Sistem prekida

Katedra za elektroniku

Sistem prekida

- Svaki procesor je u suštini **sekvencijalna mašina**
- Upravljačka jedinica **procesora prihvata, dekoduje i izvršava instrukcije** iz programske memorije prema redosledu koji je naveden u programu
- Imajući ovo u vidu jedan procesor u svakom trenutku može da izvršava samo jednu instrukciju
- Ukoliko se javi potreba za **servisiranjem periferija** od strane procesora, izvršavanje programa se mora prekinuti i programska kontrola se mora preneti na poseban blok instrukcija koji je napisan na takav način da na efikasan način može da opsluži posmatranu periferijsku jedinicu
- Kao primer, razmotrimo šta je potrebno da se opsluži zahtev izdat od strane tastature

Sistem prekida

- Tastatura se na najnižem nivou može posmatrati kao niz tastera, pri čemu sotfver daje odgovarajuće značenje svakom tasteru
- Tasteri su organizovani u **mrežnu strukturu** tako da svaki pritisnuti taster rezultuje u generisanju različitog koda na izlazu tasture
- Nakon svakog pritiska, procesor mora preuzeti generisani kod sa tastature
- Pod terminom *opsluživanje tastature* podrazumevamo akciju preuzimanja koda nakon svakog pritiska tastature i njegovo prosleđivanje programu koji će zatim interpretirani primljeni kod
- Kada govorimo o servisiranju periferijskih jedinica, postoje dva pristupa:
  - Servisiranje bazirano na prozivkama (engl. polling)
  - Servisiranje bazirano na prekidima (engl. interrupt handling)

Servisiranje bazirano na prozivkama

- Kod ovog načina servisiranja procesor neprekidno “ispituje” ili “proziva” status periferijske jedinice kako bi utvrdio da li ona zahteva servisiranje
- Kada utvrdi da periferijska jedinica zahteva servisiranje, procesor preduzima potrebne akcije kako bi servisirao periferiju
- Da bi ilustrovali princip servisiranja baziranog na prozivkama pretpostavimo da je naš zadatak da odgovaramo na telefonske pozive i preuzimamo poruke
- Pretpostavimo dalje da nemamo informaciju o tome kada će pozivi doći, i da na raspolaganju imamo telefon kome ne radi zvono, niti ima vibraciju tako da nemamo nikakav način da ustanovimo da je poziv stigao
- U ovom slučaju jedini način da saznamo da li je neki poziv stigao je da periodično podižemo slušalicu i proverimo da li je neko na liniji

Servisiranje bazirano na prozivkama

- Ovo bi bio prilično frustrirajući posao, pogotovo ako imamo i neke druge obaveze
- Međutim, ukoliko ne želimo da propustimo ni jedan poziv, moramo obustaviti sve ostale aktivnosti i posvetiti se neprekidnom ponavljanju sledeće sekvence:

  1) podizanju slušalice,
  2) prinošenju slušalice uvu,
  3) proveri da li je neko na liniji

- Obzirom da linija mora biti slobodna, kako bi poziv mogao da dođe, na kraju svake sekvence moramo spustiti slušalicu i ponoviti celu sekvencu od početka

- Kakvo gubljenje vremena! Ali to je prinicip rada sistema baziranog na prozivkama

Servisiranje bazirano na prekidima

- Kod ovog sistema, svaka periferijska jedinica koristi poseban signal da signalizira procesoru kada ima potrebu da bude servisirana
- Ovaj signal poznat je pod nazivom ***zahtev za prekidom*** (interrupt request, IRQ)
- **Procesor** može biti zauzet obavljanjem nekih drugih aktivnosti, ili može biti u režimu spavanja, međutim kada stigne neki od zahteva za prekidom, ukoliko su oni dozvoljeni, procesor će obustaviti aktivnost koju je trenutno izvodio kako bi se posvetio obradi prekidnog zahteva, izvršavajući specifičan blok instrukcija
- Ovaj događaj se zove ***prekid***
- Blok instrukcija koje se izvršavaju u cilju opsluživanja periferijske jedinice, kao odgovor na zahtev za prekidom, naziva se ***prekidni potprogram*** (**interrupt service routine, ISR**)

Servisiranje bazirano na prekidima

- Razmotrimo kako bi radio sistem za odgovaranje na pozive u slučaju da je baziran na sistemu prekida
- Pretpostavimo da naš telefon ima zvono pomoću kojega nam može signalizirati da je stigao novi poziv
- Dok čekamo na poziv, možemo obavljati neke druge aktivnosti, jer znamo da će nam zvono signalizirati dolazni poziv
- Kada se zvono oglasi, zaustavićemo tekuću aktivnost kako bismo podigli slušalicu i prihvatili poruku
- U ovom slučaju zvono predstavlja **naš indikator zahteva za prekidom**
- Jasno je da je ovaj sistem daleko efikasniji metod prijema poruka od prethodnog

