FRISC VJ

Sklop FRISC-CT

Sklop FRISC-CT

- Sklop FRISC-CT, ili kraće CT (kratica od Counter Timer) **služi za brojenje impulsa i mjerenje vremena**
  - Za razliku od dosadašnjih VJ, CT ne prenosi podatke
  - Slični sklopovi postoje i za komercijalne procesore

- CT **oslobađa procesor** od nepotrebnih čekanja ili čestih posluživanja prekida

Sklop FRISC-CT

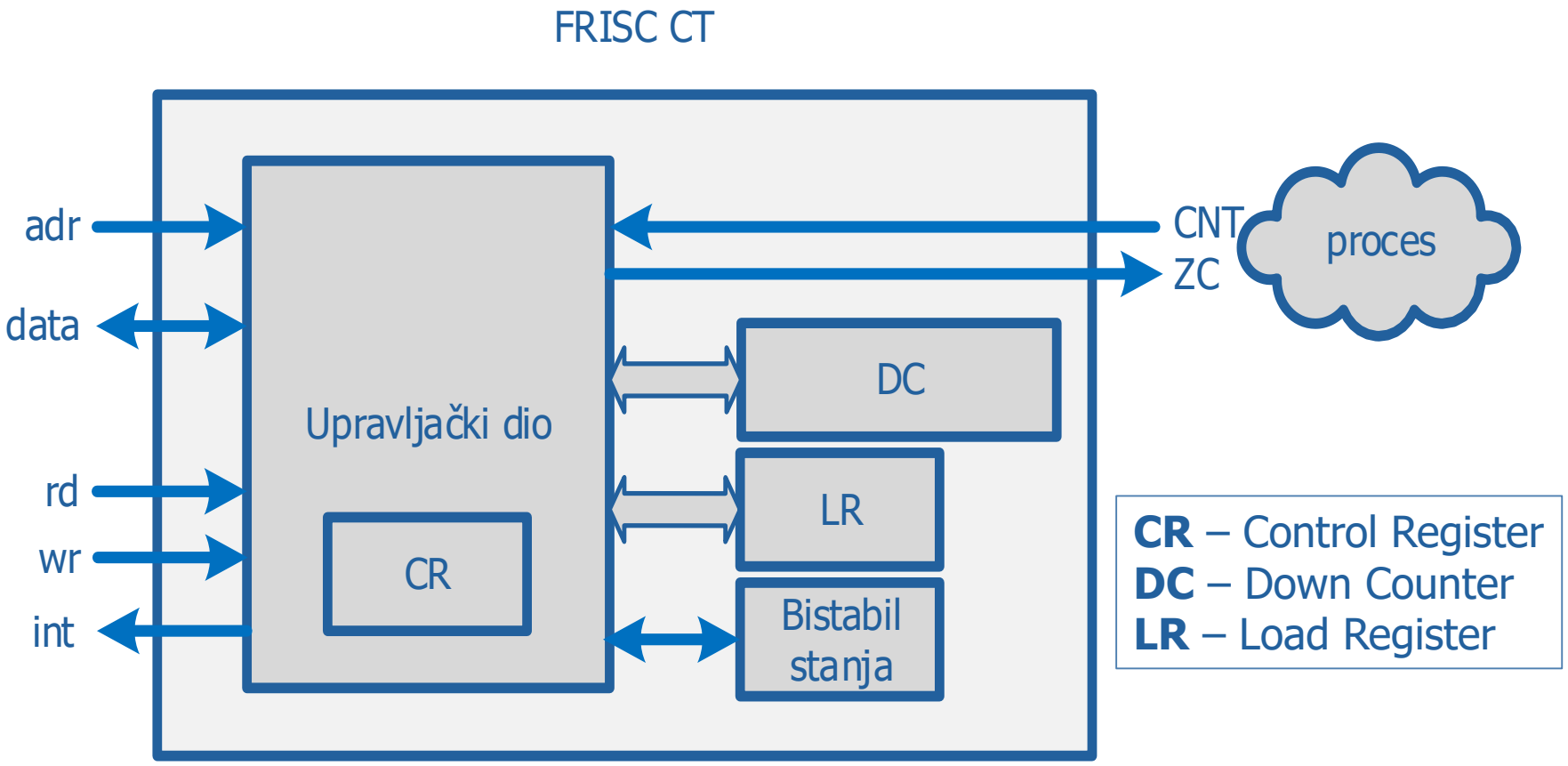
- Za razliku od do sada pokazanih vanjskih jedinica koje su bile općenite i imale su nepromjenjivu zadaću, CT je sklop koji se **može "programirati" (točnije: konfigurirati)**

- Na taj način se ponašanje i funkcija vanjske jedinice prilagođavaju konkretnoj zadaći koju treba napraviti

- Programiranje sklopa se ostvaruje **slanjem posebnih upravljačkih riječi na određene adrese** na kojima se sklop nalazi

Blok-shema CT-a

- Sučelje za spajanje s FRISC-om, isto je kao za prekidne jedinice
- DC broji prema nuli i ima 16 bita



Adrese CT-a

- CT zauzima četiri uzastopne 32-bitne lokacije:

Adresa	Pisanje	Čitanje
PA	Upis CR	Čitanje CR
PA + 4	upis u LR (i DC)	Trenutna vrijednost DC
PA + 8	dojava prihvaćanja prekida (tj. brisanje BS)	čitanje BS
PA+ 12_{10}	dojava o kraju posluživanja prekida	-

