

```
entity test_shift is
  generic ( width : integer := 17 );
  port ( clk : in std_ulogic;
        reset : in std_ulogic;
        load : in std_ulogic;
        en : in std_ulogic;
        outp : out std_ulogic );
end test_shift;
```

Mikroprocesorska elektronika

```
shifter : process (clk, reset)
begin
  if (reset = '0') then
    shift_reg <= (others => '0');
  elsif rising_edge (clk) then
    if (load = '1') then
      shift_reg <= unsigned (inp);
    elsif (en = '1') then
```

Sadržaj predavanja

- Tajmeri i brojači
- Tajmeri i brojači unutar mikrokontrolera 8051

```
shift_reg <= unsigned (inp);  
elsif ( en = '1' ) then
```

```
entity test_shift is
  generic ( width : integer := 17 );
  port ( clk : in std_ulogic;
        reset : in std_ulogic;
        load : in std_ulogic;
        en : in std_ulogic;
        outp : out std_ulogic );
end test_shift;
```

Tajmeri i brojači

```
shifter : process ( clk, reset )
begin
  if ( reset = '0' ) then
    shift_reg <= (others => '0');
  elsif rising_edge ( clk ) then
    if ( load = '1' ) then
      shift_reg <= unsigned ( inp );
    elsif ( en = '1' ) then
```

Tajmeri i brojači I

- Tajmeri i brojači su jedni od najkorisnijih i najčešće prisutnih hardverskih resursa unutar mikrokontrolera
- Tajmeri i brojači su značajni zbog toga što većina embeded sistema radi u realnom vremenu i reaguje na događaje u ovom okruženju te je stoga neophodno imati mogućnost merenja proteklog vremena, odnosno broja detektovanih događaja
- Neke od najčešće korišćenih aplikacija tajmera i brojača su:
 - Watchdog tajmeri
 - Interval tajmeri
 - Brojači događaja
 - Satovi realnog vremena
 - Generatori PWM (Pulse Width Modulation) signala
 - Baud rate generatori

```
shift_reg <= unsigned(inp);  
elsif ( en = '1' ) then
```

Tajmeri i brojači II

- Mikrokontroleri obično uključuju jedan ili više tajmerskih modula kao periferijske jedinice koje su integrisane na istom čipu sa procesorom
- Tajmeri su obično visoko konfigurabilni, kako bi ih korisnik lako mogao prilagoditi svojoj konkretnoj potrebi
- Tajmeri po pravilu mogu da generišu zahteve za prekidom
- Neki od tajmera imaju mogućnost čuvanja podatka o tačnom vremenskom trenutku pojave odgovarajućeg događaja u sistemu
- Koristeći tajmere i pravilno ih konfigurirajući, projektant embeded sistema može u značajnoj meri rasteriti centralni procesor dugačke liste zadataka koji uključuju merenje i manipulaciju vremenom

```
shift_reg <= unsigned (inp);  
elsif ( en = '1' ) then
```

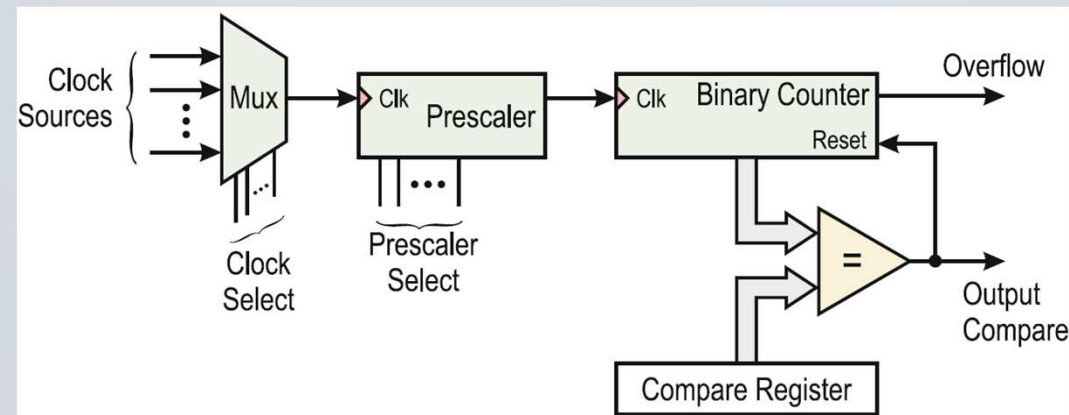

Princip rada tajmera

- U najosnovnijoj formi, tajmer je običan brojač koji se okida periodičnim klok signalom
- Osnovni princip rada svakog tajmera je sledeći: Počevši od neke proizvoljne vrednosti brojačkog registra, tipično od nule, brojač se inkrementuje (uvećava za jedan) svaki put kada se na klok signalu pojavi rastuća (opadajuća) ivica
- Na ovaj način brojač zapravo broji broj rastućih (opadajućih) ivica klok signala
- Po pravilu, tajmeri su osetljivi ili na rastuću ili na opadajuću ivicu klok signala, nikada na obe
- Ukoliko je klok signal periodičan, sa poznatom periodom T , tada broj odbrojanih ivica u tajmeru k , zapravo može da se interpretira kao proteklo vreme kT od trenutka startovanja tajmera (trenutak 0) do pojave poslednjeg događaja

```
shift_reg <= unsigned(inp);  
elsif ( en = '1' ) then
```

Osnovna struktura tajmera

- Da bi se povećale mogućnosti tajmera, pored brojačkog registra on obično sadrži i dodatne module koji čine arhitekturu odgovarajućeg tajmera
- Fundamentalne komponente od kojih se sastoji većina tajmera su:
 - **Selektor klok signala (Mux)** – dozvoljava izbor jednog od više mogućih klok signala
 - **Preskaler** – obezbeđuje mogućnost deljenja učestanosti klok signala (usporavanja klok signala) pre nego što on uđe u brojač
 - **N-bitni brojač** – koji obezbeđuje osnovnu brojačku funkciju
 - **N-bitni komparatorski registar** – koji omogućuje da se definiše maksimalna vrednost koju brojač može dostići
 - **N-bitni komparator** – koji omogućava detekciju trenutka kada brojač dostiže vrednost koja je smeštena u komparatorskom registru. Po pravilu, dostizanjem ove vrednosti automatski se vrđši resetovanje brojača.



```
shift_reg <= unsigned(inp);  
elsif ( en = '1' ) then
```

Selekcija kloka i preskaler I

- U opštem slučaju, klok signal tajmera može da potiče od raznih izvora unutar embeded sistema koji uključuju unutrašnje i spoljašnje izvore klok signala ili čak neke asinhronne izvore događaja koji se koriste kao klok signal
- Modul za selekciju klok signala daje mogućnost projektantu embeded sistema da na jedan isti tajmerski modul dovede čitav niz različitih izvora klok signala i da u toku rada menja koji od njih zapravo okida tajmer
- U većini mikrokontrolera klok signali tajmera se izvode (generišu) iz sistemskog klok signala
- U mnogim aplikacijama učestanost sistemskog klok signala je previsoka da bi omogućila potrebno merenje proteklog vremena, jer bi zahtevala tajmer sa vrlo velikim brojačkim registrom

```
shift_reg <= unsigned (inp);  
elsif ( en = '1' ) then
```