Vrste prekida

- U opštem slučaju **prekidi unutar mikroračunarskog sistema** dele se u dve kategorije:

  - Maskirajući prekidi – ovi prekidi se mogu zabraniti (maskirati) od strane programera. Način maskiranja varira od procesora do procesora. Na ovaj način moguće je kontrolisati koji izvori prekida mogu prekinuti procesor u datom trenutku.

  - **Nemaskirajuće prekide** – ovi prekidi ne mogu biti zabranjeni od strane programera i svaki put kada se generiše neki prekid ovog tipa procesor će zaustaviti izvršavanje tekuće akcije i preći na opsluživanje prekida. Ovi prekidi su obično rezervisani za signalizaciju kritičnih događaja u sistemu koji se moraju opslužiti odmah.

Maskirajući i nemaskirajući prekidi

- Kao što je ranije rečeno, svi zahtevi za prekidom mogu se podeliti u dve grupe:

  1) Maskirajuće zahteve za prekidom
  2) Nemaskirajuće zahteve za prekidom

- Većina zahteva za prekidom u sistemu spadaju u klasu maskirajućih zahteva i prosto se nazivaju prekidima

- U slučaju maskirajućih zahteva za prekidom postoji odgovarajući mehanizam pomoću kojega programer može da zabrani (maskira) odgovarajući zahtev za prekidom

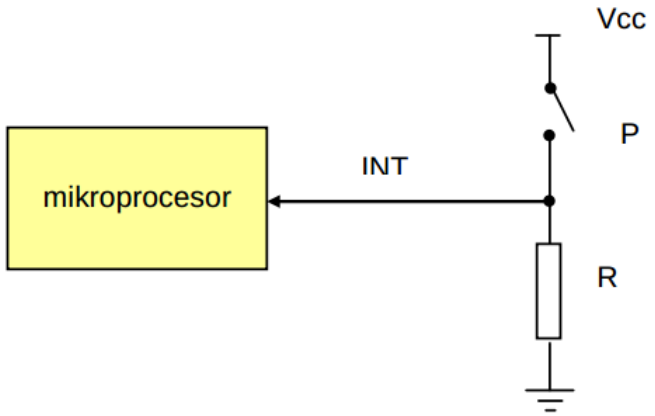
- Najčešće se to postiže postavljanjem ili brisanjem odgovarajućih bitova koji se nalaze unutar registra (Global Interrupt Enable, GIE) ili nekog posebnog registra namenjenog za tu svrhu (Interrupt Enable Register)

Sistem prekida

- Prekidi predstavljaju mehanizam pomoću kojeg mikroprocesor, ili mikrokontroler može da **odreaguje na pojedine događaje** koji zahtevaju hitnu obradu. Ovo je pogotovo od značaja u vremenski kritičnim aplikacijama, gde vremena odziva moraju biti strikno ispoštovana.
- Podsistem prekida omogućava mikroprocesoru da pod dejstvom spoljnog ili unutrašnjeg dogadaja, prekine izvršavanje tekućeg programa i pređe na izvršavanje specijalizovanog potprograma namenjenog obradi tog dogadaja.
- Prekidi mogu biti:
    - **Hardverski (eksterni)** - Izazvani od strane periferijskog uređaja sa kojim je procesor povezan. Tipično, procesor se obaveštava o zahtevu za prekidom putem specijalizovanog **elektronskog signala (INT, engl. Interrupt).** Primeri događaja koji izazivaju zahtev za prekidom: pritisak tastera na tastaturi, pomeranje miša i sl.
    - Softverski (interni) - Izazvan specifičnim stanjem procesora (npr. deljenjem nulom), ili specijalizovanom instrukcijom koja dovodi do pojave prekida (engl. trap).
- Svakom izvoru prekida dodeljuje se zaseban potprogram za obradu. Broj hardverskih prekida ograničen je brojem linija za zahteve za prekidom (engl. **IRQ = Interrupt Request**).
- Prekidi su uobičajeno korišćena tehnika u multitaskingu, pogotovo u aplikacijama koje rade u realnom vremenu.

Načini reagovanja na eksterni događaj

PRIMER: Neka je mikroprocesor povezan sa prekidačem, kao što je prikazano na slici. Potrebno je momentalno detektovati uključenje prekidača i u tom slučaju izvršiti odgovarajuću proceduru.