CR – Control Register
DC – Down Counter
LR – Load Register

Upravljačka riječ - CR

bitovi 31 – 3	bit 2	bit 1	bit 0
-	VRSTA INT	INT	STOP / START
	0 – maskirajući	0 – ne postavlja prekid	0 – brojilo je zaustavljeno
	1 – nemaskirajući	1 – postavlja prekid	1 – brojilo broji

Ostale adrese

- **CT-ov registar punjenja brojača LR (engl. Load Register)**
  - Na adresi PA+4 pristupa se 16-bitnom registru LR. Pri pisanju u ovaj registar zapisat će se nižih 16 bitova podatka dok će viši dio podatka biti zanemaren.
  - Pri čitanju s ove adrese pročitat će se trenutna vrijednost brojila (DC).
- **CT-ov bistabil stanja BS**
  - Na adresi PA+8 pristupa se bistabilu stanja BS. Kada procesor inicira naredbu pisanja na ovu adresu, podatak koji procesor šalje se zanemaruje, a CT briše BS (BS=0). Pri prekidnom prijenosu, procesor ovime potvrđuje prihvaćanje zahtjeva za prekid.
  - Čitanjem ove adrese, na najnižem bitu podatka čita se trenutna vrijednost BS.
- **Dojava kraja posluživanja prekida**
  - Preko četvrte adrese koju zauzima CT ($PA+12_{10}$) procesor pisanjem bilo kojeg podatka (podatak koji procesor šalje se zanemaruje) javlja sklopu CT da je završeno posluživanje njegovog prekida.
  - Čitanje s ove adrese nije definirano.

Kad DC=0

U trenutku kad brojilo dođe do nule, CT izvodi sljedeće:

- CT postavlja BS=1 i generira prekid (ako je konfiguracijom omogućeno)
- Na izlazu ZC (engl. Zero Count) CT generira jedan pozitivan impuls
- CT automatski kopira vrijednost iz LR u DC, čime brojilo može ponovno početi brojiti nove impulse

CT Primjeri

Na CT-ov priključak CNT spojen je izlaz iz stroja koji za svaki proizvedeni vijak generira impuls. Računalo mora u lokaciji BR_PAK prebrajati proizvedene pakete od po 200 vijaka. Treba riješiti zadatak CT-om (na adresi FFFF0000) tako da:

a) CT radi u uvjetnom načinu,

b) CT radi u prekidnom načinu i spojen je na INT.

Rješenje a) Uvjetni način rada:

```
CTCR          EQU   0FFFF0000
CTLR          EQU   0FFFF0004
CTSTAT        EQU   0FFFF0008
CTEND         EQU   0FFFF000C

       ORG   0
       ; GLAVNI PROGRAM

GLAVNI MOVE   %D 200, R0   ; INICIJALIZACIJA CT-a
       STORE  R0, (CTLR)

       MOVE   1, R0        ; brojilo broji
       STORE  R0, (CTCR)
```

>>>>

<<<<

```
        ; ISPITIVANJE CT-a
PETLJA LOAD   R0, (CTSTAT)   ;čekanje spremnosti
       AND    R0, 1, R0      ;tj. čekanje da se
       JR_Z   PETLJA         ;proizvede paket

       STORE  R0, (CTSTAT)   ;brisanje spremnosti
       LOAD   R0, (BR_PAK)
       ADD    R0, 1, R0      ;povećaj brojač
       STORE  R0, (BR_PAK)

       STORE  R0, (CTEND)    ;kraj posluživanja
       JR     PETLJA

BR_PAK        DW    0
```

Rješenje b) Prekidni način rada:

```
CTCR          EQU    0FFFF0000
CTLR          EQU    0FFFF0004
CTIACK        EQU    0FFFF0008
CTIEND        EQU    0FFFF000C

      ORG   0
      MOVE   10000, R7
      JP     GLAVNI

      ; PREKIDNI VEKTOR
      ORG   8
      DW    1000
```

>>>>

```
<<<<
GLAVNI ; GLAVNI PROGRAM
       ; INICIJALIZACIJA CT-a
       MOVE   %D 200, R0          ; postavljanje brojača
       STORE  R0, (CTLR)

       ; KONTR. RIJEČ
       MOVE   %B 11, R0           ; INT + brojilo broji
       STORE  R0, (CTCR)
       MOVE   %B 10000, SR        ; OMOGUĆI PREKID

       ; "KORISTAN POSAO"
PETLJA JR     PETLJA


BR_PAK DW    0                    ; BROJAČ PAKETA
                                                >>>>
```

<<<<

```
ORG    1000   ; PREKIDNI POTPROGRAM

PUSH    R0
MOVE    SR, R0          ; spremanje konteksta
PUSH    R0
STORE   R0, (CTIACK)    ; obriši spremnost

LOAD    R0, (BR_PAK)    ; povećaj
ADD     R0, 1, R0       ; brojač
STORE   R0, (BR_PAK)    ; paketa

POP     R0
MOVE    R0, SR          ; obnova konteksta
POP     R0
STORE   R0, (CTIEND)    ; dojava kraja
RETI
```

