Comparatoarele sunt circuite care indică, prin tensiunea de ieşire, situaţia relativă a două tensiuni aplicate la intrări (fig.3.74). Este vorba aici de un comparator pentru tensiuni cu acelaşi semn. De obicei una din tensiuni este variabilă iar cealaltă este fixă, reprezentând cu aproximaţie „pragul comparatorului”. Când tensiunea variabilă este U_1 comparatorul este „inversor”, iar când tensiunea variabilă este U_2 comparatorul este „neinversor”.

Caracteristica de transfer a acestor comparatoare este prezentată în fig.3.75a (pentru inversor) şi b (pentru neinversor).

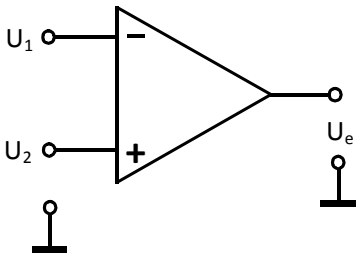


Fig. 3.74. Comparator simplu cu AO

Pentru situaţia $U_1 < U_2$ rezultă la ieşire $U_e = U_{emp}$ nivelul logic superior (pozitiv), iar pentru $U_1 > U_2$ rezultă $U_e = U_{emn}$ – nivelul logic inferior (negativ de obicei, dacă se alimentează AO cu două surse). Se foloseşte comparator inversor dacă se doreşte bascularea ieşirii de la nivel superior spre inferior, atunci când tensiunea de intrare crescătoare depăşeşte tensiunea fixă şi comparator neinversor în caz contrar.

Dacă însă tensiunile U_1 şi U_2 (sau una dintre ele) conţin zgomote, când tensiunea variabilă ajunge în dreptul zonei de indecizie apare fenomenul de „vibraţie” (oscilaţie) a tensiunii de la ieşirea comparatorului (fig.3.76) care înseamnă schimbarea de câteva ori, consecutiv, a deciziei logice – deci comenzi false (uneori supărătoare) pentru circuitele şi dispozitivele conectate la ieşire. Acesta este dezavantajul major al comparatorului simplu din fig.3.73; tensiunile ce se compară trebuie să fie foarte

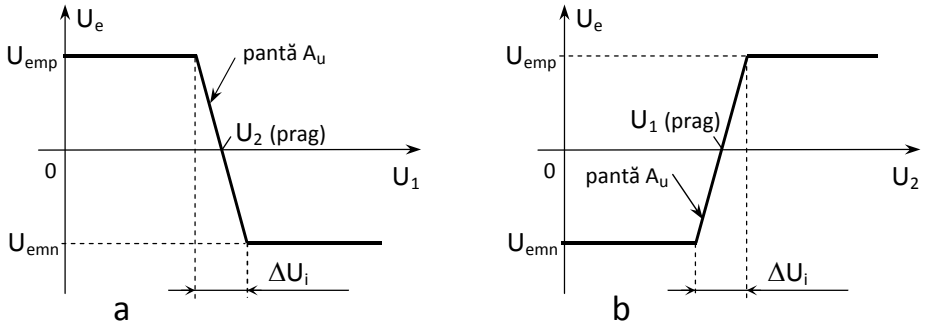


Fig. 3.75. Caracteristicile de transfer pentru comparatorul simplu inversor (a) şi neinversor (b)

„curate” pentru evitarea „vibraţiilor”.

Comparatoare cu reacţie pozitivă („cu histerezis”)

Pentru eliminarea fenomenului de „vibraţie” a tensiunii de ieşire a comparatorului, când tensiunile U_1 şi U_2 (sau una dintre ele) conţin zgomote, se utilizează o reacţie pozitivă (fig.3.79). Prin aceasta apare în caracteristica de transfer un „histerezis” (fig.3.80), care este mult mai lat decât zona de indecizie de la comparatorul fără reacţie. Aceasta conduce la o eroare de comparare sensibil mai mare, dar în schimb decizia logică este fermă.

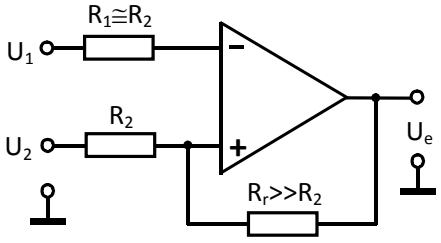
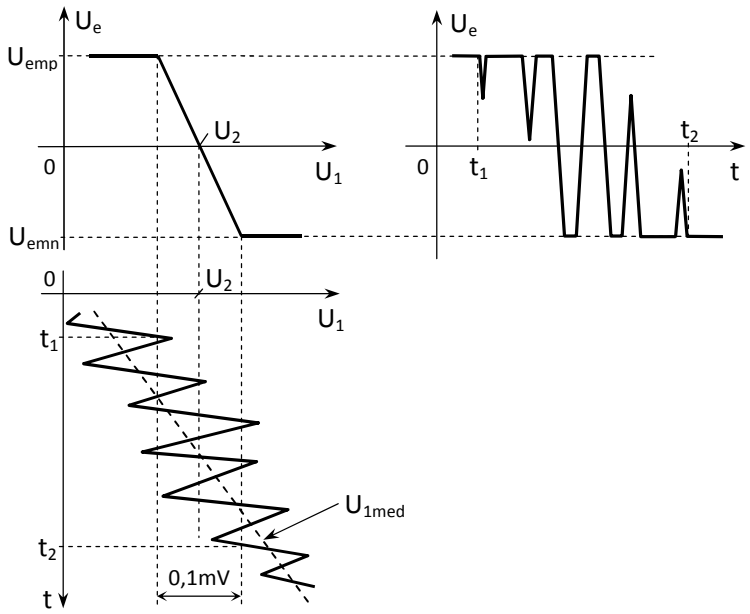


Fig. 3.79. Comparator cu reacţie pozitivă



Şi în acest caz întâlnim comparator „inversor” şi „neinversor”, după intrarea la care este aplicată tensiunea variabilă.

a) **Comparatorul inversor**

Acest comparator se foloseşte atunci când se doreşte bascularea ieşirii de la nivel superior spre inferior, dacă tensiunea de intrare crescătoare depăşeşte tensiunea fixă. Caracteristica de transfer a acestui comparator este prezentată în fig.3.80.

Pentru explicarea funcționării comparatorului se consideră inițial că $U_1 < 0$ şi de valoare absolută mare (punctul A de pe caracteristica de transfer), iar $U_2 > 0$. Atunci $U_2 >> U_1$ şi la ieşire se obține nivelul U_{emp}. Pe divizorul $R_r - R_2$ rezultă la intrarea + o tensiune, notată cu $U_1^{'}$, care îndeplineşte inegalitatea $U_1^{'} > U_2$. Dacă tensiunea U_1 creşte

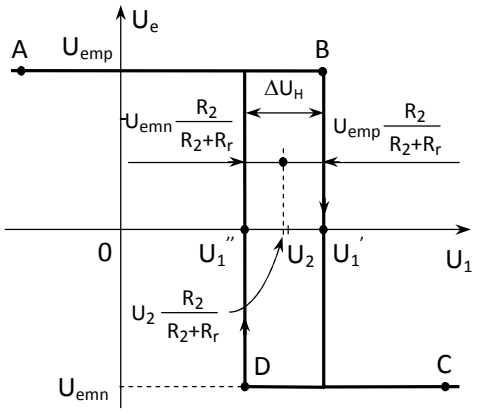


Fig. 3.80. Caracteristica de transfer a comparatorului inversor

şi atinge valoarea $U_1^{'}$ (punctul B pe caracteristică) intervine bascularea comparatorului care are loc din cauza situației tensiunilor existente direct la intrările + şi $-$. Datorită reacției pozitive realizată prin R_r , bascularea se accelerează pentru că diferența dintre tensiunile de la intrările + şi – se măreşte rapid prin scăderea tensiunii U_e începând din punctul B. Astfel, trecerea la nivelul U_{emn} are loc pentru o variație foarte mică a tensiunii U_1 şi în caracteristica de transfer apare o ramură practic verticală.

Creşterea în continuare a tensiunii variabile U_1 conduce la atingerea unui punct C pe caracterisitică. Acum, pe divizorul $R_r - R_2$ apare la intrarea + o tensi-une notată cu $U_1^{''}$ şi de valoare $U_1^{''} < U_2$ (fig.3.80). Dacă în continuare U_1 scade, bascularea spre nivelul logic superior începe la atingerea valorii $U_1^{''}$ - punctul D - şi are loc la fel de brusc ca şi prima basculare, datorită accentuării diferenței tensiunilor de la intrări prin reacție pozitivă. Nivelurile $U_1^{'}$ şi $U_1^{''}$, la care apar basculările se numesc **„pragurile”** comparatorului. Ele se pot calcula ținând cont de cele două situații ale tensiunilor pe divizorul $R_r - R_2$ (fig.3.81) la momentul începerii basculării.

Eroarea de comparare în acest caz este determinată în primul rând de distanțele dintre praguri şi tensiunea fixă U_2 şi se consideră cea mai mare dintre cele două distanțe.

(dacă acestea nu sunt egale între ele). **Lățimea zonei de histerezis** este stabilită de utilizator, întrucât ea trebuie să depăşească amplitudinea vârf-la-vârf a zgomotelor însumate ale tensiunilor ce compară, U_1 şi U_2 , (fig.3.82). În acest fel nu mai apar “vibrațiile” ieşirii comparatorului. În concluzie, se adoptă:

$$\Delta U_H > 1{,}2 \cdot \sum U_{zg.v.v.}$$

pentru a avea siguranța că la traversarea zonei de histerezis nici un vârf negativ al zgomotelor însumate nu va duce la coborârea tensiunii U_1 după momentul t_1 până sub pragul $U_1^{''}$. Bascularea va fi fermă şi are loc în momentul t_1 al atingerii pentru prima dată a pragului $U_1^{'}$ dacă ΔU_H este bine adoptată. Desigur, în prealabil se va face tot posibilul ca zgomotele suprapuse peste cele două tensiuni să fie cât mai reduse, spre a se putea lucra cu ΔU_H mic.