Selekcija kloka i preskaler II

- Upravo zbog toga javlja se potreba za smanjivanjem učestanosti sistemskog klok signala za potrebe korišćenja unutar tajmera
- Funkcija preskalera, odnosno delitalja ulaznog takta, je upravo usporavanja ulaznog klok signala za zadati faktor skaliranja
- Po pravilu, faktor skaliranja se može zadati softverski upisom u odgovarajući registar koji se nalazi unutar tajmera
- Izlaz preskalera, koji predstavlja klok signal ali sada sa manjom frekvencijom od frekvencije ulaznog klok signala, se zatim vodi na ulaz brojačkog modula

```
shift_reg <= unsigned (inp);  
elsif ( en = '1' ) then
```

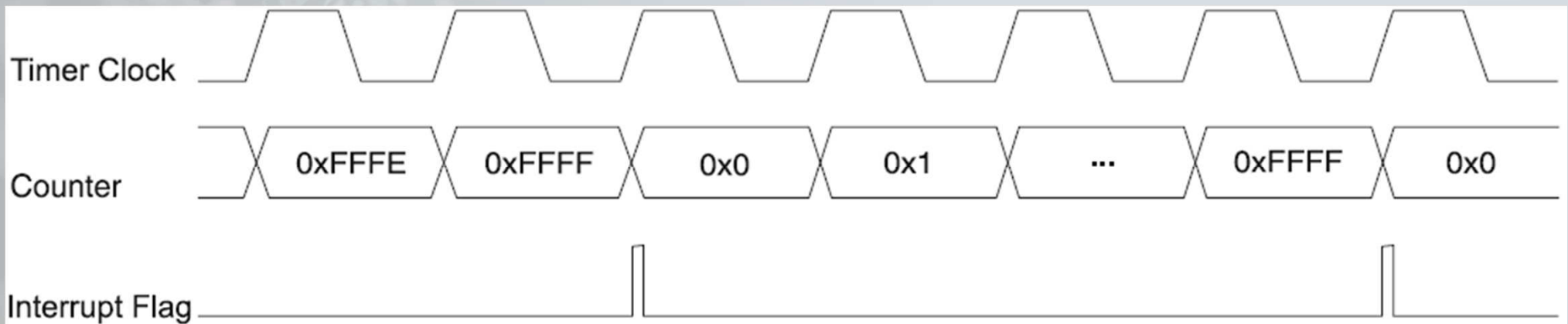
Brojački i komparatorski moduli I

- Brojački registar je najznačajnija komponenta svakog tajmerskog modula
- Brojački registar je često iste veličine kao i ostali unutrašnji registri mikrokontrolera, mada to ne mora uvek da bude slučaj
- U većini tajmera, brojački registar je istovremeno i registar sa mogućnošću upisa i čitanja, što omogućava programeru da postavlja vrednost brojačkog registra na proizvoljne vrednosti kao i da čita trenutnu vrednost koja je smeštena u brojačkom registru
- Brojač menja svoje stanje kada detektuje odgovarajući događaj (rastuću ili opadajuću ivicu) na ulaznom klok signalu, uvećavajući vrednost brojačkog registra za jedan
- Na ovaj način brojač “broji” broj događaja na ulaznom klok signalu

```
shift_reg <= unsigned(inp);  
elsif ( en = '1' ) then
```

Brojački i komparatorski moduli II

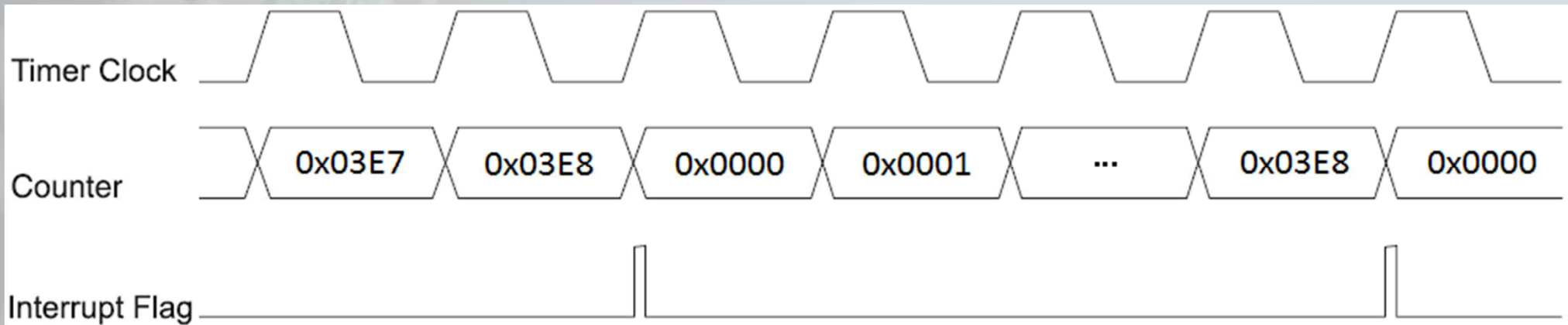
- Obzirom da je brojački registar realizovan kao N -bitni registar, brojač broji od vrednosti 0 do vrednosti 2^N-1
- Kada dostigne vrednost 2^N-1 brojač generiše signal prekoračenja opsega (overflow) i započinje brojanje ponovo od vrednosti 0
- Ovaj overflow signal se obično može konfigurisati da izazove pojavu zahteva za prekidom
- Na slici dole prikazan je rad jednog 16-bitnog brojača, gde je takođe prikazano ponašanje signala zahteva za prekidom koji se aktivira kada se u brojaču desi prekoračenje opsega (prelazak sa vrednosti 65535 na vrednost 0)



```
shift_reg <= unsigned (inp);  
elsif ( en = '1' ) then
```

Brojački i komparatorski moduli III

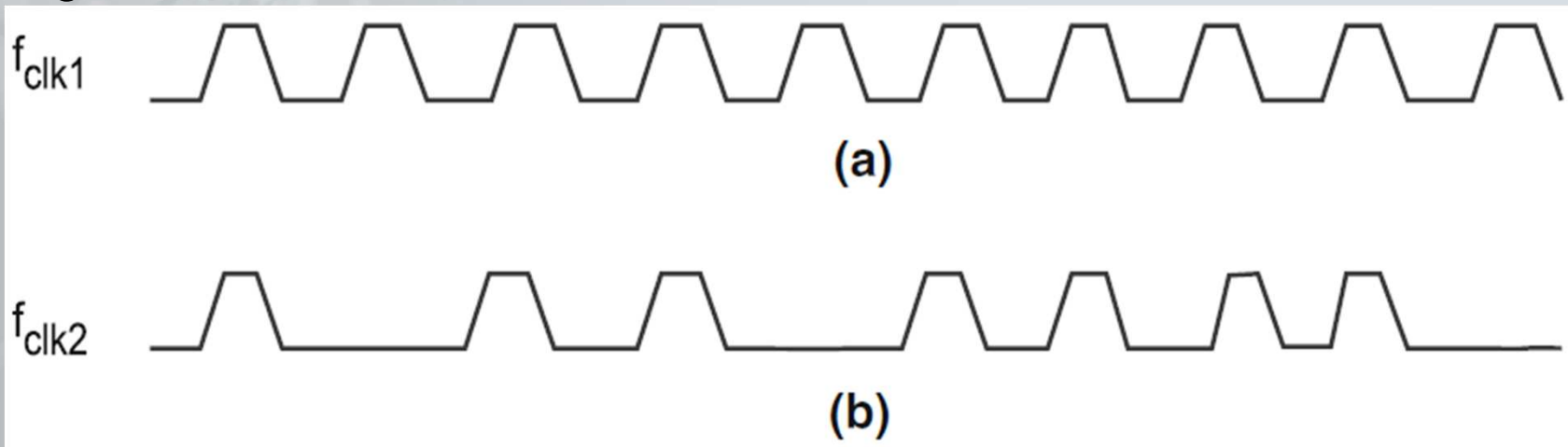
- U većini aplikacija javlja se potreba za limitiranjem maksimalne vrednosti koju brojač može dostići
- U takvim slučajevima potrebno je postojanje komparatorskog registra i komparatora kao sastavnih delova tajmera
- Korišćenjem ova dva modula može se generisati signal koji će biti aktivan kada brojač dostigne vrednost koja je specificirana u komparatorskom registru
- Ovaj signal može zatim generisati odgovarajući zahtev za prekidom i resetovati brojač, kao što je prikazano na slici, u slučaju da je vrednost komparatorskog registra jednaka 1000 (0x03E8)



```
shift_reg <= unsigned (inp);  
elsif ( en = '1' ) then
```

