Cuprins

1. Intrări și Ieșiri Digitale

1.1 Scopul lucrării

În lucrarea de față se prezintă conceptul de intrare și ieșire digitală a unui microcontroler împreună cu modul de folosire a acestora în etapa de proiectare și implementare a unei aplicații software.

1.2 Prezentarea intrărilor și ieșirilor digitale

Intrările și ieșirile digitale, sau mai general, capacitatea de a monitoriza și controla în mod direct este principala caracteristică a microcontrolerelor. Astfel, practic toate microcontrolere au cel puțin 1-2 pini pentru intrări respectiv ieșiri digitale, pini care pot fi conectați direct la elemente de circuit în limitele electrice ale microcontrolerului. Câteva exemple de elemente de circuit sunt led-urile, întrerupătoarele, punțile pentru comanda motoarelor. În general, se pot găsi intre 8-32 pini pentru intrări/ieșiri digitale la majoritatea microcontrolerelor, iar unele au chiar mai mulţi.

Pinii pentru intrări/ieșiri digitale sunt grupați, în general, în *port-uri*, fiecare port reprezintă o grupare de câte 8 pini care să poată fi accesați prin intermediul unui singur octet (byte). În cazul microcontrolerului Atmega32 avem astfel 4 porturi, fiecare fiind notat cu o litera de la A la D. Un pin poate sa funcționeze doar ca intrare, doar ca ieșire sau bidirecțional: atât ca ieșire cât și ca intrare. În afară de capacitățile lor digitale I/O, majoritatea pinilor au una sau mai multe funcționalități alternative. Toate celelalte module ale microcontroler-ului cum ar fi modulul analogic sau timer-ele utilizează de fapt funcțiile alternative ale intrărilor și ieșirilor digitale.

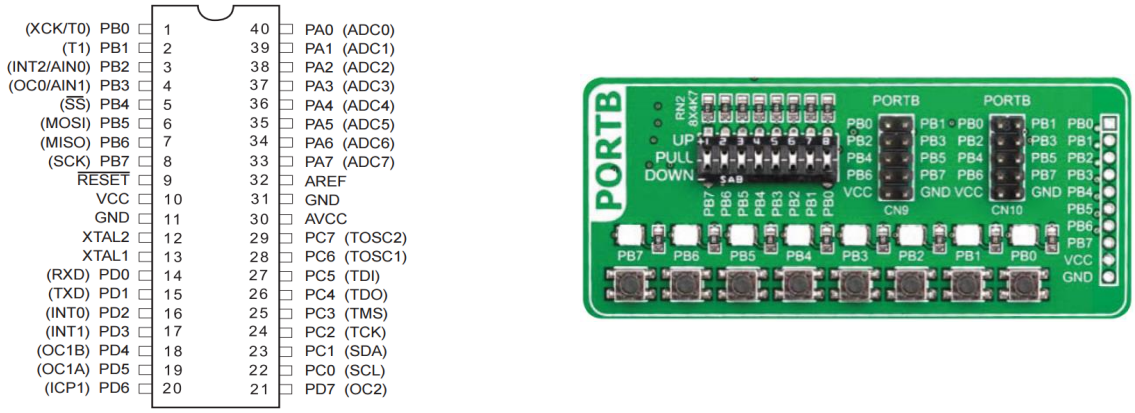


Figura 1.1. Configurația pinilor la ATmega32

Pentru fiecare pin în parte se poate selecta funcţia pe care acesta sa o aibă: pin digital sau alta funcţie alternativă. Desigur că în cazul în care se folosește o funcționalitate alternativă a unui pin acesta își pierde funcționalitatea de pin digital, și vice-versa, în cazul în care un pin este utilizat ca pin digital atunci acesta nu poate fi folosit cu funcția alternativă. Din acest motiv, proiectantul aplicației hardware trebuie să aleagă cu atenție funcționalitatea fiecărui pin în parte. În acest laborator ne vom concentra asupra funcționalităților de intrări/ieșiri digitale ale pinilor.

Placa de dezvoltare EasyAVR v7 are asociate fiecărui port cate 8 butoane, 8 LED-uri și 8 DIP switch-uri pentru selectarea rezistențelor de PULL-UP sau PULL-DOWN externe. Toate acestea sunt grupate împreună ca în Figura1.1 dreapta.

Ca prim pas, vom începe prin explicarea termenului *digital*. Dacă s-ar măsura nivelul tensiunii unui pin cu ajutorul unui Voltmetru (diferența de potential față de masă - GND) s-ar observa un anumit nivel de tensiune. Cu toate acestea microcontroler-ul transferă în domeniul digital această tensiune transformând-o în niveluri logice: 1 logic (HIGH) respectiv 0 logic (LOW). Astfel, când vorbim despre valoarea unui pin digital, spunem fie că este 1 sau 0, LOW sau HIGH.

În Figura1.2 se prezintă valorile standardizate pentru cele două stări logice la familia de circuite CMOS.

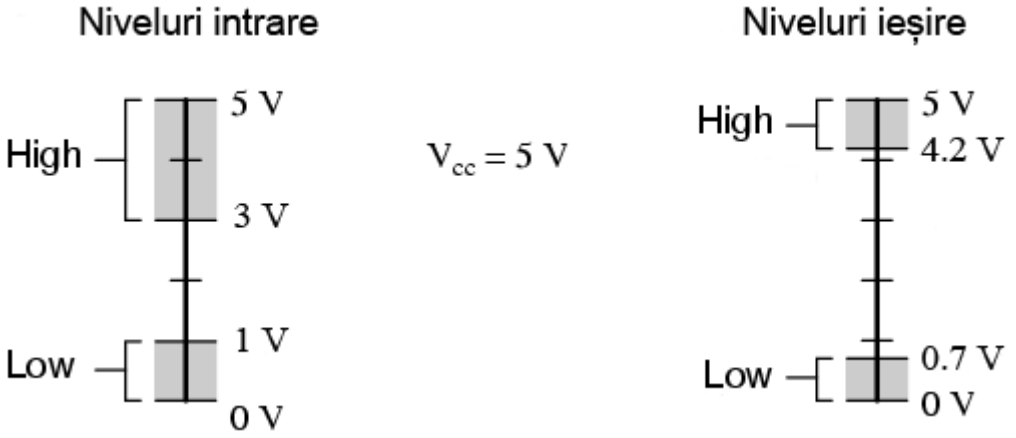


Figura 1.2. Nivelurile de tensiune CMOS.

În ceea ce privește pinii I/O digitali, există trei regiștrii prin care se controlează comportarea acestora așa cum sunt ilustrați în Figura1.3. În continuare fiecare dintre aceștia vor fi explicați:

Port A Data Register – PORTA

Bit	7	6	5	4	3	2	1	0	
	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
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	

Port A Data Direction Register – DDRA

Bit	7	6	5	4	3	2	1	0	
	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
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	

Port A Input Pins Address – PINA

Bit	7	6	5	4	3	2	1	0	
	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Figura 1.3. Descrierea regiștrilor PORT, PIN și DDR.

Registrul DDR (Data Direction Register): Fiecare *port* este asociat cu un registru DDR care are câte un bit asociat pentru fiecare pin al acelui port. Din acest registru se setează direcția unui pin (intrare sau ieșire) prin setarea valorii bitului asociat la 0 sau 1 logic. Pentru a seta un pin ca și intrare se resetează la 0 bit-ul corespunzător pin-ului respectiv. Pentru a seta

un pin ca ieșire se setează la 1 bit-ul corespunzător pin-ului respectiv. Pinii unui port pot fi configurați diferit, unii pini de intrare iar alții de ieșire. După o resetare (reset) a microcontrolerului, biții registrului DDR sunt inițializați ca și intrari (0 logic).

Exemplu: Setare a unui registru DDR:
DDRC = 0b11101000;

Efectul acestei operații se va observa asupra funcționalității pinilor din portul C al microcontrolerului. Sintaxa "0b" precizează că ceea ce urmează este o valoare binară (baza 2). Astfel că pinii sunt setați după cum urmează:

- pinii PC0 – PC2 și PC4 sunt setați ca fiind pini de intrare (vezi Figura1.3);
- pinii PC3 și PC5 – PC7 sunt setați ca fiind pini de ieșire.

Registrul PORT (Port Register): Este un registru pe 8 biți care conține un bit asociat fiecărui pin din acel *port*. Din acest registru se setează nivelul tensiunii unui pin de ieșire, HIGH respectiv LOW. Considerând un pin de ieșire, prin setarea bitului asociat din PORT la 1 logic, nivelul pinului va fi HIGH iar resetarea bitului la 0 va determina un nivel LOW pentru acel pin.

În cazul în care considerăm un pin setat ca intrare, funcționalitatea registrului PORT depinde de tipul microcontrolerului. În cazul microcontrolerului ATmega32, daca un pin este setat ca intrare, setarea la 1 a bit-ului asociat din registrul PORT va activa rezistenta pull-up internă corespunzătoare. Semnificația rezistenței de pull-up va fi explicată ulterior.

Exemplu:
DDRC = 0b11111100;
PORTC = 0b11100001;

Prima instrucțiune va seta pinii PC0 – PC1 drept intrări și PC2 – PC7 drept ieșiri. Conform cu a doua instrucțiune pinii de ieșire PC2 – PC4 vor avea un nivel logic LOW iar pinii de ieșire PC5 – PC7 vor avea un nivel HIGH. În cazul pinilor de intrare, PC0 va avea rezistența pull-up corespunzătoare activă iar pentru PC1 rezistența pull-up va fi dezactivată.

Registrul PIN (Port Input Register): Acest registru este un registru care poate fi doar citit (Read-Only). Acesta conține starea curentă a fiecărui pin din portul respectiv, indiferent daca pin-ul este de intrare sau de ieșire. Se folosește cel mai adesea pentru a verifica starea logică a unui pin de intrare dar poate fi folosit și pentru citirea stării unui pin de ieșire pentru a verifica efectul unei comenzi asupra pinului. În general, scrierea acestui registru nu are nici un efect.

La manipularea acestor regiștri trebuie luat în considerare că fiecare bit din regiștrii este asociat unui pin. Astfel, atunci când se dorește schimbarea comportării unui singur pin trebuie acționat fără a altera comportarea celorlalți pini. În acest scop este recomandată folosirea operațiilor pe biți.

1.2.1 Digital Input

Intrările digitale sunt folosite în cazurile în care se dorește monitorizarea pinilor din punct de vedere digital dar mai exact pentru detectarea schimbării stării pinului între 0 și 1 logic. În ceea ce privește stările logice, acestea depind de anumite niveluri de tensiune cum a fost prezentat și în Figura1.2, aceste nivelurile de tensiune trebuie sa fie în conformitate cu specificațiile microcontroler-ului.

Pentru a evita zona nedefinită, care poate însemna fie HIGH fie LOW, se folosesc anumite elemente de circuit precum rezistente de pull-up interne (integrate în circuitul intern microcontroler-ului) sau externe și circuite Trigger-Schmitt.

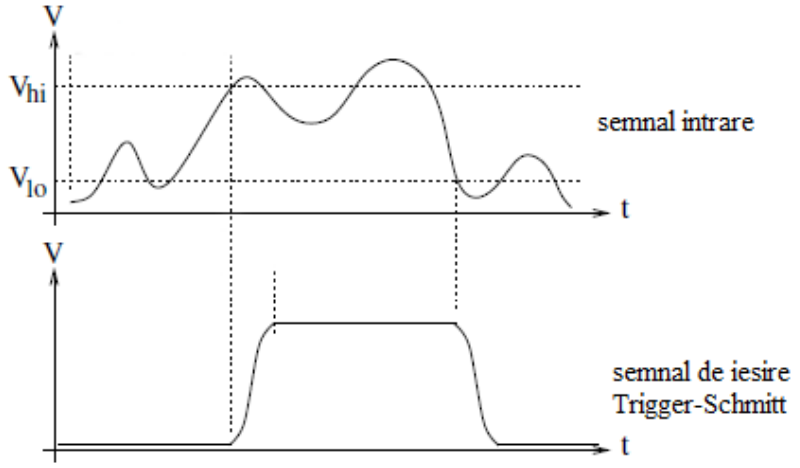


Figura 1.4. Intrarea și ieșirea unui circuit Trigger-Schmitt.

Circuitele Trigger Schmitt sunt componente folosite pentru a transpune în domeniul digital semnalele analogice. Pentru a realiza acest lucru circuitul Trigger-Schmitt are două praguri de tensiune V_{lo} și V_{hi} , unde $V_{lo} < V_{hi}$ și își modifică ieșirea din starea LOW în starea HIGH dacă amplitudinea semnalului este mai mare decat V_{hi} , respectiv din HIGH în LOW pentru cazul în care amplitudinea semnalului este mai mica decât pragul V_{lo}. Astfel circuitul Trigger-Schmitt nu va avea la ieșire fluctuații de tensiune, semnalul de ieșirea având bine definiți timpii de creștere și de scădere indiferent de semnalul de intrare. Funcționarea circuitului Trigger-Schmitt este prezentată în Figura1.4.

Pe placa EasyAVR v7 starea logică a pinilor de intrare digitală poate fi schimbată folosind butoanele asociate port-urilor. Switch-ul din Figura1.5 este folosit pentru a alege starea logică care urmează sa fie aplicată pe pin-ul microcontrolerului în momentul apăsării butonului corespunzător. Dacă de exemplu, switch-ul SW1.2 este plasat pe poziția de sus (VCC), atunci când se va apăsa un buton din grupul PORTB se va aplica o stare logică 1 (HIGH) pe pinul corespunzător. Se folosește aceeași logică pentru poziția de jos (GND).

Figura 1.5. Switch-uri pentru selectarea starii logice a butoanelor.

1.2.2 Rezistențele de pull-up

Aceste rezistențe sunt de multe ori integrate in circuitul interior al microcontrolerelor. Majoritatea microcontrolerelor oferă rezistențe de pull-up iar unele oferă atât rezistențe pull-up cât și pull-down. Scopul rezistentelor de pull-up este de a conecta un pin la un nivel de tensiune definit (HIGH în acest caz) pentru cazul în care pinul nu este comandat de o componentă hardware externă. În cazul microcontroler-ului Atmega32 rezistențele de pull-up sunt controlate prin registrul PORT (în cazul pinilor setati ca intrare) de unde se poate activa sau dezactiva pentru fiecare pin în parte.

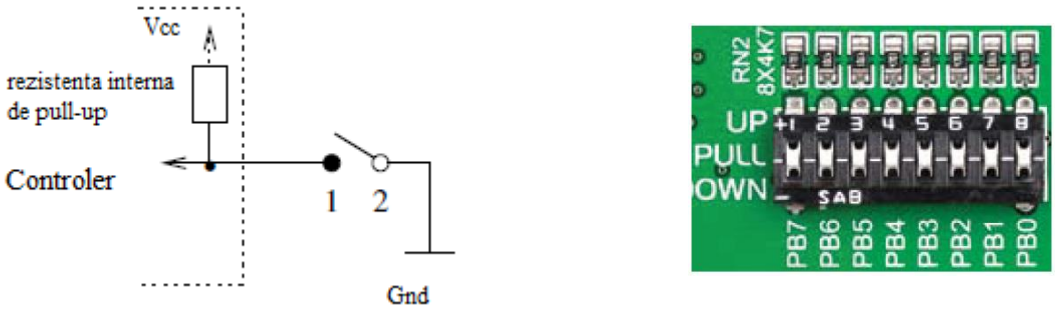


Figura 1.6. Rezistența internă de pull-up.

În Figura1.6 am conectat un intrerupator unui pin, pin-ul are activată rezistența de pull-up. Cât timp întrerupătorul este deschis pin-ul este conectat prin rezistența de pull-up la Vcc ca urmare controlerul va citi starea HIGH. În momentul în care întrerupătorul se închide, pinul va fi conectat la GND, iar starea pin-ului va fi LOW. În acest fel se evită cazul în care întrerupătorul ar fi deschis și pinul ar fi lăsat "în aer", ceea ce ar determina ca nivelul pinului să fie influențat de mediul înconjurător.

Placa de dezvoltare EasyAVR v7 folosește switch-uri cu trei stari ca în Figura1.6 dreapta, pentru fiecare port în parte, cu scopul de a activa rezistențe externe de 4.7 kΩ de pull-up sau pull-down pentru fiecare pin asociat portului. Fiecare din aceste switch-uri are trei stări:

1. poziția de mijloc dezactivează rezistențele de pull-up și pull-down asociate portului;
2. poziția de sus activează rezistența pentru starea de pull-up asociată pinului;
3. poziția de jos activează rezistența pentru starea de pull-down asociată pinului.

1.2.3 Digital Output

Setarea unui pin ca ieșire digitală se folosește în cazurile în care se dorește un nivel logic cerut (LOW sau HIGH) pentru un anumit pin. Nivelurile corespunzătoare celor două stări diferă la fiecare controler în parte și depinde de tensiunea de alimentare. Pentru ATmega32 la o tensiune de alimentare V_{cc} = 5V, tensiunea maximă pentru starea LOW este 0.7V iar tensiunea minimă pentru starea HIGH este 4.2V.

În momentul în care un pin este setat ca ieșire în registrul DDRx, starea pinului este setată conform registrului PORT. În momentul în care folosim un pin ca ieșire, trebuie acordată atenție curentului care iese sau intră din/în pin, și evitării situațiilor de scurt-circuit. Curentul maxim admis pentru un pin este de 20 mA și se adaugă condiția ca pentru regim staționar, suma tuturor curenților de pini să nu depășească 200 mA. Microcontrolerele suportă scurt-circuitul pentru momente scurte de timp dar prelungirea acestei situații poate duce la defectarea pinului sau chiar a microcontrolerului.

Pe placa EasyAVR, fiecare pin dintr-o grupare port are asociat câte un LED de putere mică (consum de 0.3 mA) care se aprinde daca starea pinului de ieșire este programată ca 1 logic (HIGH) sau daca butonul asociat este apăsat. Pentru activarea functionalității LED-urilor dintr-un anumit port se folosesc pozițiile 5-8 ale switch-ului SW10 din Figura1.7.

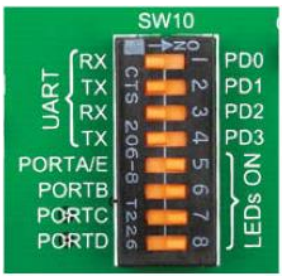


Figura 1.7. Switch-ul SW10 pentru activarea LED-urilor.

1.3 Desfășurarea lucrării:

Exerciții:

1. Să se aprindă LED-ul asociat pinului PB7 atâta timp cât butonul asociat pinului PB3 este apăsat.
   - Se setează pin-ul PB7 ca ieșire iar PB3 ca intrare în secțiune de configurări a funcției main.
   ```
   DDRB |= (1<<7);          //PB7 – Ieșire
   DDRB &= ~(1<<3);         //PB3 – Intrare
   ```
   - În bucla infinită se testează dacă starea logică a pin-ului de intrare PB3 este 1 (HIGH). Dacă este adevărată condiția, atunci se va aprinde LED-ul. Altfel, LED-ul se va stinge.
   ```
   if(PINB & (1<<3))        //testare pin PB3
      PORTB |= (1<<7);      //PB7 – HIGH (LED ON)
   else
      PORTB &= ~(1<<7);     //PB7 – LOW (LED OFF)
   ```
2. Să se schimbe starea LED-ul asociat pinului PB6 de fiecare dată când butonul asociat pinului PB2 este apăsat.
   - Se setează pinii PB6-ieșire, PB2-intrare în secțiunea de configurări a funcției main.
   ```
   DDRB |= (1<<6);          //PB6 – Ieșire
   DDRB &= ~(1<<2);         //PB2 – Intrare
   ```
   - În bucla infinită se testează dacă starea logică a pin-ului de intrare PB3 este 1 (HIGH). Dacă este adevărată condiția și daca starea anterioară a pin-ului a fost 0, atunci se schimbă starea LED-ului.
   ```
   if(PINB & (1<<2))        //testare pin PB2
   {  if(starePB2 == 0)     //variabila va ține minte starea
      {   starePB2 = 1;     //anterioară a pin-ului PB2
          PORTB ^= (1<<6);//se schimbă starea LED-ului
      }
   }else
      starePB2 = 0;
   ```
3. Să se incrementeze o variabilă la fiecare apăsare a butonului PB2 si valoarea acesteia să se codifice pe LED-urile din PORTD. Astfel se vor contoriza valori între 0 și 255.
   - Se setează toți pinii din PORTD ca ieșiri digitale iar pin-ul PB2 ca intrare digitală.
   ```
   DDRD |= 0b11111111;     //Pini de ieșire
   DDRB &= ~(1<<2);         //PB2 – Intrare
   ```
   - În bucla infinită, la fiecare apăsare a butonului asociat pin-ului PB2 se va incrementa o variabilă n.
   ```
   if(PINB & (1<<2))        //testare pin PB2
   {  if(starePB2 == 0)     //variabila va ține minte starea
      {    starePB2 = 1;    //anterioară a pin-ului PB2
           n++;             //se incrementează variabila n
      }
   }else
      starePB2 = 0;
   ```
   - Valoarea lui n se codifică în baza 2 pe led-urile din PORTD.
   ```
   PORTD = n;
   ```

2. Display 7-segmente

2.1 Scopul lucrării.

Această lucrare prezintă principiul de funcționare al unui *display 7-segmente*, variantele constructive și modul de control al unui astfel de dispozitiv folosind pinii de ieșire digitali ai microcontrolerului. Aplicațiile prezentate se bazează pe folosirea unei placi de dezvoltare EasyAVR v7 cu microcontroler Atmega32 si un display 7-segmente cu 4 digiți.

2.2 Prezentarea modulelor 7-segmente

Un digit cu 7 segmente este compus din 7+1 LED-uri aranjate într-un mod specific pentru a putea afișa cifrele de la 0 la 9 si câteva litere de asemenea. Al 8-lea led este folosit pentru punctul zecimal în cazul în care trebuie afișat pentru digitul respectiv. Placa EasyAVR v7 este prevăzută cu un display 7-segmente cu 4 digiți grupați ca în Figura2.1.

Figura 2.1. Display BCD 7 Segmente pe placa EasyAVR v7

Fiecare din cele 7 LED-uri are unul din pini (anod sau catod) ieșit în exteriorul învelișului de plastic și este notat cu o literă de la *A* la *G*. Ceilalți pini ai LED-urilor sunt legați împreună pentru a forma un pin comun. Acest pin comun stabilește tipul afișajului. Pentru că fiecare LED are 2 pini de conexiune, unul denumit *anod* si celălalt *catod*, se deosebesc două tipuri de afișaje: cu **Catod Comun** (CC) și cu **Anod Comun** (CA). Diferența între cele două, așa cum sugerează și numele, este că la configurația CC, catodul celor 7-segmente este legat impreună iar la configurația CA, anodul celor 7-segmente este legat împreună. Cele două variante constructive sunt reprezentate în Firura 2.2.

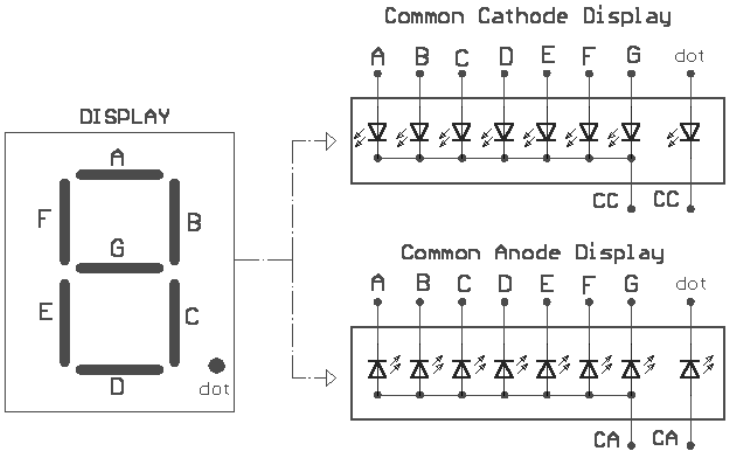


Figura 2.2. BCD 7 Segmente – variante constructive.

Pentru configurația Catod Comun, pinii de catod de la cele 7-segmente formeaza pinul comun de activare a digitului (enable) și se leagă la un potențial LOW. Segmentele individuale luminează aplicând potențial HIGH pe fiecare pin de anod individual. Pentru configurația Anod Comun, pinii de anod de la cele 7-segmente formeaza pinul comun de activare a digitului (enable) și se leagă la un potențial HIGH. Segmentele individuale luminează aplicând potențial LOW pe fiecare pin de catod individual.

Pentru cazurile în care avem un afișaj 7-segmente cu 4 digiți, așa cum este și cazul de față, cel mai adesea, cei 4 digiți au legați pinii pentru segmente împreună ca în Figura 2.3. Se folosește această procedură pentru a folosi pentru comandă un numar cât mai mic de pini ai microcontrolerului. Comanda unui astfel de dispozitiv se face folosind tehnici de multiplexare. Prin multiplexarea datelor între cei 4 digiți cu o frecvență suficient de mare (>60 Hz), se creează o iluzie optică și se pare că toți cei 4 digiți sunt aprinși simultan. Sunt folosite aceleași linii de date pentru selecția segmentelor și astfel același segment din fiecare digit este legat în paralel cu ceilalți. Totuși, fiecare digit are o linie de selecție separată care este folosită pentru activarea digitului pentru care se trimit datele la un moment dat. Liniile de selecţie a digitului sunt legate la pinii PA0-PA3 ai microcontrolerului Atmega32 și liniile de date pentru selecția segmentelor sunt legate la pinii PC0-PC7.

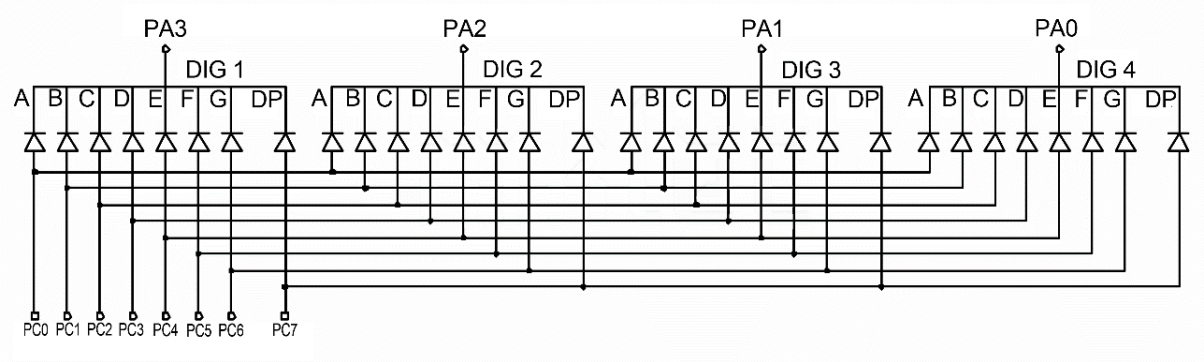


Figura 2.3. Schema 7-segmente cu catod comun și 4 digiți.

Pe placa EasyAVR v7 găsim un afișaj 7-segmente cu catod comun pentru care liniile de selecție a digitului se activează cu potențial LOW iar liniile de date pentru selecția segmentelor se activează cu potențial HIGH. Pentru a simplifica munca programatorului, pe această placă de dezvoltare, liniile de selecție a digitului sunt conectate printr-un montaj inversor cu tranzistor, care schimbă starea logică (din LOW în HIGH) ca în Figura2.4 mijloc. Astfel, activarea unui digit se face cu nivel HIGH. “1” logic pe liniile DIS0-3 conduce la saturarea tranzistoarelor Q0-3 și conectarea liniilor COM0-3 la masă (GND). De aceea, atât liniile de selecție cât și liniile de date se activează cu potențial HIGH.

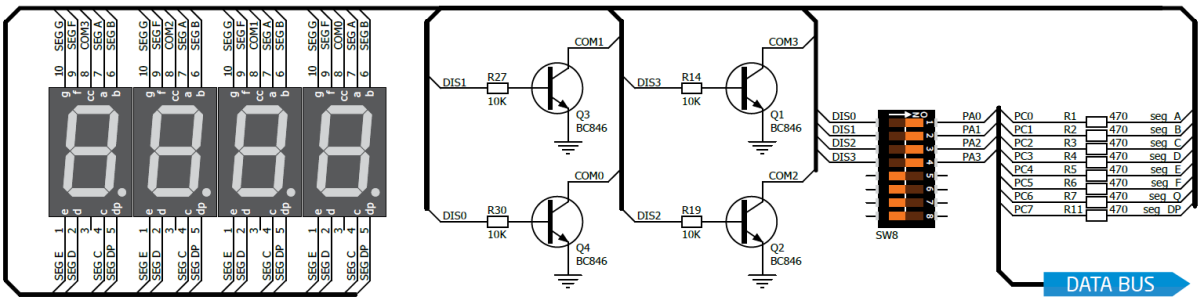


Figura 2.4. Schema circuitului pentru display-ul 7-segmente cu 4 digiți.

Pentru a activa afișajul 7-segmente de pe placă este nevoie să avem poziționat switch-ul SW8 ca și în Figura 2.5. Primele 4 poziții ale switch-ului permit conectarea pinilor PA0-PA3 cu liniile de selecție a digitului (DIS0-DIS3). Astfel că, înainte de a testa programele pentru afișarea pe display, asigurați-vă că switch-ul este poziționat corect.

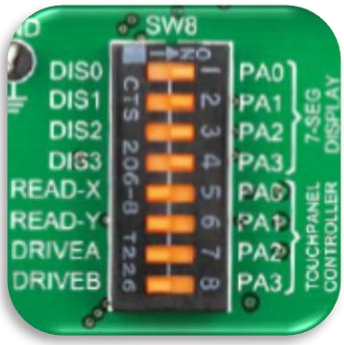


Figura 2.5. DIP Shitch SW8 pentru controlul liniilor de activare digit.

2.3 Desfășurarea lucrării:

Lucrarea de față își propune integrarea afișajelor 7-segmente în proiectele cu microcontroler pentru afișarea diferiților parametrii ai programului în vederea interacțiunii cu utilizatorul. Ne propunem să afișăm cifrele de la 0 la 9 și pentru aceasta, avem nevoie să știm cum să codificăm aceste caractere pe segmentele afișajului (Figura 2.6). Segmentele A-G si DP (punctul zecimal) sunt controlate de registrul PORTC așa cum se poate vedea în Figura 2.3 si 2.4. Astfel că pentru a codifica, de exemplu, cifra 0 pe afișaj vom atribui registrului PORTC următoarea valoare:

```
            gfedcba
PORTC = 0b00111111;
```

Acolo unde bit-ul este 1, LED-ul corespunzător va fi aprins și acolo unde bit-ul este 0, segmentul este stins. Pentru a activa un digit, linia de selecție asociată trebuie să fie “1” logic. Astfel că, pentru ca digit-ul 4 să fie activ, trebuie ca bitul 3 din PORTA sa fie 1.

```
PORTA = 0b00001000;
```

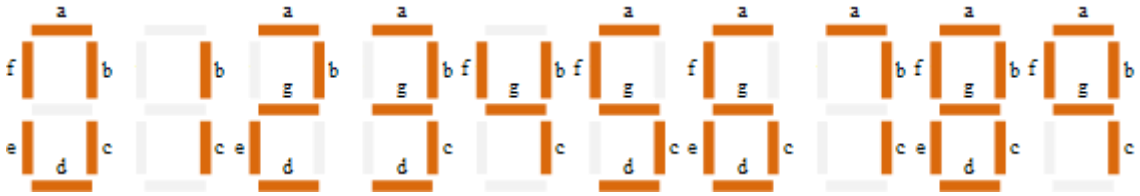


Figura2.6. Codificarea cifrelor pe un display BCD 7 segmente.

Revenind la tehnicile de multiplexare, trebuie amintit că pentru ochiul uman există o frecvență de prag dincolo de care nu se mai percepe rata de reîmprospătare. Această frecvență este cuprinsă între 60Hz și 100Hz. Pentru o frecvență de 60Hz corespunde o perioadă de 16ms care se refera la timpul necesar pentru afisarea secvențială a celor 4 digiți. Astfel putem deduce timpul necesar pentru ca un digit să fie activ: 4 ms. După fiecare 4 ms se va activa un nou digit cu caracterul corespunzător afișat.

Orice frecvență de reîmprospatare mai mare de 60Hz este acceptată.

Exerciții:

1. Să afișeze pe display un număr de 4 cifre.
   - Se setează pinii PA0-PA3 și PC0-PC7 ca ieșiri digitale în secțiune de configurări a funcției main.

```
DDRA |= 0b00001111;  //PA0-PA3 – Ieșiri
DDRC |= 0b11111111;  //PC0-PC7 – Ieșiri
```

   - În bucla infinită se activează digitul 1 prin setarea bit-ului 3 din PORTA și se setează combinația desegment pentru afișarea cifrei 1 folosind PORTC.

```
PORTA = 0b00001000;     //PA3 – HIGH (activare DIGIT 1)
PORTC = 0b00000110;     //PC0-PC7 cod pentru cifra 1
Delay_ms(4);
```

   - Se completează codul de mai sus cu instrucțiunile pentru digitul 2, 3 și 4.

2. Să se creeze o funcție care sa afișeze un caracter predefinit pe un digit al display-ului.
   - Se setează pinii PA0-PA3 și PC0-PC7 ca ieșiri digitale în secțiunea de configurări a funcției main.

```
DDRA |= 0b00001111;  //PA0-PA3 – Ieșiri
DDRC |= 0b11111111;  //PC0-PC7 – Ieșiri
```

   - Se construiește funcția de afișare înaintea funcției main; primul switch se completează până la ***case 9*** cu codificarea cifrei corespunzătoare iar al doilea, până la ***case 4*** pentru activarea fiecărui digit în parte.

```
void display(char p, char c)
{
PORTA &= 0b11110000;   //liniile de sel. digit - oprite
PORTC &= 0b00000000;   //liniile de sel. Seg. - oprite


switch(c){
     case 0:
          PORTC |= 0b00111111;break;//codificare cifra 0
          ..............................
     }
switch(p){
     case 1:
          PORTA |= 0b00000001;break;//activare digit 1
     case 2:
          PORTA |= 0b00000010;break;//activare digit 2
          ..............................
     }
     Delay_ms(4);
}
```

   - În bucla infinită se apelează funcția ***display*** pentru fiecare digit în parte. Variabila value se declara global de tip ***int*** înaintea funcției main.

```
display(1,(value%10);
display(2,(value/10)%10);
display(3,(value/100)%10);
display(4,(value/1000)%10);
```

3. Folosind și cunoștințele de la lucrarea de laborator anterioara, implementați un program care sa incrementeze o variabilă la fiecare apăsare a butonului PA7 și să afișeze valoarea variabilei pe display.

3. Modulele de Temporizare (Timere)

3.1 Scopul lucrării.

Se prezintă structura internă a unui modul de temporizare, modul de configurare al regiștrilor acestuia și diferitele moduri de operare. Este prezentată şi o aplicaţie care funcționează ca un ceas digital pe baza unui timer pe 8 biti.

3.2 Prezentarea circuitelor de temporizare

Modulele de temporizare sunt în esență constituite dintr-un numărător, majoritatea microcontrolerelor punând la dispoziție unul sau mai multe timere cu o rezoluție pe 8 sau/și 16 biți. Timer-ele sunt o parte importantă a microcontrolerelor fiind folosite într-o varietate mare de aplicații dependente de timp: de la simpla măsurare a unei perioade de timp și până la generarea de semnale cu perioadă sau frecvență impusă. Un timer poate fi folosit cu funcția sa de bază, ca și numărător dar în general timer-ele permit utilizatorului și alte funcții speciale cu ajutorul mecanismelor hardware implementate. Mecanismul de comparare permite controlul unor semnale de ieșire sau chiar generarea semnalelor PWM (Pulse Width Modulation); mecanismul de captură permite monitorizarea unor semnale de intrare, datarea sau marcarea în timp a unui eveniment extern; numărătoarele interne permit generarea de referințe de timp interne, necesare în bucle de întârziere și chiar generarea întreruperilor după trecerea unui anumit număr de cicluri de ceas sau perioade de timp.

3.2.1 Numărătorul

Fiecare timer este, la bază, un numărător care este incrementat sau decrementat la fiecare semnal de ceas, direcția de numărare fiind fixă sau configurabilă. Registrul de numărare poate fi citit sau scris la orice moment de timp de către utilizator. Pentru un timer cu o rezoluție de *n* biți, numărarea se face în intervalul [0, $2^n - 1$]. În modul normal de funcționare, numărătorul se resetează la atingerea valorii maxime de numărare ($2^n - 1$) și astfel registrul de numărare va fi setat la 0. Totuși, limita maximă de numărare poate fi programată la orice valoare din intervalul [0, $2^n - 1$], prin scrierea unei noi valori (m de exemplu) într-un registru dedicat al timer-ului. Ca urmare, domeniul de numărare al numărătorului va fi [0, m-1].

Chiar dacă un timer folosește în general aceeași sursă de ceas cu cea a microcontrolerului în sine, totuși se pot permite și alte surse de ceas pentru timer.

3.2.2 Timer având ca semnal de tact ceasul sistemului (Ceasul Intern)

Acesta este modul predefinit de funcționare al timer-ului. În acest mod, timer-ul este incrementat la fiecare front ascendent întâlnit pe semnalul de ceas al microcontrolerului.

3.2.3 Timer cu prescalare (Prescaler)

În acest mod se folosește de asemenea ceasul intern al sistemului dar acesta este divizat printr-un numărător de prescalare (*prescaler).* Un *prescaler* este de fapt un alt numărător, de o lungime variabilă (valorile tipice fiind 8 sau 10 biți), care este incrementat odată cu ceasul sistemului. Astfel, timer-ul va folosi ca semnal de ceas, semnalul provenit de la unul din biții numărătorului. Daca, de exemplu, prescaler-ul este incrementat la fiecare front crescător al ceasului sistemului, atunci cel mai nesemnificativ bit va avea perioada dublă față de cea a

ceasului sistemului. În acest caz se folosește o valoare a prescaler-ului egală cu 2 iar timer-ul va opera la jumătate din frecvența ceasului sistemului. Al doilea cel mai nesemnificativ bit va diviza frecvența din nou la jumătate, si tot așa pentru toți ceilalți biți. Modulul timer oferă posibilitatea de alegere a unei valori predefinite pentru prescaler (8, 32, 64, 256, ...). Scopul prescaler-ului este de a extinde intervalul de timp în care numără timer-ul. De exemplu, pentru un timer pe 8 biți și o frecvență de ceas de 1 MHz, domeniul de timp pentru un ciclu complet de numărare este de 255 μs, în timp ce, dacă folosim un prescaler de 1024 domeniul de timp va fi de aproximativ 260 *ms*.

3.2.4 Timer având ca semnal de tact semnale externe (Acumulatorul de Impulsuri)

În acest mod, timer-ul primește semnalul de ceas ca un semnal extern care este conectat pe un pin de intrare specific al microcotrolerului. Timer-ul incrementează valoarea de numărare la fiecare front crescător al acestui semnal. Există o constrângere în acest caz, și anume, timpul dintre două fronturi trebuie să fie mai mare decât perioada ceasului intern. În acest mod, timer-ul poate fi folosit pentru numărarea unor evenimente externe reprezentate prin fronturi crescătoare.

3.2.5 Timer având ca semnal de tact un oscilator cu cristal de cuarț extern

Deseori, un timer folosește semnalul de ceas obținut cu ajutorul unui cristal de cuarț conectat pe 2 pini predefiniți ai controlerului. Acest mod este proiectat în general pentru folosirea unui cristal de cuarț de 32.768 kHz folosit la ceasuri digitale și poate fi folosit pentru implementarea unui ceas de timp real (RTC – Real-Time Clock).

3.2.6 Captura la intrare

Funcția de *captură la intrare* este folosită pentru a marca în timp un eveniment (de cele mai multe ori extern), care poate să fie constituit de un front crescător sau/și descrescător sau de un anumit nivel logic. Atunci când are loc acest eveniment pe pinul *ICn* al microcontrolerului, timer-ul copiază automat valoarea curentă a numărătorului într-un registru *ICR* (Input Capture Register), care poate fi citit de program. La momentul evenimentului, timer-ul setează de asemenea și un *flag* care poate fi folosit pentru generarea unei întreruperi pentru a anunța programul că a avut loc un eveniment de captură la intrare. Pot exista unul sau mai mulți pini de acest fel. Un astfel de pin de captură poate fi folosit de asemenea și ca pin de intrare digitala de uz general.

3.2.7 Comparare la ieșire

Funcționalitatea de *comparare la ieșire* (OC - Output Compare) este chiar opusul *capturii la intrare.* Pentru captura la intrare, marcajul de timp este memorat atunci când are loc un eveniment extern pe linia de intrare. La compararea la ieșire, se va genera un eveniment la ieșire atunci când s-a împlinit o perioadă de timp prestabilită. Timer-ul conține un *registru de comparare la ieșire* (OCR – Output Compare Register) în care se poate seta timpul la care să aibă loc evenimentul de OC. În momentul în care valoarea curentă a timer-ului ajunge la această valoare de comparare, evenimentul de OC se activează. Acest eveniment poate să schimbe nivelul logic al unui pin de ieșire *OCn* (HIGH sau LOW) sau poate doar să genereze o întrerupere internă. Funcția de OC are de cele mai multe ori o opțiune de resetare care va reseta automat registrul de numărare al timer-ului atunci când acesta a ajuns la valoarea de comparare. Această opțiune permite implementarea unei *întreruperi periodice de timp real* (sau a unui semnal de ieșire periodic) cu un efort minim.