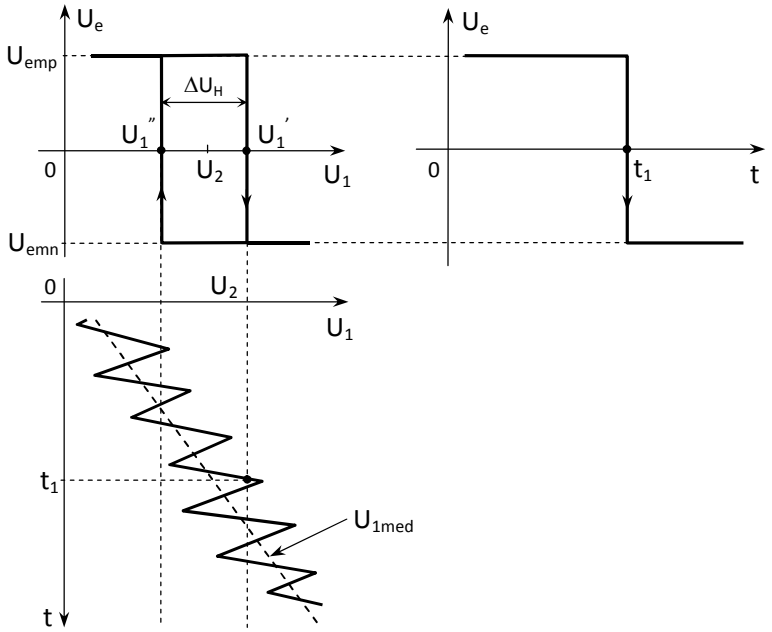


Fig. 3.82. Comportarea comparatorului cu histerezis de tip inversor

CIRCUITE INTEGRATE DIGITALE

Anul II

1. Prezentați funcționarea unui decodificator pe post de demultiplexor

Utilizarea DCD 74HC(T)138 pe post de DMUX se poate face în următoarele moduri:

- dacă intrarea de date (D_i) este o intrare de validare activă pe „0” (G_{2A} sau G_{2B}) şi codul de selecție este *A = „1”, B=„1”,* şi *C = „0”*, datele prezente la intrarea de date se vor regăsi la ieşirea Y_3. Pentru D_i = *„0”*, circuitul este validat corect şi ieşirea selectată este Y_3 = *„0”* (figura 1). Pentru D_i = *„1”*, circuitul nu este validat şi ieşirea selectată este Y_3 = *„1”* (figura 2). Astfel datele prezente la intrarea de date se regăsesc nemodificate la ieşirea selectată.

- dacă intrarea de date (D_i) este o intrare de validare activă pe „1” (G_1) şi codul de selecție este *A = „0”*, *B=„1”,* şi *C = „1”*, datele prezente la intrarea de date se vor regăsi la ieşirea Y_6. Pentru D_i = *„1”*, circuitul este validat corect şi ieşirea selectată este Y_3 = *„0”* (figura 3). Pentru D_i = *„0”*, circuitul nu este validat şi ieşirea selectată este Y_3 = *„1”* (figura 4). Astfel datele prezente la intrarea de date se regăsesc negate la ieşirea selectată.

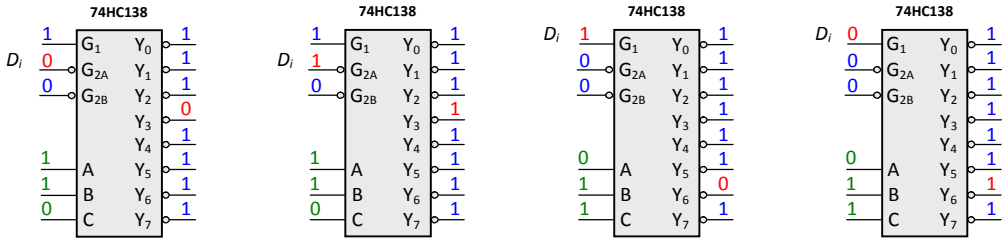


Figura 1; Figura 2; Figura 3; Figura 4.

Concluzie: Nu se fabrică DMUX. Pe post de DMUX se poate folosi orice DCD care are o intrare de validare. Dacă aceasta este activă pe „0” se obține un DMUX neinversor iar dacă este activă pe „1” se obține un DMUX inversor.

2. Desenați reprezentarea simbolică a unui bistabil de tip D care comută pe frontul crescător al impulsului de tact, tabelului lui de funcționare şi formele de undă aferente

Unul dintre cele mai simple bistabile care se produce sub formă integrată este bistabilul de tip D, activ pe frontul crescător al impulsului de tact aplicat la intrarea CK (figura 5).

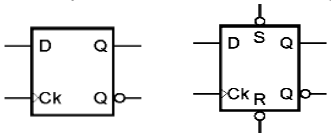


Figura 5. Bistabilul D care comută pe frontul crescător al tactului.

Informația aflată la intrarea D este transferată la ieşirea Q pe frontul crescător al tactului (conform tabelului 1). Dacă semnalul CK este pe palier (durata cât are valoarea „1” sau „0”), semnalul aplicat la intrarea D nu influențează ieşirea.

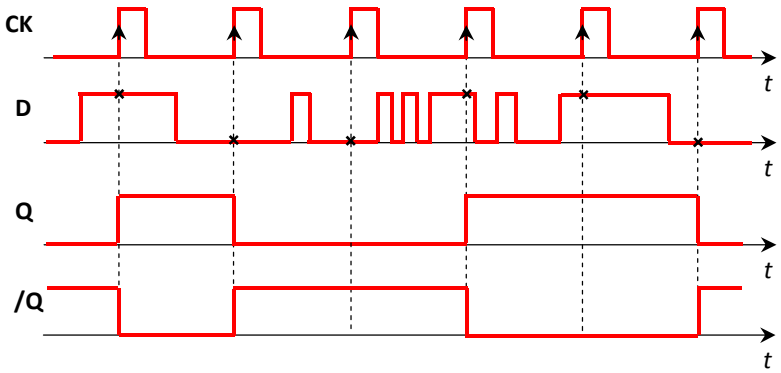
Tabelul 1

D	Q
0	0
1	1

Pe lângă intrarea D, circuitul poate avea şi două intrări asincrone prioritare /S şi /R. Funcționarea se bazează pe tabelul 2 cu observația că dacă ambele intrări prioritare sunt inactive circuitul funcționează sincron conform tabelului 1.

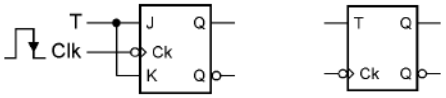
Tabelul 2

/S	/R	Q	/Q	
0	1	1	0	
1	0	0	1	
1	1			Funcționare sincronă conform tabelului 1
0	0	1	1	Stare interzisă



3. Desenați reprezentarea simbolică a unui bistabil de tip T care comută pe frontul descrescător al impulsului de tact, tabelului lui de funcționare şi formele de undă aferente

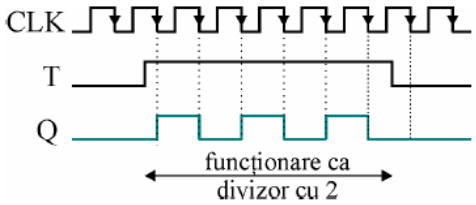
Bistabilul T se obține numai din CBB JK-MS prin conectarea împreună a intrărilor J şi K (CBB JK-MS este forțat să funcționeze doar în situațiile J = K = „0" şi J = K = „1").



Tabelul de funcționare:

T_n	Q_{n+1}
0	Q_n
1	$\overline{Q_n}$

Obs:



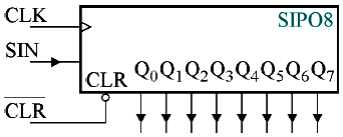
Dacă T este permanent „1", $Q_{n+1} = \overline{Q_n}$, bistabilul basculează la fiecare impuls de tact.

4. Descrieți modalitățile de realizare a conversiei serie-paralel,

respectiv paralel-serie a datelor

Conversia serie-paralel necesită utilizarea unui registru SIPO; ea se face în *n* tacte corespunzătoare celor *n* biți ai cuvântului binar.

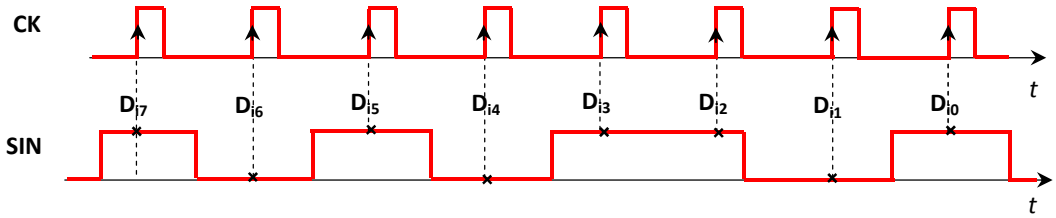
Funcționare:



Se şterge conținutul registrului punând intrarea /CLR la „0" (cu toate că principial nu este necesară inițializarea conținutului registrului, deoarece el se va suprascrie oricum după *n* impulsuri de tact).

Considerând un registru SIPO de 8 biți, secvența de înscriere a informației este D_7, D_6 ,..., D_0 – fiind necesare 8 impulsuri de tact pentru ca bitul D_7 (cel mai semnificativ) să ajungă la ieşire pe poziția corectă – Q_7. În acest moment cuvântul este înscris în totalitate în registru şi poate fi citit paralel.

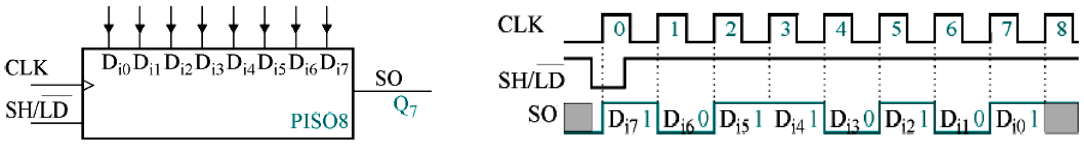
Ritmul în care sunt aduşi biții la intrarea serială SIN trebuie să fie corelat cu secvența de aplicare a impulsurilor de tact. Registrul comută pe frontul crescător al tactului (chiar dacă bistabilele comută pe frontul descrescător). Secvența care se converteşte este 10101101.