Osnovne primene tajmera: Brojač događaja I

- U ovom režimu tajmer prosto broji broj detektovanih događaja na svom ulaznom klok signalu
- Obratite pažnju da u ovoj primeni ulazni klok signal na neki način mora da bude u vezi sa događajem čiji broj realizacije se broji
- Zbog ove specifične veze sa izvorom događaja, u opštem slučaju, klok signal neće i ne mora da ima osobinu periodičnosti, koja je inače karakteristična za klok signale



```
shift_reg <= unsigned(inp);  
elsif (en = '1') then
```


Osnovne primene tajmera: Brojač događaja II

- Način rada brojača događaja biće ilustrovan na sledećem primeru
- Razmotrimo aplikaciju gde imamo potrebu za brojanjem broja ljudi koji su prošli kroz odgovarajuća vrata
- U ovom slučaju na vrata bi bio postavljen odgovarajući senzor koji bi generisao impuls svaki put kada neka osoba prođe pored njega
- Ovaj impulsni signal bi se zatim vodio na klock ulaz tajmera
- U ovoj konfiguraciji, prolazak svake osobe bi izazvao uvećavanje vrednosti brojačkog registra unutar tajmera za jedan

```
shift_reg <= unsigned (inp);  
elsif ( en = '1' ) then
```

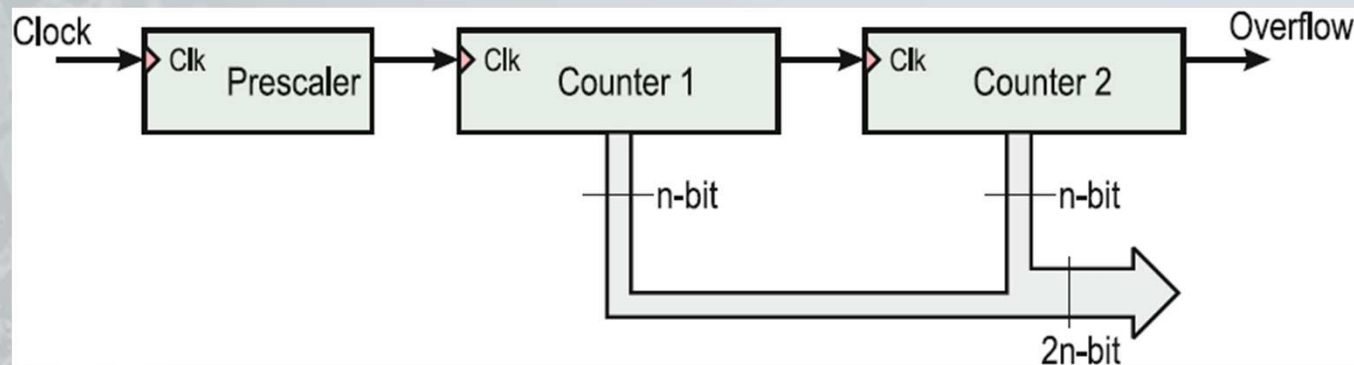
Osnovne primene tajmera: Brojač događaja III

- Obratite pažnju da je u prethodnom primeru pretpostavljeno da je preskaler konfigurisan na takav način da deli ulazni klok (u našem slučaju to je signal sa senzora na vratima) signal sa faktorom 1
- Ovo je važan detalj, jer u slučaju da je preskaler bio konfigurisan da deli ulazni klok sa faktorom p , brojački registar bi uvećao svoju vrednost za jedan nakon što je senzor detektovao prolazak p ljudi
- Ovaj komentar ukazuje na jednu interesantnu aplikaciju preskalera kada tajmer radi u brojačkom režimu
- U nekim aplikacijama može se pojaviti situacija da je maksimalni broj događaja koji je potrebno detektovati veći od opsega brojanja brojačkog registra tajmera koji nam stoji na raspolaganju

```
shift_reg <= unsigned(inp);  
elsif ( en = '1' ) then
```

Osnovne primene tajmera: Brojač događaja IV

- Navedeni problem može se rešiti na dva načina:
- **Proširenjem brojačkog opsega tajmera**
 - Korišćenjem softverske varijable - vrednost varijable bi se uvećavala za jedan svaki put kada tajmer dostigne svoj maksimum
 - Kaskadnim vezivanjem više tajmerskih modula



- **Korišćenjem preskalera**

- U slučaju korišćenja preskalera, kada bi on bio konfigurisan da deli ulazni klock sa faktorom p , tajmer bi efektivno brojao pojavu setova of p uzastopih događaja

```
shift_reg <= unsigned (inp);  
elsif ( en = '1' ) then
```

Osnovne primene tajmera: Brojač događaja V

- Sledeći primer bliže ilustruje način korišćenja preskalera za povećanje brojačkog opsega tajmera

Primer:

- Preskaler unutar tajmera konfigurisan je da deli ulazni klok sa faktorom 4. Brojač unutar tajmera je 16-bitni.
 - a) Koji je maksimalni broj događaja koji se može izbrojati sa ovako konfigurisanim tajmerom?
 - b) Koji broj treba upisati u komparatorski registar da bi tajmer nakon odbrojavanja 75000 događaja generisao zahtev za prekidom?
 - c) Da li je sa ovako konfigurisanim sistemom moguće izbrojati tačno 83253 događaja?

```
shift_reg <= unsigned (inp);  
elsif ( en = '1' ) then
```

Osnovne primene tajmera: Brojač događaja VI

Rešenje pod a)

- Obzirom da brojač uvećava svoju vrednost nakon što se detektuje 4 uzastopna događaja (jer je preskaler konfigurisan tako da deli ulazni klok signal sa 4), maksimalni broj događaja koji se može izbrojati je

$$65536 \times 4 = 262140 \text{ događaja}$$

Rešenje pod b)

- Brojač se mora resetovati kada odbroji ukupno 75000 događaja. Kako se vrednost brojača uvećava za jedan nakon svaka 4 uzastopna događaja, vrednost koju treba da sadrži komparatorski registar je

$$75000 / 4 = 18750$$

Rešenje pod c)