3.2.8 Generare PWM – Pulse Width Modulation

Funcția de generator PWM este un caz particular al modului de comparare la ieșire. Această funcție permite timer-ului să genereze un semnal de ieșire digital periodic al cărui timp de HIGH (factor de umplere) și perioadă pot fi configurate în program. Există mai mulți regiștrii prin care se pot seta acești parametri precum și tipul semnalului PWM (Fast PWM sau Phase Correct PWM). Aceste semnale PWM pot fi folosite în multe aplicații cum ar fi: construcția unor convertoare digital-analogice simple, implementarea funcției de ABS la mașini, controlul intensității unui LED sau display numeric sau pentru controlul motoarelor (servo-motoare, motoare pas cu pas sau motoare de curent continuu). Generarea de semnale PWM este subiectul unei lucrări următoare.

3.3 Desfășurarea lucrării

Această lucrare de laborator se bazează pe folosirea microcontrolerului ATmega32. Foaia de specificații (datasheet) a acestuia poate fi consultată la adresa (http://ww1.microchip.com/downloads/en/DeviceDoc/doc2503.pdf). Acest microcontroler conține trei timere dintre care doua pe 8 biți si unul pe 16 biți. Din motive de simplitate se va considera în această lucrare doar unul din timer-ele pe 8 biți numit Timer0 (pag. 69, datasheet). Structura bloc simplificată este prezentată în Figura 3.1. În aceasta diagramă, litera *n* va fi înlocuită de 0 întrucât este vorba despre Timer0. Astfel TCNTn va fi interpretat ca TCNT0.

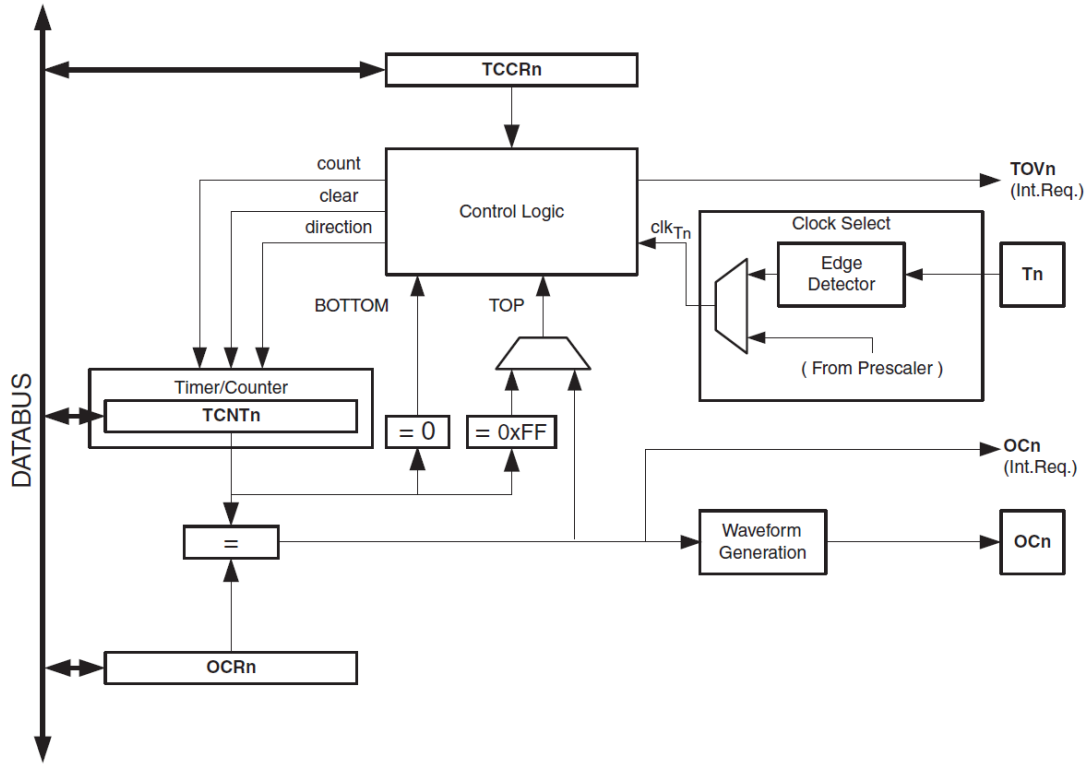


Figura 3.1. Diagrama bloc a unui timer pe 8 biți

În construcția internă a timer-ului se observa existența a 3 regiștrii pe 8 biți: TCCR0, TCNT0 și OCR0. Registrul TCCR0 (Timer/Counter Control Register) este un registru de control care se folosește pentru setarea diferitelor funcționalități ale timer-ului: selectează sursa ceasului (clock), setează valoarea pentru prescaler, setează modul de operare și schimbă funcționalitatea pinului OC0 (PB3) din pin de comparare la ieșire în pin de ieșire de uz general

și invers. Registrul TCNT0 este registrul de numărare al timer-ului. La fiecare front crescător al semnalului de ceas acesta este incrementat sau decrementat conform cu modul de funcționare setat. Registrul OCR0 reprezintă registrul de comparare la ieșire și poate fi scris cu o valoare din intervalul [0; 255]. În modul de funcționare OC, valoarea lui TCNT0 este comparată cu valoarea acestui registru pentru declanșarea unui eveniment ca de exemplu resetarea lui TCNT0, generarea unui semnal extern pe pinul OC0 sau generarea unui semnal de întrerupere.

Bazat pe semnalul de ceas clk_{Tn} și potrivit setărilor din registrul TCCR0, blocul *Control Logic* trimite comandă de incrementare, direcție de numărare și resetare a registrului de numărare TCNT0 și de asemenea activează semnalul de întrerupere pentru *Overflow.*

În această lucrare de laborator se va folosi timer-ul în *regim normal* de numărare (WGM01:0 = 0). Acesta este cel mai simplu mod de operare în care direcția de incrementare a registrului de numărare va fi tot timpul pozitivă. Când numărătorul va trece de valoarea maximă pe 8 biți (255), acesta se va reseta la valoarea minimă (0) și va continua să numere.

În modul CTC (Clear Timer on Compare Match), registrul OCR0 este folosit pentru a manipula rezoluția numărătorului și de asemenea valoarea maximă de numărare. În modul CTC (WGM01:0 = 2) numărătorul este resetat atunci când valoarea sa (TCNT0) este egală cu cea din registrul OCR0. În acest mod de funcționare se poate genera o întrerupere de fiecare dată când are loc egalitatea dintre cei 2 regiștrii.

Pentru a genera un semnal de ieșire în modul CTC, pinul OC0 trebuie setat ca pin de ieșire digitală. Modul de ieșire pentru OC0 poate fi setat ca să basculeze de fiecare dată când se găsește egalitate între TCNT0 și OCR0. Frecvența semnalului este definită de ecuația următoare:

$$f_{OC0} = \frac{f_{clk\_I/O}}{N \cdot (1 + OCR0)}$$

unde $f_{clk\_I/O}$ reprezintă frecvența ceasului intern al sistemului, N este valoarea prescaler-ului (1, 8, 64, 256, 1024) iar ORC0 este valoarea scrisă în acest registru.

Descrierea regiștrilor

TCCR0 – Acest registru permite controlul modului de funcționare a Timer-ului.

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	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	

Biții WGM01:00 permit schimbarea modului de operare între Normal, CTC și PWM.

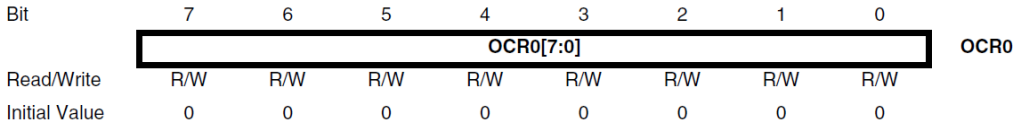
Biții COM01:00 permit schimbarea funcționalității pinului OC0 din pin digital de uz general în pin de OC. Pentru fiecare mod de operare există o altă modalitate de setare.

Biții CS02:00 oferă posibilitatea selectării semnalului de ceas dintre ceasul intern, ceasul intern filtrat prin prescaler (1, 8, 64, 256, 1024) și semnal de ceas de la pinul T0 (PB0).

TCNT0 – Acest registru permite accesul direct atât pentru scrierea cât și pentru citirea numărătorului pe 8 biți al timer-ului.

Bit	7	6	5	4	3	2	1	0	
	TCNT0[7:0]								TCNT0
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	

OCR0 – Registrul de comparare la ieșire conține o valoare pe 8 biți care este comparată mereu cu registrul de numărare (TCNT0).



Exerciții:

1. Implementarea unui ceas digital

Pentru implementarea unui ceas, prima necesitate care apare este aceea de a avea posibilitatea să măsurăm o secundă. Pentru un timer pe 8 biți cea mai mare perioada de timp care poate fi măsurată este egală cu:

$$T_0 = \frac{1}{f_0} = \frac{1}{f_{clk\_I/O}/(N \cdot TOP)}$$

iar pentru o frecvență de ceas de 8MHz, o valoare maxima de numărare de TOP = 2^8și o valoare maximă a prescaler-ului de 1024 va rezulta $T_0 = 32.7\ ms$ ($f_0 = 30.51\ Hz$), ceea ce e departe de 1s. Din cauza acesta vom folosi timer-ul pentru a masura 1ms în timp ce o secundă o vom obține prin numărarea în program a milisecundelor. Astfel pentru a ajunge la 1ms vom folosi un

N = 64 și o valoare maximă de numărare de TOP = 125. Se va folosi modul *Normal* de operare a timer-ului, astfel că se va face o resetare software a numărătorului când acesta ajunge la 125. Se va crea o funcție de inițializare a timer-ului iar în bucla *for* infinită se va implementa programul. Pe display BCD 7-segmente se vor afișa minutele (pe primele 2 poziții stânga) și secundele (pe ultimele 2 poziții dreapta).

Funcția de inițializare:

```
void init_timer(){
  TCCR0 = 0b00000011; //prescaler 64, mod normal de operare
  TCNT0 = 0;          //se inițializează numărătorul cu 0
}
```

Programul:

```
for(;;){
  if(TCNT0>=125){           //se împlinește o milisecundă
       TCNT0 = 0;           //resetare software
       ms++;
       if(ms>=1000){        //împlinirea unei secunde
            ms=0;
            s++;
       }
       if(s==60){
            s=0;
            m++;
       }
  }
  display(1,s%10);
  display(2,(s/10)%10);
  display(3,(m%10));
  display(4,(m/10)%10);
}
```

2. **Modificați exercițiul 1 pentru a folosi Timer2 și un prescaler N = 64. Verificați setarea regiștrilor.**

3. **Implementați un cronometru cu secunde și sutimi de secunde. Se vor folosi 2 butoane: unul pentru *start/stop* și unul pentru *reset*.**

4. Sistemul de Întreruperi

4.1 Scopul lucrării

Lucrarea are ca scop înțelegerea mecanismului de întreruperi prezent la cele mai multe microcontrolere precum și prezentarea câtorva aplicații în care se folosesc întreruperile.

4.2 Ce sunt întreruperile?

Cel mai adesea microcontrolerele sunt folosite în sisteme care trebuie să reacționeze la evenimente. Evenimentele pot fi constituite de schimbarea unor stări ale sistemului și în general declanșează o reacție din partea microcontrolerului, cu un timp de reacție (întârziere) minim. Aceste evenimente trebuie privite ca informații de stare a sistemului, informații ce rezultă fie ca urmare a funcționării interne a microcontrolerului, (evenimente ce generează așa numitele întreruperi interne), fie ca urmare a interacțiunii sistemului cu mediul înconjurător (generând întreruperi externe).

Considerând că microcontrolerul are posibilitatea de a observa aceste evenimente, se pune întrebarea: care este modul acestuia de a monitoriza aceste evenimente. Este posibil ca să se verifice periodic în program dacă au avut loc schimbări în stările sistemului. Sunt mai multe neajunsuri ce privesc această metodă. Pentru a nu pierde evenimente este necesar ca verificarea să se facă de mai multe ori și cu o anumită perioadă de timp maximă pe parcursul rulării programului. Aceasta duce la o încărcare nejustificată a procesorului mai ales în cazurile în care aceste evenimente au loc doar sporadic. Bineînțeles, procesorul este destinat altor activități decât verificarea periodică a schimbării unei stări.

O a doua metoda de a rezolva problema monitorizării este folosirea *sistemului de întreruperi*. Microcontrolerele au un astfel de sistem implementat hardware care se ocupă de detecția evenimentelor din sistem. Rularea programul se face normal până la momentul apariției unui eveniment, moment în care programul este întrerupt și o nouă funcție este rulată. Această funcție se numește rutină de întrerupere (ISR - Interrupt Service Routine).

Trebuie menționat faptul că, în sistemul de întrerupere al microcontrolerului fiecare tip de întrerupere, în funcție de sursa care o generează, are un anumit nivel de prioritate. Acest nivel de priorotate face ca un tip de întrerupere să fie mai importantă decât un altul. Acest aspect este discutat ulterior.

4.3 Controlul întreruperilor

Fiecare întrerupere poate fi configurată prin intermediul a doi biți. Un bit de *Interrupt Enable* (IE) care este setat de aplicație pentru a activa sau dezactiva întreruperea și indică faptul că o funcție ISR trebuie apelată ca reacție la un eveniment. Celălalt bit, numit *Interrupt Flag* (IF) este setat în momentul în care are loc un eveniment și resetat fie automat după ce se apelează ISR-ul fie manual de către programator. Poate exista cate un bit de IE pentru fiecare tip de întreruperi sau pot exista mai multe întreruperi activate de un singur bit de IE, diferențierea făcându-se prin IF (fiecare întrerupere cu flag-ul propriu). Asemănător bit-ului de IE, există la fiecare microcontroller cate un bit care activează sau dezactivează întreg sistemul de întreruperi (*Global Interrupt Enable bit*).

Așa cum se poate observa în Figura 4.1, sursele care generează întreruperile sunt în partea stângă a imaginii și pot fi atât pini fizici ai micocontrolerului cât și componente precum timere, convertoare analog-digitale, module de comunicare serială sau altele. Pentru ca o întrerupere să fie luată în considerare este nevoie ca bitul de IE individual (încercuit cu roșu) să fie activat precum și bitul de IE Global.

Pe lângă biți IE și IF, în general, microcontrolerele oferă și alți biți de control care sunt folosiți pentru a selecta tipul de stare care generează întreruperea (doar frontul crescător, doar frontul descrescător, orice front, ...).

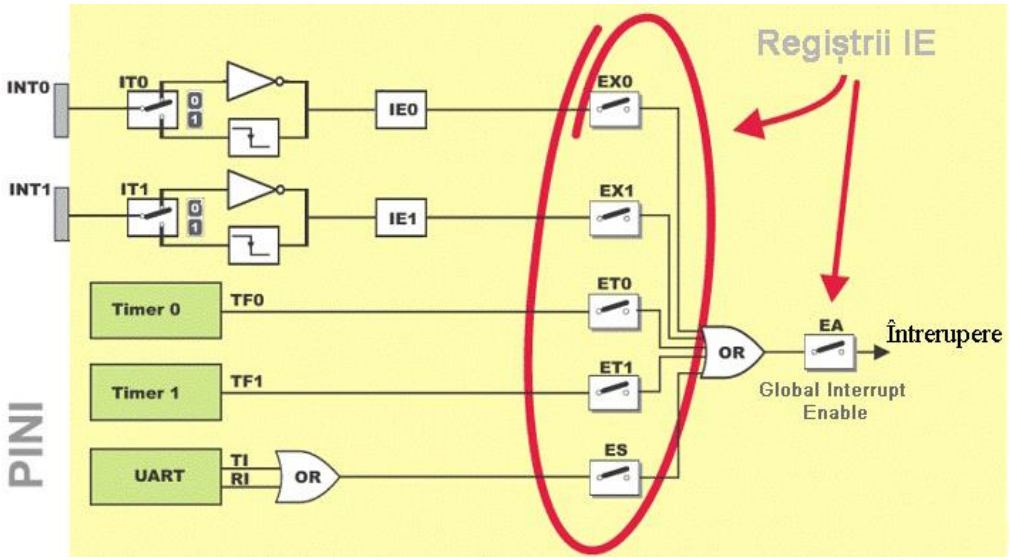


Figura 4.1. Surse de întreruperi și modul de activare a întreruperilor cu IE

Dezactivarea întreruperilor, pentru o perioadă scurtă de timp, folosind bitul de *Global Interrupt Enable* nu implică automat pierderea evenimentelor. Producerea unui eveniment se stochează în bitul IF corespunzător indiferent daca IE este setat sau nu. Astfel că, dacă un eveniment are loc în timp ce întreruperile sunt dezactivate, funcția ISR corespunzătoare se va apela în momentul în care sistemul de întreruperi se activează din nou. Astfel, un eveniment se poate pierde doar în momentul în care un al doilea eveniment are loc înainte de a se răspunde primului eveniment printr-o rutina ISR. Spre deosebire de întreruperile tipice, există așa numitele *întreruperi nemascabile* (*NMI – Non-Maskable Interrupts*) care nu pot fi dezactivate de bitul IE global. Aceste întreruperi sunt folositele în cazul evenimentelor care nu pot fi întârziate de alte cauze și necesită un răspuns prompt chiar cu riscul de a afecta mersul programului. După o resetare a microcontrolerului, în general, întreruperile sunt dezactivate atât la nivel global cât și fiecare în parte.

4.4 Organizarea întreruperilor

Întreruperile sunt împărțite în două mari categorii:

- Interne: provenite de la componente precum timere, memoria EEPROM, Convertoare Analog-Numerice, etc. Întreruperile interne nu sunt afectate de schimbările din mediul în care microcontroller-ul lucrează (ex. apăsarea unui buton).

- Externe: provenite de la pinii de intrare/ieșire digitali (Digital I/O). Întreruperile externe sunt evenimente care depind de mediul în care se afla microcontroler-ul. Spre exemplu

apăsarea unui buton conectat pe o intrare ce suportă întreruperi reprezintă o schimbare în mediul extern al microcontroller-ului.

Fiecare dintre întreruperile interne și externe sunt asociate cu o rutină de întrerupere (ISR) ce va fi apelată la apariția evenimentului declanșator. Această mapare a întreruperilor cu funcțiile ISR se realizează pe baza unui *tabel de întreruperi* (interrupt vector table) care conține câte o linie pentru fiecare tip de întrerupere. Un astfel de vector asociază fiecărei întreruperi un număr (care determină prioritatea) și o adresă fixă din memoria de program. Adresa din memoria de program reprezintă adresa de la care începe prima instrucțiune a funcției ISR. Atunci când condiția de întrerupere este îndeplinită, programul normal va produce un salt (*jump*) la adresa din vectorul de întreruperi. Un astfel de tabel este reprezentat în Figura 4.3.

Vector Number	Program Address	Source	Interrupt Definition
1	$000	RESET	External Pin, Power-on Reset, ...
2	$004	INT0	External Interrupt Request 0
3	$008	INT1	External Interrupt Request 1
4	$00C	TIMER2 COMP	Timer/Counter 2 Compare Match
...	...	...	...

Figura 4.3. Tabelul de întreruperi al microcontrolerului ATmega32

4.5 Prioritățile întreruperilor

Există situații în care în același timp pot să apară două evenimente care generează întrerupere. Aceste situații sunt tratate cu ajutorul priorităților. Determinarea ordinii de servire, sau analiza priorității cererilor de întrerupere, presupune scanarea în faza de analiză a tuturor canalelor de transfer pentru cererile de întrerupere sau a regiştrilor corelaţi cu acestea într-o anumită ordine prestabilită în hardware. Multe microcontrolere dispun de mecanismul (implementat hard) ce permite "prioritizarea" unora dintre aceste canale, ceea ce implică implementarea unor unități de memorare a priorităților (regiştri de setare a priorităților pe două sau mai multe nivele) și instituirea unor reguli de arbitrare. Fiecare dintre aceste nivele va fi scanat distinct, în ordinea descrescătoare a priorităților Figura 4.2.

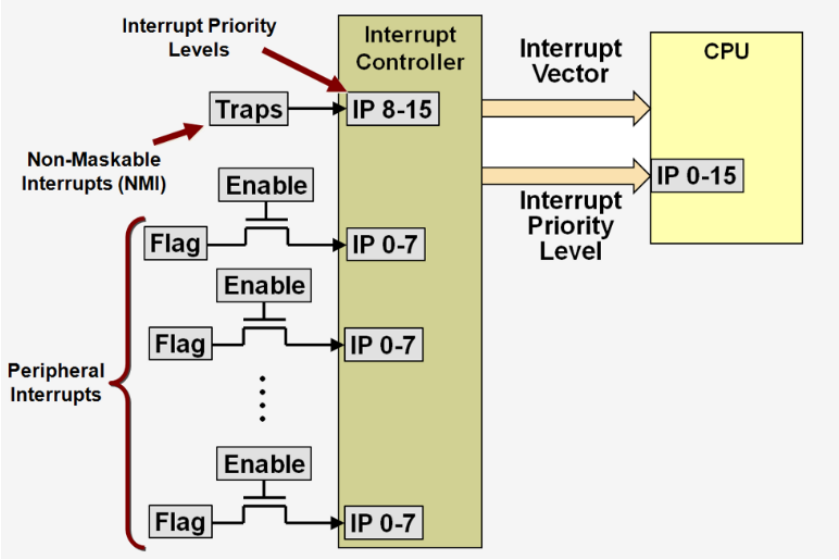


Figura 4.2. Controlerul de întreruperi, priorități

Majoritatea microcontrolerelor cu multe întreruperi ce conțin și tabel de întreruperi, folosesc vectorul de întreruperi ca și indicator de prioritate. De exemplu, ATmega32 asociază prioritatea cea mai mare, întreruperii cu cel mai mic număr în tabelul de întreruperi.

Prioritizarea întreruperilor nu e utilă doar pentru cazul în care apar mai multe întreruperi în același timp, ci și pentru a determina dacă o funcție ISR poate fi întreruptă la rândul ei. Într-un astfel de sistem, dacă este activată o astfel de opțiune, o întrerupere cu un nivel mare al priorității va întrerupe ISR-ul unei întreruperi cu nivel mai scăzut al priorității. În cazul microcontrolerului ATmega32 orice întrerupere poate să întrerupă un ISR atâta timp cât are un nivel al priorității mai mare și bitul IE este setat. Dacă o astfel de întrerupere nu e întotdeauna dorită sau permisă în aplicație, la rularea funcției ISR se vor dezactiva întreruperile din IE global iar la încheierea ISR-ului se vor activa din nou.

În cele ce urmează se prezintă pașii de la apariția unui eveniment până la reacția sistemului la acest eveniment. Pentru început un eveniment trebuie detectat. Luăm spre exemplu o eveniment extern pe un pin I/O digital. La fiecare ciclu se verifică nivelul logic de pe pin. În cazul în care se detectează o condiție de întrerupere se setează flag-ul asociat întreruperii (IF). Urmează ca logica de întrerupere sa verifice daca bitul IE corespunzător întreruperii este setat (interrupt enabled) și nu există în același moment alta întrerupere cu o prioritate mai mare. Daca se îndeplinesc aceste condiții, se apelează ISR-ul asociat întreruperii. Această logică se aplică și întreruperilor interne, de exemplu un eveniment asociat unui timer.

4.6 Librăria `<avr/interrupt.h>`

Mediul AVR-GCC oferă un număr bogat de librării dedicate programării microcontrolerelor Atmel AVR. Între aceste librării se găsește si librăria ***<avr/interrupt.h>.*** În arhitectura AVR, vectorul de întreruperi este predefinit pentru a trimite către rutinele de întreruperi ale căror nume sunt predeterminate. Prin folosirea unui nume corect, rutina ISR se va apela automat la apariția întreruperii corespunzătoare. În această librărie sunt definite anumite rutine care pot fi folosite în contextul întreruperilor. Descrierea amănunțită se găsește la adresa: http://www.nongnu.org/avr-libc/user-manual/group__avr__interrupts.html.

Câteva din aceste funcții sunt descrise mai jos:

- **sei() – set enable interrupts:** Activează întreruperile la nivel global.
- **cli() – clear interrupts:** Dezactivează întreruperile la nivel global.
- **ISR(vector) - interrupt service routine:** Definește o rutină de întrerupere pentru o întrerupere asociată

  vector – acest parametru reprezintă numele unui vector de întrerupere.

  Ex: TIMER0_COMP_vect – vectorul întreruperii de comparare la ieșire (Timer0).

  INT0_vect, INT1_vect - vectorul întreruperilor externe INT0 și INT1.

  ADC_vect - vectorul întreruperii de terminare a conversiei analog-numerice.

În locul folosirii celor două funcții sei() și cli() se poate folosi registrul SREG în care bitul 7 setează sau resetează întreruperile globale.

Bit	7	6	5	4	3	2	1	0	
	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	

- **Bit 7 – I: Global Interrupt Enable**

4.7 Desfășurarea lucrării

Pentru implementarea exercițiilor se va folosi mediul de programare mikroC PRO și va trebui consultată foaia de specificații ATmega32

(http://ww1.microchip.com/downloads/en/DeviceDoc/doc2503.pdf).

Exerciții:

1. **Folosirea întreruperii de Comparare la Ieșire (Output Compare) - Implementarea unui ceas digital**

Acest exercițiu este o continuare a implementării ceasului digital din laboratorul anterior și se vor folosi semnalele de întrerupere pentru a semnaliza procesorul despre scurgerea unei perioade de timp de 1 ms.

Din biții 3 și 6 (WGM01:0) ai registrului TCCR0 se va alege modul de funcționare al timer-ului ca CTC (Clear Timer on Compare Match) ceea ce va determina resetarea registrului TCNT0 la găsirea unei egalități între TCNT0 și OCR0. În registrul OCR0 se va copia valoarea la care să se reseteze timer-ul, în cazul aplicației curente aceasta va fi 125. În același registru TCCR0 se va seta un prescaler de 64 folosind biții 0, 1 și 2 (CS02:0).

Registrul TIMSK este folosit pentru setarea biților IE pentru întreruperile asociate timer-ului. Vrem să generăm o întrerupere de fiecare dată când are loc un eveniment de comparare la ieșire (TCNT0 = OCR0), pentru aceasta vom seta bit-ul 1 din registrul TIMSK. Totodată trebuie activate și întreruperile la nivel global prin setarea bitului 7 din Registrul SREG.

```
void init_timer(){
      SREG = 1<<7;                // Global Interrupt Enable
      TCCR0 = 0b00001011;         //CTC-3,6; Prescaler-0,1,2
      TCNT0 = 0;
      OCR0 = 125;
      TIMSK |= 0b00000010;        //set interrupt OCM
}
```

Funcția init_timer() va fi apelată în funcția *main* și va conține toate setările menționate mai sus. În următoarea etapă se va defini rutina de întrerupere (ISR – Interrupt Service Routine). În MikroC Pro acest lucru se poate face cu ajutorul Interrupt Assistant-ului care apare la scrierea sintaxei ***iv*** pe linia de cod unde se dorește definirea rutinei (Figura 4.4). Programatorul introduce un nume de funcție reprezentativ în primul câmp iar din al doilea se va alege numele vectorului de întrerupere dorit. Mai jos este un model de astfel de rutina de întrerupere.

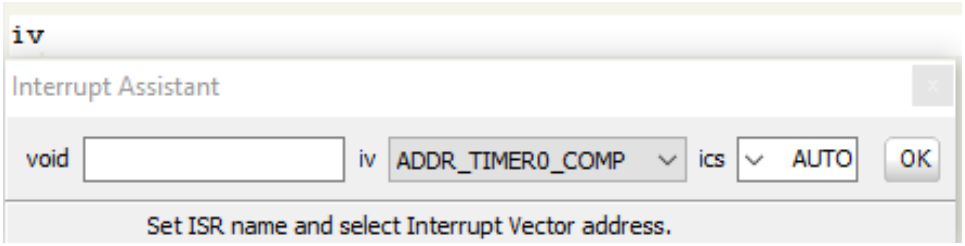


Figura 4.4. Fereastra pop-up pentru Interrupt Assistant

Structura *switch* este folosită pentru ca la fiecare milisecundă (moment în care se apelează automat funcția ISR) să se afișeze un nou caracter pe o altă poziție a display-ului BCD 7 Segmente. Astfel când variabila *n* ajunge la 4 se va reseta la 0 și va reîncepe ciclul. După această structură urmează codul care se dorește a fi executat. În cazul de față se va incrementa valoarea milisecundelor iar la 1000 se va incrementa valoarea secundelor.

```
void Timer1_OC_ISR() iv IVT_ADDR_TIMER0_COMP {        //ISR
        digit++;
        switch(digit){
                case 1: display(1,s%10); break;
                case 2: display(2,(s/10)%10); break;
                case 3: display(3,m%10); break;
                case 4: display(4,(m/10)%10); digit=0; break;
        }
        if (ms == 999){
                s++;
                ms = 0;
                if(s == 60){
                        s = 0;
                        m++;
                }
        }else ms++;
}
```

Funcția *main* va cuprinde doar instrucțiunile de setare a pinilor I/O digitali, apelul funcției de inițializare și structura în bucla infinită. În acest gen de aplicație întreaga activitate a procesorului este bazată pe întreruperi, întreruperi care determină execuția funcțiilor ISR. Funcția *display* va fi definită în program conform modelului din laboratoarele trecute.

```
void main(){
DDRA = 0b00001111;  //setează pinii de ieșire
DDRC = 0b11111111;
init_timer();
        for(;;);
}
```

2. Modificați exercițiul 1 pentru a folosi Timer2 și un prescaler N = 32. Verificați setarea regiștrilor.

3. Exemplu de setare a unei întreruperi externe.

De precizat faptul că următoarea setare se va face în mediul *mikroC PRO for AVR*, care este puțin diferit față de alte medii. Se va lua ca exemplu întreruperea externa INT0 care corespunde pinului PD2 și se va activa pe frontul crescător al acestuia.

Așa cum a fost menționat, rutina ISR se va declara alături de celelalte funcții folosind Interrupt Assistant-ul și va conține următoarele instrucțiuni:

```
void nume_functie() iv IVT_ADDR_INT0 {          // Interrupt rutine
    //
    //Secventa de cod//
    //
}
```

Nume_funcție poate lua orice nume, iar adresarea acestuia către întreruperea INT0 se face cu "org IVT_ADDR_INT0".

O nouă funcție pentru inițializarea întreruperii externe INT0 va cuprinde următoarele instrucțiuni:

```
void Init_INT0(){
GICR  |= (1<<6);     // activarea întreruperii externe INT0
MCUCR |= 0b00000010; // precizare că evenimentul va avea loc pe frontul
                     // descrescător
SREG  |= (1<<7);     // activarea globala a întreruperilor
}
```

OBS! Funcția *nume_funcție* nu trebuie apelată în cod, deoarece aceasta se apelează automat la întâlnirea evenimentului de întrerupere, în cazul de față apăsarea butonului corespunzător pinului PD2.

4. Folosiți întreruperile INT0 și INT1 pentru implementarea a 2 butoane pentru incrementarea minutelor și secundelor. Se va folosi întreruperea pe frontul crescător.

5. Convertorul Analog-Digital

5.1 Scopul lucrării.

Această lucrare prezintă principiul de funcționare a unui *modul de conversie analog-digitală* și de asemenea și structura generală împreună cu descrierea regiștrilor asociați convertorului. Aplicațiile prezentate au în vedere modul de configurarea a modulului de conversie pentru citirea unei valori analogice provenite de la un potențiometru și un senzor de temperatura. Se va explora și folosirea întreruperilor asociate convertorului și modul de autodeclanșare a conversiei A/D pe un microcontroler ATmega32.

5.2 Prezentarea modulelor de conversie A/D

Intrările și ieșirile digitale care au fost prezentate în laboratorul precedent, au, așa cum sugerează și denumirea de digital, doar 2 valori posibile - valorile 0 pentru starea LOW, respectiv 1 pentru starea HIGH. Există însă situații în care este necesar să se lucreze cu mărimi analogice și nu digitale, caz în care este nevoie de valoarea unor mărimi care se modifică în mod continuu între două limite. Aceste mărimi sunt transformate de obicei într-un semnal de tensiune, ca de exemplu folosirea unui fototranzistor ca senzor de lumină. În acest caz, ieșirea unui circuit care include senzorul este direct proporțională cu fluxul de lumină care ajunge la fototranzistor, astfel, pentru a determina corect valoarea de ieșire a senzorului este nevoie ca microcontrolerul să lucreze cu un semnal analogic (tensiunea de ieșire). Fiind un circuit digital, microcontroler-ul trebuie să convertească acest semnal analogic într-un format digital. Această conversie este realizată de *convertorul analog-digital (A/D)* integrat în microcontroler.

5.2.1 Conversia analog-digitală

Prezența modulelor A/D și D/A în structura unui MC este de o importanță majoră pentru utilitatea acestuia în aplicații deoarece interfațarea cu mediul presupune necesitatea de a prelucra sau de a elabora mărimi analogice. De exemplu, dacă dorim să folosim senzorul de lumina pentru a determina și afișa valoarea curentă a luminozității dintr-o încăpere trebuie să reprezentăm valoarea analogică a tensiunii de la ieșirea senzorului în formă digitală. În acest scop, microcontrolere au inclus un modul de conversie analog-digitală (***ADC - Analog to Digital Converter***), modul care convertește un semnal de intrare analogic într-o formă digitală. Convertoarele A/D integrate pe *chip* sunt convertoare cu aproximații succesive. Convertoarele integrate în microcontrolere sunt convertoare relativ lente în comparație cu cele implementate în circuite independente.

Figura 5.1 prezintă principiul de funcționare a unui convertor A/D cu o rezoluție pe 3 biți. Semnalul analogic cuprins între 0V și tensiunea de referință V_{ref} este împărțit în 2^r intervale, unde r reprezintă numărul de biți pe care se va reprezenta valoarea digitală. În literatură *r* poarta numele de rezoluție a convertorului. Cele mai des folosite valori pentru *r* sunt 8, 10 ,12 iar pentru cele mai bune convertoare chiar 16 biți iar precizia este de ±1/2LSB (***L**east **S**ignificant **B**it*). Cel mai nesemnificativ bit al rezultatului digital reprezintă cea mai mică diferență de tensiune ($V_{Ref}/2^r$) care poate fi detectată de convertor. Pentru a determina această valoare trebuie sa cunoaștem tensiunea de referință și numărul de biți pe care se reprezintă rezultatul digital. În literatură, aceasta valoare este întâlnita sub denumirea de rezoluție a ADC-ului.

Exemplu: Pentru o tensiune de referință de 5V și un convertor analog numeric pe 10 biți, valoarea minima detectabilă este de $Vref/2^{10}$ = 4.88mV.

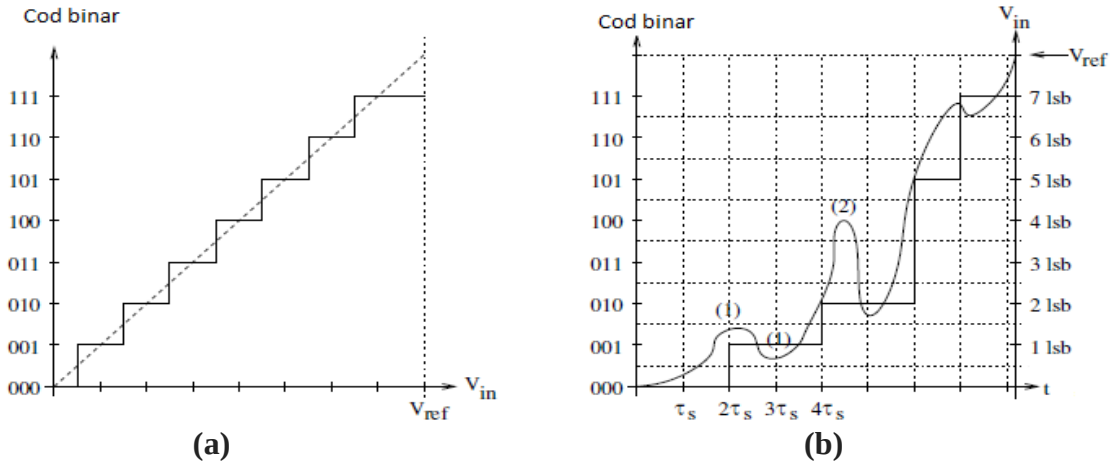


Figura 5.1. Principiu de funcționare ADC 3 biți.

Exemplul din Figura 5.1(b) ilustrează problemele care pot apărea odată cu conversia semnalului analogic în format digital. Cele doua valori ale semnalului de la punctele (1) sunt asociate împreună valorii digitale 001. Pentru a depăși aceste probleme și pentru a crește rezoluția convertorului există două soluții. Fie folosim un convertor cu o rezoluție mai mare (o valoare mai mare pentru r) fie folosim o tensiune de referință mai mică și astfel rezoluția va fi mai bună dar aceasta va scădea domeniul de măsurare. Timpul de conversie reprezintă timpul de la începutul unei conversii până în momentul în care obținem rezultatul conversiei la finalizarea acesteia. Astfel pe durata unei conversii fluctuațiile semnalului sunt pierdute, exemplu punctul (2) din Figura5.1 (b) Pentru rezolvarea acestei probleme se poate crește frecvența la care lucrează convertorul.

În continuare este prezentat modul de operare al convertorului care funcționează pe baza aproximărilor succesive. Figura5.2 prezintă schema de principiu a convertorului cu aproximări succesive iar Figura 5.3 modul de funcționare a acestuia.

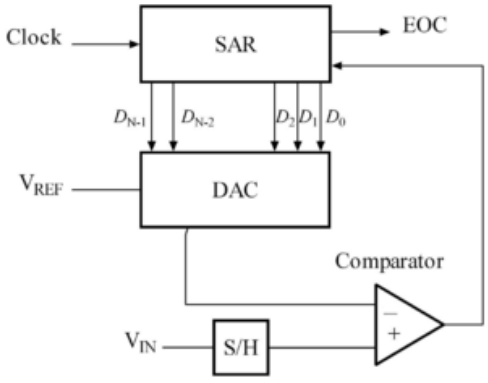


Figura 5.2. Schema de principiu a unui ADC cuaproximări succesive

În Figura 5.2: VIN reprezintă semnalul măsurat;

Vref reprezintă tensiunea de referință;

DAC este un convertor numeric-analogic;

SAR registrul de aproximări succesive;

S/H este un circuit de eșantionare și memorare a tensiuni ce se măsoară în acel moment

În continuare se va prezenta modul de funcționare a unui convertor SAR. La începutul unei conversii, circuitul de menținere va fi încărcat cu valoarea tensiunii ce urmează să fie convertită. Această valoare va fi menținută pe întreaga durată a conversiei. Apoi, registrul SAR (Succesive Approximation Register) este inițializat cu MSB = 1 (*Most Significant Bit*) în cazul de față este vorba de un convertor de 10 biți astfel, SAR = 1000000000. Această valoare (jumătate din valoarea maximă) este aplicată mai apoi convertorului digital-analogic rezultând o tensiune Vref/2. Această tensiune este mai apoi comparată cu tensiunea din circuitul de eșantionare și memorare. Dacă tensiunea de la convertorul digital-analogic este mai mică decât tensiunea de la ieșirea circuitului de eșantionare și memorare bit-ul MSB din registrul SAR rămâne la valoarea 1, în caz contrar acest bit va fi resetat și devine 0. Apoi se trece la următorul bit din SAR, se setează 1 și se generează tensiunea corespunzătoare prin DAC. Se repeta acest test pentru fiecare bit din SAR începând de la MSB spre LSB (*Least Significant Bit*). La finalul conversiei registrul SAR va conține valoarea digitală a semnalului.Figura5.3 prezintă un exemplu de conversie pe 3 biți.

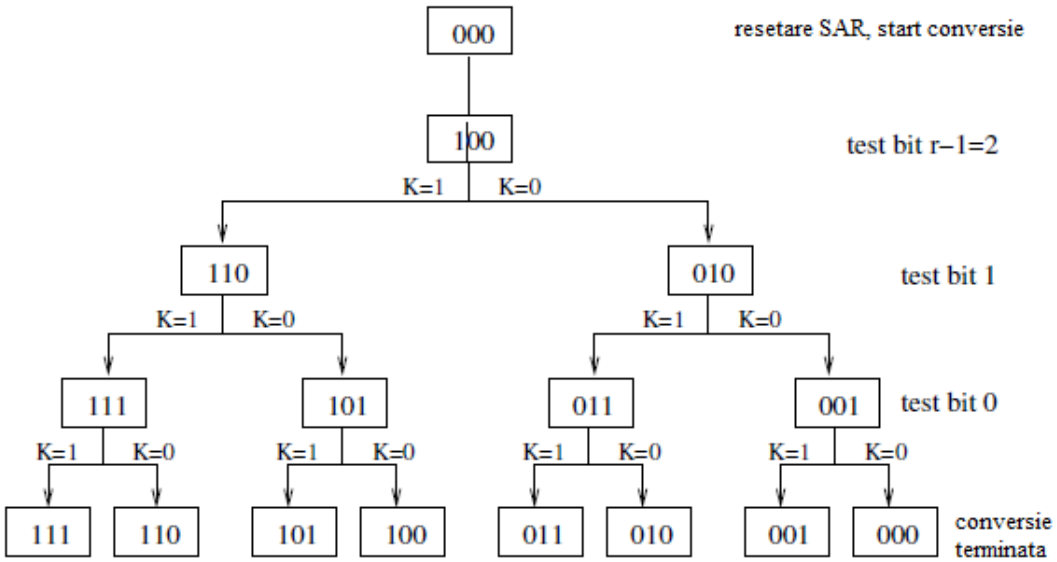


Figura5.3. Conversie SAR 3 biți.

În Figura5.4 este prezentată o schemă bloc simplă a unui modul de conversie A/D. Circuitul analogic de intrare constă într-un multiplexor analogic, un circuit de eșantionare/memorare și un convertor A/D cu aproximații succesive. Tensiunea de referință pentru convertor este furnizată din exterior la pini specializați. Semnalul de ceas necesar convertorului este generat intern folosind semnalul de ceas al unității centrale de procesare.