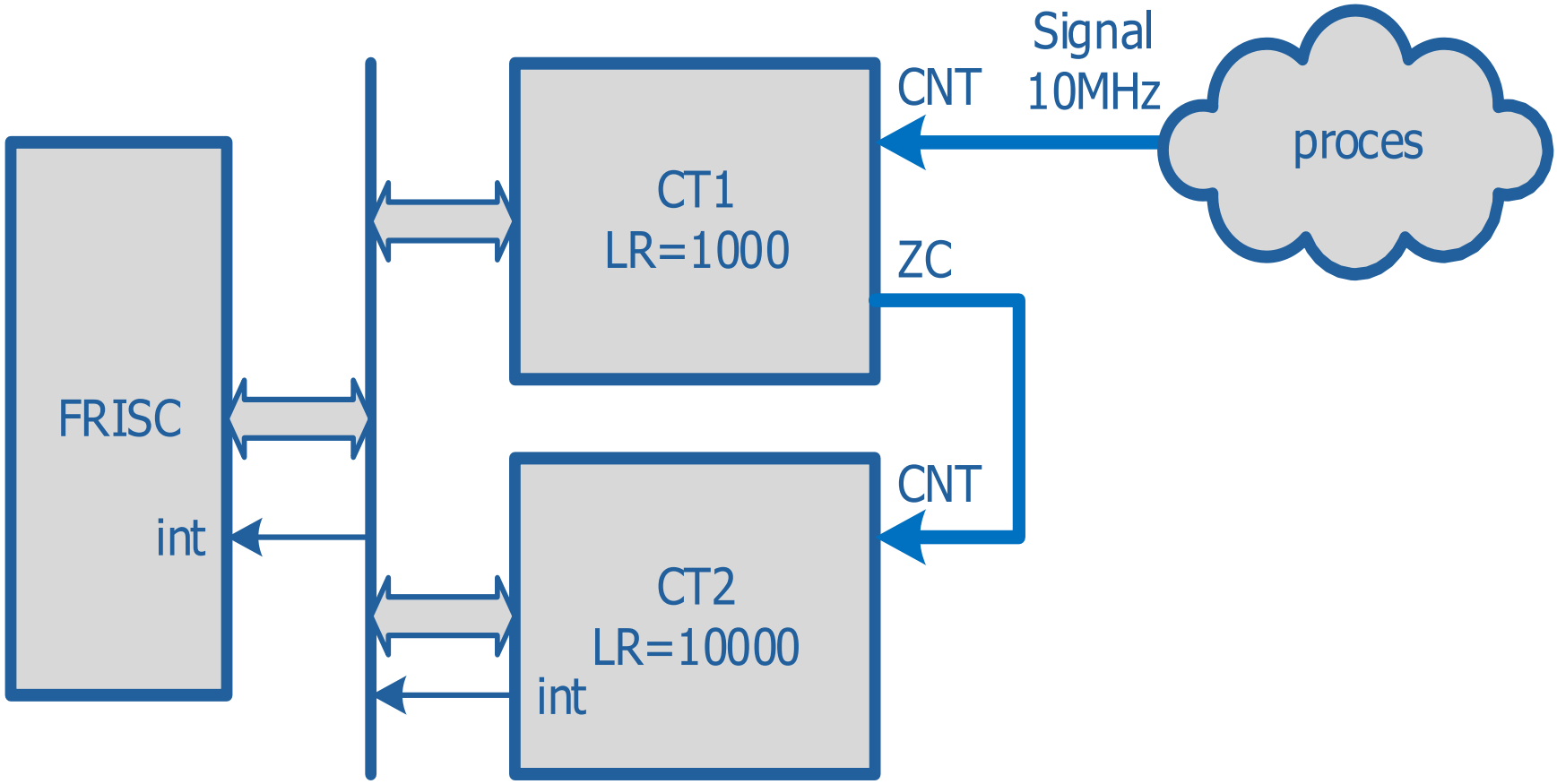
Napomena:

Iako je rješenje s prekidom, dulje, izvodi se puno efikasnije, jer uvjetno posluživanje troši gotovo svo vrijeme samo na ispitivanje spremnosti CT-a.

CT Primjeri

FRISC treba svake sekunde izvesti potprogram POTP. Vremensko kašnjenje treba ostvariti pomoću sklopova CT, a ne programskom petljom za kašnjenje. Pretpostavka je da program POTP već postoji i da njegovo izvođenje sigurno traje kraće od jedne sekunde. Signal koji se dovodi na CT ima frekvenciju 10 MHz.

Prijedlog rješenja



```
CTCR1          EQU    0FFFF1000
CTLR1          EQU    0FFFF1004
CTIACK1        EQU    0FFFF1008
CTIEND1        EQU    0FFFF100C
CTCR2          EQU    0FFFF2000
CTLR2          EQU    0FFFF2004
CTIACK2        EQU    0FFFF2008
CTIEND2        EQU    0FFFF200C

       ORG    0
       MOVE    10000, R7
       JP      GLAVNI

       ORG    8
       DW      1000     ; prekidni vektor
```

>>>>

<<<<

```
GLAVNI ; inicijaliziraj CT1
       MOVE   %D 1000, R0          ; LR1=1000
       STORE  R0, (CTLR1)
       MOVE   1, R0                ; CR1: bez INT
       STORE  R0, (CTCR1)

      ; inicijaliziraj CT2
       MOVE   %D 10000, R0         ; LR2=10000
       STORE  R0, (CTLR2)
       MOVE   %B 11, R0            ; CR2: postavlja INT
       STORE  R0, (CTCR2)

       MOVE   %B 10000, SR         ; omogući prekid INT0
PETLJA JR     PETLJA               ; nastavak glavnog programa
```