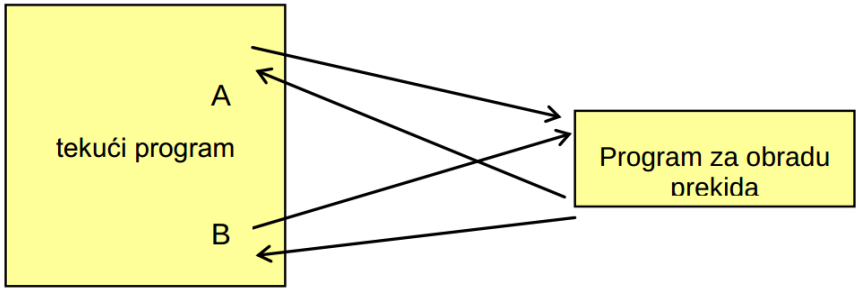


- Dva tipična pristupa rešavanju navedenog problema su:

1. Program se "vrti" u petlji tokom koje kontinualno proverava stanje ulazne linije i u slučaju detekcije logičke jedinice poziva specijalizovanu proceduru. Tokom izvršenja petlje, procesor nije u mogućnosti da obavlja neki drugi zadatak, što može znatno da umanji efikasnost sistema. Ovakav pristup se u programerskom žargonu naziva "prozivanje" (engl. polling).
2. Pritisak tastera detektuje specijalizovana hardverska jedinica - **kontroler prekida**. U međuvremenu, procesor izvršava neki zadatak manje hitnosti. Kada se pojavi zahtev za prekidom, procesor automatski prekida **trenutnu aktivnost, odrađuje potprogram za obradu prekida,** nakon čega nastavlja sa izvršenjem programa od istog mesta gde se nalazio u trenutku pojave prekida.

Potprogram za obradu prekida

- **Potprogram za obradu prekida** je deo programskog koda koji se automatski poziva po nastanku zahteva za prekidom. Naziva se još i prekidna rutina (engl. *Interrupt Service Routine* (ISR), *Interrupt Handler*).
- Program koji procesor izvršava onda kada nema prekida, u ovom kontekstu se naziva **tekući program**. Do prekida može doći tokom različitih faza izvršenja tekućeg programa. Instrukcija tokom čijeg izvršenja je nastao zahtev za prekidom naziva se **tačka prekida**.



- Potrebno je obezbediti mehanizam koji omogućava procesoru povratak u tačku prekida nakon izvršenja **prekidne rutine**. Ovo se postiže tako što se neposredno pre skoka na početak prekidne rutine trenutna vrednost **programskog brojača (PC)** koja predstavlja povratnu adresu, postavlja na **stek**.
- Na kraju prekidne rutine, povratna adresa se automatski očitava sa steka i smešta u programski brojač.

Višestruki izvori prekida

- U praksi je čest slučaj da postoje višestruki izvori prekida kao što su eksterni signali, serijski port, tajmeri i sl.
- U slučaju višestrukih izvora prekida, podsistem za prekide mora da obezbedi:
  - Potprograme (rutine) za obradu različitih prekida
  - Prepoznavanje izvora prekida
  - Rešavanje prioriteta prekida u slučaju istovremenog aktiviranja više prekidnih signala
  - Prelazak na rutinu koja odgovara izvoru prekida
  - Postupak u slučaju da se tokom obrade jednog prekida aktivira neki drugi signal prekida
- Svakom izvoru prekida pridružena su 2 bita posebne namene, koji pripadaju specijalizovanim kontrolnim registrima:
  - **Bit za maskiranje (dozvolu) prekida**, (engl. *Interrupt Enable*) - da bi prekid bio dozvoljen, ovaj bit mora biti setovan. Pored toga, uobičajeno je da postoji bit za globalnu zabranu prekida, čijim aktiviranjem se svi prekidi istovremeno zabranjuju.
  - **Indikator zahteva za prekidom** (engl. *Interrupt Flag*) - Ovaj bit se automatski setuje kada se desi događaj koji zahteva prekid.