Modulul A/D folosește un registru de control prin care se selectează canalul de intrareși modul de lucru pentru circuitul de eșantionare/memorare. Declanșarea și terminarea conversiei sunt semnalizate cu ajutorul unui bit din același registru de control. Rezultatul conversiei este stocat în registrul de date. Registrul de date va conține întotdeauna rezultatul ultimei conversii, de aceea acest registru trebuie citit înainte de terminarea următoarei conversii, în caz contrar se pierde informația.

Modulul de conversie este prevăzut și cu un multiplexor analogic, astfel sunt disponibile mai multe canale de intrare.Microcontroler-ul Atmega32 are 8 canale de intrare, rezoluția ADC-ului este de 10 biți, astfel rezultatul conversiei va reprezenta o valoare cuprinsă în intervalul [0, 1023]. Intervalul de tensiune este cuprins între 0V și Vcc. Valoarea minimă

detectabilă este: Vref/2^{10} , 4.8mV. Rezultatul conversiei poate fi calculat cu următoarea formulă, unde V_{in} este valoarea analogica ce va fi convertită:

$$ADC_{value} = (Vin \cdot 1024)/Vref. \quad (1)$$

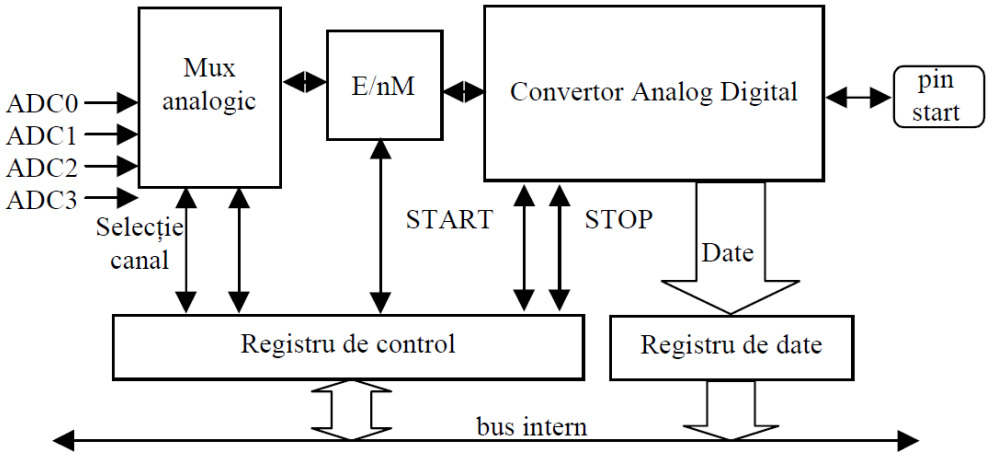


Figura5.4. Schema bloc a modulului de conversie A/D

Vom folosi ca tensiune de referință valoarea de 5V astfel pentru 0V vom obține valoarea 0 iar pentru 5V vom obține valoarea 1023. Pentru a realiza o conversie, un pas îl reprezintă setarea prescaler-ului. Convertorul A/D al microcontroler-ului lucrează pe baza unui semnal de ceas, frecvența acestuia fiind determinată de frecvența de ceas a microcontroler-ului. Frecvențele la care lucrează în general convertorul SAR sunt cuprinse între 50kHz și 200 kHz, însă frecvența microcontroler-ului este mult mai mare. Pentru a obține aceste valori este necesară o divizare a frecvenței semnalului de ceas. În cazul microcontroler-ului Atmega32 sunt predefiniți 8 factori de divizare (prescaler) 2, 4, 8, 16, 32, 64 și 128. Spre exemplu, frecvența microcontroler-ului este de 8 MHz și alegem un divizor de 64, rezultă o frecvență de conversie de 125kHz (8Mhz/64). Există cazuri în care este nevoie de o frecvență de eșantionare mult mai mare, însă cu cât această frecvență este mai mare cu atât acuratețea conversiei scade. O frecvență de conversie cuprinsă între 50kHz - 200kHz oferă rezultate corecte. Valoarea factorului de divizare va trebui aleasă în funcție de frecvența semnalului de ceas astfel încât frecvența convertorului sa fie în domeniul [50, 200] kHz.

5.3 Desfășurarea lucrării:

Schema bloc de prezentare a convertorului A/D conținut în microcontrolerul ATmega32 este prezentată în Figura5.5. Pinii digitali I/O din PORTA au ca funcție alternativă aceea de intrări analogice. Astfel, există 8 canale de intrare analogică care pot fi folosite secvențial prin intermediul unui multiplexor. Semnalele de selecție a canalului se setează din registrul ADMUX. Din același registru se selectează și valoarea de referință iar cea mai comună alegere a acesteia este valoarea pin-ului AVCC (5V). Pinul AVCC este conectat la tensiunea de alimentare a microcontrolerului printr-un filtru trece-jos ceea ce duce la stabilizarea tensiunii. Canalele folosite în aplicațiile de mai jos sunt ADC7 pentru senzorul de temperatura LM35 și ADC5 și ADC6 pentru potențiometru integrat pe placa EasyAVR v7. Urmează în continuare o descriere a regiștrilor modulului de conversie A/D.

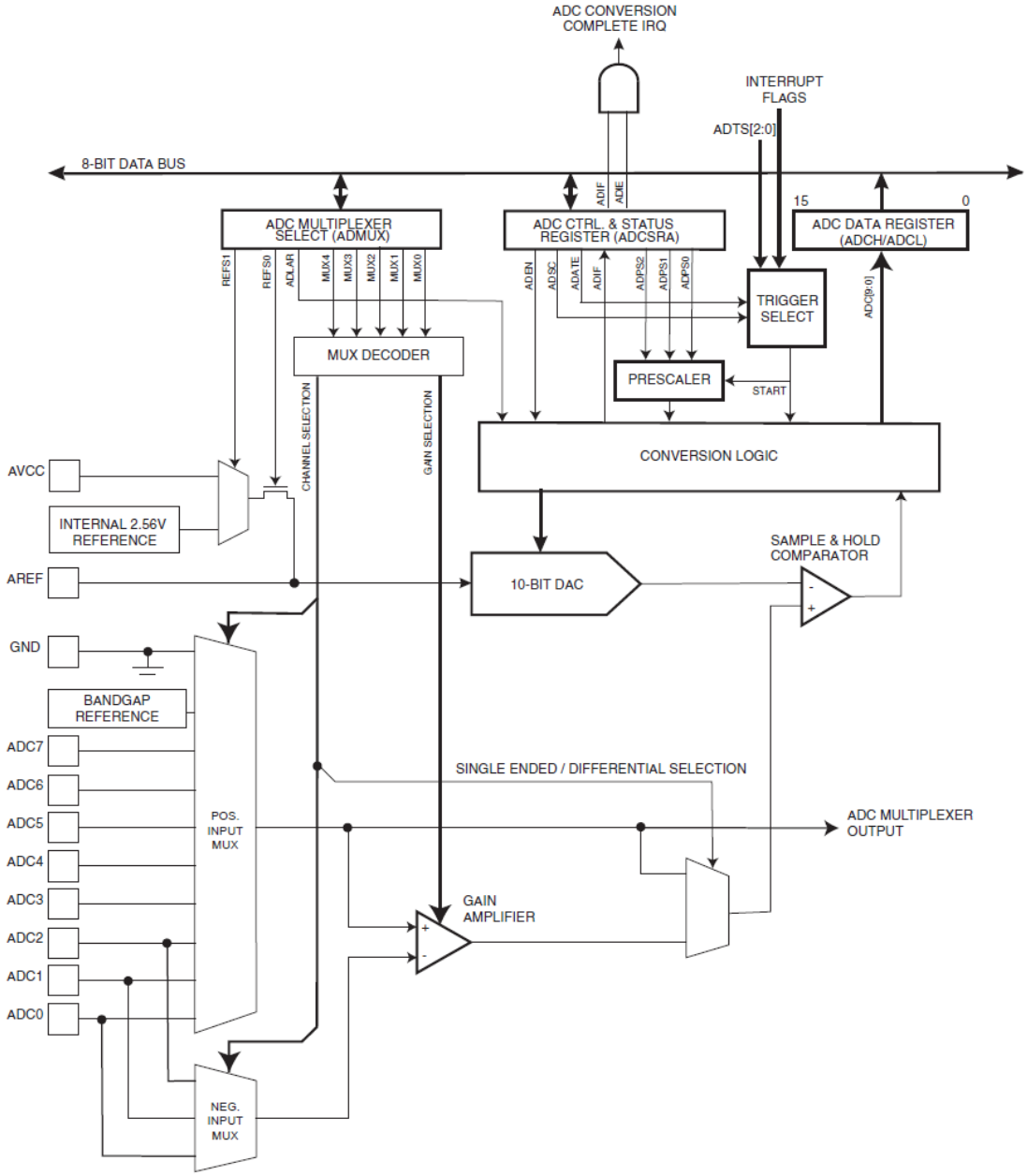


Figura5.5. Schema Bloc a Convertorului A/D

Descrierea regiștrilor

Descrierea completă a regiștrilor se găsește în foaia de specificații a microcontrolerului [ATmega32].

ADMUX – Din acest registru ne alegem tensiunea de referință și alegem canalul de pe care să se realizeze conversia. Tensiunea de referință se setează prin biții 6 și 7 (REFS1:0). Canalul de conversie se alege prin biții MUX4:0. Pentru aplicațiile de laborator se vor selecta doar opțiunile pentru conversie de pe un singur canal (*single ended input*).

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
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	

Canale conversie	Cod MUX4:0
ADC0	00000
...	...
ADC7	00111

ADCSRA – Acest registru este un registru de control al convertorului și din acest registru activăm (enable), modulul de ADC prin setarea bitului 7 ADEN. Pentru a începe o conversie este necesară setarea bitului 6 ADSC înainte de fiecare conversie, **acest bit rămâne 1 logic pe toată durata conversiei, la sfârșitul conversiei acest bit devine 0 logic.** Astfel, se poate verifica dacă s-a terminat o conversie. După terminarea unei conversii în registrul ADC avem un rezultat valid. Setarea prescaler-ului care va determina viteza de conversie se setează din acest registru prin biții ADPS2:0.

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
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	

ADC – Acest registru conține rezultatul conversiei. Se folosesc 2 regiștrii pe 8 biți, **ADCL**și **ADCH** (ADCH – HIGH byte și ADCL – LOW byte), pentru a stoca rezultatul conversie pe 10 biți. Ambii regiștrii pot fi accesați împreună prin numele **ADC**.

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	ADC9	ADC8	ADCH
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Totuși, MikroC PRO nu acceptă apelul registrului ADC pe 16 biți. Din această cauză, rezultatul va trebui obținut prin **ADCL**și **ADCH**, regiștrii pe 8 biți. Pentru obținerea rezultatului final prin compunerea valorilor pe 8 biți se folosesc instrucțiunile următoare:

```
adc_l = ADCL;
adc_h = ADCH;
adc  = (adc_h << 8) | adc_l;
```

Pe placa de dezvoltare EasyAVR v7 există două tipuri de dispozitive analogice integrate care pot fi conectate pe pinii analogici din PORTA. Este vorba despre două potențiometre analogice *P1*și *P3* și un senzor de temperatură analogic *LM35*.

Cele două potențiometre apar pe placa de dezvoltare ca în Figura5.6 dreapta. Se observa că există o multitudine de variante de conectare a celor două potențiometre cu liniile de. Se folosește jumper-ul J3 și J4 pentru conectarea potențiometrelor la liniile de intrări analogice ale microcontrolerului Atmega32. Totuși, pentru scopul acestei lucrări sunt favorabile doar două variante de conexiune. Conectarea potențiometrului P3 la pinii analogici PA5 (ADC5) sau PA6 (ADC6). Modificând de capul potențiometrului vom genera tensiuni în domeniul GND (0V), VCC (5V).

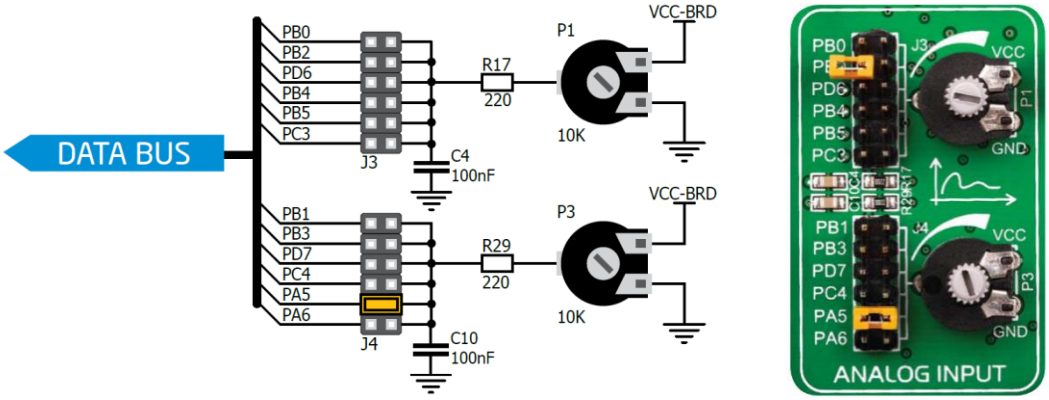


Figura5.6. Schema a potențiometrelor P1 și P3

Senzorul de temperatura LM35 este cea de-a doua componentă analogica pe care o întâlnim pe placa de dezvoltare. Acesta folosește o încapsulare de tip TO-92, capsula fiind bombată pe o parte și aplatizată pe cealaltă (Figura5.8). Se va avea deosebită atenție la modul conectării senzorului cu placa. Asigurați-vă că partea bombată a capsulei este aliniată pe direcția semicercului alb imprimat pe placă ca în Figura5.7. În caz contrar, la conectarea inversă a senzorului acesta se va arde și nuva mai fi funcțional. Legătura dintre senzor și microcontroler se face prin jumper-ul J19. Acesta conectează senzorul la pinul PA7 (ADC7) sau la PB4. Pentru cazul de față, prima variantă este favorabilă întrucât pinul PA7 este pin de intrare analogic. Pe perioada folosirii senzorului analogic, asigurați-vă că nu sunt conectate și alte dispozitive pe pinul PA7.

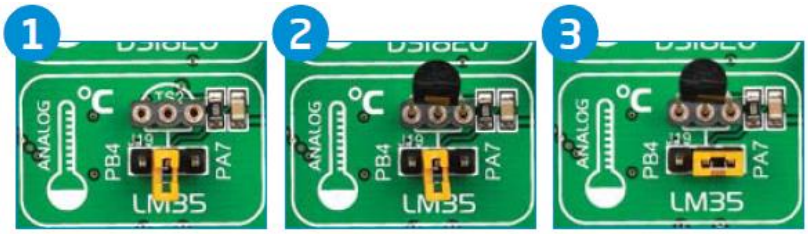


Figura 5.7. Modul de conectare a senzorului LM35

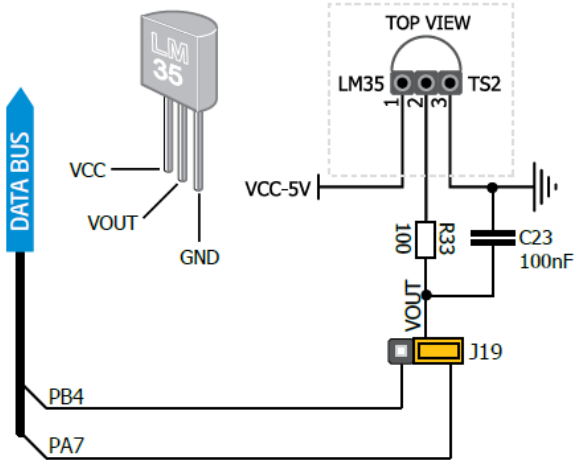


Figura5.8. Senzorul de temperatură LM35

Exerciții:

1. Realizarea unui program de citire a unui valori analogice.

Această aplicație are ca scop explorarea modului de folosire a convertorului A/D pentru a citi o valoare analogică de la un potențiometru integrat pe placa de laborator și a modului de configurare a regiștrilor asociați.

Se va lucra pe structura programului din laboratorul anterior, cu sistemul de întreruperi. Se începe prin a crea o funcție de inițializare a convertorului A/D numită *Init_ADC()*. Din registrul ADMUX se selectează ca și referință canalul AVCC iar canalul de pe care se face conversia rămâne 0, această valoare se va seta la apelul funcției de conversie. Din registrul ADCSRA se activează convertorul iar valoarea *prescaler-ului* este setată la 128. Această valoare conduce la o frecvență a ceasului pentru conversie de 62.5KHz. Cu cat frecvența e mai mică cu atât rezultatul conversiei este mai precis.

```
void Init_adc(){
  ADMUX = 0b01000000;          //Referința - AVCC
  ADCSRA= 0b10000111;          //Activare ADC; Prescaler = 128;
}
```

A doua funcție read*ADC(char ch)* are un singur parametru reprezentând canalul de pe care se va face conversia (în cazul de față 5 sau 6 corespunzând canalului ADC5 sau ADC6). În funcție se selectează canalul dorit, se pornește conversia și apoi se așteaptă resetarea bitului ADSC la 0 care arată finalizarea acesteia.

```
int readADC (char ch){
  ADMUX  &= 0b11100000;       //Resetează canalul de conversie
  ADMUX  |= ch;               //Setează canalul conversiei
  ADCSRA |= (1<<6);           //Începe conversia
    while(ADCSRA & (1<<6));   //Așteaptă finalizarea conversiei
  adc_l = ADCL;
  adc_h = ADCH;
return ((adc_h << 8) | adc_l);
}
```

Verificarea bit-ului ADSC se face software iar programul rămâne blocat în funcția *readADC* până la terminarea conversiei, aproximativ 210 µs. După finalizarea conversiei se construiește rezultatul pe 10 biți din cei doi regiștrii pe 8 biți, ADCH și ADCL. Rezultatul conversiei se returnează ca și valoare întreagă. Apelul funcției se va face la fiecare secundă, în funcția ISR a Timer-ului 0. Valoarea returnată de funcție va fi afișată pe display-ul BCD 7 Segmente.

2. Citirea senzorului de temperatură folosind întreruperi

Această aplicație are rolul de a exemplifica modul de configurare a *întreruperii ADC* pentru a semnaliza terminarea unei conversii A/D. Activarea întreruperii se face prin setarea bitului ADIE din registrul ADCSRA, instrucțiune adăugată în funcția de inițializare. De asemenea trebuie activate întreruperile la nivel global daca acest lucru nu a fost făcut altundeva în program.

```
void Init_adc(){
  ADMUX  = 0b01000000;   //Referința - AVCC
  ADCSRA = 0b10000111;   //Activare ADC; Prescaler = 128;
  ADCSRA |= (1<<3);      //Activare întrerupere ADIE
  SREG   |= (1<<7);      //Global Interrupt Enable
}
```

Pentru următorul pas se va crea rutina de întrerupere pentru convertorul A/D. Aceasta va fi apelată automat de fiecare dată când o conversie va fi finalizată. Rezultatul conversiei *adc* este convertit în tensiunea analogică (V_{in}) pe baza formulei (1). V_{in} este o variabilă de *tip float* la fel ca variabila *tmp.* Senzorul LM35 este un senzor analogic de temperatură și va avea o tensiune de ieșire proporțională cu temperatura cu o variație de 10mV/ºC. Intervalul de măsurare este [-55, 150] ºC dar în configurația de pe plca de dezvoltare, domeniul este restrâns la[2, 150]ºC iar la temperatura maxima va avea o tensiune de 1,5V pe ieșire. Unitatea de măsură pentru V_{in} este [V] iar pentru a transforma tensiunea în temperatură vom folosi ca unitate de măsură pentru tensiune [mV]. Variabila T va fi afișată pe display. Temperatura se calculează după formula:

$$T = \frac{V_{in} * 1000\ [mV]}{10\ [mV/℃]}$$

```
int adc_l, adc_h, adc, T;
float Vin, tmp;
void ADC_Completed() iv IVT_ADDR_ADC{
   adc_l = ADCL;
   adc_h = ADCH;
   adc = ((adc_h << 8) | adc_l);
   Vin = ((float)adc*5)/1024;
   tmp = Vin*1000/10;
   T = (int)tmp;
}
```

Pentru a porni conversia de pe canalul 7, se va apela funcția readADC_interrupt(7) în rutina de întrerupere ISR a Timer-ului 0, o data la o secunda. Noul corp al funcției este:

```
void readADC_interrupt(char ch){
  ADMUX  &= 0b11100000;       //Resetează canalul de conversie
  ADMUX  |= ch;               //Setează canalul conversiei
  ADCSRA |= (1<<6);           //Începe conversia
}
```

3. Continuare a punctului 2.

Să se modifice programul pentru a putea afișa pe display și prima zecimală a valorii temperaturii.

6. Generarea semnalelor PWM

6.1 Scopul lucrării

Lucrarea de față are ca scop înțelegerea mecanismului prin care se generează semnalul PWM cu ajutorul modulului de temporizare de pe un microcontroler ATmega32. De asemenea este exemplificat modul în care de folosește semnalul PWM pentru obținerea unui Convertor digital-analogic simplu.

6.2 Ce este un semnal PWM?

Semnalul PWM (sau Pulse Width Modulation) este un semnal periodic dreptunghiular la care se poate modifica în mod controlat factorul de umplere. Factorul de umplere este un procent din perioada semnalului pentru care semnalul se află în starea HIGH (T2/T1, T1 – perioada semnalului PWM, T2 – timpul pentru care semnalul este în starea HIGH). Un semnal PWM poate fi folosit în multe aplicații cum sunt cele care controlează motoarele de curent continuu sau comanda surselor de alimentare, abs-ul la mașini sau chiar obținerea unui Convertor Digital-Analogic simplu în combinație cu un filtru trece-jos.

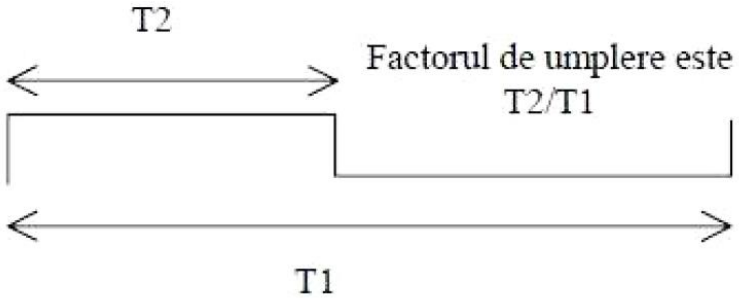


Figura6.1. Semnal PWM

Modul de operare a timer-ului în regim PWM este un caz particular al comparării la ieșire. Astfel timer-ul generează un semnal digital periodic care poate fi observat pe un pin digital de ieșire numit pin de comparare la ieșire (OCn). Timer-ul numără de la valoarea minimă (0) până la valoarea maximă (255) după care se resetează de la 0 din nou la următorul ciclu de ceas. În modul neinversat de funcționare, pinul OCn este resetat la 0 în momentul unei egalități între registrele TCNTn și OCRn și este setat la 1 când ajunge la valoarea minimă. În modul inversat pinul OCn este setat la 1 în momentul unei egalități între registrele TCNTn și OCRn și este resetat la 0 când ajunge la valoarea minimă. Funcționarea este descrisă în Figura6.2. Un timer pe 8 biți are domeniul de numărare [0, 255], iar în registrul OCR (Output Compare Register) asociat timer-ului se poate seta o valoare din acest domeniu. Registrului OCR este folosit pentru a impune factorul de umplere dorit pentru semnalul PWM generat. Pentru a seta frecvența acestui semnal PWM se va folosi semnalul de ceas al microcontrolerului în combinație cu un prescaler. Astfel că frecvența semnalului poate fi calculată după formula:

$$f_{PWM} = \frac{f_{clk\_I/O}}{\mathrm{N}\cdot 256} \quad (1)$$

unde $f_{clk_I/O}$reprezintă frecvența ceasului intern al sistemului, N este valoarea prescaler-ului (1, 8, 64, 256, 1024). Folosind această formulă se poate proiecta un semnal PWM de o frecvență dorită, totuși limitat în opțiuni posibile.

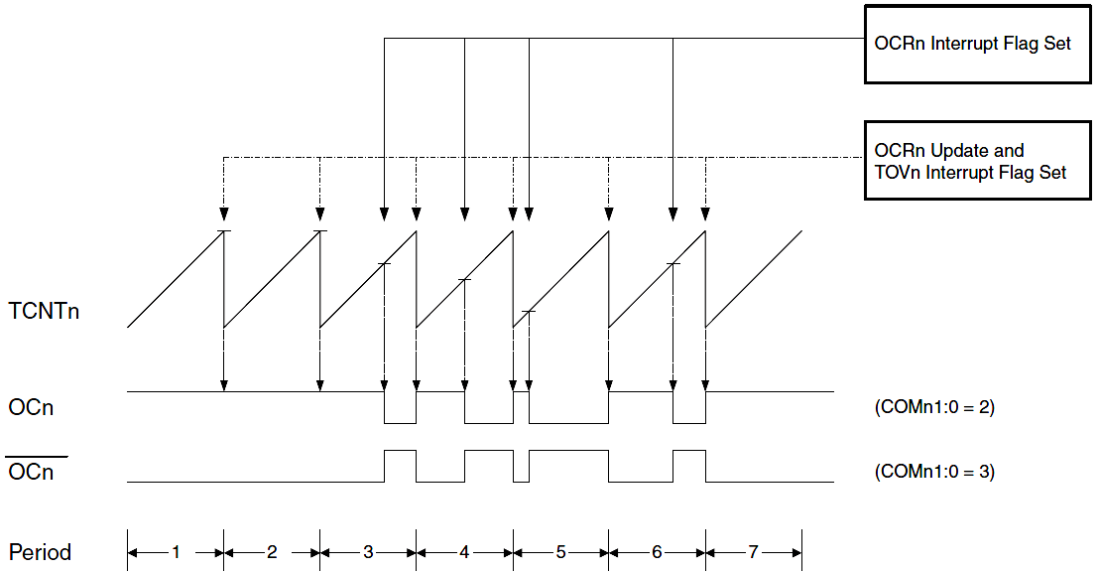


Figura6.2. Diagrama de timp pentru modul de operare PWM

Un timer care numără în domeniul [0, 255], cu registrul OCRn setat la 128, va ține nivelul de tensiune al pinului OCn la 5V cat timp TCNT < 128, odată ce TCNT trece de 128 nivelul de tensiune de pe pin va deveni 0V.

O altă caracteristică a unui mecanism de generare a semnalului PWM este rezoluția. Rezoluția este data de numărul de biți pe care este construit timer-ul. Daca un timer are registrul de numărare pe 8 biți, acesta numără de la 0 la 255, ceea ce conduce la 256 stări distincte.Chiar daca factorul de umplere este reprezentat de o valoare cuprinsă în intervalul [0, 1],totuși, setarea acestuia se face prin valorile echivalente din domeniul [0, 255],]n trepre de 1/256. Un număr de biți mai mare va duce la o rezoluție mai bună, ceea ce înseamnă că rezultatul va fi reprezentat cu o exactitate mai mare.

6.3 Convertorul digital-analogic

De cele mai multe ori microcontrolerele nu au capabilitatea unui canal analogic de ieșire de aceea daca o aplicație cere un convertor digital-analogic, acesta trebuie conectat extern. Din fericire este destul de ușor de obținut un convertor digital-analogic ieftin, astfel semnalele PWM sunt folosite de multe ori în combinație cu un filtru trece-jos RC (format dintr-o rezistenta şi un condensator) pentru a obține un convertor DAC (Digital to Analog Convertor). Semnalul PWM este netezit de filtrul RC (Figura6.3 (a)) atent dimensionat rezultând într-o valoare medie analogică a semnalului. Ieșirea filtrului care are la intrare un semnal PWM se poate observa în Figura6.3 (b).

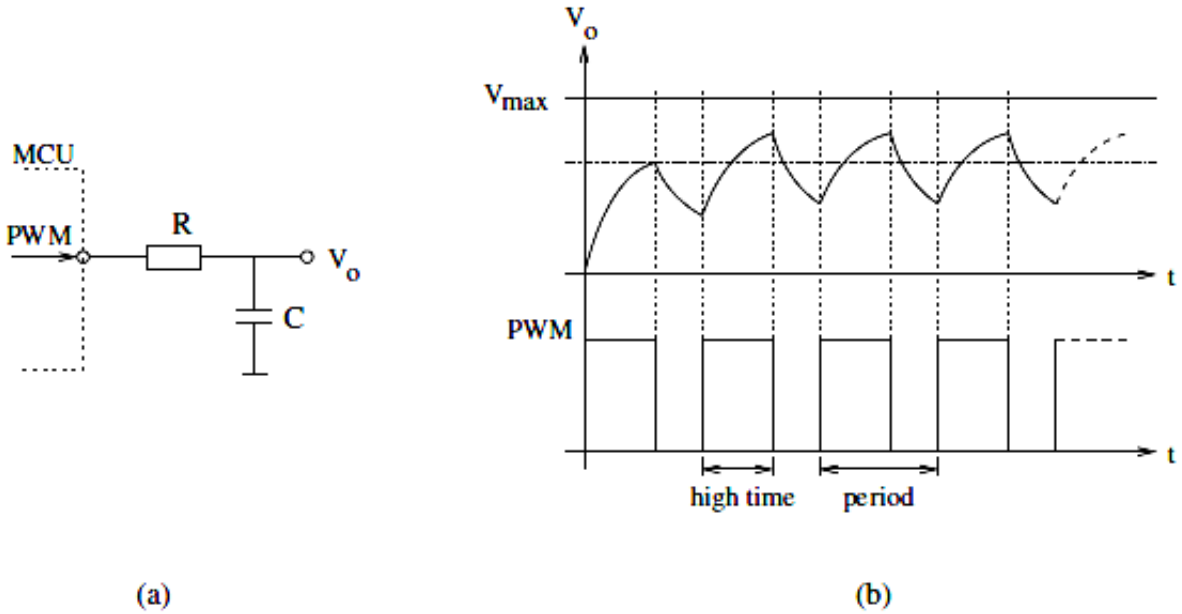


Figura6.3. Conversia digital-analogică folosind semnal PWM si filtru trece-jos RC; (a) circuitul, (b) Semnalul de ieșire în relație cu semnalul PWM

6.4 Desfășurarea lucrării

Această lucrare de laborator se bazează pe folosirea microcontrolerului ATmega32. Foaia de specificații (datasheet) poate fi consultată la adresa: (http://ww1.microchip.com/downloads/en/DeviceDoc/doc2503.pdf).

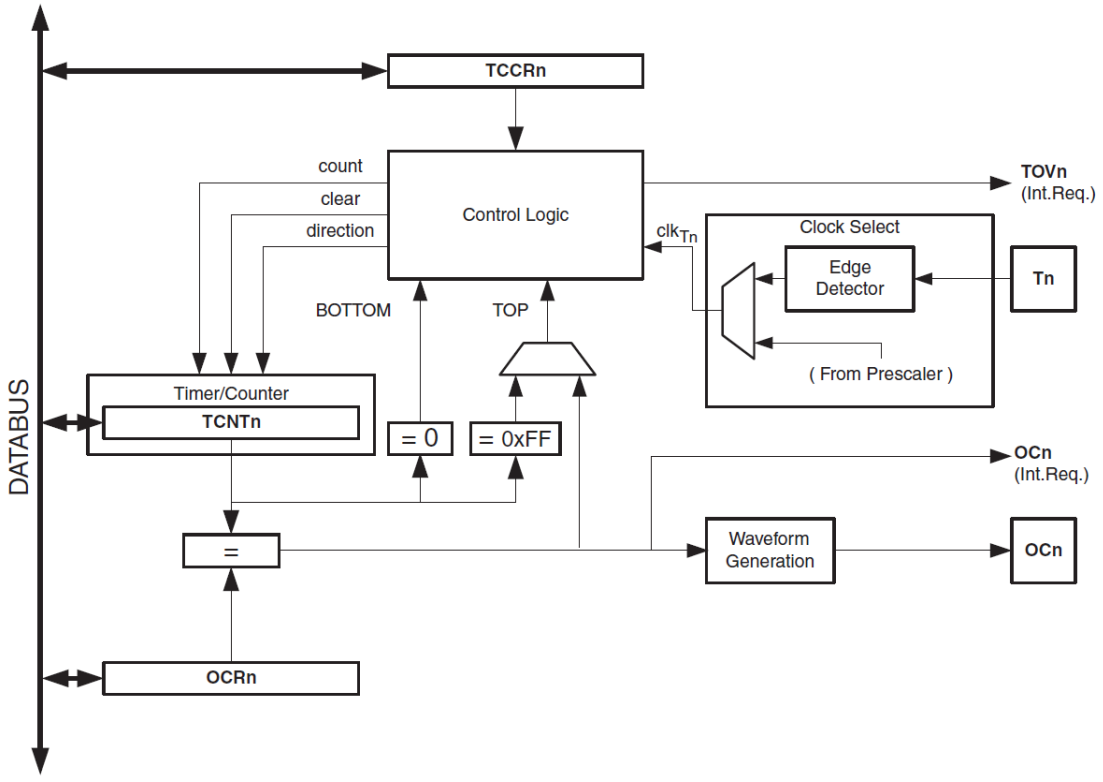


Figura6.4. Diagrama bloc a unui timer pe 8 biți

Schema bloc a unui modul timer a fost discutată într-o lucrare anterioară, totuși vom reveni asupra câtorva aspecte. Există trei regiştri importanţi în structura unui timer: TCCRn, TCNTn și OCRn. În structura microcontrolerului Atmega32 exista trei timere, Timer0 și Timer2 pe 8 biți și Timer 1 pe 16 biți. În funcție de timer-ul folosit, litera n din denumirea generică a regiștrilor esste înlocuită cu 0, 1 sau 2. În Figura6.4 este prezentată schema bloc a unui timer pe 8 biți având un singur canal de ieșire. Registrul TCCRn este folosit pentru configurarea diferitelor funcționalități ale timer-ului. Registrul TCNTn este registrul de numărare al timer-ului. Registrul OCRn este registrul de Comparare la Ieșire. Acesta din urmă este folosit și în procesul de generare a unui semnal PWM pe pin-ul OCn prin impunerea valorii factorului de umplere. Fiecare timer are un numar de pini asociați pe care se poate genera semnal PWM. În cazul microcontrolerului Atmega32 acesti pini sunt evidențiați în Figura6.5.

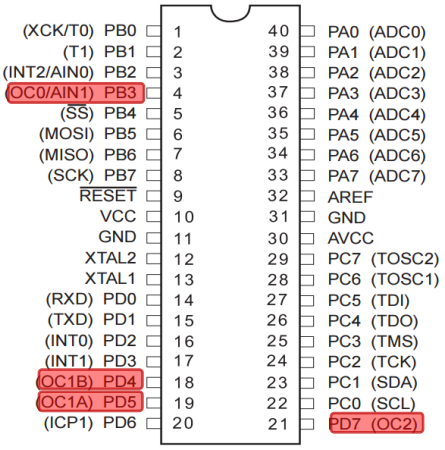


Figura6.5. Pinii cu funcția alternativă PWM

Descrierea regiștrilor

TCCR0 – Acest registru permite controlul modului de funcționare a Timer-ului.

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	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	

Biții WGM01:0 permit schimbarea modului de operare între Normal, CTC și PWM (Tabel 38 din foaia de specificații).

Table 38. Waveform Generation Mode Bit Description(1)

Mode	WGM01 (CTC0)	WGM00 (PWM0)	Timer/Counter Mode of Operation	TOP	Update of OCR0	TOV0 Flag Set-on
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR0	Immediate	MAX
3	1	1	Fast PWM	0xFF	BOTTOM	MAX

Biții COM01:0 permit schimbarea funcționalității pinului OC0 din pin digital de uz general în pin de OC precum în Tabel 40 din foaia de specificații. Pentru modul FastPWM

există 2 moduri de generare: inversat și neinversat așa cum se poate vedea și în Figura6.2 (OCn și $\overline{OCn}$).

Table 40. Compare Output Mode, Fast PWM Mode[1]

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Reserved
1	0	Clear OC0 on compare match, set OC0 at BOTTOM, (nin-inverting mode)
1	1	Set OC0 on compare match, clear OC0 at BOTTOM, (inverting mode)

Biții CS02:0 oferă posibilitatea selectării semnalului de ceas dintre ceasul intern, ceasul intern filtrat prin prescaler (1, 8, 64, 256, 1024) și semnal de ceas de la pinul T0 (PB0).

OCR0 – Registrul de comparare la ieșire conține o valoare pe 8 biți care este comparată mereu cu registrul de numărare (TCNT0). Egalitatea dintre cele doua registre este folosit pentru generarea de semnale pe pinul de ieșire OC0.

Bit	7	6	5	4	3	2	1	0	
	OCR0[7:0]								OCR0
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	

Exerciții:

1. Generarea unui semnal PWM cu factor de umplere variabil pe pinul PD7.

Potrivit cu Figura6.5, pinul PD7 are funcția alternativă OC2, ceea ce indică posibilitatea generării unui semnal PWM cu ajutorul Timer-ului 2, timer pe 8 biți. Ținând cont de formula pentru calculul frecvenței semnalului PWM, de frecvența semnalului de ceas (8MHz) și de factorii de prescalare posibili pentru Timer-ul 2 (din foaia de specificații) vom putea genera semnale PWM cu următoarele frecvențe:

N	f_{PWM}
1	31250 Hz
8	3906.25 Hz
32	976.56 Hz
64	488.28 Hz
128	244.14 Hz
256	122 Hz
1024	30.52 Hz

Se va crea o funcție de inițializare pentru configurarea Timer-ului 2 pe care o vom numi `init_PWM2`. Înainte de configurarea Timer-ului trebue însă să configurăm pinul PD7 ca pin de ieșire. Apoi, în registrul TCCR2 se vor face următoarele setări: Modul de operare – FastPWM, neinversat și setarea prescalerului dorit (aici N = 1).

Funcția de inițializare:

```
void init_PWM2(){
DDRD |= 1<<7;                  //PD7 – pin de ieșire
TCCR2 = 0b01101001;            //FastPWM neinversat, N=1
}
```

Factorul de umplere se setează în registrul OCR2. Pentru a putea să variem manual valoarea factorului de umplere ne vom folosi de potențiometrul P3 conectat pe pinul analogic PA6 (ADC6). Vom folosi aceeași funcție de conversie din laboratorul anterior pentru a obține valoarea conversiei analog-digitale de pe canalul ADC6. Conversia se va repeta la fiecare 100ms în funcția ISR de la Timerul 0 ca în laboratoarele anterioare. Rezultatul conversiei va fi folosit pentru a modifica valoarea factorului de umplere din registrul OCR2. Pentru că factorul de umplere este reprezentat de valorile echivalentedin intervalul [0, 255] iar rezultatul conversiei este cuprins în intervalul [0, 1023], acesta din urmă trebuie scalat prin împărțirea cu 4. Variabila `value` este o variabilă auxiliară și este folosită pentru a afișa pe display valoarea factorului de umplere.

Programul:

```
if (ms==99){
     value = (readADC(6)/4);
     OCR2 = value;
     ms=0;
}else ms++;
```

Dupa terminarea exercitiului cereți cardului didactic osciloscopul pentru a vizualiza semnalul PWM generat. Se observă că prin variația factorului de umplere se variază și intensitatea luminoasă a ledului asociat pinului PD7. Același semnal PWM poate fi folosit pentru variația vitezei unui motor de curent continuu spre exemplu.

2. **Modificați exercițiul 1 pentru a seta o frecvență a semnalului PWM sub 60 Hz. Ce observați?**

3. **Se va genera un semnal PWM de frecvență variabilă pe pinul PD4. Acesta va produce un semnal auditiv în buzzer-ului integrat pe placa de dezvoltare.**

Așa cum se poate vedea în Figura6.6, buzzer-ul are două posibilități de conectare cu pinii microcontrolerului prin intermediul jumper-ului J21. Consultând și Figura6.5 se observă că doar pe pinul PD4 se poate genera semnal PWM astfel, vom allege ca buzzer-ul să fie conectat cu acest pin.

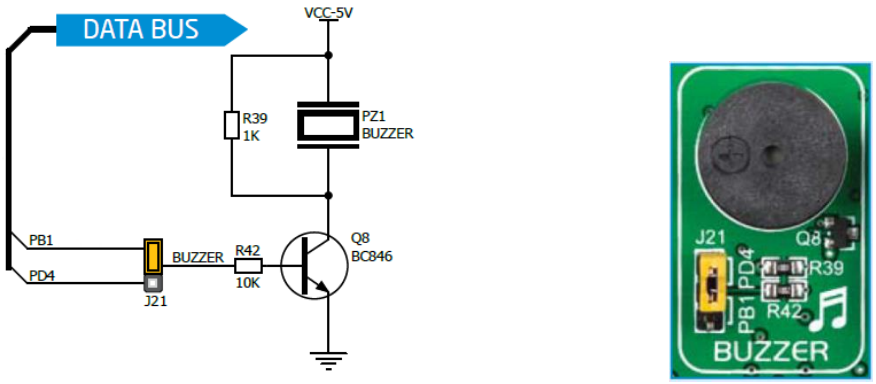


Figura6.6. Conexiune buzzer

Pentru a produce semnale auditive, buzzer-ul are nevoie de alimentarea cu un semnal periodic sinusoidal în domeniul 20 – 20000 Hz. Pentru acest scop, vom aproxima semnalul sinusoidal cu un semnal PWM cu factor de umplere de 50% ca în Figura6.7 cu singura precizare că sunetul va conține pe langă frecvența de baza si un numar mare de armonice.

