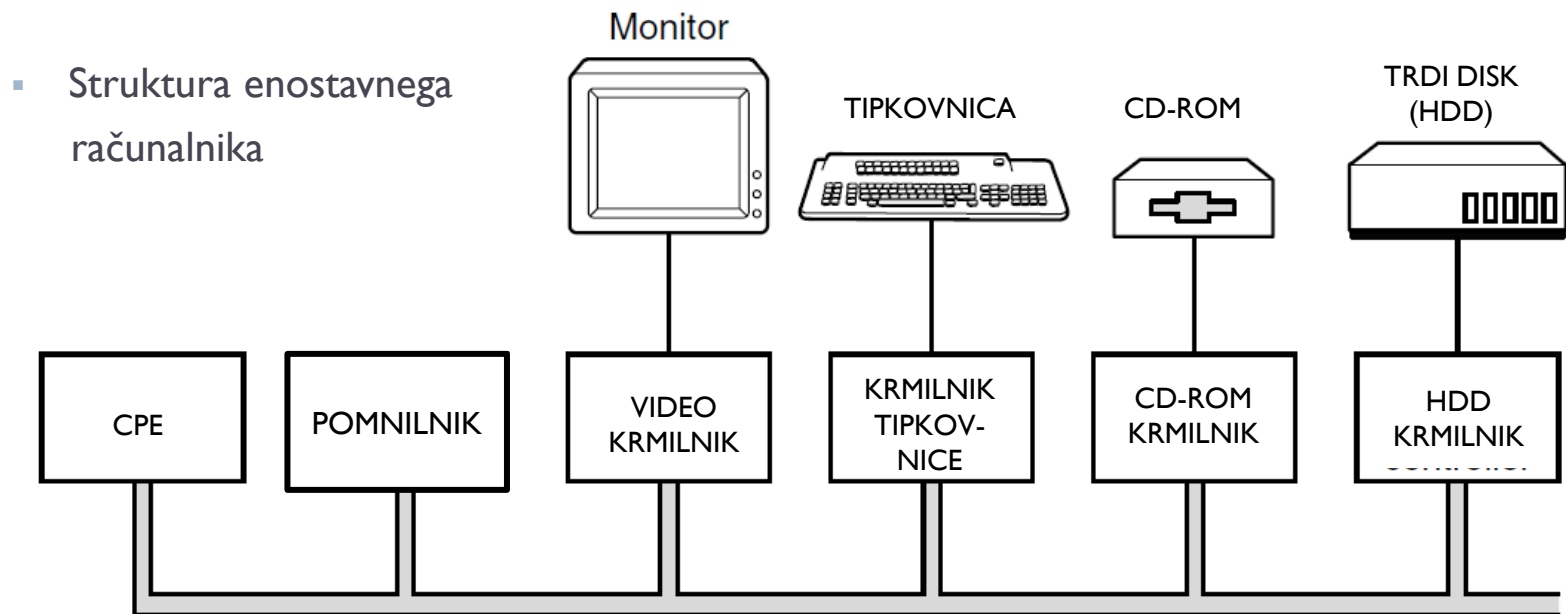
- Programski vhod/izhod
- DMA: Neposreden dostop do pomniln.

▪ Krmilniki V/I naprav



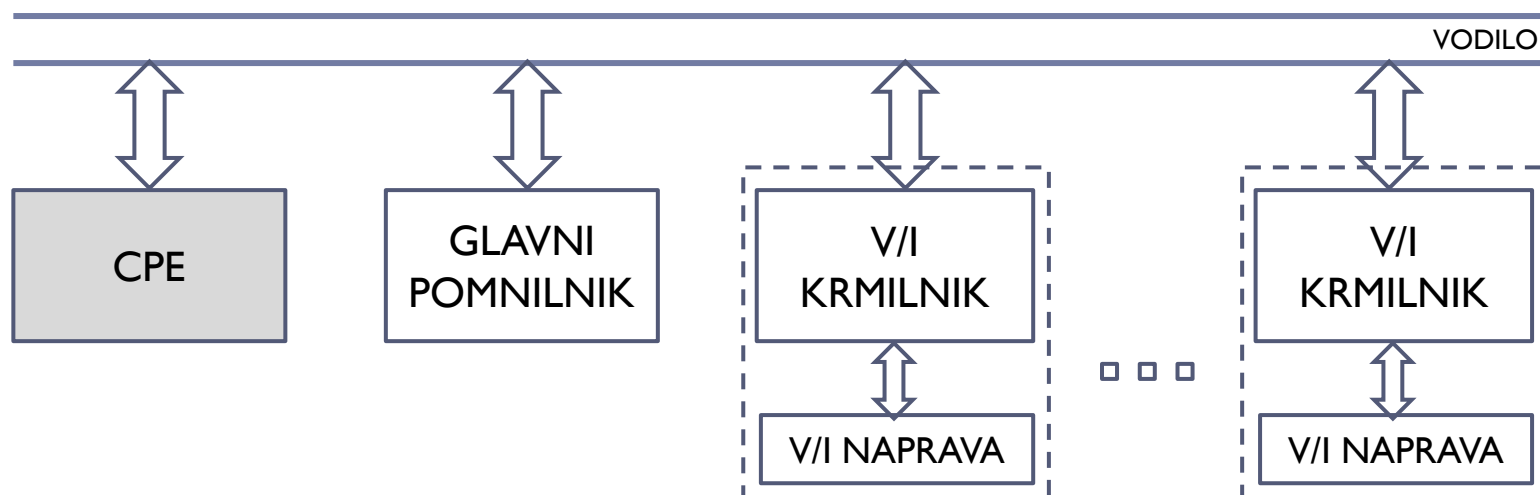
9.1 V/I naprave in prenosne poti (vodila)

- Von Neumannov računalnik - organizacija s CPE in V/I napravami, kjer upoštevamo:
 - V/I naprave za **pretvarjanje informacij** iz ene oblike v drugo (tipkovnica, miška, zaslon, tiskalnik ...).
 - V/I naprave za **shranjevanje informacij** – pomožni pomnilniki (trdi disk, SSD, magnetni trak, DVD ...).
 - **Način delovanja V/I sistema** - prenos podatkov med V/I sistemom, CPE in/ali glavnim pomnilnikom.



9.1.1 Programski prenosi V/I (ang. programmed I/O-PIO)

- Prenos med CPE in v/I napravo je realiziran z ustreznim zaporedjem ukazov. Za prenos vsakega podatka je potrebno izvršiti več ukazov. **CPE izvaja program**, ki:
 - prične V/I operacijo,
 - nadzoruje njeno izvajanje,
 - prenaša podatke,
 - zaključi V/I operacijo.
- Operaciji, ki se uporabljata: branje iz V/I naprave in pisanje na V/I napravo



BRANJE iz V/I naprave preko V/I krmilnika

- Podatek se iz V/I naprave preko CPE zapiše v glavni pomnilnik

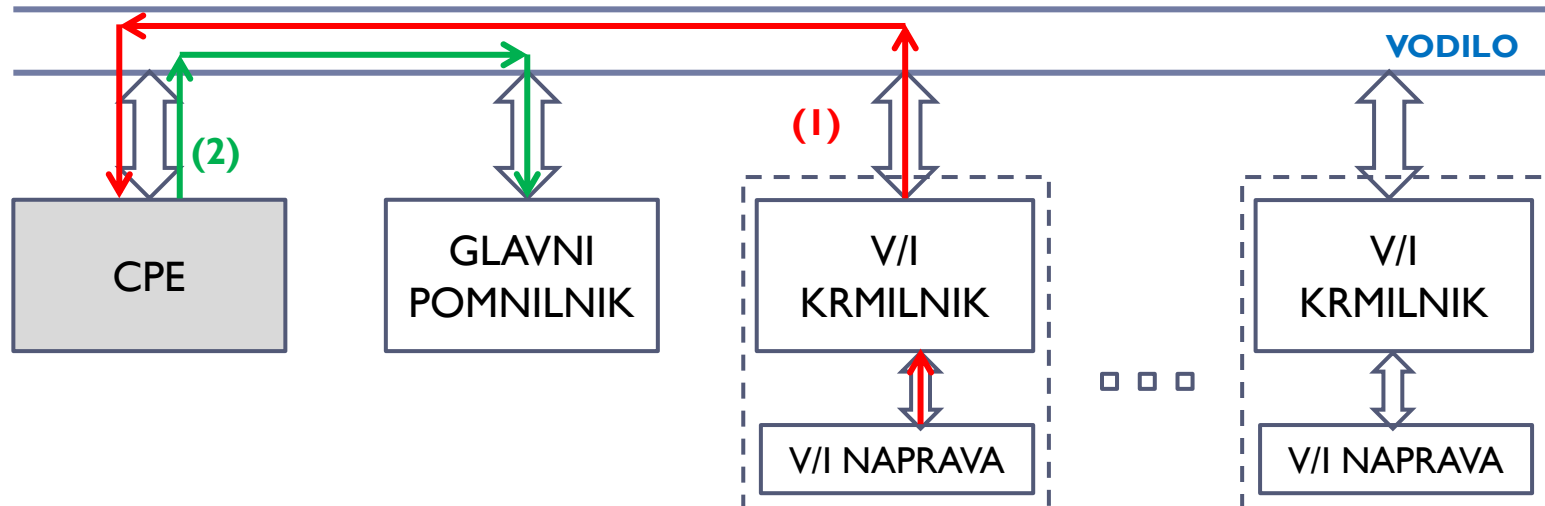
V/I naprava → CPE → glavni pomnilnik

(1) V/I enota ima shranjen podatek, ali ga pretvori iz ene oblike v drugo.

Preko V/I krmilnika se podatek pošlje na vodilo.

CPE sprejme podatek.

(2) CPE shrani podatek v glavni pomnilnik.



PISANJE v V/I napravo preko V/I krmilnika

- Podatek se iz glavnega pomnilnika preko CPE pošlje V/I napravi.

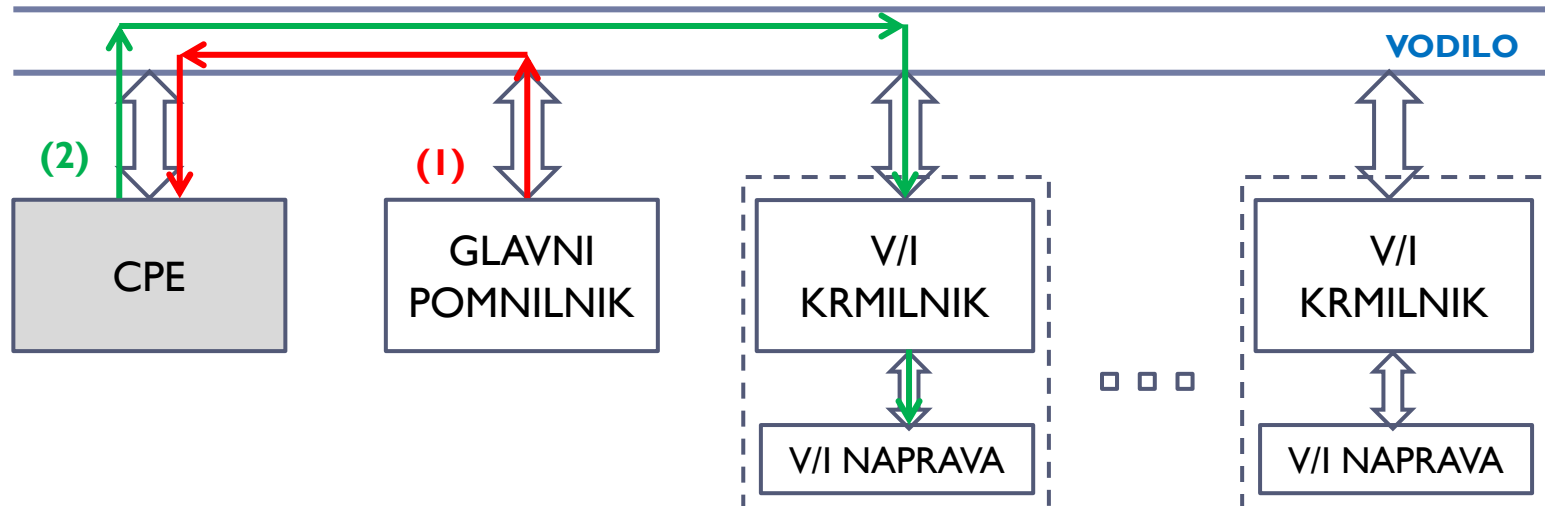
glavni pomnilnik → CPE → V/I naprava

(1) CPE prebere podatek iz glavnega pomnilnika.

(2) CPE pošlje podatek na vodilo.

V/I krmilnik sprejme podatek.

V/I naprava shrani podatek.



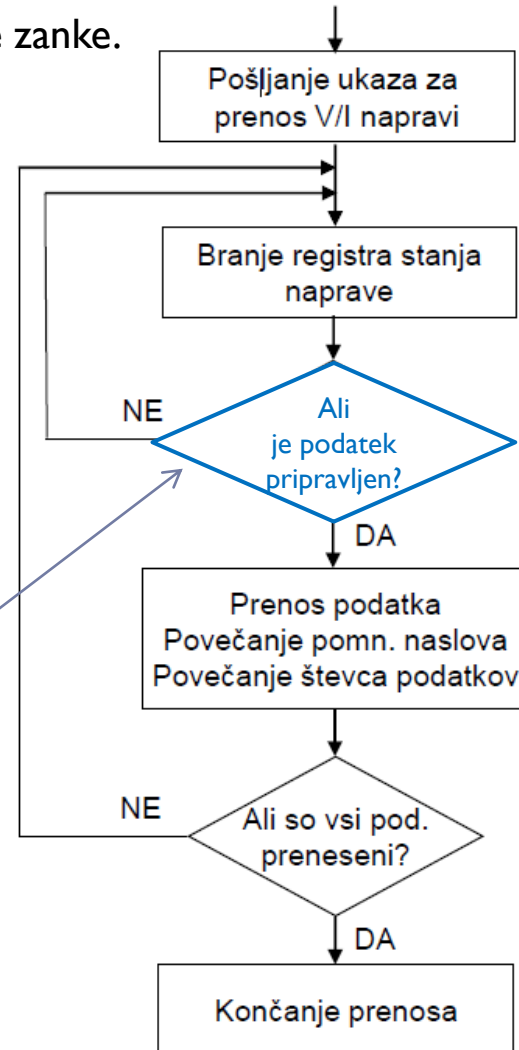
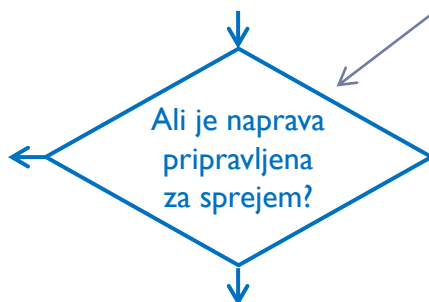
Ponazoritev programskega V/I prenosa (branje podatka iz V/I naprave)

❑ Prenos vsakega podatka zahteva obhod ene zanke.

❑ **Programsko izpraševanje (ang. pooling) -**
preverjanje pripravljenosti naprave

❑ Postopek branja preverja:
Ali je podatek pripravljen?