Obs.: Fiecare ieşire *Qi* poate fi folosită ca ieşire serială (circuitul se poate folosi ca SISO1, ... SISO8).

Conversia paralel-serie necesită utilizarea unui registru PISO. Conversia se face în *n* tacte corespunzătoare celor *n* biți ai cuvântului binar.

Pentru înscrierea paralelă a datelor D_{i7}, ..., D_{i0} se pune intrarea SH//LD = „0” şi se aplică un impuls de tact (înscrierea propriu-zisă se face pe frontul crescător al semnalului de tact). Pentru citirea serială a datelor (a cuvântului de n biți) se pune intrarea SH//LD = „1” şi se aplică n-1 impulsuri de tact.



Întreaga operație de conversie necesită *n* perioade de tact, prima fiind destinată pentru încărcarea paralelă, iar restul pentru citirea serială.

5. Descrieți, pe scurt, principalele de realizare a memoriilor temporare FIFO şi LIFO

Memoriile temporare sunt organizate pe *n* cuvinte binare de câte *b* biți compuse din *b* registre de deplasare seriale SISO de câte *n* biți fiecare.

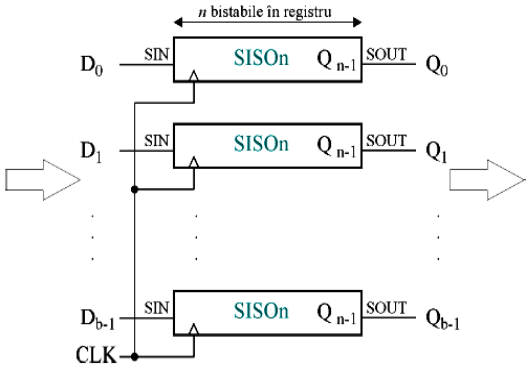
Memoria FIFO (First In First Out) se realizează cu ajutorul unor registre SISO care permit deplasarea într-un singur sens (spre dreapta).

Înscrierea cuvintelor binare de *b* biți în memorie se face în paralel pe cele *b* intrări seriale prin aplicarea a câte unui impuls de tact şi deplasarea acestora spre dreapta.

Memorie este plină atunci când s-au înscris toate cele *n* cuvinte binare. După umplerea completă a memoriei, primul cuvânt *citit* (paralel pe cele *b* ieşiri seriale) este primul cuvânt înscris în memorie.

În procesul de citire, informația se deplasează în continuare spre dreapta cu fiecare impuls de tact aplicat. Prin citire, informația se pierde!

Acest tip de memorie poate fi utilizat la gestionarea adreselor altor memorii pe durata întreruperilor unui sistem cu microprocesor.



Memoria temporară LIFO (Last In First Out) necesită registre SISO care pot deplasa informația în ambele sensuri (o intrare $R/\overline{L}$ - **R**ight//**L**eft - specifică sensul deplasării).

Înscrierea cuvintelor se face ca la memoria FIFO, prin deplasarea spre dreapta a datelor ($R/\overline{L}=1$) iar citirea se face prin deplasarea acestora spre stânga ($R/\overline{L}=0$).

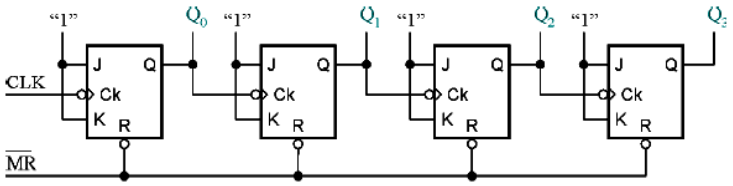
Astfel ultimul cuvânt înscris va fi primul citit.

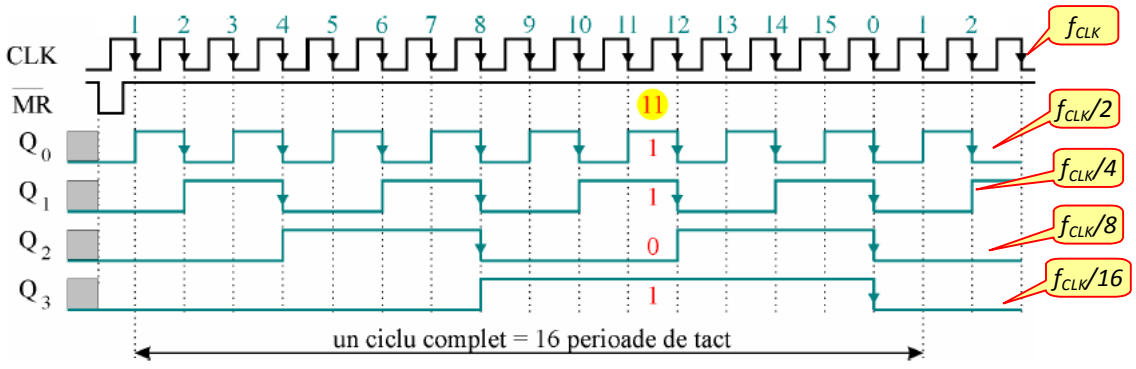
Memoria LIFO se utilizează ca memorie stivă în sistemele cu microprocesoare.

6. Desenați schema unui numărător asincron binar, pe 4 biți, explicați funcționarea sa,

şi trasați formele de undă aferente

Un numărător asincron binar, pe 4 biți, este format din 4 bistabile de tip T (provenite din JK-MS) cu T permanent pe „1”. Impulsurile de tact se aplică doar primului bistabil. Următoarele bistabile au ca semnal de tact ieşirea Q a bistabilului anterior (MR – Master Reset este o denumire sinonimă cu R - Reset sau CLR).





Obs:

1). Numărătorul numără în sens crescător (direct) adică cu fiecare impuls de CK aplicat, valoarea numărătorului creşte cu o unitate.

2). Numărătorul este modulo 16 (are 4 bistabile), al 16-lea impuls de tact încheie ciclul, el aducând numărătorul pe zero. Cel de-al 17-lea tact global este primul impuls de tact din cel de-al doilea ciclu.

3). La un moment dat, codul binar obținut citind ieşirile corespunde cu numărul de impulsuri de tact aplicate în ciclul respectiv (citind ieşirile după 11 tacte rezultă $Q_3Q_2Q_1Q_0$ = 1011 care corespunde cu numărul 11 codat binar). Aceasta este practic funcția de ***numărare***.

4). Bistabilele funcționează ca ***divizoare de frecvență cu 2***. Ieşirea Q_0 divizează cu 2 frecvența tactului, Q_1 divizează cu 2 frecvența semnalului Q_0 şi cu 4 frecvența tactului, etc.

5). Pentru extinderea capacității de numărare se pot conecta mai multe numărătoare în cascadă prin conectarea ieşirii Q_3 la intrarea de tact a următorului numărător.

7. Desenați schema unui numărător sincron binar, pe 4 biți, explicați funcționarea sa, şi trasați formele de undă aferente

Numărătoare sincrone sunt numărătoare la care impulsul de tact se aplică simultan tuturor bistabilelor (de tip T) permițând, astfel funcționarea la frecvențe de tact mult mai mari (tipic 35MHz).

În cadrul unui ciclu de numărare, la trecerea dintr-o stare în alta, unele bistabile trebuie să comute, altele nu. Înseamnă că numărătoarele trebuie realizate cu bistabile de tip T care au intrarea T accesibilă pentru a permite ca, înaintea aplicării următorului impuls de tact, intrarea T a bistabilului ce trebuie să comute să fie conectată la „1” iar intrarea T a bistabilului ce nu trebuie să comute să fie conectată la „0”.

Apare, astfel, necesitatea utilizării unor circuite logice pentru generarea valorilor T ce corespund celor *n* bistabile folosite pentru ca funcționarea numărătorului să decurgă în conformitate cu *tabelul de funcționare* dorit.

Nr. tacte	Q_3	Q_2	Q_1	Q_0
initializare	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
16 (0)	0	0	0	0

Din tabel se deduc următoarele:

- bistabilul Q_0 trebuie să basculeze la fiecare impuls de tact, deci $T_0 = 1$;
- bistabilul Q_1 basculează numai dacă înaintea aplicării tactului $Q_0 = 1$ deci $T_1 = Q_0$;
- bistabilul Q_2 basculează numai dacă înaintea aplicării tactului Q_0 şi Q_1 sunt pe „1” adică: $T_2 = Q_0 \cdot Q_1 = Q_1 \cdot T_1$.

- bistabilul Q_3 basculează numai dacă înaintea aplicării impulsului de tact Q_0, Q_1 şi Q_2 sunt pe „1” deci $T_3 = Q_0 \cdot Q_1 \cdot Q_2 = Q_2 \cdot T_2$.

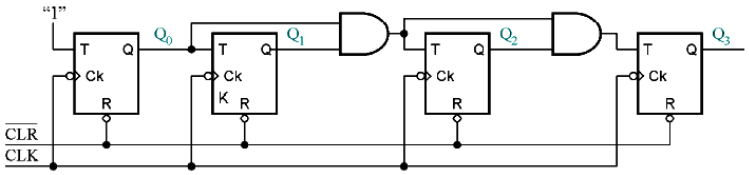
- în general se poate scrie: $T_{n-1} = Q_0 \cdot Q_1 \cdot \ldots \cdot Q_{n-2} = T_{n-2} \cdot Q_{n-2}$.

În funcție de modul de scriere al valorilor T se disting două ***metode de generare*** a acestora:

- **serială** – dacă valoarea curentă a lui T se obține din cea anterioară:

$$T_2 = T_1 \cdot Q_1 \quad \text{şi} \quad T_3 = T_2 \cdot Q_2 .$$

Schema numărătorului sincron obținut prin metoda serială:



Durata minimă a impulsului de tact este:

$$T_{CLK\,\min} = t_{PCLK \to Q} + (n-2)t_{PSI} + \Delta t .$$

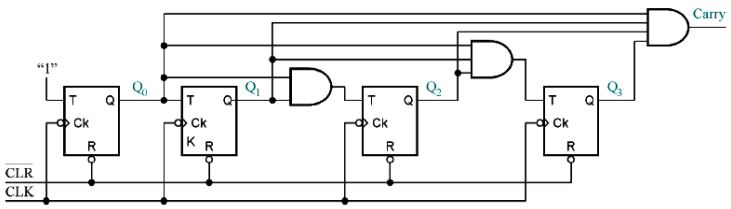
Dezavantaj: - t_p mai mare decât în cazul generării paralele a valorilor *T.*

Avantaj: - se utilizează numai porți ŞI cu două intrări.