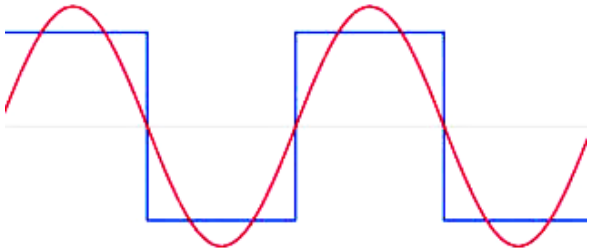


Figura6.7. Aproximarea unui semnal sinusoidal

Pinul PD4 are ca funcție alternativa OC1B, astfel, pentru generarea propriuzisă a semnalului PWM avem nevoie de Timer-ul 1, canalul B. Se va consulta foaia de specificații pentru a configura regiștrii TCCR1A și TCCR1B. Semnalul PWM se va alege ca FastPWM neinversat cu valoarea maximă de numărare (TOP) specificată în registrul ICR1 corespunzător cu modul de operare numarul 14 din tabelul Table 47 - Waveform Generation Mode din foaia de specificații. Factorul de prescalare va fi ales N = 8. Este important de asemenea ca pinul PD4 să fie configurat ca ieșire. Se va crea funcția de inițializare astfel:

```
void init_Buzzer(){
DDRD |= 1<<4;              //PD4 - output
TCCR1A = 0b00100010;
TCCR1B = 0b00011010;
}
```

Se va lucra mai departe pe structura programelor din lucrările anterioare folosind funcția de intrerupere ISR de la Timer-ul 0. În această funcție, la fiecare 10 ms se va apela funcția adc(6) pentru a obține rezultatul conversiei de pe canalul ADC6 unde este legat potențiometrul P3. Această valoare va fi folosită mai departe pentru variația frecvenței semnalului PWM. Timer-ul 1 este un timer pe 16 biți ce oferă o funcționalitate mai complexă. Astfel, frecvența PWM se va calcula după formula:

$$f_{PWM} = \frac{f_{clk\_I/O}}{N \cdot (1+TOP)} \quad (2)$$

unde TOP este valoarea pe care o vom programa în registrul ICR1 și este cuprinsă în domeniul [0; 65535]. Am ales acest mod de funcționare pentru a adăuga această variabilă în ecuație. Astfel vom putea genera un semnal PWM cu o varietate foarte mare de frecvențe posibile. De asemenea, factorul de prescalare a fost ales pentru ca frecvențele rezultate să se încadreze în domeniul auditiv 20 – 20000 Hz.

Domeniul de variație al rezultatului conversiei AD, [0, 1023], va fi transformat în domeniul [20, 20000]. Se va folosi o funcție de ordin I, $f_{(x)} = ax + b$, unde $x \in [0, 1023]$ și $f_{(x)} \in [20, 20000]$. Pentru

$$f_{(0)} = ax + b = 20 \rightarrow b = 20,$$

$$f_{(1023)} = a \cdot 1023 + 20 = 20000 \rightarrow a = \frac{20000 - 20}{1023}$$

astfel,

$$f_{(x)} = \frac{20000-20}{1023} \cdot x + 20 = \frac{f_{Max}-f_{Min}}{ADC_{Max}} \cdot x - f_{Min} \quad (3)$$

Cu formula din urmă se calculează frecvența corespunzătoare gradului de rotire a potențiometrului. Aplicând formula 2 pentru a calcula valoarea TOP în funcție de f_{PWM} și o frecvanță de ceas $f_{clk\_I/O} = 8\,MHz$, se obține $TOP = \frac{1000000}{freq} - 1$. Se observă ca valoarea TOP se încadrază în domeniul [0, 65535].

$$freq = 20 \rightarrow TOP = 49999$$
$$freq = 20000 \rightarrow TOP = 49$$

Apoi, valoarea TOP este scrisă în registrul ICR1. Pentru a păstra factorul de umplere la 50%, după fiecare schimbare, registrul OCR1B trebuie actualizat cu jumătatea valorii TOP+1. Cu scopul de a putea afișa valoarea frecvenței pe display vom împărți frecvența la 10 si vom stoca valoarea în variabila `value`. Variabila `value` va fi afișată pe display.

```
int ADCMax=1023, fMax=20000, fMin=20, freq, icr, TOP, value;
if (ms == 9){
     adv = (adc(6));
     freq = ((float)(fMax-fMin))/ADCmax*adv+fMin;
     TOP = 1000000/freq-1;
     ICR1H = TOP>>8;
     ICR1L = TOP;
     OCR1BH = ((TOP+1)/2)>>8;
     OCR1BL = (TOP+1)/2;
     value=freq/10;
     ms=0;
}else ms++;
```

Dupa terminarea exercitiului cereti cardului didactic osciloscopul pentru a vizualiza semnalul PWM generat.

7. Captura la Intrare

7.1 Scopul lucrării

Lucrarea are ca scop înțelegerea mecanismului de întrerupere de tip captură la intrare (Input Capure) prezent la cele mai multe microcontrollere precum și prezentarea unei aplicații în care se folosește.

7.2 Ce este Captura la Intrare?

Funcția de *captură la intrare*este legată de funcționarea unui timer. Este folosită pentru a detecta apariția unor evenimete externe și pentru marcarea timpul exact al apariției acestora. Evenimentele pot fii constituite de un front crescător sau/și descrescător sau de un anumit nivel logic. Atunci când are loc acest eveniment materializat printr-o tranziție pe pinul ICPn al microcontrolerului (Figura7.1), valoarea curentă a numărătorului din registrul TCNTn este copiată automat într-un registru ICRn (Input Capture Register), care poate fi citit de program. Valoarea din ICRn poate fi folosită pentru calculul frecvenței, factorului de umplere a semnalului sau pentru alte calcule cum ar fi determinarea turației unui motor cu encoder. Citirea registrului ICRn pe 16 biți se face citind mai întai byte-ul low (ICR1L) și spoi byte-ul high (ICR1H). La momentul evenimentului, timer-ul setează de asemenea și un *flag* (ICFn) care poate fi folosit pentru generarea unei întreruperi și pentru a semnaliza programul că a avut loc un eveniment de captură la intrare. Pot exista unul sau mai mulți pini de acest fel. Un astfel de pin de captură poate fi folosit de asemenea și cu funcția generală de pin digital de intrare.

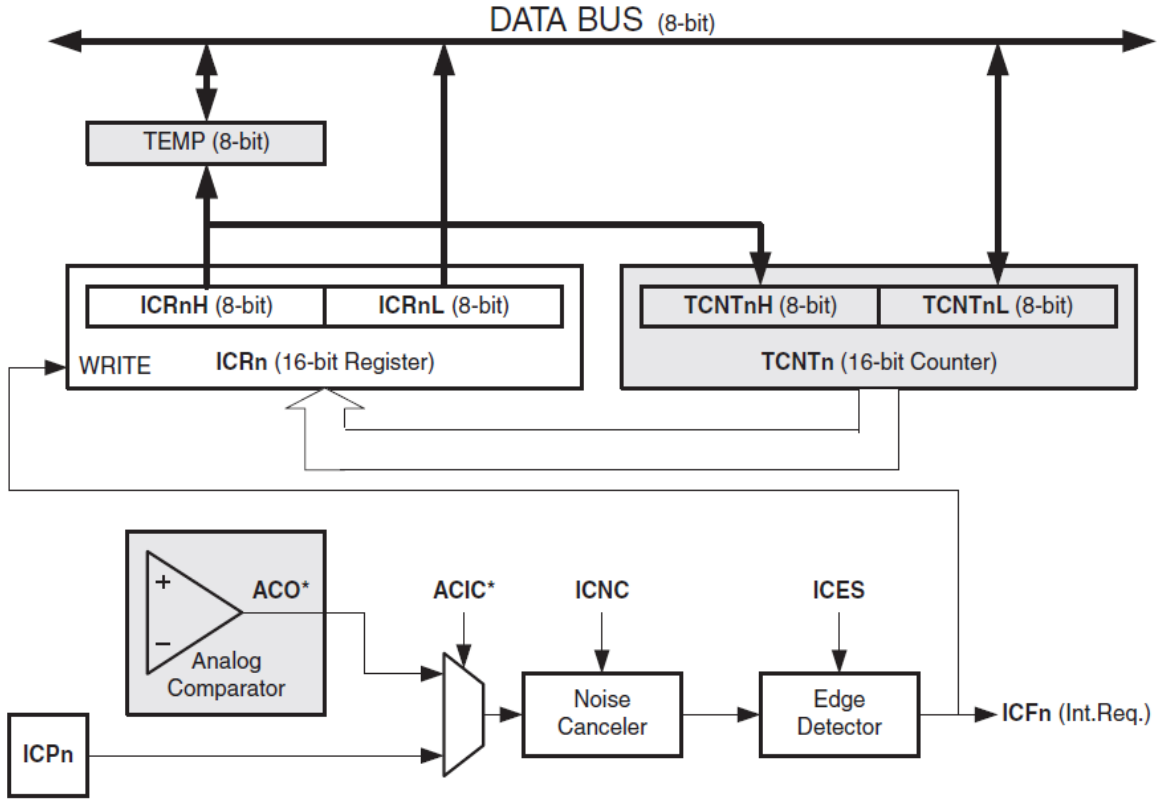


Figura7.1. Schema bloc a unității de captură la intrare

7.3 Motorul de curent continuu

Motorul folosit în această lucrare de laborator este un motor de curent continuu cu reducție **18.75:1**și encoder în cuadratură (Figura7.2). Motorul operează la o tensiune maximă de 12V şi o turație maximă a axului secundar de 500 rotații pe minut (RPM).

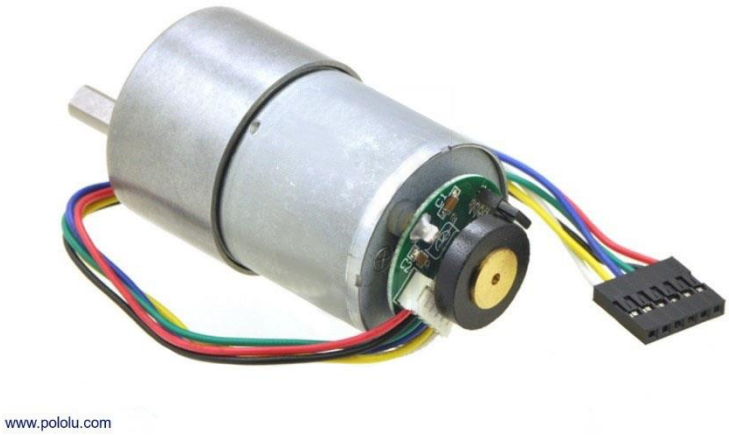


Figura7.2. Motor de curent continuu cu reducție și encoder integrat

Pentru a putea determina viteza de rotație a motorului se folosește un encoder cu senzor Hall cu două canale. Senzorul Hall detectează prezența unui disc magnetic prins pe axul principal al motorului. Encoderul în cuadraturăoferă o rezoluție de 64 impulsuri per rotație pentru axul principal. Pentru axul secundar, se multiplică numărul de impulsuri cu valoarea reducției (18.75) și se ajunge la un umar de 1200 impulsuri per rotație. Se ajunge la aceste valori dacă se folosesc ambele canale de ieșire din senzorul Hall și se contorizează atât frontul crescător cât și frontul descrescător. Ieșirea unui astfel de encoder este reprezentată de două semnale dreptunghiulare defazate unul față de celălalt cu 90° (Figura7.3). Cu ajutorul unui astfel de senzor se poate detecta si sensul de rotație. Motorul are șase terminale codificate în culori așa cum se poate vedea în Figura7.2. Tabelul următor descrie funcțiia fiecărui terminal:

Culoare	Funcție
Roșu	Alimentare motor
Negru	Alimentare motor
Verde	Masa encoderului (GND)
Albastru	Alimentare encoder VCC (3.5 – 20V)
Galben	Ieșire A encoder
Alb	Ieșire B encoder

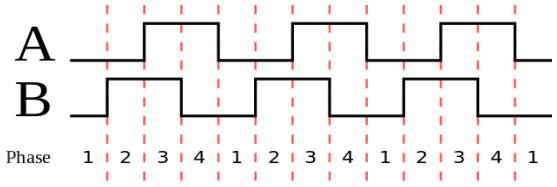


Figura7.3. Ieșirea unui encoder în cuadratură

Pentru scopul acestei lucrări se va folosi o singură ieșire a senzorului Hall și se vor detecta doar fronturile crescătoare. Astfel vom avea 16 impulsuri per rotație pe axul principal și 300 impulsuri pe axul secundar.

7.4 Desfășurarea lucrării

Pentru implementarea exercițiilor se va folosi mediul de programare mikroC PRO și va trebuie consultă foaia de specificații ATmega32

(http://ww1.microchip.com/downloads/en/DeviceDoc/doc2503.pdf).

Descrierea regiștrilor

TCCR1B – Acest registru permite controlul modului de funcționare a Timer-ului 1.

Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bitul ICES1 permit setarea frontului pe care să aibă loc întreruperea de Captură la Intrare. Prin setarea acestuia cu 1 logic se selectează frontul crescător iar prin setarea cu 0 logic se selectează frontul descrescător.

ICR1 – Acest registru este încărcat cu valoarea registrului de numărare TCNT1 în momentul apariției unui eveniment de Captură la Intrare.

Bit	7	6	5	4	3	2	1	0	
	ICR1[15:8]								ICR1H
	ICR1[7:0]								ICR1L
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	

TIMSK – Acest registru permite activarea diferitelor întreruperi care sunt legate de Timere.

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
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	

Bitul TICIE1 permite activarea întreruperii de Captură la Intrare asociat Timer-ului 1. Evenimentul care declanșează întreruperea va fi reprezentat de un front crescător sau descrescător pe pinul ICP1.

Bitul TOIE1 permite activarea unei întreruperi de Overflow care se activează în momentul când timerul ajunge la valoarea maximă de numărare.

Exerciții:

1. **Folosirea întreruperii de Captura la Intrare (Input Capture) pentru calculul turației unui motor de curent continuu cu encoder.**

Pentru configurarea întreruperii de captura la intrare (IC) se creează o funcție de inițializare `Init_Input_Capture()` după cum urmează în continuare. Inițial trebuie să ne asigurăm că pinul PD6 (ICP1) este configurat ca intrare digitală. Apoi, folosind registrul TCCR1B se

alege frontul crescător pe pinul ICP1 pentru întreruperea IC și se alege un factor de prescalare de 8 pentru divizarea semnalului de ceas până la 1 Mhz. Aceasta înseamnă că fiecare impuls de ceas al Timer-ului 1 contorizează 1µs.

Din registrul TIMSK se vor activa întreruperile de IC și de Overflow. La fiecare front crescător detectat pe pinul ICP1 se va salva valoarea registrului de numărare TCNT1 în registrul ICR1 și se va apela funcția ISR asociată. Timerul face un ciclu complet de numărare în $T_{ciclu} = 65535 \cdot 1\mu s = 65535\mu s = 65.535ms$. Dacă timpul între două fronturi crescătoare este mai mare decat T_{ciclu} rezultatele vor fi eronate pentru că nu seva tine cont că timer-ul a fost resetat. De aceea, întreruperea de Overflow are rolul de a contoriza numărul de resetări ale numărătorului între două fronturi crescătoare de pe pinul ICP1. Astfel că, la fiecare întrerupere de Overflow se va apela automat funcția ISR corespunzătoare și variabila `cycle` va fi incrementată. La apariția unei noi întreruperi IC variabila `cycle` se va reseta.

```
void Init_Input_Capture(){
     DDRD &=~(1<<6);     //PD6 - intrare
     TCCR1A = 0b00000000;
TCCR1B = 0b01000010;//bit 6-front crescător,bit0-2Pre=8
TIMSK |= 1<<5;      //Input Capture Interrupt Enable
TIMSK |= 1<<2;      //Overflow Interrupt Enable
}
```

Primul lucru care trebuie făcut după apariția unei întreruperi de IC este resetarea registrului de numărare TCNT1 și salvarea valorii din registrul ICR1 într-o variabilă de program. Timpul între două fronturi crescătoare se calculează adunând valorii din registrul ICR1 numărul de cicluri de resetare înmultit cu numărul de impulsuri de ceas dintr-un ciclu (65535). Pentru că, un ciclu de ceas este echivalent cu 1µs, valoarea rezultată reprezintă perioada dintre 2 fronturi crescătoare și înmulțind cu 300 imp/rot, obținem timpul unei rotații complete:$T_{front} = (ICR1 + 65535 \cdot cycle)\mu s$. Aplicând o regulă de trei simplă se poate calcula turația.

$$1\ imp \dots\dots\dots\dots\dots\dots T_{front}\ \mu s$$
$$x\ imp \dots\dots\dots\dots\dots\dots 1\ min$$

$$x = \frac{60000000}{T_{front}} \Rightarrow RPM = \frac{x}{300}\ rot/min$$

```
int rpm, value=0;
long cycle=0, trot, icr_l, icr_h;

void IC_ISR() iv IVT_ADDR_TIMER1_CAPT ics ICS_AUTO {
     TCNT1H=0;
     TCNT1L=0;
     icr_l = ICR1L;
     icr_h = ICR1H;
     trot = (((icr_h << 8) | icr_l)+65536*cycle)*300;
     rpm = 60000000/trot;
     cycle=0;
}

void COMP_ISR() iv IVT_ADDR_TIMER1_OVF ics ICS_AUTO {
      cycle++;
}
```

Pentru a putea varia turația motorului se foloseste generează un semnal PWMpe pinul PD7 a cărui factor de umplere poate fi variat folosind potențiometrul P3. Unul din terminalele motorului se va leaga la masă iar celălalt se alimentează cu semnalul PWM trecut printr-o punte de amplificare (L293).

8. Convertorul Digital-Analogic

8.1 Scopul lucrării

Lucrarea de față are ca scop înțelegerea conceptului de convertor digital analogic și expunerea unor caracteristici de bază pentru proicetarea unui astfel de circuit de conversie. De asemenea este exemplificat modul în care se folosește semnalul PWM în combinație cu un filtru trece-jos pentru obținerea unui Convertor digital-analogic simplu.

8.2 Convertorul digital-analogic

De cele mai multe ori microcontrolerele nu au capabilitatea unui canal analogic de ieșire de aceea dacă o aplicație cere un convertor digital-analogic (D/A), acesta trebuie conectat extern. Din fericire este destul de ușor de obținut un convertor digital-analogic ieftin utilizând o ieșire PWM.Astfel, semnalele PWM sunt folosite de multe ori în combinație cu un filtru trece-jos RC (format dintr-o rezistenta si un condensator) pentru a obține un convertor DAC (Digital to Analog Convertor). Semnalul PWM este netezit de filtrul trece-jos RC (FTJ - Figura8.1 (a)) atent dimensionat rezultând într-o valoare medie analogică a semnalului proporţională cu factorul de umplere,Figura8.1 (b). Bineînțeles, semnalul analogic rezultat va avea o întârziere pînă se va stabiliza iar semnalul va oscila în jurul valorii de ieşire dorite. Această oscilație depinde de valorile R și C ale filtrului. Pentru a reduce amplitudinea oscilațiilor putem fie să creștem valorile R și C, fie să creștem frecvența semnalului PWM.

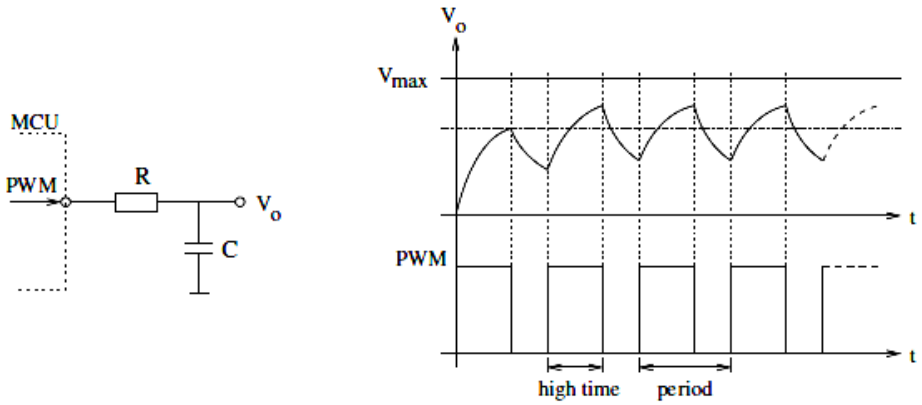


Figura8.1. Conversia digital-analogică folosind semnal PWM si filtru trece-jos RC; (a) circuitul, (b) Semnalul de ieșire în relație cu semnalul PWM

Dezavantajul folosirii semnalului PWM este acela că este necesar să folosim un Timer dedicat și de asemenea e nevoie să așteptăm o perioadă de timp până se va stabiliza semnalul de ieșire. Ca și avantaj, acest convertor D/A folosește un singur pin al microcontrolerului.

O altă metodă de conversie D/A folosește o rețea de rezistențe ponderate sau o rețea de rezistențe R-2R. Aceast tip de convertor D/Afolosește doar rezistențe de două valori R și 2R. Un astfel de convertor cu rezoluția pe r biți este prezentat în Figura8.2. Ieșirea unui astfel de convertor se bazează pe semnalele digitale $b_0 - b_{r-1}$. La nivel ideal, acest convertor dă rezultate foarte bune dar în practică, precizia circuitului este influențată de valorile de toleranță ale rezistențelor, pentru că valoarea reală a rezistenței este diferită de valoarea de catalog. De asemnea, în cazul unui astfel de circuit de conversie avem nevoie de r pini digitali (ex: 8, 10, 12 sau 16) dedicați pentru conversie.

Mai există posibilitatea folosirii circuitelor integrate pentru conversie D/A ce comunică serial cu microcontrolerul prin diferite protocoale cum ar fi: ISP, I2C, etc. Totuși pentru scopul lucrării de față vom folosi primul circuit de conversie A/D folosind semnalul PWM și un FTJ de ordin I.

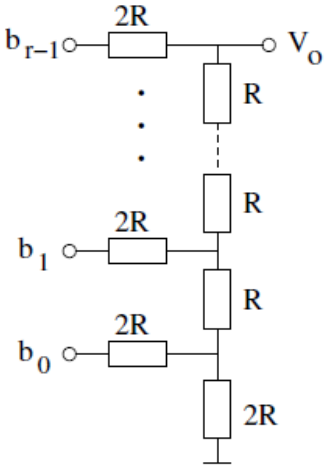


Figura8.2. Convertor Digital-Analogic cu rețea de rezistențe R-2R

1. Filtrul trece-jos

Cu scopul obținerii unui circuit de conversie D/A folosind un semnal PWM si un FTJ, se recomandă ca ordinul filtrului să nu fie mai mare de ordinul I. O creștere a ordinului va duce la o complexitate crescută a circuitului care nu se justifică. Pentru rezultate mai bune este indicată folosirea unui circuit integrat de conversie.

O mărime importantă în proiectarea unui filtru este frecvența de tăiere a filtrului care se calculează după formula:

$$f_T = \frac{1}{2\pi RC}$$

Orice semnal de frecvență mai mare decât frecvența de tăiere va fi atenuat, în timp ce semnalele cu frecvență mai mică decât frecvența de tăiere vor fi lăsate să treacă neatenuate. Pentru proiectarea FTJ se folosesc diagramele Bode în care se pune în evidență punctul de inflexiune ce corespunde cu frecvența de tăiere a filtrului ca în Figura8.3.

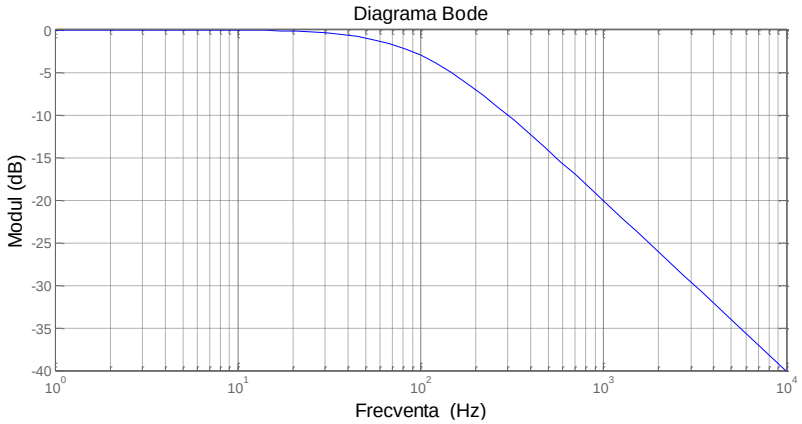


Figura8.3. Model de diagram Bode

Diagrama din Figura8.3 corespunde unui filtru FTJ cu $f_T = 100\,Hz\,(10^2)$. De pe diagram se citește că pentru semnale de 1kHz există o atenuare de -20dB ceea ce corespunde unei atenuări de 10 ori față de semnalul original. Pentru semnale de 10kHz există o atenuare de -40dB ceea ce corespunde unei atenuări de 100 ori față de semnalul original. Pentru a avea o amplitudine a oscilațiilor suficient de scăzută (Figura8.1 (b)) trebuie să avem o atenuare a semnalului PWM între 10 și 100 de ori. Folosind exemplul anterior, frecvența PWM ar trebui să fie mai mare decât $f_T = 100\,Hz$ cu cel puțin un ordin de marime (1kHz).

Semnalul de ieșire pentru care proiectăm filtrul ar trebui sa aibă o frecvență mai mică decât frecvența de tăiere pentru a nu fi atenuat.

8.3 Desfășurarea lucrării

Această lucrare de laborator se bazează pe folosirea microcontrolerului ATmega32. Foaia de specificații (datasheet) poate fi consultată la adresa (http://ww1.microchip.com/downloads/en/DeviceDoc/doc2503.pdf).

Exerciții:

1. **Generarea unui semnal sinusoidal folosind semnal PWM pe pinul PD7 și un filtru RC trece-jos.**

Se va genera un semnal PWM cu frecvența de 31250Hz pe pinul PD7 așa cum a fost exemplificat într-o lucrare anterioară. Factorul de umplere al semnalului PWM va fi variat după forma unui semnal sinusoidal în domeniul [0, 1] folosind valorile corespunzătoare din domeniul [0, 255]. După filtrarea semnalului PWM printr-un circuit FTJ va rezulta un semnal sinusoidal.

Funcția de inițializare PWM:

```
void init_PWM2(){
DDRD |= 1<<7;              //PD7 – pin de ieșire
TCCR2 = 0b01101001;        //FastPWM neinversat, N=1
}
```

Generare valori sinus:

Pentru generarea semnalului sinus se va defini un vector cu valori discrete ale sinusului pe parcursul unei perioade. Pentru exemplul de față se vor genera 20 de valori folosind programul Matlab. Codul următor va ajuta la generarea celor 20 de valori.

```
N = 20;          //Numărul de valori generate
Te = 1/N;        //Perioada de eșanitionare
t = 0:Te:1-Te;   //se crează vectorul timp
s = sin(2*pi*t);//se crează vectorul sinus
s = s+1;         //se trece s din domeniul [-1,1] în [0,2]
s = s*127.5;     //se trece s din domeniul [0,2] în [0,255]
s = round(s);    //se convertește din valori reale în întregi
```

Cu valorile generate în Matlab se va defini în MikroC PRO un vector de numere întregi. Factorul de umplere al semnalului PWM se setează în registrul OCR2 cu valorile generate. Perioada aleasă pentru semnalul sinus este 1Hz. De aici rezultă o perioada de eșantionare de 50 ms. Codul următor va apărea în funcția ISR de la Timer-ul 1.

Programul:

```
int sinus[] =
{127,166,202,230,248,254,248,230,202,166,127,88,52,24,6,0,6,24
,52,88};

if (ms==49){         //Perioada sinus T=50*20=1000ms => f =1Hz
  OCR2 = sinus[i++];//factor de umplere
  if(i==20)          //resetarea contorului
     i=0;
  ms=0;
}else ms++;
```

Filtrul trece-jos:

Pentru că frecvența semnalului PWM a fost fixată la 31250Hz și ținând cont de cele discutate mai sus, vom alege frecvența de tăiere a filtrului RC în intervalul [312, 3125] Hz. Valorile R și C se vor alege în așa fel ca produsul lor să fie cuprins în intervalul amintit.

2. **Modificați exercițiul 1 pentru a schimba frecvența sinusului la o valoare de 100 Hz.**

3. **Modificați exercițiul 1 pentru a genera un alt semnal analogic ales ca sumă de două sinusuri cu frecvențe diferite.**

9. Controlultemperaturii - Aplicație

9.1 Scopul lucrării

Lucrarea de față are ca scop folosirea noțiunilor învățate până la momentul de față și integrarea lor într-o aplicație de control bipoziţional al temperaturii(ON - OFF). Se va folosi senzorul analogic de temperatură LM35 și mediul de programare AVR Studio.

9.2 Descriere placă de laborator

Lucrarea de laborator se bazează pe folosirea plăcii de laborator din Figura9.1. Placa este construită în jurul unui microcontroler Atmega16, din aceeași familie cu Atmega32, una dintre diferențele majore fiind dimensiunea memoriei de program (FLASH), 16KB față de 32KB. Componentele integrate pe placă și conexiunea acestora cu microcontrolerul pot fi vizualizate în schema circuitului din Figura9.2.

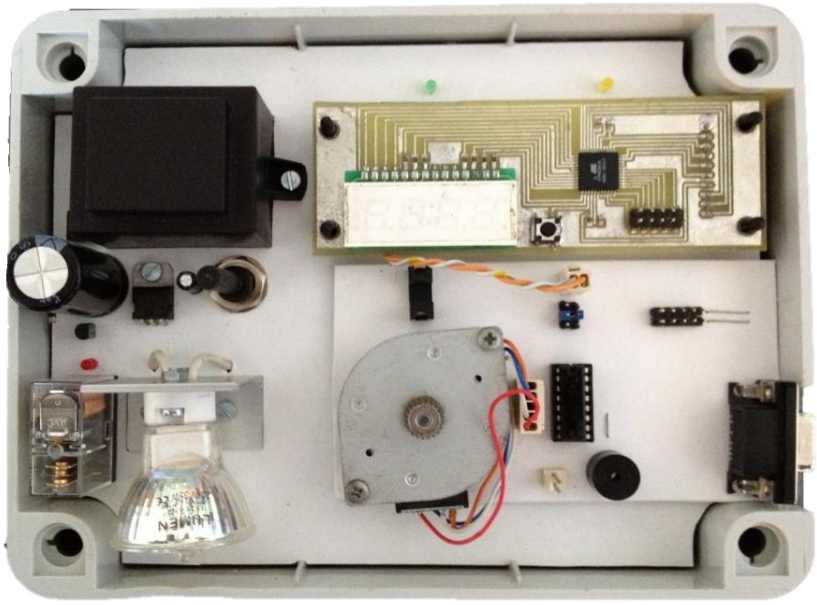


Figura9.1. Placa de laborator cu microcontroler Atmega16

Avem astfel un display 7 Segmente conectat cu pinii de activare (enable) pe pinii PA0-PA3 și pinii pentru selecția segmentelor pe pinii PB0-PB6. Se va acorda atenție la pinii de selecție a segmentelor, întrucât aceștia sunt activi pe 0 logic, corespunzător cu varianta constructivă cu anod comun descrisă într-o lucrare anterioară. Pe pin-ul PA5 este conectat un LED roșu activ pe 1 logic. În paralel cu LED-ul este conectat și un bec care se va aprinde odată cu aprinderea LED-ului roșu. Pe pin-ul PB7 este conectat un buton fără rezistență de Pull-UP externă. De aceea, în faza de configurare a pin-ului se va activa rezistența de Pull-UP internă pentru o funcționare corectă a butonului. În ceea ce privește componentele analogice, există un potențiometru conectat pe pin-ul PA6 și un senzor de temperatură LM35 pe pin-ul PA7. Ieșirea din senzorul de temperatură este amplificată de 4 ori într-un montaj cu amplificator operațional. Astfel, ieșirea senzorului va fi de 40mV/°C.

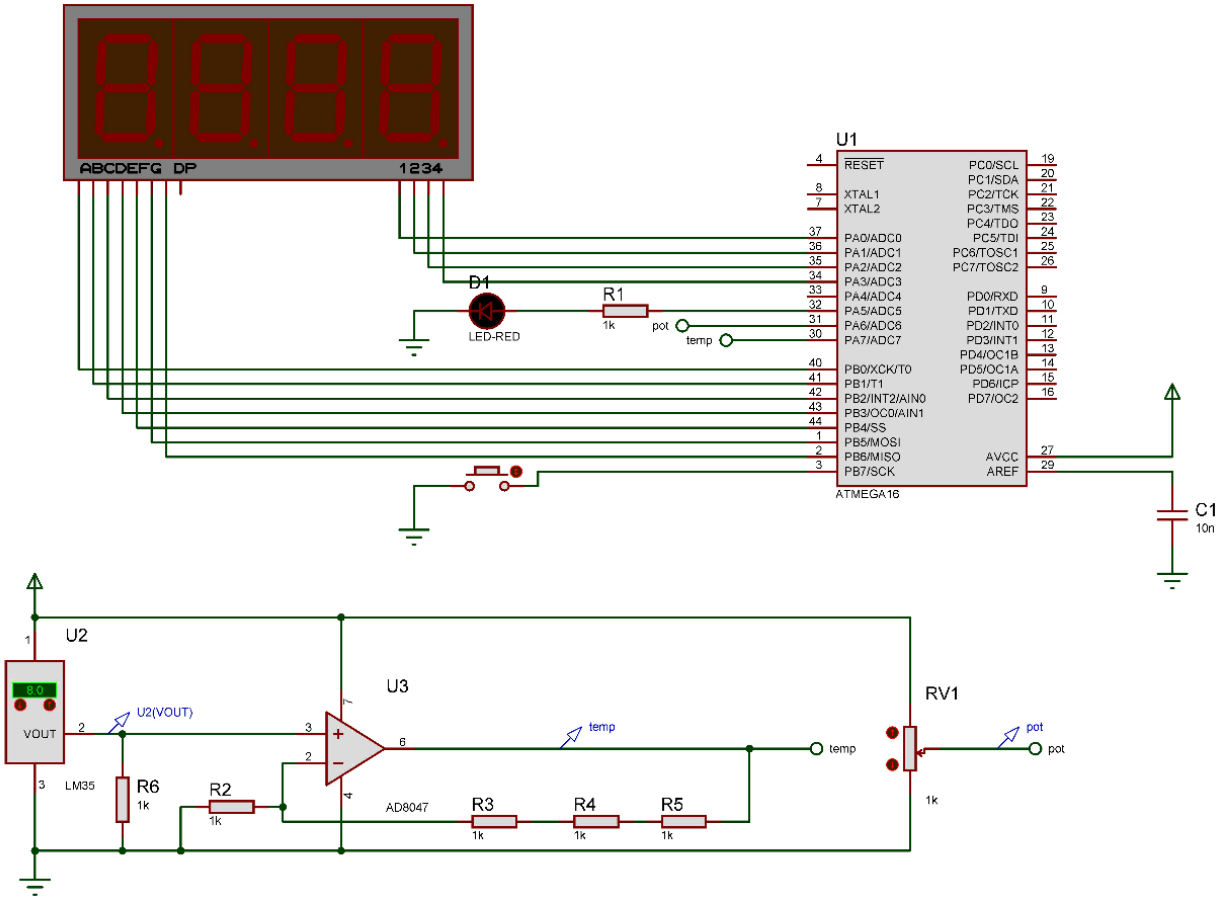


Figura9.2. Schema de conexiuni a plăcii de laborator

Placa de laborator a fost gândită astfel încât să existe o influență fizică a becului asupra senzorului de temperatură prin poziționarea acestora. Când becul se aprinde, puterea disipată de acesta va duce la încălzirea senzorului de temperatură. Prin comenzi ON-OFF ale pin-ului PA5 se va controla temperatura din incinta becului la o temperatură prestabilită.

9.3 Controlul bipozițional cu histereză

Un sistem de control este un sistem care generează o comandă pe baza unui semnal de intrare, în cazul de față semnalul de intrare este reprezentat de valoarea temperaturii obținută de la senzorul LM35. Sistemul de control bipozițional este un dispozitiv care acționează în doar două stări: ON-OFF. Aceste stări sunt calculate ca rezultat al comparării dintre temperatura de la senzor și o temperatură de referință. ON pentru cazul în care valoare de intrare (valoarea temperaturii măsurate) este mai mică decât valoarea de referință și OFF pentru cazul în care valoarea de intrare este mai mare sau egală cu valoarea de referință.

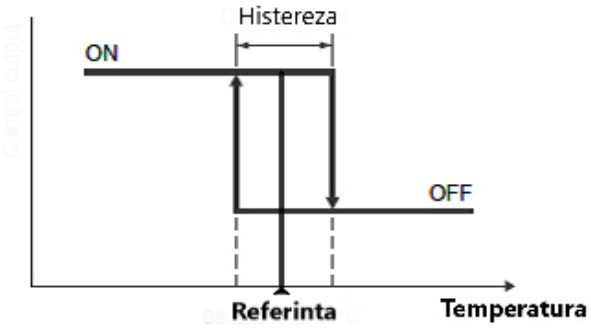


Figura9.3. Control bipoziţional cu histereză

În cele mai multe cazuri, controlul bipoziţional se întâlneşte cu o caracteristică cu aspect de histereză ca în Figura9.3. Histereza se defineşte ca un interval în jurul valorii de referinţă (r) având lăţimea Δ, Δ fiind valoarea histerezei. Astfel, într-un sistem de control bipoziţional cu histereză trecerea din stare ON în stare OFF se face la r+Δ/2, iar trecerea din starea OFF în starea ON se face la r-Δ/2. În stare ON becul este aprins şi încălzeşte senzorul iar în starea OFF se stinge şi implicit senzorul se răceşte. Lăţimea histerezei defineşte cât de mari sunt deviaţiile de temperatură în jurul valorii de referinţă. Variaţiile temperaturii într-un sistem cu control bipoziţional cu histereză este exemplificat în Figura9.4.

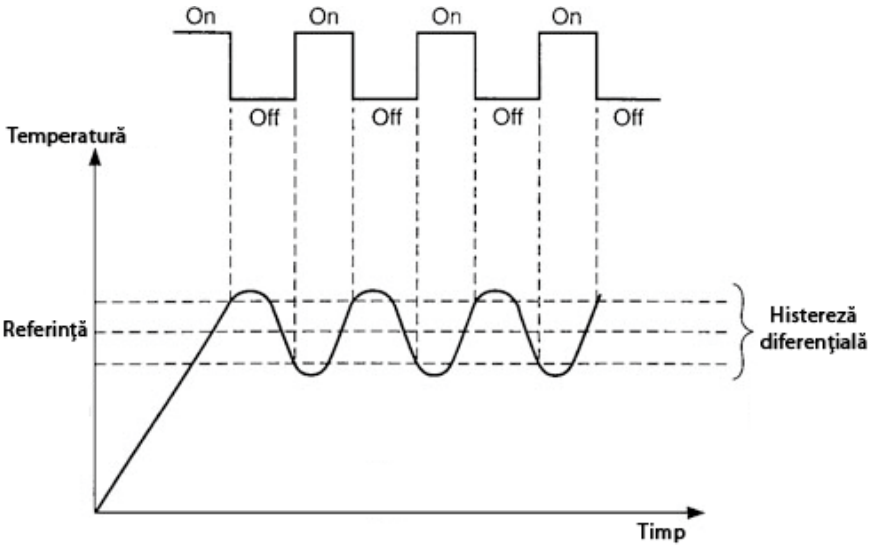


Figura9.4. Variaţiile temperaturii într-un sistem de control bipoziţional cu histereză

9.4 Desfăşurarea lucrării

Pentru implementarea exerciţiilor se va folosi mediul de programare AVR Studio 4.0 şi va trebuie consultă foaia de specificaţii ATmega16 (http://ww1.microchip.com/downloads/en/DeviceDoc/doc2466.pdf).

AVR Studio 4.0

După deschiderea programului se creează un proiect nou de tip ***AVR GCC.*** La opţiunea ***Debug Platform*** se selectează ***JTAG ICE,*** iar la ***Device***, ***ATmega16.*** După finalizarea procesului de creare a proiectului nou seaccesează ***Project/Configuration Options.*** Se verifică opţiunile ***Device: atmega16***, ***Frequency: 8000000***şi***Optimization: -O0***.