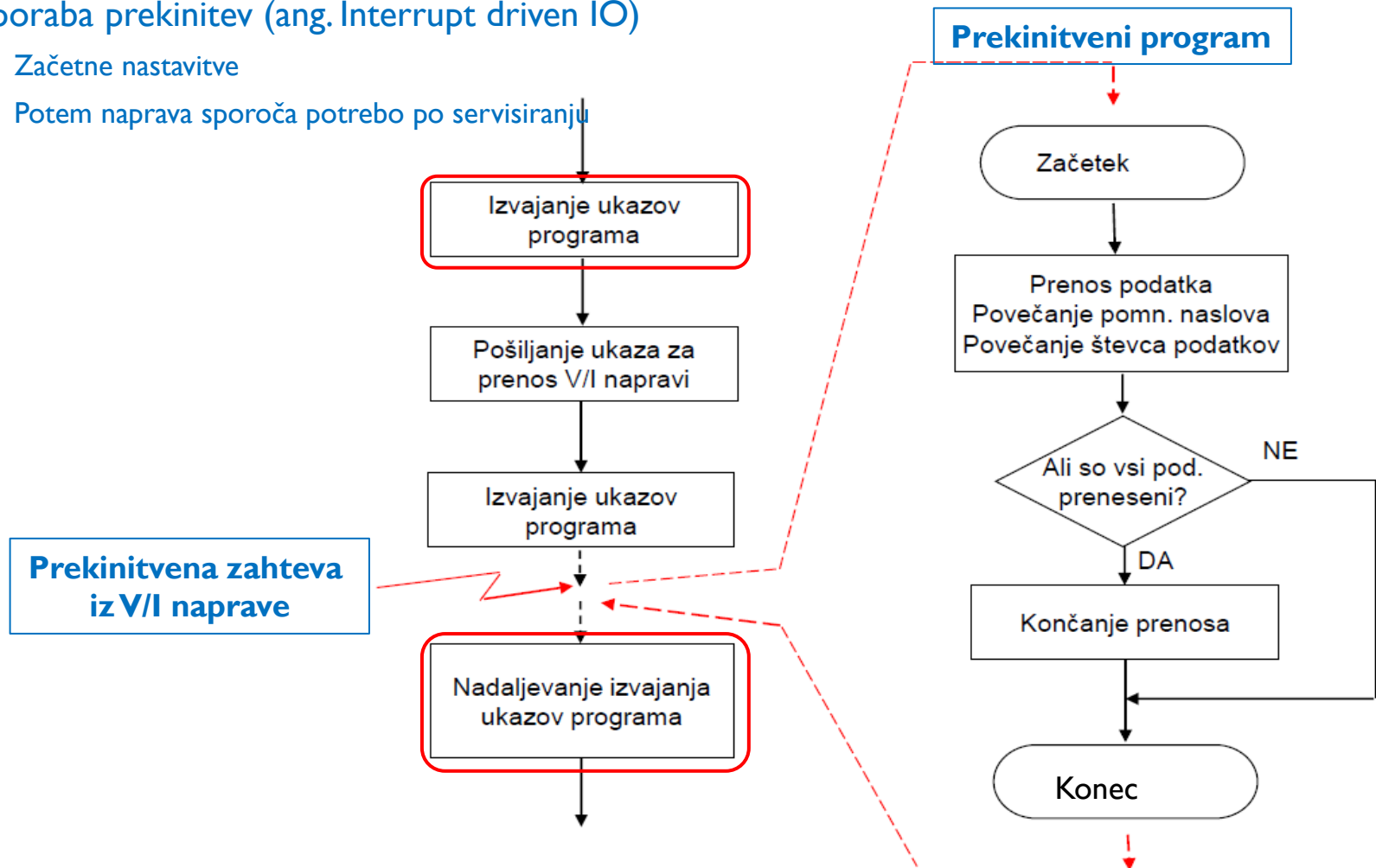
❑ **Postopek pisanja se od branja razlikuje**
samo v preverjanju:



Ponazoritev programskega V/I prenosa (branje podatka iz V/I naprave)

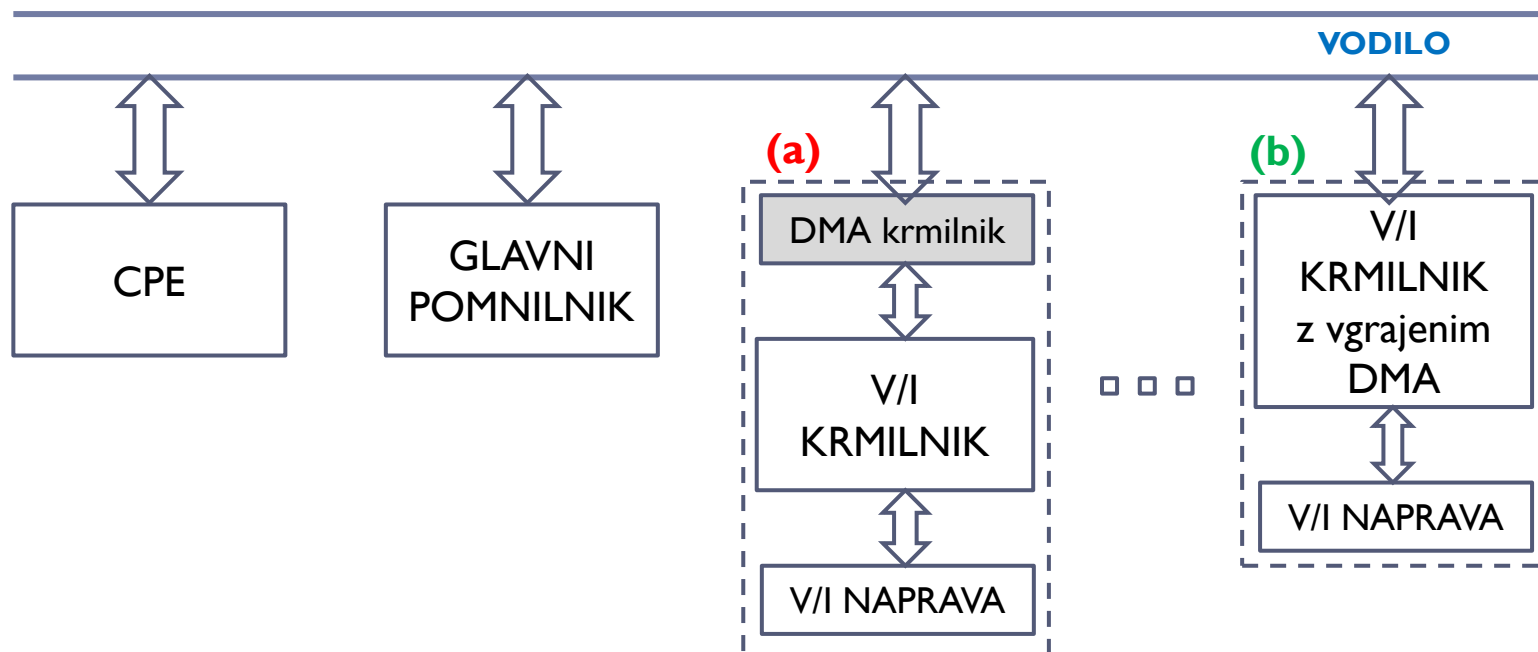
□ Uporaba prekinitev (ang. Interrupt driven IO)

- Začetne nastavitve
- Potem naprava sporoča potrebo po servisiranju



9.1.2 DMA prenos (ang. Direct Memory Access)

- ❑ DMA predstavlja aparaturno rešitev za **neposreden dostop do pomnilnika**.
- ❑ Izvedba prenosa:
 - (a) V/I naprava ima vključena dva krmilnika (V/I krmilnik in DMA krmilnik)
 - (b) V/I naprava ima vključen V/I krmilnik, ki ima vgrajen DMA krmilnik

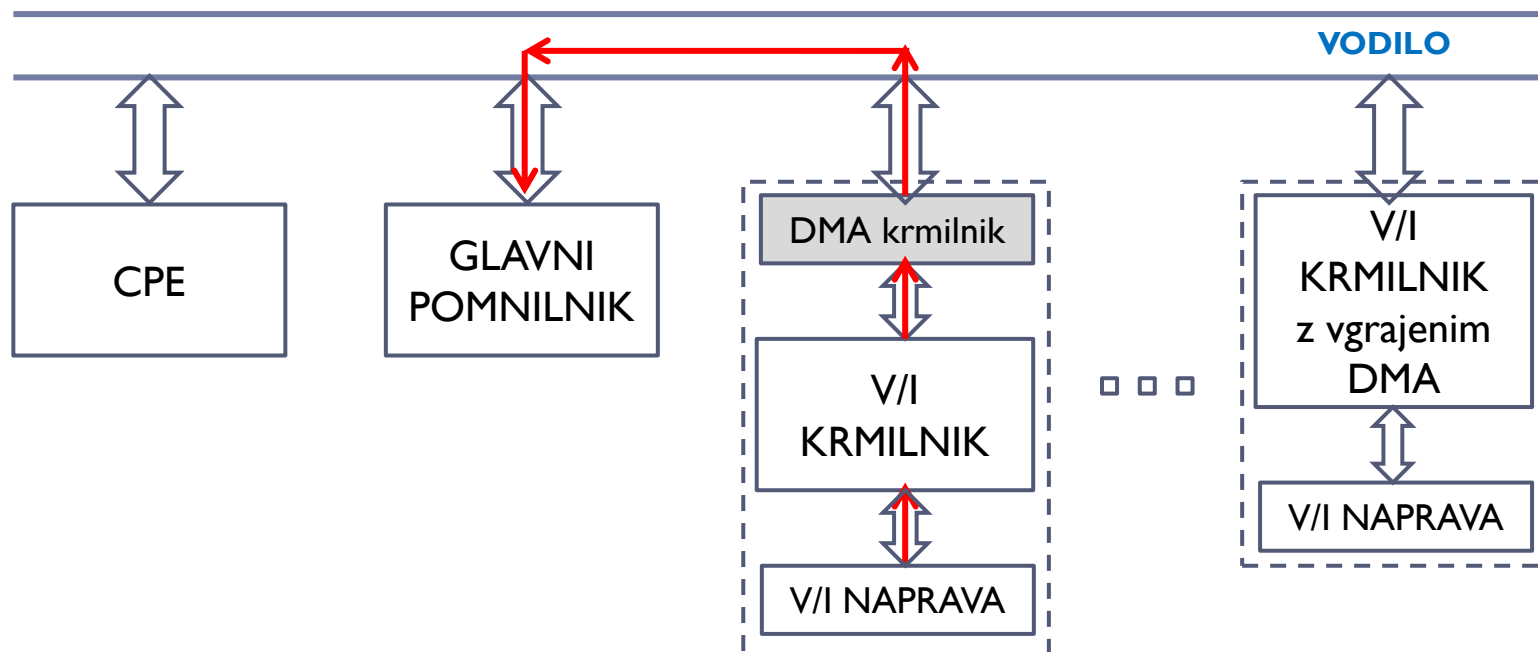


BRANJE iz V/I naprave z DMA krmilnikom

- Podatek se iz V/I naprave zapiše v glavni pomnilnik.

V/I naprava → glavni pomnilnik

- V/I enota ima shranjen podatek, ali ga pretvori iz ene oblike v drugo.
- Preko V/I krmilnika in DMA krmilnika se podatek pošlje na vodilo.
- Podatek se shrani v glavni pomnilnik.

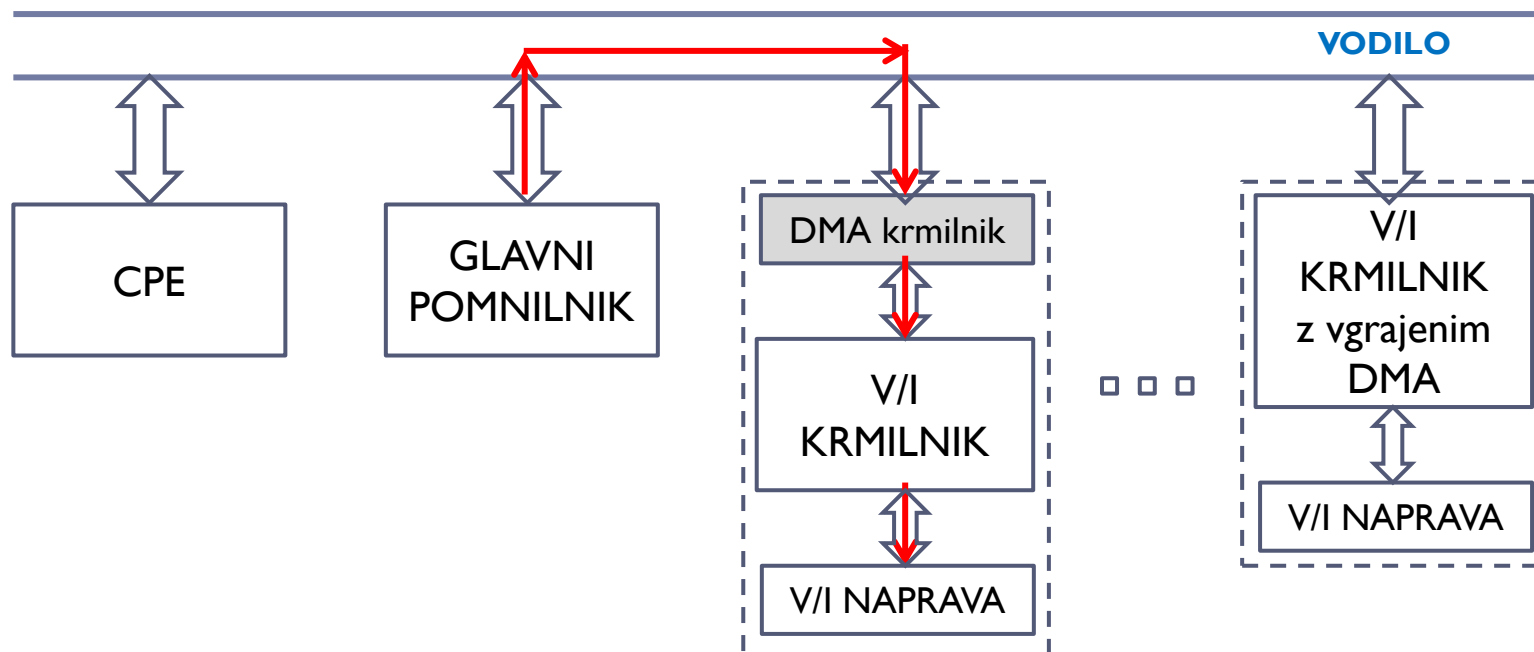


PISANJE v V/I napravo z DMA krmilnikom

- Podatek se iz glavnega pomnilnika pošlje V/I napravi.

glavni pomnilnik → V/I naprava

- Iz glavnega pomnilnika se pošlje podatek na vodilo.
- DMA krmilnik sprejme in posreduje podatek V/I krmilniku.
- V/I naprava shrani podatek.



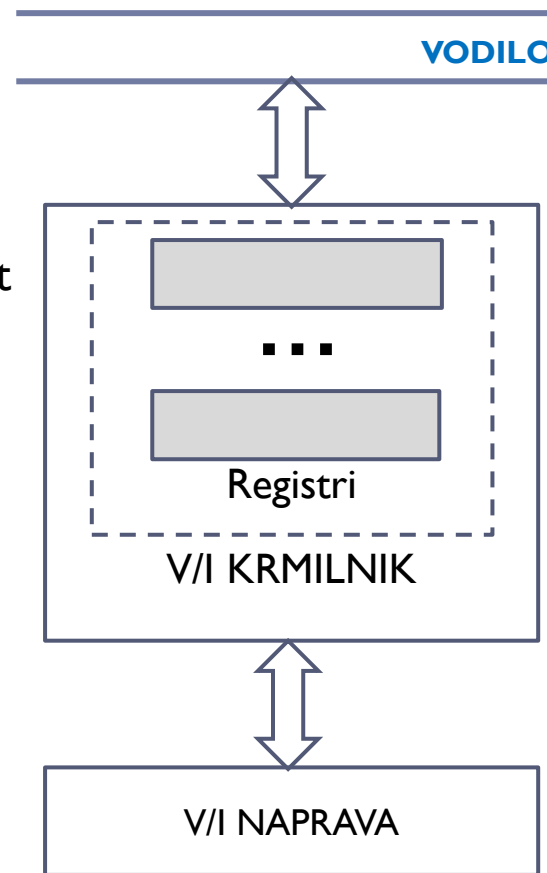
Povzetek DMA (Direct Memory Access)

- ❑ DMA omogoča **hiter prenos** velike količine podatkov.

- ❑ Podatki se med V/I napravo in pomnilnikom prenašajo **direktno, brez CPE**, če
 - obstaja povezava med krmilnikom V/I naprave in glavnim pomnilnikom in
 - obstaja krmilnik V/I naprav, ki ga imenujemo DMA krmilnik.
- ❑ DMA krmilnik:
 - tvori pomnilne naslove in kontrolne signale za dostop do pomnilnika,
 - šteje prenesene besede,
 - izvrši prenos, ko je podatek pripravljen,
 - obvesti CPE o zaključku V/I operacije (običajno s prekinitvijo).
- ❑ CPE izvede naslednje operacije:
 - v DMA krmilnik vpiše začetni naslov polja v glavnem pomnilniku.
 - v števeni register DMA krmilnika vpiše število besed, ki se prenesejo.
 - v DMA krmilnik vpiše vrednosti za način delovanja in operacije, ki se izvedejo.
 - v DMA krmilnik pošlje ukaz za začetek izvajanja prenosa.
 - preveri ali je bila operacija uspešno izvedena, ko DMA krmilnik pošlje potrditev.