- **paralelă** – dacă valorile lui *T* se obțin direct din valorile lui *Q*:

$$T_2 = Q_0 \cdot Q_1 \quad \text{şi} \quad T_3 = Q_0 \cdot Q_1 \cdot Q_2$$

Schema numărătorului sincron obținut prin metoda paralelă:



În cazul generării *paralele* a valorilor *T* durata minimă a impulsurilor de tact este:

$$T_{CLK\,\min} = t_{PCLK \to Q} + t_{PSI} + \Delta t$$

Se observă că t_p este mai mic ceea ce conduce la o frecvența de tact mai ridicată. Din acest motiv aceasta este varianta preferată la realizarea numărătoarelor sincrone integrate.

Semnalul Carry (semnalul de transport) se generează din semnalele Q_0, Q_1, Q_2, şi Q_3. $Cy = Q_0 \cdot Q_1 \cdot Q_2 \cdot Q_3$ şi se aplică intrării T a numărătorului (bistabilului) următor în cazul extinderii capacității de numărare (cascadarea numărătoarelor).

8. Prezentați, sumar, principalele metode de obținere

a divizoarelor de frecvență programabile

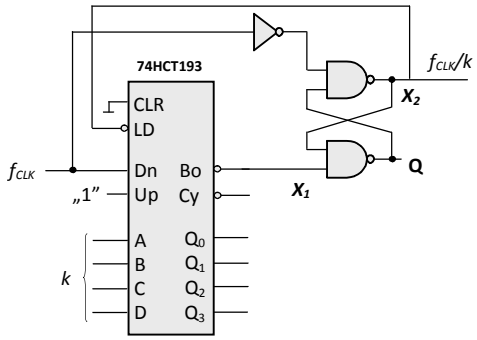
Divizoarele de frecvență programabile sunt divizoare de frecvență la care raportul de divizare se poate modifica de la un ciclu de divizare la următorul.

Varianta 1 – ***cu numărare în sens invers şi încărcare paralelă***.

Este cea mai utilizată metodă de obținere a unui divizor programabil. Se bazează pe utilizarea unui numărător reversibil cu posibilitatea de a fi încărcat paralel. Numărul cu care se realizează divizarea (*k*) se aduce la intrările paralel şi se încarcă în numărător prin activarea liniei */LD*. Numărătorul este decrementat cu frecvența f_{CLK} aplicată la intrarea Count Down (*Dn*) până când el ajunge în starea 0000. În acel moment ieşirea Borrow (*/Bo)* trece pe „0”, activează intrarea */LD*, şi inițiază o nouă încărcare a numărătorului cu numărul *k*.

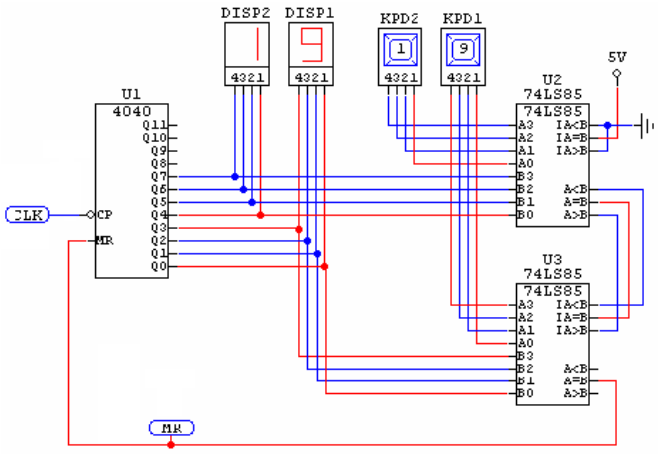
Deoarece bistabilele din componența numărătorului nu au acelaşi timp de încărcare şi, astfel apare riscul unei încărcări incomplete, este necesar intercalarea unui bistabil SR de memorare a impulsului de încărcare (la fel ca la numărătoarele modulo p).

Astfel, la ieşirea /Q a acestuia se obține semnalul f_{CLK}/k.



Varianta 2 – ***cu numărare în sens direct şi comparator***.

Metoda utilizează un numărător asincron (4040) şi două comparatoare pe 4 biți (74LS85) care specifică raportul de divizare k. Numărătorul numără în sens direct, de la 0 până la valoarea k prestabilită de comutatoarele [KPD1 şi KPD2]. În acel moment comparatoarele sesizează egalitatea şi activează semnalul de ştergere /MR. Schema prezentată este pe 8 biți.



Pentru obținerea unui divizor de frecvență pe 12 biți sunt necesare un numărător şi un comparator pe 12 biți.

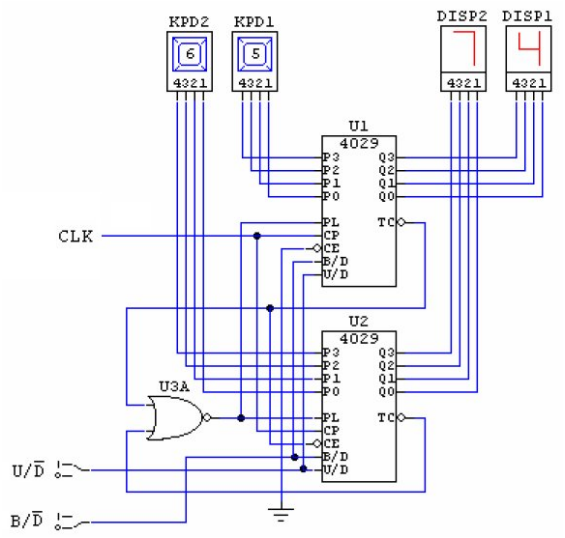
Schema prezentată este una care funcționează foarte bine în regim de simulare digitală, dar nu în realitate deoarece foloseşte circuite CMOS şi TTL LS în acelaşi montaj. Pentru a rezolva acest neajuns, cel mai bine este să se folosească variantele HC sau HCT ale circuitelor prezentate: 74HCT4040 şi 74HCT85, caz în care schema nu va mai prezenta nici un neajuns.

Varianta 3 – ***combinată, cu posibilitatea numărării în ambele sensuri***.

Este cea mai versatilă metodă. Se bazează pe folosirea numărătoarelor 4029 la care intrarea de încărcare este comandată de o poartă SAU-NU cu un număr de intrări egal cu numărul de circuite 4029 utilizate.

Circuitul oferă:

- numărare în sens crescător, de la numărul prestabilit k la 255 (dacă $U/\overline{D}=1$);
- numărare în sens descrescător, de la p la 0 (dacă $U/\overline{D}=0$);
- numărare binară (dacă $B/\overline{D}=1$);
- numărare zecimală (dacă $B/\overline{D}=0$).



9. Enumerați principalele avantaje şi dezavantaje ale memoriilor SRAM

în comparație cu memoriile DRAM

Memoriile RAM se clasifică în:

- RAM statice (**SRAM** – Static Random Access Memory) la care celula elementară de memorare este un latch D realizat în tehnologie bipolară sau unipolară;
- RAM dinamice (**DRAM** – Dynamic Random Access Memory) - celula elementară este o capacitate; sunt realizate numai în tehnologie unipolară NMOS sau CMOS.

Memoria SRAM *păstrează datele* pentru o perioadă de timp *nelimitată*, până în momentul în care ea este rescrisă. În schimb, memoria DRAM necesită *rescrierea permanentă*, la câteva fracțiuni de secundă, altfel informațiile fiind pierdute.

Avantajele memoriei SRAM: utilitatea crescută datorită modului de funcționare şi viteza foarte mare (raportul de timp de acces SRAM/DRAM = 8-16).

Dezavantajele memoriei SRAM: densitatea de integrare mai redusă şi prețul mult mai mare decât al memoriei DRAM (de obicei raportul de capacitate DRAM/SRAM = 4-8 iar raportul de cost SRAM/DRAM = 8-16).

Aplicațiile de bază ale memoriilor RAM se regăsesc la PC-urile. Memoria SRAM este folosită cel mai adesea ca memorie intermediară/cache, pe când DRAM-ul este utilizat ca memorie principală a oricărui sistem.

10. Explicați, pe scurt, funcționarea unei memorii DRAM

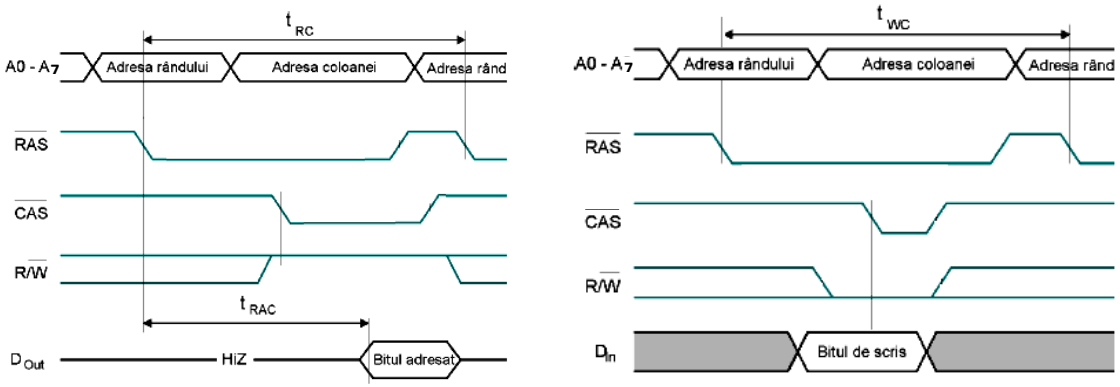
(citire, scriere, reîmprospătare)

Citirea informației memorate într-o memorie DRAM

La liniile de adresă se aduce adresa de linie A_0…A_7. După ce aceasta s-a stabilizat se activează linia /RAS pentru încărcarea adresei de linie în registrul din circuitul de comandă. În continuare adresa se decodifică, se selectează linia şi conținutul tuturor celulelor de memorare aferente liniei se scrie în registrul de linii.