În Figura9.5 se poate vedea interfața programului și componentele mediului de lucru. Programul se scrie în fereastra Editorului iar pentru compilarea programului și încărcarea acestuia în memoria microcontrolerului se execută comanda ***Build and Run.*** Această comandă va rula programul în modul de depanare (Debug). Indicarea acestui mod de operare se face printr-o sageată galbenă pe marginea stangă a Editorului. În acest timp se pot accesa regiștrii microcontrolerului din fereastra ***Debugger*** din partea dreaptă. De exemplu se poate vizualiza și schimba starea regiștrilor PORT, PIN și DDR. Pentru a ieși din modul Debug se execută comanda ***Run.***

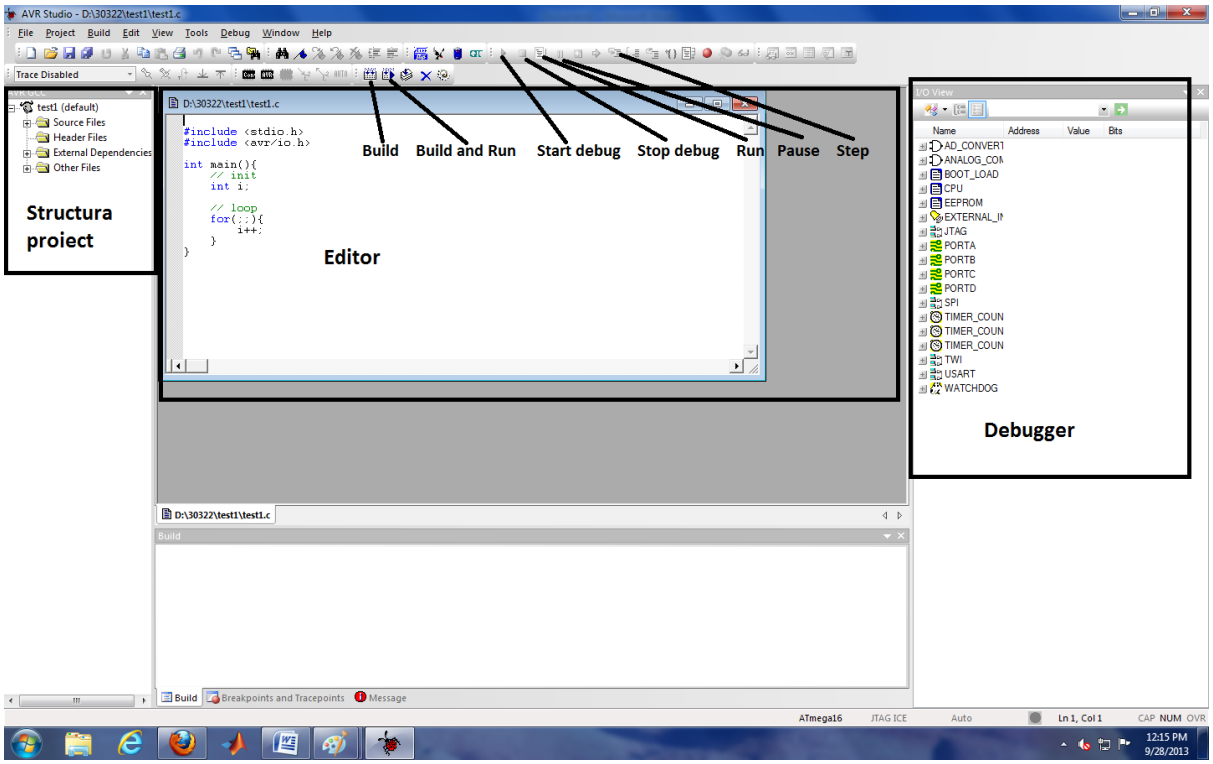


Figura9.5. Interfața programului AVR Studio 4.0

Exerciții:

1. **Setați pinul PA5 ca intrare și aprindeți becul folosind fereastra Debugger și modul Debug.**

2. **Afișați pe display-ul 7-segmente numărul 777 folosind fereastra Debugger și modul Debug.**

3. **Implementați un control de temperatură bipozițional cu histereză de 6 grade. Temperatura va fi afișată cu o zecimală pe display. Pentru a seta temperatura de referință se va poziționa potențiometrul la valoarea dorită. Pentru a vizualiza temperatura de referință se va ține apăsat butonul de pe pin-ul PB7.**

Programul va fi conceput pe structura programelor de la lucrările anterioare. Se va folosi întreruperea de la Timer-ul 0 iar liniile de cod descrise mai jos se vor scrie în funcția ISR a Timer-ului 0. Se va folosi și convertorul analog-numeric pentru a obține valori de la senzorul de temperatură și de la potențiometru.

Astfel că, la fiecare 100 de milisecunde se va converti valoarea analogică de pe canalul 7 (senzorul de temperatură LM35). Rezultatul conversiei se va transforma într-o valoare a temperaturii. Același lucru se va face și pentru canalul 6 de pe care se va obține o valoare pentru temperatura de referință.

Lățimea histerezei se va seta la 6 grade în variabila offset. Apoi, prin intermediul a două structuri `if` se va marca trecerea din stare OFF în stare ON și invers. În funcție de starea butonului PB7 pe display se va afișa temperatura curentă sau temperatura de referință.

```
if (ms==99){

//citește senzorul de temperatura si convertește valoarea in
grade Celsius
    adv=readADC(7);
    Vin=((float)adv*5)/1024;
    tmp= Vin*1000/40;
    tmp=(int)(tmp*10);

//referința - citește potențiometrul si convertește valoarea in
//grade Celsius
    adv=readADC(6);
    Vin=((float)adv*5)/1024;
    ref= Vin*1000/40;
    ref=(int)(ref*10);

    offset=60;

    if(tmp<(ref-offset/2))
        PORTA|=1<<5;      //aprinde becul
    if(tmp>(ref+offset/2))
        PORTA&= ~(1<<5);  //stinge becul

//in funcție de starea butonului, pe display se afișează
//temperatura curenta sau referința
    if(!(PINB & (1<<7)))
        value=ref;
    else
        value=tmp;

     ms=0;
     }
  else ms++;
```

10. Circuite Logice TTL

10.1 Scopul lucrării

În această lucrare sunt prezentate caracteristicile constructiv funcționale ale familiei de circuite integrate TTL, principalii parametri statici și dinamici ai acestei familii.

10.2 Considerații teoretice

10.2.1 Funcționarea circuitului

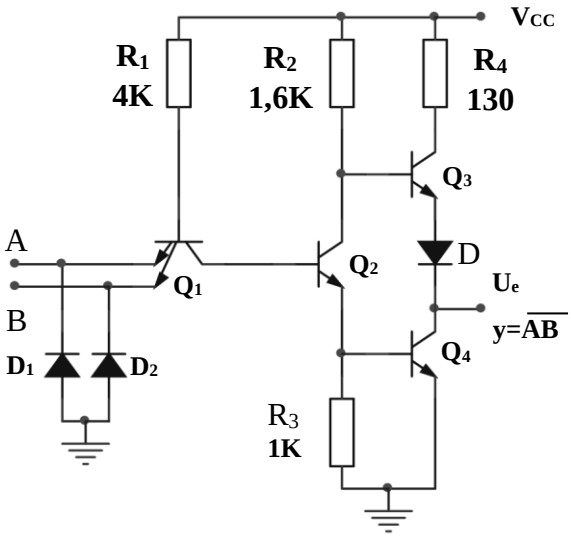


Figura 10.1.

Pentru a arăta funcționarea electrică a porţii TTL fundamentale din Figura10.1, să presupunem mai întâi că una dintre intrări este conectată la masă (nivel logic "0"). Tranzistorul Q_1 se saturează și datorită scăderii potențialului din colectorul său, tranzistorul Q_2 se blochează. Potențialul scăzut din emitorul lui Q_2 determină blocarea tranzistorului Q_4. Tranzistorul Q_3 va conduce, fiind comandat de potențialul ridicat din colectorul tranzistorului Q_2. La ieșire se va obține o valoare ridicată de tensiune, corespunzător nivelului logic "1".

Dacă la ambele intrări se aplică o tensiune corespunzătoare nivelului logic "1", joncțiunile bază-emitor ale tranzistorului Q_1 sunt polarizate invers și tranzistorul lucrează în regiunea activă inversă. În acest caz joncțiunea bază-colector a tranzistorului Q_1și joncțiunile bază-emitor ale tranzistoarelor Q_2și Q_4 formează un lanț de diode polarizate direct prin rezistența R_1 de la plusul sursei de alimentare. În consecință tranzistoarele Q_2și Q_4 se vor satura. În același timp, tranzistorul Q_3 se blochează deoarece baza lui se află la un potențial mai mic decât potențialul emitorului său datorită decalajului de tensiune introdus de dioda D_3. Se obține astfel la ieșire o tensiune egală cu tensiunea de saturație colector-emitor a tranzistorului Q_4, corespunzătoare nivelului logic "0".

Analizând funcționarea porții, din punct de vedere logic, se observă că ea realizează funcția ȘI-NU, adică:

$$y = \overline{AB}$$

10.2.2 Parametrii circuitului

Nivelele logice: V_{ILmax} = 0.8 V, V_{IHmin} = 2 V, V_{OLmax} = 0.4 V, V_{OHmin} = 2.4 V si V_T = 1.3V

Marginile de zgomot: M_L = 0.4V si M_H = 0.4V

Curenţii de intrare şi de ieşire: I_{IH} = 40 μA, I_{IL} = -1,6 mA, I_{OH} = -800 μA si I_{OL} = 16 mA

Factorul de încărcare: FO_L = 10, FO_H = 20 si FO = 10

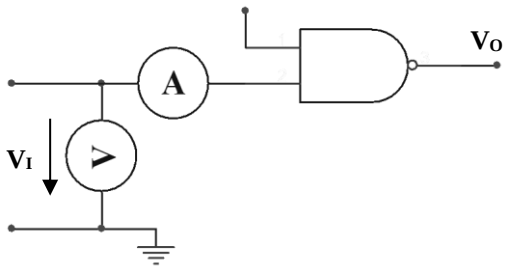


Figura 10.2.

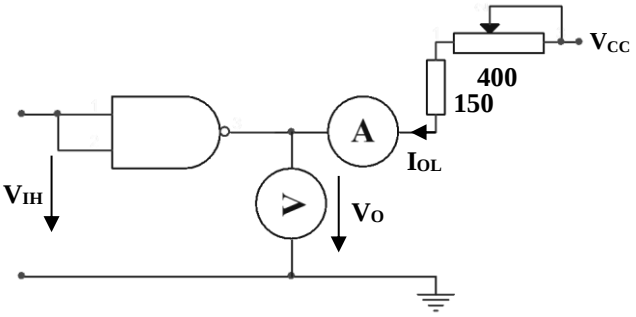


Figura 10.3.

Caracteristica de intrare $I_I=f(V_I)$ se poate ridica cu ajutorul schemei din Figura10.2.

Caracteristica de ieșire $V_{OL}=f(I_{OL})$ se poate ridica cu ajutorul schemei din Figura 10.3 iar caracteristica $V_{OH}=f(I_{OH})$ cu schema din Figura10.4.

Scurtcircuitarea ieșirii la masă poate determina prin tranzistorul Q_3 un curent cuprins între 18 și 55 mA, dacă Q_3, D_3și R_4 funcționează static corect. Acest curent nu este periculos dacă are o durată scurtă. Variația curentului de scurtcircuit cu tensiunea de alimentare se poate urmări cu ajutorul schemei din Figura 10.5.

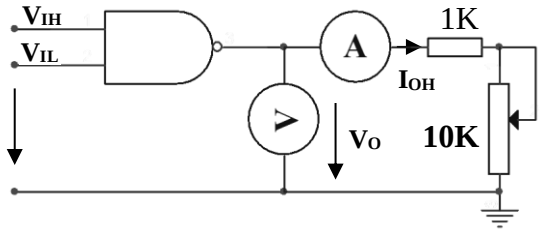


Figura 10.4.

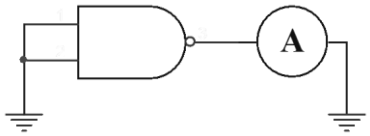


Figura 10.5.

În cadrul familiei de circuite integrate TTL există mai multe serii de circuite, care se deosebesc în principal prin compromisul realizat între puterea disipată pe poartă și timpul de propagare, așa cum rezultă din tabelul de mai jos:

	74	74LS	74S	74ALS	74AS
Puterea disipată/poartă tipic [mW] static	10	2	19	1.2	8.5
Timp de propagare tipic [ns]	10	9.5	3	4	1.5

Caracteristica de transfer a porții ȘI-NU standard se poate ridica cu ajutorul schemei din Figura10.7. Circuitul format din R_1, D_1-D_4, conectat la ieșirea porții simulează o impedanță echivalentă cu 10 sarcini TTL. Diodele sunt de tipul 1N4148 iar C_1 include capacitățile de ieșire a sondelor și ale sistemului de conectare.

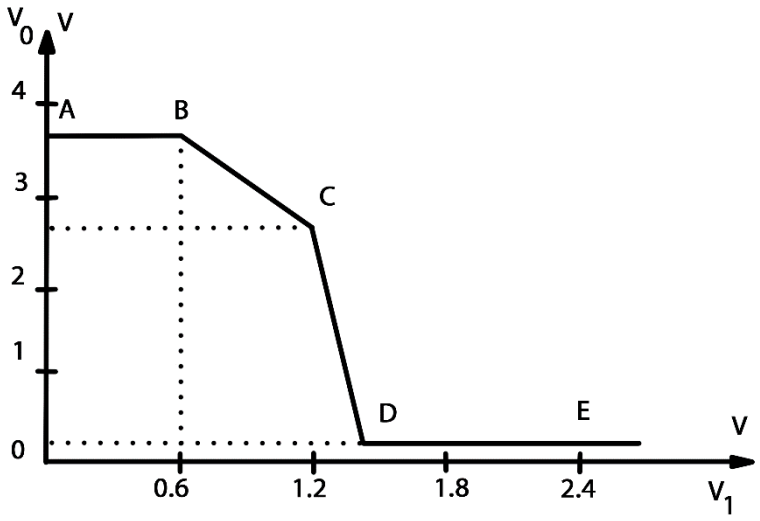


Figura10.6.

Caracteristicile dinamice ale circuitelor TTL se pot determina cu ajutorul circuitului din Figura10.8, care simulează încărcarea unei porți cu 10 sarcini TTL. Timpii de creștere t_r și de cădere t_f au valori tipice de 8ns. Timpul de propagare are următoarele valori tipice: t_{pHL}=8ns, t_{pLH}= 12ns și t_p=10ns.

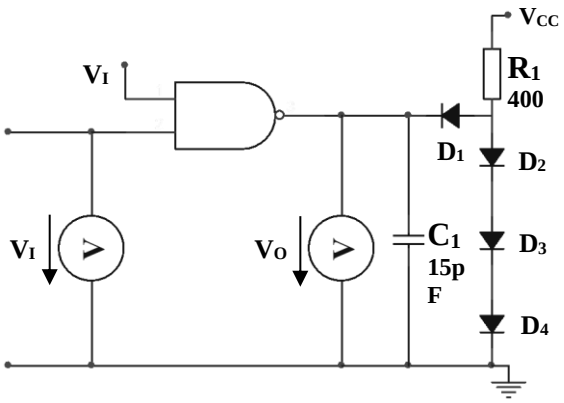


Figura 10.7.

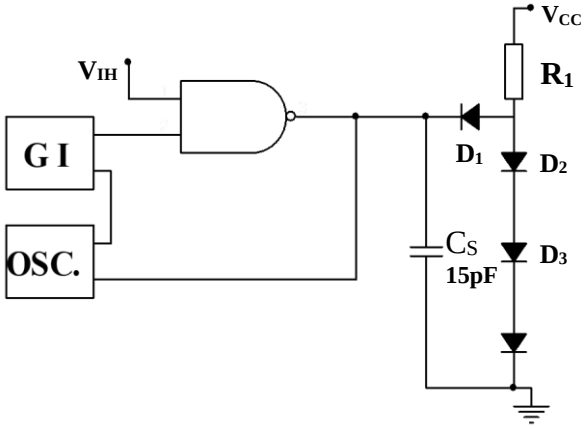


Figura 10.8.

10.3 Mersul lucrării

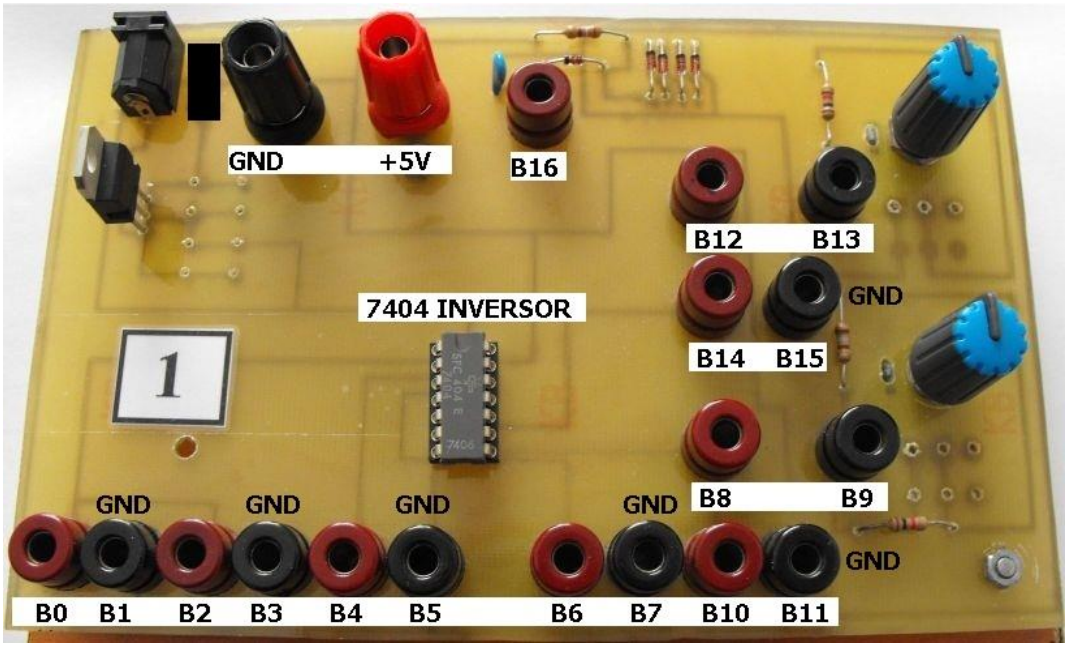


Figura 10.9. Stand de testare circuit logic inversor 74LS04

Elementele de care avem nevoie pentru a testa caracteristicile circuitelor TTL folosind acest stand sunt:

- circuit integrat TTL 7404 (poartă inversoare)
- sursă de curent continuu (0 – 5 V)
- voltmetru de curent continuu – 2 buc
- ampermetru de curent continuu
- rezistență variabilă
- osciloscop
- generator de semnal

Modalitate de lucru :

Înainte de a începe acest laborator este foarte important de a ne aminti faptul că un voltmetru se conectează în paralel cu mărimea de măsurat (față de masă), iar un ampermetrul se înseriază în circuitul în care se dorește măsurarea curentului.Dacă folosim un multimetru va trebui să fim atenți la configurația acestuia, adică dacă sondele sunt conectate pentru a măsura tensiune (voltmetru) sau pentru a măsura curent (ampermetru).

Folosind schemele electronice din îndrumătorFigura10.2,10.3,10.4, 10.5, 10.7 se realizează următoarele experimente:

Experiment 1:

Experimentul 1 corespunde montajului din Figura10.7 din îndrumător.

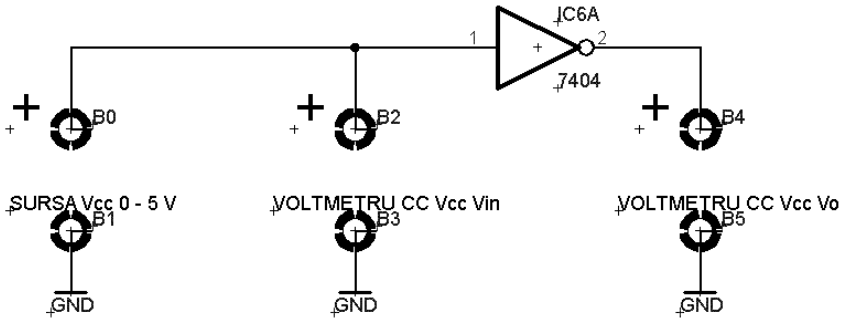


Figura 10.10. Experiment 1 - caracteristica de transfer a porții inversoare

Pentru a realiza acest experiment se procedează astfel :

- la bornele B0 și B1 se conectează sursa de tensiune continuă, borna pozitivă a sursei la borna B0 (a standului) și borna GND a sursei de tensiune continuă la borna B1 (GND-ul standului), apoi conectăm un voltmetru între bornele B2 (intrarea în poarta inversoare) și B3 și un alt voltmetru între bornele B4 (ieșirea porții inversoare) și B5;

- de la sursă, modificăm tensiunea aplicată pe intrarea porții inversoare în intervalul 0 - 5 V pentru a observa ce se întâmplă cu ieșirea;

- se realizează un tabel cu valorile citite de pe cele două voltmetre V_{in}și V_{out}

- se realizează caracteristica de transfer a porții inversoare (asemănătoare cu Figura10.6 din îndrumător), V_{out} în funcție de V_{in}și se interpretează;

Experiment 2:

Experimentul 2 corespunde montajului din Figura 10.5 din îndrumător.

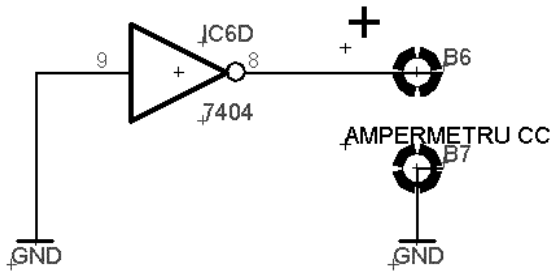


Figura 10.11. Experiment 2 – scurtcircuit la masă

La acest experiment se procedează astfel :

- realizarea scurtcircuitului la masă necesită ca valoarea ieșirii să fie pozitivă, deci la realizarea standului intrarea unei porți inversoare a fost conectată pe cablaj la GND ;

- se conectează borna pozitivă a ampermetrului la borna B6 a standului (ieșirea porții inversoare)și borna negativă la GND-ul standului, în acest caz la borna B7 ;

- se citește / se notează valoarea curentului și se interpretează rezultatul ;

Experiment 3:

Experimentul 3 corespunde montajului din Figura 10.4din îndrumător.

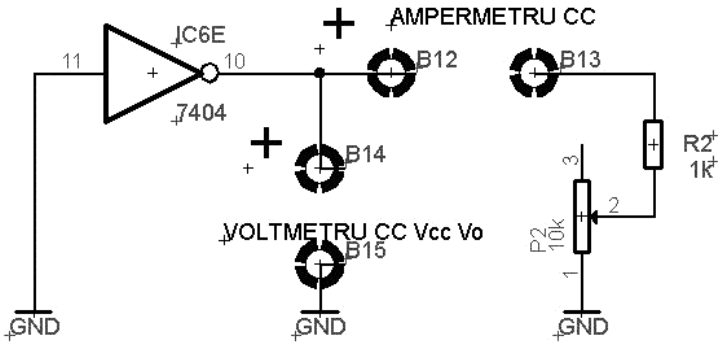


Figura 10.12. Experiment 3 – măsurarea curentului spremasă

Acest experiment este asemănător cu cel precedent, însă aici scopul nu este de a testa dacă poarta funcționează static corect, ci de a măsura valoarea maximă de curent ce poate fi "trasă" din poartă cu respectarea nivelului "1" logic la ieșire.

Pentru a realiza experimentul se procedează astfel :

- masurătorile necesită ca valoarea ieșirii să fie pozitivă, deci la realizarea standului intrarea unei porți inversoare a fost conectată pe cablaj la GND ;

- se conectează un voltmetru între bornele B14 și B15 (GND), apoi între B12 și B13 se conectează un ampermetru;

- între borna B13 și GND se conectează o rezistență variabilă ;

- pentru început valoarea rezistenței este maximă posibil;

- la pasul următor începem să micșorăm rezistența și observăm cum valoarea de tensiune citită de pe voltmetru scade, iar valoarea curentului citit pe ampermetru crește;

- continuăm cu scăderea rezistenței până ajungem la limita de 2.4 V (valoare minimă garantată la ieșire în starea "1" logic), notăm curentul și interpretăm valoarea obținută.

Experiment 4:

Experimentul 4 corespunde montajului din Figura 10.3 din îndrumător. Acesta urmează aceiași pași ca și experimentul 3, deosebirea aici este că pentru a putea măsura curentul spre VCC, trebuie să avem "0" logic la ieșirea acestei porți. Acest lucru se obține foarte ușor înseriind două porți inversoare. Dacă pe intrarea primei porți avem "0" logic, la ieșirea acesteia v-om avea "1" logic, care este intrare pentru cea de-a doua poartă și rezultă la ieșirea celei de-a doua porți "0" logic.

Scopul este același, de a măsura valoarea maximă de curent ce poate fi asorbit de poartă cu respectarea nivelului "0" logic la ieșire.

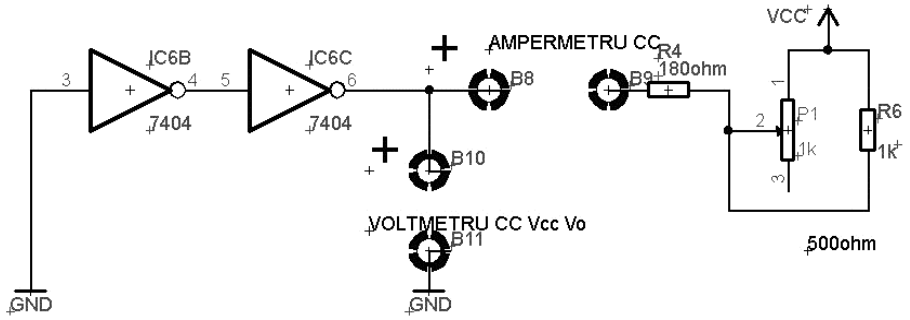


Figura10.13. Experiment 4 – măsurarea curentului de la VCC

În continuare:

- înserierea celor două porți a fost realizată prin construcție, la proiectarea cablajului;

- se conectează un voltmetru între bornele B10 și B11 (GND), apoi între B8 și B9 se conectează un ampermetru;

- între borna B9 și VCC se conectează o rezistență variabilă ;

- pentru început valoarea rezistenței este maximă;

- la pasul următor începem să scădem rezistența și observăm cum valoarea de tensiune citită de pe voltmetru crește, iar valoarea curentului citit pe ampermetru creşte;

- continuăm cu scăderea rezistenței până ajungem la limita de 0.4 V (valoare maximă garantată la ieșire în starea “0” logic), notăm curentul și interpretăm valoarea obținută;

Experiment 5:

Experimentul 5 corespunde montajului din Figura 10.8din îndrumător.Acest experiment va folosi o configurație din experimentul 1. Generatorul de impulsuri se conectează la intrarea porții inversoare și pe un canal al osciloscopului, iar ieșirea porți se v-a conecta pe al doilea canal al osciloscopului.

Acest experiment urmărește evidențierea timpilor de propagare, din “1” → “0” (aprox. 8ns) și din “0” → “1” (aprox. 12ns). Timpii de propagare sunt influențați de numărul sarcinilor din circuit.

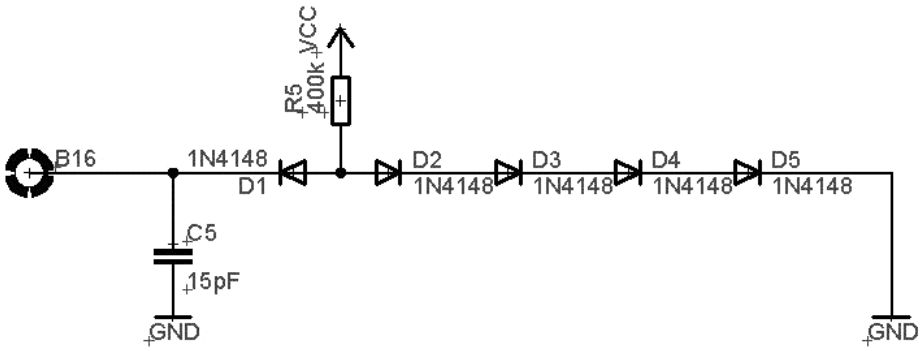


Figura10.14. Experiment 5 – simulare 10 sarcini TTL

11. Circuite Logice cu Colector Deschis

11.1 Scopul lucrării

Se vor studia circuitele logice cu colector deschis si se vor analiza posibilitățile de realizare a magistralelor utilizând funcția cablată.

11.2 Considerații teoretice

Pentru a cupla în paralel mai multe porți se utilizează circuite de tip colector în gol sau circuite cu trei stări.

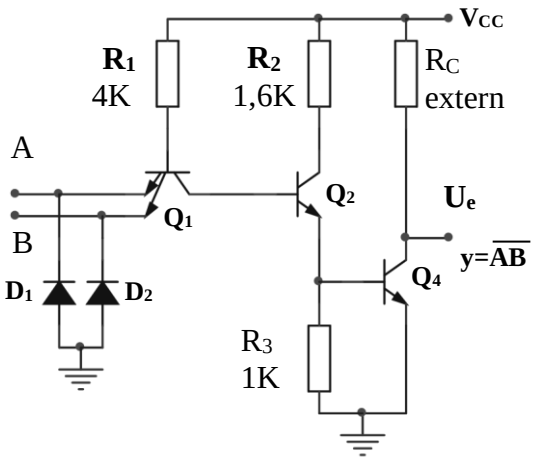


Figura 11.1.

În schema electrică a porții TTL cu colector în gol se păstrează în întregime etajul de intrare și separatorul de nivel utilizate în construcția porții fundamentale. S-a modificat însă etajul de ieșire din care s-a păstrat numai tranzistorul Q_4 (Figura11.1). În acest caz, colectoarele tranzistoarelor Q_4 aparținând diferitelor circuite pot fi legate împreună, punctul comun fiind conectat printr-o rezistență la sursă.

Rezistența comună nu este inclusă în structura integrată, ea fiind calculată de proiectantul schemei în funcție de numărul porților conectate împreună (n) și de numărul porților TTL care trebuie comandate de către această ieșire comună (N).

Calculul rezistenței R_c se face în funcție de nivelul logic de la ieșirea comună, de curentul debitat de porțile conectate în paralel și de curenții absorbiți de porțile comandate.

În cazul nivelului 1 logic la ieșire va rezulta:

$$R_{c\max} = \frac{V_{cc\min} - V_{OH\min}}{n \cdot I_{OH\max} + N \cdot I_{IH\max}},$$

iar pentru nivelul 0 logic la ieșire:

$$R_{c\min} = \frac{V_{cc\max} - V_{OL\max}}{I_{OL\max} + (n-1) I_{OH\max} - N \cdot I_{IL\max}},$$

Valorile rezistențelor de sarcină se calculează în următoarele condiții:

V_{cc}=5V±5%, I_{OH}=250μA, I_{OL}=16mA, I_{IL}=1.6mA, I_{IH}=40μA, V_{OHmin}=2.4V, V_{OLmax}=0.4V.

Având de exemplu de realizat funcția implementată în Figura11.2 este necesară o logică pe trei nivele ceea ce duce la o întârziere mare. Aceeași funcție se poate implementa cu circuite cu colectorul în gol. Funcția realizată poartă numele de ȘI-cablat.

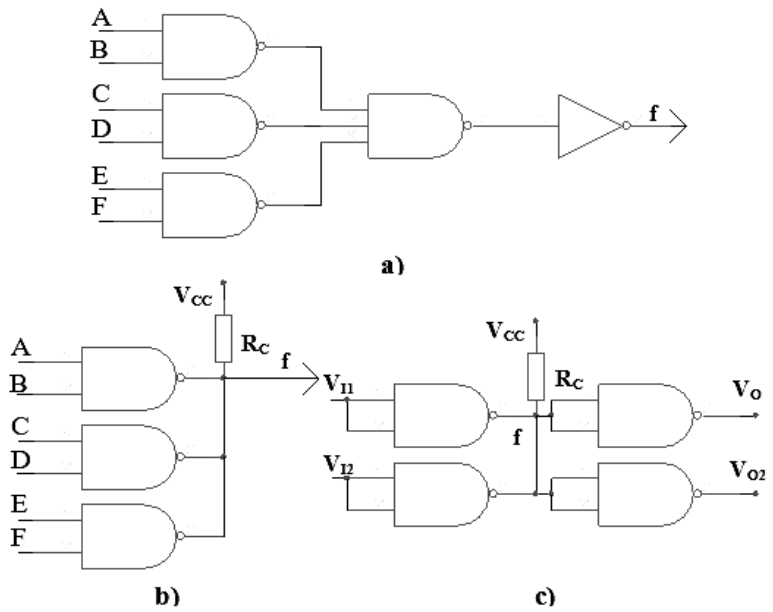


Figura 11.2.

$$f(A,\ldots,F) = \overline{AB} \cdot \overline{CD} \cdot \overline{EF} = \overline{AB + CD + EF}\,.$$

Circuitul realizează funcția ȘI între ieșirile porților ȘI-NU. Întregul circuit realizează funcția ȘI-SAU-NU pentru grupul de variabile de la intrarea porților ȘI-NU.

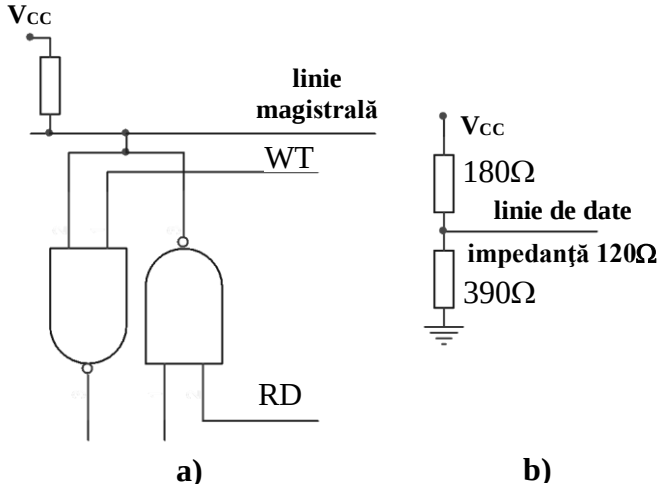


Figura 11.3.

În construcția magistralelor se utilizează pe scară largă atât circuitele cu colector în gol cât și cele cu trei stări. Unui circuit legat la o magistrală i se atribuie, de regulă, în sistem o funcționare atât de emițător, cât și de receptor. În acest caz, intrările de comandă trebuie să permită atât citirea unui cuvânt de pe magistrală, cât și scrierea unui cuvânt pe magistrală. Printr-un semnal de comandă RD cuvântul este introdus pe magistrala, iar printr-un semnal de comandă WR cuvântul este citit de pe magistrală (Figura 11.3 (a)). Dacă la magistrală sunt cuplate numai circuite TTL, în locul rezistențelor de ridicare, se pot utiliza terminatori de magistrală (grup de rezistențe montate la extremitățile traseelor magistralei pentru adaptarea împotriva reflectiilor) Figura 11.3 (b).

11.3 Mersul lucrării

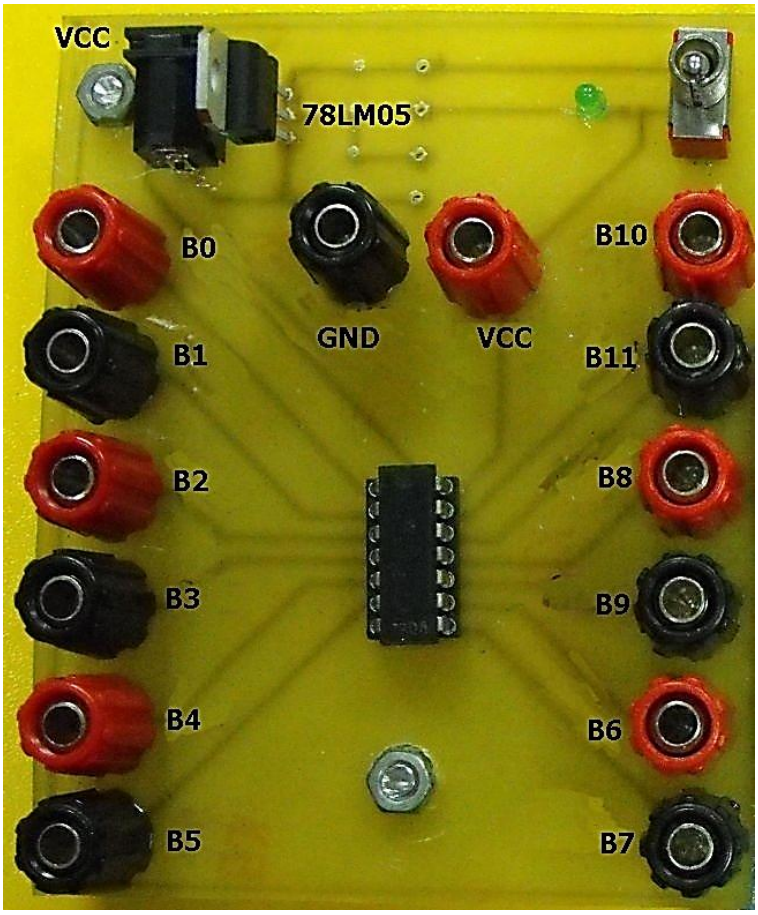


Figura 11.4. Stand de testare circuite cu colector în gol

Elementele de care avem nevoie pentru a studia caracteristicile circuitelor cu colector în gol:

- circuit CDB (poartă inversoare cu colector în gol)
- RC (rezistență de colector – variabilă)
- voltmetru

Modalitate de lucru :

După studierea lucrării din îndrumător se realizează schema electrică de mai jos utilizând un circuit CDB 405 sau CDB 406. Bornele de culoare roșie (B0, B2, B4, B6, B8, B10) sunt intrări în porțile logice ale circuitului CDB 405, iar bornele de culoare neagră (B1, B3, B5, B7, B9, B11) sunt ieșirile acestora.

Rezistența R_c se v-a conecta între punctul de intersecție al porților conectate împreună (n) și VCC.

Se stabilește experimental valoarea $R_{c\ max}$ pentru nivelul "1" logic la ieșire și în mod similar $R_{c\ min}$ pentru "0" logic la ieșire, apoi se compară cu valorile calculate după formulele din îndrumător.

Se studiază modul de funcționare a porții cu colector în gol din Figura 11.1 folosindu-se circuitul din Figura 11.5. Pentru 1 logic la ieșire se mărește R_c până V_{OH} scade sub 2.4V. Se notează valoarea R_{cmax} și se compară cu cea calculată. Pentru 0 logic la ieșire se micșorează $R_c$până

când V_{OL} crește peste 0.4V, se notează valoarea R_{cmin} și se compară cu cea calculată. Se repetă operațiile de mai sus pentu diferite încărcări.

Se pot încerca și alte combinații de realizare a funcției cablate ȘI-NU. Cel mai des, circuitele cu colectorul în gol le întâlnim acolo unde avem de realizat o magistrală de date.

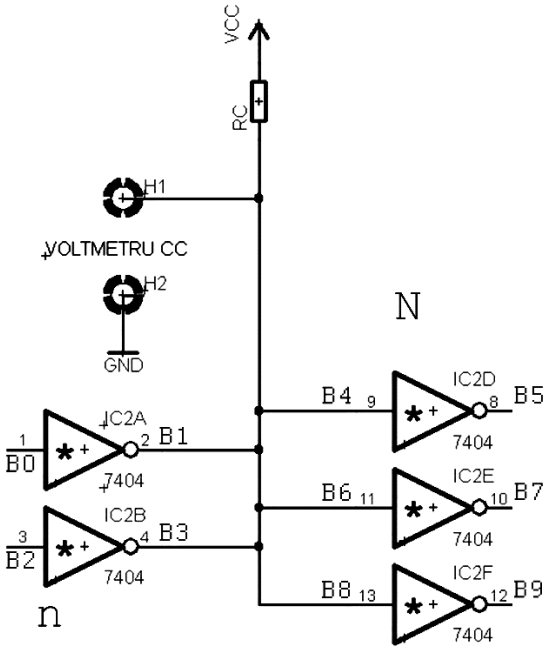


Figura 11.5. Funcția cablată ȘI-NU

12. Circuite Basculante Astabile (CBA)

12.1 CBA cu componente discrete

Numite și multivibratoare, circuitele basculante astabile (CBA) sunt caracterizate din punct de vedere al mărimilor electrice, prin existența a două stări limită distincte, ambele instabile. Trecerea dintr-o stare în alta se execută într-un timp foarte scurt și apare ca o variație bruscă a mărimilor electrice. Acest proces se numește basculare și se declanșează (fără semnale aplicate din exterior) la momentele de timp determinate de parametrii circuitului. Circuitul basculant astabil este de fapt un oscilator care produce semnal dreptunghiular la ieșirea sa. Durata impulsurilor și perioada de repetiție sunt determinate de valorile unor elemente de circuit.

La fel ca circuitele basculante monostabile, și circuitele astabile pot prezenta reacția pozitivă prin cuplajul colector-bază sau cuplajul prin emitor.

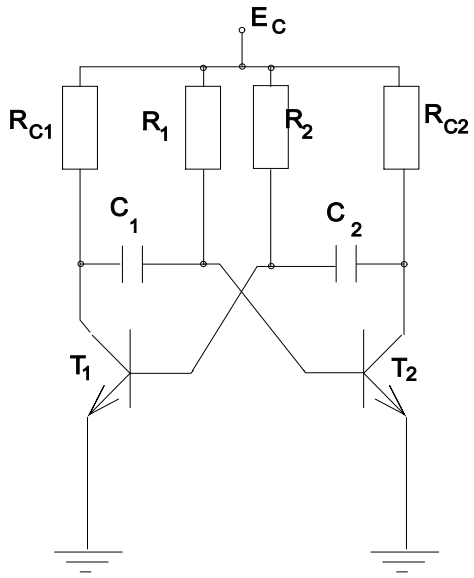


Figura 12.1.

CBA cu cuplaj colector - bază are schema ilustrată în Figura 12.1. Faptul că rezistențele R_1 și R_2 sunt legate la sursa $+E_C$ asigură ca nici-unul dintre tranzistoarele T_1 și T_2, în regim staționar, să nu rămână blocat nedefinit.

Funcționarea circuitului se descrie într-un mod similar funcționării CBM. Astfel să presupunem că la momentul curent suntem într-o stare în care tranzistorul T_1 este blocat și tranzistorul T_2 conduce; aceasta implică ca potențialul pe capacitatea C_2 să fie în scădere, condensatorul descărcându-se prin circuitul E_C, R_2, C_2 și tranzistorul conductor T_2. Pe măsura scăderii curentului de descărcare care trece prin R_2, crește potențialul în baza tranzistorului T_1, iar când potențialul U_{B1} va depăși tensiunea de prag de deschidere a tranzistorului T_1, acesta devine conductor, iar potențialul din colectorul său va începe să scadă puternic. Aceasta va duce la scăderea accentuată a potențialului bazei lui T_2, ce va determina ieșirea sa din saturare. Bucla de reacție pozitivă va impune un proces în avalanșă, ce va duce în final la blocarea tranzistorului T_2 și circuitul va trece astfel în a doua stare instabilă. Tranzistorul T_1 fiind în conducție, acum se va descărca capacitatea C_1, prin circuitul E_C, R_1, C_1 și T_1. Starea aceasta va dura până când tensiunea din baza lui T_2 va depăși valoarea de prag de deschidere și tranzistorul

T_2 se va deschide, restabilindu-se bucla de reacție pozitivă, ce va declanșa bascularea în cealaltă stare instabilă.