9.1.3 Krmilnik V/I naprave

- ❑ Vsaka V/I naprava je priključena preko V/I krmilnika naprave.
- ❑ Gledano iz strani CPE je enostaven V/I krmilnik videti kot **določeno število namenskih registrov**.
- ❑ **Pisanje** v določen register (ukazni register) lahko sproži operacijo v V/I napravi (**ukaz napravi**).
- ❑ **Branje** iz določenega registra (statusni register) odraža stanje naprave po V/I operaciji (**status naprave**).
- ❑ **Prenos informacij** poteka z branjem iz določenega V/I registra ali s pisanjem v register V/I krmilnika (**podatkovni register**)
 - ❑ Krmilnik uravnava delovanje naprave glede na stanje bitov v registrih



A. Interni (enostaven) V/I krmilnik - registri so dosegljivi z ukazi na dva načina:

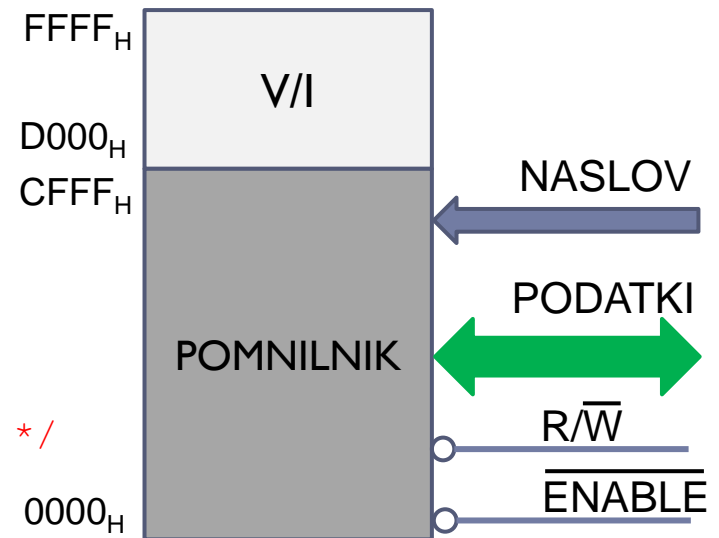
I. Pomnilniško preslikani vhod/izhod

- Registri krmilnika V/I so pomnilniške lokacije glavnega pomnilnika.
- Del pomnilniškega naslovnega prostora je uporabljen za V/I naprave.
- Uporabljajo se standardni ukazi, ki imajo enega od operandov v pomnilniku.
- Naslovi V/I registrov - fizični naslovni prostor, ki se ne sme preslikati v predpomnilnik.

• Primer: Procesor ARM

- Naslovni prostor:
 - Glavni pomnilnik: $0000_H - CFFF_H$
 - Pomnilnik V/I: $D000_H - FFFF_H$

```
.equ AIC_BASE, 0xFFFFF000 /*naslov AIC */  
.equ AIC_SMR17, 0x044 /*Odmiki registrov v AIC */  
.equ AIC_SVR17, 0x0C4  
.equ AIC_IECR, 0x120  
.equ AIC_EOICR, 0x130
```

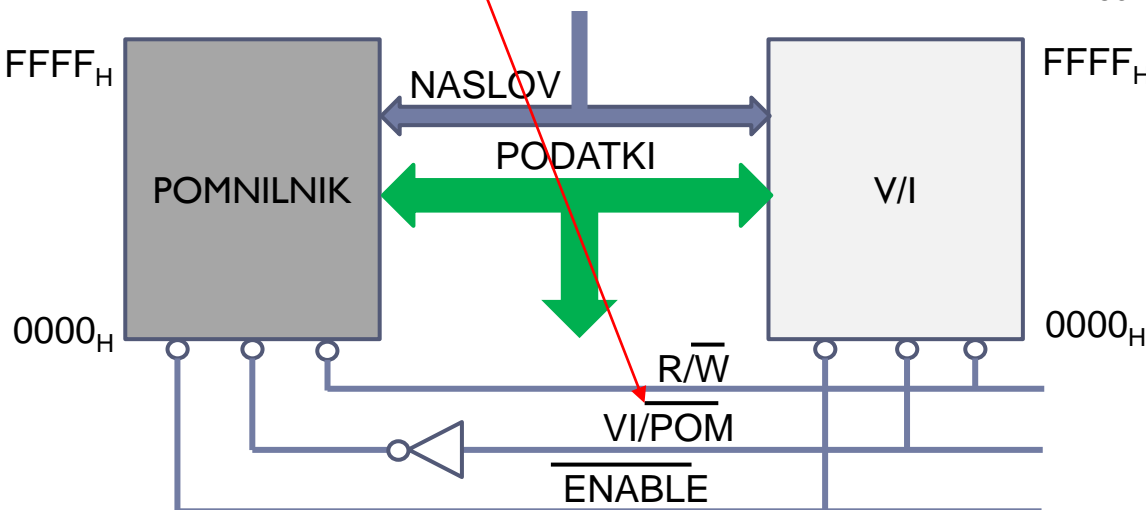


<https://www.slideshare.net/DilumBandara/I0-io-44759969>

II. Ločen vhodno/izhodni naslovni prostor

- Registri krmilnika V/I so pomnilniške lokacije v ločenem naslovnem prostoru.
- Uporabljajo se posebni V/I ukazi.
- CPE aktivira posebne signale za dostop do V/I naprav (Read/Write, izbira, omogočanje).
- Primer: Procesorji INTEL

IN Read from a port
OUT Write to a port
INS/INSB Input string from port/Input byte string from port
INS/INSW Input string from port/Input word string from port
INS/INSD Input string from port/Input doubleword string from port
OUTS/OUTSB Output string to port/Output byte string to port
OUTS/OUTSW Output string to port/Output word string to port
OUTS/OUTSD Output string to port/Output doubleword string to port



B. Zunanji (enostaven) V/I krmilnik - registri so dosegljivi s komunikacijo (npr. I2C, SPI, UART, ...):

III. Posredno upravljanje (naslavljanje) preko komunikacijskih poti

- Registri so še vedno lahko dosegljivi v posebnem naslovnem prostoru
- Tipičen potek komunikacije za dostop do registrov:
 - **Izbira** (naslavljanje naprave)
 - **Naslavljanje registra** in
 - **izvedba** operacije (branje, pisanje)

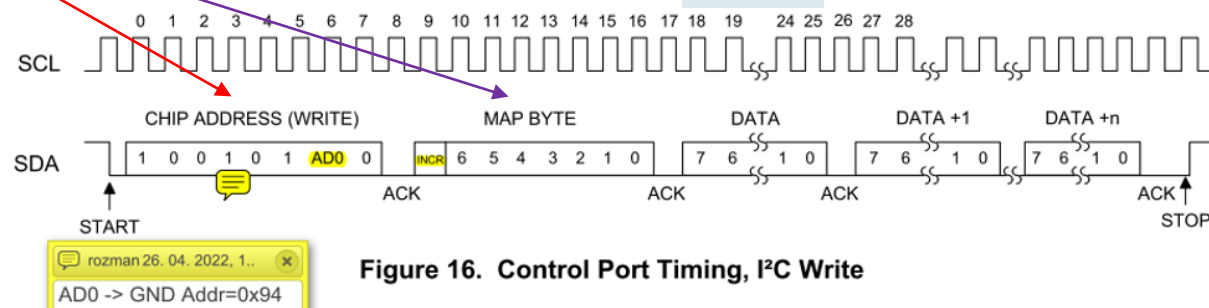
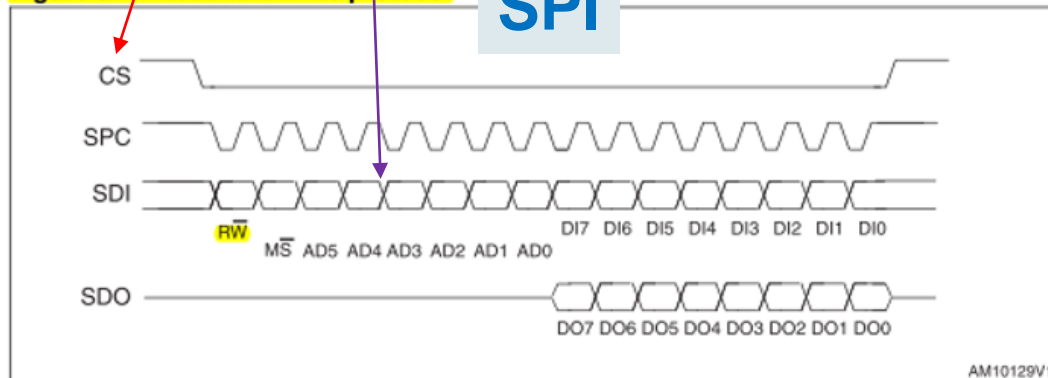


Figure 16. Control Port Timing, I²C Write

Figure 6. Read and write protocol

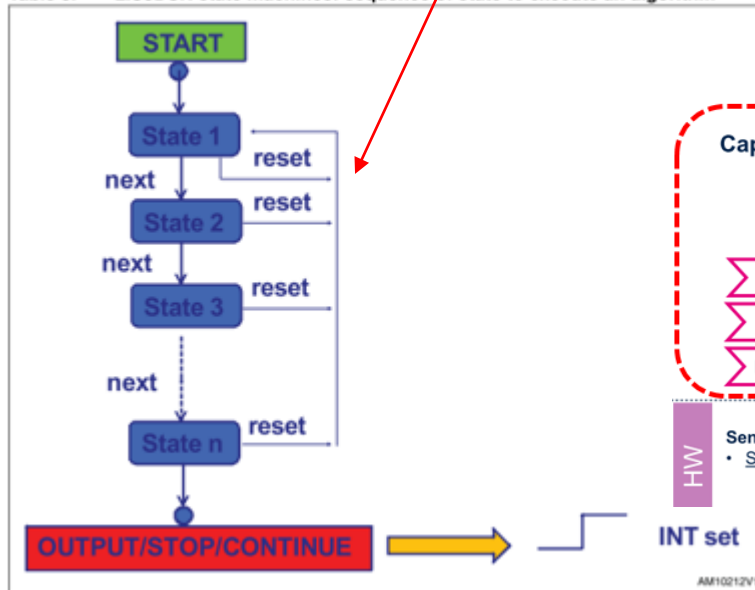


C. Naprednejši V/I krmilnik – posredni dostop do registrov ali višje nivojskih funkcij:

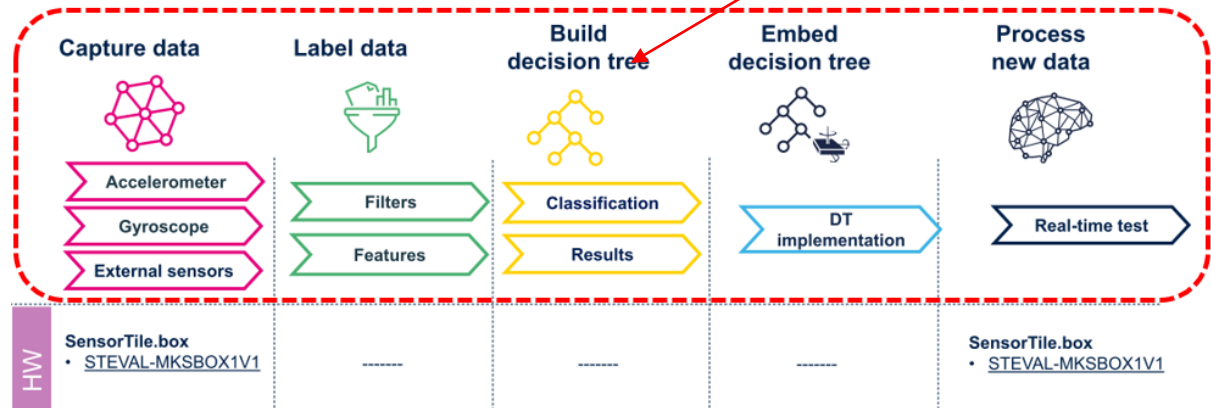
IV. Posredno upravljanje (naslavljanje) preko V/I procesorjev, mikrokrmilnikov

- Naslovi registrov krmilnikov so lahko v posebnem naslovnem prostoru, do katerega imajo dostop samo V/I procesorji, mikrokrmilniki
- Pogosto v novejših V/I napravah procesorji/krmilniki omogočajo tudi naprednejše, višjenivojske funkcije in lahko upravljamo napravo na tem nivoju.
 - Npr. končni avtomati, procesiranje podatkov, uporaba že naučenih modeli (odločitvena drevesa, plitke, globoke nevronske mreže in ostali koncepti s področja UI)

Table 8. LIS3DSH state machines: sequence of state to execute an algorithm



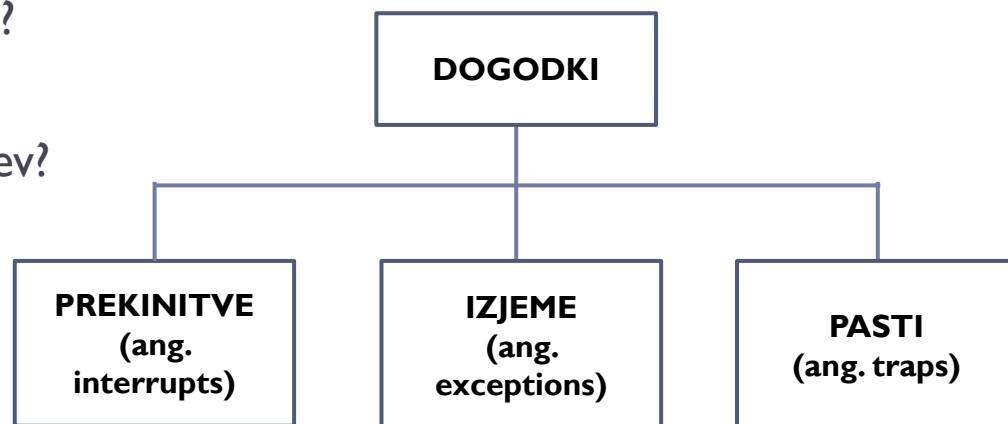
LSM6DSOX – SensorTile.box



2. Prekinitve, pasti in izjeme

□ **Prekinitev** (ang. **interrupt**) je zunanji dogodek, ki sproži PSP:

- Kaj je prekinitev (ang. interrupt)?
- Kdo sproži prekinitev?
- Kdaj se CPE odzove na prekinitev?
- Kako se izvede prekinitev?
- Kaj je prekinitveni program?
- Kaj je prioriteta prekinitev?
- Kakšne rešitve obstajajo zanjo?