- Obzirom da vrednost 83253 nije deljiva sa brojem 4, brojač nije u mogućnosti da izbroji tačno 83253 događaja. Mogao bi da izbroji ili 83252 ili 83256 događaja.

```
shift_reg <= unsigned(inp);  
elsif ( en = '1' ) then
```


Osnovne primene tajmera: Merač proteklog vremena I

- Kada ulazni klok tajmera je periodičan signal učestanosti f tajmer se može iskoristiti za merenje proteklog vremena između dva događaja
- Ukoliko je učestanost ulaznog klok signala f Hz, njegova perioda će biti jednaka
Perioda $T = 1/f$ sekundi
- U ovakvom sistemu, kada brojač izbroji k događaja na ulaznom klok signalu, on je istovremeno izmerio vreme od $kT = k/f$ sekundi

Primer 1:

- Oscilator učestanosti 38 kHz koristi se za generisanje klok signala, koji je povezan na ulaz preskalera sa faktorom 16.
 - a) Ako se tajmer resetuje nakon što brojač izbroji vrednost 0x8A39 koliko će vremena proteći između dva reseta?
 - b) Kako se u postojećoj konfiguraciji može dobiti informacija da je proteklo 50 ms? Koje su apsolutne i relativne greške prilikom merenja vremenskog intervala od 50 ms?

```
shift_reg <= unsigned(inp);  
elsif ( en = '1' ) then
```

Osnovne primene tajmera: Merač proteklog vremena II

Rešenje pod a)

- Učestanost i perioda kloka koji zapravo okida brojač unutar tajmera u ovako konfigurisanom sistemu je

$$f_{\text{brojača}} = 38 \text{ kHz} / 16 = 2.375 \text{ kHz}, T_{\text{brojača}} = 1/2.375 \text{ ms} = 421 \text{ us}$$

- Obzirom da je broj perioda ulaznog klok signala koje su bile odbrojane kada je brojač dostigao vrednost 0x8A39 jednak 0x8A39 = 35385, a trajanje svake periode iznosi 421 us, proteklo vreme između dva reseta tajmera iznosi

$$35385 * 421 \text{ us} = 14.9 \text{ sekundi}$$

```
shift_reg <= unsigned(inp);  
elsif ( en = '1' ) then
```

Osnovne primene tajmera: Merač proteklog vremena III

Rešenje pod b)

- Obzirom da je potrebno signalizirati protok vremena od 50 ms, brojač mora da odbroji ukupno

$$(50 \text{ ms}) \times (2.375 \text{ kHz}) = 118.75 \text{ perioda klok signala.}$$

- Obzirom da se u komparatorski registar može smestiti samo celobrojna vrednost, u ovom slučaju u njega je potrebno upisati vrednost 119.
- Stvarni vremenski interval koji će biti izmeren u ovom slučaju iznosi
$$119 \times 421 \text{ us} = 50.105 \text{ ms}$$
- Apsolutna i relativna greška merenja 50 ms vremenskog intervala sa ovako konfigurisanim sistemom iznose 105 us, i 0.21% respektivno.
- U zavisnosti od krajnje aplikacije, ova greška će biti prihvaljiva ili ne

```
shift_reg <= unsigned(inp);  
elsif ( en = '1' ) then
```

Osnovne primene tajmera: Merač proteklog vremena IV

Primer 2:

- Vratimo se na primer brojanja osoba koje su prošle kroz odgovarajuća vrata i pretpostavimo da bi smo želeli da merimo i prosečno vreme koje protekne između prolaska dva osobe kroz vrata

Rešenje:

- Najjednostavnije rešenje bi bilo da se računa prosečno vreme između prolaska dve osobe pomoću odgovarajućeg bafera koji se pomera u vremenu. Na primer, prosečno vreme između prolaska dve osobe izračunato usrednjavajući vremena poslednjih 8 prolazaka.
- U ovom slučaju mogli bi smo koristiti dodatni tajmer, tajmer 2, (jedan se već koristi za brojanje ljudi) koji bi bio klokovan sa klokom pozante učestanosti
- Koristeći izlaz senzora za detekciju prolaska osoba kroz vrata softver bi na svaki zahtev za prekidom pročitao tekuće stanje tajmera 2

```
shift_reg <= unsigned(inp);  
elsif ( en = '1' ) then
```

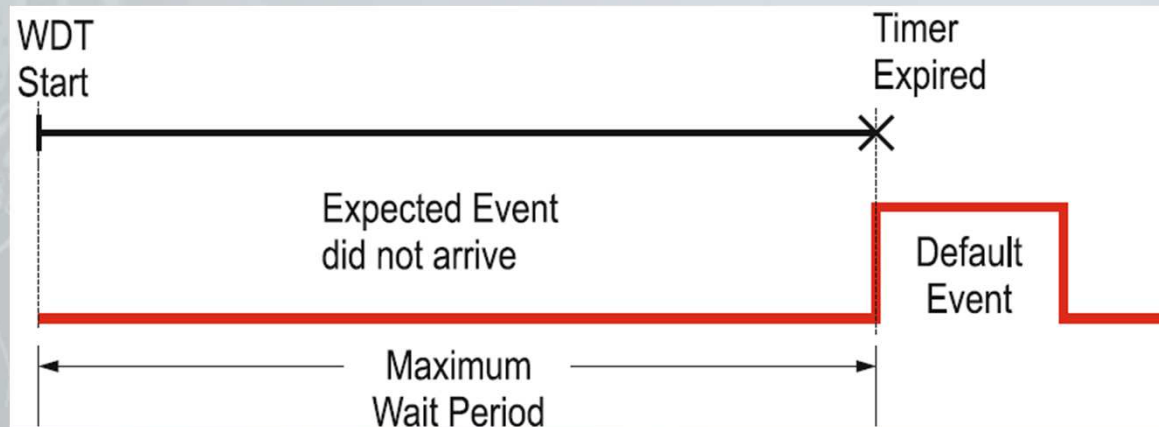
Osnovne primene tajmera: Merač proteklog vremena V

- Oduzimajući trenutnu vrednost tajmera 2 od vrednosti koja je bila očitana prilikom prethodnog generisanja zahteva za prekidom, moguće je izračunati vreme koje je proteklo između prolaska poslednje dve osobe kroz vrata
- Ova vremena između prolazaka dve sukcesivne osobe bi zatim bila smeštana u jedan cirkularni bafer dubine 8
- Računajući prosečnu vrednost vremena koja se trenutno nalaze unutar cirkularnog bafera mogli bi smo izračunati prosečno vreme između prolazaka osoba kroz vrata

```
shift_reg <= unsigned (inp);  
elsif ( en = '1' ) then
```