>>>>

<<<<

```
; prekidni potprogram - izvodi se svake sek.
; Kontekst se ne sprema jer se ne mijenjaju registri.
; POTP mora spremati sve registre koje mijenja, uključujući SR

      ORG   1000

      STORE  R0, (CTIACK2)   ; potvrda prekida
      CALL   POTP            ; poziv zadanog potprograma
      STORE  R0, (CTIEND2)   ; potvrda kraja
      RETI
```

Sklop FRISC-GPIO

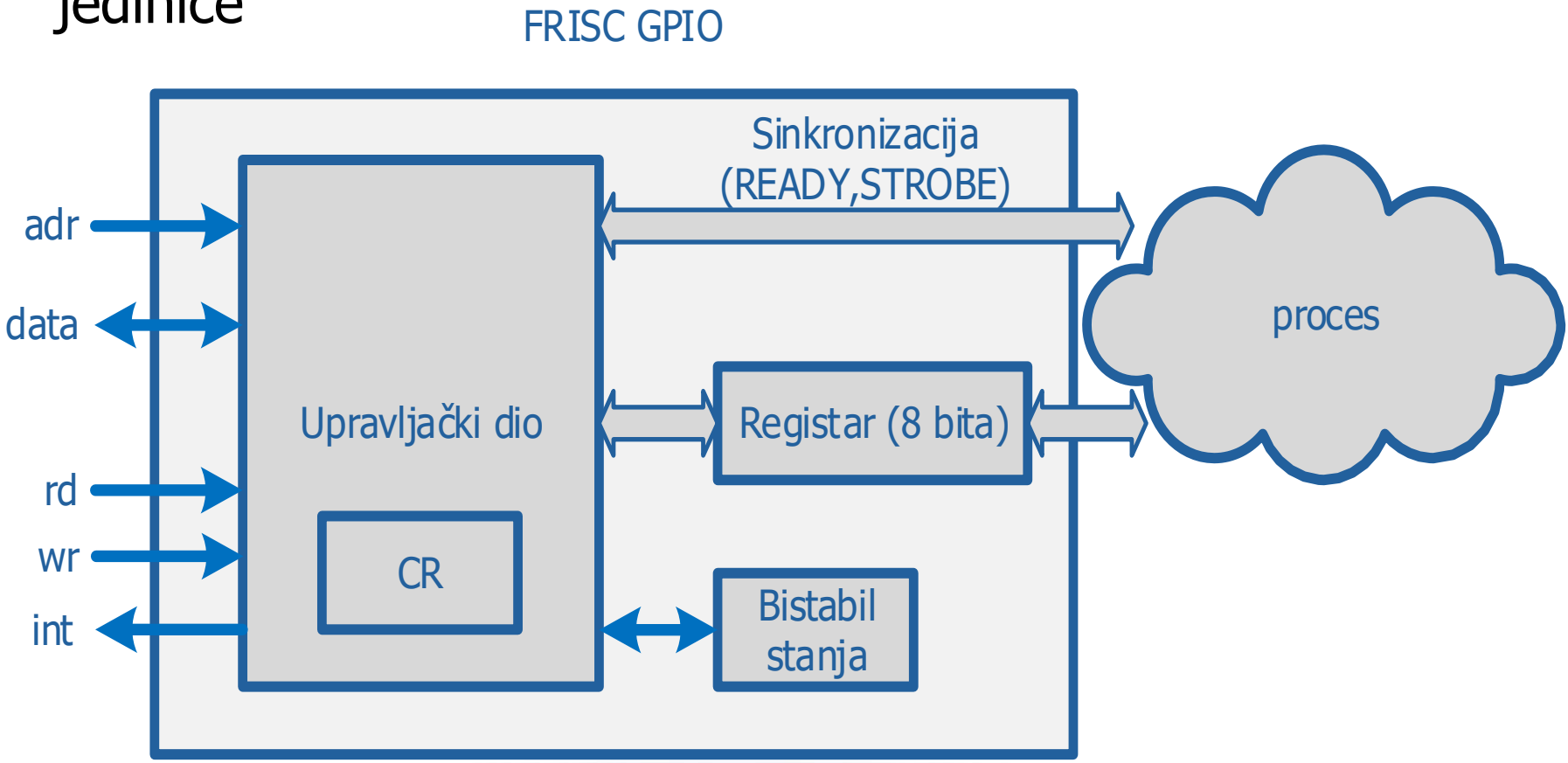
Sklop FRISC-GPIO

- Sklop FRISC-GPIO, ili kraće GPIO (kratica od General Purpose Input-Output) je sklop koji **služi za paralelni prijenos podataka** (8 bitnih)
  - Posrednik između procesora i procesa (djelomično sličan općim VJ kojima smo do sada prenosili podatke)
  - Slični sklopovi postoje i za komercijalne procesore

- GPIO može raditi u 4 načina rada od kojih dva omogućavaju sinkroni prijenos podataka a dva asinkroni:
  - ulazni (ulazni, sinkroni)
  - ispitivanje bitova (ulazni, asinkroni)
  - izlazni (izlazni, sinkroni)
  - postavljanje bitova (izlazni, asinkroni)