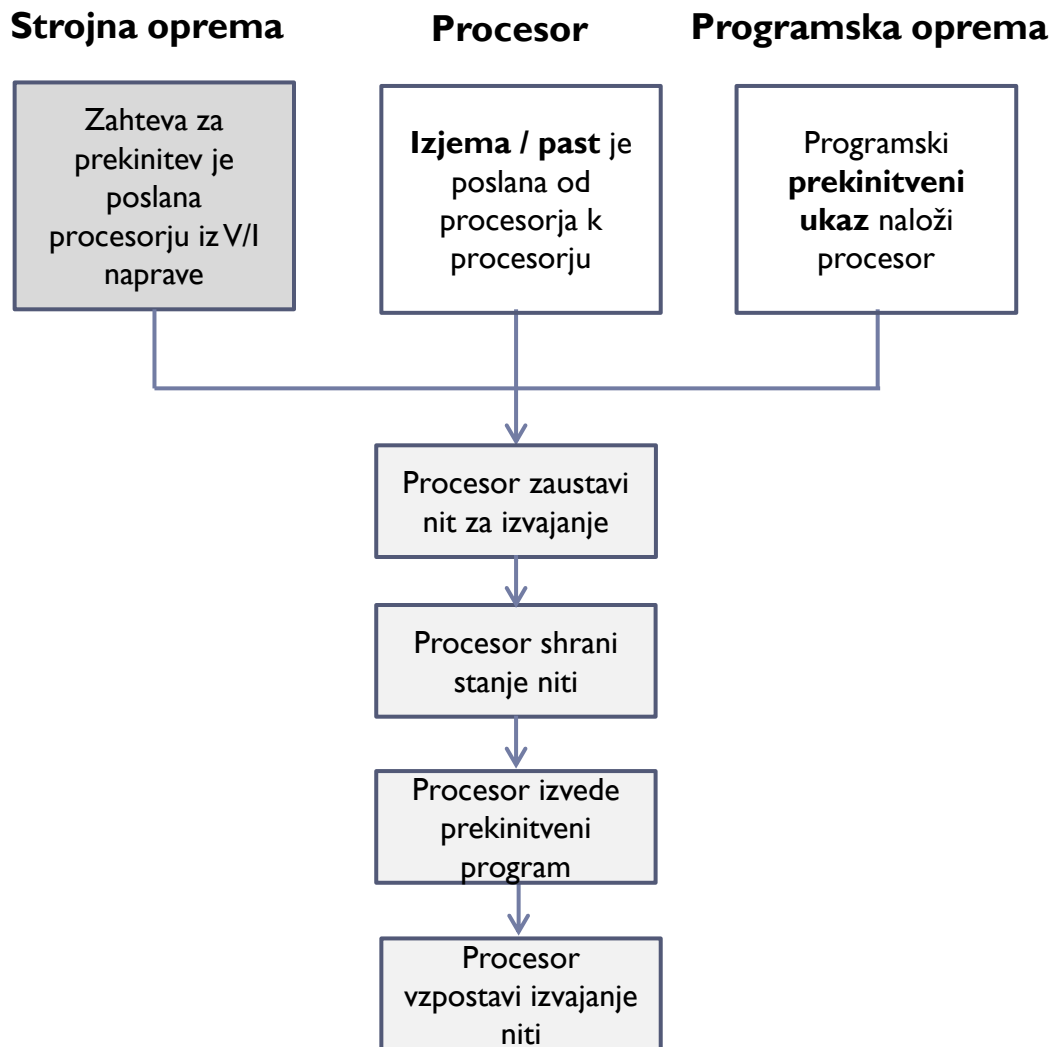


□ **Past** (ang. trap) je prekinitev, ki jo zahteva CPE ob posebnem dogodku ali na zahtevo programerja (ukazni dekoder zazna nepoznani ukaz, določena stran s podatki ni v fizičnem pomnilniku in CPE mora dostaviti podatke iz virtualnega pomnilnika, ...).

□ **Izjema** (ang. exception) – prekinitev, ki se avtomatsko generira v procesorju (aritmetična operacija – prekoračitev velikosti podatka ('overflow'), deljenje z 0, ...).

□ Prekinitev

- označuje **dogodek**, ki povzroči prenehanje izvajanja tekočega programa. Prične se izvajati drug program – **prekinitveni servisni program**.
- je **signal**, ki se pošlje procesorju, da se **prekine trenutni proces**. Lahko ga generira strojna naprava ali programska oprema.
- Potencialni izvori prekinitvev in izvedba (shema desno)



□ **Prioriteta** – pomemben je vrstni red obravnave zahtev

- Procesor ima več prekinitvenih vhodov in je nanj povezanih več V/I naprav.
- Prioriteta določa **vrstni red zahtev**, če je istočasno prisotnih več zahtev.
- Vgnezdene prekinitve – zahteva z višjo prioriteto lahko prekine izvedbo prekinitvenega programa naprave z nižjo prioriteto.
- Hitrejše V/I naprave (trdi disk) imajo višjo prioriteto kot počasnejše (tipkovnica).

- **Prekinitveni krmilnik** ima modul za določanje prioritete (**prioritetni kodirnik**) pri izvedbi prekinitvev, če ima CPE en sam bit za omogočanje prekinitvev..

- Primer: 4-bitni prioritetni kodirnik
 - D_0 - najnižja prioriteta
 - V – veljaven indikator, ki je enak 1, če je en ali več vhodov enakih 1.

VHODI				IZHODI		
D_0	D_1	D_2	D_3	Y_1	Y_0	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

Izvedba prekinitve

- ❑ S prekinitvijo je CPE obveščena o spremembah, ko V/I naprava:
 - zahteva povezavo s CPE za izvedbo operacije.
 - konča določeno operacijo.

- ❑ Prekinitiv je **asinhrona zahteva**, ki
 - ni povezana z nobenim ukazom.
 - ne prepreči izvedbe ukaza, ki se izvaja.

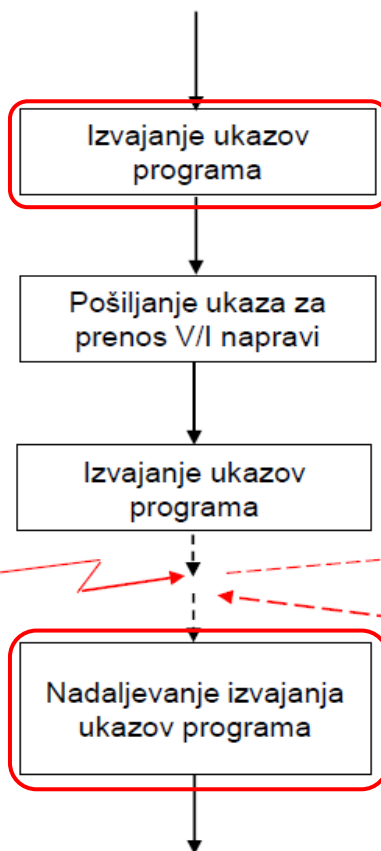
- ❑ CPE mora imeti **vgrajen prekinitveni mehanizem** za:
 - identifikacijo V/I naprave, ki je sprožila prekinitiv.
 - izvedbo prioritete V/I naprav.

❑ V/I naprava zahteva prekinitev

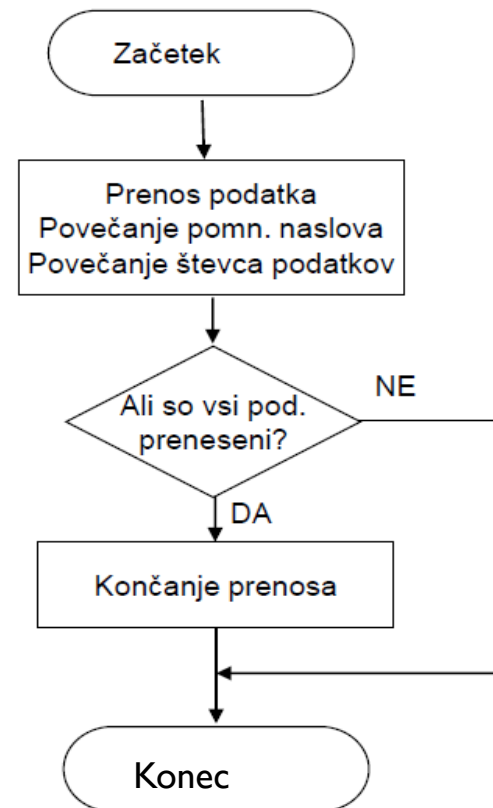
Branje iz V/I naprave poteka v naslednjih korakih:

- ❑ CPE pošlje napravi **ukaz za prenos**.
- ❑ **CPE nadaljuje** z izvajanjem programa.
- ❑ Ko je naprava pripravljena, pošlje v CPE **zahtevo za prekinitev**.
- ❑ Vključi se **prekinitveni program** za branje podatkov iz V/I naprave.
- ❑ **CPE nadaljuje** z izvajanjem programa.

Prekinitvena zahteva iz V/I naprave



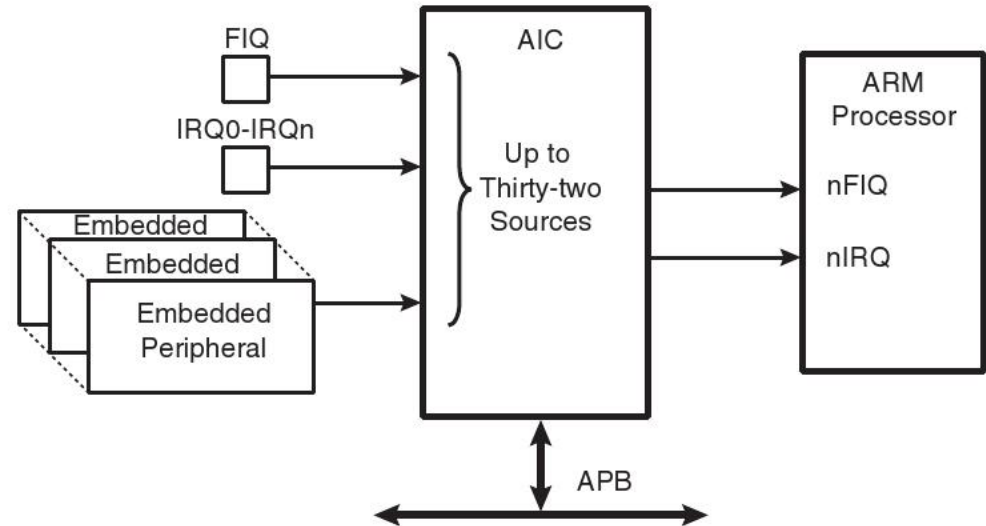
Prekinitveni program



FRI-SMS: Prekinitveni krmilnik AIC (Advanced Interrupt Controller)

- ▶ Enostavna vektorska tabela
- ▶ Specifični vektorji v AIC

Prekinitve v / Izjema	Način delovanja	Vektor
Reset	svc	0x00000000
Undefined Inst.	und	0x00000004
SWI	svc	0x00000008
Prefetch Abort	abt	0x0000000C
Data Abort	abt	0x00000010
Reserved		0x00000014
IRQ	irq	0x00000018
FIQ	fiq	0x0000001C



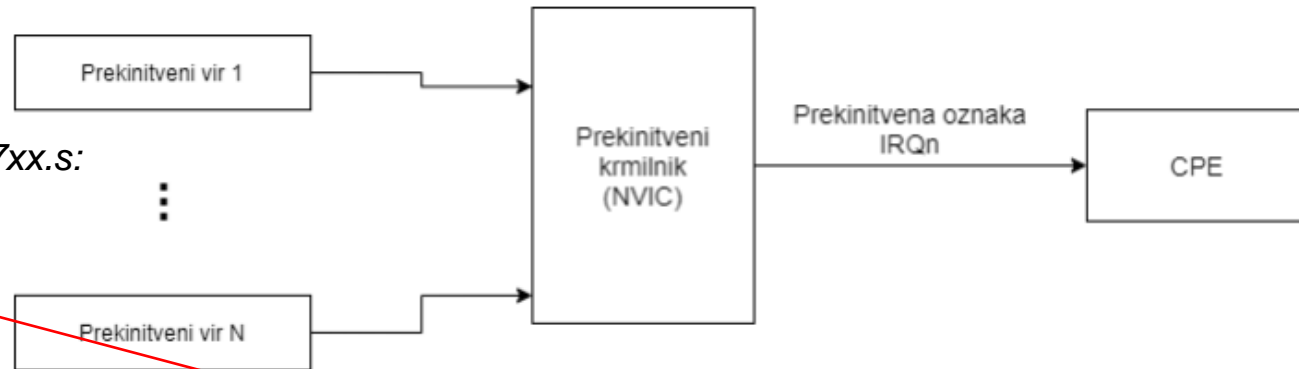
ARM: Prekinitveni krmilnik NVIC (Nested Vectored Interrupt Controller)

- ▶ Razširjena vektorska tabela
- ▶ Enostavnejša uporaba

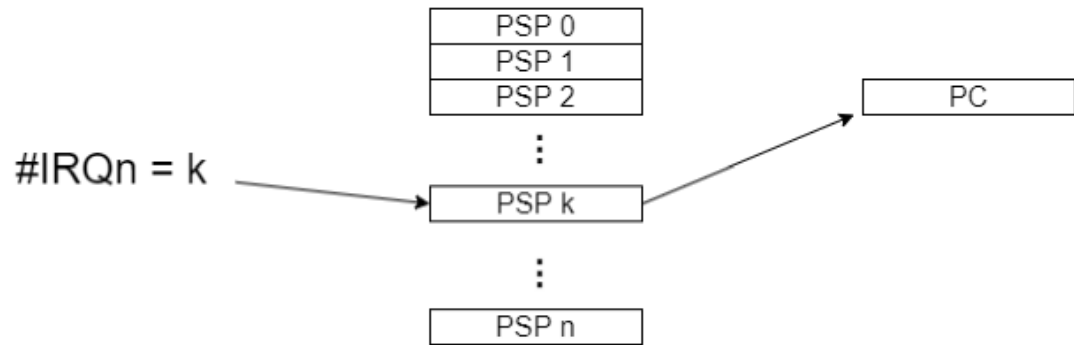
```
HAL_NVIC_SetPriority(EXTI0_IRQn, 1, 2);
HAL_NVIC_EnableIRQ(EXTI0_IRQn);
```

Core/Startup/startup_stm32f407xx.s:

```
g_pfnVectors:
.word _estack
.word Reset_Handler
.word NMI_Handler
.word HardFault_Handler
.word MemManage_Handler
.word BusFault_Handler
.word UsageFault_Handler
...
.word FLASH_IRQHandler
.word RCC_IRQHandler
.word EXTI0_IRQHandler
.word EXTI1_IRQHandler
.word EXTI2_IRQHandler
...
.word TIM3_IRQHandler
```



Vektorska tabela



Vir: ORS - Bulić

Table 143. NVIC⁽¹⁾

Signal	Priority	NVIC position	Acronym	Description	Address offset
-	-	-	-	Reserved	0x0000 0000
-	-3	-	Reset	Reset	0x0000 0004
-	-2	-	NMI	Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000 0008
-	-1	-	HardFault	All classes of fault	0x0000 000C
-	0	-	MemManage	Memory management	0x0000 0010
-	1	-	BusFault	Prefetch fault, memory access fault	0x0000 0014
-	2	-	UsageFault	Undefined instruction or illegal state	0x0000 0018
-	-	-	-	Reserved	0x0000 001C-0x0000 002B
-	3	-	SVCall	System service call via SWI instruction	0x0000 002C
-	4	-	DebugMonitor	Debug monitor	0x0000 0030
-	-	-	-	Reserved	0x0000 0034
-	5	-	PendSV	Pendable request for system service	0x0000 0038
-	6	-	SysTick	System tick timer	0x0000 003C
wwdg1_it	7	0	WWDG1	Window Watchdog interrupt	0x0000 0040
exti_pwr_pvd_wkup	8	1	PVD_PVM	PVD through EXTI line detection interrupt	0x0000 0044
exti_tamp_rtc_wkup	9	2	RTC_TAMP_STAMP_CSS_LSE	RTC tamper, timestamp	0x0000 0048
lsecss_rcc_it				CSS LSE	
exti_wkup_rtc_wkup	10	3	RTC_WKUP	RTC Wakeup interrupt through the EXTI line	0x0000 004C
flash_it	11	4	FLASH	Flash memory global interrupt	0x0000 0050
rcc_it	12	5	RCC	RCC global interrupt	0x0000 0054
exti_exti0_wkup	13	6	EXTI0	EXTI Line 0 interrupt	0x0000 0058
exti_exti1_wkup	14	7	EXTI1	EXTI Line 1 interrupt	0x0000 005C
exti_exti2_wkup	15	8	EXTI2	EXTI Line 2 interrupt	0x0000 0060

Table 143. NVIC⁽¹⁾ (continued)