Naprednije primene tajmera: Watchdog tajmer I

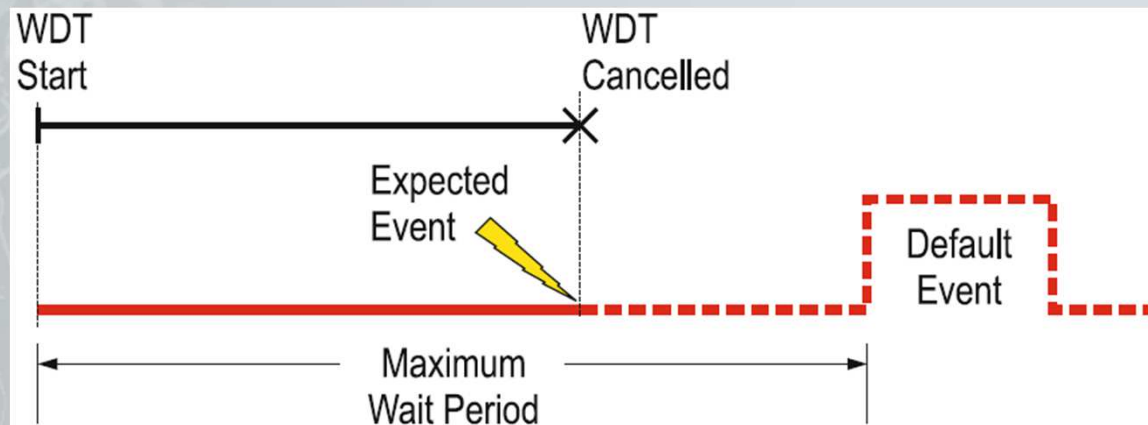
- Watchdog tajmer je posebna vrsta tajmera koji meri proteklo vreme između dva događaja
- Watchdog tajmer se koristi da generiše odgovarajuću defoltnu akciju unutar sistema, na primer da generiše zahtev za prekidom ili da resetuje čitav sistem ili neki njegov deo, u slučaju da istekne unapred definisani period vremena a ne pojavi se događaj koji treba da reinicijalizuje watchdog tajmer



```
shift_reg <= unsigned (inp);  
elsif ( en = '1' ) then
```

Naprednije primene tajmera: Watchdog tajmer II

- Maksimalno dozvoljeno vreme za pojavu događaja koji će reinicijalizovati watchdog tajmer se podešava softverski
- Podrazumevana akcija će se izvršiti u slučaju da istekne unapred definisani period vremena a ne dodje do pojave događaja koji reinicijalizuje watchdog tajmer
- Da bi smo sprečili watchdog tajmer da aktivira podrazumevanu akciju, nakon detekcije pojave odgovarajućeg događaja potrebno je reinicijalizovati watchdog tajmer
- Ovo se najčešće radi unutar prekidnog podprograma koji je asociran događaju koji treba da aktivira reinicijalizaciju watchdog tajmera



```
shift_reg <= unsigned (inp);  
elsif ( en = '1' ) then
```

Naprednije primene tajmera: Watchdog tajmer III

- Kao primer korišćenja watchdog tajmera razmotrimo sledeću aplikaciju: Pretpostavimo da imamo sigurnosna vrata koja se otključavaju pomoću elektronske kartice.
- Da bi se vrata otključala, korisnik mora da provuče karticu kroz čitač. Ova akcija otključava vrata da bi korisnik mogao da ih otvori i uđe u zaštićenu prostoriju
- Međutim, ukoliko se nakon nekon unapred definisanog perioda vremena, na primer 30 sekundi, vrata ne otvore brava koja ih zaključava će se opet aktivirati
- U ovom primeru, WDT se startuje u trenutku provlačenja kartice kroz čitač
- Podrazumevana akcija koje se izvršava ukoliko WDT dostigne svoju predefinisanu vrednost (30 sekundi) je zaključavanje vrata
- Očekivani događaj koji reinicijalizuje WDT je otvaranje vrata

```
shift_reg <= unsigned(inp);  
elsif ( en = '1' ) then
```

Naprednije primene tajmera: Watchdog tajmer IV

- Watchdog tajmer je vrsta sigurnosnog uređaja
- U normalnim okolnostima, podrazumeva se da će watchdog tajmer biti servisiran (reinicijalizovan, zaustavljen, resetovan) pre isteka definisanog perioda
- U protivnom, nešto nije u redu sa sistemom i potrebno je izvršiti odgovarajući oporavak sistema izvršavanjem podrazumevane akcije
- U embeded sistemima, watchdog tajmeri koriste se u različite svrhe:
 - Za zaustavljanje programa koji odstupaju od normalnog režima rada
 - Za izlazak iz mrtvih petlji unutar programa
 - Za detekciju nepravilnih transakcija na sistemskim magistralama
 - ...

```
shift_reg <= unsigned (inp);  
elsif ( en = '1' ) then
```

Naprednije primene tajmera: Merač realnog vremena

- Jedna od načešćih aplikacija tajmera unutar embeded sistema jeste merenje protoka realnog vremena
- Merač protoka realnog vremena (Real-Time Clock, RTC) je tajmer koji je konfigurisan da meri prolazak sekundi, minuta, sati, itd.
- Vrlo često, potrebna rezolucija RTC modula mora se spustiti na delove sekunde, i takvi RTC moduli se često nazivaju hronometrima (chronometer)
- Sa druge strane, nekada je potrebno meriti vremena koja se mere danima, nedeljama, mesecima, pa čak i godinama, kada se RTC moduli nazivaju RTC kalendarima (Real-Time Clock Calendar, RTCC)
- Iako se funkcionalnost RTCC u potpunosti može realizovati softverski, postojanje posebnog tajmera namenjenog za ove svrhe može uštedeti veliki CPU ciklusa koji se zatim mogu iskoristiti na nekom drugom mestu

```
shift_reg <= unsigned(inp);  
elsif ( en = '1' ) then
```


Naprednije primene tajmera: Baud Rate generator

- Tajmeri se takođe mogu iskoristiti za generisanje periodičnih signala koji su neophodni u serijskim komunikacionim protokolima da bi se obezbedila željena brzina razmene podataka
- U ovim aplikacijama, frekvencija ulaznog klok signala se deli sa željenom brzinom slanja podataka (*Baud Rate*) da bi se odredio potreban broj brojanja između slanja sukcesivnih bitova, odnosno da bi se odredilo vreme trajanja jednog bita (*Bit Time*)
- Izračunata vrednost se zatim smešta u komparatorski registar tajmera i na taj način se generiše zahtve za prekidom u tačno potrebnim trenucima

```
shift_reg <= unsigned(inp);  
elsif ( en = '1' ) then
```

Naprednije primene tajmera: PWM generator I

- Impulsno širinska modulacija (*Pulse Width Modulation*, PWM) predstavlja čestu aplikaciju unutar embeded sistema koja se može realizovati korišćenjem tajmera
- PWM modul generiše periodičan signal čiji se faktor ispune (*Duty Cycle*) može kontrolisati od strane mikrokontrolera, odnosno softvera
- Obzirom da se i učestanost ulaznog kloka tajmera može kontrolisati softverski (izborom odgovarajućeg izvora klok signala pomoću multipleksera), u ovoj aplikaciji tajmera moguće se softverski kontrolisati i učestanost i faktor ispune generisanog periodičnog signala

```
shift_reg <= unsigned (inp);  
elsif ( en = '1' ) then
```