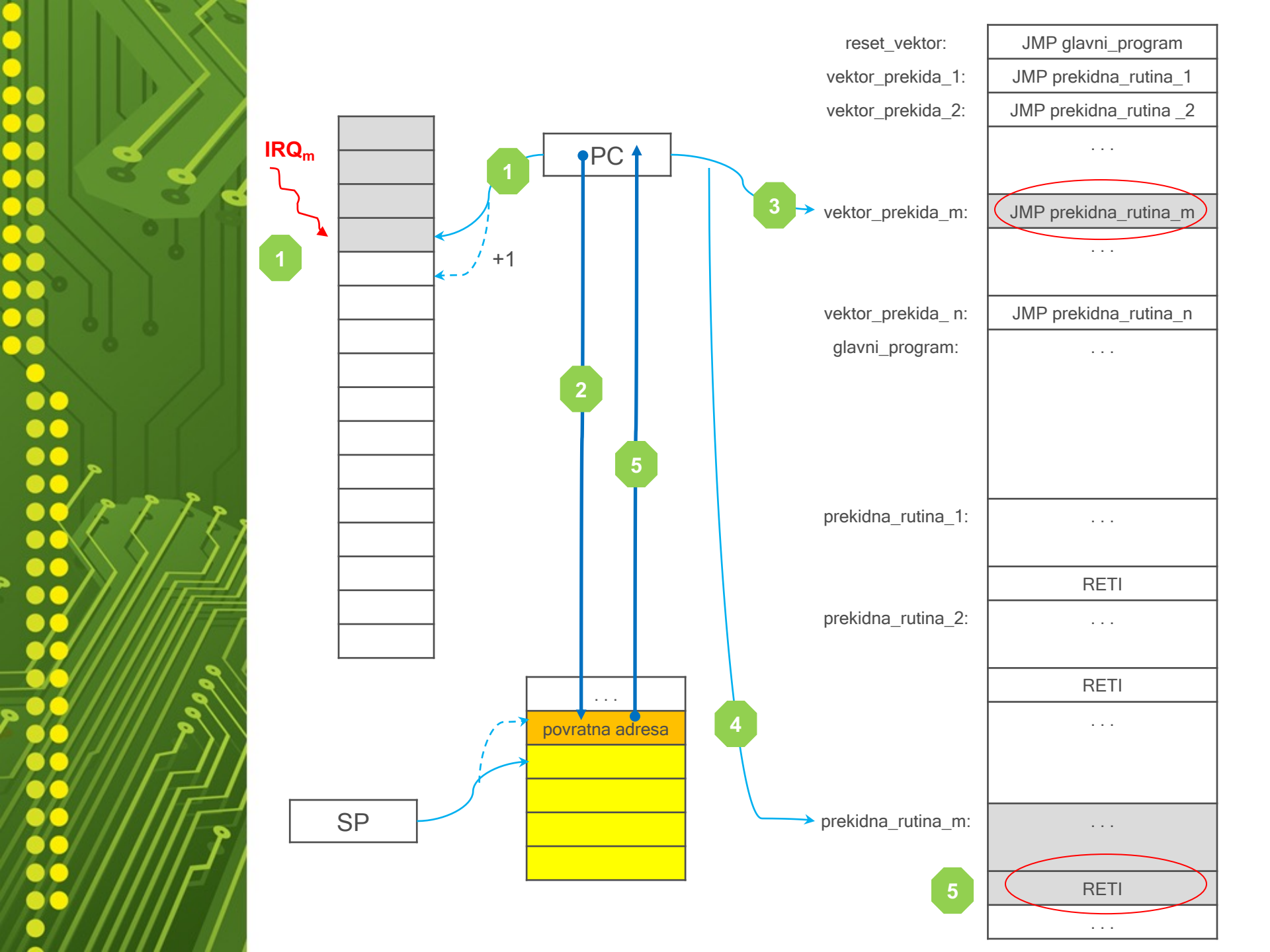
Prelazak na potprogram za obradu prekida

Na najnižim adresama u programskoj memoriji nalaze se tzv. vektori prekida.

• Ukoliko je prekid dozvoljen (setovan je bit I u SREG registru i odgovarajući bit za dozvolu prekida), po pojavi zahteva za prekidom automatski se obavljaju sledeće akcije:
o Bit I se automatski resetuje, čime dolazi do automatske zabrane svih ostalih prekida. Da bi bili omogućeni višestruki ugneždeni prekidi, potrebno je softverski setovati bit I u okviru prekidne rutine.
o Indikator zahteva za prekidom se automatski resetuje.
o Trenutna vrednost programskog brojača (PC) se postavlja na stek.
o U PC se upisuje unapred određena adresa u okviru tabele vektora prekida. Na toj adresi nalazi se skok (instrukcija JMP) na početnu adresu prekidne rutine.

• Preporuka je da se na početku prekidne rutine trenutni kontekst (tj. sadržaji svih registara koji će u okviru nje biti korišćeni, uključujući statusni registar), bude sačuvan na steku. Na kraju prekidne rutine, obavlja se restauracija konteksta, odnosno povratak sadržaja registara koji su bili sačuvani na steku.