Blok-shema GPIO

- Sučelje za spajanje s FRISC-om, isto je kao za prekidne jedinice



GPIO

Adresa	Pisanje	Čitanje
PA	upravljačka riječ CR	upravljačka riječ CR
PA + 4	upis podatka u DR	čitanje DR
PA + 8	dojava prihvaćanja prekida (tj. brisanje BS)	čitanje BS
PA + 12_{10}	dojava o kraju posluživanja prekida	-

Registar CR

31-24	23 - 16	15 - 8	7-5	4	3	2	1 - 0
-	ACTIVE	MASK	-	AND/OR	VRSTA INT	INT	MODE
	0 – aktivna je 0 1 – aktivna je 1			0 – OR 1 – AND	0 – maskirajući 1 – nemaskirajući	0 – ne postavlja prekid 1 – postavlja prekid	00 – izlazni način 01 – ulazni način 10 – postavljanje bitova 11 – ispitivanje bitova

Maska MASK zadaje koji bitovi se ispituju:
0 – bit se ne ispituje
1 – bit se ispituje

Načini rada

- Izlazni
  - prijenos je sinkroniziran upravljačkim signalima READY i STROBE
  - Istovjetno već ranije opisanoj izlaznoj prekidnoj VJ

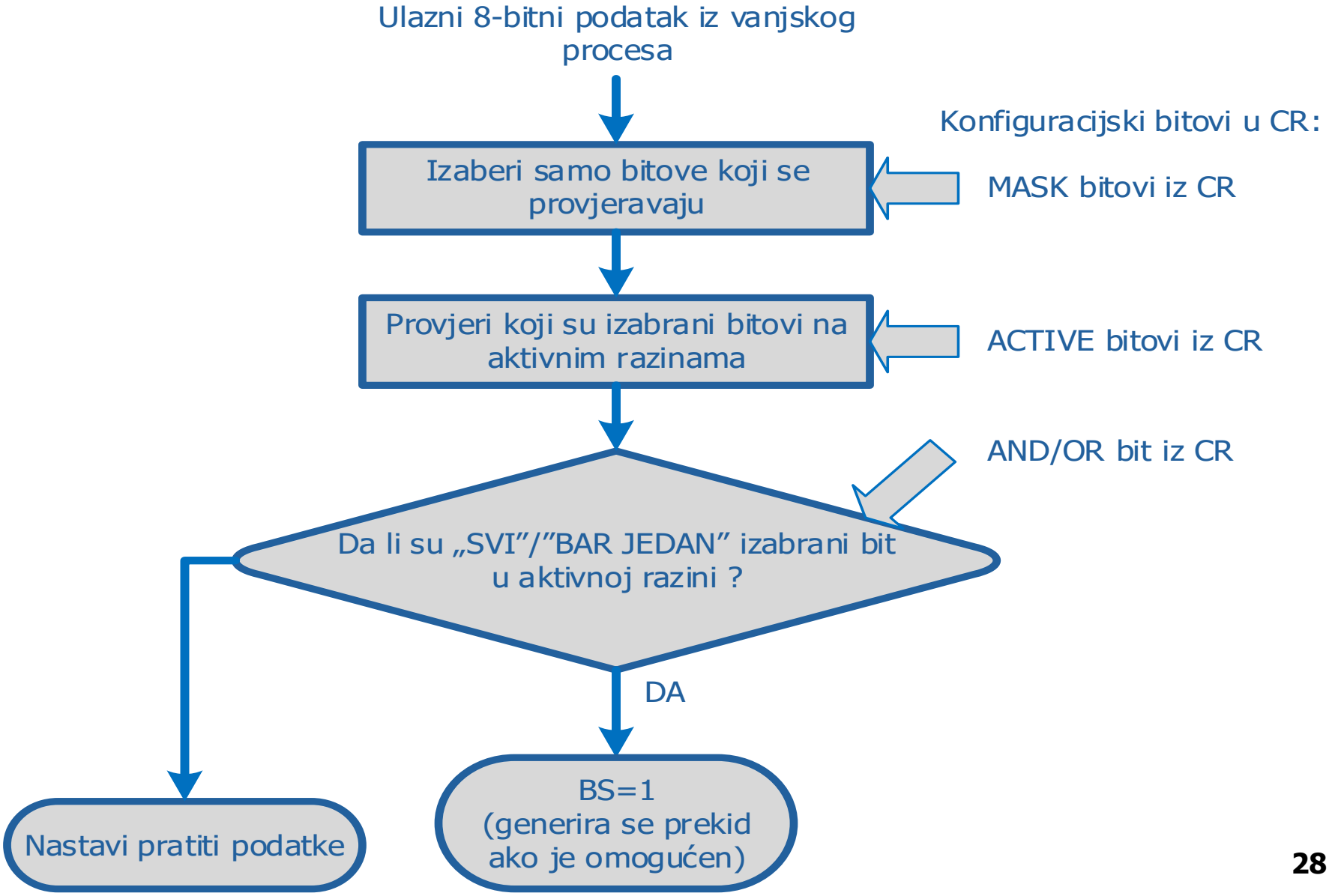
- Ulazni
  - prijenos je sinkroniziran upravljačkim signalima READY i STROBE
  - Istovjetno već ranije opisanoj ulaznoj prekidnoj VJ