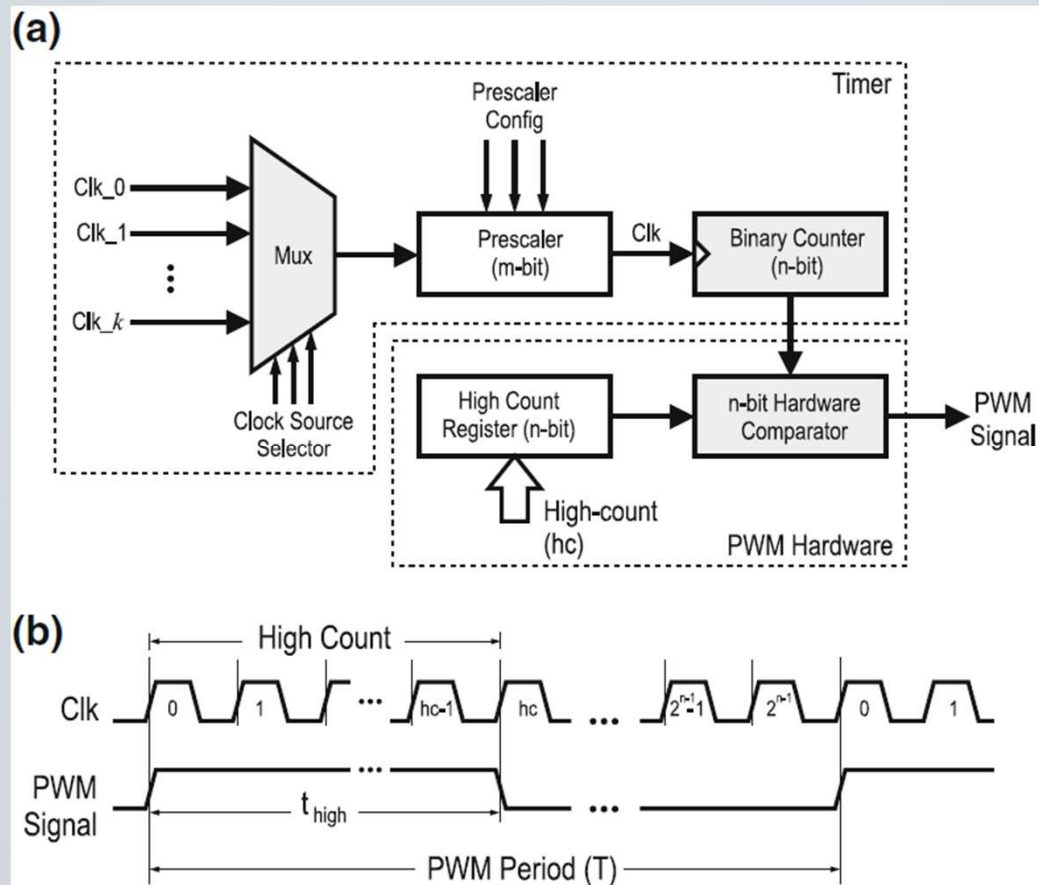
Naprednije primene tajmera: PWM generator II

- PWM signali imaju veliku primenu u sistemima koji treba da kontrolišu količinu energije koja se prenosi ka nekom uređaju
- Energija PWM signala je funkcija njegovog faktora ispune (duty cycle)
- Aplikacije u kojima se koriste PWM signali su:
 - Kontrola brzine DC motora
 - Kontrola temperature grejača
 - Kontrola inteziteta svetlosti LED dioda
 - Itd...

```
shift_reg <= unsigned (inp);  
elsif ( en = '1' ) then
```

Naprednije primene tajmera: PWM generator III

- Struktura tipičnog PWM modula prikazana je na slici (a)
- PWM modul sastoji se iz N -bitnog tajmera (sa selektorom klok signala i preskalerom) čiji se izlaz upoređuje sa sadržajem HC registra (*High Count*)
- Dok je vrednost tajmera manja od HC vrednosti, vrednost PWM signala jednaka je jedinici, u protivnom jednaka je nuli
- Vremenski dijagram PWM signala prikazan je na slici (b)
- Broj bitova (N) unutar brojačkog registra tajmera određuje rezoluciju PWM signala, dok vrednost HC registra određuje širinu impulsa (faktor ispune, t_{high}/T) PWM signala u datoj periodi
- Učestanost PWM signala može se kontrolisati pomoću preskalera i selektora klok signala



```
shift_reg <= unsigned(inp);  
elsif (en = '1') then
```

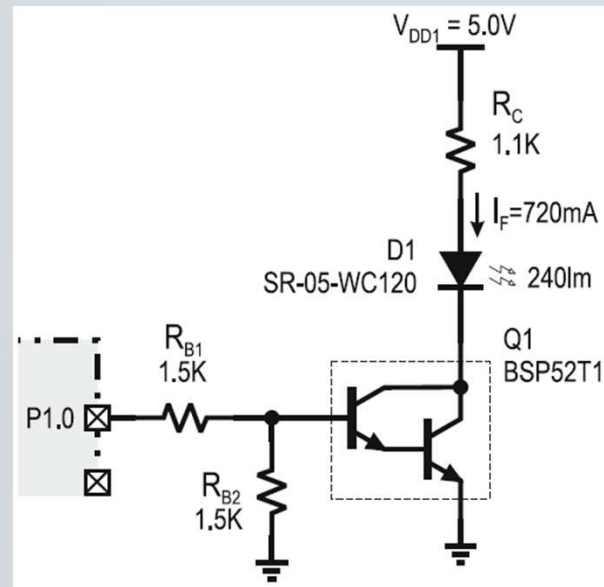
Naprednije primene tajmera: PWM generator IV

- Kao primer korišćenja PWM signala, razmotrimo kontrolu inteziteta svetla LED diode
- Sjajnost LED diode direktno je srazmerna jačini struje koja protiče kroz nju
- Pretpostavimo da u ovom primeru koristimo LED diodu model SR-05-WC120, koja može da generiše svetlost maksimalne sjajnosti od 240 lumena (lm) kada kroz nju protiče struja jačine 720mA
- Pretpostavimo da želimo da kontrolišemo sjajnost diode pomoću mikrokontrolera u 8 mogućih nivoa: *0lm*, *30lm*, *60lm*, *90lm*, *120lm*, *150lm*, *180lm* i *210lm*.
- Pretpostavimo da koristimo mikrokontroler sa 3.3V izlazima koji mogu da generišu maksimalno 20mA, a da je napajanje LED diode 5V

```
shift_reg <= unsigned(inp);  
elsif ( en == '1' ) then
```

Naprednije primene tajmera: PWM generator IV

- Obzirom da mikrokontroler radi na 3.3V i može da generiše samo 20mA po jednom izlazu, a LED dioda se napaja sa 5V napajanjem i zahteva struju jačine do 720mA moramo isprojektovati odgovarajući sprežni interfejs
- Jedno moguće rešenje prikazano je na slici desno, zajedno sa potrebnim faktorima ispunje koji će obezbediti željene nivoe sjajnosti
- Primetimo da svaki sledeći nivo sjajnosti zahteva povećanje od 1/8 faktora ispunje
- Pod pretpostavkom da koristimo 8-bitni tajmer i učestanost klok signala od 32.768kHz, vrednosti koje bi bilo potrebno upisati u HC registar PWM modula da bi se postigle željene vrednosti sjajnosti su: 0, 32, 64, 96, 128, 160, 192 i 224
- Obzirom da je perioda PWM signala jednaka $32768/256=128\text{Hz}$, neće biti vidljivog treperenja LED diode



| Brightness (lm) | Duty Cycle |
|-----------------|------------|
| 0 | 0% |
| 30 | 12.5% |
| 60 | 25.0% |
| 90 | 37.5% |
| 120 | 50.0% |
| 150 | 62.5% |
| 180 | 75.0% |
| 210 | 87.5% |
| 240 | 100.0% |

```
shift_reg <= unsigned(inp);  
elseif (en == '1') then
```



```
entity test_shift is
  generic ( width : integer := 17 );
  port ( clk : in std_ulogic;
        reset : in std_ulogic;
        load : in std_ulogic;
        en : in std_ulogic;
        outp : out std_ulogic );
end test_shift;
```