• Na kraju prekidne rutine nalazi se mašinska instrukcija RETI, koja obavlja dve akcije:
o Automatski setuje bit I registra SREG.
o Očitava povratnu adresu sa steka i upisuje je u PC.


IRQm
1
1
+1
PC
2
3
4
5
5
. . .
povratna adresa
SP
reset_vektor:
JMP glavni_program
vektor_prekida_1:
JMP prekidna_rutina_1
vektor_prekida_2:
JMP prekidna_rutina _2
. . .
vektor_prekida_m:
JMP prekidna_rutina_m
. . .
vektor_prekida_ n:
JMP prekidna_rutina_n
glavni_program:
. . .
prekidna_rutina_1:
. . .
RETI
prekidna_rutina_2:
. . .
RETI
. . .
prekidna_rutina_m:
. . .
RETI
. . .

Sistem prekida - ATmega328P

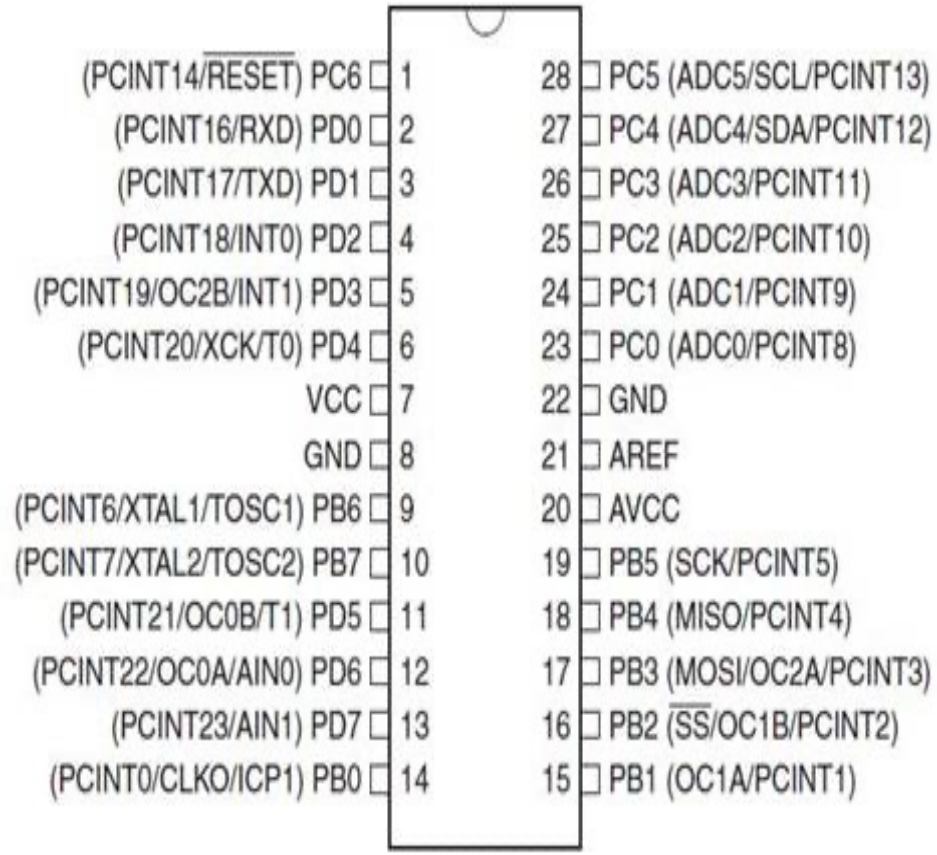
Vector No.	Program Address	Source	Interrupt Definition
1	0x0000	RESET	External pin, power-on reset, brown-out reset and watchdog system reset
2	0x002	INT0	External interrupt request 0
3	0x0004	INT1	External interrupt request 1
4	0x0006	PCINT0	Pin change interrupt request 0
5	0x0008	PCINT1	Pin change interrupt request 1
6	0x000A	PCINT2	Pin change interrupt request 2
7	0x000C	WDT	Watchdog time-out interrupt
8	0x000E	TIMER2 COMPA	Timer/Counter2 compare match A
9	0x0010	TIMER2 COMPB	Timer/Counter2 compare match B
10	0x0012	TIMER2 OVF	Timer/Counter2 overflow
11	0x0014	TIMER1 CAPT	Timer/Counter1 capture event
12	0x0016	TIMER1 COMPA	Timer/Counter1 compare match A
13	0x0018	TIMER1 COMPB	Timer/Counter1 compare match B
14	0x001A	TIMER1 OVF	Timer/Counter1 overflow
15	0x001C	TIMER0 COMPA	Timer/Counter0 compare match A
16	0x001E	TIMER0 COMPB	Timer/Counter0 compare match B
17	0x0020	TIMER0 OVF	Timer/Counter0 overflow
18	0x0022	SPI, STC	SPI serial transfer complete
19	0x0024	USART, RX	USART Rx complete
20	0x0026	USART, UDRE	USART, data register empty
21	0x0028	USART, TX	USART, Tx complete
22	0x002A	ADC	ADC conversion complete
23	0x002C	EE READY	EEPROM ready
24	0x002E	ANALOG COMP	Analog comparator
25	0x0030	TWI	2-wire serial interface
26	0x0032	SPM READY	Store program memory ready