Signal	Priority	NVIC position	Acronym	Description	Address offset
exti_exti3_wkup	16	9	EXTI3	EXTI Line 3 interrupt	0x0000 0064
exti_exti4_wkup	17	10	EXTI4	EXTI Line 4 interrupt	0x0000 0068
dma1_it0	18	11	DMA_STR0	DMA1 Stream0 global interrupt	0x0000 006C
dma1_it1	19	12	DMA_STR1	DMA1 Stream1 global interrupt	0x0000 0070
dma1_it2	20	13	DMA_STR2	DMA1 Stream2 global interrupt	0x0000 0074
dma1_it3	21	14	DMA_STR3	DMA1 Stream3 global interrupt	0x0000 0078
dma1_it4	22	15	DMA_STR4	DMA1 Stream4 global interrupt	0x0000 007C
dma1_it5	23	16	DMA_STR5	DMA1 Stream5 global interrupt	0x0000 0080
dma1_it6	24	17	DMA_STR6	DMA1 Stream6 global interrupt	0x0000 0084
adc1_it	25	18	ADC1_2	ADC1 and ADC2 global interrupt	0x0000 0088
adc2_it					
ttfdcan_intr0_it	26	19	FDCAN1_IT0	FDCAN1 Interrupt 0	0x0000 008C
fdcan_intr0_it	27	20	FDCAN2_IT0	FDCAN2 Interrupt 0	0x0000 0090
ttfdcan_intr1_it	28	21	FDCAN1_IT1	FDCAN1 Interrupt 1	0x0000 0094
fdcan_intr1_it	29	22	FDCAN2_IT1	FDCAN2 Interrupt 1	0x0000 0098
exti_exti5_wkup	30	23	EXTI9_5	EXTI Line[9:5] interrupts	0x 0000 009C
exti_exti6_wkup					
exti_exti7_wkup					
exti_exti8_wkup					
exti_exti9_wkup					
tim1_brk_it	31	24	TIM1_BRK	TIM1 break interrupt	0x0000 00A0
tim1_upd_it	32	25	TIM1_UP	TIM1 update interrupt	0x0000 00A4
tim1_trg_it	33	26	TIM1_TRG_COM	TIM1 trigger and commutation interrupts	0x0000 00A8
tim1_cc_it	34	27	TIM_CC	TIM1 capture / compare interrupt	0x0000 00AC
tim2_it	35	28	TIM2	TIM2 global interrupt	0x0000 00B0
tim3_it	36	29	TIM3	TIM3 global interrupt	0x0000 00B4
tim4_it	37	30	TIM4	TIM4 global interrupt	0x0000 00B8