Tajmeri i brojači unutar mikrokontrolera 8051

```
shifter : process ( clk, reset )
begin
  if ( reset = '0' ) then
    shift_reg <= (others => '0');
  elsif rising_edge( clk ) then
    if (load = '1') then
      shift_reg <= unsigned (inp);
    elsif ( en = '1' ) then
```

Tajmeri i brojači unutar mikrokontrolera 8051

- Mikrokontroler 8051 sadrži dva šesnaestobitna tajmersko-brojačka registra koji se označavaju se kao **tajmer0** i **tajmer1**
- Mogu služiti kao brojački registri, ili tajmeri koji mere zadati vremenski interval. Oba tajmera mogu raditi u četiri radna režima.
- Kada radi kao brojač, sadržaj brojačkog registra se uvećava za jedan na svaku silaznu ivicu odgovarajućeg ulaza mikrokontrolera (**T0** za tajmer0, **T1** za tajmer1)
- Maksimalna frekvencija ulaznog signala koju brojač može da prati je 24 puta manja od radnog takta mikrokontrolera
- Kad radi kao tajmer, sadržaj tajmerskog registra se uvećava za jedan u svakom mašinskom ciklusu. Jedan mašinski ciklus traje 12 perioda radnog takta mikrokontrolera.
- Svakom tajmeru/brojaču pripadaju po dva osmобitna registra, **TH0** i **TL0** tajmeru 0, a **TH1** i **TL1** tajmeru 1

```
shift_reg <= unsigned(inp);  
elseif ( en == '1' ) then
```

Kontrola rada tajmera 0 i 1

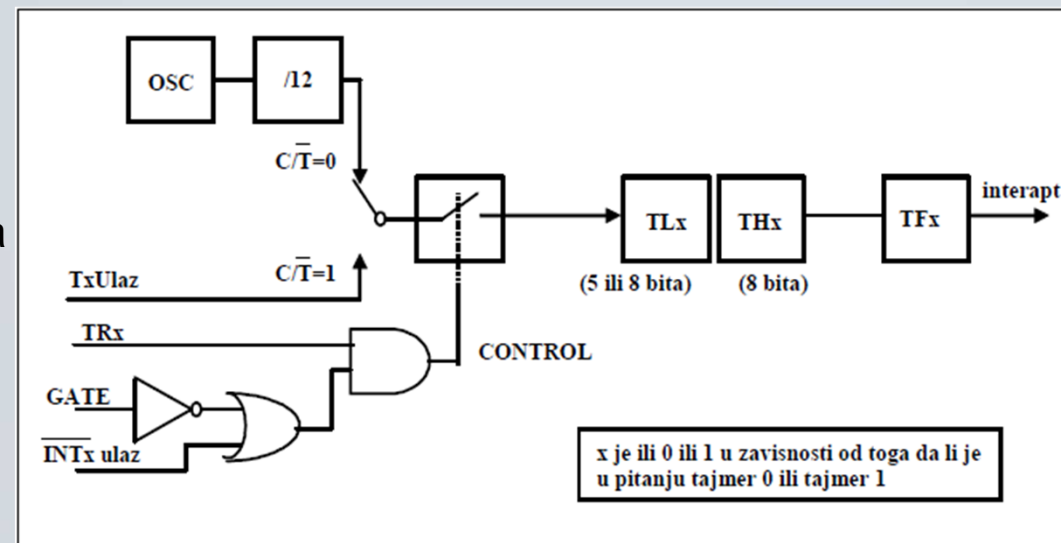
- Radom tajmera upravlja se pomoću registara **TMOD** i **TCON**
- Gornja četiri bita registra **TMOD** kontrolišu rad tajmera 1, a donja četiri rad tajmera 0
- Bitovi **TR1** i **TR0** registra **TCON** određuju da li će odgovarajući tajmer/brojač biti aktivan ili ne. Npr. ako je brojač 0 neaktivan on se neće uvećavati kada na ulazu **T0** nastupi prelaz sa visokog na nizak nivo.
- Način rada tajmera bira se kombinacijom bitova **M1** i **M0** registra **TMOD**
- Biti **C/T** registra **TMOD** definišu da li odgovarajući tajmer radi kao tajmer (**C/T=0**) ili kao brojač (**C/T=1**)
- Biti **GATE** omogućavaju kontrolu klok ulaza

TMOD

| (MSB) | | | | (LSB) | | | |
|----------|---|----|----|----------|-----|--|----|
| GATE | C/T | M1 | M0 | GATE | C/T | M1 | M0 |
| TAJMER 1 | | | | TAJMER 0 | | | |
| | | | | M1 | M0 | | |
| GATE | kontrola gejta. Kada je ovaj bit na jedinici, odgovarajući tajmer može da broji kada je pripadajući TR bit jedinica i pripadajući INT ulaz mikrokontrolera visok. Ako je vrednost GATE bita 0, dovoljno je da samo odgovarajući bit TR bude visok | | | 0 | 0 | Mod 0. tajmer radi kao 13-bitni brojač | |
| | | | | 0 | 1 | Mod 1. tajmer radi kao 16-bitni brojač | |
| | | | | 1 | 0 | Mod 2. tajmer radi kao osmobiitni brojač. Brojački registar je TL, nakon preticanja, TL se puni sadržajem TH registra (auto reload) | |
| C/T | ovaj bit određuje da li će odgovarajući tajmer da radi kao brojač opadajućih ivica na odgovarajućem T ulazu (C/T=1) ili kao tajmer (C/T=0) | | | 1 | 1 | Mod 3. Tajmer 1 je u ovom modu zaustavljen, a tajmer 0 radi kao dva odvojena osmobiitna tajmera. Za tajmer 1 ovaj mod se ne koristi (zabranjeno je). | |