- **25 vektora prekida** i **reset vektor** (na adresi 0x0000 u programskoj memoriji)
- Tabela reset vektora i vektora prekida
- Tabela vektora prekida predefinisana da pokazuje na IR (interrupt routines) sa predefinisanim imenima

Kostur programa i prekidne rutine

```
Adresa   Labela      Kod              Komentar

0x0000               jmp RESET        ;skok na pocetak glavnog programa
0x0002               jmp EXT_INT0     ;skok na rutinu eksternog prekida 0
0x0004               jmp EXT_INT1     ;skok na rutinu eksternog prekida 1

...

0x0032               jmp SPM_READY
0x0034   RESET:      ...              ;glavni program

...

         EXT_INT0:   push SREG        ;kontekst (sadrzaji registara
                     push Rx          ;koji ce biti menjani)
                     push Ry          ;cuva se na steku
                     ...
                     sei              ;I <- 1 dozvola ugnezdenih prekida
                                      ;(opciono)

                     ...              ;telo prekidne rutine

                     pop Ry           ;restauracija konteksta
                     pop Rx
                     pop SREG
                     reti             ;povratak iz prekidne rutine
```

Spoljašnji prekidi - ATmega328P



- Mikrokontroler ATmega328P podržava **spoljašnje prekide** koji se mogu pojedinačno dozvoliti postavljanjem bita INT1 (pin D3) i INT0 (pin D2) u okviru.
- **EIMSK** (*External Interrupt Mask Register*) registra

- INT0 i INT1 prekidi mogu biti trigerovani niskim logičkim nivoom, promenom logičkog nivoa (rastuća ili opadajuća ivica)

- Navedena podešavanje se vrše korišćenjem **EICRA** (**External Interrupt Control Register A**) Registra.

Bit	7	6	5	4	3	2	1	0	
(0x69)	–	–	–	–	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

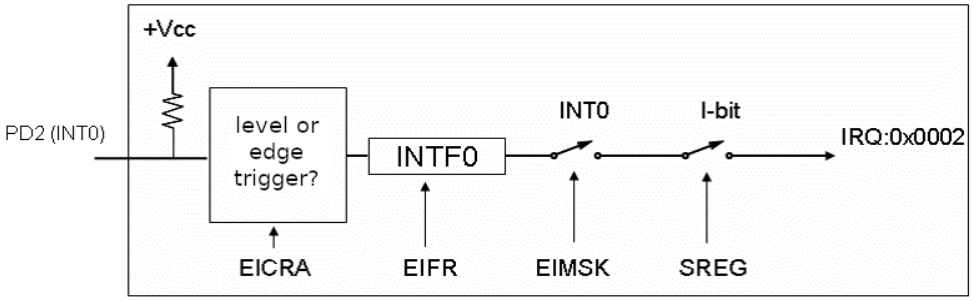
Spoljašnji prekidi - ATmega328P

Table 12-1. Interrupt 1 Sense Control

ISC11	ISC10	Description
0	0	The low level of INT1 generates an interrupt request.
0	1	Any logical change on INT1 generates an interrupt request.
1	0	The falling edge of INT1 generates an interrupt request.
1	1	The rising edge of INT1 generates an interrupt request.

Table 12-2. Interrupt 0 Sense Control

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Any logical change on INT0 generates an interrupt request.
1	0	The falling edge of INT0 generates an interrupt request.
1	1	The rising edge of INT0 generates an interrupt request.



Spoljašnji prekidi - ATmega328P

- Svi prekidi se pojedinačno postavljaju preko individualnih bitova dozvole upisom logičke 1, kao I dozvolom prekida preko **Global Interrupt Enable (I)** bit-a u okviru statusnog registra (SREG)

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Pojedinačna dozvola postavljanja bita INT1 i INT0 u okviru **EIMSK** (*External Interrupt Mask Register*) registra.