STM32H(F) 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 */
```



3. Komunikacija računalnika z V/I napravami

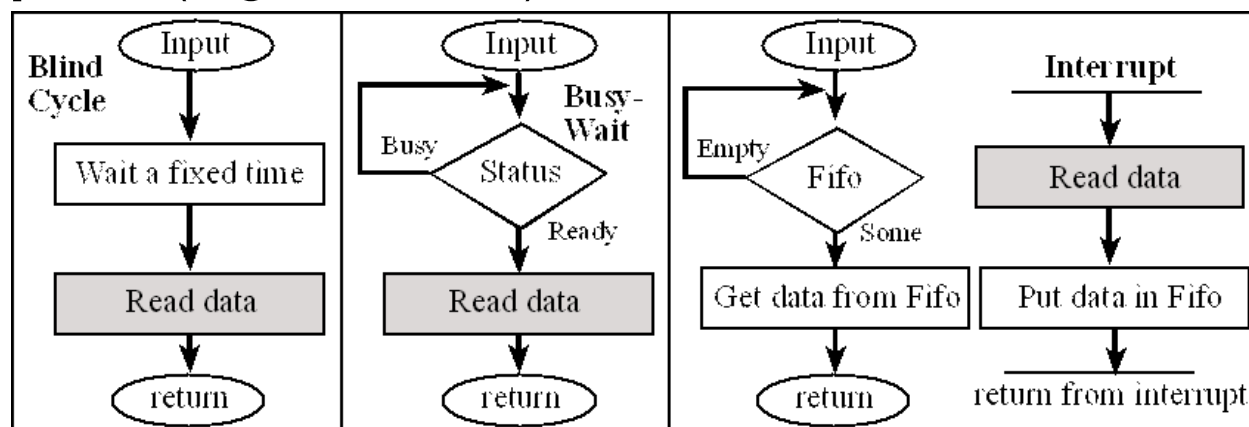
- ❑ Komunikacija med CPE in V/I napravami je običajno serijska

- ❑ Zmogljivost sistema (ang. performance) – mera za njegovo primernost, uporabnost
 - **Zakasnitev** (ang. latency) – čas med oddajo zahtevka in odzivom sistema.
 - **Pasovna širina** (ang. bandwidth) ali propustnost (ang. throughput) – št. prenesenih bitov/s.
 - **Prioriteta** (ang. priority) – vrstni red izvedbe, če je zaznanih več V/I zahtev.
 - **Sinhronizacijski mehanizmi** – strojna in programska oprema

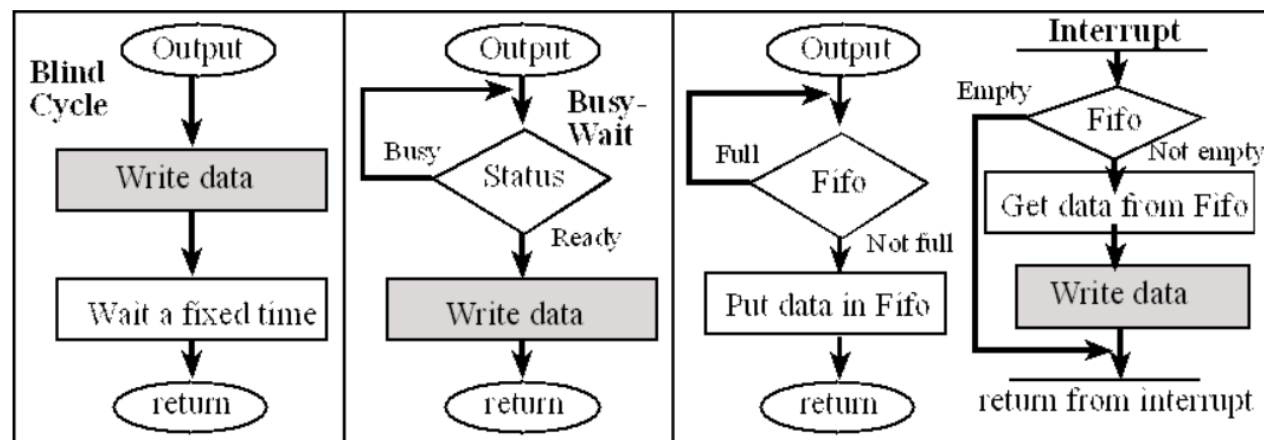
- ❑ Potek programske komunikacije (sinhronizacije) s strojno opremo:
 - Slep cikl (ang. blind cycle) – npr, vklop LCD zaslona
 - Programsko izpraševanje (ang. pooling) – npr. „čakam na zastavico“
 - Tudi Zasedeno-čakaj (ang. busy-wait) – npr. „is device busy“ ?
 - Prekinitev (ang. interrupt)
 - Neposreden dostop do pomnilnika (ang. DMA – Direct Memory Access)

- ❑ Vir: http://users.ece.utexas.edu/~valvano/VolumeI/E-Book/CI I _SerialInterface.htm

❑ Branje iz V/I naprave (ang. Read data)



❑ Pisanje na V/I napravo (ang. Write data):

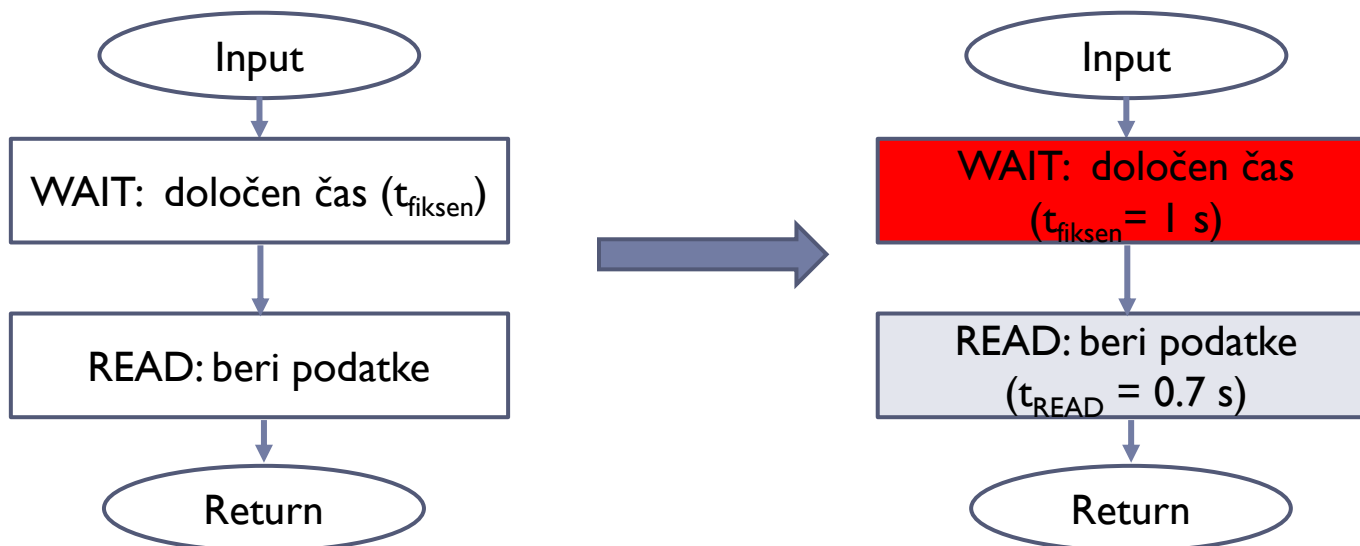


Branje iz V/I naprave

I. Slepi cikel (ang. 'Blind Cycle'): Programska oprema za vhodno napravo:

- sproži (zažene) zunanjo vhodno strojno opremo,
 - počaka določen čas in bere podatke iz naprave.
- ❑ Programska oprema počaka določen čas in predvideva, da se bo vhodni/izhodni postopek končal, preden poteče ta določen čas.
 - ❑ Uporaba: za izvedbo operacije branja je potreben kratek in predvidljiv čas izvedbe.
 - ❑ Program:

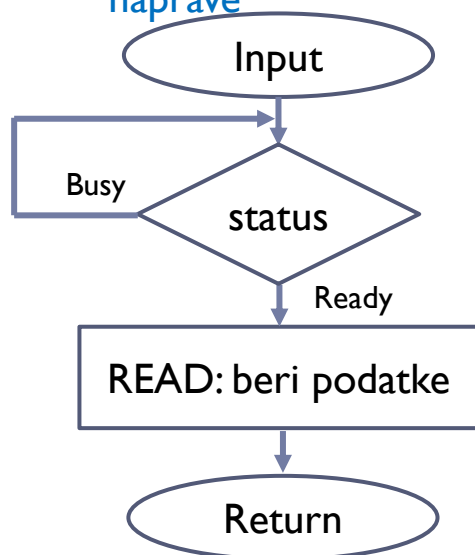
Izvedba: Branje podatka iz V/I naprave v vnaprej določenem času



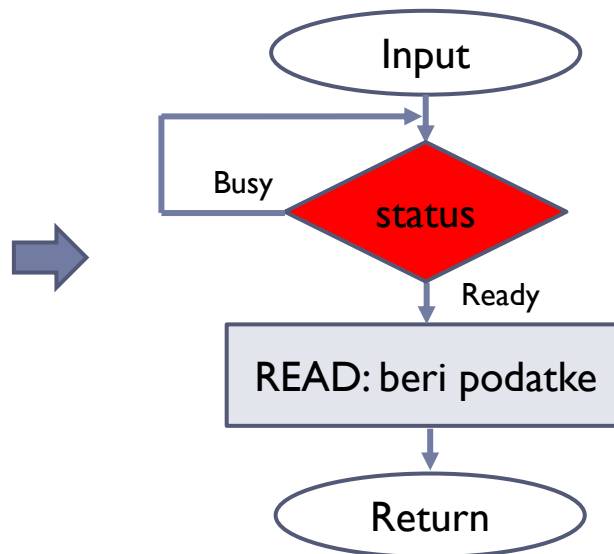
II. Spraševanje: Zasedeno-čakaj (ang. 'Busy-Wait')

- programska zanka, ki preverja stanje V/I naprave, čaka na zaključeno stanje.
 - programska oprema za vhodno napravo počaka, da ta dobi nove podatke in jih nato prebere z vhodne naprave.
- Uporaba: Sinhronizacija je primerna v situacijah, ko je programski sistem razmeroma preprost in odziv v realnem času ni pomemben.

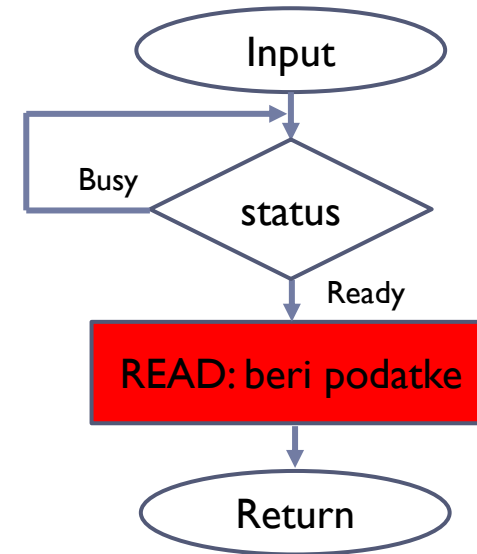
▪ Program:
naprave



Izvedba: Preverjanje zasedenosti

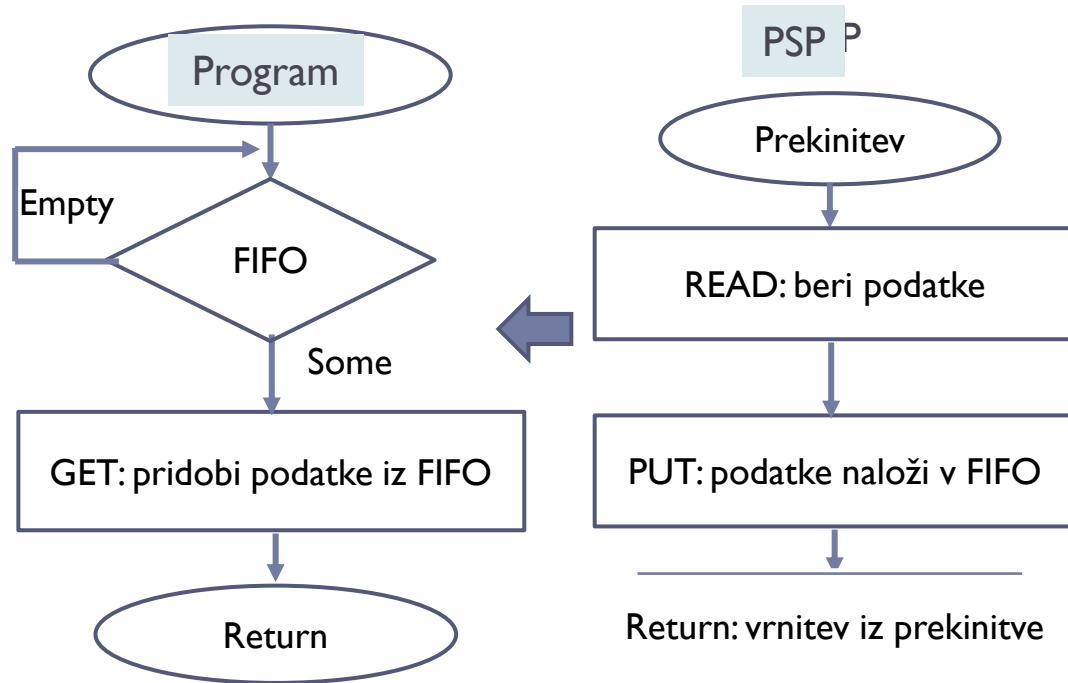


Branje podatka iz V/I



III. Prekinitev (ang. Interrupt):

- Strojna oprema **vhodne naprave običajno zahteva prekinitev ob dogodku**
 - npr. ko ima nove podatke v FIFO.
- Strojna oprema povzroči **izvajanje posebne programske opreme** (prekinitveni servisni program - PSP).
- PSP običajno prebere z vhodne naprave in shrani v RAM pomnilnik (npr. FIFO vrsta).



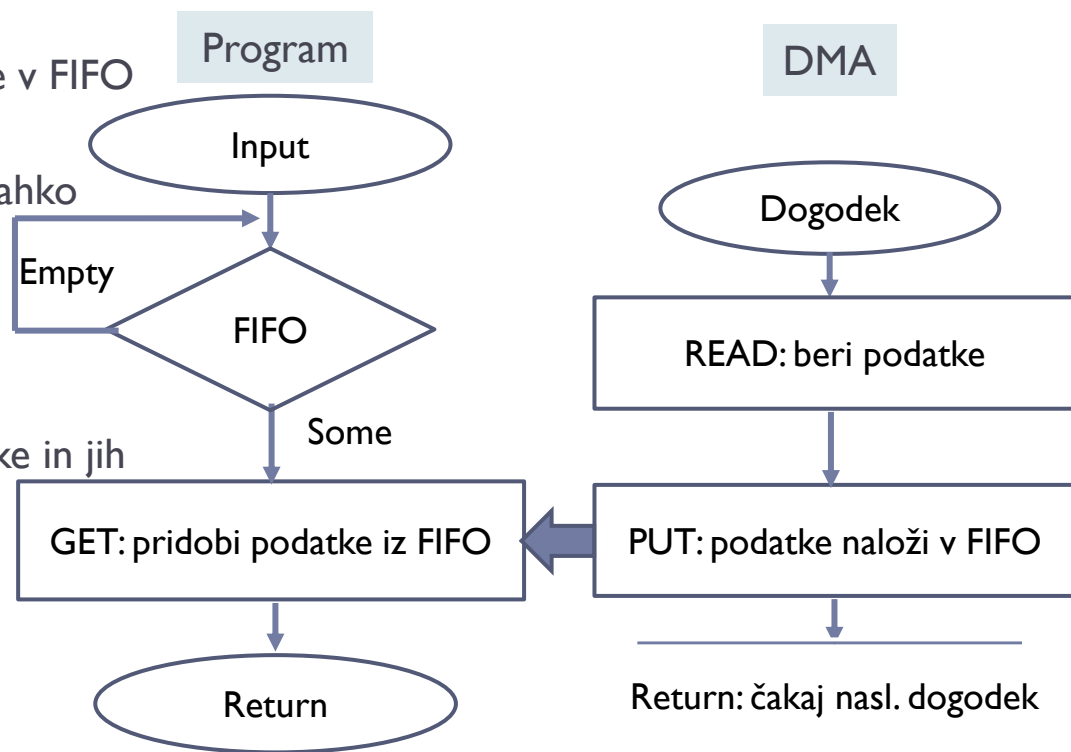
IV. DMA prenos (ang. Direct Memory Access):

▪ DMA krmilnik:

- neodvisno bere podatke in jih shranjuje v FIFO vrsto v RAM pomnilniku.
- po koncu prenosa nastavi zastavico in lahko sproži zahtevo po prekinitvi.

▪ Program:

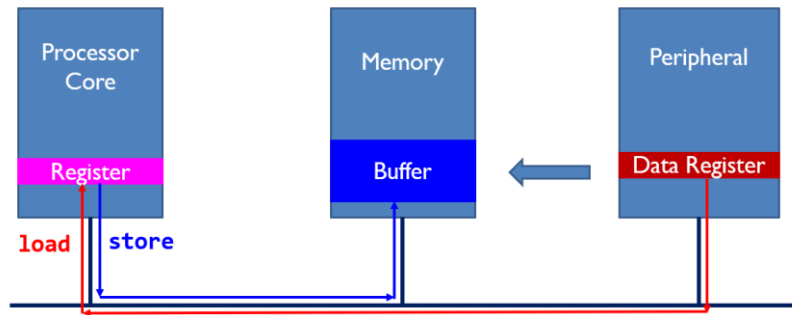
- Kadarkoli lahko preveri sprejete podatke in jih uporabi.



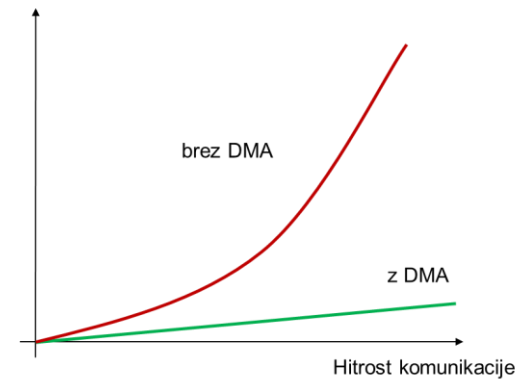
DMA- Splošno

Primer 1: CPE opravlja prenose V/I <-> pomnilnik

- Čaka na zastavico
- ldr Rx, ... in str Rx, ...

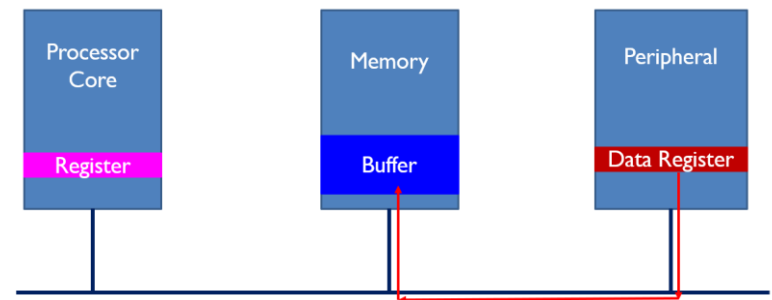


CPE zasedenost



Primer 2: DMA opravlja prenose V/I <-> pomnilnik :

- **CPE nastavi DMA krmilnik** za prenos :
 - Vrsta prenosa, naprave
 - Naslovi vira in ponora
 - Velikost in število podatkov
 - Sproži prenos
- **DMA krmilnik** (neodvisno od CPE):
 - Čaka na zastavico
 - Prebere in shrani podatek
 - odšteva preostale podatke in zaključi prenos
- **CPE izvaja svoj program**



DMA- Splošno

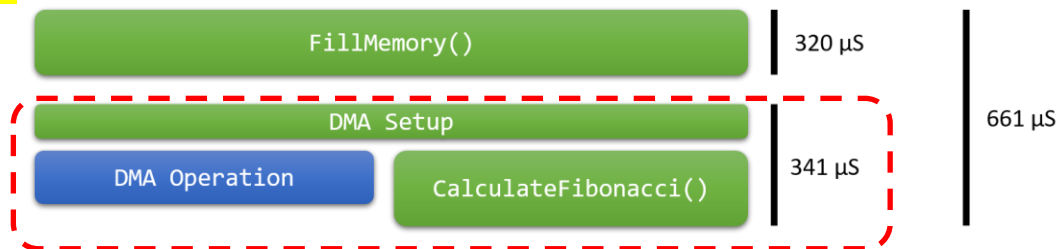
Ponovitev OR-LAB 10: USART+DMA

Case1: In this example, filling the first and the third buffer took the exactly the same time, while copying the first buffer to the second one took slightly less time:



While the DMA cannot be used to compute Fibonacci numbers, or initialize arrays with non-constant values, it can be used for copying data between 2 memory locations.

Now the DMA operation ran in parallel with the CalculateFibonacci() function, **reducing the overall program time by 21%:**

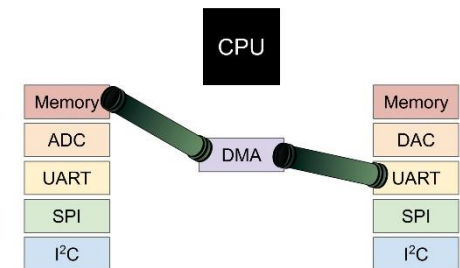


Case2: CRC calculation case:

The results are in:

80 DMA CRCs per second.
63 manual CRCs per second

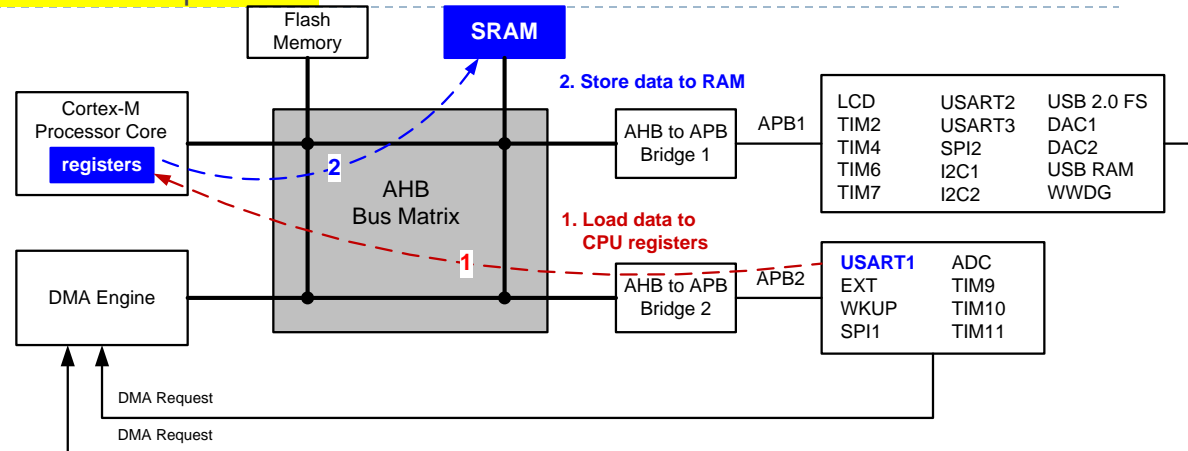
On my processor, **DMA gives a 27% advantage over iterative memory assignment.** I think it is because everything is done with a hardware mover that doesn't have to increment, involve registers, gotos, branch less than, and so on.



DMA- STM32H7 (stikalna matrika)

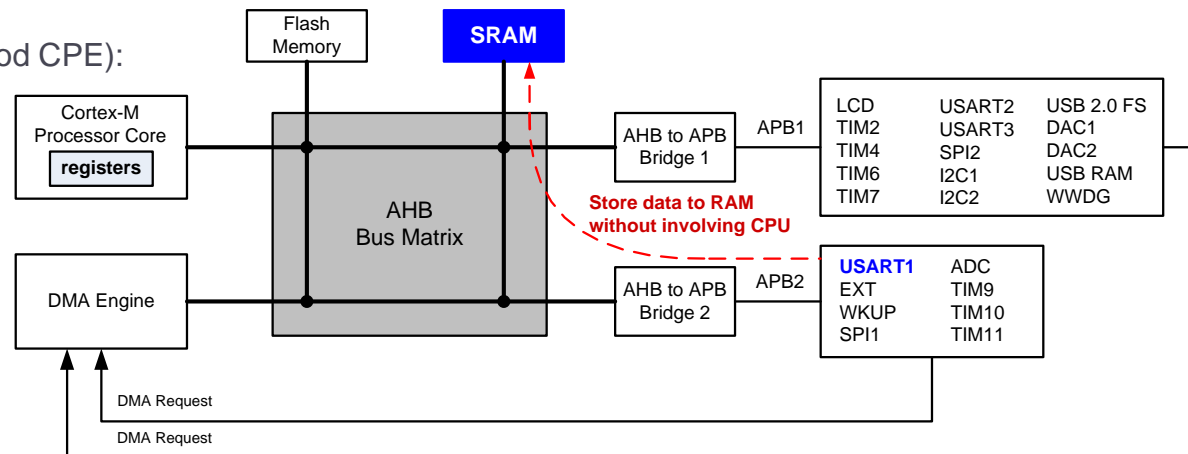
Primer 1 (**brez DMA**): CPE opravlja prenose V/I <-> pomnilnik

- Čaka na zastavico
- Idr Rx, ... in str Rx, ...



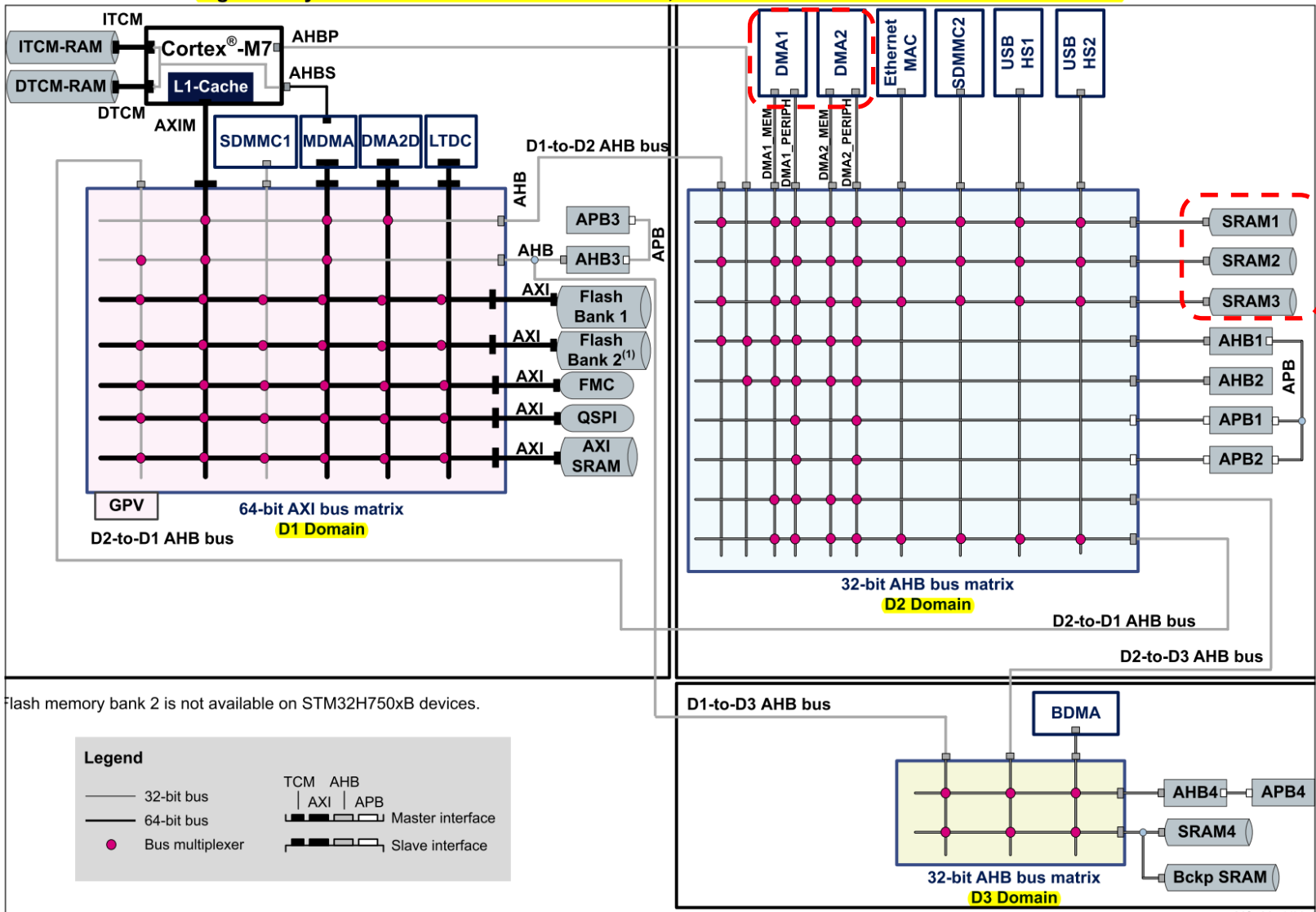
Primer 2 (**z DMA**): DMA opravlja prenose V/I <-> pomnilnik :

- CPE nastavi DMA krmilnik za prenos :
- DMA krmilnik (neodvisno od CPE):
- CPE izvaja svoj program



DMA - STM32H7

Figure 1. System architecture for STM32H742xx, STM32H743/53xx and STM32H750xB devices



Memory and bus architecture

RM0433

104/3319

RM0433 Rev 7



Primer 1: STM32 – ADC na 3 načine

❑ Programsko spraševanje (angl. Software Polling)

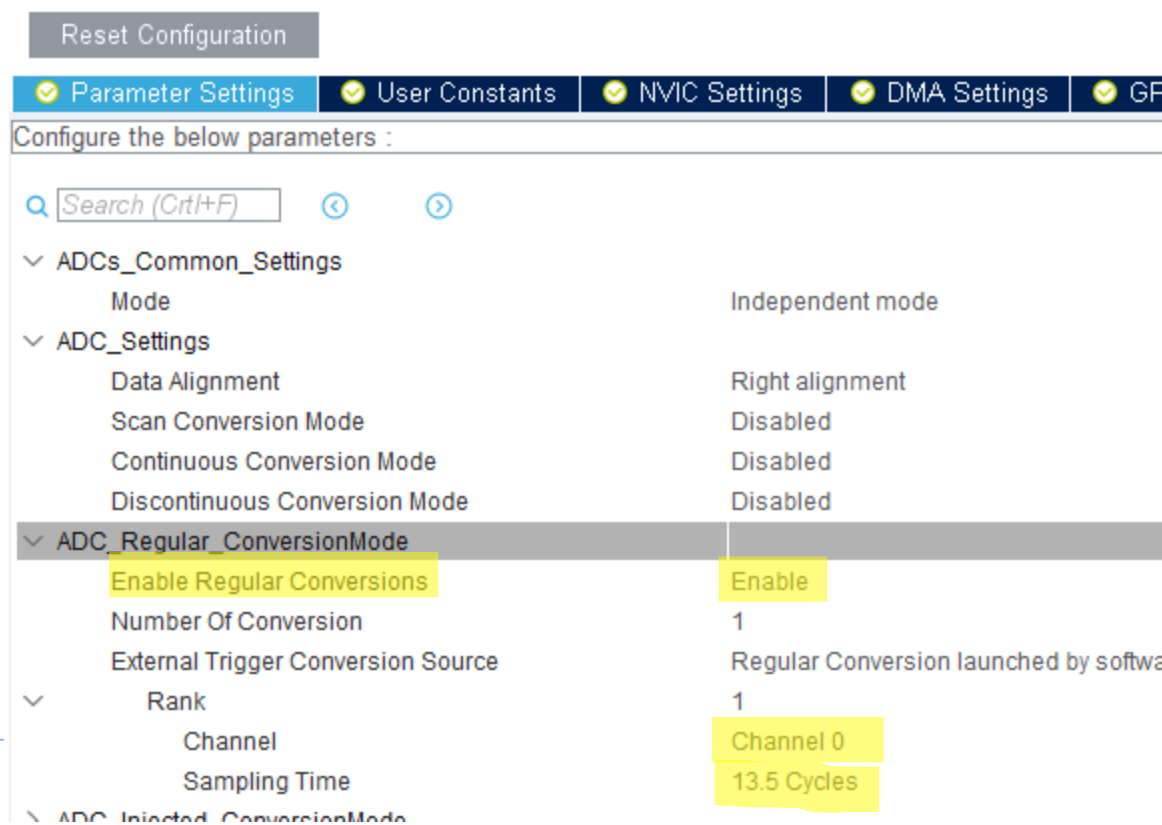
```
HAL_ADC_Start(&hadc1); // start the adc
```