Apoi se aduce la intrare adresa de coloane A_8…A_{15}. După ce aceasta s-a stabilizat se activează semnalul /CAS. Pe frontul descrescător al /CAS se investighează linia /WE. Aceasta trebuie să fie pe „1” deoarece se execută o operație de citire. Tot pe frontul descrescător al semnalului /CAS se memorează adresa coloanei A_8…A_{15} în registrul corespunzător din circuitul de comandă. Cu ajutorul lor şi al MUX-ului, se selectează una dintre cele 256 de coloane ale liniei memorate în registrul de linii, şi conținutul celulei selectate se transmite, prin buffer (aflat în stare normală), spre ieşire D_{out}.

În continuare se dezactivează /RAS-ul (conținutul registrului de linii se reînscrie în matricea de memorare) apoi se dezactivează şi /CAS-ul şi linia D_{out} trece pe Z.



Citirea informației Scrierea informației

Scrierea informației în memorie

Furnizarea adresei locației de memorare în care urmează să se scrie informația se face la fel ca la operația de citire.

Deosebirile apar pe frontul descrescător al /CAS când în urma investigării se găseşte linia /WE pe „0”. Acest fapt înseamnă că urmează o operație de scriere şi, tot în acel moment datele care urmează a fi scrise trebuie să fie prezente pe linia D_{in}. În continuare se memorează adresa coloanei A_8…A_{15} în registrul corespunzător din circuitul de comandă. Cu ajutorul lor şi al DMUX-ului, se selectează una dintre cele 256 de coloane ale liniei memorate în registrul de linii, şi informația de pe D_{in} se memorează în această celulă.

În continuare se dezactivează /RAS-ul (conținutul registrului de linii se reînscrie în matricea de memorare) apoi se dezactivează şi /CAS-ul.

Reîmprospătarea informației memorate

Se foloseşte un numărător pe 8 biți, cu funcționare continuă care generează adresele celor 256 de linii. Pe frontal descrescător al semnalului /RAS se selectează o linia ce corespunde adresei. Conținutul fiecărei celule ale acestei linii se înscrie în registru de linii. Pe frontul crescător al semnalului /RAS se reînscrie informația din registru de linii, regenerată în celulele corespunzătoare.

În continuare se trece la următoarea adresă şi se reîmprospătează informațiile din celulele liniei următoare.

Sisteme de prelucrare numerică cu procesoare

Anul II

1.Structura generală a unui sistem de prelucrare numerică cu procesor (SPN)

[1], pag. 11

În acest capitol sunt prezentate principii generale privind structura şi funcţionarea unui sistem de prelucrare numerică cu procesor (SPN).

Structura generală a unui SPN este prezentată în figura 1.1. Unitatea centrală de prelucrare (UCP), este cea mai importantă componentă a unui astfel de sistem. Principala funcţie a UCP este de a executa un program reprezentat printr-o secvenţă de instrucţiuni. Programul este încărcat în prealabil în memorie, mai concret în memoria program. Execuţia programului implică existenţa unor date care urmează să fie prelucrate. Acestea se găsesc fie în memoria de date, fie sunt preluate de la periferice. Execuţia programului se concretizează prin generarea unor date care pot fi stocate în memoria de date sau pot comanda perifericele. Perifericele asigură schimbul de informaţii cu exteriorul. În cea mai simplă formă perifericele sunt reprezentate de porturile de intrare-ieşire (intrări-ieşiri numerice). Alte exemple de periferice sunt: convertoarele analog-numerice (intrări analogice), convertoarele numeric-analogice (ieşiri analogice), interfeţele (porturile) seriale, temporizatoarele.

Un *procesor* (*microprocesor*), care este specific calculatoarelor personale, conţine doar UCP, relativ la figura 1.1. Acesta are o mare putere de calcul deoarece trebuie să execute mai multe aplicaţii în acelaşi timp. De aceea, memoria şi perifericele sunt externe, fiind poziţionate în interiorul carcasei calculatorului. Un *microcontroler*, la fel ca un *procesor de semnal*, conţine toate cele 3 elemente componente din figura 1.1 integrate în aceeaşi capsulă. Aceste două dispozitive sunt folosite pentru aplicaţii dedicate. Diferenţa între cele două este că procesorul de semnal este optimizat din punct de vedere al instrucţiunilor pentru a face prelucrări de semnal (filtrări numerice sau transformări Fourier rapide) în timp ce un microcontroler are integrate o gamă mai largă de periferice, în special intrări-ieşiri numerice.

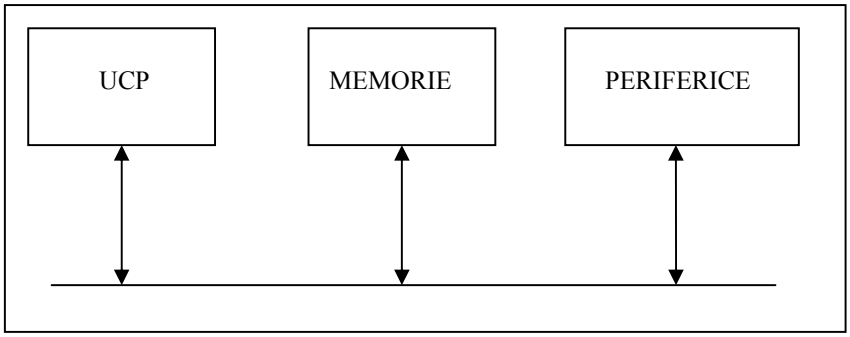


Fig. 1.1 Structura generală a unui SPN.

2. Să se precizeze funcţiile registrelor PC (Program Counter) şi, respectiv, SR (Status Register) pentru un procesor.

[1], pag. 13, 15, 34-35.

Registrul PC (*Program Counter*) indică adresa din memorie a instrucţiunii care urmează să fie executată (adresa primului octet al instrucţiunii). Rezultă că după fiecare instrucţiune registrul PC îşi măreşte conţinutul cu numărul de octeţi ai codului maşină ai instrucţiunii respective. Acest lucru este valabil când execuţia programului este liniară, adică nu există ramificaţii în program. O ramificaţie înseamnă că următoarea instrucţiune executată nu este cea de la adresa care urmează după ultimul octet al instrucţiunii curente, ci una situată la o adresă mai mare sau mai mică. Există trei posibilităţi de ramificaţii: instrucţinui de salt, apeluri de subrutine sau răspunsuri la cereri de întrerupere. În aceste situaţii, registrul PC va fi încărcat cu adresa instrucţiunii unde se va face saltul. Această adresă este cu un număr de unităţi mai mare sau mai mică decât conţinutul registrului PC înainte de salt.

Registrul SR (*Status Register*) conţine la majoritatea dispozitivelor biţii indicatori care sunt modificaţi în urma execuţiei unei instrucţiuni aritmetice sau logice: C (*Carry*), V (*Overflow*), N (*Negative*), Z (*Zero*).

Bitul indicator de transport C (*Carry bit*). Acest bit este poziţionat la nivel logic 1 dacă rezultatul unei operaţii aritmetice a produs un transport şi este poziţionat la nivel logic 0 dacă nu a avut loc un transport.

Bitul indicator de depăşire V (*Overflow bit*). Acest bit se poziţionează la nivel logic 1 dacă rezultatul unei operaţii aritmetice depăşeşte domeniul de valori corespunzător reprezentării în cod complementul lui doi.

Bitul indicator de semn N (*Negative bit*). Acest bit se poziţionează la nivel logic 1 dacă rezultatul unei operaţii este un număr negativ şi la nivel logic 0 daca rezultatul operaţiei este un număr pozitiv.

Bitul indicator de zero Z (*Zero bit*). Acest bit se poziţionează la nivel logic 1 în urma execuţiei unei instrucţiuni al cărei rezultat este zero şi este poziţionat la nivel logic 0 dacă rezultatul este diferit de zero.

De asemenea, registrul SR conţine şi bitul care permite validarea întreruperilor mascabile, notat de obicei GIE (*General Interrupt Enable*). În afară de aceşti biţi, registrul SR mai poate conţine şi alţi biţi.

3. Să se precizeze funcţia registrului SP (Stack Pointer) şi a memoriei stivă pentru un procesor.

[1], pag. 14, 15.

Funcţia registrului SP este în strânsă legătură cu registrul PC. Apelul unei subrutine sau răspunsul la o cerere de întrerupere înseamnă un salt la o adresă unde este plasată subrutina (subrutina de întrerupere).

Astfel, în figura 1.3 instrucţiunea CALL *S*_1, aflată la adresa *ADR*1 apelează subrutina *S*_1, aflată la adresa *AS*_1. La încheierea subrutinei (instrucţiunea RET) programul trebuie să se reîntoarcă la instrucţiunea care urmează după cea care a făcut apelul, adică instrucţiunea Instr. 1, aflată la adresa *ADR*1+*n* (*n* reprezintă numărul de octeţi ai instrucţiunii CALL*S*_1). Pentru a fi posibil acest lucru, registrul PC trebuie încărcat cu adresa *ADR*1+*n*. Această adresă a fost disponibilă în registrul PC după extragerea codului maşină al instrucţiunii CALL *S*_1, înainte de decodificarea şi execuţia acestei instrucţiuni. De aceea, execuţia instrucţiunii CALL *S*_1 înseamnă mai întâi salvarea registrului PC şi abia apoi încărcarea lui cu adresa de salt (*AS*_1, în acest caz). Zona de memorie unde se realizează salvarea se numeşte *stivă*.
De obicei salvarea în stivă se face la adrese descrescătoare. În acest sens, registrul SP (*Stack Pointer*, indicator al vârfului stivei) este decrementat cu 1 pentru fiecare octet salvat în stivă. Conţinutul acestui registru indică adresa ultimului octet salvat.