Bit	7	6	5	4	3	2	1	0	
0x1D (0x3D)	–	–	–	–	–	–	INT1	INT0	EIMSK
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Spoljašnji prekidi - ATmega328P

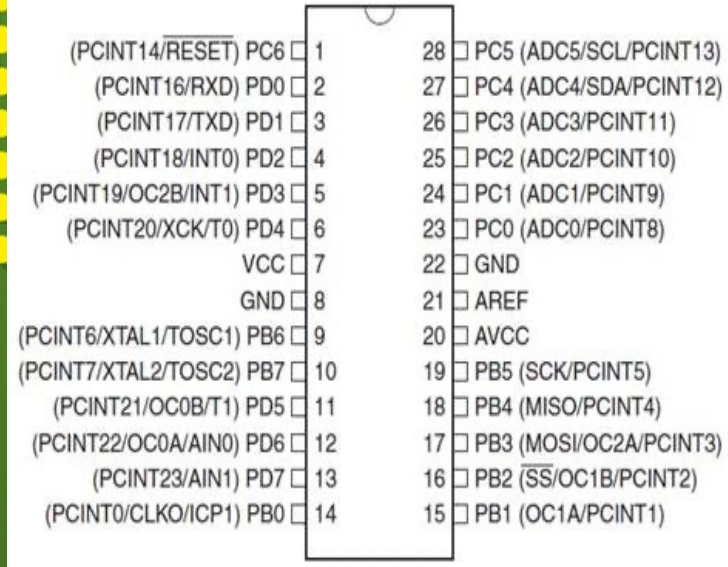
- Kada promena ivice ili logičkog nivoa na INT0 pinu trigeruje zahtev za prekidom, **INF0** se setuje na jedinicu. Ako su I-bit u SREG registru i INT0 bit u EIMSK registru setovani (jedinica), MCU će skočiti na odgovarajući vektor prekida.

- Fleg se briše kada se završi izvršavanje prekidne rutine.

- **EIFR** (*External Interrupt Flag Registar) registar*

Bit	7	6	5	4	3	2	1	0	
0x1C (0x3C)	–	–	–	–	–	–	INTF1	INTF0	EIFR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

PCINT prekidi mikrokontrolera ATmega328P



Port B Group	PCINT7 (PB7) ⇒ PCINT0 (PB0)
Port C Group	PCINT14 (PC6) ⇒ PCINT8 (PC0)
Port D Group	PCINT23 (PD7) ⇒ PCINT16 (PD0)

- **PCINT- Pin Change Interrupts**
- Prekidi usled promene na pinu
- Dodatno pored dva spoljašnja prekida, 23 pina mogu biti programirana da trigeruju prekid ako je došlo do promene stanja na pinovima.
- 23 pina su podeljena u **3 grupe** (PCI 2:0) koje odgovaraju GPIO portovima B, C i D
- Svakoj grupi je dodeljen jedan prekidni flag **PCIF (Pin Change Interrupt Flag)** bit (2:0)
- **PCIF** će biti setovan ako je prekid dozvoljen i ako je bilo koji pin koji je pripada datoj grupi promenio stanje.

Bit	7	6	5	4	3	2	1	0	
0x1B (0x3B)	–	–	–	–	–	PCIF2	PCIF1	PCIF0	PCIFR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

PCINT prekidi mikrokontrolera ATmega328P

- **PCICR-Pin Change Interrupt Control Register**

Bit	7	6	5	4	3	2	1	0	
(0x68)	–	–	–	–	–	PCIE2	PCIE1	PCIE0	PCICR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **PCIE2**- kada je **PCIE2 bit setovan** (postavljen na 1) i **I-bit statusnog registra** (SREG) setovan (1), **PCI2** (pin change interrupt) je dozvoljen, svaka promena na bilo kom dozvoljenom PCINT23..16 pinu će dovesti do prekida. Odgovarajući prekid koji pripada zahtevu za prekid na osnovu promene pina (pin change interrupt request) se izvršava iz PCI2 prekidnog vektora. PCINT23..16 pinovi se individualno dozvoljavaju preko PCMSK2 registra.

- **PCIE1**- kada je PCIE1 bit setovan (postavljen na 1) i I-bit statusnog registra (SREG) setovan (1), PCI 1 je dozvoljen, svaka promena na bilo kom dozvoljenom PCINT14..8 pinu će dovesti do prekida. Odgovarajući prekid koji pripada zahtevu za prekid na osnovu promene pina (pin change interrupt request) se izvršava iz PCI1 prekidnog vektora. PCINT14..8 pinovi se individualno dozvoljavaju preko PCMSK1 registra.