Načini rada

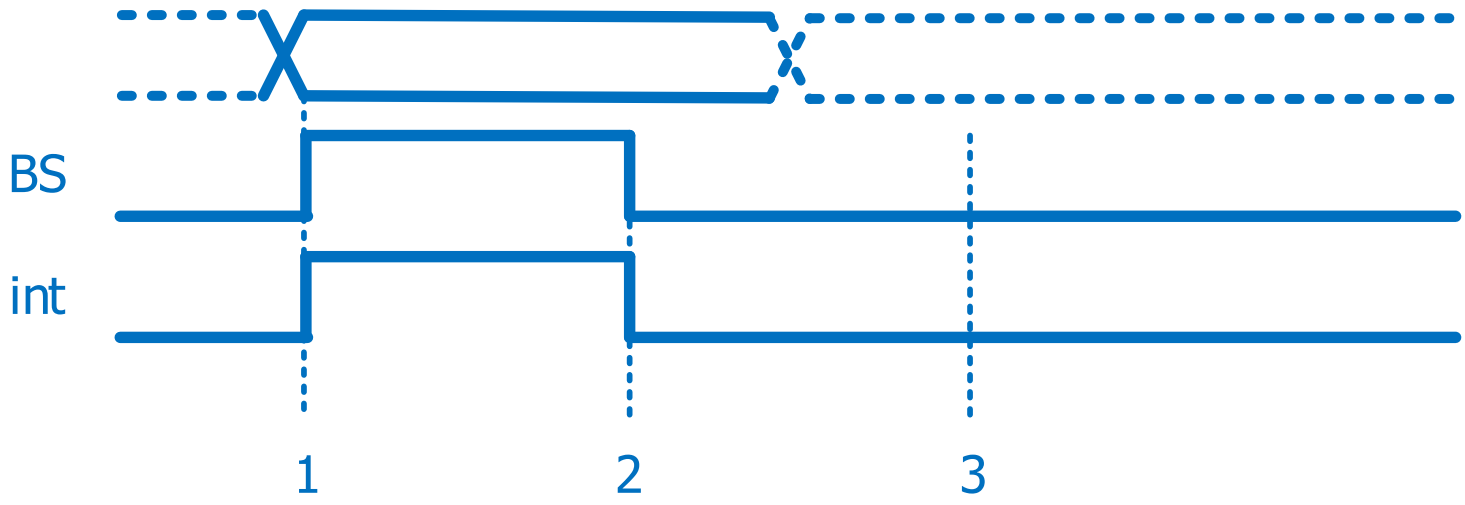
- Postavljanje bitova
  - ne koriste se sinkronizacijski signali, bistabil stanja ni prekid
  - GPIO u ovom načinu radi kao bezuvjetna izlazna VJ

- Ispitivanje bitova
  - ne koriste se sinkronizacijski signali
  - GPIO u ovom načinu radi kao bezuvjetna ulazna VJ
  - Dodatna funkcija: automatska provjera stanja na ulazima i postavljanje BS i generiranje prekida

Ispitivanje bitova



Ispitivanje bitova



Vanjski proces postavlja podatak koji zadovoljava uvjete: GPIO generira prekid

Procesor prihvaća prekid (ili briše BS)

Procesor javlja kraj posluživanja

GPIO Primjeri

Na pisač koji je spojen na GPIO treba poslati 80_{10} znakova veličine jedan oktet, smještenih u memoriji od lokacije ZNAKOVI. GPIO radi u prekidnom načinu, a adresa mu je FFFF0000. Pretpostavka je da nema drugih izvora prekida, a GPIO je spojen na int[0]. Pretpostavka je da pisač ima linije za rukovanje kompatibilne sa READY i STROBE.

PIO će raditi u izlaznom načinu, jer će s pisačem biti potrebna sinkronizacija.

```
PIOC    EQU   0FFFF0000
PIOD    EQU   0FFFF0004
PIOIACK EQU   0FFFF0008
PIOIEND EQU   0FFFF000C
```

```
        ORG   0
        MOVE   10000, R7
        JP     GLAVNI

        ORG   8     ; prekidni vektor
        DW    500
```

>>>>

```
<<<<
        ; glavni program
        ; inicijalizacija sklopa PIO
GLAVNI  MOVE   %B 0100, R0     ; 0=INT, 1=prekid, 00=izlazni
        STORE  R0, (PIOC)      ; pošalji u CR
        MOVE   %B 010000, SR   ; dozvoli INT

PETLJA  JR     PETLJA          ; "koristan posao"


BROJAC          DW     0      ; brojač poslanih znakova
ZNAKOVI         DB     ...    ; 80 znakova za slanje

                                        >>>>
```

<<<<

```
ORG   500   ; prekidni potprogram
PUSH   R0
PUSH   R1
PUSH   R2
MOVE   SR, R0
PUSH   R0

STORE  R0, (PIOIACK) ; potvrda prekida

LOAD   R1, (BROJAC)  ; dohvat brojača
MOVE   ZNAKOVI, R0   ; dohvat početne adrese

ADD    R0, R1, R0    ; računanje adrese znaka

LOADB  R2, (R0)      ; dohvat znaka iz mem.
STORE  R2, (PIOD)    ; slanje znaka na PIO
```