Se presupune că înainte de execuţia instrucţiunii CALL *S*_1 conţinutul registrului SP era 4000*h*. Deoarece fiecare adresă salvată în stivă conţine 2 octeţi , înseamnă că registrul SP a fost decrementat de 4 ori, adică conţine valoarea 3FFC*h*. La execuţia instrucţiunii RET din subrutina *S*_2, registrul PC se încarcă cu conţinutul stivei de la adresele SP şi SP+1 (adică 3FFC*h* şi 3FFD*h*), iar registrul SP se incrementează cu 2 unităţi. La execuţia instrucţiunii RET din subrutina *S*_1 registrul PC se încarcă de asemenea cu conţinutul stivei de la adresele SP şi SP+1 (3FFE*h* şi 3FFF*h* în acest caz), după care registrul SP se incrementează din nou cu 2 unităţi. Rezultă că numerele existente în stivă se citesc în ordinea inversă celei în care au fost salvate, de unde denumirea de stivă sau memorie LIFO (*Last In First Out*).

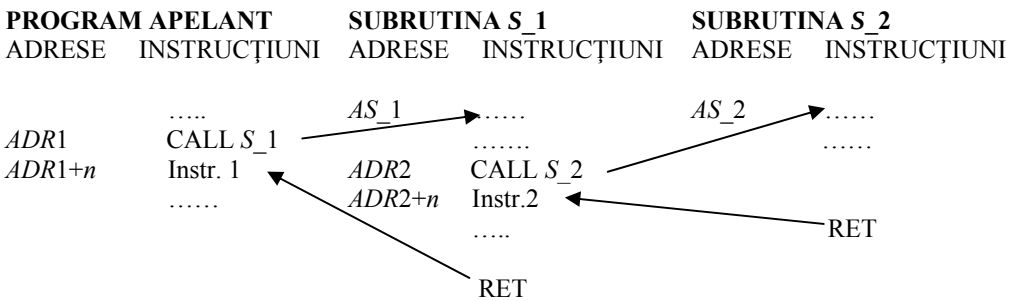


Fig. 1.3 Apelul unei subrutine.

4. Funcţionarea unui microcontroler la acceptarea unei cereri de înrerupere

[1], pag. 27.

În principiu, funcţionarea unui microntroler la acceptarea unei cereri de întrerupere se face conform figurii 1.13. Instrucţiunea în curs de execuţie în momentul primirii cererii de întrerupere se află în PROGRAMUL PRINCIPAL la adresa *ADR*1. Ea este executată complet, după care se face automat saltul pentru execuţia SUBRUTINEI DE ÎNTRERUPERE. După încheierea subrutinei de întrerupere, execuţia programului continuă de la adresa *ADR*1+*n* (*n* reprezintă numărul de octeţi ai instrucţiunii de la adresa *ADR*1). Pentru a fi posibil acest lucru, adresa *ADR*1+*n*, care este conţinută în registrul PC după extragerea instrucţiunii de la adresa *ADR*1, este automat salvată în stivă. Apoi, registrul PC este încărcat (tot automat!) cu adresa subrutinei de întrerupere (numită şi *vector de întrerupere*, *interrupt vector*) corespunzătoare sursei *i.* Efect: începe execuţia subrutinei de întrerupere. De remarcat că înaintea execuţiei subrutinei de întrerupere, o parte dintre registrele microcontrolerului, printre care registrul SR, sunt salvate în stivă. Execuţia subrutinei de întrerupere se încheie cu o instrucţiune de tip RETI (*return from interrupt*) care implică refacerea registrelor salvate în stivă, inclusiv a registrului Program Counter, având ca efect continuarea execuţiei de la adresa *ADR*1+*n*.

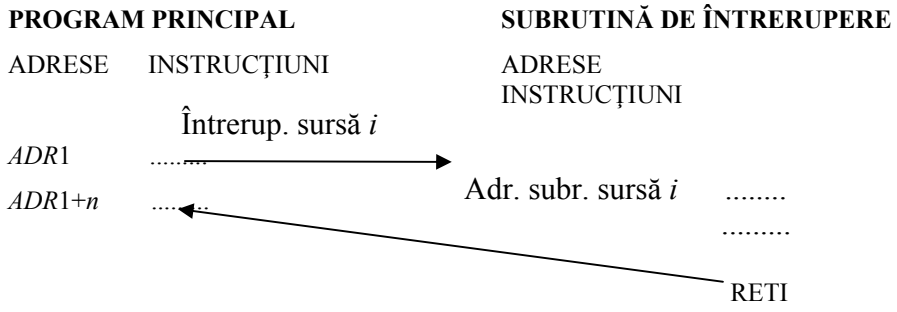


Fig. 1.13. Funcţionarea unui microntroler la acceptarea unei cereri de întrerupere.

5. Să se prezinte principial funcţiile de ieşire şi de intrare ale unui pin al unui microcontroler (Digital I/O).

[1] pag. 16,17.

În figura 1.6 se prezintă principial funcţiile de ieşire şi de intrare ale unui pin.

Astfel, pentru un pin având funcţia de ieşire, informaţia binară (un bit cu valoarea 0 sau 1 logic) este transmisă din microcontroler la pin, regăsindu-se sub forma unei tensiuni (0 logic-0V, 1 logic-tensiunea de alimentare pozitivă $+V_{cc}$). Tensiunea respectivă poate fi măsurată cu un voltmetru.

Pentru un pin având funcţia de intrare, informaţia binară aplicată la pin sub forma unei tensiuni de la o sursă (0 logic-0V sau masă, 1 logic-tensiunea de alimentare pozitivă $+V_{cc}$) este transmisă în microcontroler, regăsindu-se în valoarea unui bit (0 sau 1 logic).

Rezumând, cele două operaţii pot fi prezentate sintetic astfel:

Ieşire: bit (scris prin program)→tensiune la pin (măsurată cu un voltmetru).

Intrare: tensiune la pin (aplicată de la o sursă)→bit (citit prin program).

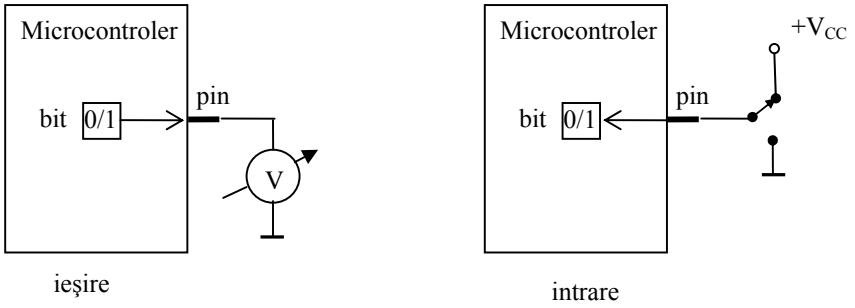


Fig. 1.6 Funcţiile de ieşire şi, respectiv, de intrare ale unui pin.

6. Care este rolul magistralei de adrese (MA) într-un sistem de prelucrare numerică cu procesor? Dacă MA are 16 linii, care este dimensiunea spaţiului memorie adresat? Determinaţi intervalul de adresare corespunzător unei capacităţi de memorie de 8 Kocteţi care începe la adresa 2500*h*.

[2], slide nr. 7, 8.

Magistrala de adrese selectează locaţia de memorie care urmează să fie scrisă sau citită. Pentru o magistrală de adrese de 16 biţi, dimensiunea spaţiului de memorie adresat este de $2^{16}1B= 2^6 2^{10}B=64$ KB.

Intervale de adresare	Capacitate de memorie
0000h÷00FFh	256 octeţi
0000h÷00FFh	256 octeţi
…	
0000h÷03FFh	1024 octeţi =1 koctet
…	
0000h÷0FFFh	4 kocteţi
…	
0000h÷3FFFh	16 kocteţi

Din tabel se deduce că la 8 kocteţi corespunde intervalul 0000-1FFFh. Pentru obţinerea intevalului cerut, se translatează capetele intervalului 0000-1FFFh cu 2500h, adică se obţine intervalul 2500h-44FFh.

7. Descrieţi funcţia de temporizare a unui periferic de tip Timer (temporizator)

[1], pag. 18.

Un periferic de tip „Timer” sau temporizator generează evenimente periodice. Perioada evenimentelor sau temporizarea se stabileşte prin numărarea unui număr prestabilit de impulsuri cu o anumită perioadă. Astfel, temporizatorul conţine un numărător şi un generator de semnal de tact. Numărătorul se încarcă cu o constantă iar apoi se decrementează cu 1 la fiecare impuls primit [1]. Temporizarea corespunde anulării conţinutului numărătorului. Primul impuls primit după anulare reîncarcă constanta şi apoi procesul se continuă în acest fel. Există şi varianta în care numărătorul începe numărarea de la 0 şi îşi incrementează conţinutul la fiecare impuls primit. În acest caz temporizarea corespunde momentului când conţinutul numărătorului egalează o constantă prestabilită. În acest scop Timer-ul conţine un comparator. La următorul impuls primit numărarea reîncepe de la 0. De fiecare dată când temporizarea se încheie (conţinutul numărătorului se anulează sau egalează constanta predefinită) un bit indicator (*flag*) este trecut pe 1 logic.

Expresia temporizării este

$$T = \frac{Cst + 1}{f_0} = (Cst + 1)T_0, \qquad (1.1)$$

unde *Cst* este constanta de temporizare, iar $f_0=1/T_0$ reprezintă frecvenţa impulsurilor de numărat.

Pe baza relaţiei (1.1) rezultă cele două posibilităţi de modificare a perioadei de temporizare: modificarea constantei *Cst* sau modificarea frecvenţei f_0. A doua variantă se realizează de obicei prin divizarea semnalului furnizat de generatorul de tact.

Există temporizatoare de 8 biţi, 16 biţi, 24 de biţi, etc. În funcţie de numărul de biţi al temporizatorului rezultă valoarea maximă a constantei *Cst*.

În practică, utilizarea unui temporizator implică să se facă o acţiune de fiecare dată când perioada de temporizare s-a încheiat (modificarea stării unui led, citirea stării unui pin programat ca intrare, declanşarea unei conversii analog-numerice, etc.). Pentru aceasta este necesară testarea continuă a bitului indicator (flag) şi executarea acţiunii respective în momentul când bitul devine 1. Bitul respectiv trebuie imediat şters (trecut pe 0) pentru ca ulterior să se poată detecta noua trecere pe 1.

8. Descrieţi principal funcţia de captură a unui timer (temporizator). Prezentaţi apoi pe baza schemei de mai jos, funcţionarea modulului Timer_A al familiei de nicrocontrolere MSP430x4xx în modul de lucru „captură”.