- **PCIE0**- kada je PCIE0 bit setovan (postavljen na 1) i I-bit statusnog registra (SREG) setovan (1), PCI 0 je dozvoljen, Svaka promena na bilo kom dozvoljenom PCINT7..0 pinu će dovesti do prekida. Odgovarajući prekid koji pripada zahtevu za prekid na osnovu promene pina (pin change interrupt request) se izvršava iz PCI0 prekidnog vektora. PCINT7..0 pinovi se individualno dozvoljavaju preko PCMSK0 registra.

PCINT prekidi mikrokontrolera ATmega328P

- **PCIFR-Pin Change Interrupt Flag Register**

Bit	7	6	5	4	3	2	1	0	
0x1B (0x3B)	–	–	–	–	–	PCIF2	PCIF1	PCIF0	PCIFR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- 7..3 rezervisani biti, biti koji se ne koriste za Atmel ATmega328P i biće očitani kao vrednost 0.
- **PCIF2** – kada nastupi logička promena na bilo kom od **PCINT23..16** pin trigeruje zahtev za prekidom, PCIF2 se setuje (1). Ako su **I-bit SREG** i **PCIE2** bit u okviru **PCICR** setovani (1), MCU će skočiti na izvršavanje odgovarajućeg prekidnog vektora. Fleg se briše kada je završena prekidna rutina. Alternativno, fleg se može obrisati upisom logične 0.
- **PCIF1** – kada nastupi logička promena na bilo kom od PCINT14..8 pin trigeruje zahtev za prekidom, PCIF1 se setuje (1). Ako su I-bit SREG i PCIE1 bit u okviru PCICR setovani (1), MCU će skočiti na izvršavanje odgovarajućeg prekidnog vektora. Fleg se briše kada je završena prekidna rutina. Alternativno, fleg se može obrisati upisom logične 0.
- **PCIF0** – kada nastupi logička promena na bilo kom od PCINT7..9 pin trigeruje zahtev za prekidom, PCIF0 se setuje (1). Ako su I-bit SREG i PCIE0 bit u okviru PCICR setovani (1), MCU će skočiti na izvršavanje odgovarajućeg prekidnog vektora. Fleg se briše kada je završena prekidna rutina. Alternativno, fleg se može obrisati upisom logične 0.

PCINT prekidi mikrokontrolera ATmega328P

- **PCMSK2- Pin Change Mask Register 2**

Bit	7	6	5	4	3	2	1	0	
(0x6D)	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	PCMSK2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Bit 7..0-PCINT 23..16: Pin Change Enable Mask 23:16

Svaki PCINT23..16 bit selekuje da li je PCI dozvoljen sa odgovarajućeg I/O pina. Ako je **PCINT23..16** setovan i **PCIE2** bit u okviru PCICR setovan, PCI je dozvoljen sa odgovarajućeg I/O pina. Ako je PCINT23..16 obrisan, PCI sa odgovarajućeg I/O pina je zabranjen.

- **PCMSK1- Pin Change Mask Register 1**

Bit	7	6	5	4	3	2	1	0	
(0x6C)	–	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	PCMSK1
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Bit 7– Res: Rezervisani Bit
- Bit 6..0-PCINT 14..8: Pin Change Enable Mask 7..0

Svaki PCINT14..8 bit selektuje da li je PCI dozvoljen sa odgovarajućeg I/O pina. Ako je **PCINT14..8 setovan** i PCIE1 bit u okviru PCICR setovan, PCI je dozvoljen sa odgovarajućeg I/O pina. Ako je PCINT14..8 obrisan, PCI sa odgovarajućeg I/O pina je zabranjen.

PCINT prekidi mikrokontrolera ATmega32

- **PCMSK0- Pin Change Mask Register 0**

Bit	7	6	5	4	3	2	1	0	
(0x6B)	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	PCMSK0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Bit7..0-PCINT 7..0: Pin Change Enable Mask 7..0

Svaki PCINT7..0 bit selektuje da li je PCI dozvoljen sa odgovarajućeg I/O pina. Ako je **PCINT7..0** setovan i PCIE0 bit u okviru PCICR setovan, PCI je dozvoljen sa odgovarajućeg I/O pina. Ako je PCINT7..0 obrisan, pci sa odgovarajućeg I/O pina je zabranjen.

PCINT prekidi mikrokontrolera ATmega32