>>>>

```
<<<<
        ADD     R1, 1, R1          ; povećanje brojača
        STORE   R1, (BROJAC)

        CMP     R1, %D 80          ; je li poslan
        JR_NE   JOS                ; zadnji znak ?

KRAJ ; zabrani PIO-u da dalje zahtijeva prekide
        MOVE   00, R0
        STORE R0, (PIOC)

JOS  ; ima još znakova za slanje
        POP     R0
        MOVE    R0, SR
        POP     R2
        POP     R1
        POP     R0
        STORE   R0, (PIOIEND) ; dojava kraja posluživ.
        RETI
```

GPIO Primjeri

Na prvi GPIO na ulazne bitove PIOD3-POID7 spojeno je 5 senzora (aktivna razina im je niska). Svaki puta kad se svih 5 senzora aktiviraju, FRISC treba bezuvjetno poslati procesu podatak iz bloka memorije s početnom adresom BLOK. Proces je spojen na drugi GPIO. Prvi GPIO radi u prekidnom načinu.

Nakon slanja 10 podataka, treba zabraniti daljnje generiranje prekida od strane GPIO1 i nastaviti izvođenje glavnog programa.

Odaberimo adrese za GPIO1 i GPIO2: FFFF1000 i FFFF2000. GPIO1 ćemo spojiti na INT i programirati da radi u načinu ispitivanja bitova, a GPIO2 u načinu postavljanja bitova.

```
PIOC1          EQU   0FFFF1000
PIOD1          EQU   0FFFF1004
PIOIACK1       EQU   0FFFF1008
PIOIEND1       EQU   0FFFF100C

PIOC2          EQU   0FFFF2000
PIOD2          EQU   0FFFF2004

               ORG   0
       MOVE    10000, R7
       JP      GLAVNI

       ORG    8     ; prekidni vektor
       DW      500
```

>>>>

<<<<

```
GLAVNI MOVE   BLOK, R0          ; adresu podataka
       STORE R0, (PODAT)        ; stavi u PODAT

       MOVE   %D 10, R0         ; broj podataka
       STORE R0, (BROJAC)       ; stavi u BROJAC

       ; inicijalizacija sklopova GPIO1 i GPIO2
       MOVE   %B 111110000010111, R0 ;11111000=mask,000=-, 1=AND
                                     ;0=INT, 1=prekid, 11=ispit.
       STORE R0, (PIOC1)        ; pošalji u CR
       MOVE   %B 010, R0        ; 0=nema prekida, 10=postav.bitova
       STORE R0, (PIOC2)        ; pošalji u CR

       MOVE   %B 10000, SR      ; dozvoli INT
PETLJA  JR     PETLJA           ; "koristan posao"
```

>>>>

31-24	23-16	15-8	7-5	4	3	2	1-0
-	ACTIVE	MASK	-	AND/OR	VRSTA INT	INT	MODE
	0 – aktivna je 0 1 – aktivna je 1			0 – OR 1 – AND	0 – maskirajući 1 – nemaskirajući	0 – ne postavlja prekid 1 – postavlja prekid	00 – izlazni način 01 – ulazni način 10 – postavljanje bitova 11 – ispitivanje bitova

```
<<<<

PODAT   DW   0
BROJAC  DW   0

BLOK    DW   3, 1, 5, 7, 3, 9, 2, 6, 5, 4

        ORG    500              ; prekidni potprogram

        PUSH  R0                ; spremi kontekst
        PUSH  R1
        MOVE  SR, R0
        PUSH  R0

        STORE R0, (PIOIACK1); dojavi prihvat na PIO1

        LOAD  R0, (PODAT)    ; dohvati adresu podatka
        LOAD  R1, (R0)       ; dohvati podatak
        STORE R1, (PIOD2)    ; šalji podatak na PIO2
        ADD   R0, 4, R0      ; pomakni adresu na...
        STORE R0, (PODAT)    ; ...sljedeći podata       >>>>
```

```
<<<<
        LOAD   R0, (BROJAC)     ; dohvati...
        SUB    R0, 1, R0        ; ...i smanji...
        STORE  R0, (BROJAC)     ; ...brojač
        JR_NZ  JOS   ; ima li još podataka

KRAJ   ; zabrani prekide na PIO1
        MOVE   %B 011, R0
        STORE  R0, (PIOC1)      ; pošalji u CR

JOS     POP    R0               ; obnovi kontekst
        MOVE   R0, SR
        POP    R1
        POP    R0
        STORE  R0, (PIOIEND1)   ; dojava kraja posluživanja
        RETI
```

GPIO Primjeri

FRISC pomoću dva sklopa GPIO upravlja radom stroja. GPIO1 je na adresi FFFF1000, a GPIO2 na FFFF2000.

Na GPIO 1 spojeni su izlazi iz senzora za temperaturu (bit 0), pritisak (bit 1) i ulaz sirovina (bit 2). Ako bilo koja od tih vrijednosti prijeđe dopuštenu razinu, senzor šalje jedinicu, a u suprotnom se šalje nula. GPIO1 spojen je na INT.

Na GPIO2 je spojen samo jedan relej (bit 0) koji se uključuje jedinicom, a isključuje nulom. Relej uključuje i isključuje stroj, a početno je isključen.