```
    HAL_ADC_PollForConversion(&hadc1, 100); // poll for conversion
```

```
    adc_val = HAL_ADC_GetValue(&hadc1); // get the adc value
```

```
HAL_ADC_Stop(&hadc1); // stop adc
```

```
HAL_Delay (500); // wait for 500ms
```



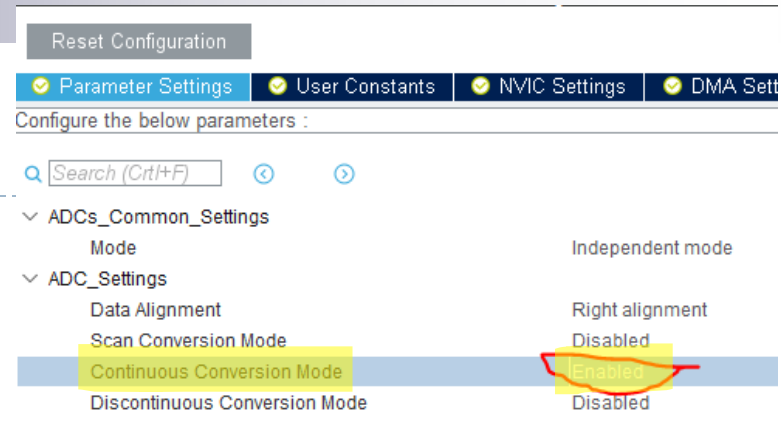
The screenshot shows the STM32CubeMX configuration tool interface. At the top, there is a 'Reset Configuration' button. Below it, several tabs are visible: 'Parameter Settings' (checked), 'User Constants' (checked), 'NVIC Settings' (checked), 'DMA Settings' (checked), and 'GPIO' (checked). The main area is titled 'Configure the below parameters :'. It features a search bar with the text 'Search (Ctrl+F)' and two navigation arrows. The configuration tree is expanded to show 'ADCs_Common_Settings' and 'ADC_Settings'. Under 'ADC_Settings', the following parameters are listed: 'Data Alignment' (Right alignment), 'Scan Conversion Mode' (Disabled), 'Continuous Conversion Mode' (Disabled), and 'Discontinuous Conversion Mode' (Disabled). The 'ADC_Regular_ConversionMode' section is highlighted in grey and contains: 'Enable Regular Conversions' (Enable), 'Number Of Conversion' (1), and 'External Trigger Conversion Source' (Regular Conversion launched by software). Below this, the 'Rank' section is expanded to show 'Channel' (Channel 0) and 'Sampling Time' (13.5 Cycles). The 'ADC_Injected_ConversionMode' section is partially visible at the bottom.

Primer 1: STM32 – ADC na 3 načine

❑ Prekinitve (angl. Interrupts)

`HAL_ADC_Start_IT (&hadc1);`

```
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc) {  
    adc_val = HAL_ADC_GetValue(&hadc1);  
    /*If continuousconversion mode is DISABLED uncomment below*/  
    //HAL_ADC_Start_IT (&hadc1);  
}
```



NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
ADC1 and ADC2 global interrupts	<input checked="" type="checkbox"/>	0	0

Primer 1: STM32 – ADC na 3 načine

❑ DMA (angl. Direct Memory Access)

```
HAL_ADC_Start_DMA (&hadc1, &buffer, 1);
```

```
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc) {  
    adc_val = buffer;  
}
```

Add Delete

DMA Request Settings

Mode	Circular	Increment Address	<input type="checkbox"/>	Peripheral	<input type="checkbox"/>	Memory	<input checked="" type="checkbox"/>
Data Width	Word						

Configuration

Reset Configuration

Parameter Settings User Constants NVIC Settings DMA Settings GPIO Settings

DMA Request	Channel	Direction	Priority
ADC1	DMA1 Channel 1	Peripheral To Memory	Low

Primer 2: STM32 – UART na 3 načine

Programsko spraševanje (angl. Software Polling)

```
uint8_t Rx_data[10]; // creating a buffer of 10 bytes
```

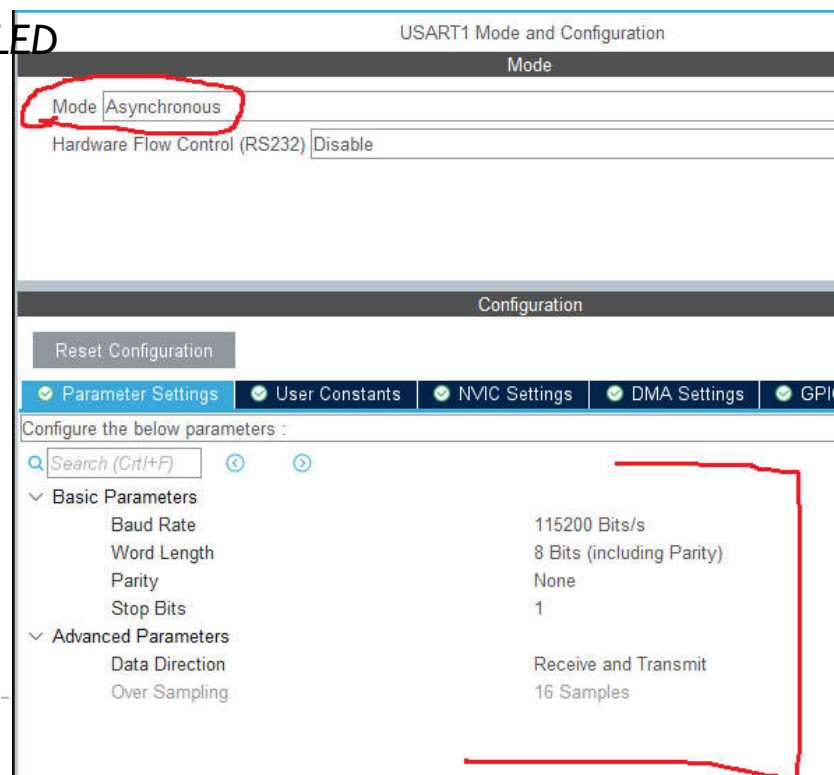
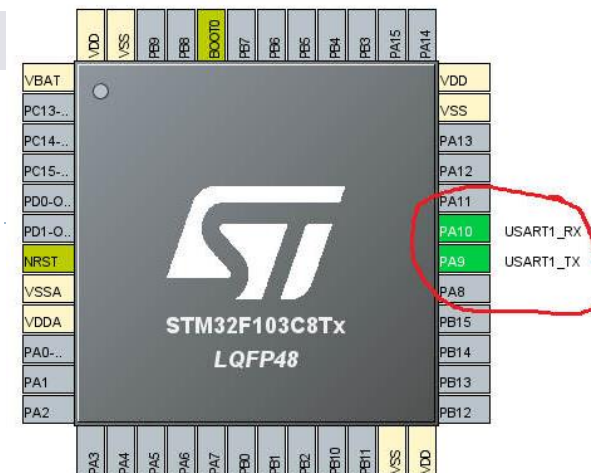
```
while (1) {
```

```
    HAL_UART_Receive (&huart2, Rx_data, 4, 100); // receive 4 bytes of data
```

```
    HAL_GPIO_TogglePin (GPIOA, GPIO_PIN_5); // toggle LED
```

```
    HAL_Delay (250);
```

```
}
```



Primer 2: STM32 – UART na 3 načine

❑ Prekinitive (angl. Interrupts)

....

```
uint8_t Rx_data[10]; // creating a buffer of 10 bytes
```

```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef  
*huart)
```

```
{
```

```
HAL_UART_Receive_IT (&huart2, Rx_data, 4);
```

```
while (1)
```

```
{
```

```
HAL_GPIO_TogglePin (GPIOA, GPIO_PIN_5);
```

```
HAL_Delay (250);
```

```
}
```

USART1 Mode and Configuration

Mode

Mode

Hardware Flow Control (RS232)

Configuration

Reset Configuration

Parameter Settings User Constants NVIC Settings DMA Settings GPIO Settings

NVIC Interrupt Table

Enabled

Preemption Priority

Sub Priority

USART1 global interrupt

0

0

Primer 2: STM32 – UART na 3 načine

❑ DMA (angl. Direct Memory Access)

```
....  
uint8_t Rx_data[10]; // creating a buffer of 10 bytes  
void HAL_UART_RxHalfCpltCallback(UART_HandleTypeDef  
*huart) {  
    HAL_GPIO_TogglePin (GPIOA, GPIO_PIN_0); // toggle PA0  
}  
void HAL_UART_RxCpltCallback(UART_HandleTypeDef*huart)  
{  
    HAL_UART_Receive_DMA(&huart2, Rx_data, 4);  
}
```

```
int main ()  
.....  
    HAL_UART_Receive_DMA (&huart2, Rx_data, 4); // Receive 4  
Bytes of data  
  