În prima stare instabilă, pe lângă descărcarea capacității C_2, are loc încărcarea capacității C_1, prin circuitul E_C, R_{C1}, C_1 și rezistența de intrare a tranzistorului conductor T_2. Constanta de timp de încărcare este $\tau_{1i} = R_{C1} \cdot C_1$. Pentru a doua stare, a doua semiperioadă a oscilațiilor, se va încărca capacitatea C_2 prin lanțul E_C, R_{C2}, C_2 și tranzistorul saturat T_1, cu o constantă de timp $\tau_{2i} = R_{C2} \cdot C_2$.

Duratele celor două perioade instabile se determină după formule asemănătoare cu cea pentru durata stării instabile a CBM:

$$T_{si1} = 0{,}69 \cdot R_1 C_1$$

$$T_{si2} = 0{,}69 \cdot R_2 C_2$$

Durata de repetiție a semnalului periodic va fi:

$$T = T_{si1} + T_{si2}$$

Pentru un circuit astabil simetric, pentru care $R_1 = R_2 = R$, iar $C_1 = C_2 = C$, se va genera un semnal rectangular (o tensiune rectangulară) în colectoarele tranzistoarelor, având o perioadă $T = 1{,}4 \cdot RC$ și cu un factor de umplere 1/2.

12.2 CBA realizate cu circuite TTL

În Figura12.2 (a) se prezintă schema electrică a unui circuit astabil realizat cu două inversoare TTL.

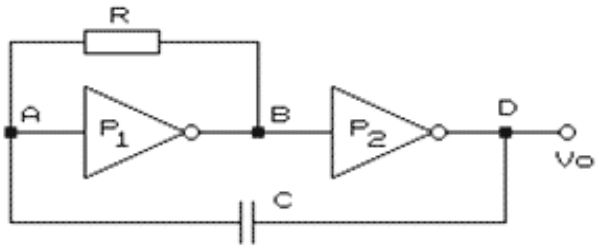


Figura 12.2. (a)

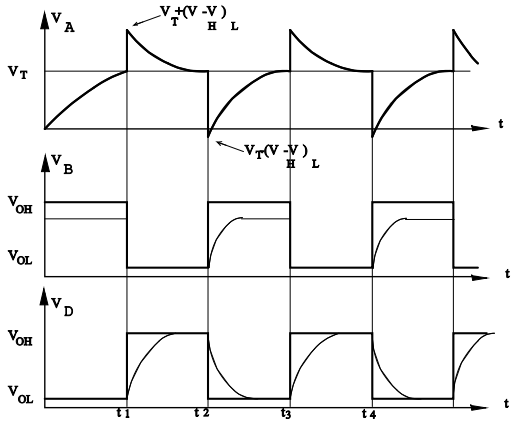


Figura12.2. (b)

Principiul de funcționare al acestei scheme este următorul: dacă la momentul $t<t_1$ la intrarea porții P_1 în punctul A avem semnal logic 0, la ieșirea acestei porți (B) vom avea 1 logic, iar la ieșirea porții P_2 vom avea 0 logic. În această situație potențialul punctului A tinde să crească spre V_H, condensatorul C încărcându-se prin rezistența R. La momentul $t=t_1$ când $V_A=V_T$, V_T= 1.5 V reprezentând potențialul de prag al porții P_1, ieșirea porții P_1 comută din 1 logic în 0 logic, ceea ce determină comutarea ieșirii porții P_2. Saltul de tensiune din D de la V_L la V_H se transmite prin capacitatea C în punctul A. În continuare tensiunea din punctul A scade exponențial spre valoarea tensiunii V_L de la ieșirea porții P_1, pe măsură ce are loc descărcarea capacității C.La momentul $t=t_2$, $V_A=V_T$ ceea ce determină din nou comutarea celor două porți. Saltul de tensiune din punctul D se transmite prin capacitatea C în A. În continuare capacitatea se va încărca prin rezistența R, iar potențialul din A va crește. Fenomenul continuă atâta timp cât circuitul este sub tensiune. În Figura12.2 (b) s-au reprodus diagramele de timp în funcționarea circuitului. Fronturile semnalului de ieșire V_0 sunt afectate de valoarea capacității C (linia punctată).Durata celor două perioade $T_1=t_2-t_1$ si $T_2=t_3-t_2$ se pot determina din relația care exprimă variația în timp a tensiunii din A:

$$V_A = V_A(\infty) + [V_A(0) - V_A(\infty)] \cdot e^{-t/RC}$$

$$t = RC \cdot \ln \frac{V_A(0) - V_A(\infty)}{V_A(t) - V_A(\infty)}$$

unde pentru $t=T_1$ avem:

$$V_A(\infty) = V_{0L}; V_A(0) = V_T + (V_{0H} - V_L);$$

$$V_A(T_1) = V_T$$

Rezultă deci:

$$T_1 = RC \cdot \ln \frac{V_T + V_{0H} - 2 \cdot V_{0L}}{V_T - V_{0L}}$$

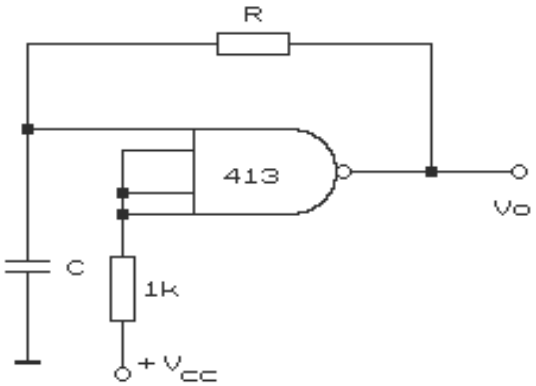
Pentru T_2 avem:

$$V_A(\infty) = V_{0H}; V_A(0) = V_T - (V_{0H} - V_{0L}); V_A(T_2) = V_T$$

de unde:

$$T_2 = RC \cdot \ln \frac{V_T + V_{0L} - 2 \cdot V_{0H}}{V_T - V_{0H}}$$

Un CBA poate fi realizat cu ușurință utilizând un circuit trigger Schmitt (Figura12.3). Condensatorul C se încarcă și se descarcă prin rezistența R tinzând spre nivelele tensiunii de ieșire, dar la atingerea pragurilor de basculare V_{T1} si V_{T2} circuitul comută dintr-o stare în alta. Pentru determinarea duratelor T_1 si T_2 se procedează ca și pentru astabilul din Figura12.2:



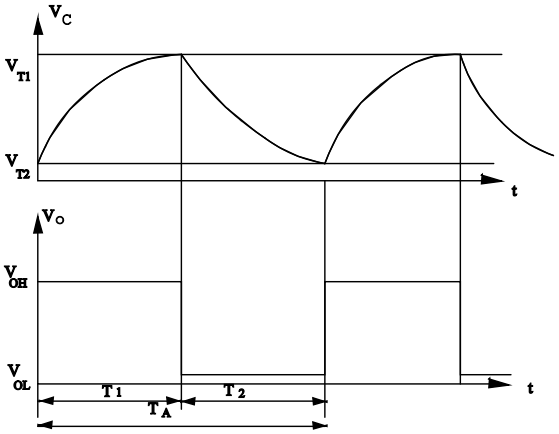


Figura12.3. (a) ***Figura12.3. (b)***

Pentru T_1 avem:

$$V_C(0) = V_{T_2}; V_C(\infty) = V_{0H}; V_C(T_1) = V_T1$$

$$T_1 = RC \cdot \ln \frac{V_{0H} - V_{T_2}}{V_{0H} - V_{T_1}}$$

Pentru T_2:

$$V_C(0) = V_{T_1}; V_C(\infty) = V_{0L}; V_C(T_2) = V_T2$$

$$T_2 = RC \cdot \ln \frac{V_{0L} - V_{T_1}}{V_{0L} - V_{T_2}}$$

Deci perioada de oscilație este egală cu:

$$T_A = RC \cdot \ln \frac{(V_{0H} - V_{T_2}) \cdot (V_{0L} - V_{T_1})}{(V_{0H} - V_{T_1}) \cdot (V_{0L} - V_{T_2})}$$

Pentru CDB413E:

$$V_{T_1} = 1{,}85V;\ V_{T_2} = 0{,}85V;\ V_{0L} = 0{,}1V;\ V_{0H} = 3{,}5V$$

Cu aceste valori se obține:

$$T_1 = 0{,}86 \cdot RC; T_2 = 0{,}83 \cdot RC; T_A = 1{,}69 \cdot RC.$$

O folosire corectă a triggerului Schmitt impune cunoașterea valorilor curenților și a tensiunilor de intrare și de ieșire. Pentru a satisface condițiile de nivele de tensiune și de curent pe intrare se recomandă valoarea rezistenței:

$$R = 390\Omega.$$

12.3 CBA realizate cu circuite CMOS

În Figura12.4 (a) este prezentată cea mai simplă schemă de astabil cu porți CMOS, care constă din două inversoare, un rezistor și un condensator.

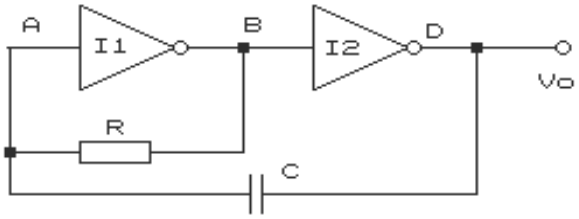


Figura 12.4. (a)

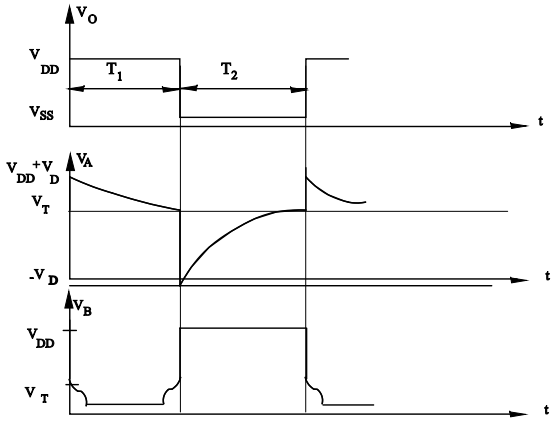


Figura 12.4. (b)

Dacă intrarea inversorului I_1 este în "1" logic,ieșirea acestuia și intrarea inversorului I_2 vor fi în "0" logic iar ieșirea lui I_2 în "1" logic. În această situație, condensatorul C se va descărca prin rezistența R. Curentul de descărcare este furnizat de tranzistorul cu canal n din etajul de ieșire al inversorului I_1. Atunci când potențialul pe intrarea inversorului I_1 (punctul A) scade sub potențialul de prag, ieșirea inversorului I_1 trece în 1 logic determinând comutarea inversorului I_2 din 1 logic în 0 logic și procesul se reia.

Întrucât intrările circuitelor CMOS de obicei sunt protejate cu circuite de protecție standard cu diode, tensiunea la intrarea inversorului I_1 va fi limitată sus la $V_{DD}+V_D$ și jos la $V_{SS}-V_D$ în care V_D este tensiunea directă pe o diodă din circuitul de protecție (V_D= 0.6V). Considerând V_{SS}=0 se obține pentru duratele T_1 și T_2:

- pentru T_1:

$$V_A(T_1) = V_T$$

$$V_A(\infty) = 0V \; ; V_A(0) = V_{DD} + V_D :$$

$$T_1 = RC \cdot \ln \frac{V_{DD} + V_D}{V_T} \; ;$$

- pentru T_2:

$$V_A(\infty) = V_{DD}\,; V_A(0) = -V_D\,; V_A(T_2) = V_T:$$

$$T_2 = RC \cdot \ln \frac{V_{DD} + V_D}{V_{DD} - V_T}\;;$$

Prin însumarea lui T_1 cu T_2 se obține durata perioadei astabilului:

$$T_A = RC \cdot \ln \frac{(V_{DD} + V_D)^2}{V_T(V_{DD} - V_T)}.$$

Din relația de mai sus rezultă că perioada astabilului depinde de tensiunea de alimentare și de tensiunea de prag. De asemenea, timpii de propagare ai inversoarelor folosite se adaugă la duratele T_1 și T_2. Întrucât acești timpi depind de tensiunea de alimentare, perioada astabilului la frecvențe înalte va depinde, și pe această cale, de tensiunea de alimentare.

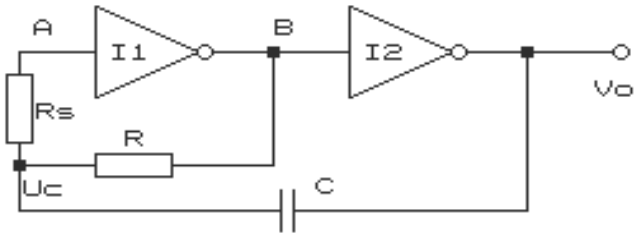


Figura12.5. (a)

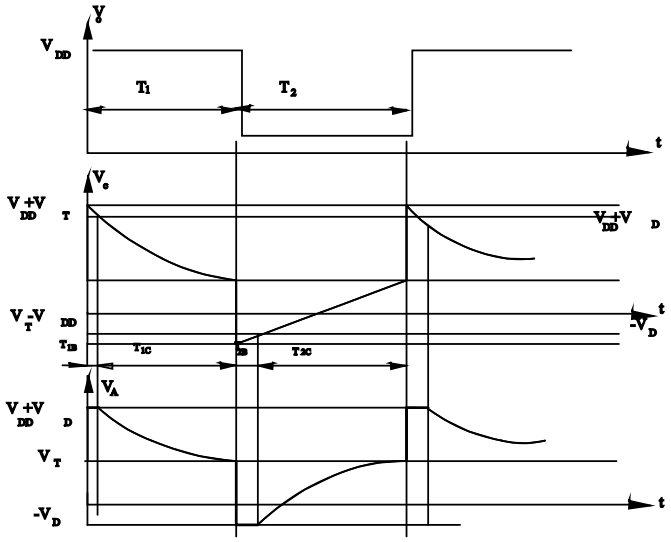


Figura12.5. (b)

Prin conectarea în serie cu intrarea inversorului I_1 a unei rezistențe R_s (Figura12.5 (a)) acest astabil va avea perioada de oscilație independentă de variațiile tensiunii de alimentare. Salturile de tensiune pe condensatorul C sunt mai mari decât în schema precedentă nefiind limitate de circuitele de protecție ale inversorului. Aceasta duce la reducerea efectului variațiilor tensiunii de prag și a caracteristicilor variabile ale circuitului de intrare.

Pentru determinarea perioadei de oscilație se obțin relațiile:

$$T_{1B} = \frac{R \cdot R_S}{R + R_S} \cdot C \cdot \ln \frac{R_S (V_{DD} + V_T) + R(V_T - V_D)}{R_S (V_{DD} + V_D)};$$

$$T_{1C} = R\,C \cdot \ln \frac{V_{DD} + V_D}{V_T};$$

Valoarea rezistenței R_s se recomandă să se aleagă de 2 până la 10 ori mai mare decât valoarea rezistenței R.

$$T_{1B} = \frac{R \cdot R_S}{R + R_S} \cdot C \cdot \ln \frac{R_S (2V_{DD} - V_T) + R(V_{DD} - V_T - V_D)}{R_S (V_{DD} + V_D)};$$

$$T_{1C} = R\,C \cdot \ln \frac{V_{DD} + V_D}{V_{DD} - V_T}.$$

CBA prezentate au dezavantajul că perioada de oscilație este dependentă de tensiunea de alimentare, temperatură, tipul și seria circuitelor logice utilizate etc, ceea ce determină o modificare în limite destul de largi (20-30%) a frecvenței de oscilație.

12.4 CBA cu cuarț și porți logice

Pentru a obține CBA cu o stabilitate ridicată a frecvenței de oscilație se recomandă utilizarea unor astabile realizate cu porți și cristale de cuarț. Oscilatoarele cu cuarț realizate cu circuite CMOS asigură, în plus, avantajul consumului de putere redus și a stabilității frecvenței pe o gamă largă a tensiunii de alimentare.

Oscilatorul fundamental conține un amplificator și o rețea de reacție. Pentru amorsarea oscilației, produsul dintre câștigul amplificatorului și atenuarea rețelei de reacție trebuie să fie supraunitar, iar defazajul total introdus de amplificator și rețeaua de reacție trebuie să fie un multiplu de 360° (criteriul lui Barkhausen).

Schema electrică echivalentă a unui cristal de cuarț conține două componente echivalente înseriate: una rezistivă (R_e) și una reactivă (X_e). Comportarea cuarțului devine pur rezistivă (X_e=0) pentru două valori ale frecvenței, definite ca frecvența de rezonanță (f_r) și frecvența de antirezonanță (f_a). Oscilatoarele cu rezonanță serie sunt proiectate să oscileze la, sau aproape de frecvența de rezonanță. Oscilatorul cu rezonanță paralel oscilează la frecvențe cuprinse între f_r și f_a, în funcție de valoarea încărcării capacitive C_L a cuarțului. Întrucât circuitele cu rezonanță paralel au performanțe bune când lucrează cu amplificatoare cu impedanță mare de intrare, acestea sunt cele mai răspândite pentru oscilatoare cu cuarț care utilizează circuite CMOS.

Circuitul prezentat în Figura12.6, denumit și rețea π cu cuarț, este indicat a fi utilizat împreună cu un amplificator care asigură un defazaj de 180°. Rețeaua π este proiectată pentru a asigura defazajul suplimentar de 180° necesar îndeplinirii condiției de oscilație. Defazajul introdus de către această rețea de rezistențe este extrem de sensibil la variațiile frecvenței, condiție necesară unei oscilații stabile. Valorile capacităților C_T și C_S se pot calcula cu formulele:

$$C_T = 4 \cdot C_L (1 - 5 \cdot f \cdot R_e \cdot C_L);$$

$$C_S = 4 \cdot C_L (3 + 5 \cdot f \cdot R_e \cdot C_L).$$

Luând în considerare capacitatea de intrare a amplificatorului, valoarea capacității C_S din rețeaua de reacție trebuie să fie cu aproximativ 7pF mai mică decât valoarea calculată.Valoarea capacității de ieșire a amplificatorului trebuie să poată fi modificată, astfel încât să compenseze capacitățile parazite ale montajului.

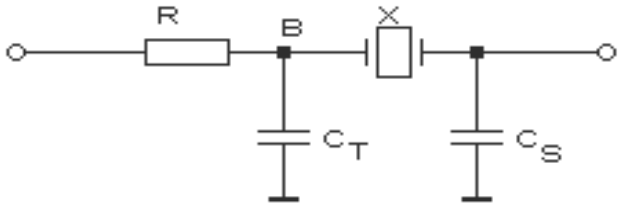


Figura12.6.

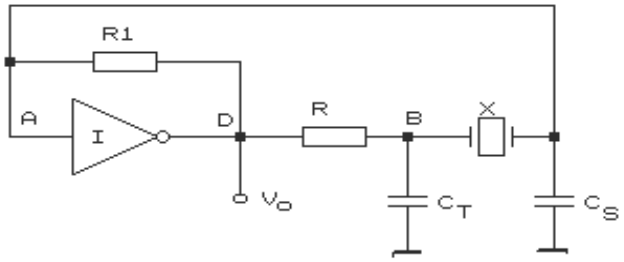


Figura12.7.

Circuitul prezentat în Figura12.7 este calculat să oscileze cu o frecvență de 32,768 kHz pentru o încărcare capacitivă de 10pF, rezistența echivalentă a cuarțului fiind de 50kΩ. Valorile calculate pentru capacitățile C_T și C_S sunt de 43pF și respectiv 13pF. Valoarea maximă a rezistenței R este de 1MΩ. Valori ridicate ale acestei rezistențe vor determina o îmbunătățire a stabilității frecvenței dar această îmbunătățire nu este semnificativă. Rezistența R_1 din rețeaua amplificatorului nu trebuie să încarce rețeaua de reacție. O valoare de 15 MΩ este suficientă.

Cu ajutorul oscilatoarelor CMOS cu cuarț se pot obține frecvențe până la 10kHz pentru V_{DD}=15V și până la 4MHz pentru V_{DD}=5V. Frecvența minimă de operare depinde de rezistența echivalentă a cuarțului care crește rapid la frecvențe joase. Se poate coboară la frecvențe joase utilizând divizoare de frecvență.

12.5 Desfășurarea lucrării

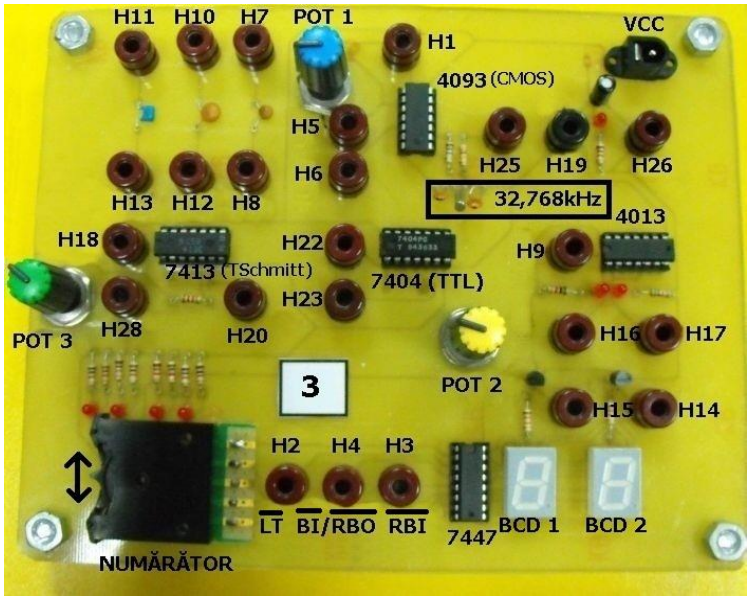


Figura12.8. Stand de testare circuite basculante astabile

Elementele de care avem nevoie pentru a studia câteva circuite basculante astabile sunt următoarele :

- circuit CMOS 4093 (ȘI-NU cu două intrări)
- circuit trigger Schmitt 7413 (ȘI-NU cu 3 intrări)
- circuit TTL 7404 (inversor)
- cristal de cuarț cu frecvența 32,768 kHz
- osciloscop

Modalitate de lucru :

Utilizând acest stand de testare vom putea construi și studia 4 tipuri de circuite basculante astabile, după cum urmează:

1.Circuit basculant astabil realizat cu circuite logice TTL ;
2.Circuit basculant astabil realizat cu circuite logice trigger Schmitt ;
3.Circuit basculant astabil realizat cu circuite logice CMOS ;
4.Circuit basculant astabil realizat cu cristal de cuarț ;

Pentru a realiza primele 3 tipuri de circuite basculante astabile avem nevoie de conectarea unui condensator “undeva” în schemă. Pe standul de testare sunt montate 3 condensatoare astfel :

- C1 între bornele H11 și H13 cu valoarea de 0,9uF ;
- C2 între bornele H10 și H12 cu valoarea de 100mF ;
- C3 între bornele H7 și H8 cu valoarea de 14nF ;

12.5.1 Circuit basculant astabil realizat cu circuite logice TTL

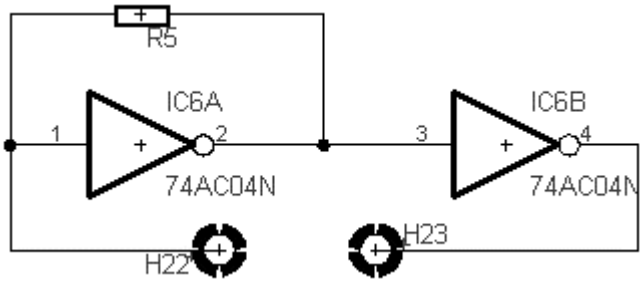


Figura 12.9. Schemă circuit basculant astabil realizat cu circuite logice TTL

Montajul din Figura12.9de mai sus este identic cu montajul din îndrumător, Figura12.2 (a).

Pentru a studia comportarea circuitului trebuie să conectăm pe rând câte un condensator (C1, C2 și C3) la bornele H22 și H23.

Potențiometrul POT2 (vezi stand – potențiometru galben) este reprezentat în schema de mai sus ca fiind R5. Primul lucru înainte de a efectua orice măsurătoare sau calcul, se măsoară (și se notează) această rezistență apoi se continuă restul operațiilor. Se recomandă R5 = 390**Ω.**

Pentru cele trei condensatoare se citesc perioadele de oscilație ale ieșirii circuitului basculant astabil de pe osciloscop și se compară cu cele calculate după formulele din îndrumător.

Ieșirea circuitului basculant astabil este reprezentată de borna H23.

12.5.2 Circuit basculant astabil realizat cu circuite logice Trigger Schmitt

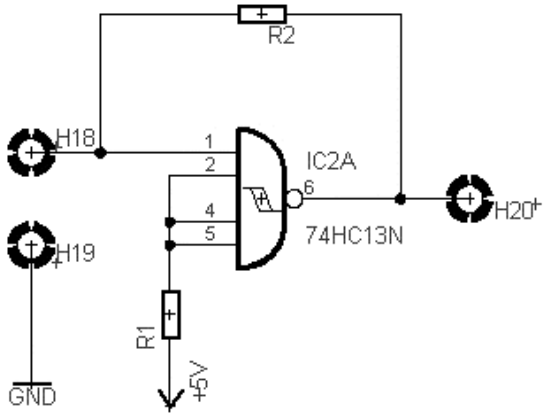


Figura12.10. Schemă circuit basculant astabil realizat cu circuite logice Trigger Schmitt

Montajul din Figura12.10 de mai sus este identic cu montajul din îndrumător, Figura12.3 (a). Asemănător cu primul tip de circuit basculant astabil, pentru a studia comportarea acestui circuit trebuie să conectăm pe rând câte un condensator (C1, C2 și C3) la bornele H18 și H19.

Potențiometrul POT3 (vezi stand – potențiometru verde) este reprezentat în schema de mai sus ca fiind R2. Primul lucru înainte de a efectua orice măsurătoare sau calcul, se măsoară (și se notează) această rezistență apoi se continuă restul operațiilor. Se recomandă R2 = 850Ω.

Pentru cele trei condensatoare se citesc perioadele de oscilație ale ieșirii circuitului basculant astabil de pe osciloscop și se compară cu cele calculate după formulele din îndrumător.

Ieșirea circuitului basculant astabil este reprezentată de borna H20.

12.5.3 Circuit basculant astabil realizat cu circuite logice CMOS

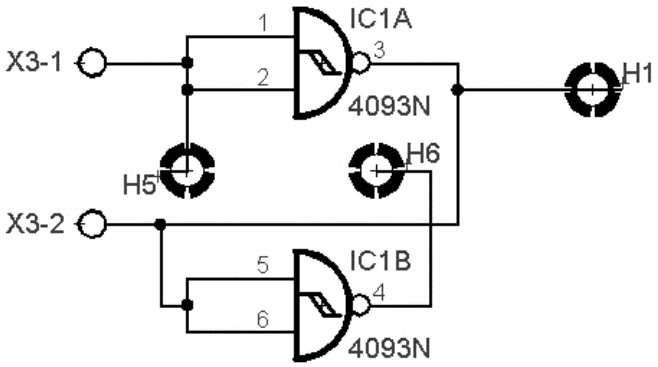


Figura12.11. Schemă circuit basculant astabil realizat cu circuite logice CMOS

Montajul din Figura12.11 de mai sus este identic cu montajul din îndrumător, Figura 12.4 (a).

Pentru a studia comportarea circuitului trebuie să conectăm pe rând câte un condensator (C1, C2 și C3) la bornele H5 și H6.

Potențiometrul POT1 (vezi stand – potențiometru albastru) este conectat în punctele X3-1 și X3-2. Primul lucru înainte de a efectua orice măsurătoare sau calcul, se măsoară (și se notează) această rezistență apoi se continuă restul operațiilor. Se recomandă R3 = 850KΩ.

Pentru cele trei condensatoare se citesc perioadele de oscilație ale ieșirii circuitului basculant astabil de pe osciloscop și se compară cu cele calculate după formulele din îndrumător.

Ieșirea circuitului basculant astabil este reprezentată de borna H1.

12.5.4 Circuit basculant astabil realizat cu cristal de cuarț

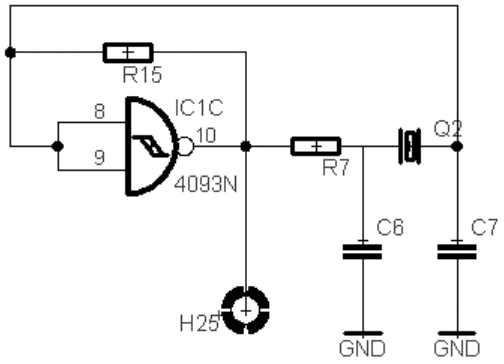


Figura 12.12. Schemă circuit basculant astabil realizat cu cristal de cuarț

Schema circuitului basculant astabil cu cuarț prezentat în îndrumător, Figura 12.7 a fost realizată pe cablaj, la proiectarea acestuia și nouă nu ne rămâne decât să vizualizăm ieșirea acestuia pe borna H25.

Dacă conectăm sonda osciloscopului la borna H25 și firul de GND al osciloscopului la borna H19 (GND-ul standului) putem citi perioada de oscilație de și apoi se poate calcula ușor frecvența cu care oscilează ieșirea circuitului.

În practică pe cristalul de cuarț se obișnuiește să se înscrie valoarea acestuia.

13. Comanda Dispozitivelor Optoelectronice

13.1 Scopul lucrării

Sunt prezentate principalele elemente optoelectronice de afișare și circuitele integrate utilizate pentru comanda lor. Se studiază caracteristicile elementelor de afișare și ale circuitelor decodificatoare precum și principalele tipuri de scheme utilizate pentru comanda afișoarelor optoelectronice.

13.2 Considerații teoretice

Elementele de afişare reprezintă punctul de atracție estetică al oricărui aparat electronic. Aproape toate afişoarele moderne satisfac într-un fel sau altul criteriul estetic, însă nu orice element de afişare este potrivit pentru o aplicaţie dată. Aici mai intervin criterii de contrast, culoare, unghi de citire, timp de răspuns, fiabilitate etc, care diferă substanțial de la un afişor la altul.

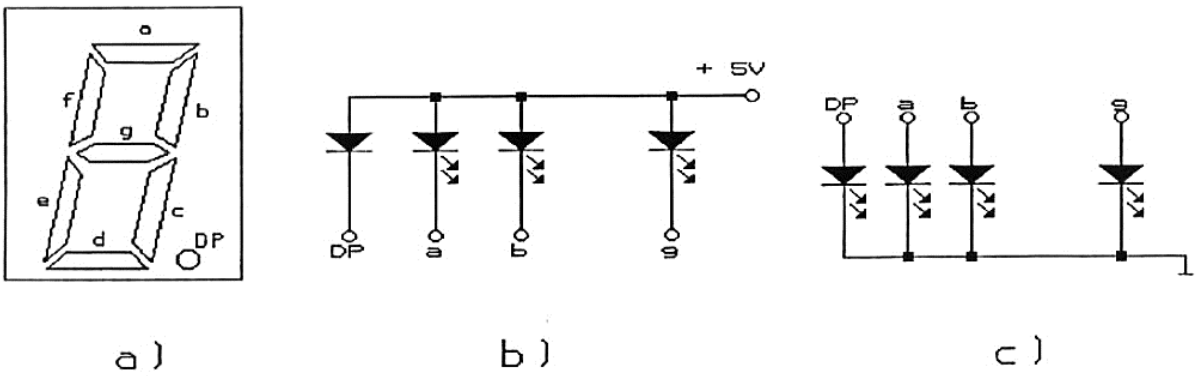


Figura 13.1.

În lucrarea de față interesează elementele de afişare a informaţiei şi în special a caracterelor zecimale. Cel mai răspândit afişor din această clasă este afişorul cu 7 segmente. După tipul dispozitivului utilizat pentru afişare, cele mai utilizate sunt afişoarele cu diode luminiscente - LED şi afişoarele cu cristale lichide - LCD.

Un circuit de afișare cu 7 segmente de tip LED este compus din 7 diode luminiscente plasate ca in Figura13.1 (a) şi conectate ca în Figura 13.1 (b) (cu anodul comun) sau 13.1 (c) (cu catodul comun). Aplicând semnale corespunzătoare la fiecare din terminalele a,b, ... g ale circuitului, diodele se pot aprinde sau stinge independent, permițând afișarea unor caractere formate cu ajutorul celor 7 segmente.

Pentru circuitul de afişare din Figura 13.1 (b) se conectează anodul comun la borna pozitivă a tensiunii de alimentare, iar semnalele logice de comandă se aplică pe catodul fiecărei diode prin intermediul unei rezistente care fixează curentul prin dioda (I) in limitele dorite. Iluminarea depinde de curentul prin diodă, existând recomandări de catalog in acest sens. De regulă curentul maxim prin circuit este de 20 mA.

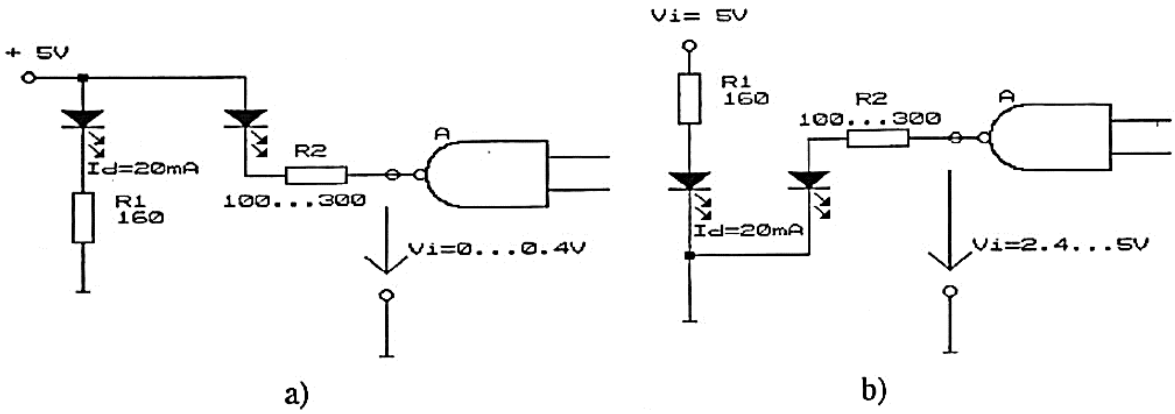


Figura13.2.

De exemplu in Figura13.2 (a) știind din datele de catalog ca la 20 mA căderea de tensiune pe segment este aproximativ 2V, dorind să limităm curentul, la această valoare se introduce rezistența:

$$R_1 = \frac{V_{cc\,max} - V_{LED} - V_{i\,min}}{I_{LED\,max}} = \frac{5.25 - 2 - 0}{20} \cdot 10^3\Omega = 162\Omega$$

Pentru un semnal de comandă TTL cu $V_i = V_{on} = 0.4V$ se poate observa o scădere a curentului la (4.75-2-0.4)/160 = 14.6 mA pentru valoarea minimă admișsi a tensiunii de alimentare.

În general rezistenta de limitare a curentului are valori între 100÷300Ω asigurându-se un curent mediu prin diodă de 10÷20 mA (inclusiv în regim de multiplexare).

Acest tip de circuit permite afișarea aplicând tensiuni având nivel logic 0, prin urmare semnalele de comandă sunt active pe 0.

În Figura13.2 (b) este prezentat un circuit similar dar cu catodul comun și semnale de comandă active pe 1 logic.

Afişorul cu 7 segmente de tip LCD este compus din 7 segmente de cristal lichid care au un electrod comun. Pentru realizarea afişorului este necesară o configuraţie adecvată a electrozilor care să permită activarea anumitor domenii ale cristalului care corespund elementelor unei celule de afişare cu 7 segmente (Figura13.1 (a)). Efectul folosit în afişoarele LCD este dispersia dinamică a luminii într-un cristal lichid. Sunt denumite cristale lichide substanțele care, in anumite condiții, prezintă o stare lichid-cristalină, adică o stare de agregare între cea lichidă, izotropă şi cea solidă, cristalina. La cristalele lichide în straturi subțiri - singurele care interesează in aplicaţiile optoelectronice - se deosebesc două orientări tipice in raport cu planul plăcilor suport (electrozii) între care se află cristalul lichid şi anume perpendicular, respectiv paralel cu plăcile suport. In primul caz avem o orientare homeotropă, cristalul fiind opac iar în al doilea caz o orientare omogenă, cristalul fiind transparent.

Dacă cristalelor lichide nematice omogene li se aplică între cei doi electrozi o tensiune U>U, cristalul devine opac. Cu U_p, s-a notat valoarea de prag a tensiunii continue de excitaţie, care este cuprinsă între 6V si 9V. Pentru a evita fenomenele electrolitice, cristalele lichide sunt întotdeauna excitate cu tensiuni alternative şi nu cu tensiuni continue. Are loc însă o creştere a tensiunii de prag odată cu frecventa semnalului astfel că, de regulă, tensiunea aplicată cristalelor lichide are o formă de semnal dreptunghiular, cu frecvență între 30 Hz (limita de clipire) şi 200 Hz (sub limita superioară de răspuns în frecventă a cristalului lichid).

Circuitele de afişare cu 7 segmente conțin și punctul zecimal care se comandă separat şi poate fi plasat în stânga sau în dreapta circuitului.

Alegerea unui anumit tip de afişor se face în funcţie de anumite criterii, cum ar fi: consum de putere, contrast, timp de răspuns, fiabilitate etc. Din punctul de vedere al puterii consumate, performantele net superioare aparțin afişoarelor cu cristale lichide 0.3A x 5V, fata de cca 20 mA x 2V cât necesită majoritatea diodelor luminiscente moderne. Această performanţă excepțională a afişoarelor LCD are însă şi un revers: în absența unor surse de lumină auxiliare, deci în întuneric, afişoarele cu cristale lichide nu pot fi utilizate. Ele sunt elemente de afişare pasive care fac apel la lumina din mediul inconjurător, spre deosebire de LED-uri care sunt elemente de afişare active, deci surse de lumina.

În ceea ce priveşte timpul de răspuns, la afişoarele cu LED-uri acesta este de circa 100ns, ceea ce satisface exigentele cele mai pretenţioase. Deşi afişajele cu cristale lichide sunt lente (frecventa semnalului de comandă nu depăşeşte de regulă 1kHz impediment în aplicațiile uzuale deoarece ochiul omenesc introduce el însuşi o limită de frecvență destul de severă.

In fine, o caracteristica de asemenea importantă pentru afişoare este durata de viață. În condiții de temperatură şi mediu normale LCD-urile au o durata de viața de 25.000-50.000 ore iar LED-urile manifestă o scădere la circa 50% a intensității luminoase după 200.000 ore de funcţionare.

Având precizate noţiunile elementare legate de dispozitivele de afişare se poate trece acum la prezentarea decodificatoarelor BCD-7 segmente. Aceste decodificatoare sunt circuite care transforma comanda pentru afişarea fiecărui caracter, exprimată de obicei in cod BCD, în 7 semnale de comandă efectivă pe fiecare segment al circuitului de afisaj a,b,c,d,e,f,g.

Circuitele TTL SN 7446 şi SN 7447 sunt circuite decodificatoare BCD 7 segmente cu ieşirile active in starea 0 logic şi sunt destinate comenzii elementelor de afişare cu anod comun. Circuitele SN 7448 și SN 7449 au ieşirile active în starea 1 logic şi sunt destinate pentru comanda afişoarelor cu catodul comun.

Etajul de ieşire al acestor circuite este cu colectorul în gol ceea ce permite conectarea elementelor comandate cu tensiuni între 5.5 și 30 V şi un curent de ieşire in starea activa de până la 40 mA.

Se poate observa de asemenea că aceste circuite dispun de o serie de semnale de comandă pentru ştergere, pentru test, etc. care le asigura performanțe superioare în exploatare.

Circuitele integrate CMOS MMC 4511 şi MMC 4543 sunt circuite decodificatoare BCD-7 segmente ce contin 4 latch-uri de stocare a datelor, decodorul BCD-7 segmente şi drivere de ieşire.

Circuitul MMC 4511 a fost conceput pentru a comanda direct afişaje cu LED-uri cu catod comun. Acest circuit este construit în logică CMOS dar cu tranzistoare bipolare non la ieşire care îi permit să furnizeze un curent de până la 25 mA in starea 1 logic.

Circuitul MMC 4543 a fost proiectat pentru a fi utilizat la comanda afişoarelor cu cristale lichide. Pentru aceasta trebuie aplicat un semnal dreptunghiular pe intrarea PH conectată la electrodul comun al afişajului. Ieşirile circuitului se conectează direct la segmentele afişajului (Figura13.3 (a)).