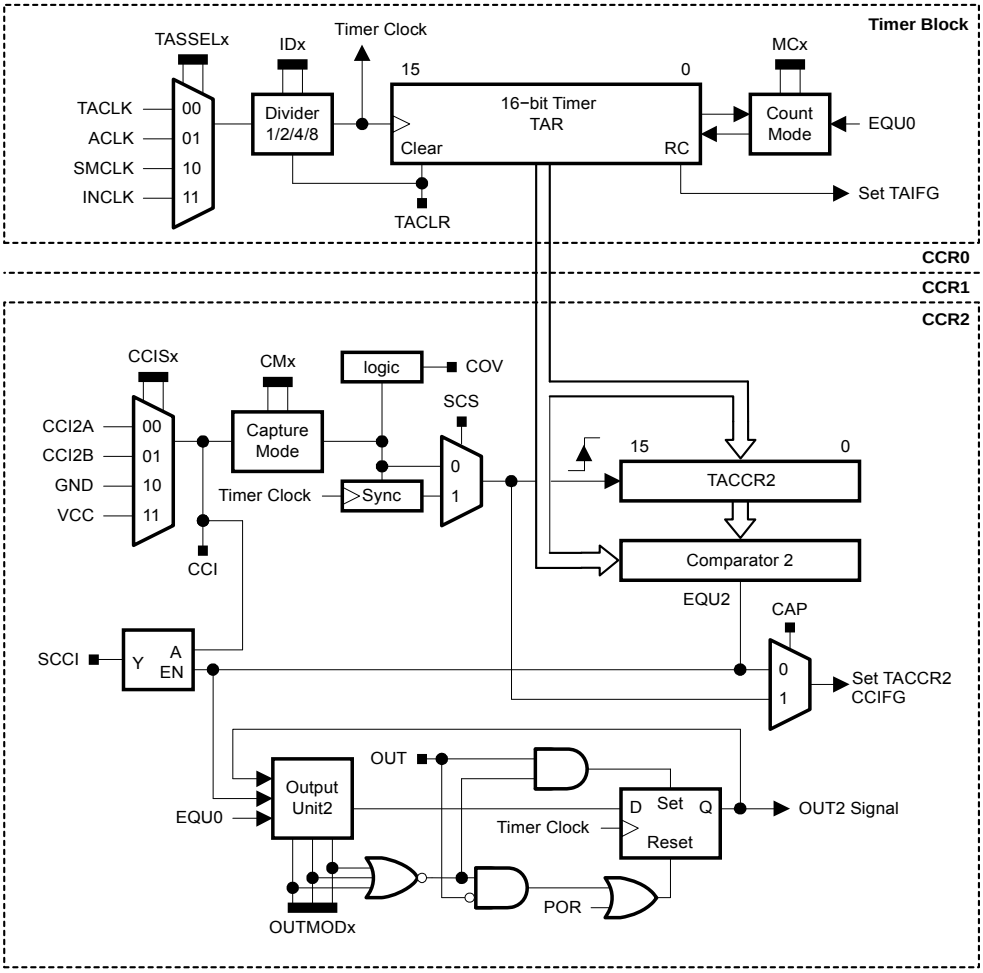


Fig. 3.7 Structura modulului Timer_A [2].

[1], pag. 18, pag. 94-95.

Funcţia de *captură* presupune existenţa unui semnal exterior microcontrolerului notat *s* (aplicat la un pin) pe lângă structura de bază care conţine numărătorul şi generatorul de semnal de tact. În acest caz numărătorul numără crescător. Operaţia de captură implică captarea (reţinerea) conţinutului numărătorului în momentele de timp corespunzătoare fronturilor semnalului *s*.

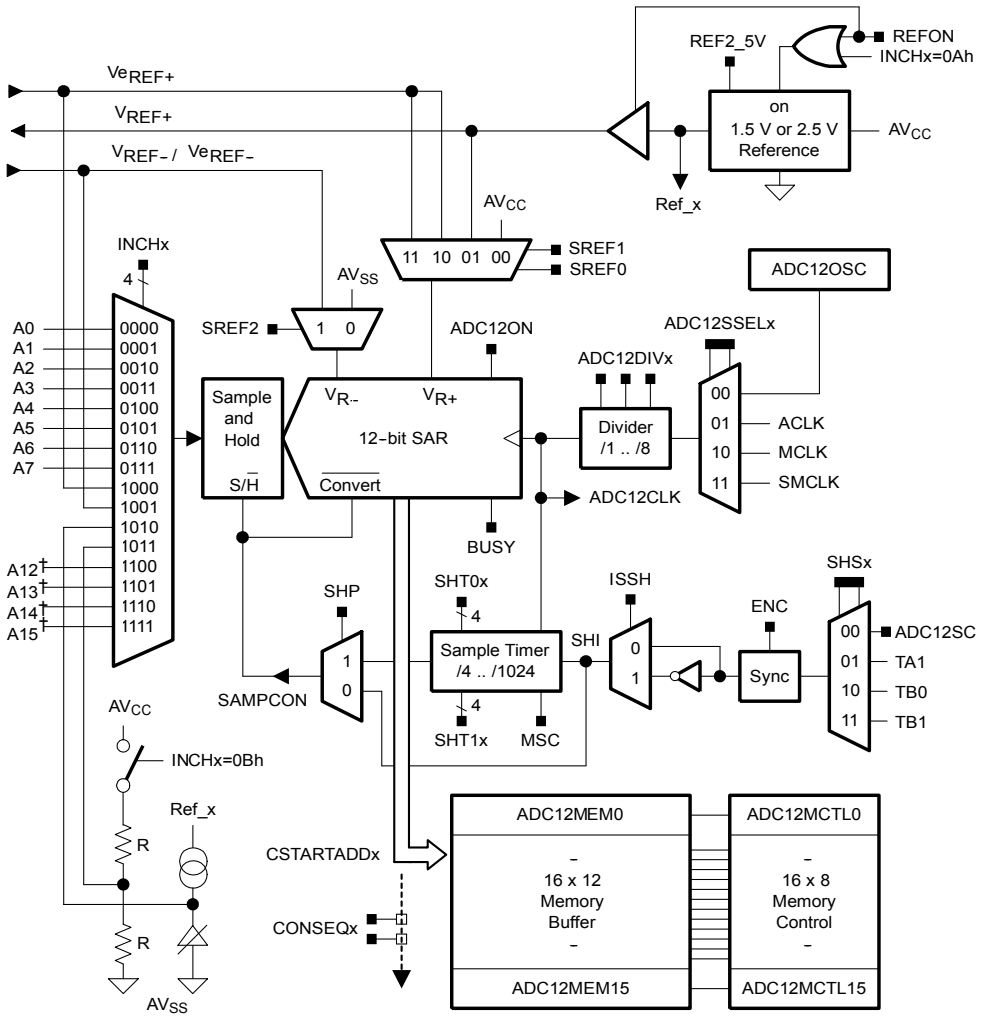
Unităţile CCR0, CCR1 şi CCR2 permit implementarea *funcţiei de captură*, când bitul CAP este pe 1 logic. Prin intermediul biţilor CCISx se selectează unul din cele 4 semnale posibile (CCIxA şi CCIxB provin de la pini). Frontul (fronturile) acestui semnal vor declanşa captura. Dacă biţii CCISx au succesiv valorile binare 11, 10, 11, 10,..., semnalul de intrare comută între Vcc şi GND. Astfel se pot face capturi fără a fi nevoie de un semnal extern. Biţii CMx permit selectarea frontului semnalului ales cu biţii

CCISx, la apariţia căruia se va face captura (crescător, descrescător sau ambele). În momentul când are loc captura:

-conţinutul registrului numărător TAR este memorat în registrul TACCRx, x=0, 1, 2;

-bitul indicator (flag) TACCRx CCIFG, x=0, 1, 2, devine 1 logic.

9. Descrieţi principial funcţia unui modul ADC (analog to digital converter) al unui microcontroler. Prezentaţi apoi pe baza schemei de mai jos (modulul ADC12 al familiei de microcontrolere MSP430x4xx) cum se măsoară temperatura.



† MSP430FG43x and MSP430FG461x devices only

Fig. 3.11 Structura ADC12 [2].

[1], pag. 21, 104, 109.

Un periferic de tip "ADC" (*Analog to Digital Converter*, convertor analog-digital sau analog-numeric) primeşte la intrare o tensiune, care poate lua orice valoare într-un interval dat, pe care o converteşte într-un număr reprezentat prin *n* biţi. Expresia numărului furnizat de ADC, notat N_{ADC}, numit şi rezultatul conversiei este

$$N_{ADC} = 2^n \frac{U_{in}}{U_{ref}}, \tag{1.4}$$

unde U_{in} reprezintă tensiunea de intrare iar U_{ref} reprezintă o tensiune de referinţă, care impune şi intervalul în care U_{in} poate lua valori, adică [0, U_{ref}). Numărul de biţi *n* are valori de tipul 8, 10, 12 sau chiar 16 şi se mai numeşte rezoluţie.

Cuanta convertorului (numită şi 1 LSB) este reprezentată prin expresia

$$q = \frac{U_{ref}}{2^n}. \tag{1.5}$$

Folosind cuanta se poate determina tensiunea de intrare în funcţie de rezultatul conversiei, $U_{in}=qN_{ADC}$. Aceasta reprezintă însă o aproximare a U_{in} deoarece determinarea rezultatului prin relaţia (1.4) implică o aproximare în sensul că N_{ADC} reprezintă de fapt cel mai apropiat întreg de numărul raţional $2^n U_{in}/U_{ref}$.

Modulul ADC12 are 16 intrări analogice (numite şi canale), A0, A1,...,A15. Dintre acestea, 12 corespund unor pini ai microcontrolerului, iar 4 sunt conectate direct la tensiuni interne (una dintre acestea, A10, selectată când biţii INCHx au valoarea 1010, corespunde unui senzor de temperatură).

Acesta furnizează o tensiune care depinde de temperatură (*Temp*), considerată în grade Celsius, conform relaţiei:

$$U_{Temp} = 0,00355 Temp + 0,986. \tag{3.5}$$

Determinarea temeraturii implică: conversia tensiunii corespunzătoare canalului 10 (adică N_{ADC}), determinarea U_{in} din (1.4), iar apoi variabila *Temp* se determină din (3.5) unde U_{Temp} este U_{in} determinat anterior.