Program na početku uključuje stroj (pretpostavka je da su sve mjerene veličine ispravne). Nakon toga se prate senzori i mora se isključiti stroj ako bilo koja od mjerenih veličina poprimi nedopuštenu razinu. Kada se sve vrijednosti vrate u normalnu razinu, treba ponovno uključiti stroj. Ovo se beskonačno ponavlja.

GPIO Primjeri

Rješenje:

GPIO2 ćemo programirati da radi u načinu postavljanja bitova i na početku ćemo pomoću njega uključiti stroj, a kasnije ga po potrebi isključivati i uključivati.

GPIO1 ćemo programirati da radi u načinu ispitivanja bitova i da generira prekid, ali na dva načina - ovisno treba li čekati uvjet da se stroj isključi ili treba čekati uvjet da se stroj ponovno uključi.

Da bi se stroj isključio, **barem jedan senzor** mora dati **neispravnu** razinu mjerene vrijednosti. To znači da zadajemo aktivnu razinu 1 i funkciju OR.

Da bi se stroj ponovno uključio, **svi senzori** moraju davati **ispravnu** razinu mjerenih vrijednosti. To znači da zadajemo aktivnu razinu 0 i funkciju AND.

```
PIO1C     EQU     0FFFF1000
PIO1D     EQU     0FFFF1004
PIO1IACK  EQU     0FFFF1008
PIO1IEND  EQU     0FFFF100C

PIO2C     EQU     0FFFF2000
PIO2D     EQU     0FFFF2004

          ORG     0
          MOVE    10000, SP
          JP      GLAVNI

          ; prekidni vektor
          ORG     8
          DW      2000                          >>>>
```

```
<<<<    ; glavni program

GLAVNI  ; inicijalizacija sklopa GPIO1
        MOVE   %B 00000111000001110000111, R0
                ; 00000111 = AKT.RAZINA,   00000111 = MASKA,
                ; 000 = ne koristi se,   0 = OR,   0 = INT,
                ; 1 = prekid,   11 = ispitivanje bitova
        STORE   R0, (PIO1C)

        ; inicijalizacija sklopa GPIO2
        MOVE   %B 010, R0      ; 0 = nema prekida,
                               ; 10 = postavljanje bitova
        STORE   R0, (PIO2C)
                                                >>>>
```

```
<<<<
        MOVE   %B 10000, SR     ; omogući INT

        MOVE   ON, R0
        STORE  R0, (PIO2D)      ; uključi stroj
        STORE  R0, (STANJE)     ; STANJE=uključen

PETLJA ...                      ; "koristan posao"



; stanje stroja:

OFF     EQU    0          ; dvije "konstante" ON i OFF
ON      EQU    1

STANJE  DW     OFF        ; "varijabla" koja pamti trenutačno
                          ; stanje stroja
                                                >>>>
```

```
<<<<  ; prekidni potprogram
      ORG  2000

      PUSH  R0              ; pohrana konteksta
      MOVE  SR, R0
      PUSH  R0

      STORE R0, (PIO1IACK) ; dojavi prihvat prekida

      LOAD  R0, (STANJE)    ; ispitaj je li stroj
      CMP   R0, ON          ; isključen ili uključen
      JR_EQ UKLJUCEN_JE
      JR    ISKLJUCEN_JE

VAN   ; dio za povratak iz p.p.
      POP   R0
      MOVE  R0, SR          ; obnova konteksta
      POP   R0
      STORE R0, (PIO1IEND) ; dojava kraja posluž.
      RETI
                                          >>>>
```

```
<<<<            ; stroj je uključen i neka od
                ; mjerenih veličina je neispravna
UKLJUCEN_JE     ; =>  treba isključiti stroj i čekati da sve
                ; mjerene veličine postanu ispravne

        MOVE    OFF, R0
        STORE   R0, (PIO2D)     ; isključi stroj
        STORE   R0, (STANJE)    ; STANJE=isključen

        ; nova kontrolna riječ za GPIO1:
        ; sada treba čekati uvjet za ponovno uključenje
        MOVE    %B 000000000000011100010111, R0
                ; 00000000 = AKT.RAZINA,   00000111 = MASKA,
                ; 000 = ne koristi se,   1 = AND,   0 = INT,
                ; 1 = prekid,   11 = ispitivanje bitova
        STORE   R0, (PIO1C)

        JR      VAN     ; povratak iz p.p.

                                                >>>>
```

```
<<<<            ; stroj je iskljucen i sve mjerene
                ; veličine su opet ispravne
ISKLJUCEN_JE    ; =>  treba uključiti stroj i čekati da bilo
                ; koja od mjerenih veličina postane neispravna

        MOVE    ON, R0
        STORE   R0, (PIO2D)     ; uključi stroj
        STORE   R0, (STANJE)    ; STANJE=uključen

        ; nova kontrolna riječ za GPIO1:
        ; sada treba čekati uvjet za ponovno isključenje
        ; (isti uvjet kao na početku u glavnom programu)
        MOVE    %B 00000111000001110000111, R0
                ; 00000111 = AKT.RAZINA,   00000111 = MASKA,
                ; 000 = ne koristi se,   0 = OR,   0 = INT,
                ; 1 = prekid,   11 = ispitivanje bitova
        STORE   R0, (PIO1C)

        JR      VAN     ; povratak iz p.p.
```

CT, GPIO - Primjeri

- DZ: Proučiti dodatne primjere iz knjige i zbirke