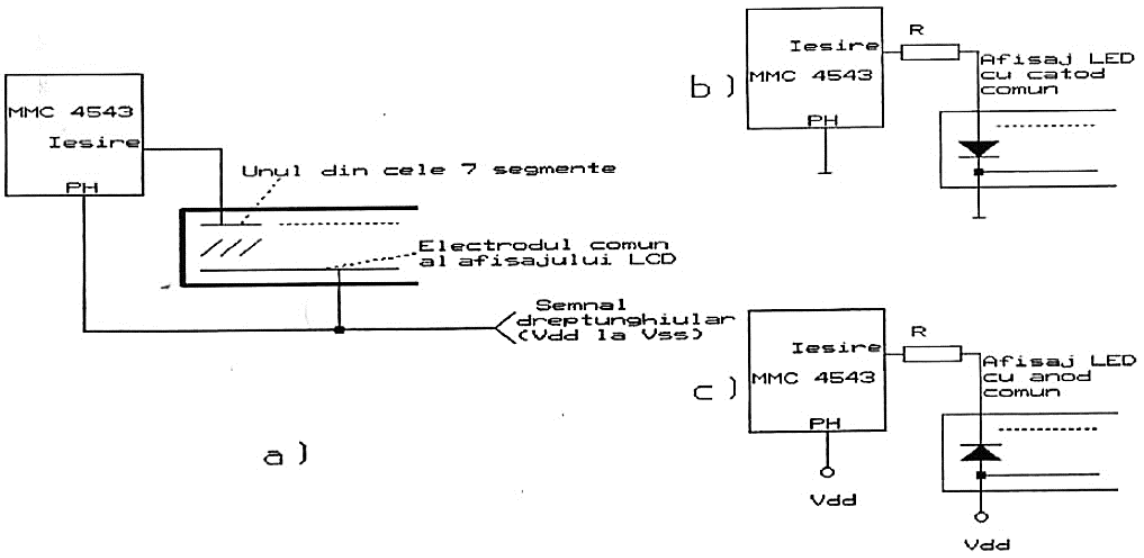


Figura13.3.

Intrarea PH (PHASE) poate fi folosită pentru a complementa nivelele logice de pe ieşiri. Astfel circuitul poate comanda afişaje cu LED-uri atât cu anod comun cât şi cu catod comun. Pentru curenți mai mici de 10 mA sau U_{dd}> 10V, afişajele se pot comanda direct. Rezistenta R se va dimensiona corespunzător curentului care va străbate LED-ul, luând în considerare şi rezistenta de ieşire a circuitului. Intrarea de comandă a fazei (PH) se va conecta la V_{ss} pentru afişajele cu catod comun şi la V_{dd} pentru afişajele cu anod comun (Figura 13.3 (b),(c)).

Schemele utilizate pentru comanda afişoarelor pot lucra prin multiplexare (secvențial) în cazul în care se cer afişate un număr mai mare de cifre şi fără multiplexare, pentru afişarea unui număr mai mic de cifre, când nu interesează numărul de componente utilizate. Modul de lucru prin multiplexare se prezintă în Figura13.4.

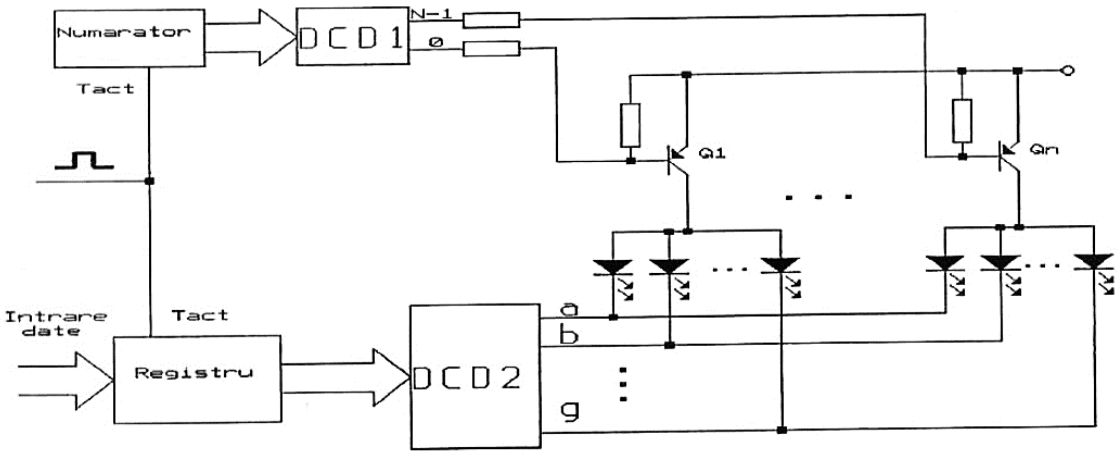


Figura13.4.

Registrul şi numărătorul işi schimbă stările sincron. Decodificatorul DCD1 comandă deschiderea succesivă a tranzistoarelor Q_1....,Q_n, deci activarea la un anumit moment numai a unui circuit de afişare şi anume a aceluia determinat de conținutul registrului. Astfel dacă toate ieşirile numărătorului sunt în 0 logic DCD1 va comanda deschiderea tranzistorului Q_1 şi se activează afişorul corespunzător. La următorul tact se modifică atât conținutul numărătorului cât şi al registrului ceea ce determină saturarea tranzistorului Q_2, activându-se cel de al doilea afişor. Primul afişor se va stinge deoarece Q_1, se blochează etc.

Dacă T este perioada de timp în care se activează toate afişoarele, un afişor va fi activat un interval de timp T/N. Pentru ca circuitele să nu pâlpâie este necesar ca frecvența de afisaj să fie mai mare de 30 Hz. La frecvente peste 30 Hz ochiul uman nu sesizează faptul că afişoarele nu sunt comandate tot timpul. În aceste condiții frecventa de repetiție a impulsurilor de tact va trebui să fie mai mare de 30·N Hz (de obicei 50·N Hz). Această condiţie este relativ uşor de îndeplinit deoarece afişoarele pot funcţiona la frecvenţe ridicate. O altă problemă care mai trebuie rezolvată este cea a curentului pe segment care trebuie mărit de N ori pentru a se menţine intensitatea luminoasă caracteristica circuitelor fără multiplexare. Curentul prin fiecare tranzistor va fi de 7·N ori mai mare decât curentul obişnuit printr-un segment de afişor.

13.3 Mersul lucrării

Cel mai simplu afișaj optoelectronic este LED-ul (Light Emitting Diode)Figura13.5. Cu ajutorul unui led obișnuit se pot afișa cel puțin 3 semnale independente:

1. Stare 1 – ledul este oprit
2. Stare 2 – ledul este pornit.
3. Stare 3 – ledul clipește.

Pentru a putea afișa alte stări trebuie să se definească timpul de clipire (de ex : o stare se poate defini ca fiind atunci când ledul clipește cu frecvența de 1 sec și o altă stare se poate defini ca fiind atunci când ledul clipește cu frecvența de 1 min).

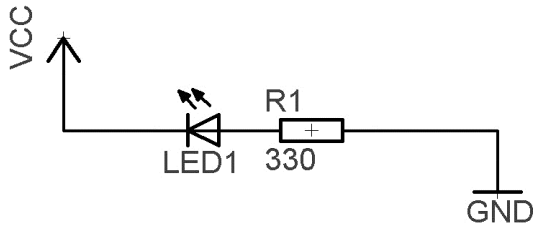


Figura 13.5. Schema standard de conectare LED

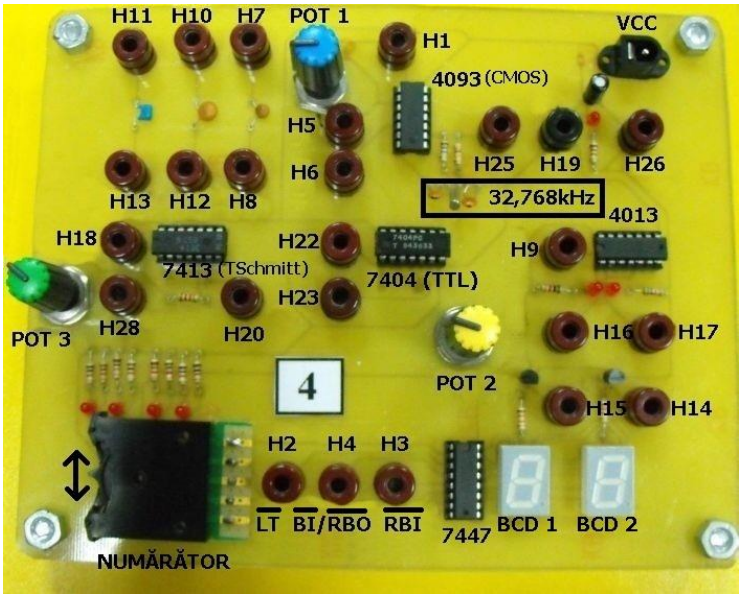


Figura 13.6. Stand de testare afișajeOptoelectronice cu 7 segmente

Un afișaj optoelectronic cu 7 segmente este format din 7 leduri așezate prin construcție la mijlocul a 7 segmente aranjate în forma cifrei 8.

Elementele componente de care avem nevoie pentru a realiza testarea a două afișaje cu 7 segmente și pentru a putea observa limita de clipire (30 Hz) sunt următoarele:

- circuit basculant astabil realizat cu circuite CMOS și cu posibilitatea de a modica frecvența ieșirii între 1 Hz, 30 Hz și până la 100 Hz;
- circuit bistabil de tip D (avem nevoie de ieșirea și ieșirea negată);
- etaj de comandă cu tranzistor a bornei comune a afișajelor (independent pentru fiecare afișaj) ;
- decodificator BCD-zecimal 74 LS 47;
- comutator decadic (numărător mecanic) 4 biți.

Semnale (active pe "0")

- LT (lamp test) – la activarea acestui semnal se aprind toate segmentele afișajului pentru a verfica buna funcționare a acestora;
- BI/RBO (blanking input, ripping blanking output);
    - BI - șterge orice ar fi de afișat;
    - RBO - se folosește în combinație cu RBI, atunci când se cascadează decodificatoare, pentru a șterge zerourile nesemnificative din partea stângă;
- RBI – (ripping blanking input) – stinge afișajul când are de afișat cifra 0.

Modalitate de lucru :

În această lucrare vom folosi un circuit basculant astabil realizat cu circuit logic CMOS prin conectarea condensatorului C1 cu bornele H11 și H13 la bornele H5 și H6.

Ieșirea circuitului basculant astabil H1 se v-a conecta la intrarea de clock a circuitului bistabil, adică la borna H9.

Ieșirile circuitului bistabil H16 și H17 se folosesc pentru a comanda alimentarea celor două afișaje. Borna H16 se conectează la borna H15, iar borna H17 se conectează la borna H14. Este binecunoscut faptul că acest tip de bistabil ne oferă la o ieșire Q și încă o ieșire $\bar{Q}$.

Dacă folosim cele două ieșiri Q și $\bar{Q}$ pentru a comanda bazele a două tranzistoare care la rândul lor comandă alimentarea celor două afișaje, înseamnă că atunci când un afișaj este pornit, sigur celălalt este oprit. Acest lucru este vizibil cu ochiul liber atunci când frecvența cu care se comandă bistabilul respectiv și implicit schimbarea informației de la ieșirea acestuia este sub 30 Hz. Peste 30 Hz ochiul uman nu mai observă oprirea/pornirea afișajelor, le percepe ca fiind pornite tot timpul.

14. Memorii Semiconductoare Volatile

14.1 Scopul lucrării

Sunt prezentate principiile constructiv-funcţionale şi operaţiile de bază pentru memoriile semiconductoare. Se studiază modul de lucru cu memoriile semiconductoare volatile (RAM).

14.2 Consideraţii teoretice

Memoriile sunt prezente azi peste tot, de ele nu se mai poate dispensa nici industria de calculatoare, nici cea de aparatură de uz casnic şi cu atât mai puţin constructorii de dispozitive de măsură şi reglare comandate prin microprocesoare.

Circuitele de memorie cu semiconductoare, privite ca structuri care pot implementa funcţii logice, se încadrează în clasa circuitelor combinaţionale deşi unele au în structurile lor celule care sunt circuite secvenţiale.

Ca o regulă, memoriile stochează date în unităţi care sunt formate din unul sau mai mulţi biţi. Cea mai mică unitate de date este bitul. În multe aplicaţii, datele sunt manipulate într-o unitate de 8 biţi numită un octet (byte) sau în unităţi, care sunt multiplii al unui octet. Octetul poate fi descompus în două unităţi de 4 biţi, numite nibble. O unitate completă de informaţii este numită un cuvânt şi în general este alcătuit din unu sau mai mulţi octeţi. Unele memorii stochează date în grupe de 9 biţi; un grup de 9 biţi este alcătuit dintr-un octet şi un bit de paritate.

Structural, un modul tipic de memorie integrată este organizat sub forma unei matrice de celule de memorie. Se consideră ca fiind o celulă de memorie circuitul elementar care realizează memorarea unui bit. Pe lângă matricea elementelor de memorie circuitul este prevăzut cu circuite de selecţie şi circuite care facilitează înscrierea, respectiv citirea informaţiei. Stocarea unei informaţii sau regăsirea unei informaţii stocate necesită furnizarea unor informaţii privind locul unde se găseşte această informaţie. Aceste semnale constituie intrări pentru circuitul de memorie şi se numesc adrese. Cuvintele binare memorate constituie date pentru acest circuit şi ele sunt semnale de intrare atunci când se înscrie în memorie şi semnale de ieşire atunci când se citeşte din memorie. Mai trebuie precizat că accesul la memorie se face la un moment de timp bine determinat, moment stabilit prin activarea unui semnal de intrare al circuitului de memorie.

Bufferele cu trei stări (TS) au trei stări de ieşire: *HIGH* (1), *LOW* (0) şi *HIGH-Z* (înaltă impedanţă). Într-o memorie, aceste buffere permit liniilor de date să funcţioneze fie ca linii de intrare, fie ca linii de ieşire, fie să treacă în starea de înaltă impedanţă, când practic nu influenţează magistrala comună de date. Ieşirile cu trei stări sunt indicate prin simbolurile logice cu un triunghi mic, inversat (▽) şi sunt folosite pentru compatibilitate cu structurile de magistrală.

Fizic, o magistrală este un set de căi conductive, care servesc pentru interconectarea a două sau mai multe componente funcţionale ale unui sistem sau a mai multor sisteme diferite. Electric, o magistrală este o colecţie de nivele specificate de tensiune şi/sau de curent şi de semnale, care permit diverselor instrumente, conectate la magistrală, să comunice şi să lucreze complet împreună.

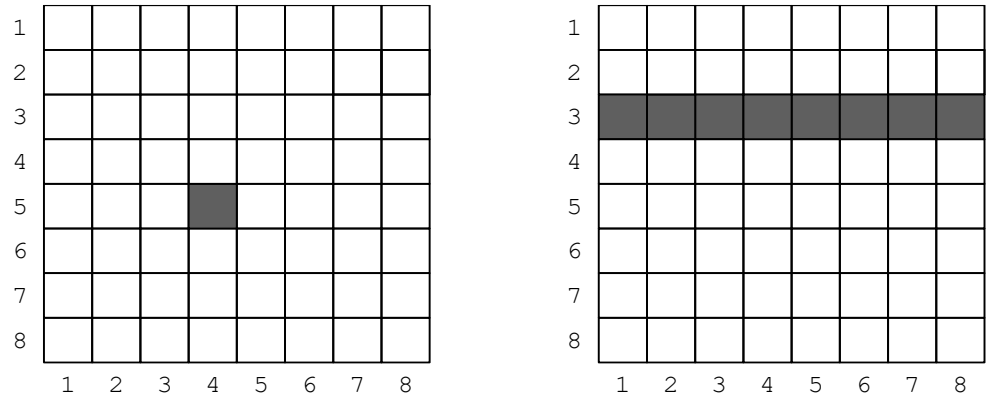
De exemplu un microprocesor este conectat la memorii şi la dispozitive de intrare/ieşire cu ajutorul anumitor structuri de magistrală. O magistrală de adrese permite microprocesorului adresarea memoriilor şi tot ea susţine transferul de date între microprocesor, memorii şi dispozitivele de intrare/ieşire, ca monitoare, imprimante, tastaturi şi modemuri. Magistrala de control permite microprocesorului controlul transferului de date şi sincronizarea componentelor.

Caracteristicile cele mai importante ale unei memorii sunt:

- geometria sau modul de organizare a memoriei, reprezentat de lungimea unui cuvânt şi numărul de cuvinte memorate;
- capacitatea memoriei, reprezentând numărul total de unităţi ce pot fi memorate; se exprimă în general în multipli de 1k =2^{10}=1024 biţi (octeţi), sau 1M=2^{20}=1048576 biţi (octeţi);
- timpul de acces la memorie; se exprimă în μs sau ns şi reprezintă timpul necesar pentru citirea sau înscrierea unor informaţii în memorie;
- puterea consumată - pentru caracterizarea din acest punct de vedere a unei memorii se foloseşte puterea consumată raportată la un bit de informaţie, respectiv raportul dintre puterea totală consumată şi capacitatea memoriei. Se măsoară în μW / bit;
- volatilitatea - o memorie este volatilă dacă informaţia înscrisă se pierde în timp. Pierderea informaţiei se poate datora fie modului de stocare a acesteia (memorii dinamice) fie datorită dispariţiei tensiunilor de alimentare a circuitului.

14.2.1 Matricea de memorie

Aşa cum s-a mai spus, un modul de memorie integrată este organizat sub forma unei matrice de celule de memorie. *Locul* unei unităţi de date într-o matrice de memorie este precizat prin adresa acestei unităţi. De exemplu, în Figura 14.1 (a) adresa unui bit din matrice este specificată de o linie (linia 5) şi o coloană (coloana 4), aşa cum este arătat. În Figura 14.1 (b) adresa unui octet este specificată numai de o linie. Se poate observa că modul de adresare depinde de felul în care memoria este organizată în unităţi de date.



a) Adresa bitului marcat cu negru este linia 5 coloana 4

b) Adresa octetului marcat cu negru este linia 3

Figura 14.1.

Capacitatea unei memorii este numărul total de unităţi de date care pot fi stocate. De exemplu, în cazul matricei de memorie organizată în biţi din Figura 14.1 (a) capacitatea este 64 biţi. În cazul matricei de memorie organizată în octeţi din Figura 14.1 (b), capacitatea este 8 octeţi, care reprezintă de fapt 64 biţi.

14.2.2 Operaţiile de bază ale memoriei

Deoarece o memorie stochează date binare, datele trebuie depuse în memorie şi trebuie copiate din memorie, atunci când este nevoie. Operaţia de scriere depune date la o adresă specificată din memorie şi operaţia de citire ia date de la o adresă specificată din memorie. Operaţia de adresare, care este parte atât a operaţiei de scriere cât şi a operaţiei de citire, selectează adresa de memorie specificată.

Unităţile de date intră în memorie, în timpul unei operaţii de scriere şi ies din memorie, în timpul unei operaţii de citire, pe un set de linii numite magistrala de date. Aşa cum este arătată în Figura 14.2, magistrala de date este bidirecţională, ceea ce înseamnă că datele pot circula în ambele direcţii (spre şi dinspre memorie).

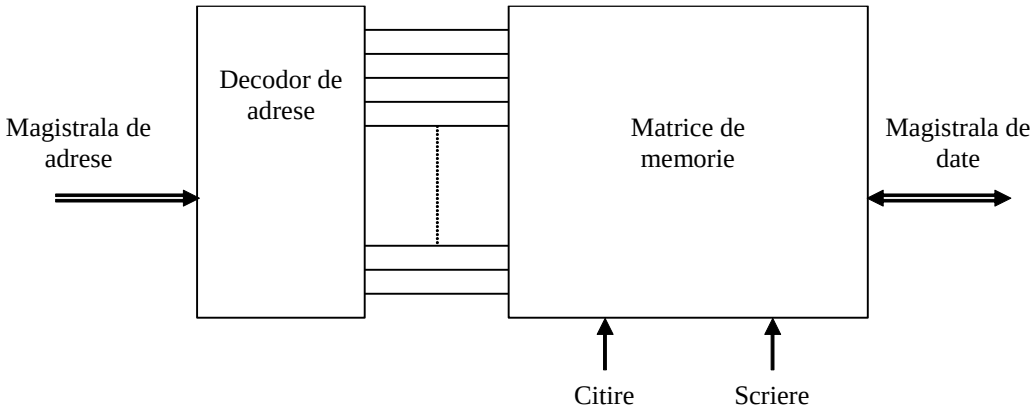


Figura 14.2.

În cazul unei memorii organizată în octeţi, magistrala de date are cel puţin 8 linii, încât toţi cei 8 biţi de la o adresă selectată sunt transferaţi paralel. Pentru o operaţie de scriere sau de citire, o adresă este selectată localizând un cod binar, reprezentând adresa dorită, pe un set de linii, numit magistrala de adrese. Codul de adresă este decodificat intern şi adresa corespunzătoare este selectată. Numărul liniilor din magistrala de adrese depinde de capacitatea memoriei. De exemplu un cod de adresă de 15 biţi poate selecta 32.768 locaţii (2^{15}) din memorie, un cod de adresă de 16 biţi poate selecta 65.536 locaţii (2^{16}) din memorie şi aşa mai departe. În calculatoarele personale, o magistrală de adrese de 32 biţi poate selecta 4.294.967.296 locaţii (2^{32}), exprimat ca 4G.

Operaţia de scriere. O operaţie de scriere simplificată este arătată în Figura 14.3. Pentru stocarea unui octet de date în memorie, un cod păstrat în registrul de adrese este plasat pe magistrala de adrese. Îndată ce codul de adresă se află pe magistrală, decodorul de adrese decodifică adresa şi selectează din memorie locaţia specificată. Apoi memoria primeşte o comandă de scriere şi octetul de date, păstrat în registrul de date, este plasat pe magistrala de date şi depozitat la adresa de memorie selectată, astfel terminând operaţia de scriere. Atunci când un octet nou de date este înscris la o adresă de memorie, octetul de date stocat curent la adresa respectivă este suprascris şi distrus.

Operaţia de citire. O operaţie de citire simplificată este ilustrată în Figura 14.4. Şi în acest caz un cod păstrat în registrul de adrese este plasat pe magistrala de adrese. Îndată ce codul de adresă se află pe magistrală, decodorul de adrese decodifică adresa şi selectează din memorie locaţia specificată. Apoi, memoria primeşte o comandă de citire şi o "copie" a octet-ului de date, care este stocat la adresa selectată din memorie, este plasată pe magistrala de date şi încărcată în registrul de date, astfel operaţia de citire fiind încheiată. Când un octet de date este citit de la o adresă din memorie, el rămâne stocat la adresa respectivă şi nu este distrus. Aceasta se numeşte citire nedistructivă.

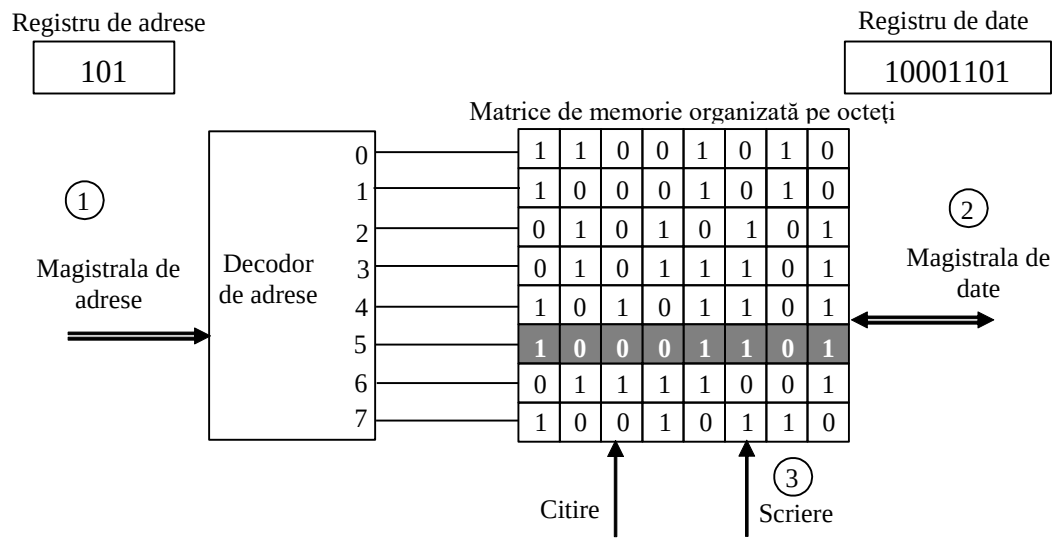


Figura 14.3.

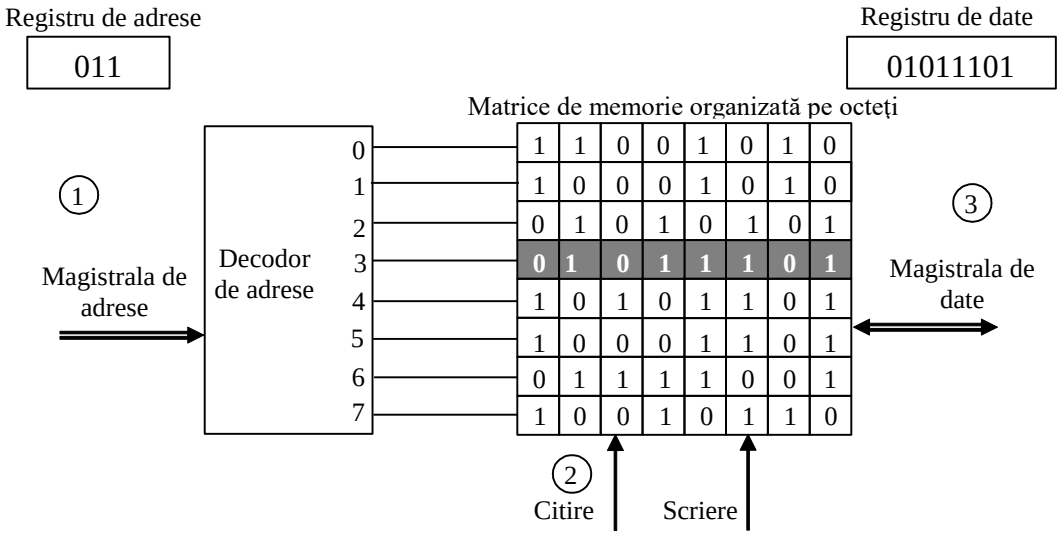


Figura 14.4.

14.2.3 Structura circuitelor de memorie

Cele două categorii principale de memorii semiconductoare sunt memoriile RAM şi memoriile ROM. RAM-ul (Random-Access Memory) este un tip de memorie, în care toate adresele sunt accesabile în durate de timp egale şi pot fi selectate în orice ordine pentru o operaţie de citire sau de scriere. Toate RAM-urile au capabilitate de citire şi de scriere. Deoarece RAM-urile pierd datele stocate atunci când alimentarea este oprită, ele sunt memorii volatile.

ROM-ul (Read-Only Memory) este un tip de memorie, în care datele sunt stocate permanent sau semipermanent. Datele pot fi citite din ROM, dar nu există operaţia de scriere ca în cazul RAM-urilor. ROM-ul ca şi RAM-ul este o memorie cu acces aleatoriu, dar termenul RAM înseamnă, tradiţional, memorie cu acces aleatoriu cu citire/scriere. Deoarece ROM-urile reţin date stocate chiar dacă alimentarea este oprită, ele sunt memorii nevolatile.

14.2.4 Memorii ROM

O memorie ROM conţine date stocate permanent sau semipermanent, care pot fi citite din memorie dar nu pot fi modificate deloc sau numai cu echipamente speciale. Un ROM reţine date care sunt folosite în mod repetat în aplicaţiile de sistem, ca şi tabele, conversii sau instrucţiuni programate pentru iniţializarea şi funcţionarea sistemului.

Memoriile semiconductoare ROM sunt folosite doar pentru citirea informaţiei (înscrisă anterior), informaţie ce este rezidentă permanent în cadrul sistemului. Pentru aceasta memoria ROM trebuie să fie de tip nevolatil, adică în lipsa tensiunii de alimentare informaţia nu se distruge.

Circuitele de memorie ROM generează deci un set fix de cuvinte (înscris anterior) atunci când este adresat. În funcţie de modul cum aceste cuvinte pot fi înscrise şi eventual şterse există mai multe tipuri de memorii ROM:

- memorii ROM cu mascare - ROM;
- circuite de memorie programabile - PROM;
- circuite ROM care pot fi şterse şi programate - EPROM şi EEPROM.

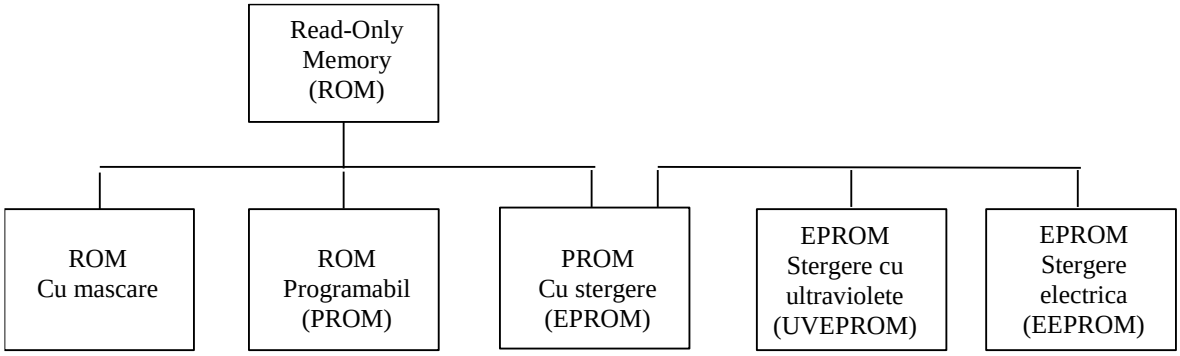


Figura 14.5.

Figura 14.5 arată modul în care ROM-urile semiconductoare pot fi clasificate. ROM-ul cu mascare este tipul de memorie în care datele sunt stocate permanent în timpul procesului de fabricaţie. Circuitele ROM cu mascare sunt denumite de obicei simplu, ROM. **PROM**-ul sau ROM-ul programabil este memoria în care datele sunt stocate electric de către utilizator cu ajutorul echipamentelor specializate. Ambele, ROM-ul cu mascare şi PROM-ul pot fi realizate în tehnologie MOS sau bipolar. **EPROM**-urile reprezintă PROM - uri care pot fi şterse şi reprogramate. **UVEPROM**-ul este programabil electric de utilizator dar datele stocate trebuie şterse prin expunerea circuitului la o radiaţie ionizantă ultravioletă pe o perioadă de timp de câteva minute. **EEPROM**-ul care poate fi programat şi şters electric.

14.2.5 Memoriile RAM statice

Denumirea de memorie cu acces aleator, RAM, sugerează că accesul la oricare cuvânt al memoriei este realizabil în acelaşi interval de timp. Dar acelaşi timp de acces, pentru oricare cuvânt este realizat şi de către o memorie ROM. Există şi memorii la care timpul de acces nu este

acelaşi pentru oricare cuvânt, denumite memorii cu acces secvenţial sau secvenţiale (benzile magnetice, discurile, memoriile cu bule magnetice, memoriile cu dispozitive cuplate prin sarcină CCD etc.) Pentru accesul secvenţial trebuie să se parcurgă toate adresele (locaţiile), de la cea prezentă la cea care se află cuvântul dorit. Accesul secvenţial este caracterizat de timpul mediu de acces. Mai corectă denumire, pentru memoriile semiconductoare cu acces aleator, ar fi memorie cu citire şi scriere RWM (Read Write Memory).

RAM-urile sunt deci memorii cu scriere/citire, în care datele pot fi scrise sau din care datele pot fi citite de la orice adresă selectată, în orice succesiune. Când o unitate de date este înscrisă la o anumită adresă din RAM, unitatea de date stocată anterior la acea adresă va fi înlocuită cu noua unitate de date. Când o unitate de date este citită de la o anumită adresă din RAM, atunci unitatea de date rămâne stocată şi nu va fi distrusă de operaţia de citire. Această operaţie de citire nedistructivă poate fi privită ca o copiere a conţinutului unei adrese în timp ce conţinutul respectiv rămâne neatins. Un RAM este tipic folosit pentru stocarea datelor pe termen scurt, deoarece nu reţine datele stocate când alimentarea este oprită.

Cele două categorii de RAM sunt: RAM-ul static (SRAM) şi RAM-ul dinamic(DRAM). RAM-urile statice folosesc circuite basculante bistabile ca elemente de memorare şi din acest motiv ele pot stoca date, pe timp nelimitat dacă alimentarea este menţinută. RAM-urile dinamice folosesc ca elemente de memorare condensatoare, pentru înmagazinarea, pe o durată de timp finită, a unei sarcini electrice şi nu pot reţine date pe termen foarte lung fără ca aceste condensatoare să fie reîncărcate într-un proces numit reîmprospătare. SRAM-urile şi DRAM-urile vor pierde datele memorate când alimentarea este oprită şi din această cauză ele sunt clasificate ca şi memorii volatile.

Datele pot fi citite mult mai repede din memoriile SRAM decât din memoriile DRAM. Totuşi circuitele de memorie DRAM pot stoca mult mai multe date decât circuitele de memorie SRAM de aceleaşi dimensiuni fizice şi la acelaşi preţ de cost, deoarece celula DRAM este mult mai simplă şi pot fi integrate pe o anumită suprafaţă mai multe astfel de celule, decât în cazul unui SRAM.

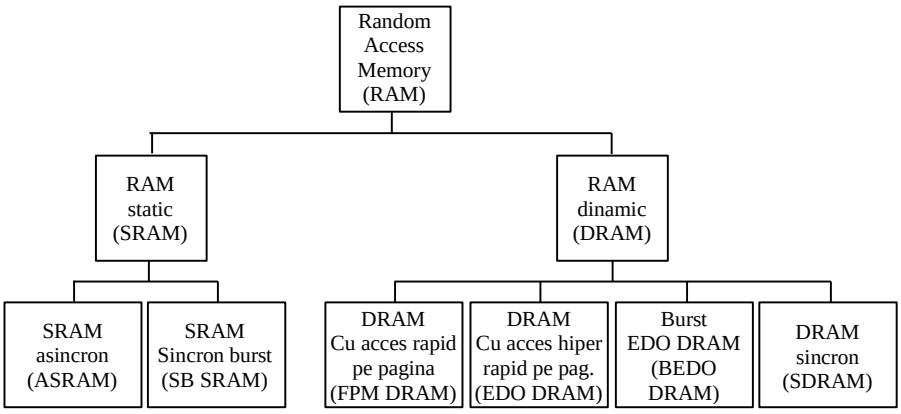


Figura 14.6.

Datorită costului mai mic şi dimensiunilor mai reduse, DRAM-ul este preferat pentru memoria principală a sistemelor de calcul, în timp ce SRAM-ul este folosit în primul rând la implementarea memoriei cache.

14.2.6 Memoriile RAM statice

Circuitul elementar de memorie statică are o structură de circuit basculant bistabil realizat cu tranzistoare MOS. Totodată există câteva SRAM-uri de capacităţi mai mici, implementate cu tranzistoare bipolare.

Celula de memorie din Figura 14.7 (a) foloseşte şase tranzistoare NMOS. Tranzistoarele Q_1 şi Q_2 reprezintă sarcini active, iar Q_3 şi Q_4 constituie bistabilul propriu-zis. Tranzistoarele Q_5 şi Q_6 permit conectarea ieşirilor bistabilului la liniile de bit (DL şi $\overline{DL}$) pentru citirea sau înscrierea informaţiei. În starea neselectată linia de selecţie cuvânt - WL se află la potenţial zero şi tranzistoarele Q_5 şi Q_6 sunt blocate. Liniile de bit sunt conectate prin rezistenţe la V_{DD}.

Pentru citire se aplică tensiune ridicată pe linia WL. Tranzistoarele Q_5 şi Q_6 se deschid formând sarcini suplimentare către V_{DD} prin rezistenţele de la capetele liniilor de bit, prin care se vor închide curenţii care vor fi sesizaţi de amplificatoarele de citire conectate pe liniile de bit DL şi $\overline{DL}$.

Pentru înscrierea informaţiei se ridică potenţialul liniei WL şi apoi se forţează cu circuite adecvate tensiune zero pe linia de bit în care dorim să obţinem zero la citire.

Matricea de memorie statică

Celulele de stocare într-un SRAM sunt organizate în linii şi coloane, aşa cum este ilustrat în Figura 14.7 (b), în cazul unei matrici nx4. Liniile de bit (DL şi $\overline{DL}$) se folosesc pentru scrierea şi citirea informaţiei în celulă; aceste linii sunt comune tuturor celulelor de pe aceeaşi coloană dintr-o matrice de memorie. Linia de selecţie cuvânt (WL) reprezentă selecţia pe linii în matricea de memorie; activarea acestei linii face posibilă citirea sau scrierea informaţiei în oricare din celulele de memorie situate pe aceeaşi linie în matrice.

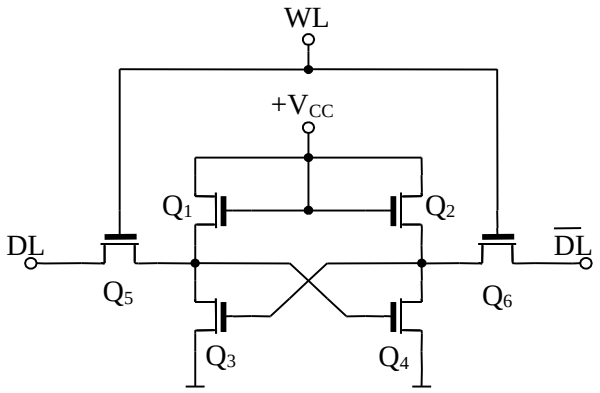


Figura 14.7. (a)

Pentru scrierea unei unităţi de date, în acest caz de 4 biţi, într-o anumită linie de celule din matricea de memorie, se activează linia WL şi se plasează cei 4 biţi de date pe liniile DL şi $\overline{DL}$. Apoi se activează linia *Scrie* şi fiecare bit va fi stocat într-o celulă selectată din coloana asociată. Pentru citirea unei unităţi de date, se activează linia C*iteşte* şi cei 4 biţi de date stocaţi în linia selectată vor apare pe liniile de ieşire de date.

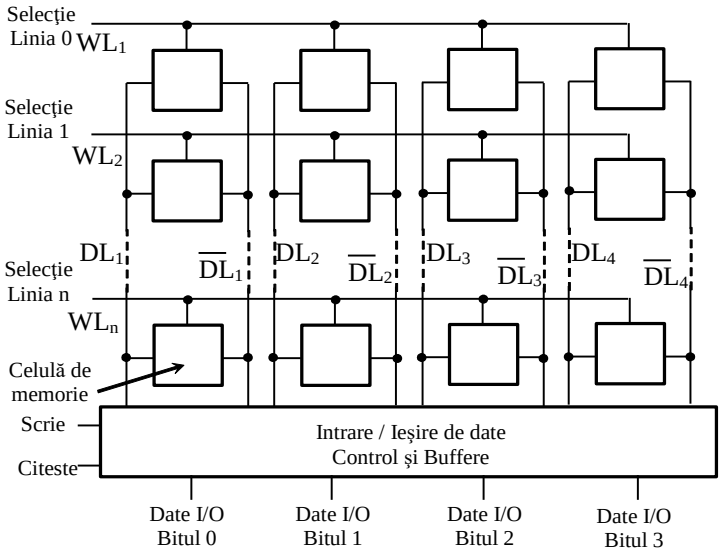


Figura 14.7. (b)

Organizarea unui SRAM asincron

Un SRAM asincron este o memorie în care operaţia nu este sincronizată cu un ceas de sistem. Pentru ilustrarea organizării unui SRAM, se va considera o memorie 32k x 8. Un simbol logic pentru această memorie este prezentat în Figura 14.8.

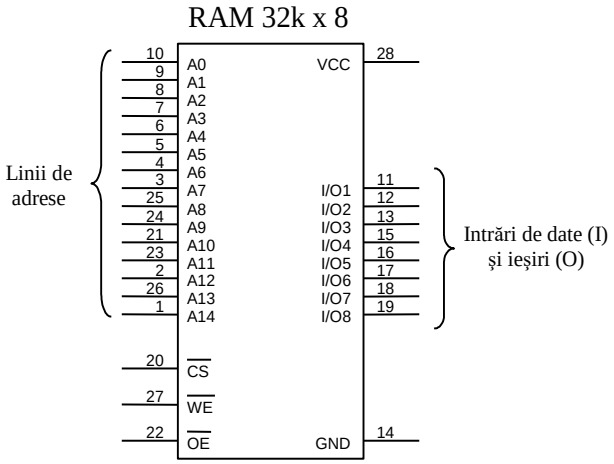


Figura. 14.8.

Citirea. În modul de citire (*READ*), linia de validare scriere, $\overline{WE}$, este pusă pe *HIGH* şi linia de validare a ieşirii, $\overline{OE}$, este pusă pe *LOW*. Buffere de intrare cu trei stări sunt invalidate (aduse în starea de înaltă impedanţă) de poarta G1 iar bufferele de ieşire cu trei stări de la ieşirea decodificatorului de coloane, sunt validate de poarta G2. Astfel cei 8 biţi de date de la adresa selectată sunt transmişi prin coloana I/O la liniile de date (de la I/O1 până la I/O8), care acţionează ca linii de date de ieşire.

Scrierea. În modul de scriere (*WRITE*) $\overline{WE}$ este *LOW* şi $\overline{OE}$ este *HIGH*. Bufferele de intrare sunt validate prin poarta G1 şi bufferele de ieşire sunt invalidate prin poarta G2. Astfel cei 8 biţi de date de intrare sunt plasaţi de pe liniile de intrare prin blocul de control al datelor de intrare şi coloanele I/O la adresa selectată şi stocate.

Ciclurile de citire şi de scriere.Figura 14.9 arată diagrame de sincronizare tipice pentru un ciclu

de scriere şi de scriere, în cazul unei memorii. În cazul ciclului de citire, prezentat în partea (a) a figurii, un cod de adresă validă este aplicat pe liniile de adresă într-un interval de timp specificat de durata ciclului de citire, t_{RC}. Pe urmă intrările $\overline{CS}$ şi $\overline{OE}$ sunt puse pe LOW. La un anumit interval de timp după ce intrarea $\overline{OE}$ a fost pusă pe LOW, un octet de date valide de la adresa selectată apare pe liniile de date. Acest interval de timp este numit timpul de acces faţă de validarea circuitelor de ieşire t_{GQ}.