10. Descrieţi principial interfaţa serială sincronă de tip SPI (*Serial Peripheral Interface*).

[1] pag. 24, 25.

Termenul de sincron se referă la faptul că există o linie *comună* de *semnal de tact* comandată de unul dintre dispozitive, cel care are funcţia de *master*. Şi în acest caz există câte o linie pentru transmisia în fiecare sens şi una de masă.

Comunicaţia serială sincronă între o componentă cu funcţie *master* şi o componentă cu funcţie *slave* este prezentată în figura 1.11.

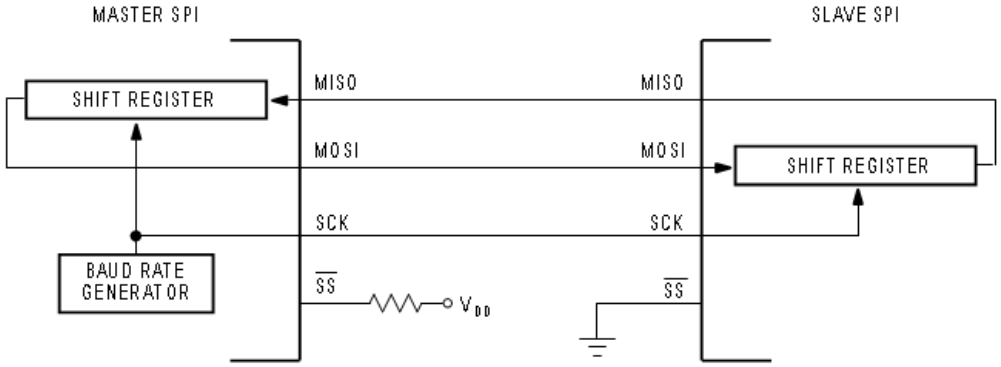


Fig. 1.11 Comunicaţia serială sincronă.

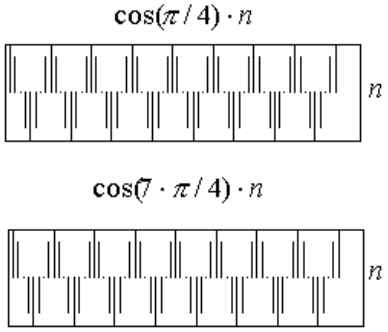
Comanda interfeţei SPI pentru funcţia *master*/*slave* se realizează prin linia de selecţie /SS (*Slave Select*). Interfaţa SPI a componentei *master* conţine circuite (*baud rate generator*) pentru generarea semnalului de tact (*Serial Clock*) SCK. Cele două dispozitive SPI conţin câte un registru de deplasare (*shift register*) de *n* biţi (valori uzuale pentru *n*: 8, 16) care sunt interconectate într-o configuraţie de registru distribuit de 2*n* biţi prin liniile de date MOSI (*Master Out*/*Slave In*) şi MISO (*Master In*/*Slave Out*), figura 1.11. Transferul de date se realizează prin deplasarea cu *n* biţi a conţinutului registrului distribuit, sincronizată cu semnalul de tact SCK, şi are ca rezultat schimbul de octeţi între cele două componente *master*/*slave*.

CIRCUITE ŞI SISTEME

ANUL II

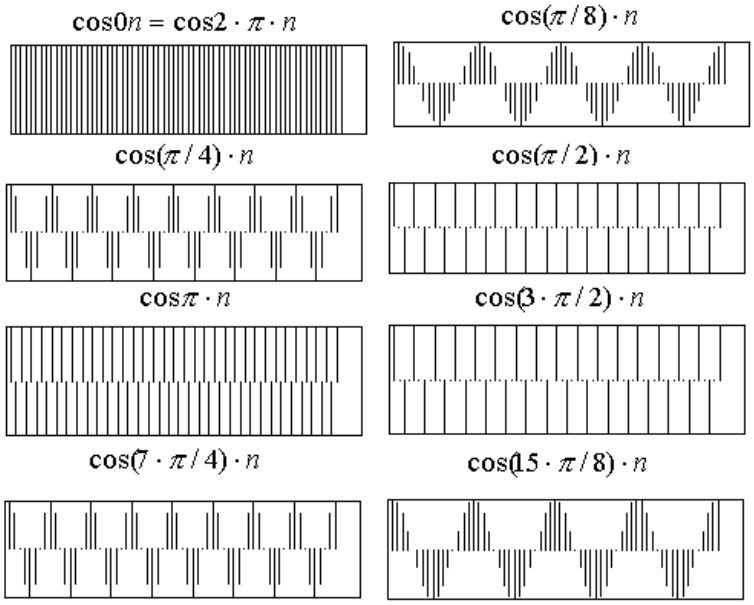
1. Confuzii datorate eşantionării. Exemple

Prin eşantionarea ideală a semnalului $x(t) = A\cos\omega_0 t$ cu pasul T_e se obține semnalul în timp discret $x[n] = A\cos\Omega_0 n$ cu $\Omega_0 = \omega_0 T_e$. Pentru diferite alegeri ale pasului de eşantionare ar trebui să se obțină semnale în timp discret diferite. Există însă alegeri diferite ale pasului de eşantionare care conduc la acelaşi semnal în timp discret. De exemplu pentru $T_{e1} = \pi/4\omega_0$ se obține semnalul în timp discret $x_1[n] = A\cos\frac{\pi}{4}n$ iar pentru $T_{e2} = 7\pi/4\omega_0$ se obține semnalul $x_2[n] = A\cos\frac{7\pi}{4}n$. Dar, datorită periodicității funcției cosinus cu perioada 2π, se poate scrie: $x_2[n] = A\cos(2\pi - \frac{\pi}{4})n = x_1[n]$. Cele două semnale în timp discret sunt reprezentate grafic în figura de mai jos.



Analizând figura se constată că cele două semnale în timp discret sunt identice.

În consecință, există alegeri diferite ale pasului de eşantionare, care pot conduce la semnale în timp discret identice, producând confuzie. Exemple similare pot fi observate şi în figura de mai jos.

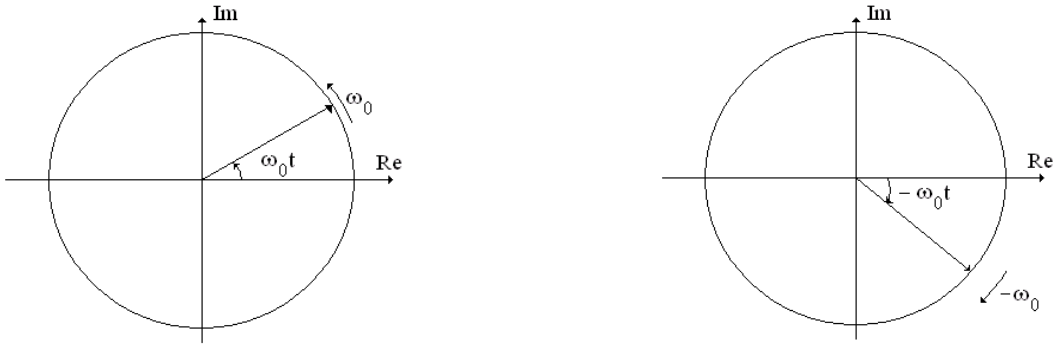


2. Semnale complexe. Fazori. Conceptul de frecvență negativă

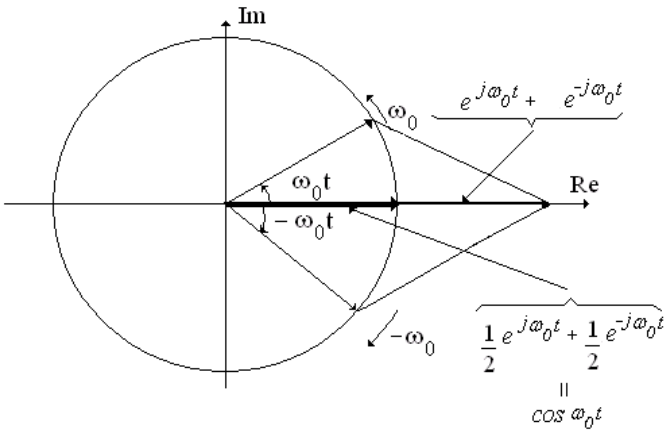
Este bine cunoscută prima formulă a lui Euler:

$$cos\,\omega_0 t = \frac{e^{j\omega_0 t} + 1/e^{j\omega_0 t}}{2}.$$

Cele două exponențiale complexe din membrul drept sunt semnale complexe care se mai numesc şi fazori. Ele pot fi reprezentate ca şi vectori rotitori în planul complex. Aceşti vectori au module unitare şi se rotesc cu viteze unghiulare ω_0 şi respectiv $-\omega_0$. Ei sunt reprezentați în figura de mai jos.



Dar ω_0 reprezintă şi pulsația (sau frecvența) semnalului $cos\,\omega_0 t$. De aceea se mai spune că frecvența fazorului $e^{-j\omega_0 t}$ este negativă. Acest concept nu are un suport fizic, dar este util pentru simplificarea calculelor. Construcția semnalului $cos\,\omega_0 t$ cu ajutorul celor doi fazori este prezentată în figura următoare.



3. Teorema proiecției. Exemplu de aplicare în teoria aproximării

Enunțul teoremei proiecției este următorul.

Fie *H* un spațiu Hilbert şi H_s un subspațiu Hilbert închis al acestuia. Oricare ar fi vectorul *x* din *H* există un vector $\tilde{x}$ din H_s care reprezintă cea mai bună aproximare a lui *x* cu elemente din H_s care are proprietățile:

- distanța de la *x* la $\tilde{x}$ este cea mai mică distanță de la *x* la orice element din H_s;
- eroarea comisă, $e=x-\tilde{x}$, este ortogonală pe subspațiul H_s.

Dacă dimensiunea spațiului Hilbert *H* este 3 şi dacă dimensiunea spațiului Hilbert H_s este 2, atunci teorema proiecției se particularizează la teorema celor trei perpendiculare aşa după cum se vede în figura de mai jos.