while (1)  
{  
    HAL_GPIO_TogglePin (GPIOA, GPIO_PIN_5);  
    HAL_Delay (250);  
}
```

Add Delete }

DMA Request Settings

Mode	Circular	Increment Address	<input type="checkbox"/>	Peripheral		Memory	<input checked="" type="checkbox"/>
		Data Width	Byte		Byte		Byte

Configuration

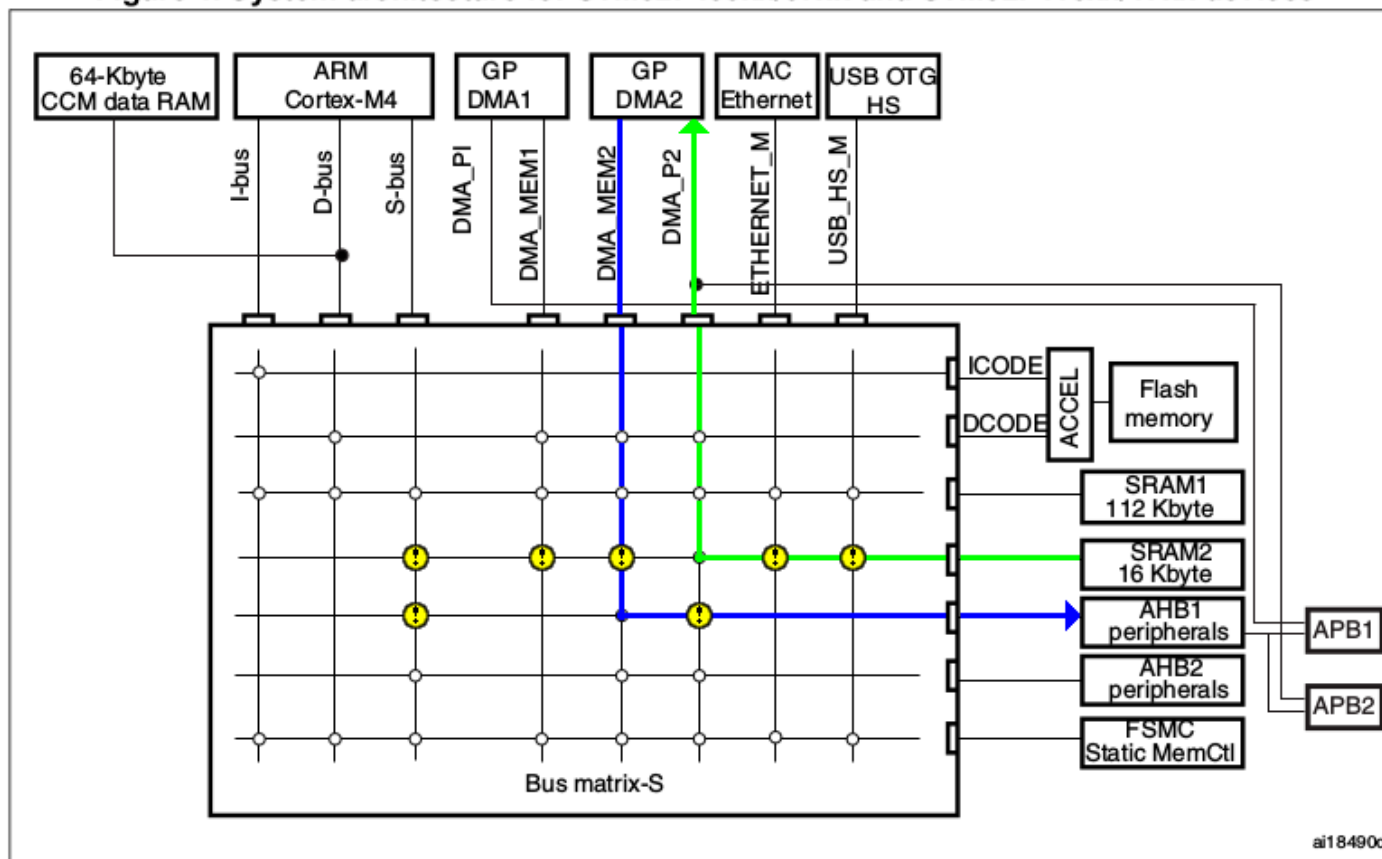
Reset Configuration

Parameter Settings User Constants NVIC Settings **DMA Settings** GPIO Settings

DMA Request	Channel	Direction	
USART1_RX	DMA1 Channel 5	Peripheral To Memory	Low

Primer 2: STM32 – Interna zgradba – stikalna matrika

Figure 1. System architecture for STM32F405xx/07xx and STM32F415xx/17xx devices



<http://cliffle.com/blog/glitch-in-the-matrix/>

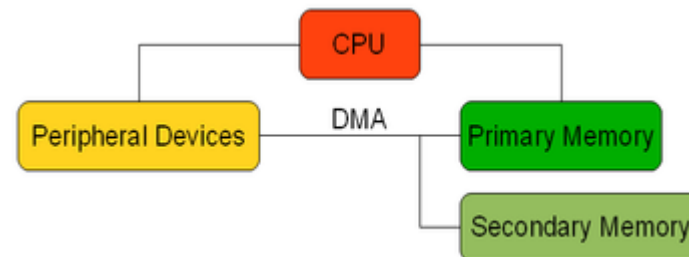
<https://github.com/cbiffle/m4vgalib/>

Primer 3: Analiza delovanja datotečnega sistema

Performance Summary

Method	Latency (μs)	Throughput (40 KB/sec)	CPU Utilization (%)
Naive	100	10	5
Batching	~1000	21	10.5
Interrupts	106	~18	8.9
DMA + Interrupt	61	91	45
DMA + Polling	56	167	84

A **file system** is a data structure which is used to store, retrieve and update a set of files. It can live in memory or secondary storage, so when designing this data structure we need to do it with an eye to speed. We want good performance from our file systems.



Metrics for Performance

Latency*	Response time. Delay from particular request to response.
Throughput*	Rate of request completion (I/O's/sec).
Utilization (closely related to throughput)	Fraction of time that the I/O system is actually doing I/O, can get as close as you can to 100%.

<http://web.cs.ucla.edu/classes/spring13/cs111/scribe/10b/>

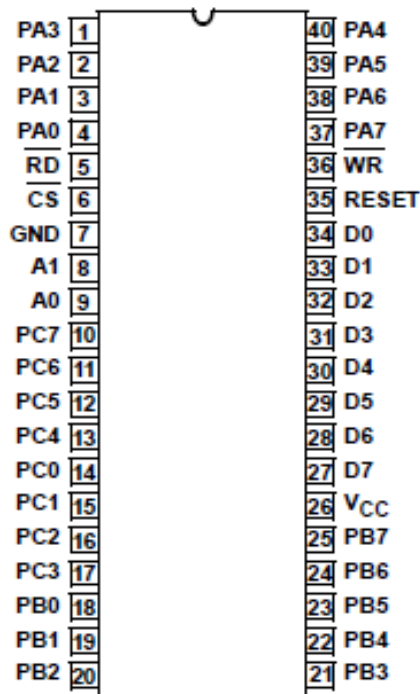
Primer 4: Povezava zaslona in tipkovnice na V/I krmilnik

□ Programabilni V/I krmilnik - Intel 82C55A Programmable Peripheral Interface

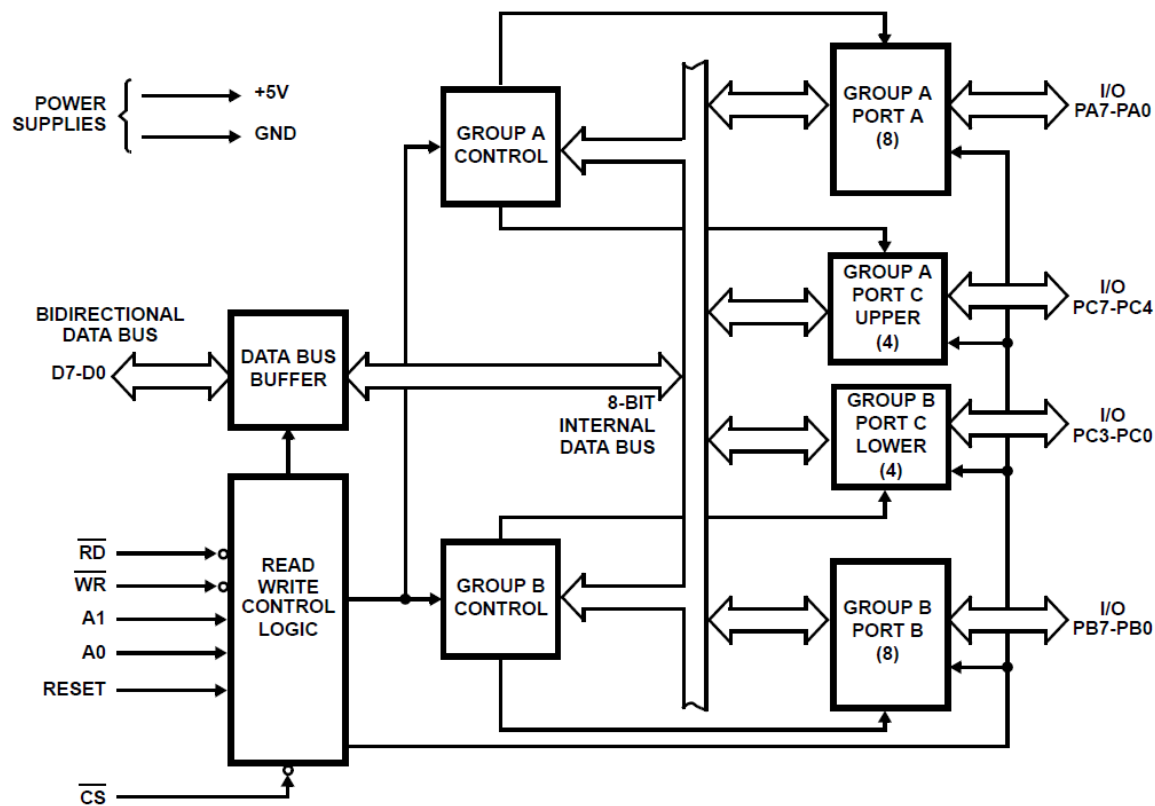
CMOS tehnologija, kompatibilen s TTL vezji

Oznake pinov

82C55A (PDIP, CERDIP)
TOP VIEW



Funkcijski diagram



<https://www.renesas.com/us/en/www/doc/datasheet/82c55a.pdf>

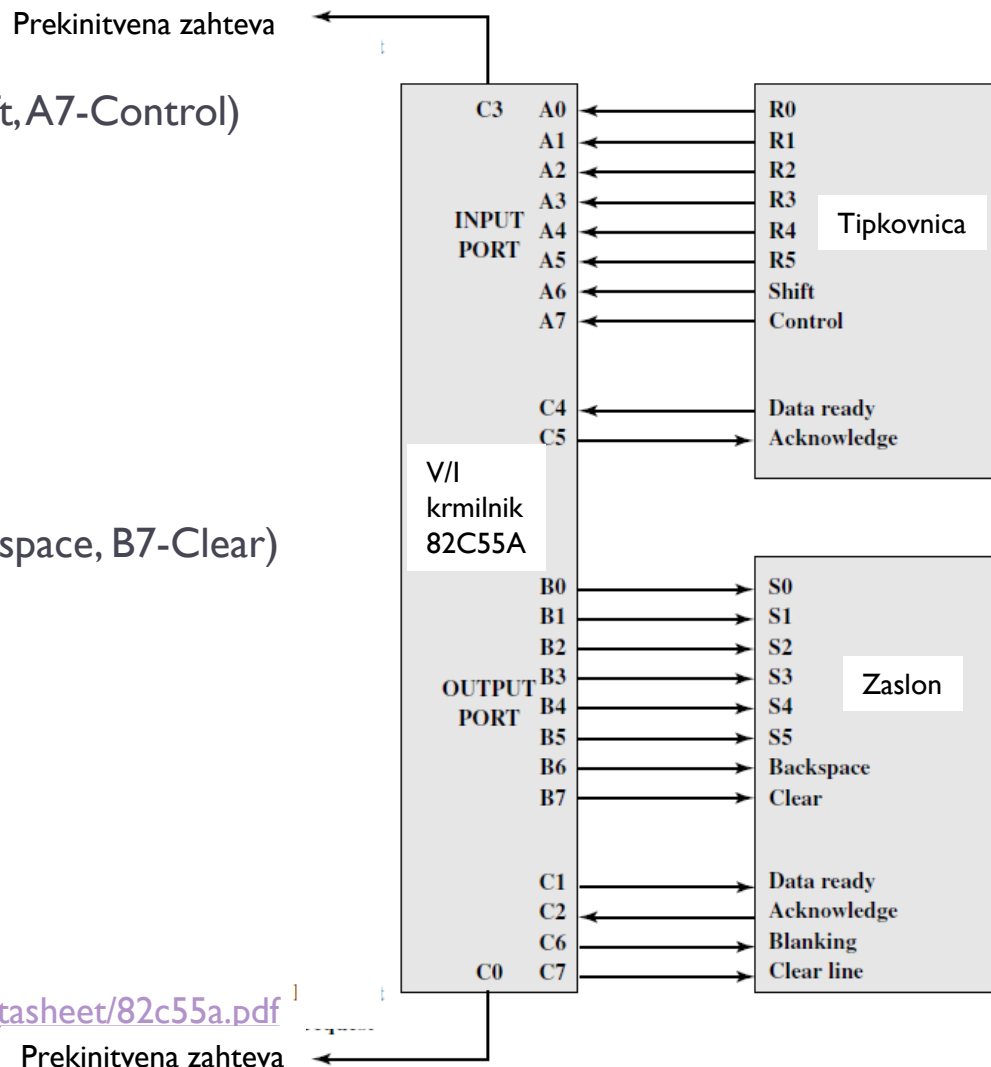
Primer 4: Povezava zaslona in tipkovnice na V/I krmilnik

❑ Povezava tipkovnice

- 8-bitna V/I vrata: A0-A7 (A6-Shift, A7-Control)
- Prekinitvena zahteva: C3
- Usklajevanje:
 - Data ready: C4
 - Acknowledge: C5

❑ Povezava zaslona

- 8-bitna V/I vrata: B0-B7 (B6-Backspace, B7-Clear)
- Prekinitvena zahteva: C0
- Usklajevanje:
 - Data ready: C1
 - Acknowledge: C2
 - Blanking: C6
 - Clear Line: C7



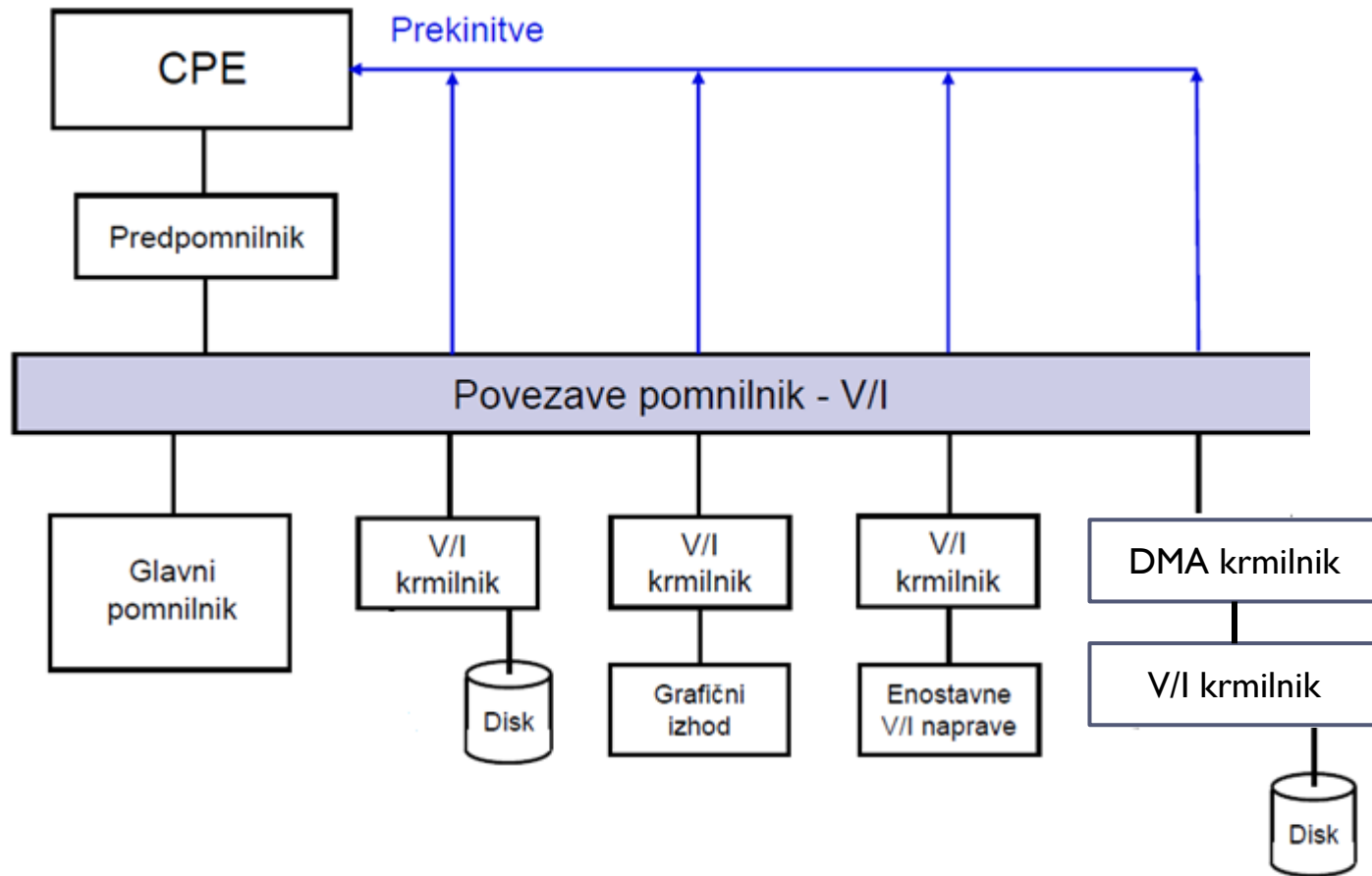
<https://www.renesas.com/us/en/www/doc/datasheet/82c55a.pdf>

Naloga 1

- ❑ Imamo von Neumannov model računalnika s CPE, predpomnilnikom in glavnim pomnilnikom. Upoštevamo naslednje zahteve:
 - Vsaka V/I naprava je na vodilo povezana preko V/I krmilnika.
 - Na V/I krmilnik je lahko povezanih več enakih naprav.
 - Delovanje V/I naprav poteka s prekinitvenim mehanizmom.

- ❑ Narišite arhitekturno shemo, ki po vodilu povezuje s CPE in pomnilnik z naslednjimi V/I napravami:
 - Trdi disk HDD
 - Trdi disk HDD z DMA prenosom DMA
 - Grafični izhod (zaslon, projektor)
 - Enostavne V/I naprave: miška, tipkovnica

Naloga I: Rešitev

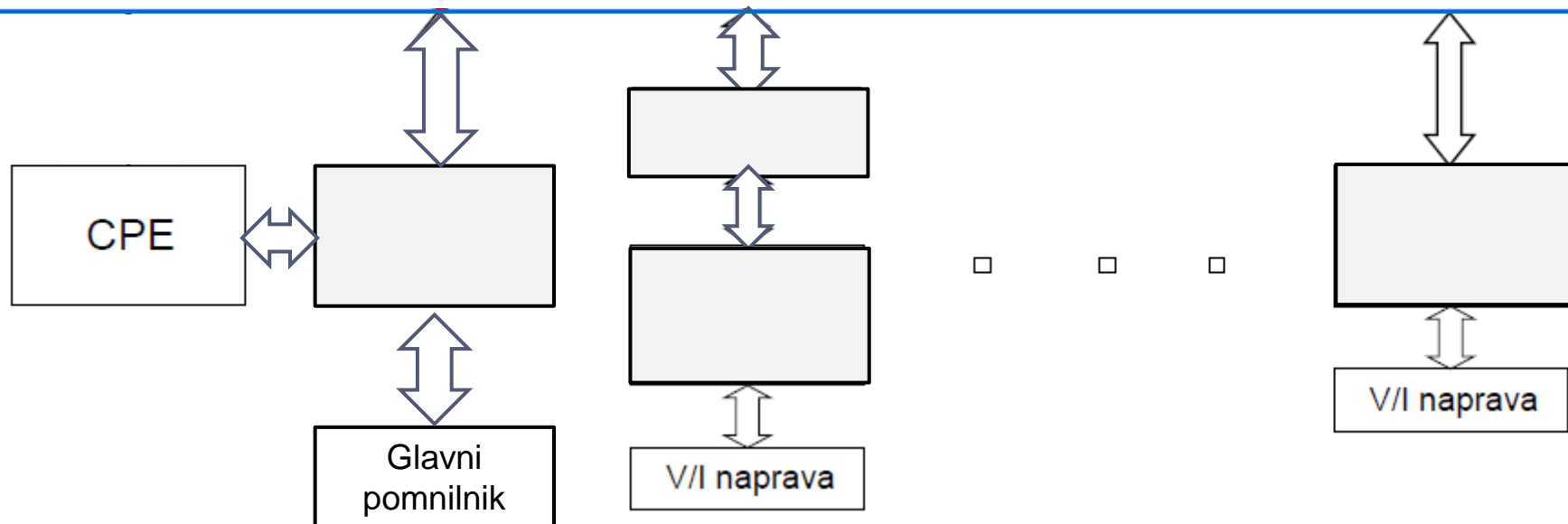


Naloga 2

Pomnilnik in V/I naprave so povezane na vodilo na tri načine, če uporabimo:

- DMA krmilnik kot samostojni modul
- V/I krmilnik z vgrajenim DMA
- Krmilnik pomnilnika, ki direktno komunicira s CPE
- **Dopolnite shemo V/I sistema s povezavami.**

vodilo – povezuje več naprav



Naloga 2 - Rešitev

- ❑ Krmilnik pomnilnika – direktna komunikacija med glavnim pomnilnikom in CPE
- ❑ DMA krmilnik, V/I krmilnik, V/I naprava,
- ❑ V/I krmilnik z vgrajenim DMA, V/I naprava

vodilo – povezuje več naprav