Se pot defini încă două intervale diferite de timp de acces pentru ciclul de citire: timpul de acces faţă de activarea liniilor de adrese, t_{AQ}, măsurat de la momentul în care adresele sunt valide până la apariţia unei date valide pe liniile de date; şi timpul de acces faţă de validarea circuitului, t_{EQ}, măsurat de la trecerea semnalului $\overline{CS}$ din HIGH în LOW până la apariţia datelor valide pe liniile de date.

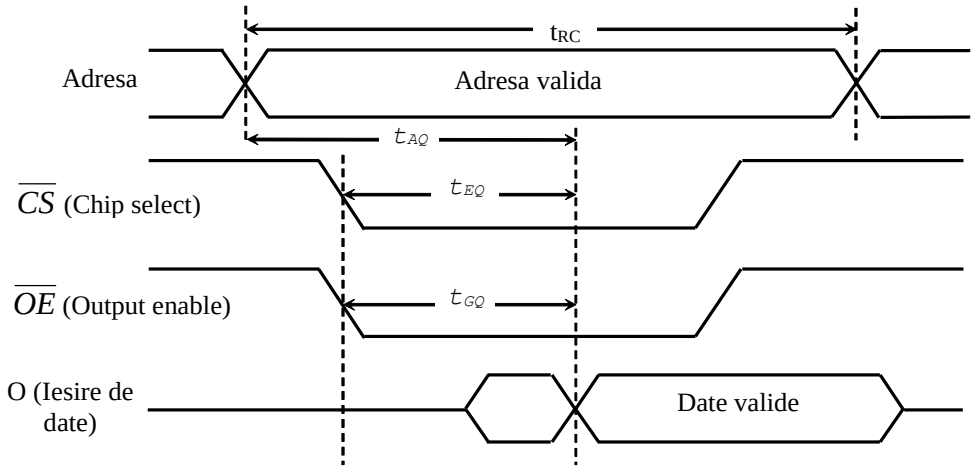


Figura 14.9. (a)

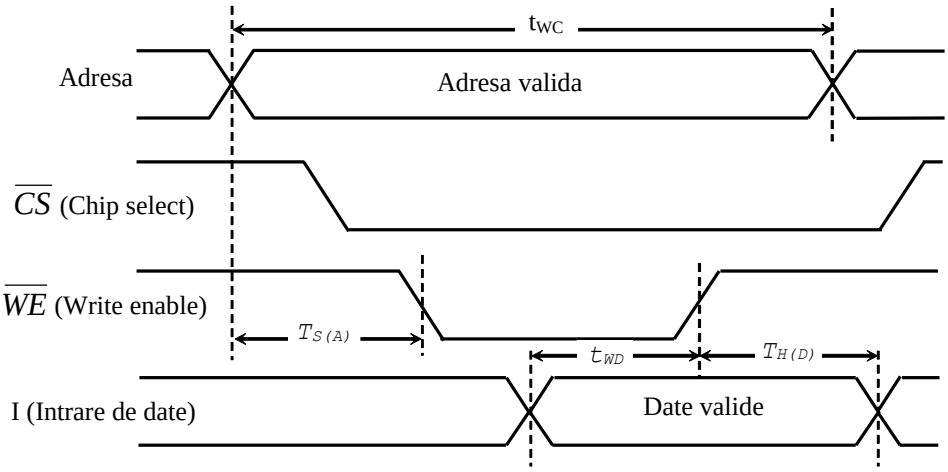


Figura 14.9. (b)

În timpul fiecărui ciclu de citire, o unitate de date, un octet în acest caz, este citită din memorie.

În cazul ciclului de scriere, prezentat în Figura 14.9 (b), un cod de adresă validă este aplicată pe liniile de adrese pe un interval de timp specificat, numit durata ciclului de scriere, t_{WC}. Pe urmă intrările $\overline{CS}$ şi $\overline{WE}$ sunt puse pe LOW. Intervalul de timp, măsurat de la momentul în care adresele sunt valide până la momentul în care intrarea $\overline{WE}$ trecere în starea LOW este numit timpul

de setare a adresei, $t_{S(A)}$. Intervalul de timp în care intrarea $\overline{WE}$ trebuie să fie menţinut în starea LOW este numit durata de scriere. Intervalul de timp în care $\overline{WE}$ trebuie să fie menţinut în starea LOW, după ce date valide sunt aplicate la intrările de date este desemnat ca t_{WD}; intervalul de timp în care data de intrare validă trebuie să rămână pe liniile de date, după ce intrarea $\overline{WE}$ trece în starea HIGH, este timpul de reţinere al datei, $t_{H(D)}$.

În timpul fiecărui ciclu de scriere o unitate de date este înscrisă în memorie.

14.3 Mersul lucrării

Figura 14.10. Stand de testare memorie 32k SRAM W24257

Elementele componente ale standului de testare sunt:

- memoria SRAM de 32k (8 biți)
- buffer 74 LS 244 (8 biți)
- decodificator BCD-zecimal 74 LS 47
- afișaj 7 segmente
- stabilizator de tensiune 78 LM 05 (input 7-19VDC, output 5 VDC)

Semnale:

VCC – alimentare stand 7-19V (DC)
W – write
R – read
CE – chip enable
HEX SWITCH – reprezintă două comutatoare, care dau informație pe 4 biți (ex: 0101, 0-conexiune la GND, 1-conexiune la +5V)
"Z" – înaltă impedanță
A0 – bitul 0 din informația ce se scrie în memorie
A1 – bitul 1 din informația ce se scrie în memorie
A2 – bitul 2 din informația ce se scrie în memorie
A3 – bitul 3 din informația ce se scrie în memorie

Modalitate de lucru :

Asupra memoriei putem executa două operații: - scriere
- citire

Scriere în memorie:

Pentru a putea scrie infomație în memoria care o avem pe standul de testare trebuie realizate următoarele operații:

1. Trebuie să activăm memoria prin conectarea la masă a semnalului CE (chip enable). Comutatorul CE trebuie mutat în poziția "0".
2. Din HEX-switch-uri se selectează adresa de memorie unde va avea loc scrierea. Prin HEX-switch-urile se realizează combinația binară de selectare a unei adrese prin punere la "0" sau la "1" a liniilor de selecție.
3. Comutatorul "HIGH Z" trebuie pus în poziția "0" pentru a activa ieșirile bufferului 74LS244. Comanda intrărilor acestui buffer se realizează prin modificarea stării comutatoarelor A0,A1,A2,A3 (se pun în poziția "0" sau "1", în funcție de valoarea dorită pentru bitul respectiv). Ieșirile bufferului sunt conectate la intrarea unui decodificator BCD-7 segmente74LS47 și la liniile I/O ale memoriei care se testează. Ieșirile decodificatorului BCD-7 segmente sunt conectate la un afișaj cu șapte segmente și în acest fel se poate vizualiza informția care se scrie în memorie.
4. Se apasă butonul W (write) apoi se lasă liber. La apăsarea butonului W, liniile I/O ale memoriei sunt definite ca I (INPUT).
5. Gata. Informația care o aveam mai devreme pe afișajul cu 7 segmente este acum înscrisă la adresa dată de HEX-switch-uri.

Citire din memorie:

Pentru a putea citi infomația din memoria care o avem pe standul de testare trebuie realizate următoarele operații:

1. Trebuie să activăm memoria prin conectarea la masă a semnalului CE (chip enable). Comutatorul CE trebuie mutat în poziția "0".
2. Din HEX-switch-uri se selectează adresa de memorie de unde urmează să citim informația.
3. Comutatorul "HIGH Z" trebuie pus în poziția "1" pentru a dezactiva ieșirile bufferului 74LS244. În acest moment dorim "trecerea" prin decodoficator a informției care urmează să fie citită din memorie.
4. Se apasă butonul R (read). La apăsarea butonului R, liniile I/O ale memoriei sunt definite ca O (OUTPUT).
5. Gata. În acest moment informația care o avem înscrisă la adresa selectată este disponibilă pe liniile I/O ale memoriei (în acest caz OUTPUT) și poate fi vizualizată pe afișajul cu 7 segmente.

Pentru a sterge memoria trebuie să deconectăm sursa de alimentare apoi să comutăm CS pe poziția "1".

1. Arhitectura unui microcontroler

Modelul intern de bază al microcontrolerelor în general este asemănător iar Figura A1 ilustrează diagrama bloc a unui microcontroler obișnuit. Toate componentele sunt interconectate printr-o magistrală internă și sunt integrate pe același cip (circuit integrat). Magistrala transferă semnale de adresă, de date și de control. Una din caracteristicile cele mai importante ale unui microcontroler este dimensiunea acestor magistrale. Prin magistrala de adrese unitatea centrală de prelucrare (UCP) selectează o locaţie de memorie sau un dispozitiv I/O, iar pe magistrala de date se face schimbul de informaţie între UCP şi memorie sau dispozitivele I/O. Modulele sunt conectate cu exteriorul prin pinii de intrare/ieșire (I/O).

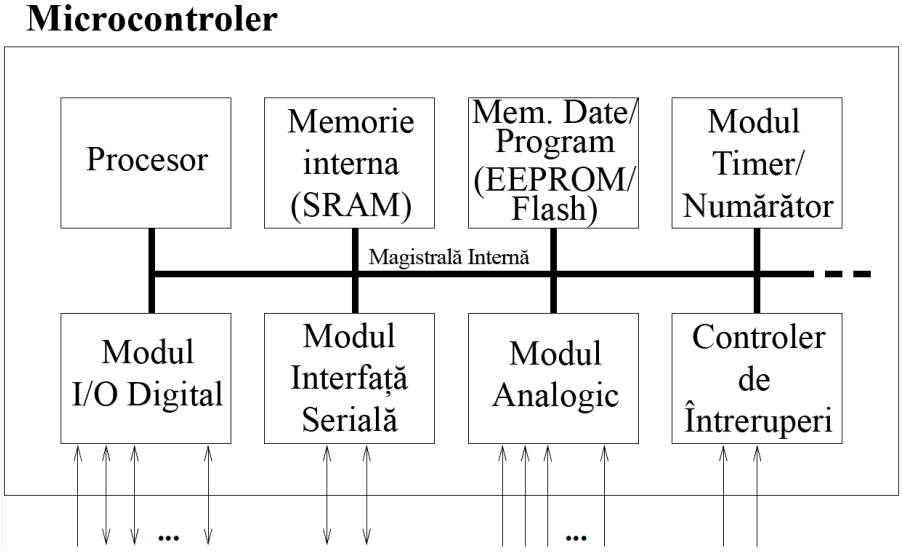


Figura A1. Structura de bază a unui microcontroler

Lista următoare conține modulele tipice care se pot găsi intr-un microcontroler. În secțiunile următoare acestea vor fi detaliate.

Procesorul: Unitatea Centrală de Procesare (UCP) a microcontrolerului. Conține Unitatea Aritmetică și Logică (UAL), Unitatea de Control și un set de regiștrii (registrul indicator de stivă, registru numărător de program, registrul acumulator, etc.)

Memoria: Uneori, memoria este împărțită în memorie de program și memorie de date. În microcontrolerele mai mari există un controler DMA (Acces Direct la Memorie) care conduce transferurile de date dintre componentele periferice și memorie.

Controlerul de întreruperi: Intreruperile sunt folositoare pentru întreruperea cursului normal al programului în cazul apariției unor evenimente interne sau externe (importante).

Timer/Numărător: Majoritatea microcontrolerelor au cel puțin unul, dar uzual 2-3 Timere/Numărătoare, care sunt folosite pentru măsurare intervalelor, numărarea evenimentelor si atașarea unor marcaje de timp pentru evenimente. Majoritatea controlerelor oferă ieșiri PWM (pulse width modulation), care pot fi folosite pentru comanda motoarelor sau pentru sistemele ABS (antilock brake system) de frânare în siguranță. Același semnal de ieșire PWM împreună cu un filtru trece-jos (FTJ) extern, poate compune un convertor digital/analogic ieftin.

I/O Digitale: Porturile paralele de I/O digitale constituie una din caracteristicile esențiale ale microcontrolerelor. Numărul pinilor I/O variază de la 3-4 pana la peste 90, în funcție de familia și tipul controlerului.

I/O Analogice: Majoritatea microcontrolerelor au integrate convertoare analog/digitale care diferă prin numărul de canale (2-16) și rezoluție (8-16 biți). Modulele analogice se bazează în general pe comparatoare analogice. Există dar sunt puține microcontrolerele care conțin convertoare digital/analogice.

Interfețele: În general controlerele au cel puțin o interfață serială prin care se poate face încărcarea programului și comunicarea cu computerul. Interfețele seriale pot fi folosite de asemenea pentru comunicarea cu componentele periferice externe astfel că majoritatea controlerelor oferă diverse interfețe precum SPI și SCI. Multe microcontrolere conțin și controlere pentru I^2C și CAN iar altele mai complexe conțin interfețe pentru PCI, USB și Ethernet.

Microcontrolerul este un procesor care este echipat cu memorie, pini I/O și alte periferice. Cel mai important element care este luat în considerare în acest context este costul. Integrarea tuturor elementelor componente pe același cip duce la economie de spațiu, timp de producție și de asemenea scurtează timpul de proiectare și producție a unui prototip. Timpul și banii sunt elemente esențiale pentru sistemele înglobate (embadded). Alte avantaje ale integrării sunt scurtarea timpului de actualizare si modernizare al sistemului, scăderea consumului de energie și creșterea fiabilității, care sunt de asemenea aspecte importante pentru sistemele înglobate.

Pentru rezolvarea unei sarcini care ar putea fi rezolvată folosind o soluție hardware sau o soluție, aceasta nu va ajunge la aceeași viteză de reacție ca și implementarea hardware. Astfel acolo unde este nevoie de timp de reacție foarte rapid este bine să se aplice soluția hardware. Totuși, majoritatea aplicațiilor în care intervine și interacțiunea cu factorul uman, nu este nevoie de un timp de răspuns atât de rapid. Pentru aceste cazuri, microcontrolerele sunt o alegere bună.

1.1 Terminologia

Înainte de a continua cu aprofundarea microcontrolerelor, se vor defini anumiți termeni care sunt des întâlniți în domeniul sistemelor înglobate (*embadded systems*).

Microprocesor: Acesta constituie defapt o Unitate Centrală de Procesare (UCP) așa cum se poate întâlni într-un PC. Comunicarea cu mediul extern se face prin folosirea magistralei de date, circuitul integrat fiind prevăzut cu pini de date, de adrese și de control. Toate dispozitivele periferice (memorie, timer, USB controller) sunt conectate la această magistrală. Astfel că un microprocesor nu poate fi folosit de sine stătător și are nevoie, la nivel minimalist, de o memorie și un dispozitiv de ieșire pentru a putea fi util.

Microcontroler: Un microcontroler are integrate toate componentele necesare pentru a funcționa de sine stătător și a fost proiectat în mod special pentru aplicațiile de monitorizare și control. Spre deosebire de microprocessor conține memorie, controlere pentru diferite interfețe, unul sau mai multe timere, un controler de întreruperi și de asemenea pini I/O care permit interacțiunea direct cu mediul extern. Microcontrolerele

execută operații pe biți care permit schimbarea unui bit dintr-un byte fară a modifica ceilalți biți.

Sistem Înglobat (Embadded System): Un domeniu de aplicații pentru microcontrolere il constituie *Sistemele Înglobate* în cadrul cărora unitatea de control este integrată în sitemul ca ansamblu. Un exemplu este telefonul mobil unde controlerul este inclus în dispozitiv. De asemenea, un PC folosit pentru pentru controlul unei linii de asamblare din cadrul unei fabrici, împlinește cerințele multor definiții ale Sistemelor Înglobate. Toate definițiile prevăd ca un computer/controler să fie folosit pentru un scop special într-o aplicație și nu pentru sarcini de uz general.

Sistem de Timp Real: Controlerele sunt folosite des în *sisteme de timp real* în care reacția la un eveniment trebuie să se încadreze într-un timp maxim specificat. Acesta este cazul multor aplicații din domeniul aerospațial, automotive și al căilor ferate.

Procesor Înglobat: Acest termen apare de obicei în contextul sistemelor înglobate iar diferențele față de un controller sunt foarte vagi. În general, termenul *procesor înglobat* se folosește pentru dispozitivele high-end (pe 32 biți), în timp ce *controler* se folosește în dreptul dispozitivelor low-end (pe 4, 8, 16 biți).

Procesor de Semnal Digital (DSP): Sunt folosite în aplicațiile de procesare de semnal, un domeniu important constituindu-l telecomunicațiile. Cel mai probabil ca un telefon mobil să conțină un DSP. Aceste procesoare sunt proiectate pentru a executa operațiile de adunare și înmulțire foarte rapid, acestea fiind principalele operații în procesarea de semnal. Pentru ca aplicațiile de procesare de semnal includ de asemenea și funcții de control, mulți producători oferă soluții hibride care combină un microcontroler cu un DSP.

1.2 Procesorul

Unitatea centrală de prelucrare (UCP) este compusă din unitatea aritmetică și logică (UAL) și din unitatea de control și este componenta principală a oricărui microcontroler. Arhitectura unei UCP este ilustrată în Figura A2 și este compusă din *calea de date* care execută instrucțiunile și *unitatea de control* care trimite comenzi caii de date.

Unitatea Aritmetica și Logică

Unitatea centrală de prelucrare este compusă din unitatea aritmetică și logică (UAL) și din unitatea de control.***U****nitatea aritmetică și logică* care este responsabilă pentru a efectuarea operațiilor aritmetice și logice (AND, ADD, MUL, INC,...). Anumite semnale de comandă selectează care este operația care trebuie sa fie aplicată de către UAL asupra datelor de intrare. UAL preia două intrări și returnează rezultatul operației ca și ieșire. Datele de intrare și de ieșire sunt luate din regiștrii sau memorie. Timpul de execuție al fiecărei operații este foarte important pentru a aprecia dacă timpul necesar procesării complete satisface cerințele de timp ale aplicației. De asemenea, UAL stochează anumite informații cu privire la natura rezultatului în *regiştrii de status*.

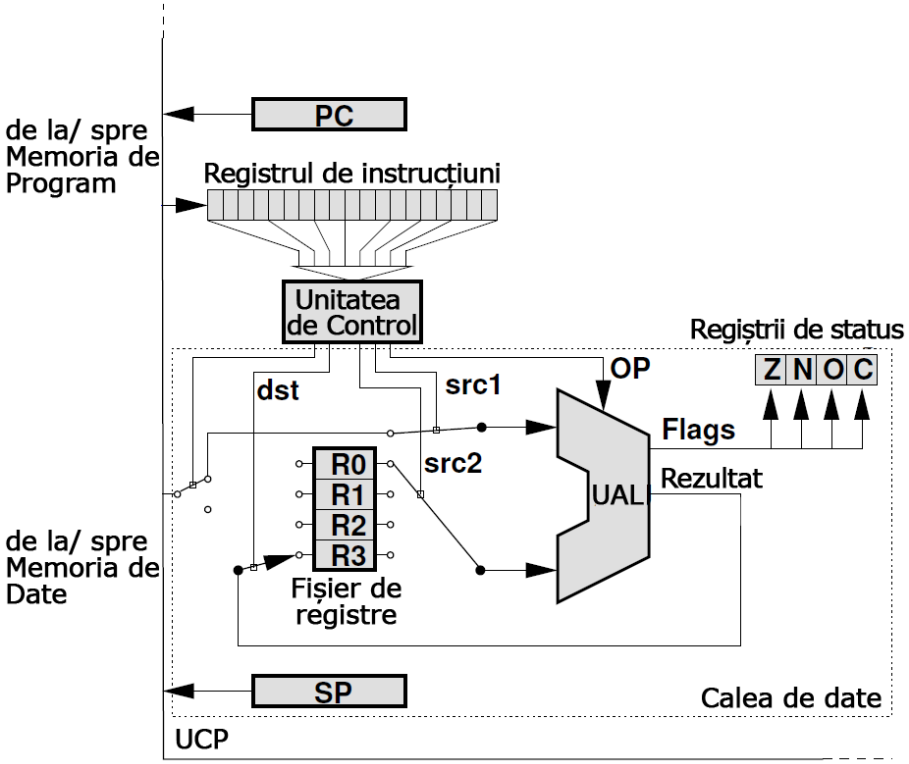


Figura A2. Structura de bază a UCP

Z (Zero): Rezultatul operației este zero.

N (Negativ): Rezultatul operației este negativ, de aceea, *cel mai semnificativ bit* (msb – most significant bit) al rezultatului este 1.

O (Overflow): Operația produce o depășire a capacității de reprezentare.

C (Carry): Indicator de transport. Devine 1 logic dacă operația anterioară a produs un transport sau împrumut la/de la rangul n-1.

Unitatea ce control este responsabilă cu decodificarea codului operației conținut de codul unei instrucțiuni. Pe baza decodificării unitatea de control elaborează semnale pentru comanda celorlalte blocuri funcționale pentru a finaliza executarea unei instrucțiuni. Modul de implementare al acestui bloc este de asemenea transparent utilizatorului.

Fișierul de Regiștrii

Acesta conține regiștrii de lucru ai UCP. Poate sa fie constituit dintru-n set de *regiștrii de uz general* (in general 16-32, dar pot sa fie și mai mulți) fiecare din ei putând fi sursa sau destinația unei operații, sau poate sa fie constituit din regiștrii dedicați. UCP poate să obțină operanzii pentru UAL din fișierul de regiștrii și poate sa stocheze rezultatul înapoi în fișier. În același mod operanzii și rezultatul pot fi stocate în memorie dar e bine de reținut că accesul la memorie este mult mai lent decât accesul la fișierul de regiștrii. De aceea e bine să se folosească fișierul de regiștrii dacă este posibil.

Indicatorul de stiva (Stack Pointer - SP)

Stiva este o porțiune consecutivă din memoria de date care este folosită de UCP pentru a stoca adresele de întoarcere (return) din program și posibil și conținutul regiștrilor în timpul execuției subrutinelor și ale rutinelor de întrerupere. Este accesată prin comenzile PUSH (pune ceva pe stivă) și POP (extrage ceva din stivă). Pentru a ști locația curentă în stivă, UCP conține

un registru special numit *indicator de stivă* (SP), care indică spre vârful stivei. Stiva crește de obicei invers, adică, de la adresa cea mai mare a memorie spre adresa cea mai mica. Astfel că, SP începe prin indicarea ultimei adrese de memorie și este decrementată la fiecare comandă PUSH și incrementată la POP. Motivul pentru care SP este așezat la sfârșitul zonei de memorie este pentru că variabilele sunt stocate în general la începutul zonei de memorie, astfel că, așezând stiva la sfârșitul ei va lua mai mult timp pentru ca acestea să se întâlnească.

Unitatea de control (UCP)

UCP execută continuu instrucțiunile programului cu excepția cazului în care se execută un RESET**(**iniţializarea MC**)**sau o instrucțiune de pauză. *Unitatea de control* are rolul de a determina care este următoarea instrucțiune ce trebuie executată și să configureze calea datelor corespunzător. Pentru aceasta există un alt registru important**,** *registru numărător de program* (Program Counter - PC) care stochează adresa următoarei instrucțiuni de executat. Unitatea de control încarcă această instrucțiune în *registrul de instrucțiuni*, o decodifică și configurează calea de date pentru a o executa. Această configurare include furnizarea intrărilor potrivite pentru UAL (din memorie sau regiștrii), selectarea operației corecte în UAL și scrierea rezultatului în locația corectă. După un RESET, registrul PC se încarcă dintr-o locație de memorie numită vector de reset. Această locație conține adresa primei instrucțiuni de executat. PC este incrementat automat la execuția unei instrucțiuni.

RISC (Reduced Instruction Set Computer): Arhitectura RISC are instrucțiuni simple, implementate la nivel hardware care de obicei au nevoie de unul sau doar câteva cicluri de timp pentru a fi executate. Dispozitivele RISC au o dimensiune mica a codului, relativ puține instrucțiuni și doar câteva moduri de adresare. Ca rezultat, execuția instrucțiunilor este foarte rapidă dar setul de instrucțiuni este oarecum simplu.

CISC (Complex Instruction Set Computer): Arhitectura CISC este caracterizată de instrucțiunile complexe realizate în microcod pentru care e nevoie de mai multe cicluri de timp pentru a fi executate. Arhitectura are de obicei o dimensiune mare și variabilă a codului și oferă multe instrucțiuni și moduri de adresare complexe și puternice. În comparație cu RISC, arhitectura CISC are nevoie de mai mult timp pentru execuția instrucțiunilor, dar setul de instrucțiuni este mult mai puternic.

Bineînțeles că atunci când există două arhitecturi, întrebarea care se ridică este despre care dintre ele e mai bună, însă răspunsul depinde de nevoia pe care o ai. Dacă soluția aleasă prevede folosirea frecventă a unei instrucțiuni sau mod de adresare specific CISC atunci aceasta este arhitectura care trebuie folosită. Dar daca este nevoie mai mult de instrucțiuni și moduri de adresare simple atunci e mai bine să se folosească arhitectura RISC. De asemenea sunt și alți factori de care depinde alegerea, cum ar fi frecvența procesorului. De asemenea trebuie studiate clar care sunt cerințele de la arhitectură pentru a putea face o alegere corectă.

Arhitectura von Neumann și Harvard

În Figura A2, memoria de instrucțiuni și cea de date sunt ilustrate ca doua entități diferite. Dar nu întotdeauna se întâmplă astfel. Există posibilitatea ca acestea sa facă parte din aceeași memorie partajată. De fapt acest aspect face distincție între două tipuri de arhitectură de bază:

Arhitectura von Neumann: În această arhitectură, memoria de program și de date sunt stocate împreună și sunt adresate prin aceeași magistrală. Apare o problemă și anume că accesul la memoria de program și de date poate intra în conflict (rezultând în ceea ce se numește *strangulare von Neumann*), ducând la întârzieri nedorite.

Arhitectura Harvard: Această arhitectură presupune ca memoria de date și de program să fie în zone separate de memorie și să fie accesate prin magistrale diferite. În consecință, accesul la cod nu intră în conflict cu accesul la date ceea ce duce la o îmbunătățire a performanțelor sistemului. Un dezavantaj al acestei arhitecturi este nevoia unui hardware mai complex pentru că cere două magistrale separate și fie două cipuri de memorie separate sau o memorie care permite două accesuri independente în același timp.

1.3 Memoria

Până aici au fost menționate diferite tipuri de memorii: *fișierul de regiștrii* care este doar o memorie de mici dimensiuni integrată în UCP și au fost amintit că datele sunt aduse din *memoria de date* și instrucțiunile din *memoria de instrucțiuni*. Astfel se va face o distincție între tipurile de memorii potrivit cu funcționalitatea lor.

Fișierul de Regiștrii: O memorie de mici dimensiuni integrată în UCP folosită pentru stocarea temporară a datelor cu care lucrează UCP și poate fi numită memoria de scurta durată a UCP.

Memoria de Date: Pentru stocarea datelor pe termen lung, UCP folosește o memorie externă care este de dimensiuni mult mai mari decât fisierul de regiștrii unde datele sunt stocate cât timp UCP funcționează. Atașarea unei memorii externe pentru UCP implică hardware auxiliar și implicit o creștere a costului de aceea microcontrolerele conțin integrată și memoria de date.

Memoria de Instrucțiuni: La fel ca și memoria de date, memoria de instrucțiuni este o memorie externă cu o dimensiune relativ mare. În cazul arhitecturii von Newmann poate sa fie în aceeași locație fizică de memorie cu memoria de date. Și această memorie este în general integrată în cipul de memorie al microcontrolerului.

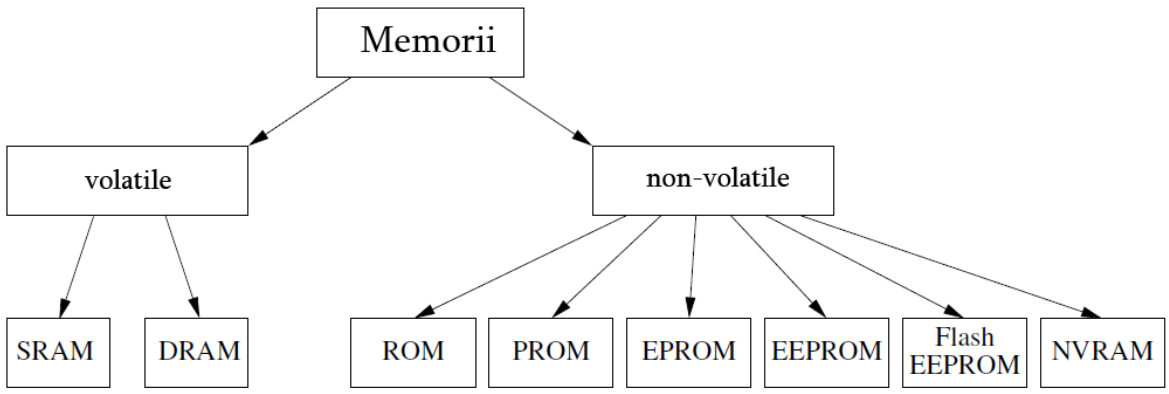


Figura A3. Tipurile de memorii semiconductoare

Mai sus s-a făcut o distincție între tipurile de memorie bazat pe funcționalitatea acestora ceea ce interesează din perspectiva unui programator. Din perspectiva unui inginer hardware

deosebirea între tipurile de memorie se face pe baza proprietăților fizice ale componentelor electrice din care e făcută memoria. Se face astfel diferența între memoriile volatile și cele non-volatile precum în Figura A3. În acest context, termenul volatil se referă la faptul că memoria își pierde conținutul la întreruperea tensiunii de alimentare. Este important de reținut ca memoria de program și cea de date este implementă ca o memorie non-volatilă în timp ce memoria de scurtă durată este o memorie volatilă.

1.3.1 **Memoria Volatilă**

După cum a fost menționat deja, memoria volatilă reține datele atâta timp cât dispozitivul este sub tensiune. Motivele pentru care aceste memorii se folosesc sunt viteza de acces a datelor si costul. Memoriile non-volatile sunt cu mult mai lente și de asemenea mult mai scumpe. În timp ce memoria volatilă a unui PC are un timp de acces de ordinul nanosecundelor, unele tipuri de memorie non-volatile vor fi inaccesibile pentru câteva milisecunde după o operație de scriere.

RAM Statică

Memoria SRAM (*Static Random Access Memory*) a fost primul tip de memorie volatilă folosită și larg răspândită în aplicații. Un cip de memorie SRAM este compus dintr-o matrice de celule, fiecare din ele fiind capabilă să rețină un bit de informație. O astfel de celulă SRAM constă dintr-un bistabil conectat la circuitul intern prin două tranzistoare de acces (Figura A4), bistabil care de obicei este construit cu patru sau șase tranzistoare. Când celula de memorie nu este adresată, cei doi tranzistori de acces sunt blocați și astfel informația conținută este păstrată în interiorul bistabilului. Bistabilul are nevoie, bineînțeles, ca să fie alimentat pentru a reține informația din această cauză SRAM este o memorie volatilă (informația se pierde când alimentarea este întreruptă).

Memoria SRAM este concepută pentru a împlini două nevoi: timp de acces mic și consum redus de energie. Memoria SRAM este mult mai rapidă decât memoria DRAM cu un timp de acces de aproximativ 10 nanosecunde față de 50 – 150 nanosecunde la memoria DRAM. Motivul din spatele acestui fapt este configurația cu șase tranzistoare care păstrează un curent ce curge într-o parte sau în cealaltă (starea 0 sau 1) astfel că fiecare stare poate fi citită sau scrisă instant. În cazul aplicațiilor low-power, memoriile SRAM oferă un avantaj și sunt folosite în majoritatea echipamentelor portabile. Microcontrolerele de obicei au doar memorie SRAM, acestea având un consum cu câteva ordine de mărime mai mic decât în cazul memoriilor DRAM.

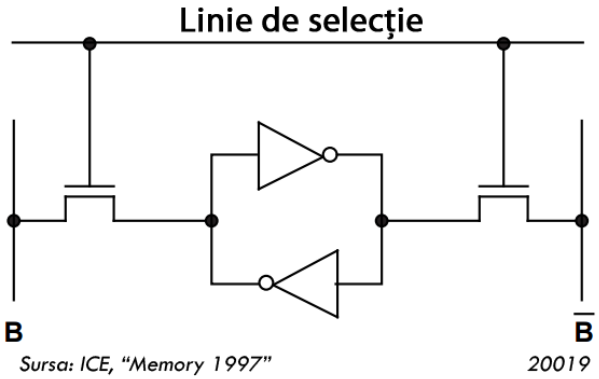


Figura A4. Celula de memorie SRAM

RAM Dinamică

Memoria DRAM (*Dynamic random-access memory*) este un tip de memorie RAM care păstrează fiecare bit de informație într-un condensator separat (Figura A5). Acest condensator poate să fie încărcat sau descărcat; aceste stări reprezintă cele două stări ale unui bit (0 sau 1). Pentru că nu există tranzistori perfect blocați, prin aceștia se va scurge mereu un curent parazit de o valoare foarte mică. Din cauza aceasta, condensatorul se va descărca încet iar informația se va pierde în cele din urmă în cazul în care sarcina condensatorului nu va fi reîmprospătată. Datorită nevoii de reîmprospătare, acest tip de memorie se numește dinamică față de memoria SRAM care este statică. Ciclul de reîmprospătare se repetă la valori de timp cuprinse între 1 – 100 ms.

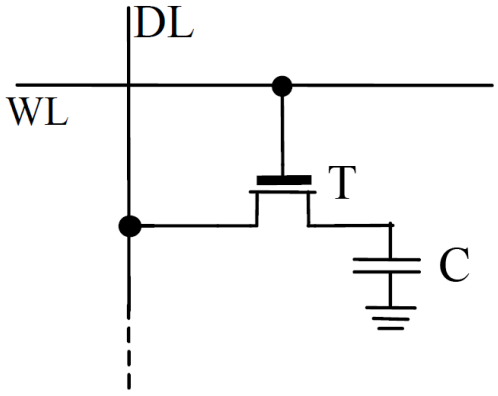


Figura A5. Celula de memorie DRAM

Avantajul memoriei DRAM este constituit de simplitatea structurală: pentru un bit de memorie este nevoie de un singur tranzistor ;i un condensator, comparat cu șase tranzistoare la memoria SRAM. Astfel, memoriile DRAM permit o densitate foarte mare de memorie per unitate de arie. Tranzistorii și condensatoarele folosite sunt extrem de mici astfel că pe un circuit integrat pot încăpea miliarde. DRAM este o memorie volatilă pentru că își pierde informația foarte rapid după întreruperea alimentării.

1.3.2 Memoria Non-volatilă

Spre deosebire de memoria SRAM și DRAM, memoriile non-volatile reține informațiile conținute chiar și după ce a fost întreruptă alimentare. Totuși, în schimbul acestui beneficiu apar și dezavantaje. Scrierea unei astfel de memorii este în general mult mai lentă și mai complicată decât a memoriilor volatile.

ROM (*Read Only Memory*)

Este cea mai ieftină şi cea mai simplă memorie și se folosește la stocarea programelor în faza de fabricație. Unitatea centrală poate citi informațiile, dar nu le poate modifica. Utilizarea memoriei ROM este o opțiune doar pentru cazurile de producție în masă.

PROM (*Programmable Read Only Memory*)

Aceasta este similară cu memoria ROM, dar ea poate fi programată de către utilizator. Acest tip de memorie este reprezentată printr-o matrice de celule de memorie, fiecare dintre ele conținând o siguranță fuzibilă din siliciu. Inițial, fiecare siguranță este intactă astfel că fiecare celulă este încărcată cu valoarea „1” logic. Prin selectarea unei celule și aplicarea unui

puls de curent scurt dar de o valoare mare, siguranța va fi distrusă astfel programându-se celula la „0” logic. Uneori, acest tip de memorie se întâlnește sub numele de ***One Time Programmable*** **(OTP)**. Memoria nu se poate șterge sau reprograma. Prețul unui MC cu OTP este mic, viteza este bună, dar aplicațiile sunt lipsite de flexibilitate.

EPROM (*Erasable PROM*)

Se poate şterge prin expunere la raze ultraviolete. MC cu EPROM au un mic geam de cuarţ care permite ca *chip-ul* să fie expus la radiaţia ultravioletă. Ştergerea este neselectivă, adică se poate şterge doar întreaga informaţie şi nu numai fragmente. Memoria poate fi ştearsă şi reînscrisă de un număr finit de ori. Memoria este constituită din *tranzistoare cu efect de câmp* (FET) care sunt programați individual de un dispozitiv electronic care furnizează tensiuni mai mari decât cele folosite în mod uzual în circuitele digitale, de exemplu 12 V.

EEPROM (*Electrically Erasable PROM)*

Are toate avantajele memoriei EPROM și funcționează aproximativ la fel ca și aceasta cu diferența că electronii din poarta tranzistoatelor FET pot fi eliminați prin aplicarea unei tensiuni ridicate. Astfel memoria se poate șterge electric și selectiv. Aceasta are și dezavantajul unui număr limitat de cicluri citire/scriere (valori între 10.000 și 100.000 cicluri) și faptul că nu reține informația pentru totdeauna. Memoria EEPROM se găsește în majoritatea microcontrolerelor si este folosită ca și memorie de date de lungă durată datorită numărului limitat de citiri/scrieri. Memoriile EEPROM sunt folosite cel mai adesea pentru stocarea parametrilor de calibrare.

Flash-EEPROM

Memoria Flash este un tip de memorie EEPROM la cere ștergerea nu este posibilă pentru fiecare adresă în parte ci doar pentru blocuri mai mari sau chiar pentru întreaga memorie. Din această cauză logica internă este simplificată, reducând astfel prețul. De asemenea, din cauză ca nu se poate șterge un singur byte al memoriei, aceasta se folosește în general ca și memorie de program a microcontrolerului. Numărul limitat de reprogramări nu este o problemă pentru că uzual, microcontrolerul nu se reprogramează de 10.000 de ori.

NVRAM

În final, există un tip de memorie care combină avantajele memoriilor volatile și non-volatile: *Non-Volatile RAM* (NVRAM). Acest lucru se poate face în mai multe moduri, nul din ele fiind adăugarea unei baterii memorii SRAM astfel conținutul se păstrează chiar și după întreruperea alimentării. Un alt mod este de a combina o memorie SRAM cu una EEPROM în același cip. Pe parcursul perioadei de inițializare datele sunt copiate din EEPROM în SRAM. În timpul operării, datele sunt citite din și scrise în SRAM. Când se întrerupe alimentarea, datele sunt copiate în EEPROM.

1.3.3 **Accesarea Memoriei**

Majoritatea microcontrolerelor sunt proiectate să aibă integrată memoria de program și memoria de date. De obicei memoria de program este de tip Flash-EEPROM iar memoria de date este constituită la unele din memorie SRAM iar la altele din EEPROM. După modul de adresare al memoriei (modul cum o adresa se transpune într-o adresă a memoriei) se disting două metode posibile:

1. Fiecare memorie este adresată separat (ATmega16, 32), Figura A6.

Intervalul de adresare poate sa fie același pentru toate trei tipurile de memorie. Programatorul specifică care memorie va fi folosită prin folosirea unor instrucțiuni de acces diferite.

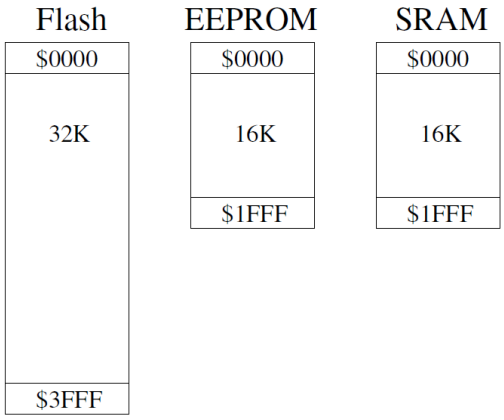


Figura A6. Adresarea separate a memoriei

2. Toate tipurile de memorie împart același interval de adresare comun, Figura A7.

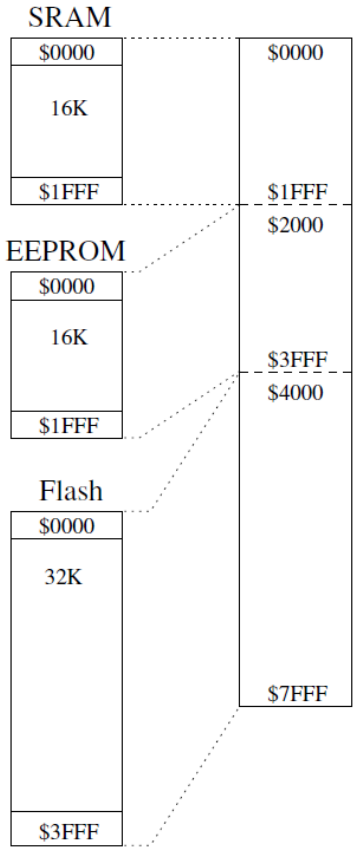


Figure A7. Maparea în același interval de adresare a mai multor tipuri de memorii

În acest caz, programatorul va accesa memoria EEPROM în același fel ca și memoria SRAM. Microcontrolerul folosește adresa pentru a decide ce memorie trebuie accesată. Spre

exemplu, memoriei EEPROM i se va putea aloca intervalul de adrese 0x1000 – 0x2000, în timp ce memoriei SRAM i se va aloca intervalul 0x2000 – 0x3000. În cazul în care se accesează adresa de memorie 0x1800, microcontrolerul va ști că este în intervalul EEPROM-ului. Chiar daca această metodă este foarte simplă și directă, totuși există posibilitatea de a greși: o adresă greșită va duce la accesare tipului de memorie nedorit. Adresarea separată, pe de altă parte, oferă o protecție implicită împotriva accesării tipului greșit de memorie.