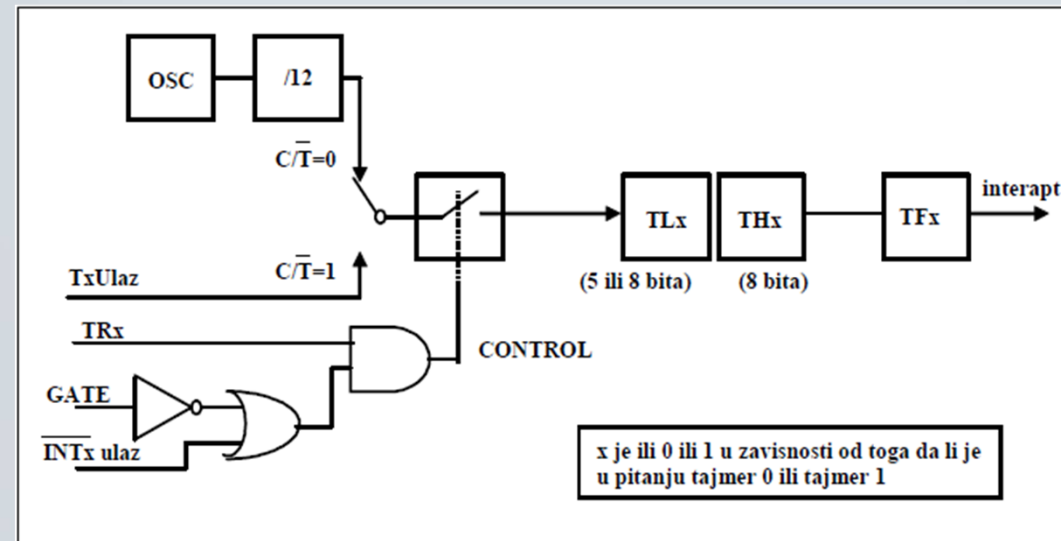
Modovi rada tajmera 0 i 1 – Mod 0

- U modu 0 tajmer predstavlja 13-bitni tajmer/brojač
- Kada svi biti registra postanu jedinice, sledećim uvećanjem sadržaja registra svi biti postaju nule
- Preticanje izaziva postavljanje odgovarajuće interapt zastavice (bit **TF1** registra **TCON** za tajmer 1 i bit **TF0** za tajmer 0)
- Postavljanje interapt zastavice izaziva prelazak na izvršavanje servisne rutine ako je odgovarajući interapt dozvoljen
- Biti **TF1** i **TF0** se **automatski** vraćaju na vrednost 0 kada se desi skok na interapt servisnu rutinu
- Pomenuti 13-bitni registar sastoji se od svih osam bita **TH** i nižih pet bita **TL** odgovarajućeg tajmer registra
- Viša tri bita **TL** registra u ovom modu su nedefinisana



Modovi rada tajmera 0 i 1 – Mod 1

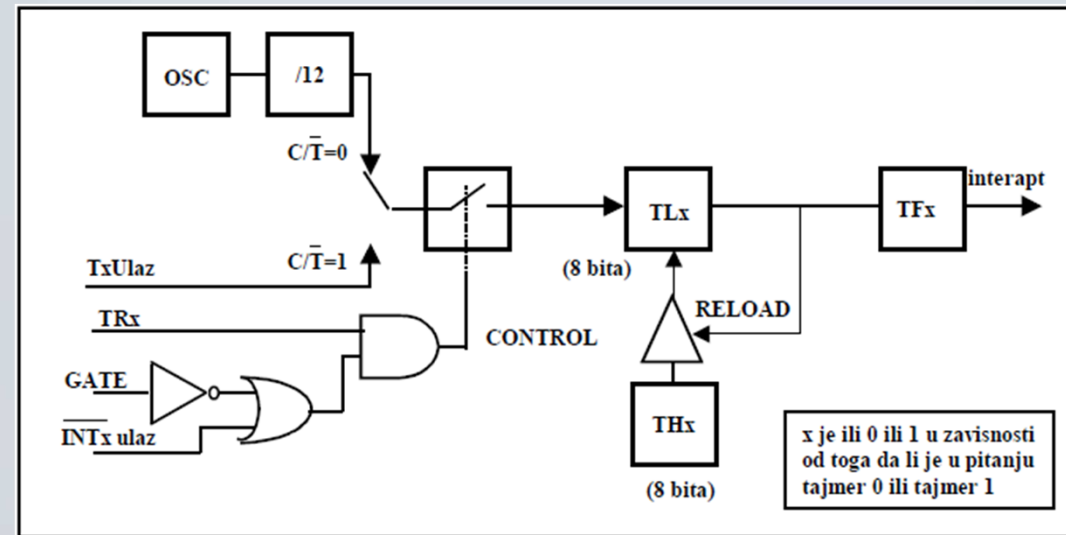
- Mod 1 je po svemu isti kao i mod 0 s tim što je brojački registar u ovom slučaju 16-bitni
- Koristi se svih osam bitova i **TL** i **TH** registra
- Bit **TF1** ili **TF0** se opet postavlja pri prelasku iz stanja svih šesnaest bita jednako 1 u stanje svih šesnaest bita jednako 0



```
shift_reg <= unsigned (inp);  
elsif ( en = '1') then
```

Modovi rada tajmera 0 i 1 – Mod 2

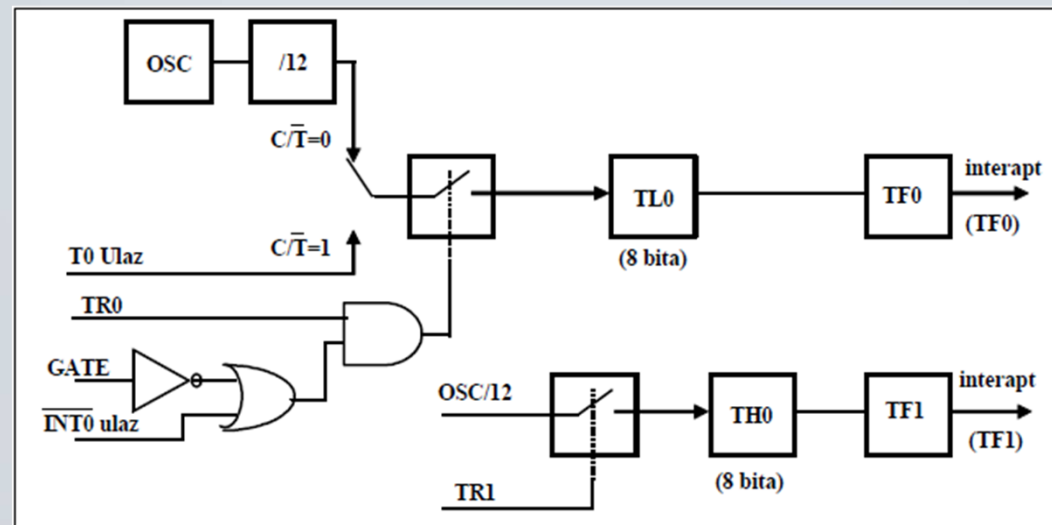
- U modu 2 tajmer radi kao osmobitni tajmer sa automatskim punjenjem inicijalne vrednosti nakon preticanja
- Kao tajmer/brojač uvećava se odgovarajući **TL** registar, a kada done do preticanja postavlja se bit **TF1** ili **TF0** u zavisnosti od tajmera, dok se vrednost iz odgovarajućeg **TH** registra upisuje u **TL** registar
- Vrednost **TH** registra ostaje neizmenjena



```
shift_reg <= unsigned (inp);  
elseif ( en = '1' ) then
```


Modovi rada tajmera 0 i 1 – Mod 3

- Ovaj mod različito prihvataju tajmer 0 i tajmer 1
- Kada se tajmer 1 prebaci u mod 3 on jednostavno prestane da broji zadržavajući svoje prethodno stanje, kao da je bit **TR1**=0
- Ako je tajmer 0 u modu 3, a tajmer 1 u bilo kom modu osim moda 3, tajmer 1 nastavlja svoj regularan rad u datom modu samo što ne može da generiše interapt jer je bit **TF1** kontrolisan tajmerom 0
- Tajmer 0 u modu 3 radi kao dva odvojena osmobiitna tajmera
- Bitovi **GATE**, **TR0** i **C/T** (i ulazi **T0** i **INT0**) kontrolišu brojanje registra **TL0** čijim preticanjem se postavlja bit zastavica **TF0**
- Registar **TH0** broji mašinske cikluse, a njegovim preticanjem postavlja se bit zastavica **TF1**
- Bit **TR1** kontroliše brojanje **TH0**



```
entity test_shift is  
  generic ( width: integer := 17 );
```



```
    shift_reg <= unsigned (inp);  
  elsif ( en = '1' ) then
```